HAL open science

The Geometry of Causality

Simon Castellan, Pierre Clairambault

To cite this version:

Simon Castellan, Pierre Clairambault. The Geometry of Causality: Multi-Token Geometry of Interaction and its Causal Unfolding. Proceedings of the ACM on Programming Languages, In press, pp.1-70. hal-03286443v2

HAL Id: hal-03286443

https://hal.science/hal-03286443v2

Submitted on 14 Nov 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

The Geometry of Causality

Multi-Token Geometry of Interaction and its Causal Unfolding

SIMON CASTELLAN, Univ Rennes, Inria, CNRS, IRISA, France
PIERRE CLAIRAMBAULT, Aix Marseille Univ, Université de Toulon, CNRS, LIS, Marseille, France

We introduce a multi-token machine for *Idealized Parallel Algol (*IPA*)*, a higher-order concurrent programming language with shared state and semaphores. Our machine takes the shape of a compositional interpretation of terms as *Petri structures*, certain coloured Petri nets. For the purely functional fragment of IPA, our machine is conceptually close to *Geometry of Interaction* token machines, originating from Linear Logic and presenting higher-order computation as the low-level process of a token walking through a graph (a proof net) representing the term. We combine here these ideas with folklore ideas on the representation of first-order imperative concurrent programs as coloured Petri nets.

To prove our machine computationally adequate with respect to the reference operational semantics, we follow game semantics and represent types as certain games specifying dependencies and conflict between computational events. *Petri strategies* are those Petri structures obeying the rules of the game extracted from the type. We show how Petri strategies *unfold* to concurrent strategies in the sense of concurrent games on event structures. This link with concurrent strategies not only allows us to prove adequacy of our machine, but also lets us generate operationally a causal description of the behaviour of programs at higher-order types, which is shown to coincide with that given denotationally by the interpretation in concurrent games.

Additional Key Words and Phrases: Geometry of Interaction, Game Semantics, Shared Memory Concurrency, Coloured Petri Nets, Higher-Order

1 INTRODUCTION

1.1 The Operational, the Denotational and the Game Semantics

Semantics of programming languages are often classified into two broad families. On the one hand, *operational semantics* formalize execution via concrete, local rules that often operate directly on syntax: this includes methods based on rewriting, labelled transition systems or abstract machines. Mathematically those are usually relatively simple inductive structures making them easy to implement and reason about. However they often focus on *closed* programs, and most formalisms struggle with compositionality. On the other hand, *denotational semantics* embed types and programs in an adequate mathematical space (domains and continuous functions, sets and relations, games and strategies, etc). Compositionality holds by definition, as the denotation is computed by induction on syntax, sending each syntactic construction to a corresponding semantic operation – this makes denotational semantics a great tool for modular reasoning on programs and to inform programming language design. However, the link with the source code is often quite remote: identifying the piece of syntax and the runtime environment responsible for a denotationally observed behaviour can be subtle, in particular because interpretation can involve complex constructions (such as composition of strategies). Denotational semantics abound with *adequacy* results relating evaluation in the sense of operational and denotational semantics, but those typically concern closed programs of

Authors' addresses: Simon Castellan, Univ Rennes, Inria, CNRS, IRISA, Rennes, France, simon.castellan@inria.fr; Pierre Clairambault, Aix Marseille Univ, Université de Toulon, CNRS, LIS, Marseille, France, pierre.clairambault@cnrs.fr.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
© 2023 Copyright held by the owner/author(s).
2475-1421/2023/1-ART24
https://doi.org/10.1145/3571217

ground type, and are not very informative for open programs. In this paper we ask ourselves: how transparent can we make the link between operational and denotational semantics, for *open* programs of *arbitrary type*? Such a link, if established convincingly, could play a valuable role in the mathematical theory of programming semantics.

To approach this question, we allow ourselves some flexibility on what we exactly mean by "operational semantics": we do not necessarily refer here to the familiar range of syntactic techniques, but more broadly to, let us say, a formalization of execution specified by concrete, local rules acting on an intermediate representation of programs obtained from the source code in a modular way, with low complexity (say, linear). The goal is that one should then be able to easily extract denotational information from execution traces in this operational semantics. Because denotational semantics predict program evaluation in any context, this implies that our executions cannot stop at computing the result of closed programs of ground types; they must, for open programs of arbitrary type, produce (lazily) a structure that is fully informative of their dynamic behaviour in any context, in a way that is both operationally meaningful and close to denotational semantics. The field that studies such interactive representations of programs is called *game semantics* [Abramsky et al. 2000; Hyland and Ong 2000]: it presents the interactive behaviour of programs as a *strategy* in a two-player game that the program plays with its execution environment. More specifically, we target *concurrent games* [Castellan and Clairambault 2020; Castellan et al. 2017]: based on *event structures*, those offer a representation of the interactive behaviour of programs that is causal and with full non-deterministic branching information. Thanks to this intensional expressivity, they are related by forgetful projections to a growing number of models in the denotational semantics landscape – this includes various other game models [Castellan and Clairambault 2020], relational or weighted relational models [Castellan et al. 2018; Clairambault and de Visme 2020], and the Scott model through Ehrhard's extensional collapse theorem [Ehrhard 2012].

Operational game semantics. Connecting operational and game semantics is far from a new idea. Though it was clear from early work on game semantics [Danos et al. 1996] that interaction between strategies was related to execution by abstract machines, the idea to generate strategies by operational means was – to our knowledge – first suggested by Laird in his trace semantics for higher-order references [Laird 2007] (with the explicit connection with game semantics later worked out by Jaber [Jaber 2015]). In the 2010s, several works were proposed blurring the lines between operational and game semantics [Ghica and Tzevelekos 2012; Levy and Staton 2014], typically via LTSs dealing with open programs by sending and receiving messages from the environment (interestingly, similar structures appear in work on compositional certified compilers [Stewart et al. 2015]); for these systems, compositionality is a theorem rather than a definition. But these developments do not yield strategies in the sense of previously established models. Furthermore they only deal with sequential deterministic programs, and it seems hard to extend LTS-based techniques to give a causal account of the execution of higher-order concurrent programs.

But in fact, a powerful connection between operational and denotational semantics was already established significantly before the above, although not then presented as such: it is Baillot's result [Baillot 1999] that Girard's *Geometry of Interaction* for IMELL generates the strategy obtained as its interpretation in so-called Abramsky-Jagadeesan-Malacaria (AJM) games [Abramsky et al. 2000]. As this result is one of our main inspirations, we take some time to introduce its main intuitions.

1.2 The GoI Token Machine and AJM Games

Girard's *Geometry of Interaction (GoI)* was first introduced as a model construction for System F [Girard 1989]; but it was soon realized, following work by Danos, Herbelin and Regnier [Danos et al. 1996; Danos and Regnier 1996] and Mackie [Mackie 1995] that it informed an abstract machine

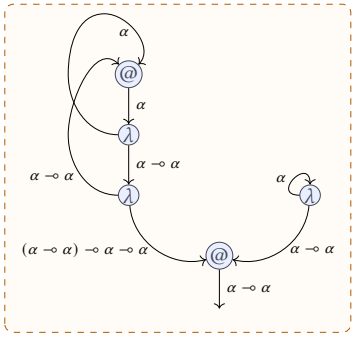


Fig. 1. The λ-graph representation of a term

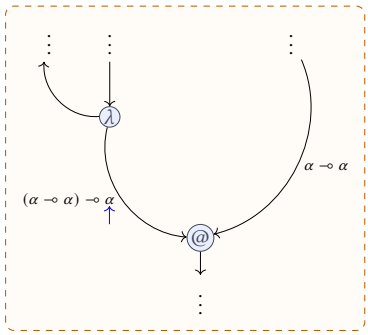


Fig. 2. A token on a wire

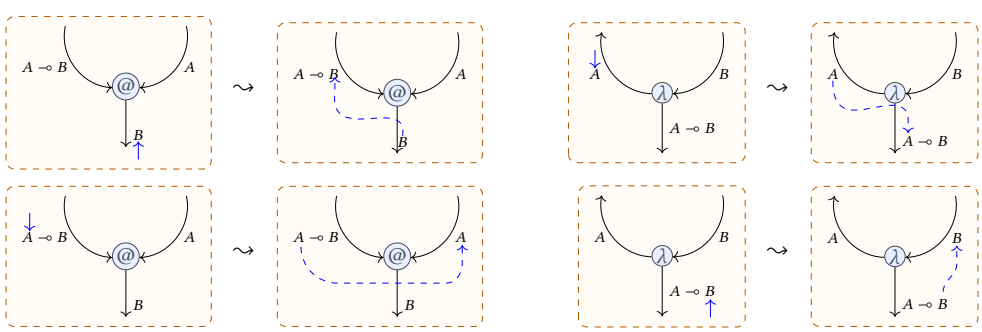
that could be used to evaluate programs, with links with Lamping's optimal reduction [Gonthier et al. 1992]. We introduce this informally, on a simply-typed linear λ-calculus with one atom α.

In order to execute the GoI token machine on a term $\vdash M : A$, one usually first translates it into a *proof net* in the sense of linear logic – or alternatively, to avoid relying too much on familiarity with linear logic, into a simple λ-*graph*: for instance, the linear λ-term

$$\vdash (\lambda f^{\alpha \multimap \alpha}.\, \lambda x^{\alpha}.\, f\, x)\, (\lambda y^{\alpha}.\, y) : \alpha \multimap \alpha \tag{1}$$

may be represented by the graph shown in Figure 1. The reader should recognize in this directed graph the structure of the term: there is one node for each application, and one node for each abstraction. An application node receives two inputs (the function and its argument, drawn on top) and emits one output (the result, drawn at the bottom). An abstraction node receives one input (the "body" of the function, drawn on top) and emits two outputs: a wire for the bound variable, drawn on the side, and a wire for the resulting function, drawn at the bottom. All wires are explicitely labelled with a type. Finally the dangling wires match the overall typing judgment; it is the interface with which the program will interact with its environment. One may define a rewriting theory for such representations (though then this is more elegantly done on linear logic proof nets).

The Geometry of Interaction token machine captures execution as a token walking through this graph, following local rules. At any point in time, the token sits on one of the wires – more precisely, on an occurrence of an atom in the type labelling the wire. The token also has a *direction*: it may follow the orientation of the wire, or go against it – we represent this as an arrow accompanying the atom, as in Figure 2. How is execution formalized? The GoI token machine is initialized by putting a token, going up, on the rightmost atom occurrence on the output dangling wire. This token is then pushed around the graph following simple local rules, of which we show a few:



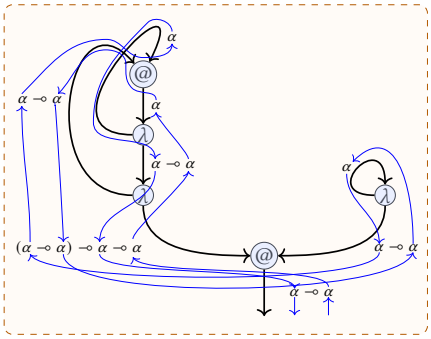


Fig. 3. An execution of the GoI token machine

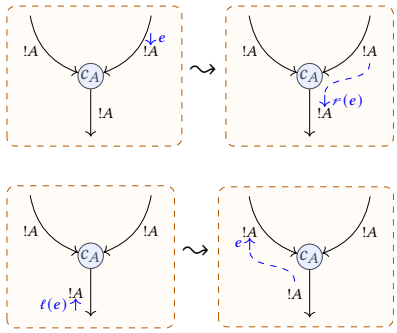
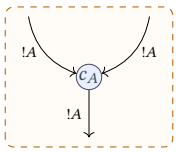


Fig. 4. Token transitions for contraction

Here, we use A, B to range over arbitrary types while α is reserved to occurrences of the atom. In the transitions shown above, tokens are shown accompanying a type A or B, but it is meant that they sit on a *particular atom occurrence* within A or B, as *e.g.* in Figure 2. Each transition moves the token to another copy of the *same* type, so it makes sense to require that it should sit on the *same* atom occurrence as before. The reader should be able to complete the missing rules: in each node, the "As" and the "Bs" are associated pairwise, and the rules simply follow this correspondence. Following these rules, the GoI token machine on our example λ-graph yields the execution depicted in Figure 3: a token thrown on the output atom α eventually makes its way back to the dangling wire, on the input atom α. This input/output behaviour is the same as observed when running the GoI token machine on (the λ-graph for) $\lambda x.\, x : \alpha \multimap \alpha$, reflecting that the λ-term in (1) indeed evaluates to $\lambda x.\, x : \alpha \multimap \alpha$ – it is possible to *read back* the normal form from executions of the GoI.

Of course, the simplicity of this presentation of the GoI token machine exploits the linearity of the language at hand. The non-linear version heavily rests on Girard's decomposition $A \to B = !A \multimap B$ of the call-by-name function type as a combination of the linear function type and the "bang" (!) modality, marking replicable resources. Proof nets for non-linear languages then involve explicit "contraction" nodes allowing us to link one abstracted variable to several variable occurrences:



This heavily impacts the token game: tokens must carry data identifying resource accesses, and helping route tokens around the net. These "*exponential signatures*" may be formalized as natural numbers or via a simple inductive definition $\mathcal{E} ::= \ell\mathcal{E} \mid r\mathcal{E} \mid \ldots$ (with elements ranged over by e and its variants). The token game exploits this, via rules including those in Figure 4. We omit the details, which we will not rely on in this paper, the full machine appears *e.g.* in [Baillot 1999].

Altogether, one "runs" a λ-term by: *(1)* translating it to a proof net; *(2)* initializing the machine by putting a token going up on a dangling wire, on a variable occurrence m with exponential signature e – hence the input is a pair (m, e); *(3)* applying the rules as long as possible. The machine may in principle get stuck or loop forever, we say it *halts* if it emits a token carrying e' going down on an atom occurrence m', corresponding to a pair $(\mathsf{m}', \mathsf{e}')$. Such pairs, called *moves*, form a set written Moves. Then the GoI token machine on $\Gamma \vdash M : A$ initialized and executed as above yields

$$f_M : \mathsf{Moves} \rightharpoonup \mathsf{Moves}\,,$$

a partial function. Now it turns out that this f_M directly informs the *strategy* $[\![M]\!]$ interpreting M in AJM game semantics. More precisely, and as Baillot proved, the interpretation of types may be set up so that the moves of the game $[\![\Gamma \vdash A]\!]$ indeed form a subset of Moves, and that we have:

Theorem 1.1 (Baillot, 1999). *Consider $\Gamma \vdash M : A$ any term.*
Then, the strategy $[\![M]\!] : [\![\Gamma \vdash A]\!]$, obtained denotationally from M, comprises exactly the sequences

$$s = s_1 s_2 \ldots s_{2n-1} s_{2n}$$

which are valid plays on $[\![\Gamma \vdash A]\!]$ and such that for all $1 \leq i \leq n$, we have $s_{2i} = f_M(s_{2i-1})$.

Thus the GoI token machine *generates operationally* the strategy $[\![M]\!]$ in AJM game semantics, by computing the "next move" by Player (the program under scrutiny) as a (partial) function of the previous move by Opponent (the execution environment). This simple generation mechanism exploits that the strategies interpreting λ-terms in AJM games are *history-free*, and only depend on the previous move and not the rest of the play (in AJM jargon, f_M is the "history-free skeleton").

Baillot proves this for IMELL, Intuitionistic Multiplicative Exponential Linear Logic, which includes the simply-typed λ-calculus. In this restricted case, this is an elegant solution to our problem: the strategy interpreting a term in a well-established, mainstream game semantics, normally obtained from the term in a denotational, compositional way, is alternatively computed operationally by executing M in the so-called "Interaction Abstract Machine" [Danos et al. 1996]. This is a great inspiration to us, but in terms of expressivity as a programming language IMELL is somewhat trivial – essentially the simply-typed λ-calculus. Baillot's technical toolbox does not extend beyond IMELL: it is unclear what is the right extension of proof nets, the token machine is too sequential and allows no memory, and even the notion of "generation of a strategy" is inadequate as the target strategies in game semantics are no longer history-free – we need new tools.

1.3 Multi-Token GoI via Petri Nets

In this paper we aim for a result analogous to Theorem 1.1 on a far end of the expressivity spectrum: a language we call *Idealized Parallel Algol* (IPA, see Section 3.1) with a combination of complex programming language features: higher-order, recursion, concurrency, local state and semaphores.

If we are to follow Baillot's lead, we need a notion of proof net for IPA, a GoI token machine, a game semantics, and an appropriate notion of operational "generation" of a strategy. Concerning the game semantics, IPA has the advantage of being well-studied: it is a cosmetic variation of the language ICA for which Ghica and Murawski have provided an interleaving fully abstract game model [Ghica and Murawski 2008] based on a non-alternating version of Hyland-Ong games [Hyland and Ong 2000]. More recently, Castellan and Clairambault [Castellan and Clairambault 2020] have refined it into a *causal* fully abstract game model based on concurrent games on event structures [Castellan et al. 2017]. The concurrent game model projects onto Ghica and Murawski's [Castellan and Clairambault 2020] but is more fine-grained: it carries all information about non-deterministic branching and causal dependence and independence between computational events, in the style of *true concurrency* models. It also deals with replication in a way that is analogous with AJM games, using *copy indices* that reflect the *exponential signatures* of the token machine.

On the GoI side, token machines were extended in multiple ways: they were redeveloped in a coalgebraic setting supporting a range of algebraic effects (*e.g.* nondeterminism, probability, exceptions, global state, interactive I/O, etc.) [Hoshino et al. 2014; Muroya et al. 2016]; and for quantitative effects up to quantum primitives [Dal Lago et al. 2017; Hasuo and Hoshino 2017]. They were extended to *multi-token* machines, for Linear Logic [Laurent 2001] or for functional programs [Dal Lago et al. 2014, 2015] (including in call-by-value) or interaction nets [Dal Lago et al. 2014]; up to multiport interaction nets [Dal Lago et al. 2017] which encompasses concurrent

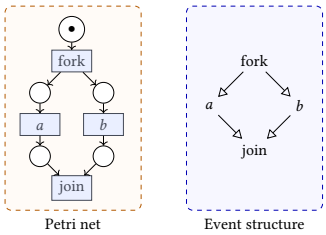


Fig. 5. Parallelism

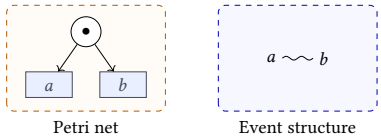
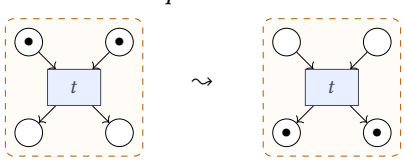


Fig. 6. Non-determinism

behaviour through the translation of the π-calculus [Mazza 2005]. In principle, one could obtain a multi-token machine for IPA by translating terms to the π-calculus and then multiport interaction nets. But generalizing Theorem 1.1 would require all translations to preserve the causal semantics, and the state of the art leaves us unprepared for that. Instead we design a multi-token machine operating on a structure directly obtained from the IPA source code. It is not so surprising that such a machine can be designed, but its correctness is a challenge: GoI correctness proofs usually rest on proof-theoretic techniques, *e.g.* exploiting the cut elimination result of proof nets or via realizability models, none of which are available for higher-order shared memory concurrency. To solve that we adopt a novel methodology: the adequacy of our multi-token GoI will leverage recent progress in concurrent game semantics of programming languages, via our unfolding theorem.

So how shall we obtain a GoI token machine for IPA? We cannot follow the traditional route exposed before, as it is unclear what would be a fitting notion of proof net. Instead, our idea is to skip proof nets altogether, and directly translate programs into graphs *already equipped with the token game*. Ignoring at first the exponential signatures, it should be clear that the proof nets of Section 1.2 equipped with the token game are finite state machines: the states are simply the possible positions of the token on the proof net. As IPA is a concurrent language, its token machine must have multiple tokens: this invites a translation of programs into *Petri nets*.

Petri Nets. Petri nets are multi-token finite automata; they are one of the most well-established "truly concurrent" models in concurrency theory; their use is widespread in computer science and beyond. Roughly speaking, a Petri net is a bipartite directed graph, whose nodes are either *locations* (drawn as circles), or *transitions* (drawn as rectangles). Locations may contain one or several *tokens*, denoted as black dots for now. Locations with edges to a transition t are called the *preconditions* of t, while locations with an edge from t are its *postconditions*. The *token game* consists in *firings*



which can occur if all pre-conditions are inhabited – the effect is to remove one token on each pre-condition (unlike what the diagram suggests, there may be several), and add one token on each post-condition. From this simple rule emerges complex concurrent phenomena, reviewed now. In Figure 5 we show a Petri net with a simple parallel behaviour: the initial transition **fork** enables transitions **a** and **b**, which occur completely independently of each other. Once both have fired, **join** may occur. This behaviour may equivalently be displayed as the partial order drawn on the right hand side. In Figure 6, we show a net with a simple non-deterministic behaviour as both transitions **a** and **b** compete for the same token. To draw this in partial order style, we need an additional *conflict* relation, pictured as a wiggly line, which indicates that **a** and **b** are incompatible – together,

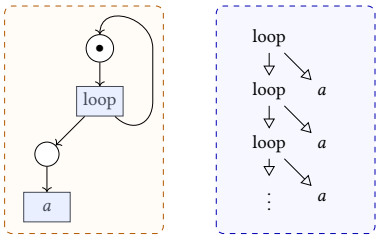


Fig. 7. Recursion

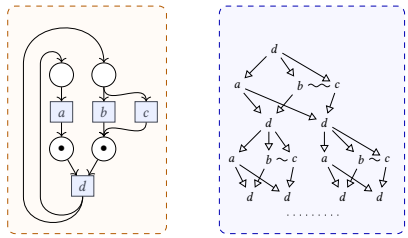
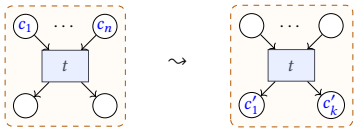


Fig. 8. Loops with conflict

the partial order and the conflict relation form an *event structure* [Winskel 1986]. Petri nets are *folded* structures: they may comprise *loops*. Just like possibly cyclic finite transition systems unfold to infinite trees, Petri nets unfold to event structures [Hayman and Winskel 2008; Nielsen et al. 1981]: this is illustrated in Figure 7 and in the presence of other phenomena in Figure 8.

In our work, a program will be translated to a Petri net and the matching concurrent strategy will be its unfolding. But IPA is not a linear or affine language. Looking back at the GoI token machine of Section 1.2, it appears that tokens must carry an additional piece of data: an *exponential signature*. Petri nets where the tokens carry an additional data are called *coloured Petri nets*: there, tokens have a distinguished identity formalized as an element of a given set of *colours*, and transitions



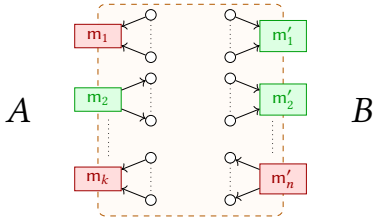
specify the new colours via a transition function, *e.g.* here $\delta\langle t\rangle(c_1, \ldots, c_n) = (c'_1, \ldots, c'_k)$.

In this paper we introduce *Petri structures*, certain coloured Petri nets, and aim to translate programs to Petri structures systematically. But how shall we do that?

1.4 Compositional Construction of Coloured Petri Nets

Just like the translation of a term to a proof net, we shall translate programs compositionally, so that the net reflects the structure of the term. It is convenient to set this up as for a denotational semantics: we provide Petri structures for the basic building blocks of IPA, and assemble them by equipping Petri structures with adequate categorical-like operations. Some existing works already propose ways to compose Petri nets: notably, the *open Petri nets* of [Baez and Master 2020], note also the compositional unfolding of coloured Petri nets of [Chatain and Fabre 2010].

In these works, Petri nets communicate via locations: the external interface of an open Petri net consists in certain locations on which tokens may be exchanged with the environment. In contrast, our Petri structures communicate via *transitions*: the external interface of a Petri structure consists in distinguished "visible" transitions via which the net can receive (for *negative* transitions) and send (for *positive* transition) tokens to the outside world. Petri structures "from A to B" have shape



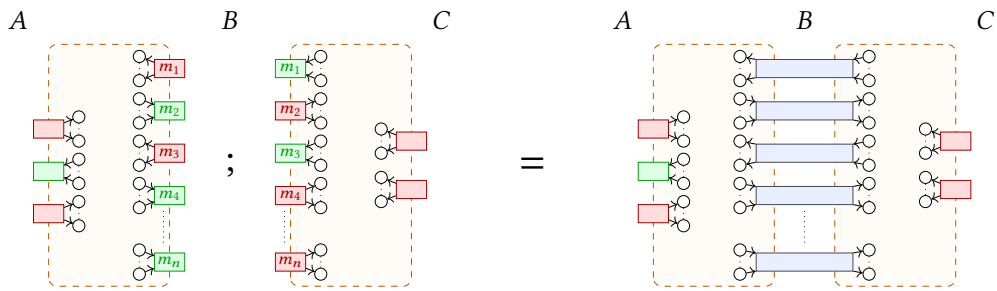


Fig. 9. Composition of Petri nets

where A and B respectively inform sets of addresses $\{\mathsf{m}_1, \ldots, \mathsf{m}_k\}$ and $\{\mathsf{m}'_1, \ldots, \mathsf{m}'_n\}$. This is inspired by the connection of the traditional GoI token machine with game semantics: the m_is and m'_is correspond to occurrences of atoms in dangling wires. In Section 1.2, receiving an Opponent move (m, e) puts a new token with colour e on the dangling wire corresponding to m – we regard this as firing the negative transition m with input e from the environment, introducing a new token inside the net. Thus we regard visible transitions as *channels* using which the net can communicate with the outside world: firing a negative transition matching m *receives* a token e on channel m and (usually, but not necessarily) puts it in an internal location; firing a positive transition matching m with token e on the precondition (usually) sends e to the outside world on channel m.

These Petri structures can be composed simply by synchronizing them on their matching visible transitions, as illustrated in Figure 9 – see the formal definition in Section 2.2.2. Formerly visible transitions synchronize into an internal transition which we consider to have a new *neutral* (0) polarity, allowing tokens to flow silently between the two composed Petri structures.

1.5 Contributions and Outline

Contributions. Firstly, we give a translation of IPA into certain coloured Petri nets called *Petri structures*. This is done as a translation $[\![-]\!]_{\mathsf{PStruct}} : \mathsf{IPA} \to \mathsf{PStruct}$ phrased as a denotational interpretation by induction on syntax. For M a program, $[\![M]\!]$ is regarded as the multi-token machine loaded with M. We define a token game, which – as a corollary from further results of the paper – satisfies *adequacy w.r.t.* the reference operational semantics of IPA (Theorem 3.10).

Secondly, we construct an unfolding to Strat, the concurrent games causal model of IPA [Castellan and Clairambault 2020]. But we cannot go directly from PStruct to Strat: morphisms in PStruct are arbitrary nets whose behaviour need not be semantically sensible. In contrast, in Strat, programs yield strategies which must respect the rules of a *game*, derived from the type of the program. To bridge the gap, we define a new (pre)category PStrat, a hybrid between Petri structures and game semantics. Its objects are games (as in Strat) and its morphisms are *strategic Petri structures* or *Petri strategies*, whose behaviour abides by the game and that satisfy an additional *safety* condition. This gives a second interpretation $[\![-]\!]_{\mathsf{PStrat}} : \mathsf{IPA} \to \mathsf{PStrat}$, but we insist that Petri strategies are simply Petri structures satisfying additional invariants: $[\![M]\!]_{\mathsf{PStruct}}$ and $[\![M]\!]_{\mathsf{PStrat}}$ are the same net, and there is an identity-on-morphisms $\mathcal{F} : \mathsf{PStrat} \to \mathsf{PStruct}$ preserving the interpretation.

We then extend the unfolding of Petri nets to an interpretation-preserving functor

$$\mathcal{U} : \mathsf{PStrat} \to \mathsf{Strat},$$

the main challenge being that unfolding is compatible with composition. Altogether we get

$$\begin{tikzcd} & \mathsf{IPA} \arrow[dl, "{[\![-]\!]_{\mathsf{PStruct}}}"'] \arrow[d, "{[\![-]\!]_{\mathsf{PStrat}}}"] \arrow[dr, "{[\![-]\!]_{\mathsf{Strat}}}"] & \\ \mathsf{PStruct} & \mathsf{PStrat} \arrow[l, "\mathcal{F}"] \arrow[r, "\mathcal{U}"'] & \mathsf{Strat} \end{tikzcd} \tag{2}$$

a commuting diagram letting us deduce adequacy of GoI from the adequacy theorem in Strat. But we regard (2) as a much stronger adequacy theorem: this diagram expresses that even for open, higher-order terms, the *causal* behaviour specified operationally by the GoI multi-token machine coincides with the behaviour specified denotationally by the concurrent games semantics in Strat.

As a final contribution, the multi-token machine for IPA is fully implemented and available at https://ipatopetrinets.github.io/.

Outline. In Section 2 we introduce Petri structures and the core operations on them. In Section 3, we recall IPA and provide Petri structures for its primitives, altogether providing our interpretation. In Section 4, we briefly recall concurrent games, introduce PStrat and describe the unfolding. In Section 5 we describe our implementation, and in Section 6 we conclude.

2 PETRI STRUCTURES

In this section we introduce our representation of programs called *Petri structures* and show their main compositionality properties. In Section 2.1, we give the formal definition of Petri structures. In Section 2.2, we show that Petri structures naturally form a (pre)category, *i.e.* a category without unit laws. In Section 2.3, we discuss how to represent nonlinear programs.

2.1 Definition and Examples

The definition of Petri structures depends on two parameters: firstly, a set Tok of *tokens*. This is the set of *colours*, the same for all locations. Secondly, a set $\mathcal{M}$ of *addresses*, serving the role of the "multiplicative addresses" from the introduction. Each $\mathsf{m} \in \mathcal{M}$ has a *polarity* $\mathsf{pol}(\mathsf{m}) \in \{-,+\}$, specifying whether it is a label for actions by the program (+) or the environment (−). In our interpretation these two sets have a concrete definition, which we shall postpone until later on.

As a convention we write $\uplus$ for set-theoretic union, when it is assumed or known disjoint.

Definition 2.1. A **Petri structure on** $M \subseteq_f \mathcal{M}$ is a tuple

$$\boldsymbol{\sigma} = \langle \mathcal{L}, \mathcal{T} = \mathcal{T}^+ \uplus \mathcal{T}^0 \uplus \mathcal{T}^-, \partial, \mathsf{pre}, \mathsf{post}, \delta \rangle$$

where $\mathcal{L}$ is a finite set of **locations**, $\mathcal{T}$ is a finite set of **transitions** sorted by *polarity* +, 0 or −,

$$\begin{array}{rl} \partial & \text{is a labeling } \mathcal{T}^+ \uplus \mathcal{T}^- \to M \text{ preserving polarity,} \\ \mathsf{pre} & \text{is a function } \mathcal{T} \to \mathcal{P}(\mathcal{L}) \text{ of } \textbf{pre-conditions}, \\ \mathsf{post} & \text{is a function } \mathcal{T} \to \mathcal{P}(\mathcal{L}) \text{ of } \textbf{post-conditions}, \end{array}$$

such that $\mathsf{pre}(t^-) = \emptyset$, $\mathsf{post}(t^+) = \emptyset$ for all transitions with the indicated polarity; and δ assigns to any $t \in \mathcal{T}$ a partial function, the **transition function**, typed according to:

$$\begin{array}{rcrcl} \delta\langle t^0 \rangle & : & \mathsf{cond}(\mathsf{pre}(t)) & \rightharpoonup & \mathsf{cond}(\mathsf{post}(t)), \\ \delta\langle t^- \rangle & : & \mathsf{Tok} & \rightharpoonup & \mathsf{cond}(\mathsf{post}(t)), \\ \delta\langle t^+ \rangle & : & \mathsf{cond}(\mathsf{pre}(t)) & \rightharpoonup & \mathsf{Tok}, \end{array}$$

where $\mathsf{cond}(L) = \mathsf{Tok}^L$ is the set of **conditions** of support L – we write cond the set of all conditions.

Hopefully, the earlier discussion makes this definition natural. To illustrate it, we start by exhibiting a few Petri structures with limited interaction with the environment.

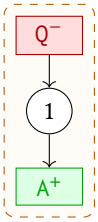
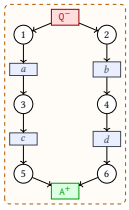
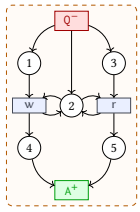


Fig. 10



$$\begin{array}{lcl} \delta\langle \mathsf{Q}\rangle\,(\bullet) & = & \{\bullet^{@1}, \bullet^{@2}\} \\ \delta\langle a\rangle\,(\{\bullet^{@1}\}) & = & \{\checkmark^{@3}\} \\ \delta\langle b\rangle\,(\{\bullet^{@2}\}) & = & \{\checkmark^{@4}\} \\ \delta\langle c\rangle\,(\{v^{@3}\}) & = & \{v^{@5}\} \\ \delta\langle d\rangle\,(\{v^{@4}\}) & = & \{v^{@6}\} \\ \delta\langle \mathsf{A}\rangle\,(\{\checkmark^{@5}, v^{@6}\}) & = & v \end{array}$$

Fig. 11. Petri structure for skip ∥ skip : $\mathbb{U}$



$$\begin{array}{lcl} \delta\langle \mathsf{Q}\rangle\,(\bullet) & = & \{1^{@1}, 0^{@2}, \bullet^{@3}\} \\ \delta\langle \mathsf{w}\rangle\,(\{n^{@1}, p^{@2}\}) & = & \{\checkmark^{@4}, n^{@2}\} \\ \delta\langle \mathsf{r}\rangle\,(\{n^{@2}, \bullet^{@3}\}) & = & \{n^{@2}, n^{@5}\} \\ \delta\langle \mathsf{A}\rangle\,(\{\checkmark^{@4}, n^{@5}\}) & = & \mathrm{iszero}(n) \end{array}$$

Fig. 12. Petri structure for coin : $\mathbb{B}$

2.1.1 Closed Petri structures. This limited interaction will take place via two addresses Q (negative, for "Question" – the channel used by the environment to initiate computation) and A (positive, for "Answer" – the channel used by the program to return a value); so we temporarily fix $\mathcal{M} = \{\mathsf{Q}^-, \mathsf{A}^+\}$. Our first examples include no duplication of resources, hence there is no need for exponential signatures yet. However, and unlike for the plain λ-calculus of Section 1.2, we will need tokens to carry *values* that may arise during computation. We define the **data signatures**

$$\mathcal{D} ::= n \mid \mathtt{tt} \mid \mathtt{ff} \mid \checkmark \mid \bullet \tag{3}$$

that reflect the values of ground datatypes in IPA, with $\bullet$ a special dummy data, used to fill the data field for tokens with no defined value. We temporarily fix $\mathsf{Tok} = \mathcal{D}$.

We draw a Petri structure $\boldsymbol{\sigma}$ following the conventions introduced earlier: *locations* are circles, *transitions* are boxes – note that in our diagrams in the paper, the integers appearing in locations are not tokens, but simply identifiers for the locations. Altogether the graph drawn carries the information of $\mathcal{L}$, $\mathcal{T}$, pre and post, while δ is a separate transition table. Whenever unambiguous, we name visible transitions simply as their label via ∂. If $L = \{l_1, \ldots, l_n\} \subseteq \mathcal{L}$, a condition $(\mathsf{t}_i)_{l_i \in L} \in \mathrm{cond}(L)$ is written $\{\mathsf{t}_1^{@l_1}, \ldots, \mathsf{t}_n^{@l_n}\}$ where each $l_i \in L$ appears exactly once. An individual $\mathsf{t}^{@l}$ (for $\mathsf{t} \in \mathsf{Tok}$, $l \in \mathcal{L}$) is called a **token-in-location**, or **tokil** – we write $\mathsf{TokIL}(\boldsymbol{\sigma})$ for the set of tokils.

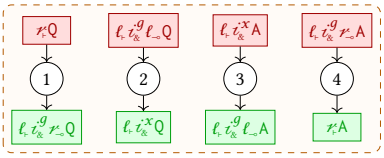
We start with the Petri structure **skip** in Figure 10: it shall interpret the constant skip : $\mathbb{U}$ of IPA, the unique value of type $\mathbb{U}$, which performs no action. Upon being triggered by receiving $\bullet$ on Q^-, the net prepares data $\checkmark$ in location 1 by $\delta\langle \mathsf{Q}^-\rangle(\bullet) = \{\checkmark^{@1}\}$. This enables A^+, which outputs the value $\checkmark$ via $\delta\langle \mathsf{A}^+\rangle(\{v^{@1}\}) = v$. Note that in this transition, v stands for any token, so – for now – any data value in $\mathcal{D}$. Next, Figure 11 presents a parallel evaluation of skip ∥ skip. When triggered the net throws two tokils $\bullet^{@1}$ and $\bullet^{@2}$ corresponding to evaluation requests for the two constants. Both tokils are forwarded to an independent copy of the structure of Figure 10. Upon receiving the two values in locations 5 and 6, the last transition fires and outputs $\checkmark$ on A^+. The example – or its closed variant obtained by the interpretation – may be run here.

Petri nets can also represent shared state: Figure 12 shows the closed Petri structure for the term

$$\vdash \mathsf{newref}\ r\ \mathsf{in}\ (r := 1 \parallel \mathsf{iszero}\ !r) : \mathbb{B} \tag{4}$$

written coin, a non-deterministic choice obtained by setting up a race in the memory (by convention newref initializes r to 0). Though we have not yet introduced IPA, we hope this example is clear – in particular it is worth pointing out our nonstandard typing rule for parallel composition, which allows us to put a program of unit type in parallel with a program of arbitrary ground type.

Upon initialization (by receiving $\bullet$ on Q^-), the net throws three tokens: the tokil $0^{@2}$ initializes the variable to 0; the tokil $1^{@1}$ is a write request for the value 1; and $\bullet^{@3}$ is a read request. There is a race between the read r and the write w: if r wins, the value in location 5 ends up being 0. If w wins then r reads value 1 instead. The final transition waits for the write acknowledgment and the result of the read to send the value read on A^+. This example may be run in the implementation here.



$$\begin{array}{lcl} \delta\langle \mathcal{r}_{\vdash}\mathsf{Q}\rangle\ (\bullet) & = & \{\bullet^{@1}\} \\ \delta\langle \ell_{\vdash}\mathcal{i}^{g}_{\&}\ell_{\multimap}\mathsf{Q}\rangle(\bullet) & = & \{\bullet^{@2}\} \\ \delta\langle \ell_{\vdash}\mathcal{i}^{x}_{\&}\mathsf{A}\rangle\ (\checkmark) & = & \{\checkmark^{@3}\} \\ \delta\langle \ell_{\vdash}\mathcal{i}^{g}_{\&}\mathcal{r}_{\multimap}\mathsf{A}\rangle(\checkmark) & = & \{\checkmark^{@4}\} \end{array} \qquad \begin{array}{lcl} \delta\langle \ell_{\vdash}\mathcal{i}^{g}_{\&}\mathcal{r}_{\multimap}\mathsf{Q}\rangle(\{\bullet^{@1}\}) & = & \bullet \\ \delta\langle \ell_{\vdash}\mathcal{i}^{x}_{\&}\mathsf{Q}\rangle\ (\{\bullet^{@2}\}) & = & \bullet \\ \delta\langle \ell_{\vdash}\mathcal{i}^{g}_{\&}\ell_{\multimap}\mathsf{A}\rangle(\{\checkmark^{@3}\}) & = & \checkmark \\ \delta\langle \mathcal{r}_{\vdash}\mathsf{A}\rangle\ (\{\checkmark^{@4}\}) & = & \checkmark \end{array}$$

Fig. 13. An open Petri structure for $g : \mathbb{U} \to \mathbb{U}, x : \mathbb{U} \vdash g\,x : \mathbb{U}$

2.1.2 Open Petri structures. The addresses Q^- and A^+ are enough to represent the limited interactive behaviour of closed programs of ground type, but *open* programs will require a wider range of addresses. For instance, Petri structures corresponding to programs typed with $g : \mathbb{U} \to \mathbb{U}, x : \mathbb{U} \vdash \mathbb{U}$ will commucate with their environment through the following addresses:

$$M \quad = \quad \{\ell_{\vdash}\mathcal{i}^{g}_{\&}\ell_{\multimap}\mathsf{Q}^-,\quad \ell_{\vdash}\mathcal{i}^{g}_{\&}\ell_{\multimap}\mathsf{A}^+,\quad \ell_{\vdash}\mathcal{i}^{g}_{\&}\mathcal{r}_{\multimap}\mathsf{Q}^+,\quad \ell_{\vdash}\mathcal{i}^{g}_{\&}\mathcal{r}_{\multimap}\mathsf{A}^-,\quad \ell_{\vdash}\mathcal{i}^{x}_{\&}\mathsf{Q}^+,\quad \ell_{\vdash}\mathcal{i}^{x}_{\&}\mathsf{A}^-,\quad \mathcal{r}_{\vdash}\mathsf{Q}^-,\quad \mathcal{r}_{\vdash}\mathsf{A}^+\}$$

where the *injections* $\ell_{\vdash}$ and $\mathcal{r}_{\vdash}$ indicate the two sides of $\vdash$; $\mathcal{i}^{x}_{\&}$ and $\mathcal{i}^{g}_{\&}$ point to a variable name; $\ell_{\multimap}$ and $\mathcal{r}_{\multimap}$ point to either side of an arrow $\to$; and Q and A stand for "Question" (call) or "Answer" (return). Without Q and A, each element of M corresponds to an occurrence of a base type in $g : \mathbb{U} \to \mathbb{U}, x : \mathbb{U} \vdash \mathbb{U}$, and each such occurrence admits a *call* (Q) and a *return* (A).

Based on this set of addresses M, we show in Figure 13 the Petri structure for $g : \mathbb{U} \to \mathbb{U}, x : \mathbb{U} \vdash g\,x : \mathbb{U}$. Upon initialization when receiving token $\bullet$ on $\mathcal{r}_{\vdash}\mathsf{Q}^-$, the net interrogates the return value of g by sending $\bullet$ on $\ell_{\vdash}\mathcal{i}^{g}_{\&}\mathcal{r}_{\multimap}\mathsf{Q}^+$. If g calls its argument with $\ell_{\vdash}\mathcal{i}^{g}_{\&}\ell_{\multimap}\ \mathsf{Q}^-$, the net interrogates the return value of x. If x returns a value with $\ell_{\vdash}\mathcal{i}^{x}_{\&}\mathsf{A}^-$, this value is propagated to the argument of g. Finally, if g returns with $\ell_{\vdash}\mathcal{i}^{g}_{\&}\mathcal{r}_{\multimap}\mathsf{A}$, the value is forwarded to the right hand side via $\mathcal{r}_{\vdash}\mathsf{A}^+$. Readers familiar with proof nets will recognize the axiom links in the wiring of Figure 13, readers familiar with game semantics will recognize pairs of Opponent moves and induced Player responses. This example may be run here — for clarity the implementation displays the hierarchical relationship between calls and returns, even though those are not part of the Petri structure.

The set M used for this example is a subset of the full set of **addresses**, defined as

$$\mathcal{M} \quad ::= \quad \ell_{\otimes}\mathcal{M} \mid \mathcal{r}_{\otimes}\mathcal{M} \mid \ell_{\vdash}\mathcal{M} \mid \mathcal{r}_{\vdash}\mathcal{M} \mid \ell_{\multimap}\mathcal{M} \mid \mathcal{r}_{\multimap}\mathcal{M} \mid \mathcal{i}^{x}_{\&}\mathcal{M} \mid \mathcal{w}_{\mathbb{V}}\mathcal{M} \mid \mathcal{r}_{\mathbb{V}}\mathcal{M} \mid \mathcal{g}_{\mathbb{S}}\mathcal{M} \mid \mathcal{r}_{\mathbb{S}}\mathcal{M} \mid \mathsf{Q} \mid \mathsf{A},$$

for x any element of a countable set Var of variables – we use m to range over addresses. These addresses allow us to represent explicitly the tags in disjoint union: for instance, the tensor of two set of addresses M and M' is $M \otimes M' = \ell_{\otimes}(M) \uplus \mathcal{r}_{\otimes}(M')$ where *e.g.* $\ell_{\otimes}(M) = \{\ell_{\otimes}m \mid m \in M\}$. Other constructions are used similarly; the constructors indexed by $\mathbb{V}$ and $\mathbb{S}$ correspond to addresses for reference and semaphore types. The **polarity** of addresses is defined as $\mathsf{pol}(\mathsf{Q}) = -$, $\mathsf{pol}(\mathsf{A}) = +$, $\mathsf{pol}(\ell_{\vdash}\mathsf{m}) = -\mathsf{pol}(\mathsf{m})$, $\mathsf{pol}(\ell_{\multimap}\mathsf{m}) = -\mathsf{pol}(\mathsf{m})$, and preserved in all other cases.

2.2 The Precategory PStruct

At the core of our approach is the composition of Petri structures, and the corresponding identities – one might expect those to form a category. We shall see that though composition is associative (up to isomorphism), the identity laws fail (even up to isomorphism), calling for the next notion:

2.2.1 Precategories. It is good to keep in mind that Petri structures should really be regarded not as a semantics, but as an alternative representation of syntax. So composing a Petri structure with the identity plugs it into a purely forwarder net which, though not changing the *behaviour* (as made formal by the unfolding in Section 4), does add new transitions and locations.

Thus we get a structure with associativity but no identity laws, a **precategory**:

Definition 2.2. A (small) **precategory** C is given by: *(1)* a set of **objects** C_0; *(2)* for each $A, B \in C_0$ a set of **morphisms** $C(A, B)$; *(3)* for each $A \in C_0$, an **identity** $\mathsf{id}_A \in C(A, A)$; and *(4)* for each

$f \in \mathcal{C}(A, B)$ and $g \in \mathcal{C}(B, C)$, a **composition** $g \circ f \in \mathcal{C}(A, C)$; which is *associative, i.e.* we have

$$h \circ (g \circ f) = (h \circ g) \circ f$$

for all $f \in \mathcal{C}(A, B), g \in \mathcal{C}(B, C)$ and $h \in \mathcal{C}(C, D)$.

A category is in particular a precategory, and the definition of **functors** between categories applies transparently to precategories. We insist that we do *not* demand $\mathsf{id}_B \circ f = f \circ \mathsf{id}_A = f$. One might wonder what it means to call something an "identity" without identity laws. The point is that functors must still preserve these "identities", so in particular a functor from a precategory to a category (such as $\mathcal{U}$ in this paper) will send id_A to an actual identity *with* identity laws.

Before defining the precategory PStruct, we need one final component:

Definition 2.3. Consider $\boldsymbol{\sigma}, \boldsymbol{\tau}$ two Petri structures on $M \subseteq \mathcal{M}$. An **isomorphism** $\varphi : \boldsymbol{\sigma} \cong \boldsymbol{\tau}$ consists of bijections $\varphi_{\mathcal{L}} : \mathcal{L}_{\boldsymbol{\sigma}} \simeq \mathcal{L}_{\boldsymbol{\tau}}$ and $\varphi_{\mathcal{T}} : \mathcal{T}_{\boldsymbol{\sigma}} \simeq \mathcal{T}_{\boldsymbol{\tau}}$ compatible with all structure.

Two Petri structures $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$ on M are **isomorphic**, written $\boldsymbol{\sigma} \cong \boldsymbol{\tau}$, if there is an (unspecified) isomorphism $\varphi : \boldsymbol{\sigma} \cong \boldsymbol{\tau}$. This is an equivalence relation on Petri structures on M. Now we set:

Definition 2.4. The precategory PStruct has: *(1)* objects, all finite subsets of $\mathcal{M}$; *(2)* morphisms from M to N, isomorphism classes of Petri structures on $M \vdash N = \ell_{\vdash}(M) \uplus \mathscr{r}_{\vdash}(N)$; *(3)* composition and identities, described respectively in Sections 2.2.2 and 2.2.3.

Though morphisms of PStruct are isomorphism classes of Petri structures, in the sequel we shall treat them as concrete Petri structures, keeping in mind the additional proof obligation that all constructions must preserve isomorphisms (which is always straightforward). Thus we may write $\boldsymbol{\sigma} \in \mathsf{PStruct}(M, N)$ for $\boldsymbol{\sigma}$ a concrete Petri structure, or simply $\boldsymbol{\sigma} : M \vdash N$.

As stated before, we shall formulate the translation of IPA to Petri structures as an interpretation of programs in PStruct. While this interpretation adopts the language and methodology of denotational semantics, it is *not* a denotational semantics in the popular sense of being an invariant of reduction. The Petri structure of a term certainly changes under reduction: the first symptom of that is the fact that identity laws are missing in PStruct. We argue that despite the missing equations the translation remains compositional, in the sense that the translation of a compound expression is a function of the interpretation of its components. It is compositional in the same sense that *e.g.* the translations of the λ-calculus to proof nets or to the π-calculus are compositional.

2.2.2 Composition. Fix two Petri structures $\boldsymbol{\sigma} \in \mathsf{PStruct}(M, N)$ and $\boldsymbol{\tau} \in \mathsf{PStruct}(N, P)$; we aim to define $\boldsymbol{\tau} \odot \boldsymbol{\sigma} \in \mathsf{PStruct}(M, P)$, their *composition*. Composing $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$ amounts to synchronizing $\boldsymbol{\sigma}$'s visible transitions on the *right* with $\boldsymbol{\tau}$'s visible transitions on the left:

Definition 2.5. Visible $t^{\boldsymbol{\sigma}} \in \mathcal{T}_{\boldsymbol{\sigma}}^{+} \uplus \mathcal{T}_{\boldsymbol{\sigma}}^{-}$ and $t^{\boldsymbol{\tau}} \in \mathcal{T}_{\boldsymbol{\tau}}^{+} \uplus \mathcal{T}_{\boldsymbol{\tau}}^{-}$ are **synchronizable** if they have opposite polarities; and $\partial_{\boldsymbol{\sigma}}(t^{\boldsymbol{\sigma}}) = \mathscr{r}_{\vdash}\mathsf{m}$, $\partial_{\boldsymbol{\tau}}(t^{\boldsymbol{\tau}}) = \ell_{\vdash}\mathsf{m}$ for some $\mathsf{m} \in N$. We define the set of transitions:

$$\begin{array}{rcl} \mathcal{T}_{\boldsymbol{\tau}} \circledast \mathcal{T}_{\boldsymbol{\sigma}} & = & \{\ell^{\odot}(t) \mid t \in \mathcal{T}_{\boldsymbol{\sigma}}^{0} \uplus \mathcal{T}_{\boldsymbol{\sigma}}^{p,\ell_{\vdash}}\} \\ & \uplus & \{\mathscr{r}^{\odot}(t) \mid t \in \mathcal{T}_{\boldsymbol{\tau}}^{0} \uplus \mathcal{T}_{\boldsymbol{\tau}}^{p,\mathscr{r}_{\vdash}}\} \\ & \uplus & \{t^{\boldsymbol{\sigma}} \circledast t^{\boldsymbol{\tau}} \mid t^{\boldsymbol{\sigma}} \in \mathcal{T}_{\boldsymbol{\sigma}}, t^{\boldsymbol{\tau}} \in \mathcal{T}_{\boldsymbol{\tau}} \text{ synchronizable.}\} \end{array}$$

with $\mathcal{T}_{\boldsymbol{\sigma}}^{p,\imath}$ transitions of polarity $p \in \{-, +\}$ and label $\partial(t) = \imath\mathsf{m}$ for $\mathsf{m} \in \mathcal{M}$; likewise for $\mathcal{T}_{\boldsymbol{\tau}}^{p,\imath}$.

Intuitively, $\mathcal{T}_{\boldsymbol{\tau}} \circledast \mathcal{T}_{\boldsymbol{\sigma}}$ imports an unsynchronized t from $\boldsymbol{\sigma}$ as $\ell^{\odot}(t)$, an unsynchronized t from $\boldsymbol{\tau}$ as $\mathscr{r}^{\odot}(t)$, but also a new transition $t^{\boldsymbol{\sigma}} \circledast t^{\boldsymbol{\tau}}$ for every synchronizable pair. Here, $\ell^{\odot}$ and $\mathscr{r}^{\odot}$ are formal injections used to keep transitions from $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$ disjoint. From now on, if X and Y are sets, we write $X +^{\odot} Y = \ell^{\odot}(X) \uplus \mathscr{r}^{\odot}(Y)$. We shall use the same convention with other tags later on – or simply write $X + Y = \ell(X) \uplus \mathscr{r}(Y)$. Now, we may finally define the **composition** $\boldsymbol{\tau} \odot \boldsymbol{\sigma} : M \vdash P$ as:

$$\begin{array}{rcl} \mathsf{pre}_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}}(\ell^{\odot}(t)) & = & \ell^{\odot}(\mathsf{pre}_{\boldsymbol{\sigma}}(t)) \\ \mathsf{pre}_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}}(r^{\odot}(t)) & = & r^{\odot}(\mathsf{pre}_{\boldsymbol{\tau}}(t)) \\ \mathsf{pre}_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}}(t^{+} \circledast t^{-}) & = & \ell^{\odot}(\mathsf{pre}_{\boldsymbol{\sigma}}(t^{+})) \\ \mathsf{pre}_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}}(t^{-} \circledast t^{+}) & = & r^{\odot}(\mathsf{pre}_{\boldsymbol{\tau}}(t^{+})) \end{array} \qquad \begin{array}{rcl} \mathsf{post}_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}}(\ell^{\odot}(t)) & = & \ell^{\odot}(\mathsf{post}_{\boldsymbol{\sigma}}(t)) \\ \mathsf{post}_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}}(r^{\odot}(t)) & = & r^{\odot}(\mathsf{post}_{\boldsymbol{\tau}}(t)) \\ \mathsf{post}_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}}(t^{+} \circledast t^{-}) & = & r^{\odot}(\mathsf{post}_{\boldsymbol{\tau}}(t^{-})) \\ \mathsf{post}_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}}(t^{-} \circledast t^{+}) & = & \ell^{\odot}(\mathsf{post}_{\boldsymbol{\sigma}}(t^{-})) \end{array}$$

Fig. 14. Pre-conditions and post-conditions for the composition

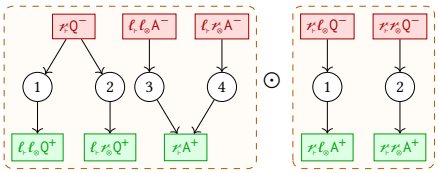


Fig. 15. A composition yielding Figure 11

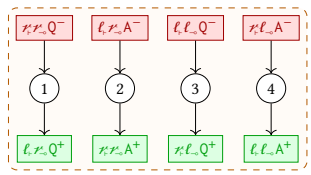


Fig. 16. The Petri structure $\mathfrak{a}_{!\mathbf{N}\multimap\mathbf{N}}$

Definition 2.6. We set $\mathcal{L}_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} = \mathcal{L}_{\boldsymbol{\sigma}} +^{\odot} \mathcal{L}_{\boldsymbol{\tau}}$; $\mathcal{T}_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} = \mathcal{T}_{\boldsymbol{\tau}} \circledast \mathcal{T}_{\boldsymbol{\sigma}}$; $\mathcal{T}^{p}_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} = \mathcal{T}^{p,\ell_{\vdash}}_{\boldsymbol{\sigma}} +^{\odot} \mathcal{T}^{p,r_{\rightarrow}}_{\boldsymbol{\tau}}$ where $p \in \{+,-\}$; $\partial_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}}(\ell^{\odot}(t)) = \partial_{\boldsymbol{\sigma}}(t)$ and $\partial_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}}(r^{\odot}(t)) = \partial_{\boldsymbol{\tau}}(t)$. Conditions are in Figure 14, and:

$$\begin{array}{rcl} \delta_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}}\langle \ell^{\odot}(t)\rangle(\ell^{\odot}(\alpha)) & = & \ell^{\odot}(\delta_{\boldsymbol{\sigma}}\langle t\rangle(\alpha)) \\ \delta_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}}\langle r^{\odot}(t)\rangle(r^{\odot}(\beta)) & = & r^{\odot}(\delta_{\boldsymbol{\tau}}\langle t\rangle(\beta)) \\ \delta_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}}\langle t^{+} \circledast t^{-}\rangle(\ell^{\odot}(\alpha)) & = & r^{\odot}((\delta_{\boldsymbol{\tau}}\langle t^{-}\rangle \circ \delta_{\boldsymbol{\sigma}}\langle t^{+}\rangle)(\alpha)) \\ \delta_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}}\langle t^{-} \circledast t^{+}\rangle(r^{\odot}(\beta)) & = & \ell^{\odot}((\delta_{\boldsymbol{\sigma}}\langle t^{-}\rangle \circ \delta_{\boldsymbol{\tau}}\langle t^{+}\rangle)(\beta)) \end{array}$$

with $\ell^{\odot}, r^{\odot}$ applied to $\alpha \in \mathsf{cond}_{\boldsymbol{\sigma}}, \beta \in \mathsf{cond}_{\boldsymbol{\tau}}$ by retagging locations, *i.e.* $\ell^{\odot}(\alpha) = \{(\mathsf{s},\mathsf{d})^{@\ell^{\odot}(l)} \mid (\mathsf{s},\mathsf{d})^{@l} \in \alpha\} \in \mathsf{cond}_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}}$ and $r^{\odot}(\beta) = \{(\mathsf{s},\mathsf{d})^{@r^{\odot}(l)} \mid (\mathsf{s},\mathsf{d})^{@l} \in \beta\} \in \mathsf{cond}_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}}$.

We show in Figure 15 an example composition, yielding (up to isomorphism) the structure of Figure 11. Notice how the two copies of Figure 10 *glue* together the disconnected components – for *calls* and *returns* – of the left hand side operand.

Runs on $\boldsymbol{\tau} \odot \boldsymbol{\sigma}$ happen as follows: at first, the token game is played independently in $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$. If $\boldsymbol{\sigma}$ wishes to play a positive transition on the right (resp. $\boldsymbol{\tau}$ wishes to play a positive transition on the left), it synchronizes with a matching negative transition on the other side – if it exists. The resulting (neutral) synchronized transition has transition function the composite of the synchronized transitions. The effect is that tokens "jump" between $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$, following the control flow.

2.2.3 The copycat Petri structure. Next, PStruct requires an "identity": the *copycat* Petri structure. *Copycat* exchanges tokens between left and right, forwarding negative moves on either side to the matching positive move on the other side, keeping tokens otherwise unchanged.

Definition 2.7. For $M \subseteq \mathcal{M}$ finite, we define the **copycat Petri structure** on M, written $\mathfrak{a}_M$. Its *locations* are $\mathcal{L}_{\mathfrak{a}_M} = M$, its *transitions* are $\mathcal{T}_{\mathfrak{a}_M} = M \times \{\ell, r\}$ with polarities as in

$$\begin{array}{rcl} \mathcal{T}^{+}_{\mathfrak{a}_M} & = & (M^{+} \times \{r\}) \uplus (M^{-} \times \{\ell\}) \\ \mathcal{T}^{-}_{\mathfrak{a}_M} & = & (M^{-} \times \{r\}) \uplus (M^{+} \times \{\ell\}) \end{array}$$

and no neutral transition, for $M^{+} = \{\mathsf{m} \in M \mid \mathsf{pol}(m) = +\}$, and likewise for M^{-}. These transitions are mapped to addresses with $\partial(\mathsf{m},\ell) = \ell_{\vdash}\mathsf{m}$ and $\partial(\mathsf{m},r) = r_{\vdash}\mathsf{m}$; we set $\mathsf{pre}(\mathsf{m}^{+},r)$, $\mathsf{pre}(\mathsf{m}^{-},\ell)$,

$$\begin{array}{rcl} \mathsf{pre}_{\boldsymbol{\sigma}\otimes\boldsymbol{\tau}}(\ell^{\otimes}(t)) & = & \ell^{\otimes}(\mathsf{pre}_{\boldsymbol{\sigma}}(t)) \\ \mathsf{pre}_{\boldsymbol{\sigma}\otimes\boldsymbol{\tau}}(r^{\otimes}(t)) & = & r^{\otimes}(\mathsf{pre}_{\boldsymbol{\tau}}(t)) \end{array} \qquad \begin{array}{rcl} \mathsf{post}_{\boldsymbol{\sigma}\otimes\boldsymbol{\tau}}(\ell^{\otimes}(t)) & = & \ell^{\otimes}(\mathsf{post}_{\boldsymbol{\sigma}}(t)) \\ \mathsf{post}_{\boldsymbol{\sigma}\otimes\boldsymbol{\tau}}(r^{\otimes}(t)) & = & r^{\otimes}(\mathsf{post}_{\boldsymbol{\tau}}(t)) \end{array}$$

Fig. 17. Pre-conditions and post-conditions for the tensor operation

$\mathsf{post}(\mathsf{m}^-, r)$, $\mathsf{post}(\mathsf{m}^+, \ell)$ to $\{\mathsf{m}\}$ and pre and post returning $\emptyset$ elsewhere. Finally:

$$\begin{array}{rcl} \delta\langle(\mathsf{m}^+, r)\rangle(\{(\mathsf{s},\mathsf{d})^{@\mathsf{m}}\}) & = & (\mathsf{s},\mathsf{d}) \\ \delta\langle(\mathsf{m}^-, \ell)\rangle(\{(\mathsf{s},\mathsf{d})^{@\mathsf{m}}\}) & = & (\mathsf{s},\mathsf{d}) \\ \delta\langle(\mathsf{m}^-, r)\rangle(\mathsf{s},\mathsf{d}) & = & \{(\mathsf{s},\mathsf{d})^{@\mathsf{m}}\} \\ \delta\langle(\mathsf{m}^+, \ell)\rangle(\mathsf{s},\mathsf{d}) & = & \{(\mathsf{s},\mathsf{d})^{@\mathsf{m}}\}, \end{array}$$

altogether giving $\mathbf{c}_M : M \vdash M$, a Petri structure from M to M.

As an example, we show in Figure 16 the Petri structure $\mathbf{c}_{!\mathbf{N}\multimap\mathbf{N}}$, writing $!\mathbf{N} \multimap \mathbf{N}$ for the set $\{\ell_{\multimap}\mathsf{Q}^+, \ell_{\multimap}\mathsf{A}^-, r_{\multimap}\mathsf{Q}^-, r_{\multimap}\mathsf{A}^+\}$ which shall arise as the interpretation of $\mathbb{N} \to \mathbb{N}$.

Composition is associative up to isomorphism, making PStruct a precategory. However, composing with copycat yields a structure with strictly more nodes – copycat will be neutral for composition only after *unfolding* as it will unfold to the copycat strategy, see Section 4.

2.2.4 Tensor. Another construction at the core of categorical models of programming languages is the *tensor* operation, which we now define for Petri structures:

Definition 2.8. Consider $\boldsymbol{\sigma} \in \mathsf{PStruct}(M_1, N_1)$ and $\boldsymbol{\tau} \in \mathsf{PStruct}(M_2, N_2)$.
We set $\mathcal{L}_{\boldsymbol{\sigma}\otimes\boldsymbol{\tau}} = \mathcal{L}_{\boldsymbol{\sigma}} +^{\otimes} \mathcal{L}_{\boldsymbol{\tau}}$; $\mathcal{T}_{\boldsymbol{\sigma}\otimes\boldsymbol{\tau}} = \mathcal{T}_{\boldsymbol{\sigma}} +^{\otimes} \mathcal{T}_{\boldsymbol{\tau}}$ with $\mathcal{T}^p_{\boldsymbol{\sigma}\otimes\boldsymbol{\tau}} = \mathcal{T}^p_{\boldsymbol{\sigma}} +^{\otimes} \mathcal{T}^p_{\boldsymbol{\tau}}$ for $p \in \{+,-\}$;

$$\begin{array}{rcll} \partial_{\boldsymbol{\sigma}\otimes\boldsymbol{\tau}}(\ell^{\otimes}(t)) & = & \ell_{+}\ell_{\otimes}\mathsf{m} & (\partial_{\boldsymbol{\sigma}}(t) = \ell_{+}\mathsf{m}) \\ \partial_{\boldsymbol{\sigma}\otimes\boldsymbol{\tau}}(\ell^{\otimes}(t)) & = & r_{+}\ell_{\otimes}\mathsf{m} & (\partial_{\boldsymbol{\sigma}}(t) = r_{+}\mathsf{m}) \\ \partial_{\boldsymbol{\sigma}\otimes\boldsymbol{\tau}}(r^{\otimes}(t)) & = & \ell_{+}r_{\otimes}\mathsf{m} & (\partial_{\boldsymbol{\tau}}(t) = \ell_{+}\mathsf{m}) \\ \partial_{\boldsymbol{\sigma}\otimes\boldsymbol{\tau}}(r^{\otimes}(t)) & = & r_{+}r_{\otimes}\mathsf{m} & (\partial_{\boldsymbol{\tau}}(t) = r_{+}\mathsf{m}) \end{array}$$

pre- and post-conditions in Figure 17, and for the transition table we set:

$$\begin{array}{rcl} \delta_{\boldsymbol{\sigma}\otimes\boldsymbol{\tau}}\langle\ell^{\otimes}(t)\rangle(\ell^{\otimes}(\alpha)) & = & \ell^{\otimes}(\delta_{\boldsymbol{\sigma}}\langle t\rangle(\alpha)) \\ \delta_{\boldsymbol{\sigma}\otimes\boldsymbol{\tau}}\langle r^{\otimes}(t)\rangle(r^{\otimes}(\alpha)) & = & r^{\otimes}(\delta_{\boldsymbol{\tau}}\langle t\rangle(\alpha)) \end{array}$$

with $\ell^{\otimes}, r^{\otimes}$ applied on conditions as in Definition 2.6. This yields $\boldsymbol{\sigma}\otimes\boldsymbol{\tau} \in \mathsf{PStruct}(M_1 \otimes M_2, N_1 \otimes N_2)$.

This simply puts $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$ side by side without interaction. As an example, the Petri structure on the right hand side of the composition symbol in Figure 15 is $\boldsymbol{\sigma} \otimes \boldsymbol{\sigma}$ for $\boldsymbol{\sigma}$ in Figure 10.

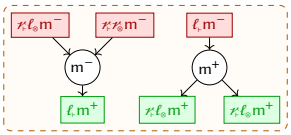
2.3 Exponentials

It is simple to complete the ingredients on Petri structures introduced so far into what could be a *symmetric monoidal closed precategory* (we lack currying and structural morphisms), introduced in Appendix B), sufficient to interpret a linear λ-calculus – the resulting interpretation would yield a Petri structure that is very close to the GoI token machine described for the linear λ-calculus in Section 1.2. However, we have yet to deal with the constructions relative to replication of resources.

2.3.1 Tokens. As in traditional GoI, we enrich tokens with *exponential signatures* so as to be able to address distinct copies of a single resource. This leads us to our definitive notion of tokens:

Definition 2.9. The set Tok of **tokens** is $\mathsf{Tok} = \mathcal{E}^* \times \mathcal{D}$, for $\mathcal{D}$ the data signatures from (3),

$$\mathcal{E} \quad ::= \quad \ell_!\mathcal{E} \mid r_!\mathcal{E} \mid \langle\mathcal{E},\mathcal{E}\rangle \mid \blacklozenge$$



$$
\begin{array}{lcl}
\delta_{\mathbf{c}_M}\langle \nearrow_{\vdash}\ell_{\otimes}\mathsf{m}^-\rangle(\mathsf{e} :: \mathsf{s}, \mathsf{d}) & = & \{(\ell_{\vdash}\mathsf{e} :: \mathsf{s}, \mathsf{d})^{@\mathsf{m}^-}\} \\
\delta_{\mathbf{c}_M}\langle \nearrow_{\vdash}\nearrow_{\otimes}\mathsf{m}^-\rangle(\mathsf{e} :: \mathsf{s}, \mathsf{d}) & = & \{(\nearrow_{\vdash}\mathsf{e} :: \mathsf{s}, \mathsf{d})^{@\mathsf{m}^-}\} \\
\delta_{\mathbf{c}_M}\langle \nearrow_{\vdash}\ell_{\otimes}\mathsf{m}^+\rangle(\{(\ell_{\vdash}\mathsf{e} :: \mathsf{s}, \mathsf{d})^{@\mathsf{m}^+}\}) & = & (\mathsf{e} :: \mathsf{s}, \mathsf{d}) \\
\delta_{\mathbf{c}_M}\langle \nearrow_{\vdash}\nearrow_{\otimes}\mathsf{m}^+\rangle(\{(\nearrow_{\vdash}\mathsf{e} :: \mathsf{s}, \mathsf{d})^{@\mathsf{m}^+}\}) & = & (\mathsf{e} :: \mathsf{s}, \mathsf{d})
\end{array}
$$

Fig. 18. Contraction $\mathbf{c}_M$

is the set of **exponential signatures**, and the set $\mathcal{E}^*$ of finite lists of $\mathcal{E}$ is referred to as **exponential stacks**. We denote exponential stacks via typical list-like notations, *e.g.* $[] \in \mathcal{E}^*$, $[\blacklozenge, \ell_{\vdash}\blacklozenge] \in \mathcal{E}^*$, etc.

All tokens walking around in Petri structures are now pairs (s, d) of an exponential stack $\mathsf{s} \in \mathcal{E}^*$, whose length may vary, and a data value $\mathsf{d} \in \mathcal{D}$. We introduced exponential signatures in Section 1.2; this mechanism corresponds to the linear decomposition of the arrow type $A \to B$ as $!A \multimap B$: $!A$ can be thought of as multiple copies of A indexed by exponential signatures.

Why use exponential stacks and not merely signatures? Linear logic representation of higher-order types have nested exponentials. Each exponential leads to a signature, so we end up with a stack whose length corresponds to the level of nesting. Consider the simple example where a program $f\,M_1 + f\,M_2$ interacts with the context $f := \lambda n.\, n + n$. The program calls f twice with distinct stacks, *e.g.* $[\ell_{\vdash}\blacklozenge]$ and $[\nearrow_{\vdash}\blacklozenge]$. In the first call for instance, the context then reacts by starting two evaluations of M_1 with stacks $[\ell_{\vdash}\blacklozenge, \mathsf{e}_1]$ and $[\ell_{\vdash}\blacklozenge, \mathsf{e}_2]$ where $\mathsf{e}_1, \mathsf{e}_2$ are two distinct signatures chosen by the context – likewise, we will have $[\nearrow_{\vdash}\blacklozenge, \mathsf{e}_3]$ and $[\nearrow_{\vdash}\blacklozenge, \mathsf{e}_4]$ for the second call. In general, stacks alternate between signatures created by the program and signatures created by the context.

If $M \subseteq_f \mathcal{M}$ (an object of PStruct), we set $!M = M$. The addresses are unchanged, but morally the tokens exchanged have a deeper exponential stack (only enforced formally with PStrat). Though there is no practical difference in PStruct, keeping the notation helps typecheck the constructions.

2.3.2 Contraction. As explained in Section 1.2, *contraction* exploits exponential signatures to give distinguished identifiers to every resource usage, and to route tokens accordingly in the net. More precisely, we introduce the following Petri structure for $M \subseteq_f \mathcal{M}$:

Definition 2.10. The **contraction**, a Petri structure $\mathbf{c}_M$ on $!M \vdash !M \otimes !M$, has locations $\mathcal{L}_{\mathbf{c}_M} = M$, transitions $\mathcal{T}_{\mathbf{c}_M} = M +^{\vdash} (M +^{\otimes} M)$ with polarity as for $\mathcal{M}$, and net and transitions in Figure 18.

In Figure 18 and other forthcoming Petri structures, we omit the transition rules that only propagate tokens with no modification: for instance here, we have $\delta_{\mathbf{c}_M}\langle \ell_{\vdash}\mathsf{m}^-\rangle(\mathsf{s}, \mathsf{d}) = \{(\mathsf{s}, \mathsf{d})^{@\mathsf{m}^+}\}$. It is also worth emphasizing that the representation of the net in Figure 18 is symbolic: for each negative $\mathsf{m}^- \in M$ there is a net as in the lhs, for each positive $\mathsf{m}^+ \in M$ a net as in the rhs.

The contraction Petri structure only operates at the top of the exponential stack, leaving everything underneath unchanged. As the token passes through contractions $\mathbf{c}_M : !M \vdash !M \otimes !M$ from right to left, it accumulates injections $\ell_{\vdash}$ and $\nearrow_{\vdash}$ at the top of the exponential stack, which may then be used in the other direction to route back to the original resource – as in Figure 4.

2.3.3 Promotion. When interpreting $M\,N$, M may evaluate N several times, each time passing a different exponential signature. However, since we work compositionally, we have computed a structure $[\![N]\!]$ (intuitively realising a type A) that expects to be evaluated once, and we need to *promote* it to a structure $[\![N]\!]^{\dagger}$ (intuitively realising a type $!A$) that can be evaluated several times.

In traditional GoI, the promotion of N is obtained by wrapping N in a special delimiter called an *exponential box*, which marks the parts of the net being promoted. Special transitions handling exponential stacks are applied when tokens cross the borders of a box. Instead here we rewrite the transitions, leaving the net invariant but changing transition functions as follows:

Definition 2.11. Consider $\boldsymbol{\sigma} \in \mathsf{PStruct}(!M, N)$. We set $\mathcal{L}_{\boldsymbol{\sigma}^\dagger} = \mathcal{L}_{\boldsymbol{\sigma}}$, $\mathcal{T}_{\boldsymbol{\sigma}^\dagger} = \mathcal{T}_{\boldsymbol{\sigma}}$ with the same polarities, $\partial_{\boldsymbol{\sigma}^\dagger} = \partial_{\boldsymbol{\sigma}}$, and pre- and post-conditions are also unchanged. Finally, the *transition table* is:

$$
\begin{array}{lrcll}
(1) & \delta_{\boldsymbol{\sigma}^\dagger}\langle t^-\rangle(\mathsf{e} :: \mathsf{s}, \mathsf{d}) & = & \mathsf{e} :: \alpha & \text{if } \partial_{\boldsymbol{\sigma}}(t) = \imath_{\vdash}^{\leftarrow}- \text{ and } \delta_{\boldsymbol{\sigma}}\langle t\rangle(\mathsf{s}, \mathsf{d}) = \alpha \\
(2) & \delta_{\boldsymbol{\sigma}^\dagger}\langle t^0\rangle(\mathsf{e} :: \alpha) & = & \mathsf{e} :: \beta & \text{if } \delta_{\boldsymbol{\sigma}}\langle t\rangle(\alpha) = \beta \\
(3) & \delta_{\boldsymbol{\sigma}^\dagger}\langle t^+\rangle(\mathsf{e} :: \alpha) & = & (\mathsf{e} :: \mathsf{s}, \mathsf{d}) & \text{if } \partial_{\boldsymbol{\sigma}}(t) = \imath_{\vdash}^{\leftarrow}- \text{ and } \delta_{\boldsymbol{\sigma}}\langle t\rangle(\alpha) = (\mathsf{s}, \mathsf{d}) \\
(4) & \delta_{\boldsymbol{\sigma}^\dagger}\langle t^+\rangle(\mathsf{e} :: \alpha) & = & (\langle \mathsf{e}, \mathsf{e}'\rangle :: \mathsf{s}, \mathsf{d}) & \text{if } \partial_{\boldsymbol{\sigma}}(t) = \ell_{\vdash}- \text{ and } \delta_{\boldsymbol{\sigma}}\langle t\rangle(\alpha) = (\mathsf{e}' :: \mathsf{s}, \mathsf{d}) \\
(5) & \delta_{\boldsymbol{\sigma}^\dagger}\langle t^-\rangle(\langle \mathsf{e}, \mathsf{e}'\rangle :: \mathsf{s}, \mathsf{d}) & = & \mathsf{e} :: \alpha & \text{if } \partial_{\boldsymbol{\sigma}}(t) = \ell_{\vdash}- \text{ and } \delta_{\boldsymbol{\sigma}}\langle t\rangle(\mathsf{e}' :: \mathsf{s}, \mathsf{d}) = \alpha
\end{array}
$$

where $\partial_{\boldsymbol{\sigma}}(t) = \imath- $ means $\partial_{\boldsymbol{\sigma}}(t) = \imath\mathsf{m}$ for some $\mathsf{m} \in \mathcal{M}$, and $\mathsf{e} :: \alpha$ is $\{(\mathsf{e} :: \mathsf{s}_i, \mathsf{d}_i)^{@l_i} \mid (\mathsf{s}_i, \mathsf{d}_i)^{@l_i} \in \alpha\}$.

With this definition, we obtain $\boldsymbol{\sigma}^\dagger \in \mathsf{PStruct}(!M, !N)$.

In $\boldsymbol{\sigma}^\dagger$, messages on the right hand side from the environment come with an extra signature, which appears at the top of the exponential stacks in incoming tokens (equation 1). Neutral transitions only fire when all the pre-conditions have the same value for that signature but otherwise leave it invariant, in effect creating one independent copy of $\boldsymbol{\sigma}$ for each exponential signature (equation 2). Positive messages on the right hand side reply with the stored signature (equation 3).

The transition functions for the right hand side, if applied unchanged also on the left hand side, would intuitively yield a Petri structure on $!!M \vdash !N$, with one additional layer for the exponential stack on the left hand side. Instead, in order to obtain a Petri structure playing on $!M \vdash !N$, we apply the constructor $\langle -, - \rangle$ to pack (equation 4) or unpack (equation 5) two levels of the stack into one. This amounts to inlining the "digging" operation from linear logic.

This way of handling promotion without boxes is original to this work, and only possible thanks to the flexibility offered by Petri structures. With token machines on proof nets a box is needed, because the transitions only depend on the nature of the crossed node and have no way to take into account in how many exponential boxes we currently are – this impossibility is patched when crossing the box borders. In contrast, here we may directly "edit" the transition function when applying promotion. Of course the correcteness of this mechanism without boxes is subtle, it seems unclear how it could be proved via any other way than the unfolding to concurrent games.

2.3.4 Dereliction. The reader may wonder what primitive introduces the "leaves" $\blacklozenge$ of exponential signatures. For each $M \subseteq_f \mathcal{M}$ there is a Petri structure $\mathbf{der}_M : !M \vdash M$ that corresponds to a single resource usage – it is defined like $\mathbf{ac}_M$, except for the only two changed clauses

$$
\begin{array}{rcl}
\delta\langle(\mathsf{m}^-, \ell)\rangle(\{(\mathsf{s}, \mathsf{d})^{@\mathsf{m}}\}) & = & (\blacklozenge :: \mathsf{s}, \mathsf{d}) \\
\delta\langle(\mathsf{m}^+, \ell)\rangle(\blacklozenge :: \mathsf{s}, \mathsf{d}) & = & \{(\mathsf{s}, \mathsf{d})^{@\mathsf{m}}\}
\end{array}
$$

adding a removing the dummy exponential signature $\blacklozenge$ as a new layer of the stack on the lhs.

3 IPA AND ITS INTERPRETATION

Now that the backbone of Petri structures is set up, we define our language of study and examine the additional primitives on Petri structures required for its interpretation.

3.1 The language IPA

IPA is a higher-order call-by-name concurrent language with shared memory and semaphores, serving as a paradigmatic language for these features in the game semantics literature [Ghica and Murawski 2008]. Our variant is more expressive in some ways (in particular, it has a let construct); but it also has the restriction that reference and semaphore types should not appear at the right hand side of an arrow. See Section 3.1.4 for a discussion on this restriction.

$$\frac{}{\Gamma \vdash \mathsf{skip} : \mathbb{U}} \qquad \frac{}{\Gamma \vdash \mathsf{tt} : \mathbb{B}} \qquad \frac{}{\Gamma \vdash \mathsf{ff} : \mathbb{B}} \qquad \frac{}{\Gamma \vdash n : \mathbb{N}} \qquad \frac{}{\Gamma, x : A, \Delta \vdash x : A} \qquad \frac{\Gamma, x : A, \Delta \vdash M : O}{\Gamma, \Delta \vdash \lambda x^A.\, M : A \to O}$$

$$\frac{\Gamma \vdash M : O \to O}{\Gamma \vdash \mathsf{Y}\, M : O} \qquad \frac{\Gamma \vdash M : A \to O \qquad \Gamma \vdash N : A}{\Gamma \vdash M\, N : O} \qquad \frac{\Gamma \vdash M : \mathbb{S}}{\Gamma \vdash \mathsf{grab}\, M : \mathbb{U}} \qquad \frac{\Gamma \vdash M : \mathbb{B} \qquad \Gamma \vdash N_1 : \mathbb{X} \qquad \Gamma \vdash N_2 : \mathbb{X}}{\Gamma \vdash \mathsf{if}\, M\, N_1\, N_2 : \mathbb{X}}$$

$$\frac{\Gamma \vdash N : \mathbb{S}}{\Gamma \vdash \mathsf{rel}\, N : \mathbb{U}} \qquad \frac{\Gamma \vdash M : \mathbb{X} \qquad \Gamma \vdash N : \mathbb{Y}}{\Gamma \vdash \mathsf{f}(M, N) : \mathbb{Z}} \qquad \frac{\Gamma \vdash M : \mathbb{U} \qquad \Gamma \vdash N : \mathbb{X}}{\Gamma \vdash M \parallel N : \mathbb{X}} \qquad \frac{\Gamma, x : \mathbb{X}, \Delta \vdash M : \mathbb{Y} \qquad \Gamma, \Delta \vdash N : \mathbb{X}}{\Gamma, \Delta \vdash \mathsf{let}\, x = N\, \mathsf{in}\, M : \mathbb{Y}}$$

$$\frac{\Gamma, r : \mathbb{V}, \Delta \vdash M : \mathbb{X}}{\Gamma, \Delta \vdash \mathsf{newref}\, r\, \mathsf{in}\, M : \mathbb{X}} \qquad \frac{\Gamma \vdash M : \mathbb{V} \qquad \Gamma \vdash N : \mathbb{N}}{\Gamma \vdash M := N : \mathbb{U}} \qquad \frac{\Gamma \vdash M : \mathbb{V}}{\Gamma \vdash !M : \mathbb{N}} \qquad \frac{\Gamma, s : \mathbb{S}, \Delta \vdash M : \mathbb{X}}{\Gamma, \Delta \vdash \mathsf{newsem}\, s\, \mathsf{in}\, M : \mathbb{X}}$$

Fig. 19. Typing rules for IPA

3.1.1 Types and terms. The **types** of IPA are:

$$A, B, C ::= O \mid \mathbb{V} \mid \mathbb{S} \qquad\qquad O ::= \mathbb{U} \mid \mathbb{B} \mid \mathbb{N} \mid A \to O$$

where types generated by O are called **well-opened**. We have $\mathbb{U}$ a *unit* type, $\mathbb{B}$ and $\mathbb{N}$ respectively types for *booleans* and *natural numbers*, a type $\mathbb{V}$ for *integer references*, and $\mathbb{S}$ for *semaphores*. The split into *standard types* and *well-opened types* implements that $\mathbb{V}$ and $\mathbb{S}$ should not appear on the right of an arrow. We refer to $\mathbb{U}, \mathbb{B}$ and $\mathbb{N}$ as **ground types**, and use $\mathbb{X}, \mathbb{Y}, \mathbb{Z}$ to range over those.

We define terms directly via typing rules – throughout this paper, we only consider well-typed terms. **Contexts** are lists of typed variables $x_1 : A_1, \ldots, x_n : A_n$, where variables come from a fixed countable set Var. **Typing judgments** have the form $\Gamma \vdash M : A$, with Γ a context and A a type.

The typing rules appear in Figure 19; we give a few comments and clarifications. The construction $\mathsf{f}(M, N)$ applies to any computable $\mathsf{f} : \mathbb{X} \times \mathbb{Y} \rightharpoonup \mathbb{Z}$ (abusing notations to treat $\mathbb{X}, \mathbb{Y}, \mathbb{Z}$ as sets), intended to be evaluated left-to-right. This covers usual primitives: we define $M;\ N = \mathsf{seq}(M, N)$, with $\mathsf{seq} : \mathbb{X} \times \mathbb{Y} \to \mathbb{Y}$ the projection function. Basic arithmetic primitives are obtained similarly. Our asymmetric rule for parallel composition is unusual: we allow *one* of the threads to return a value of ground type, which makes examples shorter without significantly affecting the language. Conditionals eliminate to ground type, but as usual in call-by-name, a more general conditional can be derived. We refer to constants of ground type as **values**; we use v to range over values of any type, and n, b or c to range over values of respective types $\mathbb{N}, \mathbb{B}$ or $\mathbb{U}$.

3.1.2 Examples. We saw in (4) a derived non-deterministic primitive. Another simple program is

$$x : \mathbb{U}, y : \mathbb{X} \vdash \mathsf{newsem}\, s\, \mathsf{in}\, \mathsf{grab}\, s;\ (x;\ \mathsf{rel}\, s \parallel \mathsf{grab}\, s;\ y) : \mathbb{X},$$

illustrating the use of semaphores. A semaphore has two state: *free*, or *locked*. The command $\mathsf{grab}\, s$ attempts to grab a semaphore s. If s is free, then $\mathsf{grab}\, s$ terminates successfully and makes it locked. If s is already locked, then the instruction $\mathsf{grab}\, s$ waits until s becomes free. The command rel (for "release") is symmetric. Thus in the example above, the second thread is stuck until x has successfully completed, triggering the release of s, authorizing the $\mathsf{grab}\, s$ and giving the green light to y. Altogether, the program behaves exactly like sequential composition.

As a final example, IPA allows dynamic creation of references and semaphores: for instance,

$$\vdash \quad (\lambda x.\, x + x)\ (\mathsf{newref}\, r\, \mathsf{in}\, r := !r + 1; !r) : \mathbb{N}$$

returns $1 + 1 = 2$, because execution causes the initialization of two independent references. An unbounded number of references can arise in this way if this happens within recursion.

Though IPA is a toy language, it is semantically highly non-trivial, raising realistic challenges.

3.1.3 Operational semantics. The reference semantics for IPA is a small-step interleaving operational semantics following closely that of [Ghica and Murawski 2008]. We fix a countable set $\mathcal{L}$ of **memory locations**. A **store** is $s : \mathcal{L} \rightharpoonup \mathbb{N}$ with finite domain where $\mathbb{N}$ stands, overloading notations, for natural numbers. **Configurations** are $\langle M, s\rangle$ with s a store, $\mathsf{dom}(s) = \{\ell_1, \ldots, \ell_n\}$ and $\Sigma \vdash M : A$ with $\Sigma = \ell_1 : \mathbb{V}, \ldots, \ell_i : \mathbb{V}, \ell_{i+1} : \mathbb{S}, \ldots, \ell_n : \mathbb{S}$. Reduction rules have the form $\langle M, s\rangle \rightsquigarrow \langle M', s'\rangle$ where $\mathsf{dom}(s) = \mathsf{dom}(s')$; we write $\rightsquigarrow^*$ for the reflexive transitive closure. If $\vdash M : \mathbb{X}$, we write $M \Downarrow$ if $\langle M, \emptyset\rangle \rightsquigarrow^* \langle v, \emptyset\rangle$ for some value v. Then M **converges**, else it **diverges**.

The detailed rules, essentially as in [Ghica and Murawski 2008], are postponed to Appendix A as they will be referred to only indirectly in this paper (via Theorem 4.5).

3.1.4 On the restriction on types. Our version of IPA is restricted to the effect that reference and semaphore types cannot occur on the right hand side of an arrow, *i.e.* types such as $\mathbb{B} \to \mathbb{V}$ are forbidden. This is because $\mathbb{V}$ is semantically treated like a product $\mathbb{V}_w \times \mathbb{V}_r$ of a write-only and a read-only reference – and while products can be curryied away when on the left hand side of an arrow, that is not the case for products on the right hand side. Dealing with such products is subtle for reasons already suggested by the call-by-name isomorphism of types $A \to (B \times C) \cong (A \to B) \times (A \to C)$: the product morally generates *two* copies of A, and tokens must carry additional data indexing those copies. Similar difficulties arise when dealing with *additives* in traditional GoI [Girard 1995] – it can be done, but with the price of additional complications.

We opted for a version of IPA without (implicit) products, because *(1)* it allows us to emphasize the important concepts of our approach, rather than obfuscating them with additional structure for additives; *(2)* we believe that functions returning variables or semaphores is a rare pattern in functional programming. In any case, this restricted IPA is sufficient as a proof of concept language – it already has all the behaviours that IPA is designed to study: races, interactions between higher-order and state or semaphores, etc; *(3)* this is an artefact of products in call-by-name. Our next step is to tackle more realistic programming languages, but those will be in call-by-value. The issue does not arise in call-by-value, as the isomorphism of types above disappears.

3.2 IPA-structures

3.2.1 Definition. What additional data should a precategory like PStruct have, so as to support an interpretation of IPA following the methodology of denotational semantics? We call our proposed answer IPA-**structures**: those start with the operations available in a categorical model of intuitionistic linear logic, to which we add primitives maching IPA (with no equations required):

Definition 3.1. An IPA-**structure** is a precategory C, with a distinguished set $C_\bullet \subseteq C_0$ of **well-opened** objects – we use A, B, C to range over all objects and O to range over $C_\bullet$ – and:

- **Constructions.** We have objects $\mathbf{U}, \mathbf{B}, \mathbf{N} \in C_\bullet$, $\mathbf{V}, \mathbf{S}, 1 \in C_0$, and:

$$\begin{array}{rl} \textit{tensor:} & \text{for } A, B \in C_0, \text{ there is } A \otimes B \in C_0, \\ \textit{product:} & \text{for } (A_x)_{x\in V} \text{ with } V \subseteq_f \mathsf{Var}, \&_{x\in V} A_x \in C_0, \\ \textit{arrow:} & \text{for } A \in C_0 \text{ and } O \in C_\bullet, \text{ there is } A \multimap O \in C_\bullet, \\ \textit{bang:} & \text{for } A \in C_0, \text{ there is } !A \in C_0, \end{array}$$

  with $\&\emptyset = \top$ the product of the empty family.
- **Operations.** We have constructions on morphisms:

$$\begin{array}{rcrcl} \otimes & : & C(A_1, B_1) \times C(A_2, B_2) & \to & C(A_1 \otimes A_2, B_1 \otimes B_2) \\ \Lambda^{\Gamma,\Delta}_{x:A,O} & : & C(!(\&[\Gamma, x : A, \Delta]), O) & \to & C(!(\&[\Gamma, \Delta]), !A \multimap O) \\ (-)^\dagger & : & C(!\Gamma, O) & \to & C(!\Gamma, !O) \end{array}$$

  where, if $\Gamma = x_1 : A_1, \ldots, x_n : A_n$ where $A_i \in C_0$ for all i, we write $[\Gamma]$ for $(A_x)_{x\in\{x_1,\ldots,x_n\}}$;

$$
\begin{array}{rcl}
[\![\Gamma \vdash \mathsf{skip} : \mathbb{U}]\!] & = & \mathbf{skip} \circ \mathbf{w}_\Gamma \\
[\![\Gamma \vdash \mathsf{tt} : \mathbb{B}]\!] & = & \mathbf{tt} \circ \mathbf{w}_\Gamma \\
[\![\Gamma \vdash \mathsf{ff} : \mathbb{B}]\!] & = & \mathbf{ff} \circ \mathbf{w}_\Gamma \\
[\![\Gamma, x : A, \Delta \vdash x : A]\!] & = & \mathbf{var}^{\Gamma,\Delta}_{x:A} \\
[\![\Gamma, \Delta \vdash \lambda x^A.\, M : A \to O]\!] & = & \Lambda^{\Gamma,\Delta}_{x:A,O}([\![M]\!]) \\
[\![\Gamma \vdash M\, N : O]\!] & = & \mathbf{ev}_{!A,O} \circ ([\![M]\!] \otimes [\![N]\!]^\dagger) \circ \mathbf{c}_\Gamma \\
[\![\Gamma \vdash \mathsf{Y}\, M : O]\!] & = & \mathbf{Y}_O \circ [\![M]\!]^\dagger \\
[\![\Gamma \vdash \mathsf{if}\, M\, N_1\, N_2 : \mathbb{X}]\!] & = & \mathbf{if}_\mathbf{X} \circ ([\![M]\!] \otimes ([\![N_1]\!] \otimes [\![N_2]\!])) \circ \mathbf{c}^3_\Gamma \\
[\![\Gamma \vdash \mathsf{f}(M, N) : \mathbb{Z}]\!] & = & \mathbf{op}(\mathsf{f})^{\mathbf{X},\mathbf{Y}}_\mathbf{Z} \circ ([\![M]\!] \otimes [\![N]\!]) \circ \mathbf{c}_\Gamma \\
[\![\Gamma \vdash M \parallel N : \mathbb{X}]\!] & = & \mathbf{par}_\mathbf{X} \circ ([\![M]\!] \otimes [\![N]\!]) \circ \mathbf{c}_\Gamma \\
[\![\Gamma, \Delta \vdash \mathsf{let}\, x = N \,\mathsf{in}\, M : \mathbb{Y}]\!] & = & \mathbf{let}_{\mathbf{X},\mathbf{Y}} \circ (\Lambda^{\Gamma,\Delta}_{\mathbf{X},\mathbf{Y}}([\![M]\!]) \otimes [\![N]\!]) \circ \mathbf{c}_{\Gamma,\Delta} \\
[\![\Gamma, \Delta \vdash \mathsf{newref}\, x \,\mathsf{in}\, M : \mathbb{X}]\!] & = & \mathbf{newref}_\mathbf{X} \circ \Lambda^{\Gamma,\Delta}_{\mathbf{V},\mathbf{X}}([\![M]\!]) \\
[\![\Gamma \vdash M := N : \mathbb{U}]\!] & = & \mathbf{assign} \circ ([\![M]\!] \otimes [\![N]\!]) \circ \mathbf{c}_\Gamma \\
[\![\Gamma \vdash !M : \mathbb{N}]\!] & = & \mathbf{deref} \circ [\![M]\!] \\
[\![\Gamma, \Delta \vdash \mathsf{newsem}\, x \,\mathsf{in}\, M : \mathbb{X}]\!] & = & \mathbf{newsem}_\mathbf{X} \circ \Lambda^{\Gamma,\Delta}_{\mathbf{V},\mathbf{X}}([\![M]\!]) \\
[\![\Gamma \vdash \mathsf{grab}\, M : \mathbb{U}]\!] & = & \mathbf{grab} \circ [\![M]\!] \\
[\![\Gamma \vdash \mathsf{rel}\, N : \mathbb{U}]\!] & = & \mathbf{release} \circ [\![N]\!]
\end{array}
$$

Fig. 20. Interpretation of IPA in an IPA-structure

$$
\begin{array}{rcl}
\mathbf{var}^{\Gamma,\Delta}_{x:A} & \in & \mathcal{C}(!(\&[\Gamma, x : A, \Delta]), A) \\
\mathbf{ev}_{A,O} & \in & \mathcal{C}((A \multimap O) \otimes A, O) \\
\mathbf{c}^n_\Gamma & \in & \mathcal{C}(!\Gamma, \otimes^n(!\Gamma)) \\
\mathbf{skip} & \in & \mathcal{C}(1, \mathbf{U}) \\
\mathbf{tt} & \in & \mathcal{C}(1, \mathbf{B}) \\
\mathbf{ff} & \in & \mathcal{C}(1, \mathbf{B}) \\
\mathbf{n} & \in & \mathcal{C}(1, \mathbf{N}) \\
\mathbf{Y}_O & \in & \mathcal{C}(!(!O \multimap O), O) \\
\mathbf{if}_\mathbf{X} & \in & \mathcal{C}(\mathbf{B} \otimes (\mathbf{X} \otimes \mathbf{X}), \mathbf{X}) \\
\mathbf{op}(\mathsf{f})^{\mathbf{X},\mathbf{Y}}_\mathbf{Z} & \in & \mathcal{C}(\mathbf{X} \otimes \mathbf{Y}, \mathbf{Z}) \\
\mathbf{par}_\mathbb{X} & \in & \mathcal{C}(\mathbf{U} \otimes \mathbf{X}, \mathbf{X}) \\
\mathbf{let}_{\mathbf{X},\mathbf{Y}} & \in & \mathcal{C}((!\mathbf{X} \multimap \mathbf{Y}) \otimes \mathbf{X}, \mathbf{Y}) \\
\mathbf{newref}_\mathbf{X} & \in & \mathcal{C}(!\mathbf{V} \multimap \mathbf{X}, \mathbf{X}) \\
\mathbf{newsem}_\mathbf{X} & \in & \mathcal{C}(!\mathbf{S} \multimap \mathbf{X}, \mathbf{X}) \\
\mathbf{assign} & \in & \mathcal{C}(\mathbf{V} \otimes \mathbf{N}, \mathbf{U}) \\
\mathbf{deref} & \in & \mathcal{C}(\mathbf{V}, \mathbf{N}) \\
\mathbf{grab} & \in & \mathcal{C}(\mathbf{S}, \mathbf{U}) \\
\mathbf{release} & \in & \mathcal{C}(\mathbf{S}, \mathbf{U})
\end{array}
$$

Fig. 21. Primitives of IPA-structures

- **Primitives.** As listed in Figure 21, where $\otimes^0 A = 1$, $\otimes^1 A = A$, $\otimes^{n+2} A = A \otimes (\otimes^{n+1} A)$, and writing $\mathbf{w}_A \in \mathcal{C}(!A, 1)$ for $\mathbf{c}^0_A$ and $\mathbf{c}_A \in \mathcal{C}(!A, !A \otimes !A)$ for $\mathbf{c}^2_A$.

3.2.2 Interpretation of IPA. The interpretation follows the standard lines of the interpretation of call-by-name languages into models of intuitionistic linear logic. Fix $\mathcal{C}$ an IPA-structure.

We interpret the *types* of IPA as objects of $\mathcal{C}$, with $[\![\mathbb{U}]\!] = \mathbf{U}$, $[\![\mathbb{B}]\!] = \mathbf{B}$, $[\![\mathbb{N}]\!] = \mathbf{N}$, $[\![\mathbb{V}]\!] = \mathbf{V}$, $[\![\mathbb{S}]\!] = \mathbf{S}$, and finally, $[\![A \to B]\!] = ![\![A]\!] \multimap [\![B]\!]$. Note that well-opened types are mapped to well-opened objects. *Contexts* are also interpreted as objects of $\mathcal{C}$, with $[\![x_1 : A_1, \ldots, x_n : A_n]\!] = \&_{x_i \in \{x_1,\ldots,x_n\}} [\![A_i]\!]$.

To any *typed term* $\Gamma \vdash M : A$ we associate as interpretation a morphism $[\![\Gamma \vdash M : A]\!] \in \mathcal{C}(![\![\Gamma]\!], [\![A]\!])$ sometimes shortened to $[\![M]\!]$, following the clauses of Figure 20. Thus if we can equip PStruct with this additional structure, we will automatically obtain following the definition above

$$[\![-]\!] : \mathsf{IPA} \to \mathsf{PStruct}$$

a translation sending any IPA program, possibly open and of higher type, to a Petri structure.

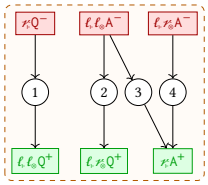
3.2.3 Relating Interpretations. Recall that at the heart of our methodology for this paper is the diagram (2) with, in particular, operations $\mathcal{U} : \mathsf{PStrat} \to \mathsf{Strat}$ and $\mathcal{F} : \mathsf{PStrat} \to \mathsf{PStruct}$ commuting with the interpretation. As our interpretations shall all follow the interpretation of IPA in an IPA-structure detailed above, interpretation-preserving operations shall be obtained by:

Definition 3.2. Consider $\mathcal{C}$ and $\mathcal{D}$ two IPA-structures, and $F : \mathcal{C} \to \mathcal{D}$ a functor.
Then, F is a **(strict)** IPA-**functor** iff it preserves all the data of IPA-structures on the nose.

It follows immediately by induction that the interpretation of types and terms is preserved:

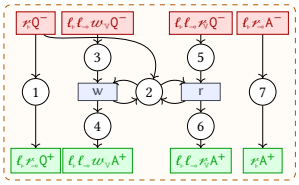
Lemma 3.3. *Consider $\mathcal{C}, \mathcal{D}$ two* IPA-*structures, and $F : \mathcal{C} \to \mathcal{D}$ an* IPA-*functor.*
Then, for any $\Gamma \vdash M : A$, we have $F([\![A]\!]_\mathcal{C}) = [\![A]\!]_\mathcal{D}$, $F([\![\Gamma]\!]_\mathcal{C}) = [\![\Gamma]\!]_\mathcal{D}$, and $F([\![M]\!]_\mathcal{C}) = [\![M]\!]_\mathcal{D}$.

To sum up: IPA-*structures* list the ingredients necessary to give an interpretation to all terms of IPA, and IPA-*functors* list the proof obligations in order to relate two interpretations. In particular, PStruct, PStrat and Strat from (2) will be IPA-structures, and $\mathcal{F}$, $\mathcal{U}$ will be IPA-functors.



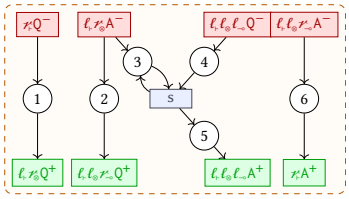
$$\begin{aligned}\delta_{\mathbf{op}(f)}\langle \ell_{\vdash} r_{\otimes} A^{-}\rangle([],d) &= \{([],\bullet)^{@2},([],d)^{@3}\}\\ \delta_{\mathbf{op}(f)}\langle r_{\vdash} A^{+}\rangle(\{([],d)^{@3},([],d')^{@4}\}) &= ([],f(d,d'))\end{aligned}$$

Fig. 22. Operators



$$\begin{aligned}\delta_{\mathbf{newref}}\langle r_{\vdash} Q^{-}\rangle([],\bullet) &= \{([],\bullet)^{@1},([],0)^{@2}\}\\ \delta_{\mathbf{newref}}\langle w\rangle(\{([e],d)^{@3},(_,_)^{@2}\}) &= \{([e],d)^{@2},([e],\checkmark)^{@4}\}\\ \delta_{\mathbf{newref}}\langle r\rangle(\{([e],\bullet)^{@5},(_,d)^{@2}\}) &= \{([e],d)^{@2},([e],d)^{@6}\}\end{aligned}$$

Fig. 23. New reference



$$\begin{aligned}\delta_{\mathbf{let}}\langle \ell_{\vdash} r_{\otimes} A^{-}\rangle([],d) &= \{([],d)^{@2},([],d)^{@3}\}\\ \delta_{\mathbf{let}}\langle s\rangle(\{([],d)^{@3},(s,\bullet)^{@4}\}) &= \{([],d)^{@3},(s,d)^{@5}\}\end{aligned}$$

Fig. 24. Let binding

3.3 PStruct as an IPA-structure

Completing the precategory PStruct into a full IPA-structure requires providing nets for a number of primitives, most of which are straightforward and detailed in Appendix B. Here we focus on three constructions which we consider more informative or noteworthy.

3.3.1 Operators. For $\mathbf{op}(f)^{X,Y}_{Z} \in \mathrm{PStruct}(\mathbf{X}\otimes\mathbf{Y},\mathbf{Z})$, the net and its transitions are shown in Figure 22. By convention we omit the transitions that are **trivial**, in the sense that they merely forward tokens without affecting them (the typing specifying uniquely where these tokens should go).

Upon initialization by receiving $([],\bullet)$ on $r_{\vdash} Q^{-}$, the net first puts $([],\bullet)$ in location 1 triggering the evaluation request for the first argument, *i.e.* sending $([],\bullet)$ on $\ell_{\vdash}\ell_{\otimes} Q^{+}$. When the first argument returns a value with a message on $\ell_{\vdash}\ell_{\otimes} A^{-}$, the net stores the value in location 3, and puts a token in location 2 to evaluate the second argument. Once the value is received on $\ell_{\vdash} r_{\otimes} A^{-}$, the pre-conditions of $r_{\vdash} A^{+}$ are met and the net emits the final result, applying f on the two stored values. Here we see that our machine may use several tokens during computation even for sequential programs: locations are used in the style of the GoI token machines to track the control flow, but also as a way to store intermediate results throughout computation.

3.3.2 New reference. For $\mathbf{newref}_{X} \in \mathrm{PStruct}(!\mathbf{V} \multimap \mathbf{X},\mathbf{X})$, the net and its non-trivial transitions are shown in Figure 23. The Petri structure **newref** takes as an argument $!\mathbf{V} \multimap \mathbf{X}$, *i.e.* a "program of type $\mathbf{X}$" that may query a reference. Location 2 stores the current value. Upon initialization with $r_{\vdash} Q^{-}$, **newref**: *(1)* initializes the reference, setting a token with data 0 in location 2; *(2)* in parallel, gives control to the argument via $\ell_{\vdash} r_{\multimap} Q^{+}$. *Write* and *read* requests arrive respectively in locations 3 and 5. The neutral transitions w and r perform the memory update and reading. Normally, *i.e.* in the context of the evaluation of a closed program, there is ever at most one token in location 2 (if the net is under a promotion there may be several, kept apart by their exponential stack), forcing reads and writes to be handled in some sequential order. Note that r overwrites the token in the store even through it does not change the value: this is essential in order to get the unfolding right.

3.3.3 Let binding. For $\mathbf{let}_{X,Y} \in C((!\mathbf{X} \multimap \mathbf{Y})\otimes\mathbf{X},\mathbf{Y})$, the net and its non-trivial transitions are shown in Figure 24. At first sight, **let** is only a write-once reference, with value stored in location 3. But we have seen that in the net for newref, read updates overwrite the token storing the current value. Doing the same for the let binding would sequentialize all reads, which is fine in terms of interleavings but would break our causal unfolding as it is *not* compatible with the expected causal behaviour of the let binding. So instead, transition r reads the stored value with token $([],d)$, and *puts back the exact same token*, so that further accesses to the value stored do not depend on it.

See Appendix B for the other constructions, which yield:

Corollary 3.4. PStruct *is an* IPA-*structure.*

Following Section 3.2.2 we obtain for any term $\Gamma \vdash M : A$ a net $[\![M]\!] \in \mathsf{PStruct}(![\![\Gamma]\!], [\![A]\!])$ regarded as the token machine with M loaded. In particular, a closed program $\vdash M : \mathbb{X}$ yields a Petri structure $[\![M]\!]$ on $\{\mathscr{r}_{\mathsf{f}}\mathsf{Q}^-, \mathscr{r}_{\mathsf{f}}\mathsf{A}^+\}$ which we can execute by playing the token game.

3.4 The Token Game and Adequacy

In order to reason formally on the execution of Petri structures, we now formalize the token game. Its states are called *markings*, in which each location contains a finite set of tokens.

Definition 3.5. Consider $\boldsymbol{\sigma}$ a Petri structure. A **marking** on $\boldsymbol{\sigma}$ is a finite subset of TokIL.
The set of markings on $\boldsymbol{\sigma}$ is written $\mathscr{M}(\boldsymbol{\sigma})$, ranged over by α, β, γ.

We use the same notation for markings as for conditions, *i.e.* $\alpha = \{([], \bullet)^{@1}, ([], \bullet)^{@2}, ([\blacklozenge], \checkmark)^{@2}\}$ is a (non-reachable) marking for the Petri structure of Figure 11. Unlike for conditions, markings allow several tokens on the same location; however we cannot have the *same* token twice on the same location – markings are sets, not multisets. It should be clear from this notation that conditions may be regarded as markings, and we shall do so silently from now on.

The *token game* is a walk on a labelled transition system of markings. While neutral transitions act on markings only, visible ones *send* or *receive* tokens on addresses. Together, an address and a token form a **move** – we set $\mathsf{Moves} = \mathcal{M} \times \mathcal{E}^* \times \mathcal{D} \simeq \mathcal{M} \times \mathsf{Tok}$, ranged over by m. The transition system is defined in two steps. First, *events* are transitions instantiated with specific input:

Definition 3.6. An **event** of a Petri structure $\boldsymbol{\sigma}$ is a pair (t, ι) written $t(\!|\iota|\!)$ where ι is an input of t, i.e. a token (s, d) if t is negative or some $\alpha \in \mathsf{cond}(\mathsf{pre}(t))$ otherwise. We write:

$$\begin{array}{rcccll} t^0(\!|\alpha|\!) & : & \alpha & \longmapsto_{\boldsymbol{\sigma}} & \beta & \text{if } \delta\langle t\rangle(\alpha) = \beta, \\ t^+(\!|\alpha|\!) & : & \alpha & \xmapsto{m}_{\boldsymbol{\sigma}} & \emptyset & \text{if } \delta\langle t\rangle(\alpha) = (\mathsf{s}, \mathsf{d}), \\ t^-(\!|(\mathsf{s}, \mathsf{d})|\!) & : & \emptyset & \xmapsto{m}_{\boldsymbol{\sigma}} & \beta & \text{if } \delta\langle t\rangle(\mathsf{s}, \mathsf{d}) = \beta, \end{array}$$

where $m = (\partial_{\boldsymbol{\sigma}}(t), \mathsf{s}, \mathsf{d})$ is the *move* played. We write $\mathscr{E}(\boldsymbol{\sigma})$ the set of events of $\boldsymbol{\sigma}$, ranged over by $\mathbf{e}$.

Events inherit from transitions a polarity. A visible $\mathbf{e}$ plays a move m, written $m = \partial_{\boldsymbol{\sigma}}(\mathbf{e}) = (\mathsf{m}, \mathsf{s}, \mathsf{d})$. For negative $\mathbf{e}$ the token (s, d) is *received* on m, while for positive $\mathbf{e}$, (s, d) is *sent* on m.

Events only mention the tokils involved in the transition, *i.e.* mentioned in the transition function. To obtain the actual token game, one must allow events to occur in the presence of other tokils sitting in the net, not playing any role in the transition. This is captured by the notion of *firings*:

Definition 3.7. The set $\mathscr{F}(\boldsymbol{\sigma})$ of **firings**, ranged over by $\mathbf{f}$, comprises

$$\begin{array}{rcccll} \mathbf{e}^0 \uplus \gamma & : & \alpha \uplus \gamma & \longrightarrow_{\boldsymbol{\sigma}} & \beta \uplus \gamma & \text{if } \mathbf{e} : \alpha \longmapsto_{\boldsymbol{\sigma}} \beta, \\ \mathbf{e}^+ \uplus \gamma & : & \alpha \uplus \gamma & \xrightarrow{m}_{\boldsymbol{\sigma}} & \gamma & \text{if } \mathbf{e} : \alpha \xmapsto{m}_{\boldsymbol{\sigma}} \emptyset, \\ \mathbf{e}^- \uplus \gamma & : & \gamma & \xrightarrow{m}_{\boldsymbol{\sigma}} & \gamma \uplus \beta & \text{if } \mathbf{e} : \emptyset \xmapsto{m}_{\boldsymbol{\sigma}} \beta. \end{array}$$

for $\gamma \in \mathscr{M}(\boldsymbol{\sigma})$ with $\gamma \cap \alpha = \gamma \cap \beta = \emptyset$.

Note the disjointness assumption, which means a transition can only fire if the tokils it emits do not collide with tokils already present. Now we can finally capture the token game with:

Definition 3.8. A **run** in $\boldsymbol{\sigma}$ is a sequence $\rho = \mathbf{f}_1 \ldots \mathbf{f}_n$ of firings s.t. $\mathbf{f}_i : \alpha_i \longrightarrow_{\boldsymbol{\sigma}} \alpha_{i+1}$ or $\mathbf{f}_i : \alpha_i \xrightarrow{m}_{\boldsymbol{\sigma}} \alpha_{i+1}$ with $\alpha_1 = \emptyset$. We write $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\sigma}} \alpha_{n+1}$ or $\rho : \emptyset \xrightarrow{s}\!\!\!\!\rightarrow_{\boldsymbol{\sigma}} \alpha_{n+1}$, where $s = m_1 \ldots m_p$ lists the moves of visible firings in ρ. We also write $s = \mathsf{play}(\rho)$ and call s the **play** of ρ.

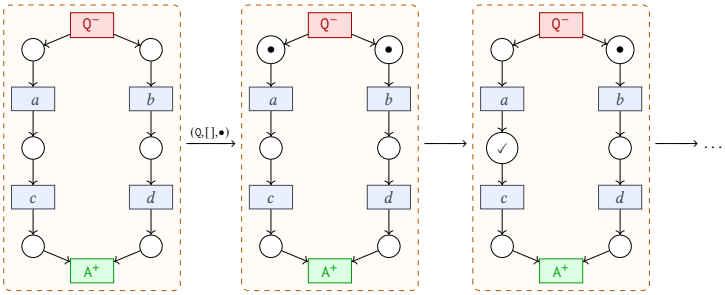


Fig. 25. A run of a Petri structure

For example, (the variant with full tokens of) Figure 11 has

$$\begin{array}{lllll} \emptyset & \xrightarrow{(\mathtt{Q},[],\bullet)} & \{([],\bullet)^{@1},([],\bullet)^{@2}\} & \longrightarrow & \{([],\checkmark)^{@3},([],\bullet)^{@2}\} \\ & \longrightarrow & \{([],\checkmark)^{@5},([],\bullet)^{@2}\} & \longrightarrow & \{([],\checkmark)^{@5},([],\checkmark)^{@4}\} \\ & \longrightarrow & \{([],\checkmark)^{@5},([],\checkmark)^{@6}\} & \xrightarrow{(\mathtt{A},[],\checkmark)} & \emptyset \end{array}$$

as a run with $\mathsf{play}(\rho) = (\mathtt{Q}, [], \bullet)(\mathtt{A}, [], \checkmark)$. We read it as tokens walking through the Petri net as in Figure 25 (omitting exponential stacks, which are always [] in this example).

We define *may-convergence* for Petri structures on $\{\not{r}_{\vdash}\mathtt{Q}^-, \not{r}_{\vdash}\mathtt{A}^+\}$:

Definition 3.9. Consider $\boldsymbol{\sigma}$ a Petri structure on $\{\not{r}_{\vdash}\mathtt{Q}^-, \not{r}_{\vdash}\mathtt{A}^+\}$. We say $\boldsymbol{\sigma}$ **may converge**, written $\boldsymbol{\sigma} \Downarrow$, iff there is a run $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\sigma}} \alpha$ such that $\mathsf{play}(\rho) = (\not{r}_{\vdash}\mathtt{Q}, [], \bullet)(\not{r}_{\vdash}\mathtt{A}, [], \mathtt{d})$ for some $\mathtt{d} \neq \bullet$.

Now that this is set up, we can state formally one of the main results of this paper:

Theorem 3.10 (Adequacy). *For any* $\vdash M : \mathbb{U}$, $M \Downarrow$ *iff* $[\![M]\!] \Downarrow$.

This is the standard way of stating that at ground type and with respect to may-convergence, the token machine is faithful to standard operational semantics. But here adequacy shall follow from a much more powerful result, also giving a proper account of higher-order: we shall prove that the Petri structure $[\![M]\!]$ *unfolds* to the concurrent strategy for M.

4 UNFOLDING AND CORRECTNESS

We link to the *denotational* model of IPA presented in [Castellan and Clairambault 2020], where both types and programs are interpreted as *event structures*, called *games* and *strategies* respectively.

In Section 4.1, we recall basic definitions about this model, yielding Strat, an adequate IPA-structure. We wish to define the unfolding as an IPA-functor from PStruct to Strat, but in general the behaviour of Petri structures is too wild for this. So our following step, performed in Section 4.2, is to tame PStruct, eliminating those nets whose behaviour fails to respect a protocol (the game) specified by the type – this leads to another IPA-structure PStrat of *Petri strategies*. In Section 4.3, we define a causal reconstruction mapping Petri strategies to strategies, i.e. event structures.

4.1 Concurrent Strategies for IPA

4.1.1 Types as Games. In PStruct, a type is merely interpreted by a finite set of addresses. This forgets information: *(1)* the possible values, for instance $\mathbb{B}$ and $\mathbb{N}$ have the same interpretation; *(2)* the dependency from function calls to returns, and the dependency arising from the scope: in $(\mathbb{N} \to \mathbb{B})$ the natural number should not be interrogated before the boolean. Adding this information yields a *game*, *i.e.* an event structure with *moves* as events. Recall first:

Definition 4.1. An **event structure (es)** is $E = (|E|, \leq_E, \#_E)$, where $|E|$ is a (countable) set of **events**, $\leq_E$ is a partial order and $\#_E$ is an irreflexive symmetric binary relation, satisfying:

$$\begin{array}{rl} \textit{finite causes}: & \forall e \in |E|,\ [e]_E \text{ is finite} \\ \textit{conflict inheritance}: & \forall e_1 \#_E e_2,\ \forall e_2 \leq_E e_2',\ e_1 \#_E e_2', \end{array}$$

where $[e]_E = \{e' \in |E| \mid e' \leq_E e\} - \#_E$ and $\leq_E$ are respectively **conflict** and **causal dependency**.

We write $\rightarrowtriangle_E$ for the immediate dependency, *i.e.* $e \rightarrowtriangle_E e'$ iff $e <_E e'$ with no event strictly in between. An es E comes with a notion of *state*, its (finite) **configurations**: those are finite $x \subseteq_f |E|$ down-closed for $\leq_E$ and compatible, *i.e.* if $e, e' \in x$, then $\neg(e \#_E e')$. We write $\mathscr{C}(E)$ the configurations of E. If $x \in \mathscr{C}(E)$ and $e \notin x$ with $x \cup \{e\} \in \mathscr{C}(E)$, we say x **enables** e, written $x \vdash_E e$.

Definition 4.2. Consider $A = (|A|, \leq_A, \#_A)$ an es with $|A| \subseteq \mathsf{Moves}$, and write $\mathsf{mult}(A) = \{\mathsf{m} \in \mathcal{M} \mid \exists(\mathsf{m}, \mathsf{s}, \mathsf{d}) \in |A|\}$. Then, A is a **game** if it satisfies the additional axiom:

$$\textit{finite addresses:} \quad \text{the set } \mathsf{mult}(A) \text{ is finite.}$$

This is the usual notion of concurrent game, except that moves are taken in Moves and hence the *polarity function* $\mathsf{pol}_A : |A| \to \{-, +\}$ is inherited from Moves and no longer part of the data.

We shall need the more specific *arenas*, capturing the causal patterns arising from types:

Definition 4.3. An **arena** is a game A satisfying:

$$\begin{array}{rl} \textit{alternating:} & \text{if } a_1 \rightarrowtriangle_A a_2, \text{ then } \mathsf{pol}(a_1) \neq \mathsf{pol}(a_2), \\ \textit{forestial:} & \text{if } a_1 \leq_A a \text{ and } a_2 \leq_A a, \text{ then } a_1 \leq_A a_2 \text{ or } a_2 \leq_A a_1, \\ \textit{local conflict:} & \text{if } a_1, a_2 \in |A| \text{ are in minimal conflict, then } [a_1)_A = [a_2)_A, \\ \textit{negative:} & \text{if } a \in \min(A), \text{ then } \mathsf{pol}(a) = -, \end{array}$$

where $\min(A)$ is the set of minimal events of A. Finally, A is **well-opened** if $\min(A)$ is a singleton.

This used the **strict dependencies** $[a)_A = \{a' \in A \mid a' <_A a\}$, and the notion of *minimal conflict*: in an event structure E, $e_1, e_2 \in |E|$ are in **minimal conflict** if $e_1 \#_E e_2$, while if $e_1' \leq_E e_1$ and $e_2' \leq_E e_2$ with at least one of these being strict, then $\neg(e_1' \#_E e_2')$ – so the conflict is not inherited.

We omit the interpretation of types and contexts as arenas, see Appendix C.1.

4.1.2 Strategies. In concurrent games, strategies are also event structures:

Definition 4.4. A **prestrategy** $\sigma : A$ on game A comprises an es $(|\sigma|, \leq_\sigma, \#_\sigma)$ with $\partial : |\sigma| \to |A|$ a function called the **display map**, subject to the following conditions:

$$\begin{array}{rl} \textit{rule-abiding:} & \text{if } x \in \mathscr{C}(\sigma), \text{ then } \partial(x) \in \mathscr{C}(A), \\ \textit{locally injective:} & \text{if } s_1, s_2 \in x \in \mathscr{C}(\sigma),\ \partial(s_1) = \partial(s_2), \text{ then } s_1 = s_2. \end{array}$$

We say that σ is a **strategy** if it satisfies the further two:

$$\begin{array}{rl} \textit{courteous:} & \text{for all } s_1 \rightarrowtriangle_\sigma s_2, \text{ if } \mathsf{pol}(s_1) = + \text{ or } \mathsf{pol}(s_2) = -, \text{ then } \partial(s_1) \rightarrowtriangle_A \partial(s_2), \\ \textit{receptive:} & \text{for all } x \in \mathscr{C}(\sigma), \text{ for all } \partial(x) \vdash_A a^-, \text{ there is a unique } x \vdash_\sigma s \in \mathscr{C}(\sigma) \text{ s.t. } \partial(s) = a, \end{array}$$

and is **negative** if any $s \in \min(\sigma)$ is negative.

Rule-abiding and *locally injective* together amount to $\partial : \sigma \to A$ being a **map of event structures**. The event structure σ presents observable computational events along with their causal dependencies and conflicts. Events of σ are *not* moves of the game, but they do correspond to moves via the action of ∂. This permits an explicit representation of non-deterministic branching: several events of σ may correspond to the same move if they are conflicting and so belong to separate branches of the execution. The strategy keeps them separate, even if they cannot be distinguished.

Theorem 4.5. *The category* Strat *forms an* IPA*-structure, and for any* $\vdash M : \mathbb{U}$, $[\![M]\!]_{\mathsf{Strat}} \Downarrow$ *iff* $M \Downarrow$.

See [Castellan and Clairambault 2020]. Here $\sigma : \mathbf{U}$ **converges**, written $\sigma \Downarrow$, if it has a +-move.

4.2 Strategic Petri Structures

Aiming for PStrat, given a game A and a Petri structure $\boldsymbol{\sigma}$ on $\mathsf{mult}(A)$, we must define what it means for $\boldsymbol{\sigma}$ to *follow the rules* of A – intuitively, the *runs* of $\boldsymbol{\sigma}$ must form valid plays on A:

Definition 4.6. A sequence $s = s_1 \dots s_n \in \mathsf{Moves}^*$ is a **play** on A, written $s \in \mathsf{Plays}(A)$, if it is:

valid:	for all $1 \leq i \leq n$, $\{s_1, \dots, s_i\} \in \mathscr{C}(A)$,
non-repetitive:	for all $1 \leq i \leq j \leq n$, if $s_i = s_j$ then $i = j$.

Recall that any *run* $\rho : \emptyset \twoheadrightarrow_{\boldsymbol{\sigma}} \alpha$ on $\boldsymbol{\sigma}$ generates a sequence $\mathsf{play}(\rho)$ obtained by collecting the visible moves in ρ – we shall ask that $\boldsymbol{\sigma}$ behaves as prescribed by the game as long as Opponent does. But we shall also need a *safety* condition, required for the unfolding. For that, we set:

Definition 4.7. Consider $\mathbf{e} : \alpha \longmapsto_{\boldsymbol{\sigma}} \beta$ an event of $\boldsymbol{\sigma}$. We call $\mathsf{pre}(\mathbf{e}) = \alpha$ the **pre-condition** of $\mathbf{e}$, and $\mathsf{post}(\mathbf{e}) = \beta$ the **post-condition** of $\mathbf{e}$. The set $\mathsf{new}(\mathbf{e}) = \beta \setminus \alpha$ contains the tokils **produced** by $\mathbf{e}$; and $\mathsf{eat}(\mathbf{e}) = \alpha \setminus \beta$ those **consumed**. Those extend to firings by ignoring the context.

The distinction between $\mathsf{post}(\mathbf{f})$ and $\mathsf{new}(\mathbf{f})$ matters for **let**: its transition s (see Section 3.3) requires $([], \mathsf{d})^{@3}$ to fire, but leaves it unchanged. So $([], \mathsf{d})^{@3}$ is both a *pre-condition* and a *post-condition*, but is not *produced*. Other than for **let**, pre- and post-conditions are always disjoint.

Our safety constraint uses the notion of *collection*, which gathers the tokils seen in a run so far:

Definition 4.8. Consider $\rho = \mathbf{f}_1 \dots \mathbf{f}_n : \emptyset \twoheadrightarrow_{\boldsymbol{\sigma}} \alpha_{i+1}$, a run of $\boldsymbol{\sigma}$, with $\mathbf{f}_i : \alpha_i \longrightarrow \alpha_{i+1}$. The **collection** of ρ is $\mathsf{Coll}(\rho) = \bigcup_{1 \leq i \leq n+1} \alpha_i$. We say that $\alpha \in \mathsf{cond}_{\boldsymbol{\sigma}}$ is **fresh in** ρ iff $\alpha \cap \mathsf{Coll}(\rho) = \emptyset$.

This lets us finally define *Petri strategies* on a game A:

Definition 4.9. We say that $\boldsymbol{\sigma}$ is **strategic on** A if for any $\rho : \emptyset \xrightarrow{s}\!\!\!\!\rightarrow_{\boldsymbol{\sigma}} \alpha$ with $s \in \mathsf{Plays}(A)$,

valid:	if $\mathbf{f}^+ : \alpha \xrightarrow{a} \beta$, then $sa \in \mathsf{Plays}(A)$,
receptive:	if $sa^- \in \mathsf{Plays}(A)$, there is a unique $\mathbf{e}^- \in \mathscr{E}(\boldsymbol{\sigma})$ s.t. $\mathbf{e}^- : \emptyset \overset{a^-}{\longmapsto} \beta$ with $\beta \cap \alpha = \emptyset$,
strongly safe:	if $\mathbf{f} : \alpha \longrightarrow \beta$ or $\mathbf{f} : \alpha \xrightarrow{a} \beta$ with $sa \in \mathsf{Plays}(A)$, then $\mathsf{new}(\mathbf{f})$ is fresh in ρ,

we write $\boldsymbol{\sigma} : A$ – additionally, $\boldsymbol{\sigma} : A$ is **negative** iff for all $\mathbf{e}^0 : \alpha \longmapsto \beta$ or $\mathbf{e}^+ : \alpha \overset{a}{\longmapsto} \emptyset$, $\alpha \neq \emptyset$.

Strong safety entails that in a given execution, a tokil can occur at most once: it cannot appear, be consumed, and reappear later. Thus for each occurring tokil we can unambiguously identify a *cause*, which will be central in the *unfolding* of Petri strategies to actual strategies – see Section 4.3.

Examples of Petri strategies abound in this paper so far, since it shall follow that for all term $\Gamma \vdash M : A$, the Petri structure $[\![M]\!]$ is a Petri strategy on $![\![\Gamma]\!] \vdash [\![A]\!]$. In contrast, the Petri structure **skip** of Section 2.1.1 does not satisfy *e.g.* **skip** : **N**, as there is a clear failure of condition *valid*.

Proposition 4.10. *There is an* IPA*-structure* PStrat*, with objects arenas (with the associated constructions), and morphisms from* A *to* B *the negative Petri strategies* $\boldsymbol{\sigma} : A \vdash B$ *up to isomorphism. There is an identity-on-morphisms* IPA*-functor* $\mathcal{F} : \mathsf{PStrat} \to \mathsf{PStruct}$ *sending* A *to* $\mathsf{mult}(A)$.

Proof. We must show that all primitives are strategic, and that all operations preserve the property. The main technical challenge is composition: to show that $\boldsymbol{\tau} \odot \boldsymbol{\sigma}$ is a Petri strategy, we exploit that none of $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$ can be the first one to break the rules. See details in Appendix D. □

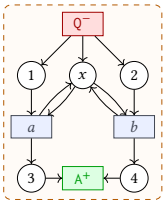


Fig. 26. **let** x = skip **in** $x \parallel x$

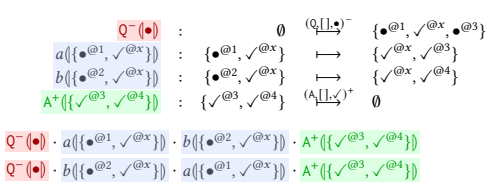


Fig. 27. Events and maximal valid runs

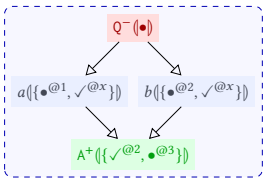


Fig. 28. Max. causal run

4.3 The Unfolding of a Petri Strategy

Leveraging strong safety we can give a rather direct definition of unfolding, close to the unfolding of 1-safe Petri nets in [Nielsen and Thiagarajan 2002]. An event structure is a global object, putting together all possible executions with explicit causal and branching information. Our definition first performs a causal reconstruction for individual runs, before patching them together via an already existing operation – that of taking the *primes* of a rigid family [Castellan et al. 2014b].

4.3.1 Causal Reconstruction of Runs. Fix a Petri strategy $\boldsymbol{\sigma} : A \vdash B$. A **valid run** on $\boldsymbol{\sigma}$ is $\rho : \emptyset \longrightarrow\!\!\rightarrow_{\boldsymbol{\sigma}} \gamma$ such that $\mathsf{play}(\rho) \in \mathsf{Plays}(A \vdash B)$. Valid runs are the executions that abide by the typing discipline of $A \vdash B$. An **event e of** ρ is an event $\mathbf{e}$ of $\boldsymbol{\sigma}$ such that there exists γ with $\mathbf{e} \uplus \gamma$ a firing in ρ. By strong safety, γ must be unique. We write $\mathscr{E}(\rho)$ for the set of events of ρ. Two runs are **equivalent** if they have the same events: two equivalent runs only differ by meaningless ordering of some events. This induces a *semantic* dependence on $\mathscr{E}(\rho)$: $\mathbf{e} \leq_\rho \mathbf{e}'$ iff $\mathbf{e}$ occurs before $\mathbf{e}'$ in any valid ρ' of $\boldsymbol{\sigma}$ equivalent to ρ. Our causal reconstruction relies on a concrete characterization of this.

Intuitively an event $\mathbf{e}$ of ρ should directly cause $\mathbf{e}'$ if $\mathsf{pre}(\mathbf{e}) \cap \mathsf{post}(\mathbf{e}') \neq \emptyset$, however, in some cases (notably for `let`) the same tokil may be both consumed and produced – *i.e.* essentially left in place – and this should be accounted for when computing the dependency. An example of this is illustrated in Figures 27 and 28 (omitting exponential stacks): we have $\mathsf{pre}(a(\!|\{\bullet^{@1}, \checkmark^{@x}\}|\!)) \cap \mathsf{post}(b(\!|\{\bullet^{@2}, \checkmark^{@x}\}|\!)) \neq \emptyset$, and yet those two events may be reordered at will.

So we adjust this and define the **dynamic dependency** between events of a run by

$$\mathbf{e} \lhd_{\boldsymbol{\sigma}} \mathbf{e}' \qquad \Leftrightarrow \qquad (\mathsf{new}(\mathbf{e}) \cap \mathsf{pre}(\mathbf{e}') \neq \emptyset) \quad \vee \quad (\mathsf{post}(\mathbf{e}) \cap \mathsf{eat}(\mathbf{e}') \neq \emptyset)\,,$$

on $\mathscr{E}(\rho)$ for any valid $\rho : \emptyset \longrightarrow\!\!\rightarrow_{\boldsymbol{\sigma}} \gamma$. In other words, $\mathbf{e} \lhd_{\boldsymbol{\sigma}} \mathbf{e}'$ if *(1)* $\mathbf{e}'$ requires a token *produced* by $\mathbf{e}$; or *(2)* $\mathbf{e}'$ *consumes* a token that $\mathbf{e}$ outputs. We also incorporate the **static dependency**

$$\mathbf{e} \lhd_{A \vdash B} \mathbf{e}' \Leftrightarrow \mathbf{e} : \alpha \xmapsto{a} \beta,\ \mathbf{e}' : \alpha' \xmapsto{a'} \beta',\ \text{where } a \rightarrowtriangle_{A \vdash B} a'$$

importing on events of ρ the dependencies imposed by the game. Altogether, we obtain:

Proposition 4.11. *For ρ a valid run of $\boldsymbol{\sigma}$, $\leq_\rho = (\lhd_{A \vdash B} \cup \lhd_{\boldsymbol{\sigma}})^*$.*

Posets $(\mathscr{E}(\rho), \leq_\rho)$ are referred to as **causal runs** of $\boldsymbol{\sigma}$; we write $\mathcal{R}(\boldsymbol{\sigma})$ for the causal runs of $\boldsymbol{\sigma}$. From Proposition 4.11, valid runs of $\boldsymbol{\sigma}$ are exactly the linearizations of causal runs in $\mathcal{R}(\boldsymbol{\sigma})$.

A valid run ρ of $\boldsymbol{\sigma}$ comprises *visible* events (generated by visible transitions) and *neutral* events (generated by neutral transitions), and this dichotomy transports to causal runs. If $q \in \mathcal{R}(\boldsymbol{\sigma})$, the set of labels of visible events is by construction a configuration of $A \vdash B$, written $\partial_{\boldsymbol{\sigma}}(q) \in \mathscr{C}(A \vdash B)$.

4.3.2 Hiding. The causal run $\mathscr{E}(\rho)$ associated to a valid run ρ reconstructs causal dependencies among *all* events in ρ, neutral or visible. This contrasts with the game semantics of Section 4.1, which records visible events only. To link them, we use the following definition:

Definition 4.12. Consider $q \in \mathcal{R}(\sigma)$. The **hiding** of q, written $q_{\downarrow}$, has

$$\begin{array}{rl} \textit{events:} & \text{those } e \in |q| \text{ that are } \textit{visible}, \\ \textit{causality:} & \text{the restriction of } \leq_q, \end{array}$$

and $\partial_\sigma(q_{\downarrow}) = \partial_\sigma(q) \in \mathscr{C}(A \vdash B)$. We call $q_{\downarrow}$ a **visible causal run** of σ, and write $q_{\downarrow} \in \mathcal{R}^V(\sigma)$.

Visible causal runs are extracted via a causal reconstruction from runs of σ, which – if σ comes from a program M – are executions of M in our multi-token machine. Besides this operational nature, we shall see now that they match the denotational interpretation of M in concurrent games.

4.3.3 Unfolding to a Strategy. We aggregate all visible causal runs into one *event structure*, via:

Theorem 4.13. *There exists a strategy $\mathcal{U}(\sigma) : A$, unique up to isomorphism, such that there is an order-isomorphism $\varphi : \mathscr{C}(\mathcal{U}(\sigma)) \cong \mathcal{R}^V(\sigma)$ satisfying $\partial_{\mathcal{U}(\sigma)} = \partial_\sigma \circ \varphi$.*

Proof. The crux is that $\mathscr{C}^V(\sigma)$ forms a *rigid family* [Castellan et al. 2014b], *i.e.* a set $\mathcal{F}$ of posets closed under prefix and such that any finite directed subset has a supremum. In that case, the structure $\mathrm{Pr}(\mathcal{F}) = (|\mathrm{Pr}(\mathcal{F})|, \leq_{\mathrm{Pr}(\mathcal{F})}, \#_{\mathrm{Pr}(\mathcal{F})})$ defined by the following components

$$\begin{array}{rcl} |\mathrm{Pr}(\mathcal{F})| & = & \{q \in \mathcal{F} \mid q \text{ prime}\} \\ q \leq_{\mathrm{Pr}(\mathcal{F})} p & \Leftrightarrow & q \subseteq p \\ \neg(q \#_{\mathrm{Pr}(\mathcal{F})} p) & \Leftrightarrow & \{q, p\}\uparrow, \end{array}$$

where $q \in \mathcal{F}$ is **prime** iff it has a top element, is an event structure with $\chi_{\mathcal{F}} : \mathscr{C}(\mathrm{Pr}(\mathcal{F})) \cong \mathcal{F}$ an order-isomorphism – this is folklore in event structures, see Appendix E.2.

Using this, we set $\mathcal{U}(\sigma) = \mathrm{Pr}(\mathcal{R}^V(\sigma))$. Its display map is uniquely determined by its action on configurations, defined as $\partial_{\mathcal{U}(\sigma)} = \partial_\sigma \circ \chi_{\mathcal{R}^V(\sigma)}$. All remaining conditions follow. □

An event of $\mathcal{U}(\sigma)$ is not quite an event of σ, but a causal run with a top – which we regard as an event of σ (the top) together with a choice of a causal history leading to it.

4.3.4 Unfolding as an IPA*-functor.* As expected, unfolding preserves the interpretation:

Theorem 4.14. *We have an* IPA*-functor* $\mathcal{U} : \mathsf{PStrat} \to \mathsf{Strat}$.

Proof. A lengthy verification. Most cases follow by inspection, the crux being that the unfolding preserves composition. □

Hence, the main result of this paper follows by Lemma 3.3:

Corollary 4.15. *Consider $\Gamma \vdash M : A$ any* IPA *term. Then, $\mathcal{U}(\llbracket M \rrbracket_{\mathsf{PStrat}}) = \llbracket M \rrbracket_{\mathsf{Strat}}$.*

In other words, the strategy $\llbracket M \rrbracket_{\mathsf{Strat}}$ obtained by induction on syntax following the methodology of denotational semantics, can also be extracted directly from running the multi-token machine $\llbracket M \rrbracket_{\mathsf{PStrat}}$, and reading back the causal dependencies between events. From this also follows the adequacy of our multi-token GoI interpretation (Theorem 3.10). Indeed, for $\vdash M : \mathbb{U}$, then

$$M \Downarrow \overset{(1)}{\Leftrightarrow} \llbracket M \rrbracket_{\mathsf{Strat}} \Downarrow \overset{(2)}{\Leftrightarrow} \llbracket M \rrbracket_{\mathsf{PStrat}} \Downarrow \overset{(3)}{\Leftrightarrow} \llbracket M \rrbracket_{\mathsf{PStruct}} \Downarrow$$

where *(1)* is by Theorem 4.5, *(2)* is by Corollary 4.15 (it is immediate that $\llbracket M \rrbracket_{\mathsf{PStrat}} \Downarrow$ iff $\mathcal{U}(\llbracket M \rrbracket_{\mathsf{PStrat}}) \Downarrow$) and for *(3)*, by Lemma 3.3 and Proposition 4.10, $\llbracket M \rrbracket_{\mathsf{PStrat}} = \llbracket M \rrbracket_{\mathsf{PStruct}}$.

5 IMPLEMENTATION

We illustrate this with an interactive web application available here with some documentation here.

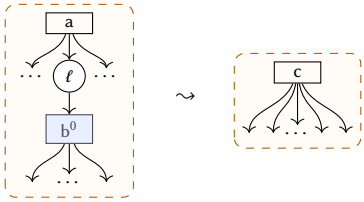


Fig. 29. Example of an optimization

Usage. The interface lets the user enter a IPA program (or choose from a list of examples) and then displays the Petri structure. The user can then fire available transitions and see the tokens flow through the net and the visible causal run obtained via causal reconstruction appear on the left. The implementation also includes a *reverse mode* letting the user undo causally maximal events, not necessarily in the reverse order in which they were played – leveraging our *causal* analysis of runs.

Optimisations. Our translation, following the categorical semantics, tends to generate large Petri structures. To keep the nets at a reasonable size, we have implemented several optimisations. We first eliminate locations and transitions that are unreachable from a negative transition, or that never reach a positive transition. Such "dead code" can arise during composition. Moreover, when we have several transitions occurring in a simple sequence, we combine them into one by composing their transition functions and eliminate the intermediate transitions and locations. One example of such optimisation is represented in Figure 29, where we merge a and b^0 and remove the location ℓ. The new transition c has $\mathsf{pre}(\mathsf{c}) = \mathsf{pre}(\mathsf{a})$ and $\mathsf{post}(\mathsf{c}) = \mathsf{post}(\mathsf{a}) \setminus \{\ell\} \cup \mathsf{post}(\mathsf{c})$ and transition function: $\delta\langle\mathsf{c}\rangle(\alpha) = \delta\langle\mathsf{a}\rangle(\alpha) \setminus \{\mathsf{t}\} \cup \delta\langle\mathsf{b}\rangle(\{\mathsf{t}\})$, where t denotes the tokil at ℓ in $\delta\langle\mathsf{a}\rangle(\alpha)$.

There are minor inconsistencies between examples in the paper and the implementation as optimisation choices are not unique. In the paper, they are chosen so as to make transition functions more intuitive, which sometimes leads to different choices (compare *e.g.* Figure 12 with this).

6 CONCLUSION

Though this is a theoretical contribution, we believe it is worth exploring applications to the compilation and analysis of higher-order, concurrent, effectful programming.

Our translation confines the infinity to tokens (*data* and *exponential* signatures). Forgetting colours, we immediately obtain a finitary over-approximation of the behaviour of programs, a Petri net in the usual sense, that may be used to prove *e.g.* safety properties. This may be refined by handling colours symbolically or over-approximating them via abstract interpretation – perhaps offering a new truly concurrent basis for the static analysis of higher-order concurrent programs.

Though it is subtle semantically, IPA is of course hardly a realistic programming language. We do not foresee any fundamental obstacle in generalizing this approach: we expect that building a machine faithful to game semantics will let us leverage the impressive array of computational features that this semantic framework has been able to model.

ACKNOWLEDGMENTS

We are grateful to Olivier Laurent for enlightening discussions on the Geometry of Interaction.

Work supported by the ANR project DyVerSe (ANR-19-CE48-0010-01); and by the Labex MiLyon (ANR-10-LABX-0070) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007), operated by the French National Research Agency (ANR).

REFERENCES

Samson Abramsky, Radha Jagadeesan, and Pasquale Malacaria. 2000. Full Abstraction for PCF. *Inf. Comput.* 163, 2 (2000), 409–470. https://doi.org/10.1006/inco.2000.2930

John C. Baez and Jade Master. 2020. Open Petri nets. *Math. Struct. Comput. Sci.* 30, 3 (2020), 314–341.

Patrick Baillot. 1999. *Approches dynamiques en sémantique de la logique linéaire: jeux et géométrie de l'interaction.* Ph. D. Dissertation. Aix-Marseille 2.

Simon Castellan and Pierre Clairambault. 2020. Disentangling Parallelism and Interference in Game Semantics. (2020). https://arxiv.org/abs/2103.15453 Submitted.

Simon Castellan, Pierre Clairambault, Hugo Paquet, and Glynn Winskel. 2018. The concurrent game semantics of Probabilistic PCF. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, Anuj Dawar and Erich Grädel (Eds.). ACM, 215–224. https://doi.org/10.1145/3209108.3209187

Simon Castellan, Pierre Clairambault, Silvain Rideau, and Glynn Winskel. 2017. Games and Strategies as Event Structures. *Logical Methods in Computer Science* 13, 3 (2017). https://doi.org/10.23638/LMCS-13(3:35)2017

Simon Castellan, Pierre Clairambault, and Glynn Winskel. 2014a. Symmetry in concurrent games. In *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, Thomas A. Henzinger and Dale Miller (Eds.). ACM, 28:1–28:10. https://doi.org/10.1145/2603088.2603141

Simon Castellan, Pierre Clairambault, and Glynn Winskel. 2019. Thin Games with Symmetry and Concurrent Hyland-Ong Games. *Log. Methods Comput. Sci.* 15, 1 (2019). https://doi.org/10.23638/LMCS-15(1:18)2019

Simon Castellan, Jonathan Hayman, Marc Lasson, and Glynn Winskel. 2014b. Strategies as Concurrent Processes. In *Proceedings of the 30th Conference on the Mathematical Foundations of Programming Semantics, MFPS 2014, Ithaca, NY, USA, June 12-15, 2014 (Electronic Notes in Theoretical Computer Science, Vol. 308)*, Bart Jacobs, Alexandra Silva, and Sam Staton (Eds.). Elsevier, 87–107. https://doi.org/10.1016/j.entcs.2014.10.006

Thomas Chatain and Eric Fabre. 2010. Factorization Properties of Symbolic Unfoldings of Colored Petri Nets. In *Applications and Theory of Petri Nets, 31st International Conference, PETRI NETS 2010, Braga, Portugal, June 21-25, 2010. Proceedings (Lecture Notes in Computer Science, Vol. 6128)*, Johan Lilius and Wojciech Penczek (Eds.). Springer, 165–184. https://doi.org/10.1007/978-3-642-13675-7_11

Pierre Clairambault and Marc de Visme. 2020. Full abstraction for the quantum lambda-calculus. *Proc. ACM Program. Lang.* 4, POPL (2020), 63:1–63:28.

Ugo Dal Lago, Claudia Faggian, Ichiro Hasuo, and Akira Yoshimizu. 2014. The geometry of synchronization. In *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, Thomas A. Henzinger and Dale Miller (Eds.). ACM, 35:1–35:10. https://doi.org/10.1145/2603088.2603154

Ugo Dal Lago, Claudia Faggian, Benoît Valiron, and Akira Yoshimizu. 2015. Parallelism and Synchronization in an Infinitary Context. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015.* IEEE Computer Society, 559–572. https://doi.org/10.1109/LICS.2015.58

Ugo Dal Lago, Claudia Faggian, Benoît Valiron, and Akira Yoshimizu. 2017. The geometry of parallelism: classical, probabilistic, and quantum effects. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017*, Giuseppe Castagna and Andrew D. Gordon (Eds.). ACM, 833–845. http://dl.acm.org/citation.cfm?id=3009859

Ugo Dal Lago, Ryo Tanaka, and Akira Yoshimizu. 2017. The geometry of concurrent interaction: Handling multiple ports by way of multiple tokens. In *LICS*. IEEE Computer Society, 1–12.

Vincent Danos, Hugo Herbelin, and Laurent Regnier. 1996. Game Semantics & Abstract Machines. In *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996.* IEEE Computer Society, 394–405. https://doi.org/10.1109/LICS.1996.561456

Vincent Danos and Laurent Regnier. 1996. Reversible, Irreversible and Optimal Lambda-machines. *Electron. Notes Theor. Comput. Sci.* 3 (1996), 40–60.

Thomas Ehrhard. 2012. The Scott model of linear logic is the extensional collapse of its relational model. *Theor. Comput. Sci.* 424 (2012), 20–45. https://doi.org/10.1016/j.tcs.2011.11.027

Dan R. Ghica and Andrzej S. Murawski. 2008. Angelic semantics of fine-grained concurrency. *Ann. Pure Appl. Logic* 151, 2-3 (2008), 89–114.

Dan R. Ghica and Nikos Tzevelekos. 2012. A System-Level Game Semantics. In *Proceedings of the 28th Conference on the Mathematical Foundations of Programming Semantics, MFPS 2012, Bath, UK, June 6-9, 2012 (Electronic Notes in Theoretical Computer Science, Vol. 286)*, Ulrich Berger and Michael W. Mislove (Eds.). Elsevier, 191–211. https://doi.org/10.1016/j.entcs.2012.08.013

Jean-Yves Girard. 1989. Geometry of interaction 1: Interpretation of System F. In *Studies in Logic and the Foundations of Mathematics.* Vol. 127. Elsevier, 221–260.

Jean-Yves Girard. 1995. Geometry of interaction III: accommodating the additives. *London Mathematical Society Lecture Note Series* (1995), 329–389.

Georges Gonthier, Martín Abadi, and Jean-Jacques Lévy. 1992. The Geometry of Optimal Lambda Reduction. In *POPL*. ACM Press, 15–26.

Ichiro Hasuo and Naohiko Hoshino. 2017. Semantics of higher-order quantum computation via geometry of interaction. *Ann. Pure Appl. Log.* 168, 2 (2017), 404–469. https://doi.org/10.1016/j.apal.2016.10.010

Jonathan Hayman. 2014. Interaction and Causality in Digital Signature Exchange Protocols. In *Trustworthy Global Computing - 9th International Symposium, TGC 2014, Rome, Italy, September 5-6, 2014. Revised Selected Papers (Lecture Notes in Computer Science, Vol. 8902)*, Matteo Maffei and Emilio Tuosto (Eds.). Springer, 128–143. https://doi.org/10.1007/978-3-662-45917-1_9

Jonathan Hayman and Glynn Winskel. 2008. The unfolding of general Petri nets. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2008, December 9-11, 2008, Bangalore, India (LIPIcs, Vol. 2)*, Ramesh Hariharan, Madhavan Mukund, and V. Vinay (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 223–234. https://doi.org/10.4230/LIPIcs.FSTTCS.2008.1755

Naohiko Hoshino, Koko Muroya, and Ichiro Hasuo. 2014. Memoryful geometry of interaction: from coalgebraic components to algebraic effects. In *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, Thomas A. Henzinger and Dale Miller (Eds.). ACM, 52:1–52:10. https://doi.org/10.1145/2603088.2603124

J. M. E. Hyland and C.-H. Luke Ong. 2000. On Full Abstraction for PCF: I, II, and III. *Inf. Comput.* 163, 2 (2000), 285–408. https://doi.org/10.1006/inco.2000.2917

Guilhem Jaber. 2015. Operational Nominal Game Semantics. In *Foundations of Software Science and Computation Structures - 18th International Conference, FoSSaCS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings (Lecture Notes in Computer Science, Vol. 9034)*, Andrew M. Pitts (Ed.). Springer, 264–278. https://doi.org/10.1007/978-3-662-46678-0_17

James Laird. 2007. A Fully Abstract Trace Semantics for General References. In *ICALP (Lecture Notes in Computer Science, Vol. 4596)*. Springer, 667–679.

Olivier Laurent. 2001. A Token Machine for Full Geometry of Interaction. In *Typed Lambda Calculi and Applications, 5th International Conference, TLCA 2001, Krakow, Poland, May 2-5, 2001, Proceedings (Lecture Notes in Computer Science, Vol. 2044)*, Samson Abramsky (Ed.). Springer, 283–297. https://doi.org/10.1007/3-540-45413-6_23

Paul Blain Levy and Sam Staton. 2014. Transition systems over games. In *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, Thomas A. Henzinger and Dale Miller (Eds.). ACM, 64:1–64:10. https://doi.org/10.1145/2603088.2603150

Ian Mackie. 1995. The Geometry of Interaction Machine. In *Conference Record of POPL'95: 22nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Francisco, California, USA, January 23-25, 1995*, Ron K. Cytron and Peter Lee (Eds.). ACM Press, 198–208. https://doi.org/10.1145/199448.199483

Damiano Mazza. 2005. Multiport Interaction Nets and Concurrency. In *CONCUR (Lecture Notes in Computer Science, Vol. 3653)*. Springer, 21–35.

Koko Muroya, Naohiko Hoshino, and Ichiro Hasuo. 2016. Memoryful geometry of interaction II: recursion and adequacy. In *POPL*. ACM, 748–760.

Mogens Nielsen, Gordon D. Plotkin, and Glynn Winskel. 1981. Petri Nets, Event Structures and Domains, Part I. *Theor. Comput. Sci.* 13 (1981), 85–108. https://doi.org/10.1016/0304-3975(81)90112-2

Mogens Nielsen and P. S. Thiagarajan. 2002. Regular Event Structures and Finite Petri Nets: The Conflict-Free Case. In *Applications and Theory of Petri Nets 2002, 23rd International Conference, ICATPN 2002, Adelaide, Australia, June 24-30, 2002, Proceedings (Lecture Notes in Computer Science, Vol. 2360)*, Javier Esparza and Charles Lakos (Eds.). Springer, 335–351. https://doi.org/10.1007/3-540-48068-4_20

Gordon Stewart, Lennart Beringer, Santiago Cuellar, and Andrew W. Appel. 2015. Compositional CompCert. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015*, Sriram K. Rajamani and David Walker (Eds.). ACM, 275–287. https://doi.org/10.1145/2676726.2676985

Glynn Winskel. 1982. Event Structure Semantics for CCS and Related Languages. In *Automata, Languages and Programming, 9th Colloquium, Aarhus, Denmark, July 12-16, 1982, Proceedings (Lecture Notes in Computer Science, Vol. 140)*, Mogens Nielsen and Erik Meineche Schmidt (Eds.). Springer, 561–576. https://doi.org/10.1007/BFb0012800

Glynn Winskel. 1986. Event Structures. In *Advances in Petri Nets (Lecture Notes in Computer Science, Vol. 255)*. Springer, 325–392.

Basic red. for PCF

$$
\begin{array}{rcl}
(\lambda x^A.\, M)\, N & \rightsquigarrow & M[N/x] \\
\mathsf{if}\ b\ N_{\mathsf{tt}}\ N_{\mathsf{ff}} & \rightsquigarrow & N_b \\
\mathsf{Y}\, M & \rightsquigarrow & M\,(\mathsf{Y}\, M) \\
\mathsf{let}\ x = v\ \mathsf{in}\ M & \rightsquigarrow & M[v/x] \\
\mathsf{f}(v_1, v_2) & \rightsquigarrow & v \\
(\mathsf{if}\ \mathsf{f}(v_1, v_2) & = & v)
\end{array}
$$

Basic reductions for references and semaphores

$$
\begin{array}{rcl}
\mathsf{newref}\ x\ \mathsf{in}\ v & \rightsquigarrow & v \\
\mathsf{newsem}\ x\ \mathsf{in}\ v & \rightsquigarrow & v
\end{array}
$$

Stateful reductions

$$
\begin{array}{rcl}
\langle !\ell, s \uplus \{\ell \mapsto n\}\rangle & \rightsquigarrow & \langle n, s \uplus \{\ell \mapsto n\}\rangle \\
\langle \ell{:=}n, s \uplus \{\ell \mapsto _\}\rangle & \rightsquigarrow & \langle \mathsf{skip}, s \uplus \{\ell \mapsto n\}\rangle \\
\langle \mathsf{grab}(\ell), s \uplus \{\ell \mapsto 0\}\rangle & \rightsquigarrow & \langle \mathsf{skip}, s \uplus \{\ell \mapsto 1\}\rangle \\
\langle \mathsf{rel}(\ell), s \uplus \{\ell \mapsto n\}\rangle & \rightsquigarrow & \langle \mathsf{skip}, s \uplus \{\ell \mapsto 0\}\rangle\ (n > 0)
\end{array}
$$

Stateless context rules

$$
\frac{M \rightsquigarrow M'}{M\,N \rightsquigarrow M'\,N}
\qquad
\frac{M \rightsquigarrow M'}{\mathsf{if}\ M\ N_1\ N_2 \rightsquigarrow \mathsf{if}\ M'\ N_1\ N_2}
\qquad
\frac{M \rightsquigarrow M'}{!M \rightsquigarrow !M'}
\qquad
\frac{N \rightsquigarrow N'}{M{:=}N \rightsquigarrow M{:=}N'}
$$

$$
\frac{M \rightsquigarrow M'}{\mathsf{grab}(M) \rightsquigarrow \mathsf{grab}(M')}
\qquad
\frac{M \rightsquigarrow M'}{\mathsf{rel}(M) \rightsquigarrow \mathsf{rel}(M')}
\qquad
\frac{M \rightsquigarrow M'}{M{:=}v \rightsquigarrow M'{:=}v}
$$

$$
\frac{N \rightsquigarrow N'}{\mathsf{let}\ x = N\ \mathsf{in}\ M \rightsquigarrow \mathsf{let}\ x = N'\ \mathsf{in}\ M}
\qquad
\frac{M_1 \rightsquigarrow M_1'}{\mathsf{f}(M_1, M_2) \rightsquigarrow \mathsf{f}(M_1', M_2)}
\qquad
\frac{M_2 \rightsquigarrow M_2'}{\mathsf{f}(M_1, M_2) \rightsquigarrow \mathsf{f}(M_1, M_2')}
$$

Stateful context rules

$$
\frac{\langle M[\ell/x], s \uplus \{\ell \mapsto n\}\rangle \rightsquigarrow \langle M'[\ell/x], s' \uplus \{\ell \mapsto n'\}\rangle}{\langle \mathsf{newref}\ x{:=}n\ \mathsf{in}\ M, s\rangle \rightsquigarrow \langle \mathsf{newref}\ x{:=}n'\ \mathsf{in}\ M', s'\rangle}\ (\ell \in \mathcal{L}\ \text{fresh})
$$

$$
\frac{\langle M[\ell/x], s \uplus \{\ell \mapsto n\}\rangle \rightsquigarrow \langle M'[\ell/x], s' \uplus \{\ell \mapsto n'\}\rangle}{\langle \mathsf{newsem}\ x{:=}n\ \mathsf{in}\ M, s\rangle \rightsquigarrow \langle \mathsf{newsem}\ x{:=}n'\ \mathsf{in}\ M', s'\rangle}\ (\ell \in \mathcal{L}\ \text{fresh})
$$

Fig. 30. Operational semantics of IPA

A OPERATIONAL SEMANTICS

We include in Figure 30 the operational semantics of IPA. Here, $M \rightsquigarrow N$ stands for

$$\langle M, s\rangle \rightsquigarrow \langle N, s\rangle\,,$$

i.e. all rules operate on configurations $\langle M, s\rangle$, but only the relevant part of the tuple is shown.

B THE IPA-STRUCTURE PStruct

We gather here the missing pieces for the construction of the IPA-structure PStruct.

B.1 PStruct as an IPA-structure: Constructions, Operations

Now, we introduce the IPA-structure operations for Petri structures, following Definition 3.1.

For PStruct we fix $\mathsf{PStruct}_\bullet = \mathsf{PStruct}_0$: we do not need to distinguish well-opened objects.

B.1.1 Constructions. First of all, the *constructions* of Section 2.1 are applied to finite sets of addresses simply by applying the corresponding injections from Definition 2.1. In other words we set $M \otimes N = M +^{\otimes} N$; $\&_{x \in V} M_x = \uplus_{x \in V} \dot{\iota}^x_{\&}(M_x)$; $M \multimap N = M +^{\multimap} N$ and $!M = M$.

For basic types, we set $\mathbf{U}$, $\mathbf{B}$ and $\mathbf{N}$ as $\mathbf{G} = \{\mathsf{Q}^-, \mathsf{A}^+\}$. We postpone $\mathbf{V}$ and $\mathbf{S}$ to Section B.3.

B.1.2 Tensor. The tensor is described in Section 2.2.4.

B.1.3 Currying. Rather than merely introducing *currying*, we introduce a general operation to *rename* the addresses associated with visible transitions in a Petri structure:

Definition B.1. Take $\boldsymbol{\sigma}$ a Petri structure on M and $f : \mathcal{M} \rightharpoonup \mathcal{M}$ s.t. $M \subseteq \mathsf{dom}(f)$, $f(M) \subseteq N$.

The **renaming** $\boldsymbol{\sigma}[f]$, a Petri structure on N, is as $\boldsymbol{\sigma}$ except for $\partial_{\boldsymbol{\sigma}[f]}(t) = f(\partial_{\boldsymbol{\sigma}}(t))$.

The net itself is not affected, only the labelling function for visible transitions. Now:

Definition B.2. Consider $\boldsymbol{\sigma} \in \mathsf{PStruct}(!(\&[\Gamma, x : A, \Delta]), O)$. We define the function $\Lambda_x : \mathcal{M}_{\vdash} \rightharpoonup \mathcal{M}_{\vdash}$ by $\Lambda_x(\mathscr{r}_{\vdash}\mathsf{m}) = \mathscr{r}_{\vdash}\mathscr{r}_{\multimap}\mathsf{m}$, $\Lambda_x(\ell_{\vdash}\imath^x_{\&}\mathsf{m}) = \mathscr{r}_{\vdash}\ell_{\multimap}\mathsf{m}$, and $\Lambda_x(\mathsf{m}) = \mathsf{m}$ otherwise.
Then, setting $\Lambda^{\Gamma,\Delta}_{x:A,O}(\boldsymbol{\sigma}) = \boldsymbol{\sigma}[\Lambda_x]$, we obtain $\Lambda^{\Gamma,\Delta}_{x:A,O}(\boldsymbol{\sigma}) \in \mathsf{PStruct}(!(\&[\Gamma, \Delta]), !A \multimap O)$.

This reassigns visible transitions corresponding to variable x to the left hand side of $\multimap$ on the right hand side of $\vdash$. Transitions initially assigned to the right hand side must also be relabelled, but the rest are unchanged. The net itself (*i.e.* the graph) remains the same.

B.1.4 Promotion. Promotion is described in Section 2.3.3.

B.2 PStruct as an IPA-structure: Stateless Primitives

Next, we describe all the primitives involved in the interpretation of the λ-calculus and recursion.

B.2.1 Variable, evaluation. First, the *variable* is simply a copycat set to component $\imath^x_{\&}$ of the context.

Definition B.3. Consider $M \subseteq \mathcal{M}$ finite, and $x \in \mathsf{Var}$.
We define $\mathbf{var}_{x:M}$ as $\mathcal{L}_{\mathbf{var}_{x:M}} = \mathcal{L}_{\mathbf{c}_M}$, $\mathcal{T}_{\mathbf{var}_{x:M}} = \mathcal{T}_{\mathbf{c}_M}$, with the same pre- and post-conditions as for $\mathbf{c}_M$. We set $\partial_{\mathbf{var}_{x:M}}(\mathsf{m}, \ell) = \ell_{\vdash}\imath^x_{\&}\mathsf{m}$ and $\partial_{\mathbf{var}_{x:M}}(\mathsf{m}, \mathscr{r}) = \mathscr{r}_{\vdash}\mathsf{m}$. Finally, the transition table is:

$$\begin{array}{rcl}
\delta\langle(\mathsf{m}^+, \mathscr{r})\rangle(\{(\mathsf{s}, \mathsf{d})^{@\mathsf{m}}\}) & = & (\mathsf{s}, \mathsf{d}) \\
\delta\langle(\mathsf{m}^-, \ell)\rangle(\{(\mathsf{s}, \mathsf{d})^{@\mathsf{m}}\}) & = & (\blacklozenge :: \mathsf{s}, \mathsf{d}) \\
\delta\langle(\mathsf{m}^-, \mathscr{r})\rangle(\mathsf{s}, \mathsf{d}) & = & \{(\mathsf{s}, \mathsf{d})^{@\mathsf{m}}\} \\
\delta\langle(\mathsf{m}^+, \ell)\rangle(\blacklozenge :: \mathsf{s}, \mathsf{d}) & = & \{(\mathsf{s}, \mathsf{d})^{@\mathsf{m}}\}\,.
\end{array}$$

Recall that terms $\Gamma \vdash M : A$ are meant to be interpreted as morphisms $[\![M]\!] \in \mathcal{C}(![\![\Gamma]\!], [\![A]\!])$. The ! explains the $\blacklozenge$ in the transition table, which corresponds to *dereliction* in linear logic terminology.
Note that the top two of these transitions do not affect the token: they only forward it following the structure of the net. From now on, we call **trivial** such transitions and omit them for succinctness.
Evaluation, used for application, is also a copycat:

Definition B.4. For $M, N \subseteq \mathcal{M}$ finite sets, $\mathbf{ev}_{M,N} = \mathbf{c}_{M \multimap N}[\Omega]$, where $\Omega : \mathcal{M} \rightharpoonup \mathcal{M}$ is $\Omega(\mathscr{r}_{\vdash}\mathscr{r}_{\multimap}\mathsf{m}) = \mathscr{r}_{\vdash}\mathsf{m}$, $\Omega(\mathscr{r}_{\vdash}\ell_{\multimap}\mathsf{m}) = \ell_{\vdash}\mathscr{r}_{\otimes}\mathsf{m}$, $\Omega(\ell_{\vdash}\mathscr{r}_{\multimap}\mathsf{m}) = \ell_{\vdash}\ell_{\otimes}\mathscr{r}_{\multimap}\mathsf{m}$, $\Omega(\ell_{\vdash}\ell_{\multimap}\mathsf{m}) = \ell_{\vdash}\ell_{\otimes}\ell_{\multimap}\mathsf{m}$, and $\Omega(\mathsf{m}) = \mathsf{m}$ otherwise.

B.2.2 Other stateless primitives. For finite $M \subseteq \mathcal{M}$, the (binary) **contraction** $\mathbf{c}_M$ was presented in Section 2.3.2. The n-ary contraction, defined likewise, is omitted – it may also be obtained by induction, composing binary contractions.
The **fixpoint combinator** $\mathbf{Y}_M$ is similar: it has $\mathcal{L}_{\mathbf{Y}_M} = M$, $\mathcal{T}_{\mathbf{Y}_M} = (M +^{\multimap} M) +^{\vdash} M$, net in Figure 32a and transition rules in Figure 31. The net has no loops, but loops may arise by composition.
Operations were described in Section 3.3.1. **Constants**, **conditionals** are introduced in Figures 32b, 32c respectively, with transition rules in Figure 31. Hopefully their behaviour is clear at this point. Finally, the **let** was introduced in Section 3.3.3.

B.3 PStruct as an IPA-structure: Stateful Primitives

For types $\mathbb{V}$ and $\mathbb{S}$ we have addresses $\mathbf{V} = \{w_{\mathbb{V}}\mathsf{Q}^-, w_{\mathbb{V}}\mathsf{A}^+, \mathscr{r}_{\mathbb{V}}\mathsf{Q}^-, \mathscr{r}_{\mathbb{V}}\mathsf{A}^+\}$ and $\mathbf{S} = \{\mathscr{g}_{\mathbb{S}}\mathsf{Q}^-, \mathscr{g}_{\mathbb{S}}\mathsf{A}^+, \mathscr{r}_{\mathbb{S}}\mathsf{Q}^-, \mathscr{r}_{\mathbb{S}}\mathsf{A}^+\}$, where $w_{\mathbb{V}}$ is the address for *write* requests, $\mathscr{r}_{\mathbb{V}}$ for *read* requests, $\mathscr{g}_{\mathbb{S}}$ for *grab*, $\mathscr{r}_{\mathbb{S}}$ for *release*.

$$
\begin{aligned}
\delta_{\mathbf{if}}\langle \ell_{\vdash} \mathscr{r}_{\otimes} \ell_{\otimes} \mathsf{Q}^{+}\rangle(\{([],\mathsf{tt})^{@2}\}) &= ([],\bullet)\\
\delta_{\mathbf{if}}\langle \ell_{\vdash} \mathscr{r}_{\otimes} \ell_{\otimes} \mathsf{Q}^{+}\rangle(\{([],\mathsf{ff})^{@2}\}) &= ([],\bullet)\\
\\
\delta_{\mathrm{Y}_M}\langle \mathscr{r}_{\vdash} \mathsf{m}^{-}\rangle(\mathsf{s},\mathsf{d}) &= \{(\ell_{!}\blacklozenge :: \mathsf{s},\mathsf{d})^{@\mathsf{m}^{-}}\}\\
\delta_{\mathrm{Y}_M}\langle \ell_{\vdash} \ell_{\multimap} \mathsf{m}^{-}\rangle(\mathsf{e}_1 :: \mathsf{e}_2 :: \mathsf{s},\mathsf{d}) &= \{(\mathscr{r}_{!}\langle \mathsf{e}_1,\mathsf{e}_2\rangle :: \mathsf{s},\mathsf{d})^{@\mathsf{m}^{-}}\}\\
\delta_{\mathrm{Y}_M}\langle \mathscr{r}_{\vdash} \mathsf{m}^{+}\rangle(\{(\ell_{!}\blacklozenge :: \mathsf{s},\mathsf{d})^{@\mathsf{m}^{+}}\}) &= (\mathsf{s},\mathsf{d})\\
\delta_{\mathrm{Y}_M}\langle \ell_{\vdash} \ell_{\multimap} \mathsf{m}^{+}\rangle(\{(\mathscr{r}_{!}\langle e_1,e_2\rangle :: \mathsf{s},\mathsf{d})^{@\mathsf{m}^{+}}\}) &= (\mathsf{e}_1 :: \mathsf{e}_2 :: \mathsf{s},\mathsf{d})\\
\\
\delta_{\mathbf{newsem}}\langle \mathscr{r}_{\vdash} \mathsf{Q}^{-}\rangle([],\bullet) &= \{([],\bullet)^{@1},([],\mathsf{tt})^{@2}\}\\
\delta_{\mathbf{newsem}}\langle \mathsf{g}\rangle(\{([\mathsf{e}],\bullet)^{@3},(_,\mathsf{tt})^{@2}\}) &= \{([\mathsf{e}],\mathsf{ff})^{@2},([\mathsf{e}],\checkmark)^{@4}\}\\
\delta_{\mathbf{newsem}}\langle \mathsf{r}\rangle(\{([\mathsf{e}],\bullet)^{@3},(_,\mathsf{ff})^{@2}\}) &= \{([\mathsf{e}],\mathsf{tt})^{@2},([\mathsf{e}],\checkmark)^{@6}\}
\end{aligned}
$$

Fig. 31. Transition tables for IPA-structure primitives

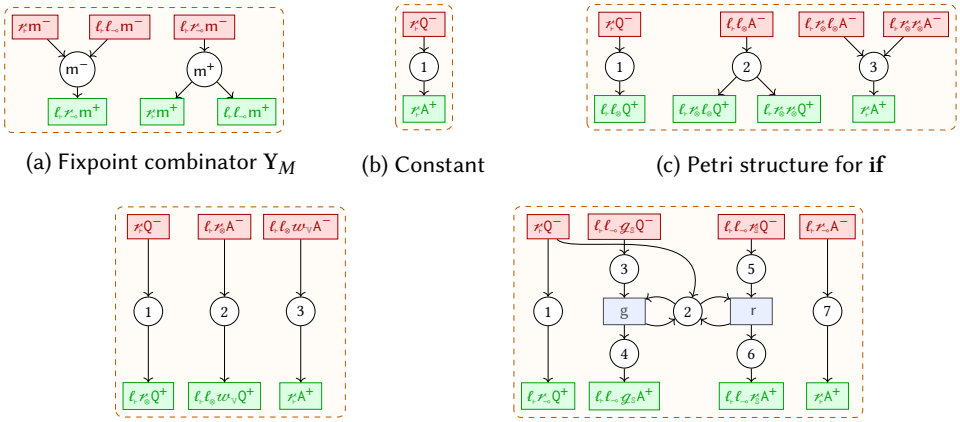


Fig. 32. The nets of Petri structures for IPA primitives

B.3.1 Reference and semaphore queries. Recall that Definition 3.1 requires primitives

$$
\begin{aligned}
&\mathbf{assign} \in C(\mathbf{V}\otimes\mathbf{N},\mathbf{U}), \quad && \mathbf{deref} \in C(\mathbf{V},\mathbf{N})\\
&\mathbf{grab} \in C(\mathbf{S},\mathbf{U}), && \mathbf{release} \in C(\mathbf{S},\mathbf{U})
\end{aligned}
$$

for reference and semaphore queries. Among these, we set $\mathbf{deref} = \mathfrak{a}_{\mathbf{N}}[\ell_{\vdash}\mathsf{m} \mapsto \ell_{\vdash}\mathscr{r}_{\forall}\mathsf{m}, \mathscr{r}_{\vdash}\mathsf{m} \mapsto \mathscr{r}_{\vdash}\mathsf{m}]$, $\mathbf{grab} = \mathfrak{a}_{\mathbf{U}}[\ell_{\vdash}\mathsf{m} \mapsto \ell_{\vdash}\mathscr{g}_{\mathbb{S}}\mathsf{m}, \mathscr{r}_{\vdash}\mathsf{m} \mapsto \mathscr{r}_{\vdash}\mathsf{m}]$ and $\mathbf{release} = \mathfrak{a}_{\mathbf{U}}[\ell_{\vdash}\mathsf{m} \mapsto \ell_{\vdash}\mathscr{r}_{\mathbb{S}}\mathsf{m}, \mathscr{r}_{\vdash}\mathsf{m} \mapsto \mathscr{r}_{\vdash}\mathsf{m}]$ copycat strategies simply accessing the matching component of the reference or semaphore.

The remaining query **assign** is more elaborate: upon being evaluated, it sends an evaluation request to its integer argument. Upon receiving a value, it sends the write request, and propagates the acknowledgement. The net is in Figure 32d and its transition rules are trivial.

B.3.2 Initialization. The actual stateful behaviour is provided by nets for new references and semaphores $\mathbf{newref} \in C(!\mathbf{V} \multimap \mathbf{X},\mathbf{X})$ and $\mathbf{newsem} \in C(!\mathbf{S} \multimap \mathbf{X},\mathbf{X})$. New references were introduced in Section 3.3.2. The (identical) net for new semaphores appears in Figure 32e with non-trivial transitions in Figure 31.

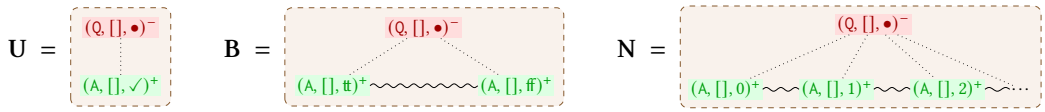


Fig. 33. Interpretation of IPA ground types

For new semaphores, in normal operation, location 2 contains exactly one token with data either tt or ff, encoding if the semaphore is free. The transitions g and r grab and release the semaphore, and may only fire if the semaphore currently has the adequate state.

C CONCURRENT STRATEGIES

We now provide details about the construction of the IPA structure of concurrent strategies as presented in [Castellan and Clairambault 2020].

C.1 Types and Contexts as Games

We detail the interpretation of types. Figure 33 presents the interpretation of ground types.

We now move on to constructions on games and arenas. We write either 1 or $\top$ for the empty arena. If $(\mathsf{m}, \mathsf{s}, \mathsf{d}) \in \mathsf{Moves}$, we write $\ell_\otimes((\mathsf{m}, \mathsf{s}, \mathsf{d})) = (\ell_\otimes(\mathsf{m}), \mathsf{s}, \mathsf{d})$ and likewise for all injections appearing in the definition of addresses. This extends to sets of moves by direct image. Address injections also apply to binary relations on moves in the obvious way, *e.g.* $\ell_\otimes(R) = \{(\ell_\otimes(m), \ell_\otimes(m')) \mid m\, R\, m'\}$.

Definition C.1. Consider A, B games. Then we set games $A \otimes B$ and $A \vdash B$ with components:

$$\begin{array}{rcl} |A \otimes B| & = & \ell_\otimes(|A|) \uplus r_\otimes(|B|) \\ \leq_{A\otimes B} & = & \ell_\otimes(\leq_A) \uplus r_\otimes(\leq_B) \\ \#_{A\otimes B} & = & \ell_\otimes(\#_A) \uplus r_\otimes(\#_B) \end{array}$$

$$\begin{array}{rcl} |A \vdash B| & = & \ell_\vdash(|A|) \uplus r_\vdash(|B|) \\ \leq_{A\vdash B} & = & \ell_\vdash(\leq_A) \uplus r_\vdash(\leq_B) \\ \#_{A\vdash B} & = & \ell_\vdash(\#_A) \uplus r_\vdash(\#_B) \end{array}$$

$A \otimes B$ is called the **tensor**, and $A \vdash B$ is called the **hom-game**.

If A, B are arenas, so is $A \otimes B$. In contrast, $A \vdash B$ is never an arena unless A is empty. Formally, both constructions are defined in the same way, but with a distinct injection – remember that $\ell_\vdash$ inverts polarity, so that in $A \vdash B$ the polarity is inverted in A.

From the definition, *configurations* of $A \otimes B$ have a restricted shape: any $x \in \mathscr{C}(A \otimes B)$ decomposes uniquely as $x = \ell_\otimes(x_A) \uplus r_\otimes(x_B)$ with $x_A \in \mathscr{C}(A)$ and $x_B \in \mathscr{C}(B)$ – we write $x = x_A \otimes x_B$. Likewise, any configuration $x \in \mathscr{C}(A \vdash B)$ decomposes as $x = x_A \vdash x_B = \ell_\vdash(x_A) \uplus r_\vdash(x_B)$.

C.1.1 Further arena constructions. We give the remaining constructions required by Definition 3.1.

First a new notation: for $m = (\mathsf{m}, \mathsf{s}, \mathsf{d}) \in \mathsf{Moves}$ and $\mathsf{e} \in \mathcal{E}$, we write $\mathsf{e} :: m = (\mathsf{m}, \mathsf{e} :: \mathsf{s}, \mathsf{d})$, *i.e.* the exponential signature implicitly applies to the exponential stack of m. This is an injection, and accordingly we apply it to sets and relations as with the injections from addresses.

Definition C.2. We define three other constructions on arenas:

linear arrow: for A, O arenas with O well-opened, we define $A \multimap O$ well-opened in Figure 34a.
product: for $(A_x)_{x \in V}$ a family of arenas with $V \subseteq \mathsf{Var}$, we define $\&_{x \in V} A_x$ in Figure 34b.
bang: for A an arena, we define $!A$ in Figure 34c.

$$
\begin{array}{rcl}
|A \multimap O| & = & \ell_{\multimap}(|A|) \uplus r_{\multimap}(|O|) \\
\leq_{A \multimap O} & = & \ell_{\multimap}(\leq_A) \uplus r_{\multimap}(\leq_O) \\
 & \uplus & r_{\multimap}(\min(O)) \times \ell_{\multimap}(|A|) \\
\#_{A \multimap O} & = & \ell_{\multimap}(\#_A) \uplus r_{\multimap}(\#_O)
\end{array}
$$

(a) Arrow

$$
\begin{array}{rcl}
|\,\&(A_x)_{x \in V}| & = & \biguplus_{x \in V} \iota^x_{\&}(|A_x|) \\
\leq_{\&(A_x)_{x \in V}} & = & \biguplus_{x \in V} \iota^x_{\&}(\leq_{A_x}) \\
\iota^x_{\&}(a) \# \iota^y_{\&}(a') & \Leftrightarrow & (x = y \wedge a \#_{A_x} a') \\
 & & \vee (x \neq y)
\end{array}
$$

(b) Product

$$
\begin{array}{rcl}
|!A| & = & \biguplus_{\mathsf{e} \in \mathcal{E}} \mathsf{e} :: |A| \\
\leq_{!A} & = & \biguplus_{\mathsf{e} \in \mathcal{E}} \mathsf{e} :: (\leq_A) \\
\#_{!A} & = & \biguplus_{\mathsf{e} \in \mathcal{E}} \mathsf{e} :: (\#_A)
\end{array}
$$

(c) Bang

Fig. 34. Arena constructions

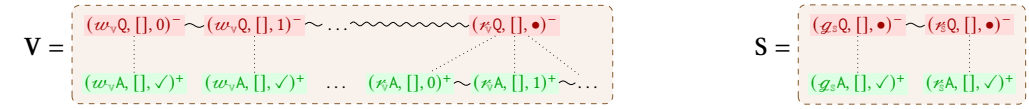


Fig. 35. Arenas for references and semaphores

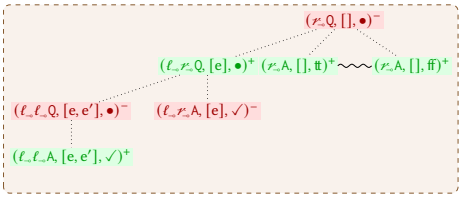


Fig. 36. Arena $!(!\mathbb{U} \multimap \mathbb{U}) \multimap \mathbb{B}$

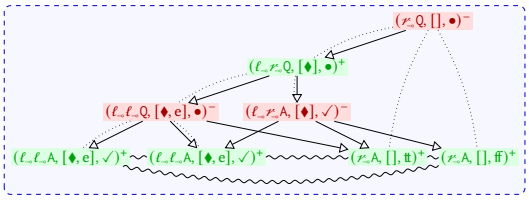


Fig. 37. Part of a concurrent strategy

It remains to give the arenas for references and semaphores, in Figure 35.

The arrow $A \multimap O$ enforces that an argument cannot be called before the function has been called. The bang $!A$ creates countably many independent copies of A, one for each exponential signature, for *thread indexing* (for that, [Castellan and Clairambault 2020] uses *integers*).

A more elaborate example of arena is in Figure 36. The representation is symbolic: from the exponentials in the construction, the full arena is infinite and comprises moves as in the diagram for all exponential signatures $\mathsf{e}, \mathsf{e}' \in \mathcal{E}$. However, we have *e.g.* $(\ell_{\multimap} r_{\multimap} \mathsf{Q}, [e], \bullet) \leq (\ell_{\multimap} \ell_{\multimap} \mathsf{Q}, [\mathsf{e}_1, \mathsf{e}_2], \bullet)$ only when $e = e_1$ – in an exponential stack, the first element corresponds to the outermost $!(-)$.

C.2 The Category of Arenas and Strategies

Events of a strategy σ inherit a polarity from $\mathrm{pol}_\sigma(s) = \mathrm{pol}(\partial(s))$ – a definition used implicitly from now on. Again, this definition is as in [Castellan and Clairambault 2020] without the component and conditions – unnecessary for this paper – pertaining to symmetry.

The event structure σ presents observable computational events along with their causal dependencies and conflicts. Events of σ are *not* moves of the game, but they do correspond to moves via the action of ∂. This permits an explicit representation of non-deterministic branching: several events of σ may correspond to the same move if they are conflicting and so belong to separate branches of the execution. The strategy keeps them separate, even if they cannot be distinguished.

We show in Figure 37 (part of) a concurrent strategy playing on the game in Figure 36. In such diagrams we represent the strategy and the explored part of the arena in a single picture. Nodes correspond to events of the strategy, drawn directly as their display through ∂ – this means that two conflicting nodes may have the same label, as happens in Figure 37. Arrows $\rightarrowtriangle$ correspond to the immediate causal dependency in σ, while dotted lines correspond as before to the causal dependency from the game. The diagram in Figure 37 is a part of the strategy for:

Example C.3. We introduce the term $\vdash$ **strictness** : $(\mathbb{U} \to \mathbb{U}) \to \mathbb{B}$, defined as

$$\begin{array}{ll} \vdash & \lambda f^{\mathbb{U}\to\mathbb{U}}.\, \mathsf{newref}\, x\, \mathsf{in}\, f\ (x := 1);\ \mathsf{not}\ (\mathsf{iszero}\ !x) \\ : & (\mathbb{U} \to \mathbb{U}) \to \mathbb{B}\,, \end{array}$$

using encapsulated state to test if a function f calls its argument (run it here).

Figure 37 reads as follows: Opponent initiates computation with $(\not\leftarrow_\circ \mathsf{Q}, [], \bullet)^-$. This lets Player call his argument with $(\ell_\multimap \not\leftarrow_\circ \mathsf{Q}, [\blacklozenge], \bullet)^+$. This lets Opponent return the call with $(\ell_\multimap \not\leftarrow_\circ \mathsf{A}, [\blacklozenge], \checkmark)^-$, or call its argument with $(\ell_\multimap \ell_\multimap \mathsf{Q}, [\blacklozenge, \mathsf{e}])^-$ for any $\mathsf{e} \in \mathcal{E}$ (in fact Opponent may call this argument arbitrarily many times with distinct exponential signatures, but Figure 37 only represents one call). But these events are not incompatible: though this behaviour is not realizable within IPA, our model lets Opponent call its argument *and* return *concurrently*! In that case both $x := 1$ and $!x$ are running, so there is a *race*. The conflicts in Figure 37 allow two outcomes, depending on who wins.

It remains to introduce the two conditions *courteous* and *receptive*: the former simply states that a strategy has no control over Opponent moves, and must acknowledge each Opponent move A uniquely. *Courtesy* entails that with respect to the immediate causal links of the game, a strategy can only add new dependencies from negative to positive moves. This formalizes the idea that strategies interact in an *asynchronous* environment, where *e.g.* causal links between positive moves may not be preserved by propagation of moves through buffers – or through a Petri structure.

C.2.1 Isomorphic strategies. Strict equality of strategies is too strict to be useful; instead we use:

Definition C.4. Consider $\sigma, \tau : A$ two (pre)strategies on game A.
An **isomorphism** $\varphi : \sigma \cong \tau$ is an invertible map of es $\varphi : \sigma \to \tau$ such that $\partial_\tau \circ \varphi = \partial_\sigma$.

We write $\sigma \cong \tau$ to mean that σ and τ are isomorphic, leaving the isomorphism unspecified. Clearly, this is an equivalence relation. It is a basic fact from the theory of event structures that isomorphisms between σ and τ are in one-to-one correspondence with order-isos $\varphi : \mathscr{C}(\sigma) \cong \mathscr{C}(\tau)$ such that $\partial_\tau \circ \varphi = \partial_\sigma$, *i.e.* any such order-iso is given by a unique isomorphism of strategies – this is convenient as it is often easier to construct a bijection between configurations rather than events.

For *strategies*, this can be simplified by ignoring trailing Opponent moves. A $x \in \mathscr{C}(\sigma)$ is **+-covered** if any $m \in x$ maximal in x is positive; we write $\mathscr{C}^+(\sigma)$ for the +-covered configurations of σ. The action of an iso φ on +-covered configurations suffices to completely describe it:

Lemma C.5. *Consider $\sigma, \tau : A$ two strategies, and $\varphi : \mathscr{C}^+(\sigma) \cong \mathscr{C}^+(\tau)$ an order-iso s.t. $\partial_\tau \circ \varphi = \partial_\sigma$. Then, there is a unique $\hat{\varphi} : \sigma \cong \tau$ such that $\hat{\varphi}(x) = \varphi(x)$ for all $x \in \mathscr{C}^+(\sigma)$.*

This is an application in the trivial case without symmetry of Lemma 5.11 from [Castellan and Clairambault 2020], which will be very helpful in constructing isomorphisms in this paper.

C.2.2 Composition. Working towards an IPA-structure, we must first define composition.

A strategy σ **from game** A **to game** B is defined as a strategy $\sigma : A \vdash B$. Let us fix for now games A, B and C and strategies $\sigma : A \vdash B$ and $\tau : B \vdash C$ that we wish to compose to $\tau \odot \sigma : A \vdash C$.

Lemma C.5 puts the emphasis on +-covered configurations, so we investigate what should be the +-covered configurations of $\tau \odot \sigma$. It turns out that they correspond to pairs $x^\sigma \in \mathscr{C}^+(\sigma)$ and $x^\tau \in \mathscr{C}^+(\tau)$ whose synchronization of x^σ and x^τ through B is "sound", which we must now define.

We fix the convention that if $x^\sigma \in \mathscr{C}(\sigma)$ and $x^\tau \in \mathscr{C}(\tau)$, we write $\partial_\sigma x^\sigma = x^\sigma_A \vdash x^\sigma_B$ and $\partial_\tau x^\tau = x^\tau_B \vdash x^\tau_C$ where $x^\sigma_A \in \mathscr{C}(A)$, $x^\sigma_B, x^\tau_B \in \mathscr{C}(B)$, and $x^\tau_C \in \mathscr{C}(C)$. We say $x^\sigma \in \mathscr{C}(\sigma)$ and $x^\tau \in \mathscr{C}(\tau)$ are **matching** if $x^\sigma_B = x^\tau_B$, in which case it is unambiguous to write $\partial_\sigma(x^\sigma) = x_A \vdash x_B$ and $\partial_\tau(x^\tau) = x_B \vdash x_C$. So x^σ and x^τ reach the same state on B, but it remains to see if they do it

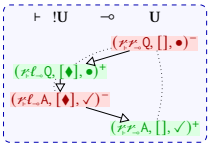


Fig. 38. Matching, secured configurations

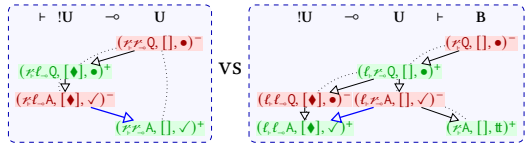


Fig. 39. Matching, non-secured configurations

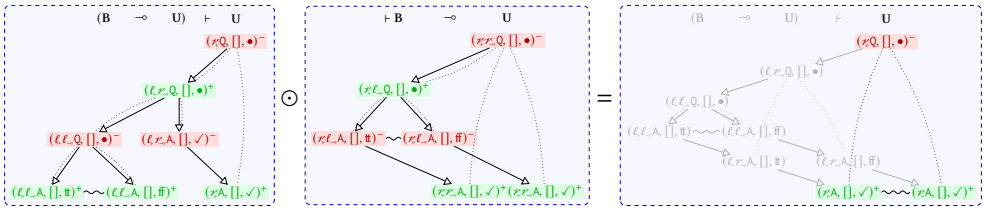


Fig. 40. Example of a composition

with compatible causal constraints. For that, we set $x_A \parallel x_B \parallel x_C = \ell(x_A) \uplus m(x_B) \uplus r(x_C)$, and

$$\begin{array}{rcccl} \partial^{\ell}_{\sigma} & : & x^{\sigma} & \longrightarrow & x_A \parallel x_B \parallel x_C \\ & & m & \mapsto & \ell(a) \qquad \text{if } \partial_{\sigma}(m) = \ell_{+}(a), \\ & & m & \mapsto & m(b) \qquad \text{if } \partial_{\sigma}(m) = r_{+}(b), \end{array}$$

$$\begin{array}{rcccl} \partial^{r}_{\tau} & : & x^{\tau} & \longrightarrow & x_A \parallel x_B \parallel x_C \\ & & n & \mapsto & m(b) \qquad \text{if } \partial_{\tau}(n) = \ell_{+}(b), \\ & & n & \mapsto & r(c) \qquad \text{if } \partial_{\tau}(n) = r_{+}(c). \end{array}$$

are variants of the display maps set to embed x^{σ} and x^{τ} in the common space $x_A \parallel x_B \parallel x_C$.

This lets us check the presence of deadlocks by importing all causal constraints to $x_A \parallel x_B \parallel x_C$:

Definition C.6. Consider $x^{\sigma} \in \mathscr{C}(\sigma)$ and $x^{\tau} \in \mathscr{C}(\tau)$ matching configurations.

They are **causally compatible** if the relation $\lhd = \lhd_{\sigma} \uplus \lhd_{\tau}$ on $x_A \parallel x_B \parallel x_C$ set with:

$$\begin{array}{lcll} \partial^{\ell}_{\sigma}(m) & \lhd_{\sigma} & \partial^{\ell}_{\sigma}(m') & \text{for } m <_{\sigma} m' \\ \partial^{r}_{\tau}(n) & \lhd_{\tau} & \partial^{r}_{\tau}(n') & \text{for } n <_{\tau} n' \end{array}$$

is acyclic. We also say that the pair x^{σ}, x^{τ} is **secured**.

We show in Figures 38 and 39 examples of matching secured and non-secured pairs involved in computing the composition of the strategy of Figure 37 with $\lambda x^{\mathbb{U}}.\, x$. In Figure 39, a deadlock directly arises from opposite causal constraints (highlighted in blue). This entails that the only result arising from this composition will be tt from Figure 38, as expected since $\lambda x.\, x$ is strict.

Then $\tau \odot \sigma$ is the *unique* strategy with as +-covered configurations the causally compatible pairs:

Proposition C.7. *Consider A, B, C games, and $\sigma : A \vdash B$ and $\tau : B \vdash C$ strategies. Then there is a strategy $\tau \odot \sigma : A \vdash C$, unique up to iso, s.t. there are order-isos:*

$$\begin{array}{rcl} (- \odot -) & : & \{(x^{\tau}, x^{\sigma}) \in \mathscr{C}^{+}(\tau) \times \mathscr{C}^{+}(\sigma) \mid \textit{causally compatible}\} \\ & \simeq & \mathscr{C}^{+}(\tau \odot \sigma) \end{array}$$

such that for any $x^{\sigma} \in \mathscr{C}^{+}(\sigma)$ and $x^{\tau} \in \mathscr{C}^{+}(\tau)$ causally compatible, $\partial_{\tau \odot \sigma}(x^{\tau} \odot x^{\sigma}) = x^{\sigma}_{A} \vdash x^{\tau}_{C}$.

This is a simplification of Proposition 3.3.1 from [Castellan and Clairambault 2020].

Concretely, composition is performed by parallel interaction (via the synchronizing product of es used by Winskel to model CCS [Winskel 1982]); followed by *hiding* which keeps the visible events only. In this paper, the above characterization suffices for our purposes.

Figure 40 shows an example composition which, ignoring exponentials, corresponds to:

$$[\![f : \mathbb{B} \to \mathbb{U} \vdash f\ \mathsf{coin}]\!] \odot [\![\vdash \lambda x^{\mathbb{B}}.\ \mathsf{if}\ x\ \mathsf{skip}\ \mathsf{skip}]\!] : 1 \vdash \mathbb{U}\,.$$

Observe the resulting strategy has two distinct ways to converge, even though the two occurrences of $(\mathsf{r}_{\mathsf{r}}\mathsf{A}, [\,], \checkmark)^+$ correspond to the same event of the left hand side strategy. Each event of the composition carries its whole causal history, including the exact synchronizations that lead to it.

C.3 The IPA-structure Strat

C.3.1 Categorical structure. Section C.2 already contains the data of Strat, and composition. It remains to define:

Copycat. Copycat may be defined on any game, but it is slightly simpler on arenas:

Definition C.8. For each arena A, the **copycat strategy** $\mathfrak{cc}_A : A \vdash A$ is defined as having:

$$\begin{array}{rcl}
|\mathfrak{cc}_A| & = & |A \vdash A| \\
\partial_{\mathfrak{cc}_A}(m) & = & m \\
\imath(a) \leq_{\mathfrak{cc}_A} \imath'(a) & \Leftrightarrow & a <_A a';\ \text{or}\ a = a',\ \mathrm{pol}(\imath(a)) = - \\
 & & \qquad\qquad \text{and}\ \mathrm{pol}(\imath'(a)) = + \\
\imath(a) \#_{\mathfrak{cc}_A} \imath'(a') & \Leftrightarrow & a \#_A a'\,,
\end{array}$$

where $\imath, \imath' \in \{\ell_{\mathsf{r}}, \mathsf{r}_{\mathsf{r}}\}$.

Copycat acts as an asynchronous forwarder, simply receptive to all Opponent moves and prepared to forward them to the other side as soon as they become available. This means that its +-covered configurations, where all moves have been successfully forwarded, have a particularly simple shape:

Lemma C.9. *Consider A any arena. Then, we have $\mathscr{C}^+(\mathfrak{cc}_A) = \{x_A \vdash x_A \in \mathscr{C}(A \vdash A) \mid x_A \in \mathscr{C}(A)\}$.*

From Lemma C.5 this characterizes $\mathfrak{cc}_A$ uniquely up to iso, just like Proposition C.7 for composition. It follows from these two facts that composition preserves isomorphisms, that it is associative and that identities are neutral for composition up to isomorphism, see [Castellan et al. 2017] for details.

Corollary C.10. *There is* Strat*, a category having arenas as objects, as morphisms from A to B the negative strategies on the game $A \vdash B$ up to isomorphism, and copycat strategies as identities.*

Unlike PStruct or PStrat, Strat *is* a category satisfying the identity laws – though this fact will not be directly useful for us in this paper.

C.3.2 Strat *as an* IPA*-structure: Operations.* We detail the different operations involved in the IPA-structure.

Tensor. Fix $\sigma_1 \in \mathsf{Strat}(A_1, B_1)$ and $\sigma_2 \in \mathsf{Strat}(A_2, B_2)$. We define:

Definition C.11. We define $\sigma_1 \otimes \sigma_2 \in \mathsf{Strat}(A_1 \otimes A_2, B_1 \otimes B_2)$ with:

$$\begin{array}{rcl}
|\sigma_1 \otimes \sigma_2| & = & |\sigma_1| + |\sigma_2| \\
\leq_{\sigma_1 \otimes \sigma_2} & = & \leq_{\sigma_1} + \leq_{\sigma_2} \\
\#_{\sigma_1 \otimes \sigma_2} & = & \#_{\sigma_1} + \#_{\sigma_2} \\
\partial_{\sigma_1 \otimes \sigma_2}(\ell(m)) & = & \partial_{\sigma_1}(m) \\
\partial_{\sigma_1 \otimes \sigma_2}(\mathsf{r}(m)) & = & \partial_{\sigma_2}(m)
\end{array}$$

The conditions for a strategy are straightforward, and so is:

Proposition C.12. *The strategy* $\sigma_1 \otimes \sigma_2 \in \mathsf{Strat}(A_1 \otimes A_2, B_1 \otimes B_2)$ *satisfies:*

$$(- \otimes -) : \mathscr{C}^+(\sigma_1) \times \mathscr{C}^+(\sigma_2) \to \mathscr{C}^+(\sigma_1 \otimes \sigma_2)$$

such that $\partial_{\sigma_1 \otimes \sigma_2}(x^{\sigma_1} \otimes x^{\sigma_2}) = (x^{\sigma_1}_{A_1} \otimes x^{\sigma_2}_{A_2}) \vdash (x^{\sigma_1}_{B_1} \otimes x^{\sigma_2}_{B_2})$ *for all* $x^{\sigma_1} \otimes x^{\sigma_2} \in \mathscr{C}^+(\sigma_1 \otimes \sigma_2)$.
Moreover, $\sigma_1 \otimes \sigma_2$ *is the* unique *strategy on* $A_1 \otimes A_2 \vdash B_1 \otimes B_2$ *satisfying this property.*

Proof. The property is a direct verification, and uniqueness follows from Lemma C.5. □

Currying. As for Petri structures, we start with *renaming.*

Definition C.13. Consider σ a strategy on game A, and $f : |A| \to |B|$.
Then, we define the **renaming** to be as σ except $\partial_{\sigma[f]}(m) = f(\partial_\sigma(m))$.

Without additional conditions, there is no reason why $\sigma[f]$ would be a strategy in general. A convenient situation is when f preserves sufficiently rigidly the rules of the game:

Proposition C.14. *We say that* $f : |A| \to |B|$ *is* ***valid*** *if it is a map of es, additionally satisfying hypotheses* receptive *and* courteous *from Definition 4.4*
If $\sigma : A$ *and* f *is valid, then* $\sigma[f]$ *is a strategy on* B.

Proof. Straightforward. □

However, we cannot use this directly for currying, because the function

$$\begin{array}{rcl} \Lambda^{\Gamma,\Delta}_{x:A,O} \quad : \quad |!\&[\Gamma, x : A, \Delta] \vdash O| & \to & |!\&[\Gamma, \Delta] \vdash !A \multimap O| \\ (\mathsf{m}, \mathsf{s}, \mathsf{d}) & \mapsto & (\Lambda^x(\mathsf{m}), \mathsf{s}, \mathsf{d}) \end{array}$$

using Λ_x from Definition B.2, is not valid (a singleton configuration in A on the left hand side is indeed sent to a non-configuration on the right hand side). However, we do have:

Proposition C.15. *Consider* $\sigma \in \mathsf{Strat}(!\&[\Gamma, x : A, \Delta], O)$.
Then, there exists a unique $\Lambda(\sigma) \in \mathsf{Strat}(!\&[\Gamma, x : A, \Delta] \vdash O)$ *such that*

$$\varphi : \mathscr{C}^+(\sigma) \cong \mathscr{C}^+(\Lambda(\sigma))$$

and satisfying that $\partial_{\Lambda(\sigma)}(\varphi(x^\sigma)) = \Lambda^{\Gamma,\Delta}_{x:A,O}(\partial_\sigma(x^\sigma))$ *for all* $x^\sigma \in \mathscr{C}^+(\sigma)$.

Proof. *Existence.* We set $\Lambda(\sigma)$ as $\sigma[\Lambda^{\Gamma,\Delta}_{x:A,O}]$ even though $\Lambda^{\Gamma,\Delta}_{x:A,O}$ is not valid; that this is still well-defined follows directly from σ negative (see [Castellan and Clairambault 2020, Lemma 4.25]).
Uniqueness. Direct from Lemma C.5. □

Promotion. Next we define the promotion of $\sigma \in \mathsf{Strat}(!A, B)$.
First, for any arena A, we define the function

$$\begin{array}{rcl} \mathsf{dig}_A \quad : \quad !!A & \to & !A \\ \mathsf{e}_1 :: (\mathsf{e}_2 :: m) & \mapsto & \langle \mathsf{e}_1, \mathsf{e}_2 \rangle :: m \end{array}$$

yielding a map of event structures. If σ is a strategy, we write $\mathscr{C}^{+,\neq\emptyset}(\sigma)$ the set of +-covered, non-empty configurations of σ. Finally, for X a set we write $\mathsf{Fam}(X)$ for the set of families $(x_i)_{i \in I}$ where $x_i \in X$ and $I \subseteq \mathcal{E}$ is a finite subset of exponential signatures.
With these notations in place, we have:

Proposition C.16. *There is a strategy* $\sigma^\dagger \in \mathsf{Strat}(!A, !B)$, *unique up to iso, such that there is*

$$[-] : \mathsf{Fam}(\mathscr{C}^{+,\neq\emptyset}(\sigma)) \cong \mathscr{C}^+(\sigma^\dagger)$$

satisfying that for all $(x^{\mathsf{e}})_{\mathsf{e}\in E} \in \mathsf{Fam}(\mathscr{C}^{+,\neq\emptyset}(\sigma))$*, we have*

$$\partial_{\sigma^\dagger}([(x^{\mathsf{e}})_{\mathsf{e}\in E}]) = \mathsf{dig}\left(\biguplus_{\mathsf{e}\in E} \mathsf{e} :: x^{\mathsf{e}}_{!A}\right) \vdash \left(\biguplus_{\mathsf{e}\in E} \mathsf{e} :: x^{\mathsf{e}}_{B}\right)$$

writing $\partial_\sigma(x^{\mathsf{e}}) = x^{\mathsf{e}}_{!A} \vdash x^{\mathsf{e}}_{B}$ *for all* $\mathsf{e} \in E$*.*

Proof. *Existence.* Straightforward from [Castellan and Clairambault 2020, Definition 4.27] and renaming following dig.

Uniqueness. Direct from Lemma C.5. □

C.3.3 Strat *as an* IPA*-Structure: Primitives.*

Copycat strategies. We first address the three primitives arising as copycat-like strategies: variable, evaluation, and contraction.

Definition C.17. Consider $\Gamma, x : A, \Delta$ a semantic context. Then we set $\mathsf{Var}^{\Gamma,\Delta}_{x:A}$ as $\mathfrak{a}_A[\mathsf{Var}_x]$ where

$$\begin{array}{rcrcl} \mathsf{Var}_x & : & (A \vdash A) & \longrightarrow & (!\,\&[\Gamma, x : A, \Delta] \vdash A) \\ & & (\mathscr{r}_{\vdash}\mathsf{m}, \mathsf{s}, \mathsf{d}) & \mapsto & (\mathscr{r}_{\vdash}\mathsf{m}, \mathsf{s}, \mathsf{d}) \\ & & (\ell_{\vdash}\mathsf{m}, \mathsf{s}, \mathsf{d}) & \mapsto & (\ell_{\vdash}\mathscr{i}^{x}_{\&}\mathsf{m}, \blacklozenge :: \mathsf{s}, \mathsf{d}) \end{array}$$

Likewise, the evaluation morphism is simply by renaming.

Definition C.18. Consider A, O arenas with O well-opened.
Then we set $\mathsf{ev}_{A,O} = \mathfrak{a}_{A\multimap O}[\Omega]$ where Ω is that of Definition B.4 canonically extended to moves.

Finally, we define copycat. As in the main text, for simplicity we give the binary case.

Definition C.19. Consider A an arena. Then we set $\mathsf{c}_A = \mathfrak{a}_{!A\otimes !A}[\mathsf{c}] \in \mathsf{Strat}(!A, !A \otimes !A)$ where

$$\begin{array}{rcrcl} \mathsf{c} & : & (!A \otimes !A \vdash !A \otimes !A) & \rightharpoonup & (!A \vdash !A \otimes !A) \\ & & (\ell_{\vdash}\ell_{\otimes}\mathsf{m}, \mathsf{e} :: \mathsf{s}, \mathsf{d}) & \mapsto & (\ell_{\vdash}\mathsf{m}, (\ell_{!}\mathsf{e}) :: \mathsf{s}, \mathsf{d}) \\ & & (\ell_{\vdash}\mathscr{r}_{\otimes}\mathsf{m}, \mathsf{e} :: \mathsf{s}, \mathsf{d}) & \mapsto & (\ell_{\vdash}\mathsf{m}, (\mathscr{r}_{!}\mathsf{e}) :: \mathsf{s}, \mathsf{d}) \\ & & (\mathscr{r}_{\vdash}\mathsf{m}, \mathsf{s}, \mathsf{d}) & \mapsto & (\mathscr{r}_{\vdash}\mathsf{m}, \mathsf{s}, \mathsf{d}) \end{array}$$

For the unfolding, it will be convenient to have the following characterization:

Proposition C.20. *For any arena* A*,* $\mathscr{C}^{+}(\mathsf{c}_A) = \{\ell_{!}(x_{!A}) \uplus \mathscr{r}_{!}(y_{!A}) \vdash x_{!A} \otimes y_{!A} \mid x_{!A}, y_{!A} \in \mathscr{C}(!A)\}$.

Proof. Immediate by Lemma C.9 and definition. □

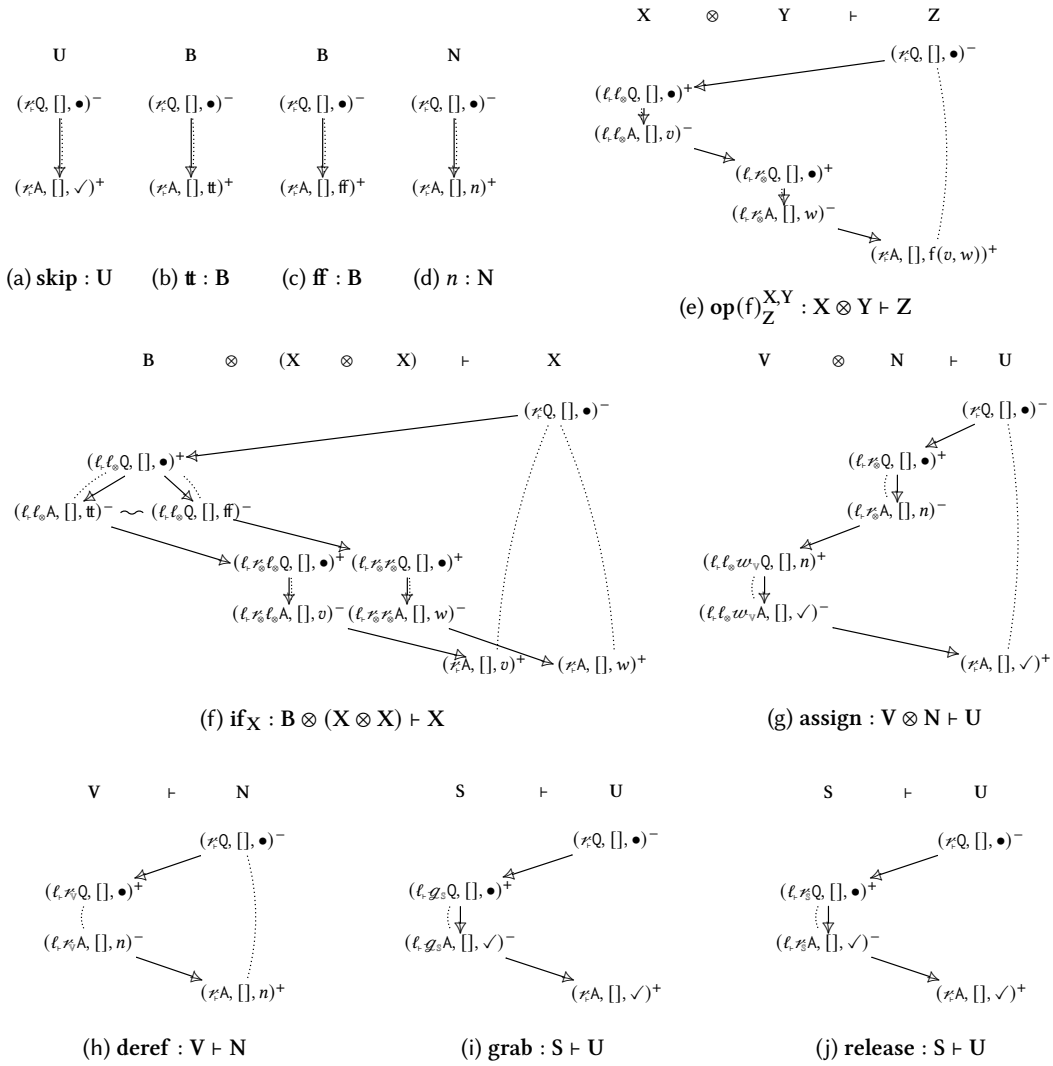
Constants, conditional, queries. The strategies are displayed in Figure 41. Note that some of these diagrams use a symbolic representation; whenever there is a branch starting with a negative move with some data, there actually is a branch for any instance of the data allowed in the game.

Let. We illustrate the strategy **let** in Figure 42. Note that there is a similar call to **!X** for all exponential signature $\mathsf{e} \in \mathcal{E}$.

Recursion. In [Castellan and Clairambault 2020; Castellan et al. 2019], the recursion combinator is obtained via the usual recipe in denotational semantics, as the least fixed point of

$$F \mapsto (f : O \to O \vdash F\,f)\,.$$

Let us give a direct description of the strategy obtained (the recursive equation gives a different choice of copy indices, which does not matter up to the equivalence of strategies in [Castellan and Clairambault 2020; Castellan et al. 2019] – the choice we use here allows for a lighter presentation).



(a) **skip** : **U** (b) **tt** : **B** (c) **ff** : **B** (d) n : **N** (e) $\mathbf{op}(\mathrm{f})^{\mathbf{X},\mathbf{Y}}_{\mathbf{Z}} : \mathbf{X} \otimes \mathbf{Y} \vdash \mathbf{Z}$

(f) $\mathbf{if}_{\mathbf{X}} : \mathbf{B} \otimes (\mathbf{X} \otimes \mathbf{X}) \vdash \mathbf{X}$ (g) **assign** : $\mathbf{V} \otimes \mathbf{N} \vdash \mathbf{U}$

(h) **deref** : $\mathbf{V} \vdash \mathbf{N}$ (i) **grab** : $\mathbf{S} \vdash \mathbf{U}$ (j) **release** : $\mathbf{S} \vdash \mathbf{U}$

Fig. 41. Basic strategies for IPA primitives

Let us start by drawing the strategy on **U**. We use particular exponential signatures generated by

$$\begin{aligned} () &:= \ell_{\uparrow}\blacklozenge \\ (\mathrm{e}_{n+1}, \mathrm{e}_n, \ldots, \mathrm{e}_1) &:= \pi_{\uparrow}\langle \mathrm{e}_{n+1}, (\mathrm{e}_n, \ldots, \mathrm{e}_1)\rangle\,, \end{aligned}$$

yielding $(\mathrm{e}_n, \ldots, \mathrm{e}_1) \in \mathcal{E}$ for each $\mathrm{e}_1, \ldots, \mathrm{e}_n \in \mathcal{E}$. With this convention, we draw the recursion combinator for **U** in Figure 43. Again the representation is symbolic, with similar branches for all $\mathrm{e}_1, \ldots, \mathrm{e}_{n+1} \in \mathcal{E}$. We leave in grey the answers, which always propagate back to the previous call.

We consider Y_O for O well-opened, so the recursion strategy in general has a spine exactly as the black part of Figure 43. The rest of the strategy is simple copycat behaviour; which may be simply described via the following direct characterization of the +-covered configurations of Y_O:

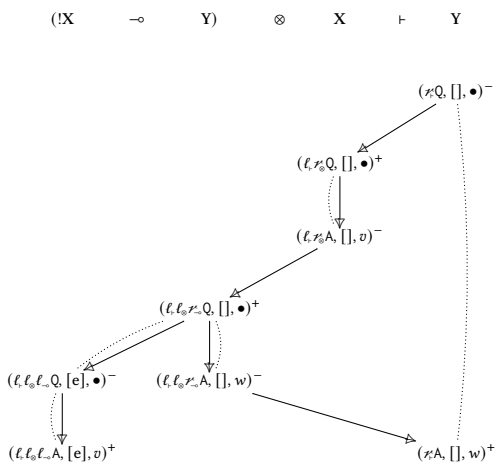


Fig. 42. $\mathbf{let}_{\mathbf{X},\mathbf{Y}} : (!\mathbf{X} \multimap \mathbf{Y}) \otimes \mathbf{X} \vdash \mathbf{Y}$

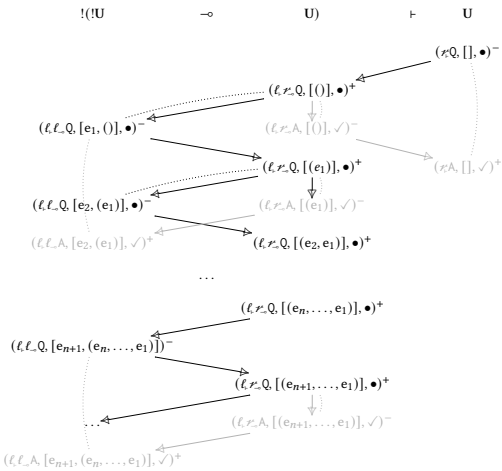


Fig. 43. Fixpoint combinator on $\mathbf{U}$

Proposition C.21. *For O a well-opened arena, the strategy Y_O has **events** a subset $|\mathsf{Y}_O| \subseteq |!(!O \multimap O) \vdash O|$, and **+-covered configurations** the compatible unions of configurations of the form*

$$\begin{array}{rcl} \emptyset \multimap (() :: x) & \vdash & x \\ (\mathsf{e}_{n+1} :: (\mathsf{e}_n, \ldots, \mathsf{e}_1) :: x) \multimap ((\mathsf{e}_{n+1}, \ldots, \mathsf{e}_1) :: x) & \vdash & \emptyset, \end{array}$$

*for $n \in \mathbb{N}, \mathsf{e}_1, \ldots, \mathsf{e}_{n+1} \in \mathcal{E}, x \in \mathscr{C}(O)$; **compatible** means that the union is in $\mathscr{C}(!(!O \multimap O) \vdash O))$.*

We slightly reformulate this proposition to give a more explicit description of +-covered configurations of Y_O. In the next lemma, we use the injection of $(\cdot) : \mathcal{E}^* \to \mathcal{E}$.

Lemma C.22. *There is an order-isomorphism between $\mathscr{C}^+(\mathsf{Y}_O)$ and tuples $\langle J \subseteq \mathcal{E}^+, z \in \mathscr{C}(O), (x_{\mathsf{s}} \in \mathscr{C}^{\neq\emptyset}(O))_{\mathsf{s}\in J}\rangle$ such that J is* suffix-*closed, and empty if z is.*[1] *The isomorphism sends $\langle J, z, (x_{\mathsf{s}})\rangle$ to*

$$(\emptyset \multimap () :: z) \uplus (\mathsf{e} :: (\mathsf{s}) :: x_{\mathsf{e}\cdot\mathsf{s}} \multimap (\mathsf{e} \cdot \mathsf{s}) :: x_{\mathsf{e}\cdot\mathsf{s}})_{\mathsf{e}\cdot\mathsf{s}\in J} \vdash z\,.$$

New reference. Next, we introduce the strategy for reference initialization.

The intuition is that $\mathbf{newref}_{\mathbb{X}} : !\mathbf{V} \multimap \mathbb{X} \vdash \mathbb{X}$ applies its argument to cell : $!\mathbf{V}$, the *memory cell*, which implements the stateful behaviour. Just like an actual memory, cell is inherently *sequential*: it treats read and write requests in some sequential order. In order to define it, we first define, for all $n \in \mathbb{N}$, cell_n as the language of non-empty prefixes of the infinite tree $\mathsf{cell}_n^{\emptyset}$, defined as:

$$\begin{array}{rcl} \mathsf{cell}_n^E & = & (\not{r}_{\mathbb{V}}\mathsf{Q}, [\mathsf{e}], \bullet)^- \cdot (\not{r}_{\mathbb{V}}\mathsf{A}, [\mathsf{e}], n)^+ \cdot \mathsf{cell}_n^{E\uplus\{\mathsf{e}\}} \\ & | & (w_{\mathbb{V}}\mathsf{Q}, [\mathsf{e}], k)^- \cdot (w_{\mathbb{V}}\mathsf{A}, [\mathsf{e}], \checkmark)^+ \cdot \mathsf{cell}_k^{E\uplus\{\mathsf{e}\}}\,, \end{array}$$

with $\mathsf{e} \notin E$. Intuitively, words in cell_n^E are alternating (*i.e.* Opponent and Player alternate) executions of read and write requests, for a memory cell initialized with value n: upon a read request, the memory cell returns n and carries on with cell_n. Upon a write request for k, the memory cell returns an acknowledgement and proceeds as cell_k. The set E propagates the set of exponential signatures already encountered: this lets us always pick fresh exponential signatures for new queries, ensuring words in cell_n^E are plays on $!\mathbf{V}$ in the sense of Definition 4.6. We may then define:

[1] A stack in J represents a call stack: a list $\mathsf{e}_n \cdot \mathsf{e}_{n-1} \cdot \ldots \cdot \mathsf{e}_1$ represents the calls made to the argument: e_1 is the first call made in the execution, and so on.

Definition C.23. We define a *prestrategy* precell : !**V** with components:

$$\begin{array}{rcl} |\mathsf{precell}| & = & \mathsf{cell}_0 \\ \leq_{\mathsf{precell}} & = & \sqsubseteq \\ s \#_{\mathsf{precell}} s' & \Leftrightarrow & \neg(s \sqsubseteq s' \vee s' \sqsubseteq s) \\ \partial_{\mathsf{precell}}(sa) & = & a\,, \end{array}$$

where $\sqsubseteq$ is the prefix ordering.

It is easy to see that this indeed defines a prestrategy. An illustration may be found in [Castellan and Clairambault 2020, Figure 39] (with a slightly different notation for moves). Next we define:

Definition C.24. We define a *strategy* cell : !**V** as: cell = $\mathfrak{a}_{!\mathbf{V}} \odot$ precell : !**V**.

Indeed, composition is well-defined on prestrategies, and the copycat envelope of a prestrategy is always a strategy [Castellan et al. 2017]. Intuitively, this wraps the sequential behaviour of cell by buffers, which exactly match the buffers of the Petri structure in Figure **??**.

To obtain the strategy for $\mathsf{newref}_{\mathbf{X}}$, we must add a copycat behaviour on **X**:

Definition C.25. We define a strategy $\mathsf{newref}_{\mathbf{X}} : !\mathbf{V} \multimap \mathbf{X} \vdash \mathbf{X}$ with components:

$$\begin{array}{rcl} |\mathsf{newref}_{\mathbf{X}}| & = & |\mathsf{cell}| + |\mathfrak{a}_{\mathbf{X}}| \\ \leq_{\mathsf{newref}_{\mathbf{X}}} & = & (\leq_{\mathsf{cell}} + \leq_{\mathfrak{a}_{\mathbf{X}}}) \\ & & \uplus\{(\mathscr{r}((\mathscr{r}_{\vdash}\mathsf{Q}, [\,], \bullet)), \ell(s)) \mid m \in |\mathsf{cell}|\} \\ & & \uplus\{(\mathscr{r}((\ell_{\vdash}\mathsf{Q}, [\,], \bullet)), \ell(s)) \mid m \in |\mathsf{cell}|\} \\ \#_{\mathsf{newref}_{\mathbf{X}}} & = & \#_{\mathsf{cell}} + \#_{\mathfrak{a}_{\mathbf{X}}} \\ \partial_{\mathsf{newref}_{\mathbf{X}}}(\ell(m)) & = & \ell_{\vdash}\ell_{\multimap}\partial_{\mathsf{cell}}(m) \\ \partial_{\mathsf{newref}_{\mathbf{X}}}(\mathscr{r}(\ell_{\vdash}m)) & = & \ell_{\vdash}\mathscr{r}_{\multimap}m \\ \partial_{\mathsf{newref}_{\mathbf{X}}}(\mathscr{r}(\mathscr{r}_{\vdash}m)) & = & \mathscr{r}_{\vdash}m \end{array}$$

So cell is plugged after the first Player move of $\mathfrak{a}_{\mathbf{X}}$. In other words, $\mathsf{newref}_{\mathbf{X}}$ first plays as copycat on **X**; and plays as cell on !**V** when it becomes available.

For the later unfolding, we shall use the following characterization of +-covered configurations:

Proposition C.26. *There is an order-isomorphism:*

$$\langle -, - \rangle : \mathscr{C}^{+}(\mathsf{precell}) \times \mathscr{C}^{\neq\emptyset}(\mathbf{X}) \cong \mathscr{C}^{+,\neq\emptyset}(\mathsf{newref}_{\mathbf{X}})$$

such that $\partial_{\mathsf{newref}_{\mathbf{X}}}(\langle s, x\rangle) = |s| \multimap x \vdash x \in \mathscr{C}(!\mathbf{V} \multimap \mathbf{X} \vdash \mathbf{X})$.

Proof. By Definition C.25, the characterization of confs. of composition, and Lemma C.9. □

Above, we implicitly use the one-to-one correspondence between $\mathscr{C}^{+}(\mathsf{precell})$ and even-length words in cell_0; and if s is an even length word in cell_0, then $|s| \in \mathscr{C}(!\mathbf{V})$ is its set of events.

In a +-covered configuration of newref, all read and write requests have been successfully handled. Proposition C.26 shows that besides the almost independent copycat behaviour on **X**, +-covered configurations of newref exactly correspond to some sequential ordering of these requests.

New semaphore. The interpretation of semaphores work exactly as for references. We first define the alternating behaviour of a semaphore as the language of non-empty prefixes of:

$$\begin{array}{rcll} \mathsf{lock}_0^E & = & (\mathscr{g}_{\mathbb{S}}\mathsf{Q}, [\mathsf{e}], \bullet)^- \cdot (\mathscr{g}_{\mathbb{S}}\mathsf{A}, [\mathsf{e}], \checkmark)^+ \cdot \mathsf{lock}_1^{E\uplus\{\mathsf{e}\}} & \mathsf{e} \notin E \\ \mathsf{lock}_n^E & = & (\mathscr{r}_{\mathbb{S}}\mathsf{Q}, [\mathsf{e}], \bullet)^- \cdot (\mathscr{r}_{\mathbb{S}}\mathsf{A}, [\mathsf{e}], \checkmark)^+ \cdot \mathsf{lock}_0^{E\uplus\{\mathsf{e}\}} & \mathsf{e} \notin E, n > 0 \end{array}$$

A semaphore with value 0 may be grabbed, carrying on with value 1. A semaphore with value $n > 0$ may be released, carrying on with value 0. As for references, the next step is to form:

Definition C.27. We define a *prestrategy* prelock : !S with components:

$$\begin{array}{rcl} |\mathsf{prelock}| & = & \mathsf{lock}_0 \\ \leq_{\mathsf{prelock}} & = & \sqsubseteq \\ s \#_{\mathsf{prelock}} s' & \Leftrightarrow & \neg(s \sqsubseteq s' \vee s' \sqsubseteq s) \\ \partial_{\mathsf{prelock}}(sa) & = & a\,. \end{array}$$

Definition C.28. We define a strategy lock : !S as lock = $\mathfrak{a}_{!\mathbf{S}} \odot$ prelock : !S.

Definition C.29. We define a strategy $\mathsf{newsem}_{\mathbf{X}} : !\mathbf{S} \multimap \mathbf{X} \vdash \mathbf{X}$ with components:

$$\begin{array}{rcl} |\mathsf{newsem}_{\mathbf{X}}| & = & |\mathsf{lock}| + |\mathfrak{a}_{\mathbf{X}}| \\ \leq_{\mathsf{newref}_{\mathbf{X}}} & = & (\leq_{\mathsf{lock}} + \leq_{\mathfrak{a}_{\mathbf{X}}}) \\ & & \uplus\{(\not{r}((\not{r}_{\vdash}\mathsf{Q}, [\,], \bullet)), \ell(s)) \mid m \in |\mathsf{lock}|\} \\ & & \uplus\{(\not{r}((\ell_{\vdash}\mathsf{Q}, [\,], \bullet)), \ell(s)) \mid m \in |\mathsf{lock}|\} \\ \#_{\mathsf{newsem}_{\mathbf{X}}} & = & \#_{\mathsf{lock}} + \#_{\mathfrak{a}_{\mathbf{X}}} \\ \partial_{\mathsf{newsem}_{\mathbf{X}}}(\ell(m)) & = & \ell_{\vdash}\ell_{\multimap}\partial_{\mathsf{lock}}(m) \\ \partial_{\mathsf{newsem}_{\mathbf{X}}}(\not{r}(\ell_{\vdash}m)) & = & \ell_{\vdash}\not{r}_{\multimap}m \\ \partial_{\mathsf{newsem}_{\mathbf{X}}}(\not{r}(\not{r}_{\vdash}m)) & = & \not{r}_{\vdash}m \end{array}$$

And, finally, we have a similar characterization of (non-empty) +-covered configurations:

Proposition C.30. *There is an order-isomorphism:*

$$\langle -, - \rangle : \mathscr{C}^{+}(\mathsf{prelock}) \times \mathscr{C}^{\neq\emptyset}(\mathbf{X}) \cong \mathscr{C}^{+,\neq\emptyset}(\mathsf{newsem}_{\mathbf{X}})$$

such that $\partial_{\mathsf{newsem}_{\mathbf{X}}}(\langle s, x\rangle) = |s| \multimap x \vdash x \in \mathscr{C}(!\mathbf{S} \multimap \mathbf{X} \vdash \mathbf{X})$.

This concludes the description of the IPA-structure of Strat.

C.3.4 Complement: on symmetry.

Games and strategies with symmetry. Thin concurrent games [Castellan et al. 2019] share with AJM games the fact that strategies play explicit copy indices – in [Castellan et al. 2019] and AJM games those are natural numbers, whereas here they are exponential signatures. The consequence is that in order to satisfy required equational laws (typically, making ! a well-behaved exponential modality), one must be able to consider strategies up to their choice of copy indices.

In concurrent games, this *reindexing* is handled by *event structures with symmetry*:

Definition C.31. A **event structure with symmetry** is $E = (|E|, \leq_E, \#_E, \mathscr{S}(E))$ is an es $(|E|, \leq_E, \#_E)$ with $\mathscr{S}(E)$ a set of bijections between configurations:

$$\theta : x \cong_E y$$

comprising all identity bijections, closed under composition and inverse, and satisfying further bisimulation-like properties, omitted here [Castellan et al. 2019].

In [Castellan and Clairambault 2020; Castellan et al. 2019], both games and strategies are event structures with symmetry. Intuitively, in a game A, we have $\theta : x \cong_A y$ in $\mathscr{S}(A)$ when θ is an order-isomorphism only affecting copy indices – in the terminology of this paper, it changes the exponential signatures in the exponential stack, but leaves all other components unchanged.

The symmetry on the game yields a more permissive equivalence on strategies: namely, a *weak isomorphism* $\varphi : \sigma \simeq \tau$ is an invertible map of event structure such that the triangle

$$\begin{tikzcd} \sigma \arrow[rr, "\varphi"] \arrow[dr, "\partial_\sigma"'] & & \tau \arrow[dl, "\partial_\tau"] \\ & A & \end{tikzcd}$$

commutes *up to symmetry*, defined as $\{(\partial_\sigma s, \partial_\tau(\varphi(s))) \mid s \in x\} \in \mathscr{S}(A)$ for all $x \in \mathscr{C}(\sigma)$. Weak isomorphism makes the exponential satisfy all the required laws (typically, making each $!A$ a commutative comonoid), which were not satisfied up to plain isomorphism.

In turn, composition must preserve weak isomorphism. But that holds only for strategies that are *uniform*, *i.e.* invariant under the choice of copy indices. In [Castellan and Clairambault 2020; Castellan et al. 2019], uniformity of strategies is ensured by also adjoining them a symmetry. On a strategy $\sigma : A$, the bisimulation-like properties of $\mathscr{S}(\sigma)$ ensures that if Opponent changes their copy indices, σ may change its copy indices accordingly, but not more. This makes $\simeq$ a congruence, and strategies up to $\simeq$ satisfy all the required equational laws to model higher-order languages. This issue is discussed at length in [Castellan et al. 2019].

Removing symmetry. The model developed in [Castellan and Clairambault 2020] is a structure StratSym with two equivalences $\cong$ (standard isomorphism) and $\simeq$ (weak isomorphism[2]). Both are preserved by all constructions, but the laws of Seely categories are satisfied with respect to $\simeq$ only.

Now, the first observation is that though $\simeq$ is crucial in establishing adequacy for StratSym (for instance, the β-law in IPA is validated by the interpretation only up to $\simeq$), the statement itself (Theorem 4.40 in [Castellan and Clairambault 2020]) is independent of the equivalence relation. So once adequacy is established we can ignore $\simeq$, and from [Castellan and Clairambault 2020] we get an IPA-structure StratSym$/\cong$ with an adequate interpretation of IPA.

The next point is that symmetries do not carry operational behaviour, they are merely there to witness uniformity so that $\simeq$ is a congruence. As mere uniformity witnesses, they can be safely forgotten once $\simeq$ is out of the picture. Concretely, in all operations involved in the interpretation, symmetries of the operand strategies are used in computing the symmetries of the resulting strategies only – the event structure itself never depends on the symmetries of operands. Consequently:

Proposition C.32. *There is a symmetry-forgetting* IPA*-functor* StratSym$/\cong \longrightarrow$ Strat.

From this, it follows by Lemma 3.3 that Strat is an adequate IPA-structure.

On a foundational level, it would be interesting to see how symmetries may be obtained by unfolding just as plain strategies. It would require setting up symmetries between histories of runs of Petri strategies as well. But this is not necessary for our purposes, so we leave that for later.

D THE IPA-STRUCTURE PStrat

D.1 The Precategory PStrat

The main step is to prove that Petri strategies are stable under composition.

D.1.1 Composition. Consider A, B, C arenas, $\boldsymbol{\sigma} : A \vdash B$ and $\boldsymbol{\tau} : B \vdash C$ Petri strategies.

The idea is simple: as Petri strategies, both $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$ abide by the rules of the game as long as the external Opponent does so. As no agent can be the first to break the rules, the whole interaction ends up correct. This kind of reasoning is very common in game semantics. To formalize it, the difficulty is not conceptual but purely notational: we need tools to project a run ρ on $\boldsymbol{\tau} \odot \boldsymbol{\sigma}$ to runs on $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$, and to an *interaction* in more familiar game-semantic terms:

Definition D.1. Consider the set $\mathsf{MInt} = \ell(\mathsf{Moves}) \uplus m(\mathsf{Moves}) \uplus r(\mathsf{Moves})$.
An **interaction** is a sequence $u \in \mathsf{MInt}^*$. We write Int for the set of all interactions.

As in traditional play-based game semantics, an interaction has three components: $\boldsymbol{\sigma}$ "plays" on ℓ, m, $\boldsymbol{\tau}$ "plays" on m, r, and the composite $\boldsymbol{\tau} \odot \boldsymbol{\sigma}$ "plays" on ℓ, r. Again as in game semantics, we shall *restrict* interactions to these various components. In more generality, we define:

[2]In fact, [Castellan and Clairambault 2020] uses a variant called *positive iso*, but the difference is irrelevant for this discussion.

$$\frac{t^0(\!|\alpha|\!) : \alpha \longmapsto_{\boldsymbol{\sigma}} \beta}{\ell^{\odot}(t^0)(\!|\ell^{\odot}(\alpha)|\!) : \ell^{\odot}(\alpha) \longmapsto_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \ell^{\odot}(\beta)} \qquad \frac{t^0(\!|\alpha|\!) : \alpha \longmapsto_{\boldsymbol{\tau}} \beta}{\mathscr{r}^{\odot}(t^0)(\!|\mathscr{r}^{\odot}(\alpha)|\!) : \mathscr{r}^{\odot}(\alpha) \longmapsto_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \mathscr{r}^{\odot}(\beta)}$$

$$\frac{t^+(\!|\alpha|\!) : \alpha \overset{\ell_{\vdash}(m)}{\longmapsto}_{\boldsymbol{\sigma}} \emptyset}{\ell^{\odot}(t^+)(\!|\ell^{\odot}(\alpha)|\!) : \ell^{\odot}(\alpha) \overset{\ell_{\vdash}(m)}{\longmapsto}_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \emptyset} \qquad \frac{t^+(\!|\alpha|\!) : \alpha \overset{\mathscr{r}_{\vdash}(m)}{\longmapsto}_{\boldsymbol{\tau}} \emptyset}{\mathscr{r}^{\odot}(t^+)(\!|\mathscr{r}^{\odot}(\alpha)|\!) : \mathscr{r}^{\odot}(\alpha) \overset{\mathscr{r}_{\vdash}(m)}{\longmapsto}_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \emptyset}$$

$$\frac{t^-(\!|(\mathsf{s},\mathsf{d})|\!) : \emptyset \overset{\ell_{\vdash}(m)}{\longmapsto}_{\boldsymbol{\sigma}} \beta}{\ell^{\odot}(t^-)(\!|(\mathsf{s},\mathsf{d})|\!) : \emptyset \overset{\ell_{\vdash}(m)}{\longmapsto}_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \ell^{\odot}(\beta)} \qquad \frac{t^-(\!|(\mathsf{s},\mathsf{d})|\!) : \emptyset \overset{\mathscr{r}_{\vdash}(m)}{\longmapsto}_{\boldsymbol{\tau}} \beta}{\mathscr{r}^{\odot}(t^-)(\!|(\mathsf{s},\mathsf{d})|\!) : \emptyset \overset{\mathscr{r}_{\vdash}(m)}{\longmapsto}_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \mathscr{r}^{\odot}(\beta)}$$

$$\frac{t^+(\!|\alpha|\!) : \alpha \overset{\mathscr{r}_{\vdash}(m)}{\longmapsto} \emptyset \qquad t^-(\!|(\mathsf{s},\mathsf{d})|\!) : \emptyset \overset{\ell_{\vdash}(m)}{\longmapsto} \beta}{(t^+ \circledast t^-)(\!|\ell^{\odot}(\alpha)|\!) : \ell^{\odot}(\alpha) \longmapsto \mathscr{r}^{\odot}(\beta)} \qquad \frac{t^-(\!|(\mathsf{s},\mathsf{d})|\!) : \emptyset \overset{\mathscr{r}_{\vdash}(m)}{\longmapsto} \beta \qquad t^+(\!|\alpha|\!) : \alpha \overset{\ell_{\vdash}(m)}{\longmapsto} \emptyset}{(t^- \circledast t^+)(\!|\mathscr{r}^{\odot}(\alpha)|\!) : \mathscr{r}^{\odot}(\alpha) \longmapsto \ell^{\odot}(\beta)}$$

Fig. 44. Description of instantiated transitions of $\boldsymbol{\tau} \odot \boldsymbol{\sigma}$

		$\mathsf{lbl}_{\circledast}$	$\pi_{\boldsymbol{\sigma}}$	$\pi_{\boldsymbol{\tau}}$
$\ell^{\odot}(t^0)(\!\|\ell^{\odot}(\alpha)\|\!)$	$\mapsto$		$t^0(\!\|\alpha\|\!)$	
$\mathscr{r}^{\odot}(t^0)(\!\|\mathscr{r}^{\odot}(\alpha)\|\!)$	$\mapsto$			$t^0(\!\|\alpha\|\!)$
$\ell^{\odot}(t^+)(\!\|\ell^{\odot}(\alpha)\|\!)$	$\mapsto$	$\ell(m)$	$t^+(\!\|\alpha\|\!)$	
$\mathscr{r}^{\odot}(t^+)(\!\|\mathscr{r}^{\odot}(\alpha)\|\!)$	$\mapsto$	$\mathscr{r}(m)$		$t^+(\!\|\alpha\|\!)$
$\ell^{\odot}(t^-)(\!\|(\mathsf{s},\mathsf{d})\|\!)$	$\mapsto$	$\ell(m)$	$t^-(\!\|(\mathsf{s},\mathsf{d})\|\!)$	
$\mathscr{r}^{\odot}(t^-)(\!\|(\mathsf{s},\mathsf{d})\|\!)$	$\mapsto$	$\mathscr{r}(m)$		$t^-(\!\|(\mathsf{s},\mathsf{d})\|\!)$
$(t^+ \circledast t^-)(\!\|\ell^{\odot}(\alpha)\|\!)$	$\mapsto$	$\mathscr{m}(m)$	$t^+(\!\|\alpha\|\!)$	$t^-(\!\|(\mathsf{s},\mathsf{d})\|\!)$
$(t^- \circledast t^+)(\!\|\mathscr{r}^{\odot}(\alpha)\|\!)$	$\mapsto$	$\mathscr{m}(m)$	$t^-(\!\|(\mathsf{s},\mathsf{d})\|\!)$	$t^+(\!\|\alpha\|\!)$

Fig. 45. Projections of instantiated transitions

Definition D.2. Consider $f : X \rightharpoonup Y$, and $s \in X^*$, we define $s \upharpoonright f \in Y^*$ the **restriction of** s **following** s as $\varepsilon \upharpoonright f = \varepsilon$, $sa \upharpoonright f = (s \upharpoonright f)f(a)$ if $f(a)$ is defined, and $sa \upharpoonright f = s \upharpoonright f$ otherwise.

A first direct application of this notion is to project $u \in \mathsf{Int}$ to its various components with

$$u_{\boldsymbol{\sigma}} = u \upharpoonright \ell_{\vdash}\ell^* \cup \mathscr{r}_{\vdash}\mathscr{m}^*, \qquad u_{\boldsymbol{\tau}} = u \upharpoonright \ell_{\vdash}\mathscr{m}^* \cup \mathscr{r}_{\vdash}\mathscr{r}, \qquad u_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} = u \upharpoonright \ell_{\vdash}\ell^* \cup \mathscr{r}_{\vdash}\mathscr{r}^*$$

where $\ell^* : \ell(\mathsf{Moves}) \rightharpoonup \mathsf{Moves}$ sends $\ell(m)$ to m and is undefined otherwise, and likewise for $\mathscr{m}^*$ and $\mathscr{r}^*$. Juxtaposition is function composition, and $\cup$ is the union of their graph.

We also use restriction to extract from a run ρ on $\boldsymbol{\tau} \odot \boldsymbol{\sigma}$ an *interaction*, and runs $\rho_{\boldsymbol{\sigma}}$ and $\rho_{\boldsymbol{\tau}}$ on $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$ respectively. But the partial functions involved are more complex, and require us to understand better the shape of instantiated transitions of $\boldsymbol{\tau} \odot \boldsymbol{\sigma}$:

Lemma D.3. *Consider $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$ Petri structures.*
Then, instantiated transitions of $\boldsymbol{\tau} \odot \boldsymbol{\sigma}$ are exactly as in Figure 44 – in the sense that that there is a one-to-one correspondence between instantiated transitions in the premises and in the conclusion.

Using this description, we define in Figure 45 partial functions $\mathsf{lbl}_{\circledast} : \mathscr{E}_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \rightharpoonup \mathsf{MInt}$, $\pi_{\boldsymbol{\sigma}} : \mathscr{E}_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \rightharpoonup \mathscr{E}_{\boldsymbol{\sigma}}$ and $\pi_{\boldsymbol{\tau}} : \mathscr{E}_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \rightharpoonup \mathscr{E}_{\boldsymbol{\tau}}$ extracting various data from instantiated transitions, following the characterization of instantiated transitions of $\boldsymbol{\tau} \odot \boldsymbol{\sigma}$ given in Figure 44.

Finally, those projection functions are extended to instantiated transitions in context via:

$$\begin{array}{rcccl} \mathsf{lbl}_{\circledast} & : & \mathscr{F}_{\tau\odot\sigma} & \rightharpoonup & \mathsf{MInt} \\ & & \mathbf{e} \uplus \gamma & \mapsto & \mathsf{lbl}_{\circledast}(\mathbf{e}) \\ \pi_\sigma & : & \mathscr{F}_{\tau\odot\sigma} & \rightharpoonup & \mathscr{F}_\sigma \\ & & \mathbf{e} \uplus (\gamma +^{\odot} \gamma') & \mapsto & \pi_\sigma(\mathbf{e}) \uplus \gamma \end{array}$$

and symmetrically for $\pi_\tau : \mathscr{F}_{\tau\odot\sigma} \rightharpoonup \mathscr{F}_\tau$. Using these, from a run $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\tau\odot\sigma} \alpha$ we extract:

$$\mathsf{Int}(\rho) = \rho \upharpoonright \mathsf{lbl}_{\circledast} \qquad \rho_\sigma = \rho \upharpoonright \pi_\sigma\,, \qquad \rho_\tau = \rho \upharpoonright \pi_\tau\,,$$

which allow us to prove the following property:

Lemma D.4. *Consider $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\tau\odot\sigma} \alpha$. Then, $\alpha = \alpha_\sigma +^{\odot} \alpha_\tau$ and*

$$\rho_\sigma : \emptyset \longrightarrow\!\!\!\!\rightarrow \alpha_\sigma\,, \qquad \rho_\tau : \emptyset \longrightarrow\!\!\!\!\rightarrow \alpha_\tau$$

where $\mathsf{play}(\rho) = \mathsf{Int}(\rho)_{\tau\odot\sigma}$, $\mathsf{play}(\rho_\sigma) = \mathsf{Int}(\rho)_\sigma$ *and* $\mathsf{play}(\rho_\tau) = \mathsf{Int}(\rho)_\tau$.

Moreover, $\mathsf{Coll}(\rho) = \mathsf{Coll}(\rho_\sigma) +^{\odot} \mathsf{Coll}(\rho_\tau)$.

Proof. A lengthy and grindy induction on ρ. For ρ empty this is clear, otherwise we reason by cases on the last instantiated transition in context of ρ, following the Figure 44.

Consider first that we have $\rho' = \rho(\ell^{\odot}(t^0)(\!|\ell^{\odot}(\mu)|\!) \uplus \gamma) : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\tau\odot\sigma} \beta$ where $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\tau\odot\sigma} \alpha$,

$$\ell^{\odot}(t^0)(\!|\ell^{\odot}(\alpha)|\!) : \ell^{\odot}(\mu) \longmapsto_{\tau\odot\sigma} \ell^{\odot}(\nu)\,, \qquad \ell^{\odot}(t^0)(\!|\ell^{\odot}(\alpha)|\!) \uplus \gamma : \alpha \longrightarrow_{\tau\odot\sigma} \beta$$

with necessarily $\alpha = \ell^{\odot}(\mu) \uplus \gamma$ and $\beta = \ell^{\odot}(\nu) \uplus \gamma$, and $t^0(\!|\mu|\!) : \mu \longmapsto_\sigma \nu$. By IH, $\alpha = \alpha_\sigma +^{\odot} \alpha_\tau$ with

$$\rho_\sigma : \emptyset \longrightarrow\!\!\!\!\rightarrow_\sigma \alpha_\sigma\,, \qquad \rho_\tau : \emptyset \longrightarrow\!\!\!\!\rightarrow_\tau \alpha_\tau\,,$$

so in particular that entails that $\gamma = \gamma_\sigma +^{\odot} \gamma_\tau$ with $\alpha_\sigma = \mu \uplus \gamma_\sigma$. Now, we have

$$t^0(\!|\mu|\!) \uplus \gamma_\sigma : \alpha_\sigma \longrightarrow_\sigma \beta_\sigma$$

writing $\beta_\sigma = \nu \uplus \gamma_\sigma$. Writing $\beta_\tau = \alpha_\tau = \gamma_\tau$, we have

$$(\rho(\ell^{\odot}(t^0)(\!|\ell^{\odot}(\mu)|\!) \uplus \gamma))_\sigma = \rho_\sigma(t^0(\!|\mu|\!) \uplus \gamma_\sigma) : \emptyset \longrightarrow\!\!\!\!\rightarrow_\sigma \beta_\sigma$$

and $(\rho(\ell^{\odot}(t^0)(\!|\ell^{\odot}(\mu)|\!) \uplus \gamma))_\tau = \rho_\tau : \emptyset \longrightarrow\!\!\!\!\rightarrow_\tau \beta_\tau$. Moreover $\mathsf{play}(\rho') = \mathsf{play}(\rho)$, $\mathsf{Int}(\rho') = \mathsf{Int}(\rho)$, $\mathsf{play}(\rho'_\sigma) = \mathsf{play}(\rho_\sigma)$ and $\mathsf{play}(\rho'_\tau) = \mathsf{play}(\rho_\tau)$, making required properties obvious. Finally, $\mathsf{Coll}(\rho') = \mathsf{Coll}(\rho) \cup \ell^{\odot}(\nu)$, $\mathsf{Coll}(\rho'_\sigma) = \mathsf{Coll}(\rho_\sigma) \cup \nu$ and $\mathsf{Coll}(\rho'_\tau) = \mathsf{Coll}(\rho_\tau)$, so $\mathsf{Coll}(\rho') = \mathsf{Coll}(\rho'_\sigma) +^{\odot} \mathsf{Coll}(\rho'_\tau)$ follows from IH. The case of a neutral transition from τ is symmetric.

Next, consider that $\rho' = \rho(\ell^{\odot}(t^+)(\!|\ell^{\odot}(\alpha)|\!) \uplus \gamma) : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\tau\odot\sigma} \beta$, where $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\tau\odot\sigma} \alpha$,

$$\ell^{\odot}(t^+)(\!|\ell^{\odot}(\mu)|\!) : \ell^{\odot}(\mu) \overset{\ell_+(m)}{\longmapsto}_{\tau\odot\sigma} \emptyset\,, \qquad \ell^{\odot}(t^+)(\!|\ell^{\odot}(\alpha)|\!) \uplus \gamma : \alpha \longrightarrow_{\tau\odot\sigma} \beta$$

where necessarily $\alpha = \ell^{\odot}(\mu) \uplus \gamma$ and $\beta = \gamma$, and $t^+(\!|\mu|\!) : \mu \overset{\ell_+(m)}{\longmapsto}_\sigma \emptyset$. Now, by IH, $\alpha = \alpha_\sigma +^{\odot} \alpha_\tau$ with

$$\rho_\sigma : \emptyset \longrightarrow\!\!\!\!\rightarrow_\sigma \alpha_\sigma\,, \qquad \rho_\tau : \emptyset \longrightarrow\!\!\!\!\rightarrow_\tau \alpha_\tau\,,$$

so in particular that entails $\gamma = \gamma_\sigma +^{\odot} \gamma_\tau$ with $\alpha_\sigma = \mu \uplus \gamma_\sigma$. Now, we have

$$t^+(\!|\mu|\!) \uplus \gamma_\sigma : \alpha_\sigma \longrightarrow_\sigma \beta_\sigma$$

writing $\beta_\sigma = \gamma_\sigma$. Writing $\beta_\tau = \alpha_\tau = \gamma_\tau$, we have

$$(\rho(\ell^{\odot}(t^+)(\!|\ell^{\odot}(\mu)|\!) \uplus \gamma))_\sigma = \rho_\sigma(t^+(\!|\mu|\!) \uplus \gamma_\sigma) : \emptyset \longrightarrow\!\!\!\!\rightarrow_\sigma \beta_\sigma$$

and $(\rho(\ell^{\odot}(t^+)(\!|\ell^{\odot}(\mu)|\!) \uplus \gamma))_\tau = \rho_\tau : \emptyset \longrightarrow\!\!\!\!\rightarrow_\tau \beta_\tau$. Moreover, $\mathsf{play}(\rho') = \mathsf{play}(\rho)\ell_+(m)$, $\mathsf{Int}(\rho') = \mathsf{Int}(\rho)\ell(m)$, $\mathsf{play}(\rho'_\sigma) = \mathsf{play}(\rho_\sigma)\ell_+(m)$ and $\mathsf{play}(\rho'_\tau) = \mathsf{play}(\rho_\tau)$, making the required properties clear. Finally, $\mathsf{Coll}(\rho') = \mathsf{Coll}(\rho), \mathsf{Coll}(\rho'_\sigma) = \mathsf{Coll}(\rho_\sigma)$ and $\mathsf{Coll}(\rho'_\tau) = \mathsf{Coll}(\rho_\tau)$, so $\mathsf{Coll}(\rho') = \mathsf{Coll}(\rho'_\sigma) +^{\odot} \mathsf{Coll}(\rho'_\tau)$ follows from IH. The case of a positive transition from τ is symmetric.

Next, consider that we have $\rho' = \rho(\mathscr{r}^{\odot}(t^-)(\!|(\mathrm{s},\mathrm{d})|\!) \uplus \gamma) : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \beta$, where $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \alpha$,

$$\ell^{\odot}(t^-)(\!|(\mathrm{s},\mathrm{d})|\!) : \emptyset \overset{\ell_{\vdash}(m)}{\longmapsto} \ell^{\odot}(\nu)\,, \qquad\qquad \ell^{\odot}(t^-)(\!|(\mathrm{s},\mathrm{d})|\!) \uplus \gamma : \alpha \longrightarrow_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \beta$$

where necessarily, $\alpha = \gamma$ and $\beta = \ell^{\odot}(\nu) \uplus \gamma$, and $t^-(\!|(\mathrm{s},\mathrm{d})|\!) : \emptyset \overset{\ell_{\vdash}(m)}{\longmapsto}_{\boldsymbol{\sigma}} \nu$. By IH, $\alpha = \alpha_{\boldsymbol{\sigma}} +^{\odot} \alpha_{\boldsymbol{\tau}}$ with

$$\rho_{\boldsymbol{\sigma}} : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\sigma}} \alpha_{\boldsymbol{\sigma}}\,, \qquad \rho_{\boldsymbol{\tau}} : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\tau}} \alpha_{\boldsymbol{\tau}}\,,$$

so that $\gamma = \gamma_{\boldsymbol{\sigma}} +^{\odot} \gamma_{\boldsymbol{\tau}}$ with $\gamma_{\boldsymbol{\sigma}} = \alpha_{\boldsymbol{\sigma}}$, $\gamma_{\boldsymbol{\tau}} = \alpha_{\boldsymbol{\tau}}$. Writing $\beta_{\boldsymbol{\sigma}} = \gamma_{\boldsymbol{\sigma}} \uplus \nu$ and $\beta_{\boldsymbol{\tau}} = \gamma_{\boldsymbol{\tau}} = \alpha_{\boldsymbol{\tau}}$, we have

$$(\rho(\mathscr{r}^{\odot}(t^-)(\!|(\mathrm{s},\mathrm{d})|\!) \uplus \gamma))_{\boldsymbol{\sigma}} = \rho_{\boldsymbol{\sigma}}(t^-(\!|(\mathrm{s},\mathrm{d})|\!) \uplus \gamma_{\boldsymbol{\sigma}}) : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\sigma}} \beta_{\boldsymbol{\sigma}}$$

and $(\rho(\mathscr{r}^{\odot}(t^-)(\!|(\mathrm{s},\mathrm{d})|\!) \uplus \gamma))_{\boldsymbol{\tau}} = \rho_{\boldsymbol{\tau}} : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\tau}} \beta_{\boldsymbol{\tau}}$. Moreover, $\mathsf{play}(\rho') = \mathsf{play}(\rho)\ell_{\vdash}(m)$, $\mathsf{Int}(\rho') = \mathsf{Int}(\rho)\ell(m)$, $\mathsf{play}(\rho'_{\boldsymbol{\sigma}}) = \mathsf{play}(\rho_{\boldsymbol{\sigma}})\ell_{\vdash}(m)$ and $\mathsf{play}(\rho'_{\boldsymbol{\tau}}) = \mathsf{play}(\rho_{\boldsymbol{\tau}})$, making the required properties clear. Finally, $\mathsf{Coll}(\rho') = \mathsf{Coll}(\rho) \cup \ell^{\odot}(\nu)$, with $\mathsf{Coll}(\rho'_{\boldsymbol{\sigma}}) = \mathsf{Coll}(\rho_{\boldsymbol{\sigma}}) \cup \nu$ and $\mathsf{Coll}(\rho'_{\boldsymbol{\tau}}) = \mathsf{Coll}(\rho_{\boldsymbol{\tau}})$, so $\mathsf{Coll}(\rho') = \mathsf{Coll}(\rho'_{\boldsymbol{\sigma}}) +^{\odot} \mathsf{Coll}(\rho'_{\boldsymbol{\tau}})$ follows from IH. Negative transitions from $\boldsymbol{\tau}$ are symmetric.

Consider finally $\rho' = \rho((t^+ \circledast t^-)(\!|\ell^{\odot}(\mu)|\!) \uplus \gamma : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\tau\odot\boldsymbol{\sigma}} \beta$ where $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \alpha$,

$$(t^+ \circledast t^-)(\!|\ell^{\odot}(\mu)|\!) : \ell^{\odot}(\mu) \longmapsto_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \mathscr{r}^{\odot}(\nu)\,, \qquad\qquad (t^+ \circledast t^-)(\!|\ell^{\odot}(\mu)|\!) \uplus \gamma : \alpha \longrightarrow_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \beta$$

where necessarily, $\alpha = \ell^{\odot}(\mu) \uplus \gamma$, $\beta = \mathscr{r}^{\odot}(\nu) \uplus \gamma$, and where we necessarily have

$$t^+(\!|\mu|\!) : \mu \overset{\mathscr{r}_{\vdash}(m)}{\longmapsto}_{\boldsymbol{\sigma}} \emptyset\,, \qquad t^-(\!|(\mathrm{s},\mathrm{d})|\!) : \emptyset \overset{\ell_{\vdash}(m)}{\longmapsto}_{\boldsymbol{\tau}} \nu\,,$$

with $\delta_{\boldsymbol{\sigma}}\langle t^+\rangle(\mu) = (\mathrm{s},\mathrm{d})$, $\ell_{\vdash}(m) = (\ell_{\vdash}(\mathrm{m}),\mathrm{s},\mathrm{d})$ where $\ell_{\vdash}(\mathrm{m}) = \partial_{\boldsymbol{\sigma}}(t^+)$; and $\delta_{\boldsymbol{\tau}}\langle t^-\rangle(\mathrm{s},\mathrm{d}) = \nu$. Now,

$$\rho_{\boldsymbol{\sigma}} : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\sigma}} \alpha_{\boldsymbol{\sigma}}\,, \qquad \rho_{\boldsymbol{\tau}} : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\tau}} \alpha_{\boldsymbol{\tau}}$$

for $\alpha = \alpha_{\boldsymbol{\sigma}} +^{\odot} \alpha_{\boldsymbol{\tau}}$ by IH, and necessarily $\gamma = \gamma_{\boldsymbol{\sigma}} +^{\odot} \gamma_{\boldsymbol{\tau}}$ with $\alpha_{\boldsymbol{\sigma}} = \mu \uplus \gamma_{\boldsymbol{\sigma}}$ and $\alpha_{\boldsymbol{\tau}} = \gamma_{\boldsymbol{\tau}}$. Writing $\beta_{\boldsymbol{\sigma}} = \gamma_{\boldsymbol{\sigma}}$ and $\beta_{\boldsymbol{\tau}} = \gamma_{\boldsymbol{\tau}} \uplus \nu$, we have $\beta = \beta_{\boldsymbol{\sigma}} \uplus \beta_{\boldsymbol{\tau}}$. Hence, we can form the projected runs

$$\rho'_{\boldsymbol{\sigma}} = \rho_{\boldsymbol{\sigma}}(t^+(\!|\mu|\!) \uplus \gamma_{\boldsymbol{\sigma}}) : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\sigma}} \beta_{\boldsymbol{\sigma}}\,, \qquad\qquad \rho'_{\boldsymbol{\tau}} = \rho_{\boldsymbol{\tau}}(t^-(\!|(\mathrm{s},\mathrm{d})|\!) \uplus \gamma_{\boldsymbol{\tau}}) : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\tau}} \beta_{\boldsymbol{\tau}}\,,$$

satisfying $\mathsf{play}(\rho') = \mathsf{play}(\rho)$, $\mathsf{Int}(\rho') = \mathsf{Int}(\rho)\mathscr{m}(m)$, $\mathsf{play}(\rho'_{\boldsymbol{\sigma}}) = \mathsf{play}(\rho_{\boldsymbol{\sigma}})\mathscr{r}_{\vdash}(m)$ and $\mathsf{play}(\rho'_{\boldsymbol{\tau}}) = \mathsf{play}(\rho_{\boldsymbol{\tau}})\ell_{\vdash}(m)$ from which the required verifications are immediate. Finally, $\mathsf{Coll}(\rho') = \mathsf{Coll}(\rho) \cup \mathscr{r}^{\odot}(\nu)$ with $\mathsf{Coll}(\rho'_{\boldsymbol{\sigma}}) = \mathsf{Coll}(\rho_{\boldsymbol{\sigma}})$ and $\mathsf{Coll}(\rho'_{\boldsymbol{\tau}}) = \mathsf{Coll}(\rho_{\boldsymbol{\tau}}) \cup \nu$, so $\mathsf{Coll}(\rho') = \mathsf{Coll}(\rho'_{\boldsymbol{\sigma}}) +^{\odot} \mathsf{Coll}(\rho'_{\boldsymbol{\tau}})$ follows from IH. The case $t^- \circledast t^+$ is symmetric, concluding the proof. □

Using this lemma, we shall now prove that a valid run of $\boldsymbol{\tau} \odot \boldsymbol{\sigma}$ projects to valid runs on $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$. For this we will also exploit the following easy lemma.

Lemma D.5. *Consider A, B arenas, and $s \in |A \vdash B|^*$.*
Then, $s \in \mathsf{Plays}(A \vdash B)$ iff $s \upharpoonright \ell_{\vdash}^* \in \mathsf{Plays}(A)$ and $s \upharpoonright \mathscr{r}_{\vdash}^* \in \mathsf{Plays}(B)$.

Proof. Straightforward. □

Lemma D.6. *Consider $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \alpha$ such that $\mathsf{play}(\rho) \in \mathsf{Plays}(A \vdash C)$.*
Then, $\mathsf{play}(\rho_{\boldsymbol{\sigma}}) \in \mathsf{Plays}(A \vdash B)$ and $\mathsf{play}(\rho_{\boldsymbol{\tau}}) \in \mathsf{Plays}(B \vdash C)$.

Proof. By induction on ρ. For ρ empty this is clear.
Consider first that we have $\rho' = \rho(\ell^{\odot}(t^0)(\!|\ell^{\odot}(\mu)|\!) \uplus \gamma) : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \beta$ where $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \alpha$,

$$\ell^{\odot}(t^0)(\!|\ell^{\odot}(\alpha)|\!) : \ell^{\odot}(\mu) \longmapsto_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \ell^{\odot}(\nu)\,, \qquad\qquad \ell^{\odot}(t^0)(\!|\ell^{\odot}(\alpha)|\!) \uplus \gamma : \alpha \longrightarrow_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \beta\,,$$

in that case $\mathsf{play}(\rho'_{\boldsymbol{\sigma}}) = \mathsf{play}(\rho_{\boldsymbol{\sigma}})$ and $\mathsf{play}(\rho'_{\boldsymbol{\tau}}) = \mathsf{play}(\rho_{\boldsymbol{\tau}})$, so the property follows from IH. The case of a neutral transition from $\boldsymbol{\tau}$ is symmetric.

Consider next that $\rho' = \rho(\ell^{\odot}(t^+)(\!|\ell^{\odot}(\alpha)|\!) \uplus \gamma) : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \beta$, where $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \alpha$,

$$\ell^{\odot}(t^+)(\!|\ell^{\odot}(\mu)|\!) : \ell^{\odot}(\mu) \overset{\ell_{\vdash}(m)}{\longmapsto}_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \emptyset\,, \qquad\qquad \ell^{\odot}(t^+)(\!|\ell^{\odot}(\alpha)|\!) \uplus \gamma : \alpha \longrightarrow_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \beta$$

where necessarily $\alpha = \ell^{\odot}(\mu) \uplus \gamma$ and $\beta = \gamma$, and $t^+(\!|\mu|\!) : \mu \overset{\ell_{\vdash}(m)}{\longmapsto}_{\sigma} \emptyset$. By IH, we have

$$\mathsf{play}(\rho_{\sigma}) \in \mathsf{Plays}(A \vdash B)\,, \qquad \mathsf{play}(\rho_{\tau}) \in \mathsf{Plays}(B \vdash C)$$

and as $\mathsf{play}(\rho'_{\tau}) = \mathsf{play}(\rho_{\tau})$, we have $\mathsf{play}(\rho'_{\tau}) \in \mathsf{Plays}(B \vdash C)$ as required. Now, we have

$$\rho_{\sigma} : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\sigma} \alpha_{\sigma}\,, \qquad t^+(\!|\mu|\!) \uplus \gamma_{\sigma} : \alpha_{\sigma} \overset{\ell_{\vdash}(m)}{\longrightarrow}_{\sigma} \beta_{\sigma}$$

with components named as in the proof of Lemma D.4, and with $s = \mathsf{play}(\rho_{\sigma}) \in \mathsf{Plays}(A \vdash B)$. Hence by condition *valid* of Petri strategies, $s\ell_{\vdash}(m) = \mathsf{play}(\rho'_{\sigma}) \in \mathsf{Plays}(A \vdash B)$, which concludes this case. The case of a positive transition from τ is symmetric.

Consider next that $\rho' = \rho(\varkappa^{\odot}(t^-)(\!|(\mathsf{s},\mathsf{d})|\!) \uplus \gamma) : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\tau\odot\sigma} \beta$, where $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\tau\odot\sigma} \alpha$,

$$\ell^{\odot}(t^-)(\!|(\mathsf{s},\mathsf{d})|\!) : \emptyset \overset{\ell_{\vdash}(m)}{\longmapsto} \ell^{\odot}(\nu)\,, \qquad \ell^{\odot}(t^-)(\!|(\mathsf{s},\mathsf{d})|\!) \uplus \gamma : \alpha \longrightarrow_{\tau\odot\sigma} \beta\,,$$

in that case $\mathsf{play}(\rho'_{\sigma}) = \mathsf{play}(\sigma)\ell_{\vdash}(m)^-$ and $\mathsf{play}(\rho'_{\tau}) = \mathsf{play}(\tau)$. By IH we have $\mathsf{play}(\rho'_{\sigma}) \in \mathsf{Plays}(A \vdash B)$ and $\mathsf{play}(\rho'_{\tau}) = \mathsf{play}(\rho_{\tau}) \in \mathsf{Plays}(B \vdash C)$. But by hypothesis, we have $\mathsf{play}(\rho') = \mathsf{play}(\rho)\ell_{\vdash}(m) \in \mathsf{Plays}(A \vdash C)$. By Lemma D.5, $\mathsf{play}(\rho)\ell_{\vdash}(m) \upharpoonright \ell_{\vdash}^* = (\mathsf{play}(\rho) \upharpoonright \ell_{\vdash}^*)m \in \mathsf{Plays}(A)$. But $\mathsf{play}(\rho'_{\sigma}) \upharpoonright \ell_{\vdash}^* = (\mathsf{play}(\rho) \upharpoonright \ell_{\vdash}^*)m \in \mathsf{Plays}(A)$, and $\mathsf{play}(\rho'_{\sigma}) \upharpoonright \varkappa_{\vdash}^* = \mathsf{play}(\rho_{\sigma}) \upharpoonright \varkappa_{\vdash}^*$, so $\mathsf{play}(\rho'_{\sigma}) \in \mathsf{Plays}(A \vdash B)$ by Lemma D.5. The case of a negative transition from τ is symmetric.

Consider finally $\rho' = \rho((t^+ \circledast t^-)(\!|\ell^{\odot}(\mu)|\!) \uplus \gamma : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\tau\odot\sigma} \beta$ where $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\tau\odot\sigma} \alpha$,

$$(t^+ \circledast t^-)(\!|\ell^{\odot}(\mu)|\!) : \ell^{\odot}(\mu) \longmapsto_{\tau\odot\sigma} \varkappa^{\odot}(\nu)\,, \qquad (t^+ \circledast t^-)(\!|\ell^{\odot}(\mu)|\!) \uplus \gamma : \alpha \longrightarrow_{\tau\odot\sigma} \beta$$

where necessarily, $\alpha = \ell^{\odot}(\mu) \uplus \gamma$, $\beta = \varkappa^{\odot}(\nu) \uplus \gamma$, and where we necessarily have

$$t^+(\!|\mu|\!) : \mu \overset{\varkappa(m)}{\longmapsto}_{\sigma} \emptyset\,, \qquad t^-(\!|(\mathsf{s},\mathsf{d})|\!) : \emptyset \overset{\ell_{\vdash}(m)}{\longmapsto}_{\tau} \nu\,,$$

with $\delta_{\sigma}\langle t^+\rangle(\mu) = (\mathsf{s},\mathsf{d})$, $\ell_{\vdash}(m) = (\ell_{\vdash}(\mathsf{m}),\mathsf{s},\mathsf{d})$ where $\ell_{\vdash}(\mathsf{m}) = \partial_{\sigma}(t^+)$; and $\delta_{\tau}\langle t^-\rangle(\mathsf{s},\mathsf{d}) = \nu$. By IH,

$$\mathsf{play}(\rho_{\sigma}) \in \mathsf{Plays}(A \vdash B)\,, \qquad \mathsf{play}(\rho_{\tau}) \in \mathsf{Plays}(B \vdash C)\,.$$

Summing up the situation on the side of σ, we have

$$\rho_{\sigma} : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\sigma} \alpha_{\sigma}\,, \qquad t^+(\!|\mu|\!) \uplus \gamma_{\sigma} : \alpha_{\sigma} \overset{\varkappa(m)}{\longrightarrow}_{\sigma} \beta_{\sigma}$$

with components as in the proof of Lemma D.4. By condition *valid* of Petri strategies, $\mathsf{play}(\rho'_{\sigma}) = \mathsf{play}(\rho_{\sigma})\varkappa_{\vdash}(m) \in \mathsf{Plays}(A \vdash B)$. By Lemma D.5, this entails $\mathsf{play}(\rho_{\sigma})\varkappa_{\vdash}(m) \upharpoonright \varkappa_{\vdash}^* = (\mathsf{play}(\rho_{\sigma}) \upharpoonright \varkappa_{\vdash}^*)m \in \mathsf{Plays}(B)$. Now, Lemma D.4 also entails $\mathsf{play}(\rho'_{\sigma}) = \mathsf{Int}(\rho)_{\sigma}$ and $\mathsf{play}(\rho'_{\tau}) = \mathsf{Int}(\rho)_{\tau}$. So:

$$\mathsf{play}(\rho'_{\sigma}) \upharpoonright \varkappa_{\vdash}^* = \mathsf{Int}(\rho) \upharpoonright m^* = \mathsf{play}(\rho'_{\tau}) \upharpoonright \ell_{\vdash}^*\,,$$

so that $\mathsf{play}(\rho'_{\tau}) \upharpoonright \ell_{\vdash}^* = (\mathsf{play}(\rho_{\sigma}) \upharpoonright \ell_{\vdash}^*)m \in \mathsf{Plays}(B)$. Now, we also have $\mathsf{play}(\rho'_{\tau}) \upharpoonright \varkappa_{\vdash}^* = \mathsf{play}(\rho_{\tau}) \upharpoonright \varkappa_{\vdash}^* \in \mathsf{Plays}(C)$ by Lemma D.5; so by Lemma D.5 again we deduce $\mathsf{play}(\rho'_{\tau}) \in \mathsf{Plays}(B \vdash C)$ as required. The case of a synchronized transition $t^- \circledast t^+$ is symmetric. □

We are finally in position to prove that Petri strategies are stable under composition.

Proposition D.7. *If $\sigma : A \vdash B$ and $\tau : B \vdash C$ are Petri strategies, then so is $\tau \odot \sigma : A \vdash C$. Moreover, if σ and τ are negative, so is $\tau \odot \sigma$.*

Proof. *Valid.* Consider $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\tau\odot\sigma} \alpha$ with $\mathsf{play}(\rho) \in \mathsf{Plays}(A \vdash C)$, and $\mathbf{f}^+ : \alpha \xrightarrow{m'} \beta$. *W.l.o.g.*, assume $m' = \ell_{\vdash}(m)$. By Lemma D.4, α decomposes as $\alpha_{\sigma} +^{\odot} \alpha_{\tau}$, and we have:

$$\rho_{\sigma} : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\sigma} \alpha_{\sigma}\,, \qquad \rho_{\tau} : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\tau} \alpha_{\tau}\,.$$

Now, $\mathbf{f}^+$ must have the form $\mathbf{f}^+ = (\ell^{\odot}(t^+)(\!|\ell^{\odot}(\mu)|\!)) \uplus \gamma$ with $t^+(\!|\mu|\!) : \mu \overset{\ell_{\vdash}(m)}{\longmapsto} \emptyset$, and $\beta = \gamma$. Hence,

$$t^+(\!|\mu|\!) \uplus \gamma_{\sigma} : \alpha_{\sigma} \overset{\ell_{\vdash}(m)}{\longrightarrow} \beta_{\sigma}\,,$$

with components named as in the proof of Lemma D.4. At this point we apply Lemma D.6, which ensures that $\mathsf{play}(\rho_{\boldsymbol{\sigma}}) \in \mathsf{Plays}(A \vdash B)$. Hence since $\boldsymbol{\sigma}$ is *valid*, $\mathsf{play}(\rho_{\boldsymbol{\sigma}})\ell_{\vdash}(m) \in \mathsf{Plays}(A \vdash B)$ as well. It follows by Lemma D.5 that $\mathsf{play}(\rho') = \mathsf{play}(\rho)\ell_{\vdash}(m) \in \mathsf{Plays}(A \vdash C)$ as well.

Receptive. Consider $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \alpha$ with $s = \mathsf{play}(\rho) \in \mathsf{Plays}(A \vdash C)$, and $sm' \in \mathsf{Plays}(A \vdash C)$ with m' negative – say *w.l.o.g.* that $m' = \ell_{\vdash}(m)^{-}$. By Lemma D.6, we have

$$\mathsf{play}(\rho_{\boldsymbol{\sigma}}) \in \mathsf{Plays}(A \vdash B)\,, \qquad \mathsf{play}(\rho_{\boldsymbol{\tau}}) \in \mathsf{Plays}(B \vdash C)\,,$$

and by Lemma D.6 we may deduce that $\mathsf{play}(\rho_{\boldsymbol{\sigma}})\ell_{\vdash}(m) \in \mathsf{Plays}(A \vdash B)$. By *receptive*, there is

$$\mathbf{e}^{-} = t^{-}(\!|(\mathsf{s},\mathsf{d})|\!) : \emptyset \overset{\ell_{\vdash}(m)}{\longmapsto}_{\boldsymbol{\sigma}} \beta$$

for $\beta \cap \alpha_{\boldsymbol{\sigma}} = \emptyset$. Hence $\ell^{\odot}(t^{-})(\!|(\mathsf{s},\mathsf{d})|\!) : \emptyset \overset{\ell_{\vdash}(m)}{\longmapsto}_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \ell^{\odot}(\beta)$ with $\ell^{\odot}(\beta) \cap \alpha = \emptyset$. For uniqueness, if

$$\mathbf{e}' : \emptyset \overset{\ell_{\vdash}(m)}{\longmapsto}_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \beta'$$

for some β'. Necessarily, $\mathbf{e}' = \ell^{\odot}(t')(\!|(\mathsf{s}',\mathsf{d}')|\!)$ for $t'(\!|(\mathsf{s}',\mathsf{d}')|\!) : \emptyset \overset{\ell_{\vdash}(m)}{\longmapsto}_{\boldsymbol{\sigma}} \beta''$ for $\beta' = \ell^{\odot}(\beta'')$. But by uniqueness of *receptivity* for $\boldsymbol{\sigma}$, we have $t'(\!|(\mathsf{s}',\mathsf{d}')|\!) = t(\!|(\mathsf{s},\mathsf{d})|\!)$, so that $t = t'$, $(\mathsf{s},\mathsf{d}) = (\mathsf{s}',\mathsf{d}')$.

Strongly safe. Consider $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \alpha$ with $\mathsf{play}(\rho) \in \mathsf{Plays}(A \vdash C)$, with a new instantiated transition in context $\mathbf{f}$ – we distinguish cases depending on its form.

Assume first $\mathbf{f} = \ell^{\odot}(t^{0})(\!|\ell^{\odot}(\mu)|\!) \uplus \gamma$ for $t^{0}(\!|\mu|\!) : \mu \longmapsto_{\boldsymbol{\sigma}} \nu$. By Lemma D.6, we have $\mathsf{play}(\rho_{\boldsymbol{\sigma}}) \in \mathsf{Plays}(A \vdash B)$, so that since $\boldsymbol{\sigma}$ is *strongly safe*, $\mathsf{new}(t^{0}(\!|\mu|\!))$ is fresh in $\rho_{\boldsymbol{\sigma}}$. But $\mathsf{new}(\ell^{\odot}(t^{0})(\!|\ell^{\odot}(\mu)|\!)) = \ell^{\odot}(\mathsf{new}(t^{0}(\!|\mu|\!)))$. Moreover, by Lemma D.4, $\mathsf{Coll}(\rho) = \mathsf{Coll}(\rho_{\boldsymbol{\sigma}}) +^{\odot} \mathsf{Coll}(\rho_{\boldsymbol{\tau}})$. So it follows that $\mathsf{new}(\mathbf{f})$ is fresh in ρ as required. The case of a neutral transition from $\boldsymbol{\tau}$ is symmetric.

Next assume $\mathbf{f} = \ell^{\odot}(t^{+})(\!|\ell^{\odot}(\mu)|\!) \uplus \gamma$. Then $\mathsf{new}(\mathbf{f}) = \emptyset$. Idem for a positive transition from $\boldsymbol{\tau}$.

Next assume $\mathbf{f} = \ell^{\odot}(t^{-})(\!|(\mathsf{s},\mathsf{d})|\!) \uplus \gamma : \alpha \overset{\ell_{\vdash}(m)}{\longrightarrow}_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \beta$ with $\mathsf{play}(\rho)\ell_{\vdash}(m) \in \mathsf{Plays}(A \vdash C)$. We have

$$t^{-}(\!|(\mathsf{s},\mathsf{d})|\!) : \emptyset \overset{(\ell_{\vdash}(m))}{\longmapsto} \nu\,,$$

and by Lemma D.6, we have $\mathsf{play}(\rho_{\boldsymbol{\sigma}}) \in \mathsf{Plays}(A \vdash B)$. From $\mathsf{play}(\rho)\ell_{\vdash}(m) \in \mathsf{Plays}(A \vdash C)$ and $\mathsf{play}(\rho_{\boldsymbol{\sigma}}) \in \mathsf{Plays}(A \vdash B)$, it follows easily via Lemma D.5 that $\mathsf{play}(\rho_{\boldsymbol{\sigma}})\ell_{\vdash}(m) \in \mathsf{Plays}(A \vdash B)$. So since $\boldsymbol{\sigma}$ is *strongly safe*, $\mathsf{new}(t^{-}(\!|(\mathsf{s},\mathsf{d})|\!))$ is fresh in $\rho_{\boldsymbol{\sigma}}$. We conclude as in the neutral case using Lemma D.4. The case of a negative transition from $\boldsymbol{\tau}$ is similar.

Finally, assume $\mathbf{f} = (t^{+} \circledast t^{-})(\!|\ell^{\odot}(\mu)|\!) \uplus \gamma$. Say that we have

$$(t^{+} \circledast t^{-})(\!|\ell^{\odot}(\mu)|\!) : \ell^{\odot}(\mu) \longmapsto_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \mathscr{r}^{\odot}(\nu)\,,$$

so that $\mathsf{new}(\mathbf{f}) = \mathscr{r}^{\odot}(\nu)$ – assume $t^{+}(\!|\mu|\!) : \mu \overset{\mathscr{r}^{\odot}(m)}{\longmapsto}_{\boldsymbol{\sigma}} \emptyset$ and $t^{-}(\!|(\mathsf{s},\mathsf{d})|\!) : \emptyset \overset{\ell_{\vdash}(m)}{\longmapsto}_{\boldsymbol{\tau}} \nu$. By Lemma D.5, $\mathsf{play}(\rho_{\boldsymbol{\tau}})\ell_{\vdash}(m) \in \mathsf{Plays}(B \vdash C)$. Therefore since $\boldsymbol{\tau}$ is *strongly safe*, $\mathsf{new}(t^{-}(\!|(\mathsf{s},\mathsf{d})|\!)) = \nu$ is fresh in $\rho_{\boldsymbol{\tau}}$. But by Lemma D.4, $\mathsf{Coll}(\rho) = \mathsf{Coll}(\rho_{\boldsymbol{\sigma}}) +^{\odot} \mathsf{Coll}(\rho_{\boldsymbol{\tau}})$, so $\mathsf{new}(\mathbf{f}) = \mathscr{r}^{\odot}(\nu)$ is fresh in ρ as required. The other synchronized case is symmetric.

Negative. Straightforward by inspection and negativity of $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$. □

D.1.2 Copycat. First, we characterize the markings of copycat reachable through a play:

Recall that $\mathcal{L}_{\mathfrak{cc}_A} = \mathsf{mult}(A)$, so that the set $\mathsf{TokIL}(\mathfrak{cc}_A)$ of tokils of $\mathfrak{cc}_A$ is in bijection with $|A|$. This lets us silently coerce a configuration $x \in \mathscr{C}(A)$ into a marking $x \in \mathscr{M}(\mathfrak{cc}_A)$. In order to show that copycat is a Petri strategy, we first characterize the markings reachable by rule-abiding runs:

Lemma D.8. *Consider A an arena, and $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\mathfrak{cc}_A} \alpha$ such that $\mathsf{play}(\rho) \in \mathsf{Plays}(A \vdash A)$. Then, $|\mathsf{play}(\rho)| = x \vdash y$ with $x, y \in \mathscr{C}(A)$ and $y \supseteq^{-} x \cap y \subseteq^{+} x$; and $\alpha = (y^{-} \setminus x) \uplus (x^{+} \setminus y)$. Moreover, $\mathsf{Coll}(\rho) = y^{-} \uplus x^{+}$, where x^{p} is the subset of $x \in \mathscr{C}(A)$ whose moves have polarity p.*

Proof. By induction on ρ. For ρ empty, this is clear. Consider $\rho' = \rho\mathbf{f} : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\mathbf{cc}_A} \beta$ with $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\mathbf{cc}_A} \alpha$, and $\mathsf{play}(\rho') \in \mathsf{Plays}(A \vdash A)$. We also have $\mathsf{play}(\rho) \in \mathsf{Plays}(A \vdash A)$, so by IH,

$$|\mathsf{play}(\rho)| = x \vdash y\,, \qquad y \supseteq^- x \cap y \subseteq^+ x\,, \qquad \alpha = (y^- \setminus x) \uplus (x^+ \setminus y)\,,$$

we reason by cases depending on $\mathbf{f}$. If $\mathbf{f} = (\mathsf{m}^+, \curvearrowright)(\!|\{(\mathsf{s},\mathsf{d})^{@\mathsf{m}}\}|\!) \uplus \gamma$, then $\alpha = \{(\mathsf{s},\mathsf{d})^{@\mathsf{m}}\} \uplus \gamma$, and

$$\begin{array}{rcl} |\mathsf{play}(\rho')| & = & x \vdash y \uplus \{(\mathsf{m},\mathsf{s},\mathsf{d})\} \\ \beta & = & \alpha \setminus \{(\mathsf{s},\mathsf{d})^{@\mathsf{m}}\}\,; \end{array}$$

since $\alpha = (y^- \setminus x) \uplus (x^+ \setminus y)$, $(\mathsf{s},\mathsf{d})^{@\mathsf{m}} \in \alpha$ and $(\mathsf{m},\mathsf{s},\mathsf{d})$ is positive, $(\mathsf{m},\mathsf{s},\mathsf{d})$ is positive and must be in x. It follows that the invariant is preserved. The case $\mathbf{f} = (\mathsf{m}^-, \ell)(\!|(\mathsf{s},\mathsf{d})^{@\mathsf{m}}|\!) \uplus \gamma$ is symmetric.

If $\mathbf{f} = (\mathsf{m}^-, \curvearrowright)(\!|(\mathsf{s},\mathsf{d})|\!) \uplus \gamma$, then $\alpha = \gamma$, and

$$\begin{array}{rcl} |\mathsf{play}(\rho')| & = & x \vdash (y \uplus \{(\mathsf{m},\mathsf{s},\mathsf{d})\}) \\ \beta & = & \alpha \uplus \{(\mathsf{s},\mathsf{d})^{@\mathsf{m}}\}\,; \end{array}$$

and $(\mathsf{m},\mathsf{s},\mathsf{d})$ is negative. Since $|\mathsf{play}(\rho')| = x \vdash (y \uplus \{(\mathsf{m},\mathsf{s},\mathsf{d})\})$, $(\mathsf{m},\mathsf{s},\mathsf{d}) \notin y$. So, it cannot be in x. The invariant directly follows. The case $\mathbf{f} = (\mathsf{m}^+, \ell)(\!|(\mathsf{s},\mathsf{d})@\mathsf{m}|\!) \uplus \gamma$ is symmetric. □

It is a direct application of this lemma to prove that copycat is a Petri strategy:

Proposition D.9. *For any arena A, $\mathbf{cc}_A : A \vdash A$ is a negative Petri strategy.*

Proof. *Valid.* Consider $s \in \mathsf{Plays}(A \vdash A)$ and

$$\rho : \emptyset \xrightarrow{s}\!\!\!\!\rightarrow \alpha\,, \qquad\qquad \mathbf{f}^+ : \alpha \xrightarrow{m} \beta\,,$$

and say *w.l.o.g.* that $m = \curvearrowright a$ for $a \in |A|$. This means that $\mathbf{f}^+ = (\mathsf{m}^+, \curvearrowright)(\!|\{(\mathsf{s},\mathsf{d})^{@\mathsf{m}}\}|\!) \uplus \gamma$, where $\alpha = \gamma \uplus \{(\mathsf{s},\mathsf{d})^{@\mathsf{m}}\}$, and $m = (\curvearrowright\mathsf{m},\mathsf{s},\mathsf{d})$. We must show that $s(\curvearrowright\mathsf{m},\mathsf{s},\mathsf{d}) \in \mathsf{Plays}(A \vdash A)$.

By Lemma D.8, $\alpha = (y^- \setminus x) \uplus (x^+ \setminus y)$ where $|s| = x \vdash y$. As $\mathsf{m} \in \mathsf{mult}^+(A)$, $(\mathsf{m},\mathsf{s},\mathsf{d}) \in x^+ \setminus y$. As $(\mathsf{m},\mathsf{s},\mathsf{d}) \notin y$, we have condition *non-repetitive*. Next, we show $|sm| = x \vdash (y \uplus \{a\})$ is *down-closed*. Consider $a' \rightarrowtriangle_A a$. Since A is alternating, a' is negative. Since $x \in \mathscr{C}(A)$ and $a \in x$, we must have $a' \in x$ as well. But by Lemma D.8 we have $y \supseteq^- x \cap y \subseteq^+ x$, so $a' \in y$ as well. Finally, from conditions *locally conflicting*, *alternating* and *negative* of arenas, pairs of events in minimal conflict have the same polarity. Therefore, as $x \cap y \subseteq^- x$ and $x \cap y \subseteq^+ y$, we have $x \cup y$ consistent, so in particular $y \cup \{a\} \in \mathscr{C}(A)$. It follows that $sm \in \mathsf{Plays}(A \vdash A)$ as needed.

Receptive. Consider $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\mathbf{cc}_A} \alpha$ with $\mathsf{play}(\rho) = s \in \mathsf{Plays}(A \vdash A)$ and $sm^- \in \mathsf{Plays}(A \vdash A)$. *W.l.o.g.* consider $m = \curvearrowright(a)$. Decompose $a = (\mathsf{m},\mathsf{s},\mathsf{d})$ for $\mathsf{m}^- \in \mathsf{mult}(A)$ and token (s,d). By inspection there is a unique matching instantiated transition, namely $(\mathsf{m}^-, \curvearrowright)(\!|(\mathsf{s},\mathsf{d})|\!) : \emptyset \overset{m}{\longmapsto} \{(\mathsf{s},\mathsf{d})^{@\mathsf{m}}\}$. Moreover, by Lemma D.8 we have $\alpha = (y^- \setminus x) \uplus (x^+ \setminus y)$ where $|s| = x \vdash y$. But as $sm^- \in \mathsf{Plays}(A \vdash A)$, by *non-repetitive* we have $m^- \notin y$. It follows that $(\mathsf{s},\mathsf{d})^{@\mathsf{m}} \notin \alpha$ as required.

Strongly safe. Consider $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\mathbf{cc}_A} \alpha$ with $\mathsf{play}(\rho) = s \in \mathsf{Plays}(A \vdash A)$ extended with $\mathbf{f}$. If $\mathbf{f}$ is positive, then $\mathsf{new}(\mathbf{f}) = \emptyset$ and there is nothing to prove. Hence, consider *w.l.o.g.* $\mathbf{f} = (\mathsf{m}^-, \curvearrowright)(\!|(\mathsf{s},\mathsf{d})|\!) \uplus \alpha$ with $s(\curvearrowright(\mathsf{m}),\mathsf{s},\mathsf{d})^- \in \mathsf{Plays}(A \vdash A)$. Then, $\mathsf{new}(\mathbf{f}) = \{(\mathsf{s},\mathsf{d})^{@\mathsf{m}}\}$. Write $|s| = x \vdash y \in \mathscr{C}(A \vdash A)$. By *non-repetitive*, $(\mathsf{m},\mathsf{s},\mathsf{d}) \notin y$. By Lemma D.8, $\mathsf{Coll}(\rho) = y^- \uplus x^+$; but as $(\mathsf{m},\mathsf{s},\mathsf{d}) \notin y$ and $(\mathsf{m},\mathsf{s},\mathsf{d}) \notin x^+$ (for polarity reasons), it follows that $\{(\mathsf{s},\mathsf{d})^{@\mathsf{m}}\}$ is fresh in ρ.

Negative. Straightforward by inspection. □

We do not detail the clear fact that composition is stable under isomorphism of Petri strategies. Altogether, this concludes the proof of:

Corollary D.10. *There is* PStrat, *a precategory with objects arenas, and morphisms negative Petri strategies up to isomorphism.*

$$\frac{t^0(\!|\alpha|\!) : \alpha \longmapsto_{\sigma} \beta}{\ell^{\otimes}(t^0)(\!|\ell^{\otimes}(\alpha)|\!) : \ell^{\otimes}(\alpha) \longmapsto_{\sigma\otimes\tau} \ell^{\otimes}(\beta)}$$

$$\frac{t^+(\!|\alpha|\!) : \alpha \overset{\ell_+ m}{\longmapsto}_{\sigma} \emptyset}{\ell^{\otimes}(t^+)(\!|\ell^{\otimes}(\alpha)|\!) : \ell^{\otimes}(\alpha) \overset{\ell_+ \ell_{\otimes} m}{\longmapsto}_{\sigma\otimes\tau} \emptyset}$$

$$\frac{t^+(\!|\alpha|\!) : \alpha \overset{\not{\ell} m}{\longmapsto}_{\sigma} \emptyset}{\ell^{\otimes}(t^+)(\!|\ell^{\otimes}(\alpha)|\!) : \ell^{\otimes}(\alpha) \overset{\not{\ell} \ell_{\otimes} m}{\longmapsto}_{\sigma\otimes\tau} \emptyset}$$

$$\frac{t^-(\!|(\mathrm{s},\mathrm{d})|\!) : \emptyset \overset{\ell_+ m}{\longmapsto}_{\sigma} \beta}{\ell^{\otimes}(t^-)(\!|(\mathrm{s},\mathrm{d})|\!) : \emptyset \overset{\ell_+ \ell_{\otimes} m}{\longmapsto}_{\sigma\otimes\tau} \ell^{\otimes}(\beta)}$$

$$\frac{t^-(\!|(\mathrm{s},\mathrm{d})|\!) : \emptyset \overset{\not{\ell} m}{\longmapsto}_{\sigma} \beta}{\ell^{\otimes}(t^-)(\!|(\mathrm{s},\mathrm{d})|\!) : \emptyset \overset{\not{\ell} \ell_{\otimes} m}{\longmapsto}_{\sigma\otimes\tau} \ell^{\otimes}(\beta)}$$

$$\frac{t^0(\!|\alpha|\!) : \alpha \longmapsto_{\tau} \beta}{\not{r}^{\otimes}(t^0)(\!|\not{r}^{\otimes}(\alpha)|\!) : \not{r}^{\otimes}(\alpha) \longmapsto_{\sigma\otimes\tau} \not{r}^{\otimes}(\beta)}$$

$$\frac{t^+(\!|\alpha|\!) : \alpha \overset{\ell_+ m}{\longmapsto}_{\tau} \emptyset}{\not{r}^{\otimes}(t^+)(\!|\not{r}^{\otimes}(\alpha)|\!) : \not{r}^{\otimes}(\alpha) \overset{\ell_+ \not{r}_{\otimes} m}{\longmapsto}_{\sigma\otimes\tau} \emptyset}$$

$$\frac{t^+(\!|\alpha|\!) : \alpha \overset{\not{\ell} m}{\longmapsto}_{\tau} \emptyset}{\not{r}^{\otimes}(t^+)(\!|\not{r}^{\otimes}(\alpha)|\!) : \not{r}^{\otimes}(\alpha) \overset{\not{\ell} \not{r}_{\otimes} m}{\longmapsto}_{\sigma\otimes\tau} \emptyset}$$

$$\frac{t^-(\!|(\mathrm{s},\mathrm{d})|\!) : \emptyset \overset{\ell_+ m}{\longmapsto}_{\tau} \beta}{\not{r}^{\otimes}(t^-)(\!|(\mathrm{s},\mathrm{d})|\!) : \emptyset \overset{\ell_+ \not{r}_{\otimes} m}{\longmapsto}_{\sigma\otimes\tau} \not{r}^{\otimes}(\beta)}$$

$$\frac{t^-(\!|(\mathrm{s},\mathrm{d})|\!) : \emptyset \overset{\not{\ell} m}{\longmapsto}_{\tau} \beta}{\not{r}^{\otimes}(t^-)(\!|(\mathrm{s},\mathrm{d})|\!) : \emptyset \overset{\not{\ell} \not{r}_{\otimes} m}{\longmapsto}_{\sigma\otimes\tau} \not{r}^{\otimes}(\beta)}$$

Fig. 46. Description of instantiated transitions of $\sigma \otimes \tau$

		π_{σ}	π_{τ}
$\ell^{\otimes}(t^0)(\!\|\ell^{\otimes}(\alpha)\|\!)$	$\mapsto$	$t^0(\!\|\alpha\|\!)$	
$\ell^{\otimes}(t^+)(\!\|\ell^{\otimes}(\alpha)\|\!)$	$\mapsto$	$t^+(\!\|\alpha\|\!)$	
$\ell^{\otimes}(t^-)(\!\|(\mathrm{s},\mathrm{d})\|\!)$	$\mapsto$	$t^-(\!\|(\mathrm{s},\mathrm{d})\|\!)$	
$\not{r}^{\otimes}(t^0)(\!\|\not{r}^{\otimes}(\alpha)\|\!)$	$\mapsto$		$t^0(\!\|\alpha\|\!)$
$\not{r}^{\otimes}(t^+)(\!\|\not{r}^{\otimes}(\alpha)\|\!)$	$\mapsto$		$t^+(\!\|\alpha\|\!)$
$\not{r}^{\otimes}(t^-)(\!\|(\mathrm{s},\mathrm{d})\|\!)$	$\mapsto$		$t^-(\!\|(\mathrm{s},\mathrm{d})\|\!)$

Fig. 47. Projections of instantiated transitions

D.2 PStrat as an IPA-Structure: Operations

We examine the operations involved in the IPA-structure, and show preservation of Petri strategies.

D.2.1 Tensor. The preservation of Petri strategies by the tensor operation is a simplification of composition, without the synchronized events.

Lemma D.11. *Consider σ and τ Petri structures.*

Then, instantiated transitions of $\sigma \otimes \tau$ are exactly as in Figure 46 – in the sense that there is a one-to-one correspondence between instantiated transitions in the premises and in the conclusion.

Using this description, we define in Figure 47 partial functions $\pi^{\otimes}_{\sigma} : \mathscr{E}_{\sigma\otimes\tau} \rightharpoonup \mathscr{E}_{\sigma}$ and $\pi^{\otimes}_{\tau} : \mathscr{E}_{\sigma\otimes\tau} \rightharpoonup \mathscr{E}_{\tau}$ extracting various data from instantiated transitions, following the characterization of instantiated transitions of $\sigma \otimes \tau$ given in Figure 46.

Finally, those projection functions are extended to instantiated transitions in context via:

$$\begin{array}{rcll} \pi^{\otimes}_{\sigma} & : & \mathscr{F}_{\sigma\otimes\tau} & \rightharpoonup \mathscr{F}_{\sigma} \\ & & \mathbf{e} \uplus (\gamma +^{\otimes} \gamma') & \mapsto \pi^{\otimes}_{\sigma}(\mathbf{e}) \uplus \gamma \\ \pi^{\otimes}_{\tau} & : & \mathscr{F}_{\sigma\otimes\tau} & \rightharpoonup \mathscr{F}_{\tau} \\ & & \mathbf{e} \uplus (\gamma +^{\otimes} \gamma') & \mapsto \pi^{\otimes}_{\tau}(\mathbf{e}) \uplus \gamma' . \end{array}$$

Using these, from a run $\rho : \emptyset \twoheadrightarrow_{\sigma\otimes\tau} \alpha$ we extract:

$$\rho_{\sigma} = \rho \upharpoonright \pi^{\otimes}_{\sigma}, \qquad \rho_{\tau} = \rho \upharpoonright \pi^{\otimes}_{\tau},$$

we will also use for restriction the partial functions

$$\begin{array}{rcccl} u & : & \mathsf{Moves} & \rightharpoonup & \mathsf{Moves} \\ & & \ell_{+}\ell_{\otimes}m & \mapsto & \ell_{+}m \\ & & r_{+}\ell_{\otimes}m & \mapsto & r_{+}m \\ d & : & \mathsf{Moves} & \rightharpoonup & \mathsf{Moves} \\ & & \ell_{+}r_{\otimes}m & \mapsto & \ell_{+}m \\ & & r_{+}r_{\otimes}m & \mapsto & r_{+}m \end{array}$$

which allow us to prove the following property:

Lemma D.12. *Consider $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\sigma\otimes\tau} \alpha$. Then, $\alpha = \alpha_\sigma +^{\otimes} \alpha_\tau$ and*

$$\rho_\sigma : \emptyset \longrightarrow\!\!\!\!\rightarrow \alpha_\sigma\,, \qquad\qquad \rho_\tau : \emptyset \longrightarrow\!\!\!\!\rightarrow \alpha_\tau\,.$$

where $\mathsf{play}(\rho_\sigma) = \mathsf{play}(\rho) \upharpoonright u$ *and* $\mathsf{play}(\rho_\tau) = \mathsf{play}(\rho) \upharpoonright d$.
Moreover, $\mathsf{Coll}(\rho) = \mathsf{Coll}(\rho_\sigma) +^{\otimes} \mathsf{Coll}(\rho_\tau)$.

Proof. Exactly as for Lemma D.4, without synchronized transitions. □

Next, as for composition, we observe that these projections preserve valid plays. For that we shall first need the following easy lemma:

Lemma D.13. *Consider A, B arenas, and $s \in |A \otimes B|^*$.*
Then, $s \in \mathsf{Plays}(A \otimes B)$ iff $s \upharpoonright \ell_{\otimes}^* \in \mathsf{Plays}(A)$ and $s \upharpoonright r_{\otimes}^* \in \mathsf{Plays}(B)$.

Proof. Straightforward. □

Using this and Lemma D.12, we show that projections preserve valid runs. Consider $\sigma : A_1 \vdash B_1$ and $\tau : A_2 \vdash B_2$ Petri strategies.

Lemma D.14. *Consider $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\sigma\otimes\tau} \alpha$ such that* $\mathsf{play}(\rho) \in \mathsf{Plays}(A_1 \otimes A_2 \vdash B_1 \otimes B_2)$.
Then, $\mathsf{play}(\rho_\sigma) \in \mathsf{Plays}(A_1 \vdash B_1)$ *and* $\mathsf{play}(\rho_\tau) \in \mathsf{Plays}(A_2 \vdash B_2)$.

Proof. As for the proof of Lemma D.6 (without synchronization), using condition *valid* of Petri strategies along with Lemma D.13. □

Using Lemmas D.12, D.13, and D.14, we prove as for Proposition D.7:

Proposition D.15. *If $\sigma : A_1 \vdash B_1$ and $\tau : A_2 \vdash B_2$ are Petri strategies, so is $\sigma \otimes \tau : A_1 \otimes B_1 \vdash A_2 \otimes B_2$.*
Moreover, if σ and τ are negative, so is $\sigma \otimes \tau$.

D.2.2 Renamings. Before we go on to currying and promotion, we introduce a technical tool useful in ensuring that they preserve Petri strategies.

First, for any game A we write $\mathsf{Plays}_(A)$ for the set of **negative plays** on A, *i.e.* those $s_1 \dots s_n \in \mathsf{Plays}(A)$ such that $\mathsf{pol}(s_1) = -$. If $f : \mathsf{Moves} \rightharpoonup \mathsf{Moves}$ and $s = s_1 \dots s_n \in \mathsf{Plays}(A)$ such that f is defined on $|A|$, then we write $f(s) = f(s_1) \dots f(s_n)$. In the sequel, we should be particularly interested in such functions on moves that can be decomposed in f and $(g_{\mathsf{m}})_{\mathsf{m}\in\mathsf{dom}(f)}$ where

$$f : \mathcal{M} \rightharpoonup \mathcal{M}\,, \qquad\qquad g_{\mathsf{m}} : \mathsf{Tok} \to \mathsf{Tok}\,,$$

in which case we obtain a partial function between moves set as

$$\begin{array}{rcccl} [f, (g_{\mathsf{m}})] & : & \mathsf{Moves} & \rightharpoonup & \mathsf{Moves} \\ & & (\mathsf{m}, \mathsf{s}, \mathsf{d}) & \mapsto & (f(\mathsf{m}), \mathsf{s}', \mathsf{d}') \\ & & & & \quad \text{where } (\mathsf{s}', \mathsf{d}') = g_{\mathsf{m}}(\mathsf{s}, \mathsf{d}). \end{array}$$

Definition D.16. Consider A, B games, $f, (g_{\mathsf{m}})$ s.t. $[f, (g_{\mathsf{m}})] : \mathsf{Moves} \rightharpoonup \mathsf{Moves}$ partial injection.
We say $h = [f, (g_{\mathsf{m}})]$ is a **global renaming** from A to B, written $[f, (g_{\mathsf{m}})] : A \rightsquigarrow B$, if:

defined: for all $a \in |A|$, $h(a)$ defined.
polarity-preserving: $\forall a \in |A|,\ \mathsf{pol}(ha) = \mathsf{pol}(a)$
validity: $\forall s \in \mathsf{Plays_}(A), h(s) \in \mathsf{Plays_}(B)$
receptivity: for all $s \in \mathsf{Plays_}(A)$, for all $h(s)b^- \in \mathsf{Plays_}(B)$, there exists $sa^- \in \mathsf{Plays_}(A)$ such that $h(a) = b$.
courtesy: for all $a \rightarrowtriangle_A b$, either $h(a) \rightarrowtriangle_B h(b)$ or $(\mathsf{pol}(a), \mathsf{pol}(b)) = (-, +)$.

Global renamings are used to transport Petri strategies across games. The following definition, first applied simply on Petri structures, extends Definition B.1 in that it also renames tokens rather than merely rerouting visible transitions.

Definition D.17. Consider A, B games, $\boldsymbol{\sigma}$ a Petri structure on $\mathsf{mult}(A)$, and $h = [f, (g_{\mathsf{m}})] : A \rightsquigarrow B$. We define the **renaming** $\boldsymbol{\sigma}[h]$ on $\mathsf{mult}(B)$, with the same components as $\boldsymbol{\sigma}$, except:

$$\begin{array}{rcll} \partial_{\boldsymbol{\sigma}[h]}(t) & = & f(\partial_{\boldsymbol{\sigma}}(t)) & \\ \delta_{\boldsymbol{\sigma}[h]}\langle t^+\rangle(\alpha) & = & g_{\mathsf{m}}(\delta_{\boldsymbol{\sigma}}\langle t^+\rangle(\alpha)) & \text{for } \mathsf{m} = \partial_{\boldsymbol{\sigma}}(t^+) \\ \delta_{\boldsymbol{\sigma}[h]}\langle t^-\rangle(g_{\mathsf{m}}(\mathsf{s},\mathsf{d})) & = & \delta_{\boldsymbol{\sigma}}\langle t^-\rangle(\mathsf{s},\mathsf{d}) & \text{for } \mathsf{m} = \partial_{\boldsymbol{\sigma}}(t^-). \end{array}$$

observing that by hypothesis, g_{m} is injective for all $\mathsf{m} \in \mathsf{dom}(f)$.

In order to use global renaming to transport Petri strategies, we must transport valid runs. Consider A, B games, $h = [f, (g_{\mathsf{m}})] : A \rightsquigarrow B$, and $\boldsymbol{\sigma} : A$ a Petri strategy. Then we set

$$\begin{array}{rcrcll} -[h] & : & \mathscr{E}_{\boldsymbol{\sigma}} & \to & \mathscr{E}_{\boldsymbol{\sigma}[h]} & \\ & & t^0\langle\alpha\rangle & \mapsto & t^0\langle\alpha\rangle & \\ & & t^-\langle(\mathsf{s},\mathsf{d})\rangle & \mapsto & t^-\langle g_{\mathsf{m}}(\mathsf{s},\mathsf{d})\rangle & \text{where } \mathsf{m} = \partial_{\boldsymbol{\sigma}}(t^-) \\ & & t^+\langle\alpha\rangle & \mapsto & t^+\langle\alpha\rangle & \end{array}$$

extended to instantiated transitions in context with $(\mathbf{e} \uplus \gamma)[h] = \mathbf{e}[h] \uplus \gamma$. It is immediate from the definition that this substitution leaves pre- and post-conditions of instantiated transitions unchanged, so that it lifts to *runs*: for any $\rho : \emptyset \longrightarrow\!\!\!\!\!\rightarrow_{\boldsymbol{\sigma}} \alpha$, $\rho[h] : \emptyset \longrightarrow\!\!\!\!\!\rightarrow_{\boldsymbol{\sigma}[h]} \alpha$ is defined pointwise.

We shall now prove that this preserves valid runs. First, an easy observation:

Lemma D.18. *Consider A a game, and $\boldsymbol{\sigma} : A$ a negative Petri strategy. Then, for all $\rho : \emptyset \longrightarrow\!\!\!\!\!\rightarrow_{\boldsymbol{\sigma}} \alpha$ s.t. $\mathsf{play}(\rho) \in \mathsf{Plays}(A)$, we have $\mathsf{play}(\rho) \in \mathsf{Plays_}(A)$.*

Proof. By *negative*, the first transition of ρ cannot be positive or neutral (as those require at least one tokil). Thus, it is negative. □

Lemma D.19. *Consider A a game, $\boldsymbol{\sigma} : A$ negative, $h = [f, (g_{\mathsf{m}})] : A \rightsquigarrow B$, and $\rho : \emptyset \longrightarrow\!\!\!\!\!\rightarrow_{\boldsymbol{\sigma}} \alpha$. If $\mathsf{play}(\rho) \in \mathsf{Plays_}(A)$, then $\mathsf{play}(\rho[h]) = h(\mathsf{play}(\rho)) \in \mathsf{Plays_}(B)$.*

Proof. Straightforward by induction on ρ. □

We shall also use a sort of reciprocal statement:

Lemma D.20. *Consider A a game, $\boldsymbol{\sigma} : A$ negative, $h = [f, (g_{\mathsf{m}})] : A \rightsquigarrow B$, and $\rho' : \emptyset \longrightarrow\!\!\!\!\!\rightarrow_{\boldsymbol{\sigma}[h]} \alpha$. If $\mathsf{play}(\rho') \in \mathsf{Plays}(B)$, there is a unique $\rho : \emptyset \longrightarrow\!\!\!\!\!\rightarrow_{\boldsymbol{\sigma}} \alpha$ s.t. $\mathsf{play}(\rho) \in \mathsf{Plays}(A)$ and $\rho' = \rho[h]$.*

Proof. Straightforward by induction on ρ'. □

Proposition D.21. *Consider A a game, $\boldsymbol{\sigma} : A$ negative, $h = [f, (g_{\mathsf{m}})] : A \rightsquigarrow B$. Then, $\boldsymbol{\sigma}[h] : B$ is a negative Petri strategy.*

Proof. *Valid.* Consider $\rho' : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\sigma[h]} \alpha$ such that $\mathsf{play}(\rho') \in \mathsf{Plays}(B)$. Consider $\mathbf{f}^+ : \alpha \xrightarrow{b}_{\sigma[h]} \beta$, write $\mathbf{f} = t^+(\!|\mu|\!) \uplus \gamma$. By Lemma D.20, there is a unique $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\sigma} \alpha$ such that $\mathsf{play}(\rho) \in \mathsf{Plays}(A)$ and $\rho' = \rho[h]$. By definition of transitions of $\boldsymbol{\sigma}[h]$, we have $b = h(a)$ with

$$t^+(\!|\mu|\!) \uplus \gamma : \alpha \xrightarrow{a} \beta$$

and $\mathsf{play}(\rho)a \in \mathsf{Plays}(A)$ as $\boldsymbol{\sigma}$ is *valid*. Note actually $\mathsf{play}(\rho)a \in \mathsf{Plays}_(A)$ by Lemma D.18. Hence, $h(\mathsf{play}(\rho)a) = \mathsf{play}(\rho[h])b \in \mathsf{Plays}_(B)$ by condition *validity* of global renamings, as required.

Receptive. Consider $\rho' : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\sigma[h]} \alpha$ such that $s' = \mathsf{play}(\rho') \in \mathsf{Plays}(B)$. Consider $sb^- \in \mathsf{Plays}(B)$. By Lemma D.20, there is a unique $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\sigma} \alpha$ such that $s = \mathsf{play}(\rho) \in \mathsf{Plays}(A)$ and $h(s) = s'$. By condition *receptivity* of global renamings, there is $sa^- \in \mathsf{Plays}(A)$ such that $h(a) = b$. As $\boldsymbol{\sigma}$ is *receptive*, there is a unique $\mathbf{e}^- \in \mathscr{E}_{\sigma}$ such that $\mathbf{e}^- : \emptyset \xmapsto{a^-}_{\sigma} \beta$ for some β. By definition of $\boldsymbol{\sigma}[h]$, $\mathbf{e}^- : \emptyset \xmapsto{b^-}_{\sigma[h]} \beta$ as required. Uniqueness follows immediately from uniqueness for $\boldsymbol{\sigma}$.

Strongly safe. Consider $\rho' : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\sigma[h]} \alpha$ such that $s' = \mathsf{play}(\rho') \in \mathsf{Plays}(B)$. By Lemma D.20, there is a unique $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\sigma} \alpha$ such that $s = \mathsf{play}(\rho) \in \mathsf{Plays}(A)$ and $h(s) = s'$. If $\mathbf{f}' : \alpha \longrightarrow_{\sigma[h]} \beta$ then also $\mathbf{f}' : \alpha \longrightarrow_{\sigma} \beta$, and $\mathsf{new}(\mathbf{f}')$ is fresh in ρ, so fresh in ρ'. If $\mathbf{f}' : \alpha \xrightarrow{b}_{\sigma[h]} \beta$ with $s'b \in \mathsf{Plays}(B)$, then again by Lemma D.20, $\mathbf{f}' = \mathbf{f}[h]$ and $b = h(a)$ for $\mathbf{f} : \alpha \xrightarrow{a}_{\sigma} \beta$ with $sa \in \mathsf{Plays}(A)$. As $\boldsymbol{\sigma}$ is *strongly safe*, it follows that $\mathsf{new}(\mathbf{f})$ is fresh in ρ, but $\mathsf{new}(\mathbf{f}) = \mathsf{new}(\mathbf{f}')$ so $\mathsf{new}(\mathbf{f}')$ is fresh in ρ' as required.

Negative. Straightforward from the fact that $\boldsymbol{\sigma}$ is *negative*. □

D.2.3 Currying. This is a simple application of global renaming.

Lemma D.22. *Consider $\Gamma, x : A, \Delta$ a list of variable/arena declarations, and O well-opened. Then,*

$$(\Lambda_x, (\mathsf{id})) : (!(\&[\Gamma, x : A, \Delta]) \vdash O) \rightsquigarrow (!(\&[\Gamma, \Delta]) \vdash !A \multimap O)$$

where Λ_x is defined in Definition B.2.

Proof. Immediate verification. □

Corollary D.23. *Consider $\boldsymbol{\sigma} : !(\&[\Gamma, x : A, \Delta]) \vdash O$ a negative Petri strategy.*
Then, $\Lambda^{\Gamma,\Delta}_{x:A,O}(\boldsymbol{\sigma}) : !(\&[\Gamma, \Delta]) \vdash !A \multimap O$ is a negative Petri strategy.

D.2.4 Functorial promotion. Rather than directly dealing with Definition 2.11, we decompose it: first, a functorial promotion, and secondly, a renaming corresponding to *digging*.

We first define functorial promotion on Petri structures:

Definition D.24. Consider $\boldsymbol{\sigma} \in \mathsf{PStruct}(M, N)$. We set $\mathcal{L}_{!\sigma} = \mathcal{L}_{\sigma}$, $\mathcal{T}_{!\sigma} = \mathcal{T}_{\sigma}$ with the same polarities, $\partial_{!\sigma} = \partial_{\sigma}$, and pre- and post-conditions are also unchanged. Finally, the *transition table* is:

$$\begin{array}{rcll}
\delta_{!\sigma}\langle t^0\rangle(\mathsf{e} :: \alpha) & = & \mathsf{e} :: \beta & \text{if } \delta_{\sigma}\langle t\rangle(\alpha) = \beta \\
\delta_{!\sigma}\langle t^+\rangle(\mathsf{e} :: \alpha) & = & (\mathsf{e} :: \mathsf{s}, \mathsf{d}) & \text{if } \delta_{\sigma}\langle t\rangle(\alpha) = (\mathsf{s}, \mathsf{d}) \\
\delta_{!\sigma}\langle t^-\rangle(\mathsf{e} :: \mathsf{s}, \mathsf{d}) & = & \mathsf{e} :: \alpha & \text{if } \delta_{\sigma}\langle t\rangle(\mathsf{s}, \mathsf{d}) = \alpha
\end{array}$$

where $\mathsf{e} :: \alpha$ is $\{(\mathsf{e} :: \mathsf{s}_i, \mathsf{d}_i)^{@l_i} \mid (\mathsf{s}_i, \mathsf{d}_i)^{@l_i} \in \alpha\}$.

With this definition, we obtain $!\boldsymbol{\sigma} \in \mathsf{PStruct}(!M, !N)$.

We prove that this operation preserves Petri strategies – the proof follows closely that of tensor, of which the ! can be regarded as an infinitary version.

Lemma D.25. *Consider $\boldsymbol{\sigma}$ a Petri structure.*
Then, instantiated transitions of $!\boldsymbol{\sigma}$ are exactly as in Figure 48 – in the sense that there is a one-to-one correspondence between instantiated transitions in the premises and in the conclusion.

$$\frac{t^0(\!|\alpha|\!) : \alpha \longmapsto_{\boldsymbol{\sigma}} \beta}{t^0(\!|\mathsf{e} :: \alpha|\!) : \mathsf{e} :: \alpha \longmapsto_{!\boldsymbol{\sigma}} \mathsf{e} :: \beta} \qquad \frac{t^-(\!|(\mathsf{s},\mathsf{d})|\!) : \emptyset \overset{m}{\longmapsto}_{\boldsymbol{\sigma}} \beta}{t^-(\!|(\mathsf{e} :: \mathsf{s},\mathsf{d})|\!) : \emptyset \overset{\mathsf{e}::m}{\longmapsto}_{!\boldsymbol{\sigma}} \mathsf{e} :: \beta} \qquad \frac{t^+(\!|\alpha|\!) : \alpha \overset{m}{\longmapsto}_{\boldsymbol{\sigma}} \emptyset}{t^+(\!|\mathsf{e} :: \alpha|\!) : \mathsf{e} :: \alpha \overset{\mathsf{e}::m}{\longmapsto}_{\boldsymbol{\sigma}} \emptyset}$$

Fig. 48. Description of instantiated transitions of $!\boldsymbol{\sigma}$

Using this description, we define for each $\mathsf{e} \in \mathcal{E}$ a partial function

$$\begin{array}{rccc} \pi^!_{\mathsf{e}} : & \mathscr{C}_{!\boldsymbol{\sigma}} & \rightharpoonup & \mathscr{C}_{\boldsymbol{\sigma}} \\ & t^0(\!|\mathsf{e} :: \alpha|\!) & \mapsto & t^0(\!|\alpha|\!) \\ & t^-(\!|(\mathsf{e} :: \mathsf{s},\mathsf{d})|\!) & \mapsto & t^-(\!|(\mathsf{s},\mathsf{d})|\!) \\ & t^+(\!|\mathsf{e} :: \alpha|\!) & \mapsto & t^+(\!|\alpha|\!) \end{array}$$

and undefined otherwise. In order to extend those to instantiated transitions in context, first define

$$\mathop{|||}_{e \in \mathcal{E}} \alpha_e = \mathop{\biguplus}_{e \in \mathcal{E}} e :: \alpha_e$$

for $(\alpha_{\mathsf{e}})_{\mathsf{e} \in \mathcal{E}}$ a family of conditions empty almost everywhere. We may then set:

$$\begin{array}{rccc} \pi^!_{\mathsf{e}} : & \mathscr{F}_{!\boldsymbol{\sigma}} & \rightharpoonup & \mathscr{F}_{\boldsymbol{\sigma}} \\ & \mathbf{e} \uplus (|||_{\mathsf{e} \in \mathcal{E}} \gamma_{\mathsf{e}}) & \mapsto & \pi^!_{\mathsf{e}}(\mathbf{e}) \uplus \gamma_{\mathsf{e}} \,. \end{array}$$

Using these, from a run $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{!\boldsymbol{\sigma}} \alpha$ we extract, for all $\mathsf{e} \in \mathcal{E}$:

$$\rho_{\mathsf{e}} = \rho \upharpoonright \pi^!_{\mathsf{e}} \,,$$

we will also use for restrictions the partial functions

$$\begin{array}{rccc} \mathsf{e} : & \mathsf{Moves} & \rightharpoonup & \mathsf{Moves} \\ & (\mathsf{m}, \mathsf{e} :: \mathsf{s}, \mathsf{d}) & \mapsto & (\mathsf{m}, \mathsf{s}, \mathsf{d}) \end{array}$$

and undefined otherwise – the abuse of notations should not create confusion.

Now, as for the tensor we can prove:

Lemma D.26. *Consider $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{!\boldsymbol{\sigma}} \alpha$. Then, $\alpha = |||_{\mathsf{e} \in \mathcal{E}} \alpha_{\mathsf{e}}$ and*

$$\rho_{\mathsf{e}} : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\sigma}} \alpha_{\mathsf{e}}$$

for all $\mathsf{e} \in \mathcal{E}$, where $\mathsf{play}(\rho_{\mathsf{e}}) = \mathsf{play}(\rho) \upharpoonright \mathsf{e}$.

Moreover, $\mathsf{Coll}(\rho) = |||_{\mathsf{e} \in \mathcal{E}} \mathsf{Coll}(\rho_{\mathsf{e}})$.

Proof. The proof is the same as for Lemma D.4, without synchronized transitions. □

The construction goes on as for the tensor, with preservation of plays via projections:

Lemma D.27. *Consider A an arena and $s \in |!A|^*$.*
Then, $s \in \mathsf{Plays}(!A)$ iff $s \upharpoonright \mathsf{e} \in \mathsf{Plays}(A)$ for all $\mathsf{e} \in \mathcal{E}$.

Proof. Straightforward. □

Consider now $\boldsymbol{\sigma} : A \vdash B$ a Petri strategy.
Using Lemmas D.27 and D.26, we show that projections preserve valid runs.

Lemma D.28. *Consider $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{!\boldsymbol{\sigma}} \alpha$ such that $\mathsf{play}(\rho) \in \mathsf{Plays}(!A \vdash !B)$.*
Then, for all $\mathsf{e} \in \mathcal{E}$, $\mathsf{play}(\rho_{\mathsf{e}}) \in \mathsf{Plays}(A \vdash B)$.

Proof. As for the proof of Lemma D.6 (without synchronization), using condition *valid* of Petri strategies along with Lemma D.27. □

Using Lemmas D.26, D.27 and D.28, we prove as for Proposition D.7:

Proposition D.29. *If $\boldsymbol{\sigma} : A \vdash B$ is a Petri strategy, then so is $!\boldsymbol{\sigma} : !A \vdash !B$. Moreover, if $\boldsymbol{\sigma}$ is negative then so is $!\boldsymbol{\sigma}$.*

D.2.5 Local renamings. To match Definition 2.11, we must also rename following *digging*. Recall that digging is the following map:

$$\begin{array}{rcrcl} \mathsf{dig} & : & \mathsf{Moves} & \rightharpoonup & \mathsf{Moves} \\ & & (\mathsf{m}, \mathsf{e} :: \mathsf{e}' :: \mathsf{s}, \mathsf{d}) & \mapsto & (\mathsf{m}, \langle \mathsf{e}, \mathsf{e}' \rangle :: l, \mathsf{d}) \end{array}$$

and undefined otherwise. To rename a strategy following this, it is convenient to introduce:

Definition D.30. Consider A, B arenas.
A **(local) renaming** from A to B is a partial injection $f : \mathsf{Moves} \rightharpoonup \mathsf{Moves}$ defined on $|A|$, s.t.:

validity: for all $x \in \mathscr{C}(A)$, $fx \in \mathscr{C}(B)$,
polarity-preserving: for all $a \in |A|$, $\mathsf{pol}(f(a)) = \mathsf{pol}(a)$,
receptivity: for all $x \in \mathscr{C}(A)$, if $f(x) \vdash_B b^-$, then there is $x \vdash_A a$ such that $f(a) = b$,
courtesy: for all $a \rightarrowtriangle_A a'$, either $f(a) \rightarrowtriangle_B f(a')$ or $(\mathsf{pol}(a), \mathsf{pol}(a')) = (-, +)$.

We write $f : A \curvearrowright B$ to mean that f is a renaming from A to B.

It is clear in particular that $\mathsf{dig} : !!A^\perp \to !A$ is a renaming, for any arena A. This is a variant of Definition D.16, closer to the usual lifting operation used for this purpose in concurrent games.

Clearly, a local renaming is a global renaming. But local renamings are sometimes more convenient, because if $f : A^\perp \curvearrowright B^\perp$ and $g : A' \curvearrowright B'$ are local renaming, then it is obvious that $f \vdash g : A \vdash B \curvearrowright A' \vdash B'$ (defined in the obvious way) is still a local renaming – this is not always the case for global renamings for non-negative games.

Definition D.31. Consider $\boldsymbol{\sigma} : A \vdash B$ a negative Petri strategy and $f : A^\perp \curvearrowright A'^\perp, g : B \curvearrowright B'$. Then, we define $g \cdot \boldsymbol{\sigma} \cdot f = \boldsymbol{\sigma}[f \vdash g] : A' \vdash B'$.

This yields a negative Petri strategy by Proposition D.21.

D.2.6 Digging. We may finally perform digging and deduce the correctness of promotion:

Proposition D.32. *Consider $\boldsymbol{\sigma} : !A \vdash B$ a negative Petri strategy. Then, $\boldsymbol{\sigma}^\dagger : !A \vdash !B$ is a negative Petri strategy.*

Proof. It is a direct verification that $\boldsymbol{\sigma}^\dagger = (!\boldsymbol{\sigma})[\mathsf{dig} \vdash \mathsf{id}]$, which is a negative Petri strategy by Propositions D.29 and Definition D.31. □

D.3 PStrat as an IPA-Structure: Primitives

We now show that the Petri structures representing the primitives of IPA are indeed Petri strategies.

Given a Petri structure $\boldsymbol{\sigma}$ and a play s of a game A, we say that s is reachable by $\boldsymbol{\sigma}$ when there exists a run ρ of $\boldsymbol{\sigma}$ with $\mathsf{play}(\rho) = s$. Given a game A, we define the Scott order on $\mathscr{C}(A)$ as follows: $x \sqsubseteq_A y := (x \supseteq^- \subseteq^+ y)$ which was already encountered in Lemma D.8 for copycat.

D.3.1 Variable and Evaluation. For variable, we notice that $\mathbf{var}_{x:M} = \mathfrak{a}_M[\iota_\&^x \vdash \mathsf{id}]$ and we conclude easily by D.21 since $\iota_\&^x : M^\perp \curvearrowright [\Gamma, x : M, \Delta]^\perp$ is a local renaming. For the evaluation, the map Ω defined in Section B.2 is a global renaming $((M \multimap N) \vdash (M \multimap N)) \curvearrowright ((M \multimap N) \otimes M \vdash N)$.

D.3.2 Contraction. We now show that the Petri structure $\mathbf{c}_A$ is a Petri strategy on $!A \vdash !A \otimes !A$.

Given a move a, we write $\ell_!(a)$ for $(\mathsf{m}, \ell_!(\mathsf{e}) :: \mathsf{s}, \mathsf{d})$ for $a = (\mathsf{m}, \mathsf{e} :: \mathsf{s}, \mathsf{d})$, and similarly for $r_!(a)$. It is not defined on moves with an empty stack.

Lemma D.33. *Consider $s \in \mathsf{play}(!A \vdash !A \otimes !A)$ reachable by $\mathbf{c}_A$. Then, $|s| = (\ell_!(x_1) \uplus r_!(x_2) \vdash y_1 \otimes y_2)$ and $y_i \sqsubseteq x_i$.*

Proof. We prove the implication by induction on the length of s. It holds for all plays of length zero. We assume the implication holds for all plays of length n.

Consider $s' = s \cdot a$ reachable by $\boldsymbol{\sigma}$ and s has length n. We apply the induction hypothesis to s (which is reachable by $\boldsymbol{\sigma}$) and obtain that, writing $|s| = (\ell_!(x_1) \uplus \ell_!(x_2) \vdash y_1 \otimes y_2)$, we have $y_i \sqsubseteq x_i$.

- If a is negative, and on $\ell_\vdash$, then the inequality for s' holds by definition of $\sqsubseteq$.
- If a is negative, and on $r_\vdash$, then because s' is a play, the parent of a must exist and belong to s. By induction, that parent must have an exponential stack of the form $\ell_!\mathsf{e} :: \mathsf{s}$ or $r_!\mathsf{e} :: \mathsf{s}$ and we can conclude.
- If a is positive, and on the left, ie. $m = \ell_\vdash m_0$: in the run producing a, there must be a token in m_0^- that triggered t. That token must be (s, d) since t has a trivial transition function. That token can only be produced by one of the negative transitions $r_\vdash \ell_\otimes m_0$ or $r_\vdash r_\otimes m_0$ – assume the former. This directly shows that $\mathsf{s} = \ell_!\mathsf{e} :: \mathsf{s}'$ for some s', and that $(r_\vdash \ell_\otimes m_0, \mathsf{e} :: \mathsf{s}, \mathsf{d}) \in |s|$, which implies that $(m_0, \mathsf{e} :: \mathsf{s}, \mathsf{d}) \in y$. As a result $|s| = (x \cup \{(m_0, \ell_!\mathsf{e} :: \mathsf{s}', \mathsf{d})\}) \vdash y \otimes z$ satisfies the desired property.
- If a is positive and on the right for instance $m = r_\vdash \ell_\otimes m_0$. Then a similar line of reasoning shows that $\mathsf{s} = \mathsf{e} :: \mathsf{s}'$ and we must have $(\ell_\vdash m_0, \ell_!\mathsf{e} :: \mathsf{s}', \mathsf{d}) \in |s|$ which entails the desired property. □

Lemma D.34. *$\mathbf{c}_A$ is a negative Petri strategy on $!A \vdash !A \otimes !A$.*

Proof. *Negative.* Simple inspection of the net.

Strong safety. Consider $\rho : \emptyset \xrightarrow{s}\!\!\!\!\rightarrow \alpha$ with s a play, and $\mathbf{f} : \alpha \xrightarrow{a} \beta$ with sa also a play (note that there is no neutral transition). Note that positive transitions do not create tokens, so there is nothing to check. For negative transitions, it follows from the injectivity of transition functions and the fact that plays are non-repetitive.

Validity. Consider $\rho : \emptyset \xrightarrow{s}\!\!\!\!\rightarrow \alpha$ a run of $\mathbf{c}_A$ and s a play of the game. Assume that ρ can extend by $\mathbf{f}^+ : \alpha \xrightarrow{a} \beta$. By Lemma D.33, we know that $|s| = (\ell_!(x_1) \uplus r_!(x_2) \vdash y_1 \otimes y_2)$ with $y_i \sqsubseteq x_i$. There are three transitions, hence three cases. We detail the case for $a = \ell_\vdash a_0$: then by inspecting the net we have that a_0 must be of the form $\ell_!(a_1)$ with $a_1 \in y$ or $r_!(a_1)$ with $a_1 \in z$ – assume the former. From $a_1 \in z$, we deduce that the justifier of a is already present in s; and moreover a cannot conflict with anything in s. That sa is non-repetitive follows from strong safety and that transition functions are injective.

Receptivity. Consider sa^- a play of $!A \vdash !A \otimes !A$ and $\rho : \emptyset \xrightarrow{s}\!\!\!\!\rightarrow \alpha$. If a is on the right of $\vdash$, then we can use the corresponding transition whose function domain is total on stacks of $!A$. For a on the left, a cannot be minimal so its justifier a_0 must occur in s. By Lemma D.33, its exponential stack must start with $\ell_!(\mathsf{e})$ or $r_!(\mathsf{e})$, and thus so must that of a. As a result, a will be accepted by the transition corresponding to its address. □

D.3.3 Fixpoint. We start by characterising the plays of Y_O, where O is a well-opened arena. We reuse the same encodings as in Section C.3.3.

Lemma D.35. *Let $\rho : \emptyset \xrightarrow{s}\!\!\!\!\rightarrow \alpha$ be a run of Y_O such that s is a play.*

Then there exists suffix-closed $J \subseteq \mathcal{E}^+$, configurations $z, y_\epsilon \in \mathscr{C}(O)$, $(y_{\mathsf{s}})_{\mathsf{s}\in J}$ and $(z_{\mathsf{s}} \in \mathscr{C}^{\neq\emptyset}(O))_{\mathsf{s}\in J}$ with J empty if y_ϵ is, $z \sqsubseteq y_\epsilon$ and $z_{\mathsf{s}} \sqsubseteq y_{\mathsf{s}}$ for all s and

$$|s| = (\emptyset \multimap () :: y_\epsilon) \uplus (\mathsf{e} :: (\mathsf{s}) :: z_{e\cdot\mathsf{s}} \multimap (\mathsf{e}\cdot\mathsf{s}) :: y_{\mathsf{s}}) \vdash z,$$

plus if $y_\epsilon = \emptyset$ then $J = \emptyset$, and if $e \cdot \mathsf{s} \in J$, then $y_{\mathsf{s}} \neq \emptyset$.

Proof. A direct induction over the run, using the transition table. □

Lemma D.36. *Y_O is a negative Petri strategy $!(!O \multimap O) \vdash O$.*

Proof. *Negativity* and *receptivity* are easily verified.

Validity. Consider $\rho : \emptyset \xrightarrow{s}\!\!\!\!\rightarrow \alpha$ be a run of Y_O such that s is a play of the game. Consider now an extension of ρ by the positive transition $\mathbf{f} : \alpha \xrightarrow{a} \beta$. We show that sa is a valid play. First, if a or a conflicting move occurs already in s, given the shape of the net, this means that Opponent played twice the same move or two conflicting moves earlier in s which is absurd. It remains to show that the predecessor of a occurs in s, which is a consequence of Lemma D.35

Strong-safety. All negative transitions have injective transition functions, and the two negative transitions $\nearrow_{\mathsf{r}}\mathsf{m}^-$ and $\ell_{\mathsf{r}}\ell_{\multimap}\mathsf{m}^-$ which have a common postcondition (m^-), have disjoint codomains, hence Y_O is strongly safe. □

D.3.4 Queries, Conditional, Constants. For these IPA structures defined on linear games, a simple inspection shows that they define they are IPA strategies.

D.3.5 Let bindings. We now move on to showing that **let** is a Petri strategy on $(!\mathbf{X} \multimap \mathbf{Y}) \otimes \mathbf{X} \vdash \mathbf{Y}$.

Lemma D.37. *Let $\rho : \emptyset \xrightarrow{s}\!\!\!\!\rightarrow \alpha$ be a valid run for* **let**. *Then $|s| = ((\uplus_{\mathsf{e}\in I}\mathsf{e} :: x_{\mathsf{e}}) \multimap y) \otimes z \vdash w$ with: (1) if $y \neq \emptyset$ then z is maximal in $\mathscr{C}(\mathbf{X})$; (2) $w \sqsubseteq y$ and $x_{\mathsf{e}} \subseteq z$ for all $\mathsf{e} \in I$.*

Proof. By induction on ρ. □

Lemma D.38. **let** *is a Petri strategy on* $(!\mathbf{X} \multimap \mathbf{Y}) \otimes \mathbf{X} \vdash \mathbf{Y}$.

Proof. As usual, receptivity and negativity are clear. Strong safety is clear on the forwarding transitions. For the transition s, we note that the token in location 3 is never in $\mathsf{eat}(\mathsf{s})$, and the other token at location 5 has a stack given by Opponent, so there cannot be any risk of confusion.

Validity follows from Lemma D.37. □

D.3.6 Newref and newsem. We now show that **newref** and **newsem** are valid Petri strategies. We focus our attention on **newref**, the proof for **newsem** being similar.

We start by recovering, out of a run of **newref**, a memory trace. A memory trace is a word on the alphabet $\Sigma := \mathcal{E} \times \{r, w\} \times \mathcal{D}$. There is a a partial function $\pi : \mathscr{E}_{\mathbf{newref}} \rightharpoonup \Sigma$ as follows:

$$\begin{array}{rcl} \pi(\mathsf{w}(\!|\{([\mathsf{e}], \mathsf{d})^{@3}, _\}|)) & = & (\mathsf{e}, w, \mathsf{d}) \\ \pi(\mathsf{r}(\!|\{([\mathsf{e}], _)^{@5}, (_, \mathsf{d})^{@2}\}|)) & = & (\mathsf{e}, r, \mathsf{d}) \end{array}$$

and undefined everywhere else. We write $\mathsf{Tr}(\rho) = \rho \upharpoonright \pi$.

A memory trace is **consistent** when (1) exponential signatures occurring in it are all distinct, and (2) each read reads the last value written before, or zero if there are no writes.

Lemma D.39. *Consider a run $\rho : \emptyset \xrightarrow{s}\!\!\!\!\rightarrow \alpha$ of* **newref** *such that s is a play. Then:*

- *$\mathsf{Tr}(\rho)$ is a consistent memory trace.*
- *If ρ is not empty, then there is a unique tokil $(\mathsf{s}, \mathsf{d})^{@2}$ in α such that: if $\mathsf{Tr}(\rho)$ is empty then $\mathsf{s} = []$ and $\mathsf{d} = 0$, otherwise $\mathsf{s} = [\mathsf{e}]$ with e and d the components of the last operation in $\mathsf{Tr}(\rho)$.*
- *$|s|$ has the shape $((\uplus_{\mathsf{e}\in I}\mathsf{e} :: x_{\mathsf{e}}) \multimap y) \vdash z$ with $z \sqsubseteq y$ and $x_{\mathsf{e}} \in \mathscr{C}(\mathbf{V})$ such that:*
  - *if x_{e} is non-empty then e occurs in $\mathsf{Tr}(\rho)$ and the value coincide in the case of a read.*

– *For every signature* e *occurring in* $\mathsf{Tr}(\rho)$, x_{e} *is non-empty.*

Proof. We proceed by induction on ρ, the base case being trivial. We assume $\rho = \rho' \cdot \mathbf{f}$ with $\mathbf{f} : \alpha \longrightarrow \beta$. For the visible transitions $r_{\vdash}(-)$ and $\ell_{\vdash} r_{\multimap}(-)$, this is a proof similar to copycat.

- If $\mathbf{f}$ is on $\ell_{\vdash}\ell_{\multimap} w_{\mathbb{V}}\mathsf{Q}^-$ or $\ell_{\vdash}\ell_{\multimap} r_{\mathbb{V}}\mathsf{Q}^-$, there is nothing to add to the induction hypothesis.
- If $\mathbf{f} = \gamma \uplus \mathsf{w}\{([\mathsf{e}], \mathsf{d})^{@2}, ([\mathsf{e}'], \mathsf{d}')^{@3}\}$: then $\mathsf{Tr}(\rho) = \mathsf{Tr}(\rho')([\mathsf{e}'], w, \mathsf{d}')$ is still a consistent trace. Moreover, from the tokil $([\mathsf{e}'], \mathsf{d}')^{@3}$, we deduce that in ρ' there must be a visible transition with move $(\ell_{\vdash}\ell_{\multimap} w_{\mathbb{V}}\mathsf{Q}^-, [\mathsf{e}], \mathsf{d}')$.
- If $\mathbf{f} = \gamma \uplus \mathsf{w}\{([\mathsf{e}], \mathsf{d})^{@2}, ([\mathsf{e}'], \bullet)^{@5}\}$: the same line of reasoning works, except that $\mathsf{Tr}(\rho) = \mathsf{Tr}(\rho')([\mathsf{e}', r, \mathsf{d}])$ is no longer automatically consistent. However, by induction we know that in α there is a unique token at location 2, and that its value is the last value written or zero if there is not any – which shows that $\mathsf{Tr}(\rho)$ is indeed consistent.
- If $\mathbf{f} = \gamma \uplus \ell_{\vdash}\ell_{\multimap} w_{\mathbb{V}}\mathsf{A}([\mathsf{e}, \mathsf{d}]^{@4})$: the first two conditions are trivially true. Moreover, since $([\mathsf{e}], \mathsf{d})^{@4}$ belongs to α, there must have been a transition w in α before that put it there. That shows that there must be an element in $\mathsf{Tr}(\rho)$ with exponential token e as desired. □

Lemma D.40. **newref** *is a Petri strategy on* $!\mathbf{V} \multimap \mathbf{X} \vdash \mathbf{X}$.

Proof. Negativity and receptivity follow by inspection of the net and transition tables.

Strong-safety. Consider a run $\rho : \emptyset \xrightarrow{s}\!\!\!\!\rightarrow \alpha$ such that s is a play, that can be extended by a transition $\mathbf{f} : \alpha \longrightarrow \beta$ that is negative or neutral, with $\mathsf{play}(\rho\mathbf{f})$ being a play.

Initial question. If $\mathbf{f}$ is a negative transition $\alpha \xrightarrow{a} \beta$ on the address $r_{\vdash}\mathsf{Q}^-$. Then necessarily $\rho = \epsilon$ and so $\mathsf{Coll}(\rho) = \emptyset$.

Final return. If $\mathbf{f}$ is a negative transition $\alpha \xrightarrow{a} \beta$ on the address $\ell_{\vdash} r_{\multimap}\mathsf{A}^-$: trivial since the function of this transition is simply the identity, it follows from sa being a play hence non-repetitive.

Request. If $\mathbf{f}$ is a negative transition $\alpha \xrightarrow{a} \beta$ on the address $\ell_{\vdash}\ell_{\multimap} w_{\mathbb{V}}\mathsf{Q}^-$ or $\ell_{\vdash}\ell_{\multimap} r_{\mathbb{V}}\mathsf{Q}^-$. In both cases, the transition function is again the identity, so we can conclude by the same argument.

Atomic operation. If $\mathbf{f}$ arises from w or r. The two cases being symmetric, we only show for w. From the Petri structure, we get that $\alpha = \gamma \uplus \{([\mathsf{e}], d)^{@3}, (\mathsf{s}, \mathsf{d}')^{@2}\}$ and $\beta = \gamma \uplus \{([\mathsf{e}], \mathsf{d})^{@2}, ([\mathsf{e}], \checkmark)^{@6}\}$.

We show that $\mathsf{new}(\mathbf{f})$ is fresh in ρ. For the token in location 6, which is always in $\mathsf{new}(\mathbf{f})$, only the transition w writes to 6, so if the tokil $([\mathsf{e}], \checkmark)^{@6}$ appeared before in ρ, it means that there would be already a tokil $([\mathsf{e}], \mathsf{d}'')^{@3}$ in $\mathsf{Coll}(\rho)$. This is not possible because 3 is only fed via the negative transition on $\ell_{\vdash}\ell_{\multimap} w_{\mathbb{V}}\mathsf{Q}^-$. This means that Opponent would have played $(\ell_{\vdash}\ell_{\multimap} w_{\mathbb{V}}\mathsf{Q}^-, [\mathsf{e}], \mathsf{d}'')$ which violates the fact that s is non-repetitive (if $\mathsf{d} = \mathsf{d}''$) or that s is a play (if $\mathsf{d} \neq \mathsf{d}''$ as those moves are in conflict).

Finally, if $([\mathsf{e}], \mathsf{d})^{@2}$ is in $\mathsf{Coll}(\rho)$, then it means that a previous instance of w or r produced it, which means that there must have been a tokil $([\mathsf{e}], \mathsf{d})^{@3}$ (for w) or $([\mathsf{e}], \bullet)^{@5}$ (for r). That implies there has been two Opponent moves on addresses of the form $\ell_{\vdash}\ell_{\multimap}(-)$ with the same exponential address, which is not allowed by the game as they are all in conflict.

Valid. Consider a run $\rho : \emptyset \xrightarrow{s}\!\!\!\!\rightarrow \alpha$ with s a play, and a positive extension $\mathbf{f} : \alpha \xrightarrow{a} \beta$. There are several cases depending on the address of a:

- If a is on $r_{\vdash}\mathsf{A}$: easy since ρ is non empty it must contain its justifying move. Moreover a or a conflicting move with a cannot occur in s, since we simply forward moves received from address $\ell_{\vdash} r_{\multimap}\mathsf{A}$.
- If a is on $\ell_{\vdash} r_{\multimap}\mathsf{Q}$: same reasoning.
- If a is for instance on $\ell_{\vdash}\ell_{\multimap} w_{\mathbb{V}}\mathsf{A}$ (the case for $r_{\mathbb{V}}$ is similar). This means that in location 4, there must be a tokil $([\mathsf{e}], \checkmark)^{@4}$. That tokil proves that, there must be an entry (e, w, d) (for some d) in $\mathsf{Tr}(\rho)$. By Lemma D.39, we know that in s there must be justifying move $(\ell_{\vdash}\ell_{\multimap} w_{\mathbb{V}}\mathsf{Q}, [\mathsf{e}], d)$.

Moreover, if a or a conflicting move would be already present in a, then we could apply the same reasoning and find a contradiction with the fact that $\mathsf{Tr}(\rho)$ cannot repeat twice the same exponential token. □

E THE UNFOLDING

We provide some detailed proofs of the unfolding to strategies.

E.1 Construction of the Unfolding

Fix a game A, and a Petri strategy $\boldsymbol{\sigma} : A$. First, for a valid run $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\sigma}} \alpha$, we write $\mathsf{post}(\rho) = \alpha$. If $\mathsf{x} \in \mathsf{Hist}(\boldsymbol{\sigma})$, we write $\mathsf{post}(\mathsf{x}) = \mathsf{post}(\rho)$ for any ρ such that $\mathsf{x} = \mathscr{E}(\rho)$. This is justified by:

Lemma E.1. *Consider $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\sigma}} \alpha$ and $\rho' : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\sigma}} \alpha'$ valid runs such that $\mathscr{E}(\rho) = \mathscr{E}(\rho')$. Then, $\alpha = \alpha'$.*

Proof. Exploiting *strong safety*, it is immediate by induction on ρ that:

$$\alpha = (\uplus\{\mathsf{post}(\mathbf{e}) \mid \mathbf{e} \in \mathscr{E}(\rho)\}) \setminus (\uplus\{\mathsf{pre}(\mathbf{e}) \mid \mathbf{e} \in \mathscr{E}(\rho)\})$$

from which the result immediately follows. □

We aim to prove that valid runs exactly correspond to linearizations of histories. In this section, we write $\preceq_\rho$ for $(\lhd_{A \vdash B} \cup \lhd_{\boldsymbol{\sigma}})^*$.

Lemma E.2. *Consider ρ a valid run of $\boldsymbol{\sigma}$ of the form $\rho = \rho_0 \cdot (\mathbf{e} \uplus \alpha) \cdot (\mathbf{e}' \uplus \alpha')$. If $\mathbf{e}$ is maximal in $\mathscr{E}(\rho)$ for $\preceq_\rho$, then $\rho_0 \cdot (\mathbf{e}' \uplus \beta') \cdot (\mathbf{e} \uplus \beta)$ is a valid run for some β, β'.*

Proof. First, we show that for $\beta' = \mathsf{post}(\rho_0) \setminus \mathsf{pre}(\mathbf{e}')$, ρ_0 extends by $\mathbf{e}' \uplus \beta'$ which means showing:

$$\begin{array}{lrcl} (1) & \beta' \cap \mathsf{pre}(\mathbf{e}') & = & \emptyset \\ (2) & \mathsf{pre}(\mathbf{e}') & \subseteq & \mathsf{post}(\rho_0) \\ (3) & \beta' \cap \mathsf{post}(\mathbf{e}') & = & \emptyset \end{array}$$

First, (1) is by construction of β'. For (2), consider $e \in \mathsf{pre}(\mathbf{e}')$. Then e must either be in $\alpha \subseteq \mathsf{post}(\rho_0)$, or in $\mathsf{post}(\mathbf{e})$. If it is in $\mathsf{post}(\mathbf{e})$, then it cannot be a token *produced* by $\mathbf{e}$ (i.e. in $\mathsf{new}(\mathbf{e})$) as $\mathbf{e}$ and $\mathbf{e}'$ are incomparable. So it must be in $\mathsf{pre}(\mathbf{e}) \subseteq \mathsf{post}(\rho_0)$ as desired. For (3), consider $e \in \beta' \cap \mathsf{post}(\mathbf{e}')$. The tokil e must be in $\mathsf{new}(\mathbf{e}')$, which implies since $\mathbf{e}$ and $\mathbf{e}'$ are incomparable that e does not appear in $\mathsf{pre}(\mathbf{e})$. Since $e \in \mathsf{post}(\rho_0)$, it must be that $e \in \alpha$. Since α' must be disjoint from $\mathsf{post}(\mathbf{e}')$, we have $e \in \mathsf{pre}(\mathbf{e}')$ which is absurd. Hence, $\rho_1 = \rho_0 \cdot (\mathbf{e}' \uplus \beta')$ is indeed a valid run.

We now let $\beta = \mathsf{post}(\rho_1) \setminus \mathsf{pre}(\mathbf{e})$ and must prove (1) $\mathsf{pre}(\mathbf{e}) \subseteq \mathsf{post}(\rho_1)$; and (2) $\beta \cap \mathsf{post}(\mathbf{e}) = \emptyset$. For (1), if $e \in \mathsf{pre}(\mathbf{e})$, then e is in $\mathsf{post}(\rho_0)$. As a result, either e is not in $\mathsf{pre}(\mathbf{e}')$, which implies that $e \in \beta'$ hence $e \in \mathsf{post}(\rho_1)$ (as desired), or $e \in \mathsf{pre}(\mathbf{e}')$ as well. In the second case, we have then $e \in \mathsf{pre}(\mathbf{e}) \cap \mathsf{pre}(\mathbf{e}')$. Because, in ρ, $\mathbf{e}$ comes before $\mathbf{e}'$, this implies that e cannot be eaten by $\mathbf{e}$, in other words $e \in \mathsf{post}(\mathbf{e})$. This implies $e \notin \mathsf{eat}(\mathbf{e}')$ as the two transitions are incomparable, i.e. $e \in \mathsf{post}(\mathbf{e}') \subseteq \mathsf{post}(\rho_1)$. For (2), consider $e \in \beta \cap \mathsf{post}(\mathbf{e})$, ie. in particular $e \in \mathsf{new}(\mathbf{e})$. As $e \in \mathsf{post}(\rho_1)$, e either belongs to β' or $\mathsf{post}(\mathbf{e}')$. In the first case, this means that $e \in \mathsf{post}(\rho_0) \cap \mathsf{post}(\mathbf{e})$, which can only be if $e \in \mathsf{pre}(\mathbf{e})$ which is absurd. In the second case, it means that $e \in \mathsf{post}(\mathbf{e}')$, which in turn means that $e \in \mathsf{pre}(\mathbf{e}')$ as e is produced by $\mathbf{e}$ so it cannot be produced by $\mathbf{e}'$ as well by strong safety. But that is not possible either as it would imply a dependency from $\mathbf{e}$ to $\mathbf{e}'$. □

Lemma E.3. *Consider $\mathsf{x} \in \mathsf{Hist}(\boldsymbol{\sigma})$ and $\mathbf{e}$ a maximal element of x. Then, there exists a valid run ρ ending in $\mathbf{e} \uplus \alpha$ (for some context α) such that $\mathscr{E}_\rho = \mathsf{x}$.*

Proof. Consider a run ρ_0 spanning x, which must have the shape:

$$\rho_0 = \rho_1 \cdot (\mathbf{e} \uplus \alpha) \cdot (\mathbf{e}_1 \uplus \alpha_1) \cdot \ldots \cdot (\mathbf{e}_n \uplus \alpha_n)\,.$$

We proceed by induction on n. If $n = 0$, then $\mathbf{e}$ already occurs at the end of ρ. For $n+1$, we consider ρ' the prefix of ρ where the last transition has been removed. By IH, we get a run χ with $\mathscr{E}(\chi) = \mathsf{x} \setminus \{\mathbf{e}_{n+1}\}$ and χ ends with $\mathbf{e}$. By Lemma E.1, we have $\mathrm{post}(\chi) = \mathrm{post}(\rho')$; from that it is immediate that $\chi \cdot (\mathbf{e}_{n+1} \uplus \alpha_{n+1})$ is a valid run, and we conclude by Lemma E.2. □

Lemma E.4. *Consider* $\mathsf{x} \in \mathsf{Hist}(\boldsymbol{\sigma})$.
Then, valid runs ρ *such that* $\mathsf{x} = \mathscr{E}(\rho)$ *exactly correspond to linearizations of* x.

Proof. Clearly, all runs preserve $\leq_{\mathsf{x}}$. For the converse, for any transition $\mathbf{e}$ maximal in $\mathscr{E}(\mathsf{x})$, we obtain a run where it is played last by taking any valid run ρ such that $\mathsf{x} = \mathscr{E}_\rho$, and pushing $\mathbf{e}$ to the end via local permutations – maximality of $\mathbf{e}$ ensures that there is no obstruction – see Lemma E.3. Iterating this process, we can indeed obtain any linearization. □

We can now show the correctness of $\preceq_\rho$:

Proposition 4.11. *For* ρ *a valid run of* $\boldsymbol{\sigma}$, $\preceq_\rho = (\lhd_{A \vdash B} \cup \lhd_{\boldsymbol{\sigma}})^*$.

Proof. It is a simple corollary of Lemma E.4. Indeed suppose that $\mathbf{e} \preceq_\rho \mathbf{e}'$. Then, all equivalent ρ' are linearisations of $\preceq_\rho$, i.e. in which $\mathbf{e}$ must occur before $\mathbf{e}'$. Conversely, if $\mathbf{e} \leq_\rho \mathbf{e}'$, then there cannot be any linearisation of $\preceq_\rho$ where $\mathbf{e}'$ occurs before $\mathbf{e}$, which implies the desired result. □

E.2 Hiding

Rigid families seem to have remained in the concurrency theory folklore for a while and to have first appeared in published form in [Castellan et al. 2014a,b; Hayman 2014].

Consider $q = (|q|, \leq_q)$ and $p = (|p|, \leq_p)$ finite partial orders. We say that q is **rigidly included** in p, written $q \hookrightarrow p$, if $|q| \subseteq |p|$, and if that inclusion: *(1)* preserves down-closed sets, *i.e.* $\mathscr{C}(q) \subseteq \mathscr{C}(p)$; and *(2)* preserves causality, *i.e.* for all $e, e' \in |q|$, if $e \leq_q e'$ then $e \leq_p e'$ as well.

Definition E.5. A **rigid family** $\mathcal{F}$ is a non-empty set of finite partial orders which is:

rigid-closed: if $p \in \mathcal{F}$ and $q \hookrightarrow p$, then $q \in \mathcal{F}$,
binary-compatible: if $X \subseteq_f \mathcal{F}$, then $X\uparrow$ iff for all $q, p \in X$, $\{q, p\}\uparrow$.

where $X\uparrow$ means that there is $r \in \mathcal{F}$ such that for all $q \in X$, $q \hookrightarrow r$.

We added *binary-compatible* to the definition, to match our event structures with binary conflict.

A rigid family $\mathcal{F}$ collects causal executions. Particularly interesting are the **primes** of $\mathcal{F}$, *i.e.* those $q \in \mathcal{F}$ with a top element $\mathrm{top}(q) = e$: those can be thought of as a single event e, with a causal history leading to e. Indeed, the reconstructed event structure will have the primes as events:

Proposition E.6. *For* $\mathcal{F}$ *a rigid family, the data* $\mathrm{Pr}(\mathcal{F}) = (|\mathrm{Pr}(\mathcal{F})|, \leq_{\mathrm{Pr}(\mathcal{F})}, \#_{\mathrm{Pr}(\mathcal{F})})$ *defined by:*

$$\begin{array}{rcl} |\mathrm{Pr}(\mathcal{F})| & = & \{q \in \mathcal{F} \mid q \text{ prime}\} \\ q \leq_{\mathrm{Pr}(\mathcal{F})} p & \Leftrightarrow & q \subseteq p \\ \neg(q \#_{\mathrm{Pr}(\mathcal{F})} p) & \Leftrightarrow & \{q, p\}\uparrow, \end{array}$$

is an event structure with $\chi_{\mathcal{F}} : \mathscr{C}(\mathrm{Pr}(\mathcal{F})) \cong \mathcal{F}$ *an order-isomorphism.*

The proof is routine – $\chi_{\mathcal{F}}$ takes $x \in \mathscr{C}(\mathrm{Pr}(\mathcal{F}))$ to its sup $\vee x \in \mathcal{F}$ obtained as the (necessarily) compatible union of all partial orders in x, while $\chi_{\mathcal{F}}^{-1}$ takes $q \in \mathcal{F}$ to the set of primes below q.

Proposition E.7. *The set comprising all* $\mathscr{E}(\mathsf{x})$ *for* $\mathsf{x} \in \mathsf{Hist}(\boldsymbol{\sigma})$, *is a rigid family written* $\mathscr{E}(\boldsymbol{\sigma})$.
Moreover, $\mathscr{E}(\boldsymbol{\sigma})$ *(ordered by rigid inclusion) is order-isomorphic to* $\mathsf{Hist}(\boldsymbol{\sigma})$ *(ordered by inclusion).*

Proof. First, the claimed order-isomorphism is clear by construction.

Rigid-closed. Now if $p \in \mathscr{E}(\sigma)$ and $q \hookrightarrow p$, by Lemma E.4 there is a valid run ρ playing q first. Truncating ρ after q, we get ρ' such that $\mathscr{E}(\rho') = q$ by construction.

Binary-compatible. Take $X \subseteq_f \mathsf{Hist}(\sigma)$. Clearly, if *(1)* there are $\mathsf{x}, \mathsf{y} \in X$, visible instantiated transitions $\mathbf{e}$ in x and $\mathbf{e}'$ in y labelled by conflicting events of $A \vdash B$; or *(2)* there are $\mathsf{x}, \mathsf{y} \in X$, $\mathbf{e} : \alpha \longrightarrow_\sigma \beta$ in x and $\mathbf{e}' : \alpha' \longrightarrow_\sigma \beta'$ in y such that $\alpha \cap \alpha' \neq \emptyset$; then there cannot be a valid run witnessing $\cup X$: *(1)* would contradict validity of the run, while *(2)* would contradict *strong safety* as the same tokil would have to be consumed twice. Reciprocally, if we have neither *(1)* nor *(2)*, then any valid runs $(\rho_{\mathsf{x}})_{\mathsf{x} \in X}$ may be directly "zipped" into a valid run witnessing $\cup X \in \mathsf{Hist}(\sigma)$.

This concludes the proof, as it brings compatibility of $X \subseteq_f \mathscr{E}(\sigma)$ to pairwise compatibility. □

Proposition E.8. *The event structure $\mathscr{U}(\sigma) = \mathrm{Pr}(\mathscr{E}(\sigma)) \downarrow \mathscr{V}_\sigma$, equipped with the display map*

$$\begin{array}{rcccl} \partial_{\mathscr{U}(\sigma)} & : & |\mathscr{U}(\sigma)| & \to & |A \vdash B| \\ & & q & \mapsto & \partial_\sigma(\mathsf{top}(q)) \end{array}$$

is a strategy in the sense of Definition 4.4. Moreover, $\mathscr{U}(\sigma)$ is negative if σ is.

Proof. It remains to prove *courteous*, *receptive* and *negative*. From the order-isos $\mathscr{C}(\mathscr{U}(\sigma)) \cong \mathscr{C}^{\mathscr{V}_\sigma}(\mathrm{Pr}(\mathscr{E}(\sigma)))$ and $\mathscr{C}(\mathrm{Pr}(\mathscr{E}(\sigma))) \cong \mathscr{E}(\sigma)$, we get an order-iso:

$$K_\sigma : \mathscr{C}(\mathscr{U}(\sigma)) \cong \mathscr{E}^V(\sigma) \tag{5}$$

We now prove the remaining conditions.

Courteous. Consider $\mathsf{x}_1 \rightarrowtriangle_{\mathscr{U}(\sigma)} \mathsf{x}_2$ such that $\mathsf{pol}(\mathsf{x}_1) = +$ or $\mathsf{pol}(\mathsf{x}_2) = -$. So $\mathsf{x}_1, \mathsf{x}_2 \in \mathscr{E}(\sigma)$ with respective top elements $\mathsf{top}(\mathsf{x}_1) = \mathbf{e}_1$ and $\mathsf{top}(\mathsf{x}_2) = \mathbf{e}_2$, such that $\mathsf{pol}(\mathbf{e}_1) = +$ or $\mathsf{pol}(\mathbf{e}_2) = -$. By definition, $\mathsf{x}_1 \hookrightarrow \mathsf{x}_2$, so that $\mathbf{e}_1, \mathbf{e}_2 \in \mathsf{x}_2$ with $\mathbf{e}_1 <_{\mathsf{x}_2} \mathbf{e}_2$. By definition, this means there is a sequence

$$\mathbf{e}_1 \lhd_{\mathsf{x}_2} \mathbf{e}'_1 \lhd_{\mathsf{x}_2} \dots \mathbf{e}'_n \lhd_{\mathsf{x}_2} \mathbf{e}_2$$

where, each $\lhd_{\mathsf{x}_2}$ is either $\lhd_\sigma$ or $\lhd_A$. Now, seeking a contradiction, assume $n \geq 1$. Assume first that $\mathbf{e}_1$ is positive. Then, we cannot have $\mathbf{e}_1 \lhd_{\mathsf{x}_2} \mathbf{e}'_1$ as $\mathsf{post}(\mathbf{e}_1) = \mathsf{new}(\mathbf{e}_1) = \emptyset$. But we also cannot have $\mathbf{e}_1 \lhd_A \mathbf{e}'_1$, as $\mathbf{e}_1$ is visible but not $\mathbf{e}'_1$. Assuming that $\mathbf{e}_2$ is negative is symmetric: we cannot have $\mathbf{e}'_n \lhd_{\mathsf{x}_2} \mathbf{e}_2$ as $\mathsf{pre}(\mathbf{e}_2) = \mathsf{eat}(\mathbf{e}_2) = \emptyset$, and we cannot have $\mathbf{e}'_n \lhd_A \mathbf{e}_2$ because $\mathbf{e}_2$ is visible but not $\mathbf{e}'_n$. So, $n = 0$ and we have $\mathbf{e}_1 \lhd_{\mathsf{x}_2} \mathbf{e}_2$. But again, for the same reason this cannot be because $\mathbf{e}_1 \lhd_\sigma \mathbf{e}_2$, so $\mathbf{e}_1 \lhd_A \mathbf{e}_2$, which means $\partial_\sigma(\mathbf{e}_1) \rightarrowtriangle_A \partial_\sigma(\mathbf{e}_2)$. Hence, $\partial_{\mathscr{U}(\sigma)} \rightarrowtriangle_A \partial_{\mathscr{U}(\sigma)}$.

Receptive. Consider $x \in \mathscr{C}(\mathscr{U}(\sigma))$ and $\partial_{\mathscr{U}(\sigma)}(x) \vdash_A a^-$. So we have $K_\sigma(x) \in \mathscr{E}^V(\sigma)$ with $\partial_\sigma(K_\sigma(x)) \vdash_A a^-$. We show that there is a unique matching extension $\mathbf{e}^-$ of $K_\sigma(x)$, and conclude by the fact that K_σ is an order-isomorphism. For *existence*, consider $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_\sigma \alpha$ a valid run such that $K_\sigma(x) = \mathscr{E}(\rho)$. Consider $s \in \mathsf{play}(\rho)$, so in particular $s \in \mathsf{Plays}(A)$. By hypothesis, $sa \in \mathsf{Plays}(A)$ as well. So by condition *receptive* of Petri strategies, there is a unique $\mathbf{e}^- : \emptyset \xmapsto{a^-}_\sigma \beta$ for some $\beta \cap \alpha = \emptyset$, so that $\rho' = \rho(\mathbf{e}^- \uplus \gamma)$ is a valid run for some γ; providing the expected extension of $K_\sigma(x)$. *Uniqueness* follows immediately from uniqueness of $\mathbf{e}^-$.

Negative. Consider $\mathsf{x} \in |\mathscr{U}(\sigma)|$ minimal. This means that x is a prime history with exactly one visible event $\mathbf{e}$, with $\mathbf{e} = \mathsf{top}(\mathsf{x})$. So there is a run

$$\mathbf{f}_1 \dots \mathbf{f}_n(\mathbf{e} \uplus \gamma) : \emptyset \longrightarrow\!\!\!\!\rightarrow_\sigma \alpha\,,$$

but by condition *negative* of Petri strategies, $\mathbf{f}_1$ cannot be neutral and have no preconditions. So $n = 0$, and we have a one-event run $\mathbf{e} : \emptyset \longrightarrow\!\!\!\!\rightarrow_\sigma \alpha$. But likewise, by condition *negative* of Petri strategies this entails that $\mathbf{e}$ is negative, so x is negative as required. □

Lemma E.9. *The order-isomorphism (5) of Lemma E.8 specializes to $K^+_\sigma : \mathscr{C}^+(\mathscr{U}(\sigma)) \cong \mathscr{E}^+(\sigma)$ s.t. for all $x \in \mathscr{C}^+(\mathscr{U}(\sigma))$, we have $\partial_{\mathscr{U}(\sigma)}(x) = \partial_\sigma(K^+_\sigma(x))$ the labels of visible transitions in $K^+_\sigma(x)$.*

E.3 The Unfolding as a Functor

Proposition E.10. *Consider $\boldsymbol{\sigma} : A \vdash B$ and $\boldsymbol{\tau} : B \vdash C$ Petri strategies. Then, there is an order-iso:*

$$\begin{array}{rcl} (- \odot -) & : & \{(\mathsf{x}^{\boldsymbol{\tau}}, \mathsf{x}^{\boldsymbol{\sigma}}) \in \mathscr{E}^+(\boldsymbol{\tau}) \times \mathscr{E}^+(\boldsymbol{\sigma}) \\ & & \quad \mid \textit{causally compatible}\} \\ & \cong & \mathscr{E}^+(\boldsymbol{\tau} \odot \boldsymbol{\sigma}) \end{array}$$

such that for $\mathsf{x}^{\boldsymbol{\sigma}} \in \mathscr{E}^+(\boldsymbol{\sigma}), \mathsf{x}^{\boldsymbol{\tau}} \in \mathscr{E}^+(\boldsymbol{\tau})$ causally compatible, $\partial_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}}(\mathsf{x}^{\boldsymbol{\tau}} \odot \mathsf{x}^{\boldsymbol{\sigma}}) = \mathsf{x}^{\boldsymbol{\sigma}}_A \vdash \mathsf{x}^{\boldsymbol{\tau}}_C$.

Proof. For $\mathsf{x}^{\boldsymbol{\sigma}} \in \mathscr{E}(\boldsymbol{\sigma})$ and $\mathsf{x}^{\boldsymbol{\tau}} \in \mathscr{E}(\boldsymbol{\tau})$, we set $\mathsf{x}^{\boldsymbol{\tau}} \odot \mathsf{x}^{\boldsymbol{\sigma}}$ as the set of instantiated transitions obtained from $\mathsf{x}^{\boldsymbol{\sigma}}$ and $\mathsf{x}^{\boldsymbol{\tau}}$ by the rules of Figure 44 (following Lemma D.3). We prove by induction that for all $\mathsf{x}^{\boldsymbol{\sigma}} \in \mathscr{E}(\boldsymbol{\sigma})$ and $\mathsf{x}^{\boldsymbol{\tau}} \in \mathscr{E}(\boldsymbol{\tau})$ causally compatible, then $\mathsf{x}^{\boldsymbol{\tau}} \odot \mathsf{x}^{\boldsymbol{\sigma}} \in \mathscr{E}(\boldsymbol{\tau} \odot \boldsymbol{\sigma})$, and

$$\mathsf{post}(\mathsf{x}^{\boldsymbol{\tau}} \odot \mathsf{x}^{\boldsymbol{\sigma}}) = \mathsf{post}(\mathsf{x}^{\boldsymbol{\sigma}}) +^{\odot} \mathsf{post}(\mathsf{x}^{\boldsymbol{\tau}}) \,.$$

If $\mathsf{x}^{\boldsymbol{\tau}} \odot \mathsf{x}^{\boldsymbol{\sigma}}$ is empty, there is nothing to prove. If $\mathsf{x}^{\boldsymbol{\sigma}}$ or $\mathsf{x}^{\boldsymbol{\tau}}$ have a maximal neutral instantiated transition, say *w.l.o.g.* that it is $\mathsf{x}^{\boldsymbol{\sigma}}$ with maximal $\mathbf{e} = t^0(\!|\mu|\!) \in \mathsf{x}^{\boldsymbol{\sigma}}$. Then, setting $\mathsf{y}^{\boldsymbol{\sigma}} = \mathsf{x}^{\boldsymbol{\sigma}} \setminus \{\mathbf{e}\}$ yields $\mathsf{y}^{\boldsymbol{\sigma}} \in \mathscr{E}(\boldsymbol{\sigma})$ by Proposition E.7; and with also $\mathsf{y}^{\boldsymbol{\tau}} = \mathsf{x}^{\boldsymbol{\tau}}$, it is direct that $\mathsf{y}^{\boldsymbol{\sigma}}$ and $\mathsf{y}^{\boldsymbol{\tau}}$ are still causally compatible. By IH, we have $\mathsf{y}^{\boldsymbol{\tau}} \odot \mathsf{y}^{\boldsymbol{\sigma}} \in \mathscr{E}(\boldsymbol{\tau} \odot \boldsymbol{\sigma})$ and $\mathsf{post}(\mathsf{y}^{\boldsymbol{\tau}} \odot \mathsf{y}^{\boldsymbol{\sigma}}) = \mathsf{post}(\mathsf{y}^{\boldsymbol{\sigma}}) +^{\odot} \mathsf{post}(\mathsf{y}^{\boldsymbol{\tau}})$. This means that there is a run $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \alpha$ projecting to $\rho_{\boldsymbol{\sigma}} : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\sigma}} \alpha_{\boldsymbol{\sigma}}$ with $\alpha_{\boldsymbol{\sigma}} = \mathsf{post}(\mathsf{y}^{\boldsymbol{\sigma}})$. Since $\mathbf{e}$ is enabled in $\mathsf{post}(\mathsf{y}^{\boldsymbol{\sigma}})$ it follows that $\ell^{\odot}(t^0)(\!|\ell^{\odot}(\mu)|\!)$ is enabled in $\mathsf{y}^{\boldsymbol{\tau}} \odot \mathsf{y}^{\boldsymbol{\sigma}}$, and

$$\rho(\ell^{\odot}(t^0)(\!|\ell^{\odot}(\mu)|\!) : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \beta$$

where by construction $\beta = \mathsf{post}(\mathsf{x}^{\boldsymbol{\sigma}}) +^{\odot} \mathsf{post}(\mathsf{x}^{\boldsymbol{\tau}})$ as needed. The symmetric reasoning applies if $\mathsf{x}^{\boldsymbol{\tau}}$ has a maximal neutral instantiated transition – so assume all maximal transitions in $\mathsf{x}^{\boldsymbol{\sigma}}, x^{\boldsymbol{\tau}}$ visible.

Now, by causal compatibility of $\mathsf{x}^{\boldsymbol{\sigma}}$ and $\mathsf{x}^{\boldsymbol{\tau}}$, there is an element of $\mathsf{x}_A \parallel \mathsf{x}_B \parallel \mathsf{x}_C$ (following the notations of Section C.2.2) which is maximal for $\lhd$. If it is in x_A, it has the form $\partial^{\ell}_{\boldsymbol{\sigma}}(\mathbf{e})$ for $\mathbf{e} \in \mathsf{x}^{\boldsymbol{\sigma}}$ positive or negative. In both cases, the same argument as in the neutral case applies (with the additional observation that the obtained run yields a valid play from the hypothesis). The reasoning is the same if it is in x_C. The last (key) case is if it is in x_B. Then there are instantiated transitions

$$t^+(\!|\mu|\!) : \mu \xmapsto{\not{r}m}_{\boldsymbol{\sigma}} \emptyset \,, \qquad\qquad t^-(\!|(\mathsf{s},\mathsf{d})|\!) : \emptyset \xmapsto{\ell_{\vdash}m}_{\boldsymbol{\tau}} \nu \,,$$

or the dual – symmetric – situation, respectively maximal in $\mathsf{x}^{\boldsymbol{\sigma}}$ and $\mathsf{x}^{\boldsymbol{\tau}}$; and by necessity $m = (\mathsf{m}, \mathsf{s}, \mathsf{d})$ where $\partial_{\boldsymbol{\sigma}}(t^+) = \not{r}\mathsf{m}$, $\partial_{\boldsymbol{\tau}}(t^-) = \ell_{\vdash}\mathsf{m}$, $\delta\langle t^+\rangle(\mu) = (\mathsf{s}, \mathsf{d})$ and $\delta\langle t^-\rangle(\mathsf{s}, \mathsf{d}) = \nu$. Setting $\mathsf{y}^{\boldsymbol{\sigma}} \setminus \{t^+(\!|\mu|\!)\}$ and $\mathsf{y}^{\boldsymbol{\tau}} \setminus \{t^-(\!|(\mathsf{s},\mathsf{d})|\!)\}$, it is straightforward that they are still causally compatible histories. By IH, $\mathsf{y}^{\boldsymbol{\tau}} \odot \mathsf{y}^{\boldsymbol{\sigma}} \in \mathscr{E}(\boldsymbol{\tau} \odot \boldsymbol{\sigma})$ with $\mathsf{post}(\mathsf{y}^{\boldsymbol{\tau}} \odot \mathsf{y}^{\boldsymbol{\sigma}}) = \mathsf{post}(\mathsf{y}^{\boldsymbol{\sigma}}) +^{\odot} \mathsf{post}(\mathsf{y}^{\boldsymbol{\tau}})$. It follows that there is a run

$$\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \mathsf{post}(\mathsf{y}^{\boldsymbol{\tau}} \odot \mathsf{y}^{\boldsymbol{\sigma}}) \,.$$

Since $\mathsf{x}^{\boldsymbol{\sigma}} \in \mathscr{E}(\boldsymbol{\sigma})$ with $t^+(\!|\mu|\!)$ maximal and $\mathsf{x}^{\boldsymbol{\tau}} \in \mathscr{E}(\boldsymbol{\sigma})$ with $t^-(\!|(\mathsf{s},\mathsf{d})|\!)$ maximal, there are

$$\begin{array}{rcl} \xi_{\boldsymbol{\sigma}}(t^+(\!|\mu|\!) \uplus \gamma_{\boldsymbol{\sigma}}) & : & \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\sigma}} \mathsf{post}(\mathsf{x}^{\boldsymbol{\sigma}}) \,, \\ \xi_{\boldsymbol{\tau}}(t^-(\!|(\mathsf{s},\mathsf{d})|\!) \uplus \gamma_{\boldsymbol{\tau}}) & : & \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\tau}} \mathsf{post}(\mathsf{x}^{\boldsymbol{\tau}}) \,, \end{array}$$

valid runs by Lemma E.3, with $\mathsf{x}^{\boldsymbol{\sigma}} = \mathscr{E}_{\xi_{\boldsymbol{\sigma}}}$ and $\mathsf{x}^{\boldsymbol{\tau}} = \mathscr{E}_{\xi_{\boldsymbol{\tau}}}$. By Lemma E.1, $\mathsf{post}(\xi_{\boldsymbol{\sigma}}) = \mathsf{post}(\rho_{\boldsymbol{\sigma}}) = \mathsf{post}(\mathsf{y}^{\boldsymbol{\sigma}})$ and $\mathsf{post}(\xi_{\boldsymbol{\tau}}) = \mathsf{post}(\rho_{\boldsymbol{\tau}}) = \mathsf{post}(\mathsf{y}^{\boldsymbol{\tau}})$. It follows that the transition

$$(t^+ \circledast t^-)(\!|\ell^{\odot}(\mu)|\!) : \ell^{\odot}(\mu) \longmapsto_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \not{r}^{\odot}(\nu)$$

is enabled in $\mathsf{post}(\mathsf{y}^{\boldsymbol{\tau}} \odot \mathsf{y}^{\boldsymbol{\sigma}})$, hence it can be appended to ρ, witnessing $\mathsf{x}^{\boldsymbol{\tau}} \odot \mathsf{x}^{\boldsymbol{\sigma}} \in \mathscr{E}(\boldsymbol{\tau} \odot \boldsymbol{\sigma})$.

In the other direction, given $\mathsf{y} \in \mathscr{E}(\boldsymbol{\tau} \odot \boldsymbol{\sigma})$, consider a valid run $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\tau}\odot\boldsymbol{\sigma}} \alpha$ such that $\mathsf{y} = \mathscr{E}_{\rho}$. By Lemmas D.4 and D.6, we then have $\alpha = \alpha_{\boldsymbol{\sigma}} +^{\odot} \alpha_{\boldsymbol{\tau}}$ with

$$\rho_{\boldsymbol{\sigma}} : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\sigma}} \alpha_{\boldsymbol{\sigma}} \,, \qquad \rho_{\boldsymbol{\tau}} : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\tau}} \alpha_{\boldsymbol{\tau}}$$

valid runs. Recall that $\rho_{\boldsymbol{\sigma}} = \rho \upharpoonright \pi_{\boldsymbol{\sigma}}$ and $\rho_{\boldsymbol{\tau}} = \rho \upharpoonright \pi_{\boldsymbol{\tau}}$ – hence, setting $\mathsf{x}^{\boldsymbol{\sigma}} = \pi_{\boldsymbol{\sigma}}(\mathsf{y})$ and $\mathsf{x}^{\boldsymbol{\tau}} = \pi_{\boldsymbol{\tau}}(\mathsf{y})$, we have $\mathsf{x}^{\boldsymbol{\sigma}} = \mathscr{E}_{\rho_{\boldsymbol{\sigma}}}$ and $\mathsf{x}^{\boldsymbol{\tau}} = \mathscr{E}_{\rho_{\boldsymbol{\tau}}}$ so that $\mathsf{x}^{\boldsymbol{\sigma}} \in \mathscr{E}(\boldsymbol{\sigma})$ and $\mathsf{x}^{\boldsymbol{\tau}} \in \mathscr{E}(\boldsymbol{\tau})$. Causal compatibility is direct as ρ provides a linearization of $\lhd$.

It is direct that these constructions are inverse; it remains to show that they preserve +-covered histories. If $\mathsf{x}^{\boldsymbol{\sigma}}$ and $\mathsf{x}^{\boldsymbol{\tau}}$ causally compatible are +-covered, then consider $\mathbf{e}$ maximal in $\mathsf{x}^{\boldsymbol{\tau}} \odot \mathsf{x}^{\boldsymbol{\sigma}}$. If $\mathbf{e} = \ell^{\odot}(t^0)(\!|\ell^{\odot}(\mu)|\!)$, this directly contradicts +-coveredness of $\mathsf{x}^{\boldsymbol{\sigma}}$, and likewise for $\mathscr{r}^{\odot}(t^0)(\!|\mathscr{r}^{\odot}(\mu)|\!)$. If $\mathbf{e} = \ell^{\odot}(t^-)(\!|(\mathsf{s},\mathsf{d})|\!)$, then $t^-(\!|(\mathsf{s},\mathsf{d})|\!)$ is maximal in $\mathsf{x}^{\boldsymbol{\sigma}}$, contradiction – likewise for $\mathscr{r}^{\odot}(t^-)(\!|(\mathsf{s},\mathsf{d})|\!)$. If $\mathbf{e} = (t^+ \circledast t^-)(\!|\ell^{\odot}(\mu)|\!)$ with $t^+(\!|\mu|\!) \in \mathsf{x}^{\boldsymbol{\sigma}}$ and $t^-(\!|(\mathsf{s},\mathsf{d})|\!) \in \mathsf{x}^{\boldsymbol{\tau}}$, then $t^-(\!|(\mathsf{s},\mathsf{d})|\!)$ is maximal in $\mathsf{x}^{\boldsymbol{\tau}}$, contradiction – likewise, $\mathbf{e} = (t^- \circledast t^+)(\!|\mathscr{r}^{\odot}(\mu)|\!)$ leads to a contradiction. So, $\mathsf{x}^{\boldsymbol{\tau}} \odot \mathsf{x}^{\boldsymbol{\sigma}}$ is +-covered.

Reciprocally, assume $\mathsf{x}^{\boldsymbol{\tau}} \odot \mathsf{x}^{\boldsymbol{\sigma}}$ +-covered. Consider $\mathbf{e} \in \mathsf{x}^{\boldsymbol{\sigma}}$ maximal. If $\mathbf{e} = t^0(\!|\mu|\!)$, $\ell^{\odot}(t^0)(\!|\ell^{\odot}(\mu)|\!)$ is maximal in $\mathsf{x}^{\boldsymbol{\tau}} \odot \mathsf{x}^{\boldsymbol{\sigma}}$, contradiction. If $\mathbf{e} = t^-(\!|(\mathsf{s},\mathsf{d})|\!)$, then since $\mathsf{x}^{\boldsymbol{\tau}} \odot \mathsf{x}^{\boldsymbol{\sigma}}$ is +-covered, there is

$$\ell^{\odot}(t^-)(\!|(\mathsf{s},\mathsf{d})|\!) \rightarrowtriangle_{\mathsf{x}^{\boldsymbol{\tau}} \odot \mathsf{x}^{\boldsymbol{\sigma}}} \mathbf{e}'$$

and a direct case analysis shows that $\pi_{\boldsymbol{\sigma}}\mathbf{e}'$ is defined with $t^-(\!|(\mathsf{s},\mathsf{d})|\!) \rightarrowtriangle_{\mathsf{x}^{\boldsymbol{\sigma}}} \pi_{\boldsymbol{\sigma}}\mathbf{e}'$, contradicting the maximality of $\mathbf{e}$. The last case has $\mathbf{e}$ positive; and symmetrically, $\mathsf{x}^{\boldsymbol{\tau}}$ is +-covered. □

Proposition E.11. *For $\boldsymbol{\sigma} : A \vdash B$ and $\boldsymbol{\tau} : B \vdash C$ Petri strategies, $\mathscr{U}(\boldsymbol{\tau} \odot \boldsymbol{\sigma}) \cong \mathscr{U}(\boldsymbol{\tau}) \odot \mathscr{U}(\boldsymbol{\sigma})$.*

Proof. We may deduce preservation of composition simply by manipulating isos:

$$\begin{aligned}
& \mathscr{C}^+(\mathscr{U}(\boldsymbol{\tau} \odot \boldsymbol{\sigma})) \\
\cong\ & \mathscr{E}^+(\boldsymbol{\tau} \odot \boldsymbol{\sigma}) \\
\cong\ & \{(\mathsf{x}^{\boldsymbol{\sigma}}, \mathsf{x}^{\boldsymbol{\tau}}) \in \mathscr{E}^+(\boldsymbol{\sigma}) \times \mathscr{E}^+(\boldsymbol{\tau}) \mid \text{causally compatible}\} \\
\cong\ & \{(x^{\mathscr{U}(\boldsymbol{\sigma})}, x^{\mathscr{U}(\boldsymbol{\tau})}) \in \mathscr{C}^+(\mathscr{U}(\boldsymbol{\sigma})) \times \mathscr{C}^+(\mathscr{U}(\boldsymbol{\tau})) \\
& \qquad \mid \text{causally compatible}\} \\
\cong\ & \mathscr{C}^+(\mathscr{U}(\boldsymbol{\tau}) \odot \mathscr{U}(\boldsymbol{\sigma}))
\end{aligned}$$

via Lemma E.9, Proposition E.10, and Lemma E.9 – preservation of causal compatibility follows directly from the order-isomorphism. All these steps preserve displays to the game. By Proposition C.7, it follows that $\mathscr{U}(\boldsymbol{\tau} \odot \boldsymbol{\sigma}) \cong \mathscr{U}(\boldsymbol{\tau}) \odot \mathscr{U}(\boldsymbol{\sigma})$ as required. □

As detailed in Proposition E.11, it follows that unfolding preserves composition up to iso.

Next, we show the same for copycat. If A is an arena, then we have obvious bijections

$$\mathscr{E}^+_{\mathbf{cc}_A} \cong |A \vdash A|^+ \qquad\qquad \mathscr{E}^-_{\mathbf{cc}_A} \cong |A \vdash A|^- ,$$

and coercing silently through these, we have:

Lemma E.12. *Consider A an arena, and $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\mathbf{cc}_A} \alpha$ a valid run.*
Then, $\mathscr{E}_\rho = \mathsf{play}(\rho)$, with negative maximal transitions in bijection with α.

Proof. Straightforward by induction on ρ. □

Lemma E.13. *Consider A an arena. Then, we have the order-isomorphism*

$$\mathscr{E}^+(\mathbf{cc}_A) \cong \{x \vdash x \mid x \in \mathscr{C}(A)\}$$

with $\partial_{\mathbf{cc}_A}(x \vdash x) = x \vdash x$.

Proof. The isomorphism simply applies the bijection $\mathscr{E}_{\mathbf{cc}_A} \simeq |A|$. From left to right, recall first that by Lemma D.8, for $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\mathbf{cc}_A} \alpha$ a valid run, we have

$$|\mathsf{play}(\rho)| = x \vdash y\,, \qquad y \supseteq^- x \cap y \subseteq^+ x\,, \qquad \alpha = (y^- \setminus x) \uplus (x^+ \setminus y)\,.$$

By Lemma E.12, the history $\mathscr{E}_\rho$ is +-covered iff $\alpha = \emptyset$, *i.e.* $y^- \subseteq x$ and $x^+ \subseteq y$. But as $y \supseteq^- x \cap y \subseteq^+ x$ this entails $x = y$. In that case, $\partial_{\mathfrak{a}_A}(\mathscr{E}_\rho) = |\mathsf{play}(\rho)| = x \vdash x$ as needed. Reciprocally, for any $x \in \mathscr{C}(A)$, it is straightforward to build a valid run $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\mathfrak{a}_A} \emptyset$ s.t. $\mathscr{E}_\rho = x \vdash x$ as required. □

From that, preservation of copycat follows:

Proposition E.14. *Consider A an arena. Then, $\mathscr{U}(\mathfrak{a}_A) \cong \mathfrak{a}_A$.*

Proof. We compose label-preserving order-isomorphisms:

$$\begin{array}{rcl} \mathscr{C}^+(\mathscr{U}(\mathfrak{a}_A)) & \cong & \mathscr{E}^+(\mathfrak{a}_A) \\ & \cong & \{x \vdash x \mid x \in \mathscr{C}(A)\} \\ & \cong & \mathscr{C}^+(\mathfrak{a}_A) \end{array}$$

by Lemmas E.9 and E.13. From this it follows that $\mathscr{U}(\mathfrak{a}_A) \cong \mathfrak{a}_A$ by Lemma C.5. □

Corollary E.15. *We have a functor of precategories $\mathscr{U}$: PStrat $\to$ Strat.*

E.4 The Unfolding Preserves Operations

Next, we prove that the unfolding preserves all operations of the IPA-structure.

E.4.1 Tensor. Preservation of the tensor operation is easy via the following observation:

Lemma E.16. *Consider $\boldsymbol{\sigma} : A_1 \vdash B_1, \boldsymbol{\tau} : A_2 \vdash B_2$ Petri strategies. Then, we have an order-isomorphism*

$$(- \otimes -) : \mathscr{E}^+(\boldsymbol{\sigma}) \times \mathscr{E}^+(\boldsymbol{\tau}) \cong \mathscr{E}^+(\boldsymbol{\sigma} \otimes \boldsymbol{\tau})$$

s.t. $\partial_{\boldsymbol{\sigma}\otimes\boldsymbol{\tau}}(\mathsf{x}^\sigma \otimes \mathsf{x}^\tau) = (x_{A_1} \otimes x_{A_2}) \vdash (x_{B_1} \otimes x_{B_2})$ where $\partial_{\boldsymbol{\sigma}}(\mathsf{x}^\sigma) = x_{A_1} \vdash x_{B_1}$ and $\partial_{\boldsymbol{\tau}}(\mathsf{x}^\tau) = x_{A_2} \vdash x_{B_2}$.

Proof. Consider $\mathsf{x}^\sigma \in \mathscr{E}^+(\boldsymbol{\sigma})$ and $\mathsf{x}^\tau \in \mathscr{E}^+(\boldsymbol{\tau})$. By definition, there are valid runs

$$\rho^\sigma : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\sigma}} \alpha_{\boldsymbol{\sigma}}\,, \qquad\qquad \rho^\tau : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\tau}} \alpha_{\boldsymbol{\tau}}$$

such that $\mathsf{x}^\sigma = \mathscr{E}_{\rho^\sigma}$ and $\mathsf{x}^\tau = \mathscr{E}_{\rho^\tau}$. We define the history $\mathsf{x}^\sigma \otimes \mathsf{x}^\tau$ as

$$\begin{array}{rcl} \mathsf{x}^\sigma \otimes \mathsf{x}^\tau & = & \{\ell^\otimes(t^{0,+})(\!|\ell^\otimes(\mu)|\!) \mid t^{0,+}(\!|\mu|\!) \in \mathsf{x}^\sigma\} \\ & \uplus & \{\ell^\otimes(t^-)(\!|(\mathsf{s},\mathsf{d})|\!) \mid t^-(\!|(\mathsf{s},\mathsf{d})|\!) \in \mathsf{x}^\sigma\} \\ & \uplus & \{r^\otimes(t^{0,+})(\!|r^\otimes(\mu)|\!) \mid t^{0,+}(\!|\mu|\!) \in \mathsf{x}^\tau\} \\ & \uplus & \{r^\otimes(t^-)(\!|(\mathsf{s},\mathsf{d})|\!) \mid t^-(\!|(\mathsf{s},\mathsf{d})|\!) \in \mathsf{x}^\tau\}\,. \end{array}$$

This must be the history of a valid run – to show that, we build

$$\begin{array}{rcl} \ell^\otimes(\rho^\sigma) & : & \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\sigma}\otimes\boldsymbol{\tau}} \ell^\otimes(\alpha_{\boldsymbol{\sigma}})\,, \\ r^\otimes(\rho^\tau) \uplus \ell^\otimes(\alpha_{\boldsymbol{\sigma}}) & : & \ell^\otimes(\alpha_{\boldsymbol{\sigma}}) \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\sigma}\otimes\boldsymbol{\tau}} \ell^\otimes(\alpha_{\boldsymbol{\sigma}}) \uplus r^\otimes(\alpha_{\boldsymbol{\tau}}) \end{array}$$

which by concatenation (and Lemma D.13) yields a valid run $\rho^\sigma \otimes \rho^\tau : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\sigma}\otimes\boldsymbol{\tau}} \alpha_{\boldsymbol{\sigma}} +^\otimes \alpha_{\boldsymbol{\tau}}$; and it is immediate that $\mathsf{x}^\tau \otimes \mathsf{x}^\sigma = \mathscr{E}_{\rho^\tau \otimes \rho^\sigma}$. By definition of the causal ordering of instantiated transitions, it is also immediate that $\mathsf{x}^\tau \otimes \mathsf{x}^\sigma$ is +-covered; and that this preserves the labelling.

Reciprocally, for any $\mathsf{x} \in \mathscr{E}^+(\boldsymbol{\sigma} \otimes \boldsymbol{\tau})$ we consider the projections

$$\mathsf{x}^\sigma = \pi_{\boldsymbol{\sigma}}(\mathsf{x})\,, \qquad\qquad \mathsf{x}^\tau = \pi_{\boldsymbol{\tau}}(\mathsf{x})\,,$$

and it follows from Lemma D.14 that $\mathsf{x}^\sigma \in \mathscr{E}(\boldsymbol{\sigma})$ and $\mathsf{x}^\tau \in \mathscr{E}(\boldsymbol{\tau})$. From the definition of the causal ordering of instantiated transitions, x^σ and x^τ are still +-covered.

Finally, these two transformations are inverses as required. □

Again, from this we can conclude that the unfolding preserves the tensor.

Corollary E.17. *Consider $\boldsymbol{\sigma} : A_1 \vdash B_1$, $\boldsymbol{\tau} : A_2 \vdash B_2$ Petri strategies. Then, we have $\mathscr{U}(\boldsymbol{\sigma} \otimes \boldsymbol{\tau}) \cong \mathscr{U}(\boldsymbol{\sigma}) \otimes \mathscr{U}(\boldsymbol{\tau})$.*

Proof. We compose label-preserving isomorphisms:

$$\begin{aligned}
\mathscr{C}^+(\mathscr{U}(\boldsymbol{\sigma} \otimes \boldsymbol{\tau})) &\cong \mathscr{C}^+(\boldsymbol{\sigma} \otimes \boldsymbol{\tau}) \\
&\cong \mathscr{C}^+(\boldsymbol{\sigma}) \times \mathscr{C}^+(\boldsymbol{\tau}) \\
&\cong \mathscr{C}^+(\mathscr{U}(\boldsymbol{\sigma})) \times \mathscr{C}^+(\mathscr{U}(\boldsymbol{\tau})) \\
&\cong \mathscr{C}^+(\mathscr{U}(\boldsymbol{\sigma}) \otimes \mathscr{U}(\boldsymbol{\tau}))
\end{aligned}$$

by Lemmas E.9, E.16, Lemma E.9 again, and Proposition C.12. □

E.4.2 Renaming. Before detailing the unfolding of currying and promotion, we show that it preserves renaming. We have already established in Lemma D.19 that (global) renamings preserve valid runs. In order for renamings to preserve the unfolding, we must ensure that the dependency between instantiated transitions is preserved as well.

Lemma E.18. *Consider A, B games, $h = [f, (g_{\mathrm{m}})] : A \rightsquigarrow B$, $\boldsymbol{\sigma} : A$, and $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\sigma}} \alpha$ valid. For all $\mathbf{e}, \mathbf{e}' \in \mathscr{E}_\rho$, we have $\mathbf{e} \leq_\rho \mathbf{e}'$ iff $\mathbf{e}[h] \leq_{\rho[h]} \mathbf{e}'[h]$.*

Proof. Consider $\mathbf{e} \rightarrowtriangle_\rho \mathbf{e}'$. *W.l.o.g.* we assume that this dependency cannot be deduced otherwise by transitivity. By Lemma E.4, we can assume that $\mathbf{e}$ and $\mathbf{e}'$ appear subsequently in ρ.

Assume first $\mathbf{e} \rightarrowtriangle_A \mathbf{e}'$. By definition,

$$\mathbf{e} : \alpha \xmapsto{a}_{\boldsymbol{\sigma}} \beta\,, \qquad\qquad \mathbf{e}' : \alpha' \xmapsto{a'}_{\boldsymbol{\sigma}} \beta'$$

with $a \rightarrowtriangle_A a'$, while by construction, $\mathbf{e}[h] : \alpha \xmapsto{ha}_{\boldsymbol{\sigma}[h]} \beta$ and $\mathbf{e}'[h] : \alpha' \xmapsto{ha'}_{\boldsymbol{\sigma}[h]} \beta'$. Now, we distinguish cases depending on the polarity of a, a'. If $\mathrm{pol}_A(a) = +$ or $\mathrm{pol}_A(a') = -$, then by *courtesy* we have $ha \rightarrowtriangle_B ha'$, so that $\mathbf{e}[h] \rightarrowtriangle_B \mathbf{e}'[h]$. If $\mathrm{pol}_A(a) = -$ and $\mathrm{pol}_A(a') = +$, then

$$\mathbf{e} = t^-(\!|(\mathrm{s}, \mathrm{d})|\!) : \emptyset \xmapsto{a}_{\boldsymbol{\sigma}} \beta\,, \qquad\qquad \mathbf{e}' = t^+(\!|\alpha'|\!) : \alpha' \xmapsto{a'}_{\boldsymbol{\sigma}} \emptyset\,.$$

Assume, seeking a contradiction, that $\beta \cap \alpha' = \emptyset$, and consider the prefix of ρ:

$$\rho'(t^-(\!|(\mathrm{s}, \mathrm{d})|\!) \uplus \gamma)(t^+(\!|\alpha'|\!) \uplus \gamma') : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\sigma}} \nu\,,$$

but if indeed $\beta \cap \alpha' = \emptyset$, then $\mathbf{e}$ and $\mathbf{e}'$ permute as in

$$\rho'(t^+(\!|\alpha'|\!) \uplus \mu)(t^-(\!|(\mathrm{s}, \mathrm{d})|\!) \uplus \mu') : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\sigma}} \nu$$

and by *valid*, this entails $\mathrm{play}(\rho')a' \in \mathrm{Plays}(A)$, contradicting $a \rightarrowtriangle_A a'$. Assume now $\mathbf{e} \rightarrowtriangle_{\boldsymbol{\sigma}} \mathbf{e}'$. This comes either from $\mathrm{new}(\mathbf{e}) \cap \mathrm{pre}(\mathbf{e}') \neq \emptyset$, or $\mathrm{post}(\mathbf{e}) \cap \mathrm{eat}(\mathbf{e}') \neq \emptyset$. But the renaming of instantiated transtions does not change pre- and post-conditions, so $\mathbf{e}[h] \rightarrowtriangle_{\rho[h]} \mathbf{e}'[h]$ still.

Reciprocally, assume $\mathbf{e}[h] \leq_{\rho[h]} \mathbf{e}'[h]$. Seeking a contradiction, assume $\neg(\mathbf{e} \leq_\rho \mathbf{e}')$. By Lemma E.4, we can assume that $\mathbf{e}'$ appears before $\mathbf{e}$ in ρ. Hence, $\mathbf{e}'[h]$ appears before $\mathbf{e}[h]$ in $\rho[h]$. But by Lemma D.19 $\rho[h]$ is valid, so by Lemma E.4 $\mathbf{e}[h]$ must appear before $\mathbf{e}'[h]$, contradiction. □

Next, we need to show that valid runs are also *reflected* by renamings:

Lemma E.19. *Consider A, B games, $\boldsymbol{\sigma} : A$, $h = [f, (g_{\mathrm{m}})] : A \rightsquigarrow B$, and $\rho' : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\sigma}[h]} \alpha$ valid. Then, there is a unique $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\sigma}} \alpha$ valid such that $\rho' = \rho[h]$.*

Proof. By induction on ρ'. If it is empty, this is clear. Consider $\rho'\mathbf{f}^0$ with $\rho' : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\sigma}[h]} \alpha$ and $\mathbf{f}^0 = t^0(\!|\mu|\!) \uplus \gamma : \alpha \longrightarrow_{\boldsymbol{\sigma}[h]} \beta$, with $t^0(\!|\mu|\!) : \mu \longmapsto_{\boldsymbol{\sigma}[h]} \nu$. By IH, there is $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\sigma}} \alpha$. By definition, we still have $t^0(\!|\mu|\!) \uplus \gamma : \alpha \longrightarrow_{\boldsymbol{\sigma}} \beta$, so $\rho\mathbf{f}^0 : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\sigma}} \beta$ and as required, $(\rho\mathbf{f}^0)[h] = \rho'\mathbf{f}^0$.

Next, consider $\rho'\mathbf{f}^+$ with $\rho' : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\sigma}[h]} \alpha$ and $\mathbf{f}^+ = t^+(\!|\mu|\!) \uplus \gamma : \alpha \longrightarrow_{\boldsymbol{\sigma}[h]} \beta$, with $t^+(\!|\mu|\!) : \mu \overset{b'}{\longmapsto}_{\boldsymbol{\sigma}[h]} \emptyset$. Necessarily, $b' = (f(\mathsf{m}), \mathsf{d}', \mathsf{s}')$ for $\mathsf{m} = \partial_{\boldsymbol{\sigma}}(t^+)$, and $(\mathsf{d}', \mathsf{s}') = g_{\mathsf{m}}(\mathsf{d}, \mathsf{s})$ for $(\mathsf{d}, \mathsf{s}) = \delta\langle t^+\rangle(\mu)$ – so $b = h(b)$ for $b = (\mathsf{m}, \mathsf{d}, \mathsf{s})$. But then, by definition, $t^+(\!|\mu|\!) : \mu \overset{b}{\longmapsto}_{\boldsymbol{\sigma}} \emptyset$ as well, so $\mathbf{f}^+ : \alpha \longrightarrow_{\boldsymbol{\sigma}} \beta$. By IH, there is $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\sigma}} \alpha$ such that $\rho[h] = \rho'$. By *valid*, $\rho\mathbf{f}^+$ is still valid, and $(\rho\mathbf{f}^+)[h] = \rho'\mathbf{f}^+$.

Finally, consider $\rho'\mathbf{f}^-$ with $\rho' : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\sigma}[h]} \alpha$ and $\mathbf{f}^- = t^-(\!|(\mathsf{s}', \mathsf{d}')|\!) \uplus \gamma : \alpha \longrightarrow_{\boldsymbol{\sigma}[h]} \beta$, with $t^-(\!|(\mathsf{s}', \mathsf{d}')|\!) : \emptyset \overset{b'}{\longmapsto}_{\boldsymbol{\sigma}[h]} \nu$. Here, necessarily, $b' = (\mathsf{m}', \mathsf{s}', \mathsf{d}')$ where $\mathsf{m}' = f(\mathsf{m})$, $\mathsf{m} = \partial_{\boldsymbol{\sigma}}(t^-)$, $(\mathsf{s}', \mathsf{d}') = g_{\mathsf{m}}(\mathsf{s}, \mathsf{d})$. In other words, $b' = h(b)$ with $b = (\mathsf{m}, \mathsf{s}, \mathsf{d})$. Now, by IH we have $\rho : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\sigma}} \alpha$ with $\rho[h] = \rho'$. Besides, we have the transition $t^-(\!|(\mathsf{s}, \mathsf{d})|\!) : \emptyset \overset{b}{\longmapsto}_{\boldsymbol{\sigma}} \nu$, so that $(t^-(\!|(\mathsf{s}, \mathsf{d})|\!) \uplus \gamma) : \alpha \longrightarrow_{\boldsymbol{\sigma}} \beta$. Its validity follows immediately from *receptivity* of global renamings (and the fact that they are injective). It is clear that $(\rho(t^-(\!|(\mathsf{s}, \mathsf{d})|\!) \uplus \gamma))[h] = \rho'\mathbf{f}^-$ as required.

Finally, uniqueness is immediate by induction and injectivity of h. □

Lemma E.20. *Consider A, B games, $\boldsymbol{\sigma} : A$ a Petri strategy, and $h = [f, (g_{\mathsf{m}})] : A \rightsquigarrow B$. Then, $\mathscr{U}(\boldsymbol{\sigma}[h]) \cong \mathscr{U}(\boldsymbol{\sigma})[h]$.*

Proof. We construct an order-isomorphism

$$-[h] : \mathscr{E}^+(\boldsymbol{\sigma}) \cong \mathscr{E}^+(\boldsymbol{\sigma}[h]) \tag{6}$$

such that $\partial_{\boldsymbol{\sigma}[h]}(\mathsf{x}) = h(\partial_{\boldsymbol{\sigma}}(\mathsf{x}))$ for all $\mathsf{x} \in \mathscr{E}^+(\boldsymbol{\sigma})$. Given $\mathsf{x} \in \mathscr{E}^+(\boldsymbol{\sigma})$, there is some ρ a valid run in $\boldsymbol{\sigma}$ such that $\mathsf{x} = \mathscr{E}(\rho)$. By Lemma D.19, $\rho[h]$ is a valid run of $\boldsymbol{\sigma}[h]$, so we may consider $\mathsf{x}[h] = \mathscr{E}(\rho[h])$. By Lemma E.18, $-[h]$ on instantiated transitions is an order-isomorphism $-[h] : \mathsf{x} \cong \mathsf{x}[h]$, so $\mathsf{x}[h] \in \mathscr{E}^+(\boldsymbol{\sigma}[h])$. Together with Lemma E.19, this easily entails that we get $-[h] : \mathscr{E}(\boldsymbol{\sigma}) \cong \mathscr{E}(\boldsymbol{\sigma}[h])$ an order-isomorphism. By definition, for each $\mathsf{x} \in \mathscr{E}(\boldsymbol{\sigma})$ we have an order-iso $\mathsf{x} \cong \mathsf{x}[h]$ defined by applying $-[h]$ on each transition – thus, $-[h]$ preserves and reflects +-covered histories. The requirement *w.r.t.* labels is obvious by construction.

By Lemma E.9, we also obtain an order-isomorphism

$$-[h] : \mathscr{C}^+(\mathscr{U}(\boldsymbol{\sigma})) \cong \mathscr{C}^+(\mathscr{U}(\boldsymbol{\sigma}[h]))$$

such that $\partial_{\mathscr{U}(\boldsymbol{\sigma}[h])}(x[h]) = h(\partial_{\mathscr{U}(\boldsymbol{\sigma})}(x))$ for any $x \in \mathscr{C}^+(\mathscr{U}(\boldsymbol{\sigma}))$, so, from Definition B.1, an order-isomorphism $-[h] : \mathscr{C}^+(\mathscr{U}(\boldsymbol{\sigma})[h]) \cong \mathscr{C}^+(\mathscr{U}(\boldsymbol{\sigma}[h]))$ such that $\partial_{\mathscr{U}(\boldsymbol{\sigma}[h])}(x[h]) = \partial_{\mathscr{U}(\boldsymbol{\sigma})[h]}(x)$ for all $x \in \mathscr{C}^+(\mathscr{U}(\boldsymbol{\sigma})[h])$. By Lemma E.9, it follows that $\mathscr{U}(\boldsymbol{\sigma})[h]$ and $\mathscr{U}(\boldsymbol{\sigma}[h])$ are isomorphic. □

E.4.3 Currying. Follows from Lemma E.20, as currying is obtained with the same global renaming both in PStrat and in Strat.

E.4.4 Promotion. Let us start with characterizing +-covered traces of the functorial promotion.

Lemma E.21. *Consider $\boldsymbol{\sigma} : A \vdash B$ a Petri strategy. Then, we have an order-iso*

$$[-] : \mathsf{Fam}(\mathscr{E}^{+,\neq\emptyset}(\boldsymbol{\sigma})) \cong \mathscr{E}^+(!\boldsymbol{\sigma})$$

satisfying that for all $(\mathsf{x}^{\mathsf{e}})_{\mathsf{e}\in E} \in \mathsf{Fam}(\mathscr{E}^{+,\neq\emptyset}(\boldsymbol{\sigma}))$, we have

$$\partial_{!\boldsymbol{\sigma}}([(\mathsf{x}^{\mathsf{e}})_{\mathsf{e}\in E}]) = \left(\biguplus_{\mathsf{e}\in E} \mathsf{e} :: x_A^e\right) \vdash \left(\biguplus_{\mathsf{e}\in E} \mathsf{e} :: x_B^e\right)$$

writing $\partial_{\boldsymbol{\sigma}}(\mathsf{x}^e) = x_A^{\mathsf{e}} \vdash x_B^{\mathsf{e}}$ for all $\mathsf{e} \in E$.

Proof. This is a n-ary adaptation of the proof of Lemma E.16. Consider $(\mathsf{x}^{\mathsf{e}})_{\mathsf{e}\in E} \in \mathsf{Fam}(\mathscr{E}^{+,\neq\emptyset}(\boldsymbol{\sigma}))$. By definition, there is a valid run

$$\rho^{\mathsf{e}} : \emptyset \longrightarrow\!\!\!\!\rightarrow_{\boldsymbol{\sigma}} \alpha^{\mathsf{e}}$$

for all $\mathsf{e} \in E$ such that $\mathsf{x}^{\mathsf{e}} = \mathscr{E}_{\rho^{\mathsf{e}}}$. We define the history $[(\mathsf{x}^{\mathsf{e}})_{\mathsf{e}\in E}]$ as

$$\begin{array}{rcl}[\mathsf{x}^{\mathsf{e}} \mid \mathsf{e} \in E] & = & \{t^0(\!|\mathsf{e} :: \alpha|\!) \mid \mathsf{e} \in E, t^0(\!|\alpha|\!) \in \mathsf{x}^{\mathsf{e}}\} \\ & \uplus & \{t^+(\!|\mathsf{e} :: \alpha|\!) \mid \mathsf{e} \in E, t^+(\!|\alpha|\!) \in \mathsf{x}^{\mathsf{e}}\} \\ & \uplus & \{t^-(\!|(\mathsf{e} :: \mathsf{s}, \mathsf{d})|\!) \mid \mathsf{e} \in E, t^-(\!|(\mathsf{s}, \mathsf{d})|\!) \in \mathsf{x}^{\mathsf{e}}\}\,.\end{array}$$

We construct a run ρ obtained by concatenating all ρ^{e}s in the obvious way as in Lemma E.16. Exploiting Lemma D.27, it is a valid run and $[\mathsf{x}^{\mathsf{e}} \mid \mathsf{e} \in E] = \mathscr{E}(\rho)$ by construction. By definition of the causal ordering of instantiated transitions, it is also immediate that $[\mathsf{x}^{\mathsf{e}} \mid \mathsf{e} \in E]$ is +-covered; and that this preserves the labelling. Reciprocally, for any $\mathsf{x} \in \mathscr{E}(!\boldsymbol{\sigma})$, we consider the projections

$$\mathsf{x}^{\mathsf{e}} = \pi^!_{\mathsf{e}}(\mathsf{x})$$

and it follows from Lemma D.28 that $\mathsf{x}^{\mathsf{e}} \in \mathscr{E}(\boldsymbol{\sigma})$ for all $\mathsf{e} \in E$. From the definition of the causal ordering of instantiated transitions, each x^{e} is still +-covered.

Finally, these two transformations are inverses as required. □

Corollary E.22. *Consider $\boldsymbol{\sigma} : !A \vdash B$ a Petri strategy. Then, we have $\mathscr{U}(\boldsymbol{\sigma}^\dagger) \cong \mathscr{U}(\boldsymbol{\sigma})^\dagger$.*

Proof. We exploit the following sequence of label-preserving order-isomorphisms:

$$\begin{array}{rcl}\mathscr{C}^+(\mathscr{U}(\boldsymbol{\sigma}^\dagger)) & \cong & \mathscr{E}^+(\boldsymbol{\sigma}^\dagger) \\ & = & \mathscr{E}^+((!\boldsymbol{\sigma})[\mathsf{dig} \vdash \mathsf{id}]) \\ & \cong & \mathscr{E}^+(!\boldsymbol{\sigma}) \\ & \cong & \mathsf{Fam}(\mathscr{E}^{+,\neq\emptyset}(\boldsymbol{\sigma})) \\ & \cong & \mathsf{Fam}(\mathscr{C}^{+,\neq\emptyset}(\mathscr{U}(\boldsymbol{\sigma})))\end{array}$$

using first Lemma E.9, then via a direct verification as in Proposition D.32, then applying (6), followed by Lemma E.21, and then Lemma E.9 – with the obvious verification that it specializes to an iso between non-empty configurations and histories. It is a direct verification that this sequence of isomorphisms preserves display maps.

Consequently, it follows that $\mathscr{U}(\boldsymbol{\sigma}^\dagger) \cong \mathscr{U}(\boldsymbol{\sigma})^\dagger$ from Proposition C.16. □

E.5 The Unfolding Preserves Primitives

E.5.1 Variable and Evaluation. Follows from Lemma E.20.

E.5.2 Queries, Conditional, Constants. It is a simple calculation to compute the unfolding for these linear Petri strategies and check we obtain the desired finite strategy.

E.5.3 Fixpoint. We prove the following proposition:

Proposition E.23. *For any well-opened arena O, $\mathscr{U}(\mathsf{Y}_O) \cong \mathsf{Y}_O$.*

Proof. Using Lemmas C.5 and E.9, the required isomorphism boils down to an order-iso

$$\mathscr{E}^+(\mathsf{Y}_O) \cong \mathscr{C}^+(\mathsf{Y}_O)$$

preserving display maps. We build it using Lemmas D.35 and C.22. It is clearly injective so we simply have to show it is surjective. Consider a +-covered configuration x of Y_O represented as a tuple $\langle J, z, (y_{\mathsf{s}})_{\mathsf{s}\in J}\rangle$. We can construct a set of events realising x as follows:

- For each $(\mathsf{m}, \mathsf{s}, d)^- \in z$ we include the event $\nu_{\vdash}\mathsf{m}(\!|(\mathsf{s}, \mathsf{d})|\!)$, and $\ell_{\vdash}\nu_{\multimap}(\!|\{(\ell_!\blacklozenge :: \mathsf{s}, \mathsf{d})^{@\mathsf{m}^-}\}|\!)$
- For each $(\mathsf{m}, \mathsf{s}, \mathsf{d})^+ \in z$, we include the events $\ell_{\vdash}\nu_{\multimap}(\!|(\ell_!\blacklozenge :: \mathsf{s}, \mathsf{d})|\!)$ and $\nu_{\vdash}\mathsf{m}(\!|\{(\ell_!\blacklozenge :: \mathsf{s}, \mathsf{d})^{@\mathsf{m}^+}\}|\!)$
- For each $(\mathsf{m}, \mathsf{s}_0, \mathsf{d})^- \in y_{\mathsf{e}\cdot\mathsf{s}}$, we include the events $\nu_{\vdash}\nu_{\multimap}\mathsf{m}(\!|(\mathsf{e} :: (\mathsf{s}) :: \mathsf{s}_0, \mathsf{d})|\!)$ and $\nu_{\vdash}\nu_{\multimap}\mathsf{m}(\!|\{((\mathsf{e}\cdot\mathsf{s}) :: \mathsf{s}_0, \mathsf{d})^{@\mathsf{m}^-}\}|\!)$.

- For each $(\mathsf{m}, \mathsf{s}_0, \mathsf{d})^+ \in y_{\mathsf{e}\cdot\mathsf{s}}$, we include the events $\not{r}_{\vdash}\ell_{\multimap}\mathsf{m}(\!|((\mathsf{e}\cdot\mathsf{s}) :: \mathsf{s}_0, \mathsf{d})|\!)$ and $\not{r}_{\vdash}\ell_{\multimap}\mathsf{m}(\!|\{((\mathsf{e}\cdot\mathsf{s}) :: \mathsf{s}_0, \mathsf{d})^{@\mathsf{m}^+}\}|\!)$,

and it is easy to see that this set of transitions is reachable by a valid run of $\mathbf{Y}_O$. □

E.5.4 Contraction. The reasoning follows a similar and simpler route as for the fixpoint operator.

E.5.5 Let bindings. We first characterise the configurations of the strategy interpreting lets.

Lemma E.24. *The +-covered configurations of* **let** *are order-isomorphic to tuples* $\langle I \subseteq \mathcal{E}, x \in \mathscr{C}(\mathbf{X}), y, y' \in \mathscr{C}(\mathbf{Y})\rangle$ *such that:*

(1) $y \neq \emptyset$ *iff* $x \neq \emptyset$
(2) x *is maximal iff* $y' \neq \emptyset$
(3) *if* $y' \neq \emptyset$, *then* $y = y'$.
(4) *if* $y' = \emptyset$, *then* $I = \emptyset$.

The isomorphism sends such tuples to $((\uplus_{\mathsf{e}\in I}(\mathsf{e} :: x)) \multimap y') \otimes x \vdash y$.

Lemma E.25. *We have* $\mathscr{U}(\mathbf{let}) \cong \mathbf{let}$.

Proof. As before we rely on Lemmas C.5 and E.9 to build the isomorphism. Lemma D.37 together with Lemma E.24 induce an injective map from $\mathscr{E}^+(\mathbf{let})$ into $\mathscr{C}^+(\mathbf{let})$. We show it is surjective by constructing a set of eventse of **let** from $x \in \mathscr{C}^+(\mathbf{let})$ corresponding to a $\langle I, x, y, y'\rangle$.

- If $y \neq \emptyset$, then we have events $\not{r}_{\vdash}\mathsf{Q}^-(\!|([], \bullet)|\!)$ and $\ell_{\vdash}\not{r}_{\otimes}\mathsf{Q}(\!|\{([], \bullet)^{@1}|\!)$
- If x has an event $([], \mathsf{d})$ maximal in $\mathbf{X}$, then we have the two events $\ell_{\vdash}\not{r}_{\otimes}\mathsf{A}^-(\!|([], \mathsf{d})|\!)$, and $\ell_{\vdash}\ell_{\otimes}\not{r}_{\multimap}\mathsf{Q}^+(\!|\{([], \bullet)^{@2}\}|\!)$
- For every $\mathsf{e} \in I$, we have three events written $\ell_{\vdash}\ell_{\otimes}\ell_{\multimap}\mathsf{Q}^-(\!|([\mathsf{e}], \bullet)|\!)$, $\mathsf{s}(\!|\{([], \bullet)^{@4}, ([], \mathsf{d})^{@3}\}|\!)$, and $\ell_{\vdash}\ell_{\otimes}\ell_{\multimap}\mathsf{A}^+(\!|\{([\mathsf{e}], \mathsf{d})^{@5}\}|\!)$ where d is the value of the maximal event in x.
- If y' has a maximal $([], \mathsf{d})$, we have events $\ell_{\vdash}\ell_{\otimes}\not{r}_{\multimap}\mathsf{A}^-(\!|([], \mathsf{d})|\!)$ and $\not{r}_{\vdash}\mathsf{A}^+(\!|\{([], \mathsf{d}]^{@6}\}|\!)$. □

E.5.6 Newref and newsem. We now show that the unfolding of the net for newref is indeed the strategy newref. Our first step is to show that the consistent memory traces described in Section D.3.6 correspond to +-covered configurations of **newref**:

Lemma E.26. *There is an order-isomorphism between* $\mathscr{C}^+(\mathsf{precell})$ *and the set of consistent memory traces ordered by prefix.*

Proof. Direct from the definition of precell. □

We now show the main result:

Proposition E.27. $\mathscr{U}(\mathbf{newref}) \cong \mathsf{newref}$.

Proof. By Lemmas C.5 and E.9, this amounts to building an order-iso:

$$\mathscr{E}^+(\mathbf{newref}) \cong \mathscr{C}^+(\mathsf{newref}).$$

From left-to-right. We focus on non-empty histories and configurations and use characterisation of $\mathscr{C}^+(\mathsf{newref})$ from Proposition C.26.

Consider a non-empty history $\mathsf{y} \in \mathscr{E}^+(\mathbf{newref})$. It is reached by a run $\rho : \emptyset \xrightarrow{s}\!\!\!\!\rightarrow \alpha$. By Lemma D.39, we know that there $\mathsf{Tr}(\rho)$ is a consistent memory trace, and that $|s|$ must have the shape $((\uplus_{\mathsf{e}\in I}(\mathsf{e} :: x_\mathsf{e})) \multimap w) \vdash z$. Since y is +-covered, we observe that $w = z$. We can thus map y to $\langle \mathsf{Tr}(\rho), w\rangle \in \mathscr{C}^+(\mathsf{newref})$ using the isomorphism of Proposition C.26. Note that by the side conditions of Lemma D.39, the family $(x_\mathsf{e})_{\mathsf{e}\in I}$ is entirely determined by $\mathsf{Tr}(\rho)$: I matches the length of $\mathsf{Tr}(\rho)$ and each x_e is a two-event configuration corresponding to the memory operation $\mathsf{Tr}(\rho)_\mathsf{e}$.

The last check it to show that this does not depend on the particular run ρ chosen. Clearly x only depends on $|s|$. For $\mathsf{Tr}(\rho)$, we observe that it is actually directly recoverable from the set of transitions y. First, we define the set of memory operations O_{y} to contain $(w, \mathsf{e}, \mathsf{d})$ if $\mathsf{w}\{([\mathsf{e}], \mathsf{d})^{@3}, _\} \in \mathsf{y}$ and $(r, \mathsf{e}, \mathsf{d})$ if $\mathsf{r}\{([\mathsf{e}], _)^{@5}, (_, \mathsf{d})^{@2}\} \in \mathsf{y}$. Then, the causal order on y induces a linear order on O_{y} due to the threading of exponential signatures. The resulting trace is exactly $\mathsf{Tr}(\rho)$.

From right-to-left. Consider now a $\langle \rho, x \rangle \in \mathscr{C}^{+,\neq\emptyset}(\mathsf{newref})$ where ρ is a consistent memory trace. We can build a history y containing the following instantiated transitions:

- The initial negative event on $\mathscr{r}_{\vdash}\mathsf{Q}(\!|([], \bullet)|\!)$.
- The positive event $\ell_{\vdash}\ell_{\multimap}\mathsf{Q}(\!|\{([], \bullet)^{@1}\}|\!)$.
- If x contains a move $(\mathsf{A}, [], d)$, then the events $\ell_{\vdash}\mathscr{r}_{\multimap}\mathsf{A}^{-}(\!|([], \mathsf{d})|\!)$ and $\mathscr{r}_{\vdash}\mathsf{A}^{+}(\!|\{([], \mathsf{d})^{@7}\}|\!)$.
- If $\rho_i = (w, \mathsf{e}, \mathsf{d})$ then the events $\ell_{\vdash}\ell_{\multimap}\mathscr{w}_{\mathbb{V}}\mathsf{Q}^{-}(\!|([\mathsf{e}], \mathsf{d})|\!)$, $\mathsf{w}(\!|\{([\mathsf{e}], \mathsf{d})^{@3}, \{[\mathsf{e}'], \mathsf{d}'\}^{@2}\}|\!)$ where: e' is the exponential token of ρ_{i-1} (or $[]$ if $i = 0$), and d' the value observed by ρ_{i-1} (or zero if $i = 0$); and $\ell_{\vdash}\ell_{\multimap}\mathscr{w}_{\mathbb{V}}\mathsf{A}^{+}(\!|\{([\mathsf{e}], \checkmark)^{@4}\}|\!)$.
- And similarly if $\rho_i = (r, \mathsf{e}, \mathsf{d})$.

From this description, it is easy to build a valid run reaching y establishing that $\mathsf{y} \in \mathscr{E}(\mathbf{newref})$. An easy verification shows that all maximal events in y are positive. □

Received 2022-07-07; accepted 2022-11-07