We introduce the concurrent games abstract machine: a multi-token machine for Idealized Parallel Algol (IPA), a higher-order concurrent programming language with shared state and semaphores. Our abstract machine takes the shape of a compositional interpretation of terms as Petri structures, certain coloured Petri nets. For the purely functional fragment, 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 pair here these ideas with folklore ideas on the representation of first-order imperative concurrent programs as coloured Petri nets.

To prove our machine correct, we follow game semantics and represent types as certain games specifying dependencies and conflict between computational events. We define Petri strategies as those Petri structures obeying the rules of the game. In turn, we show how Petri strategies unfold to concurrent strategies in the sense of concurrent games on event structures. This not only entails correctness and adequacy of our machine, but also lets us generate operationally a causal description of the behaviour of programs at higher-order types.

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

1 INTRODUCTION

Interactive semantics, and in particular Game Semantics [Abramsky et al. 2000; Hyland and Ong 2000] or Geometry of Interaction (GoI) [Danos et al. 1996; Girard 1989], aim at describing formally the execution of programs as an interactive process. They are particularly relevant in the presence of programming features that impact the geometry of the control flow (e.g. concurrency, higher-order, exceptions, etc.), which pose a significant obstacle to modular reasoning. For example, Geometry of Interaction has long been proposed as a basis for the compilation of functional programs [Ghica 2007; Mackie 1995], while structures from game semantics are behind recent work on compositional certified compilation [Koenig and Shao 2020; Stewart et al. 2015; Xia et al. 2020].

GoI can be presented as an abstract machine: these GoI token machines seem to originate in [Danos and Regnier 1996] (the Interaction Abstract Machine (IAM), for the $\lambda$-calculus) and [Mackie 1995] (for PCF). Token machines usually present a program as a graph (a proof net) and its execution as the walk of a token (standing for the control flow) through the graph, as determined by local rules. Token machines were extended in multiple ways: to multi-token machines, for Linear Logic [Laurent 2001] (without synchronization) or for functional programs [Dal Lago et al. 2014, 2015] (including in call-by-value) or interaction nets [Dal Lago et al. 2014]. They were redeveloped in a coalgebraic setting supporting a range of algebraic effects (e.g. nondeterminism, probability, exception, global states, 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]. Despite this remarkable breadth, many combinations of effects ubiquitous in realistic programming languages are missing, including shared memory concurrency (a notable exception is Ghica’s geometry of synthesis [Ghica 2007], however for a language with a type discipline ensuring that state causes no race). This is
not due to an inherent restriction of GoI; but the usual correctness arguments (typically via cut elimination or realizability) may not extend well. This leaves open:

**How to build a GoI token machine for a concurrent higher-order language with shared memory?**  (1)

In contrast, game semantics, after the initial models of PCF [Abramsky et al. 2000; Hyland and Ong 2000], offered a wealth of fully abstract models for a wide range of programming features among which local state [Abramsky and McCusker 1996], control operators [Laird 1997], exceptions [Laird 2001], higher-order state [Abramsky et al. 1998] and many others, including in particular higher-order shared memory concurrency [Ghica and Murawski 2008] – up to realistic languages [Murawski and Tzevelekos 2021]. It seems fair to say that game semantics has historically been more versatile than GoI. On the other hand, game semantics lack the direct operational flavour of GoI. Though game semantics is effective and can be computed and manipulated from the source code, the connection between a program and its semantics is quite remote and obfuscated by the many-layered definition of the interpretation of a program into a denotational model; this is an obstacle to it being used as formal basis in software tools, for validating optimizations or in certified compilation. This led researchers to investigate more operational, direct descriptions of the game semantics [Ghica and Tzevelekos 2012; Jaber 2015; Levy and Staton 2014] (though really, connections between game and operational semantics date back to [Danos et al. 1996]), in essence generating strategies by purely operational means. But such connections lack in generality; in particular they only exist for a few sequential languages. So a second motivating open question is:

**How best to link game semantics and operational semantics?**  (2)

In particular, can we construct such an operational–denotational connection for a language with higher-order shared memory concurrency, say Idealized Parallel Algol (IPA)? Can we relate not merely to the interleaving model of [Ghica and Murawski 2008] but to the interpretation in the truly concurrent framework of concurrent games on event structures [Castellan and Clairambault 2020; Castellan et al. 2017, 2019], yielding an operational handle on causal reasoning?

**Contributions.** In this paper, we attack birds (1) and (2) with one stone. We give a multi-token abstract machine, called the *concurrent games abstract machine*, for IPA. More precisely, we give an interpretation of IPA transforming a program $M$ into a coloured Petri net regarded as the concurrent games abstract machine loaded with $M$. For PCF, the obtained Petri net is similar to the usual token machine [Mackie 1995]; but our machine goes way beyond PCF.

Our correctness proof departs from usual GoI methods. We show that the multi-token machine for $M$ unfolds to the event structure serving as interpretation of $M$ in the truly concurrent games interpretation of [Castellan and Clairambault 2020; Castellan et al. 2019]; in the sense of the well-known unfolding of Petri nets to event structures [Hayman and Winskel 2008b; Nielsen et al. 1981]. This is proved compositionally, by defining an unfolding that preserves all operations used in the interpretation. Besides answering (2) for IPA, this also provides an answer to (1) as the correctness of our token machine then follows from adequacy of the concurrent games model of IPA.

In fact, we regard this as bringing a contribution to a third question:

**How can we best state correctness of GoI on higher-order types?**  (3)

Indeed, for many token machines, correctness is stated for programs of ground type only, which is not ideal if GoI is to be thought of as a framework for compositional reasoning on programs. Another approach is to link GoI with game semantics: Baillot proved [Baillot 1999] that GoI generates the corresponding AJM-style strategy for IMELL. Ghica and Smith also prove the correctness of *Geometry of Synthesis* by linking it to (interleaving) game semantics [Ghica and Smith 2010]. Our result generalizes both, showing that our abstract machine generates the same causal structure as
Where types generated by \( \Gamma \) are called \textbf{well-opened}. We have \( \mathbb{U} \) a \textit{unit} type, \( \mathbb{B} \) and \( \mathbb{N} \) respectively types for \textit{booleans} and \textit{natural numbers}, a type \( \mathbb{V} \) for \textit{integer references}, and \( \mathbb{S} \) for \textit{semaphores}. The split into \textit{standard types} and \textit{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 \textbf{ground types}, and use \( \mathbb{X}, \mathbb{Y}, \mathbb{Z} \) to range over those.

---

**Fig. 1. Typing rules for IPA**

---

2 \ IPA AND ITS COMPOSITIONAL INTERPRETATIONS

2.1 The language IPA

IPA is a higher-order call-by-name concurrent language with shared memory and semaphores, serving as 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 variable and semaphore types should not appear at the right hand side of an arrow. This simplifies the exposition without significantly harming expressiveness.

2.1.1 Types and terms. We first describe the types of IPA, generated by

\[
A, B, C ::= O \mid \mathbb{V} \mid \mathbb{S} \quad O ::= \mathbb{U} \mid \mathbb{B} \mid \mathbb{N} \mid A \rightarrow O
\]

where types generated by \( O \) are called \textbf{well-opened}. We have \( \mathbb{U} \) a \textit{unit} type, \( \mathbb{B} \) and \( \mathbb{N} \) respectively types for \textit{booleans} and \textit{natural numbers}, a type \( \mathbb{V} \) for \textit{integer references}, and \( \mathbb{S} \) for \textit{semaphores}. The split into \textit{standard types} and \textit{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 \textbf{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 $\text{Var}$. **Typing judgments** have the form $\Gamma \vdash M : A$, with $\Gamma$ a context and $A$ a type.

The construction $f(M, N)$ applies to any computable partial function $f : \mathbb{X} \times \mathbb{Y} \to \mathbb{Z}$ (abusing notations to treat the ground types $\mathbb{X}, \mathbb{Y}, \mathbb{Z}$ as the underlying sets), and its two operands are intended to be evaluated in parallel. This covers many usual primitives: for instance, if $\Gamma \vdash M : \mathbb{U}$, we define $M \parallel N = \parallel((M, N))$, with $\parallel : \mathbb{U} \times \mathbb{U} \to \mathbb{U}$ the trivial function – in fact, we shall use implicitly $\parallel : \mathbb{U} \times \mathbb{X} \to \mathbb{X}$, so that parallel composition propagates a value. The usual predecessor, successor, zero test primitives of PCF can be obtained similarly as particular cases of this operation. Conditionals eliminate only to ground type, but as usual in call-by-name, a more general conditional can be obtained as syntactic sugar. 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 $\mathbb{U}$.

### 2.1.2 Examples

We include a few examples of simple IPA programs. First:

```
\[ \vdash \text{coin} = \text{newref } r \in (r := 1 \parallel \text{iszero } r) : \mathbb{B} \]
```

is a non-deterministic boolean – by convention $\text{newref}$ automatically initializes $r$ to 0, so that coin directly sets up a race. Though we have no primitive for sequential composition, for $\Gamma \vdash M : \mathbb{X}$ and $\Gamma \vdash N : \mathbb{Y}$, we define $M; N$ as let $x = M$ in $N$ : $\mathbb{Y}$. Another interesting example is

```
x : \mathbb{U}, y : \mathbb{X} \vdash \text{newsem } s \in \text{grab } s; (x; \text{release } s \parallel \text{grab } s; y) : \mathbb{X}
```

which behaves like sequential composition, using a semaphore for synchronization.

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

```
\[ \vdash (\ldots F. (\lambda g : \mathbb{N} \to \mathbb{N} . g \, 1) ;F (\lambda g : \mathbb{N} \to \mathbb{N} \to \mathbb{N} . n)) (\lambda f : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N} \to \mathbb{N}. \text{newref } r \in f (\lambda x. r := x; 0) ! r) : \mathbb{N} \]
```

returns 0, because execution causes the initialization of two independent references. An unbounded number of references can arise in this way if this happens within recursion.

We hope these show that though IPA is a toy language, it is a semantically highly non-trivial one which poses realistic challenges. We chose IPA because of its historical importance in the game semantics community, and because there already is a detailed concurrent games model ready for this language [Castellan and Clairambault 2020]. But unlike in [Ghica and Murawski 2008], we made sure to include a let construct to show that the concurrent games abstract machine does handle an explicit control of the evaluation order more typical of call-by-value languages.

### 2.1.3 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 $L$ of **memory locations**. A **store** is a partial map $s : L \to \mathbb{N}$ with finite domain where $\mathbb{N}$ stands, overloading notations, for the set of natural numbers. **Configurations** of the operational semantics are tuples $(M, s)$ where $s$ is a store with $\text{dom}(s) = \{ \ell_1, \ldots, \ell_n \}$ and $\Sigma \vdash M : A$ with $\Sigma = \ell_1 : \mathbb{V}, \ldots, \ell_i : \mathbb{V}, \ell_{i+1} : \mathbb{S}, \ldots, \ell_t : \mathbb{S}$. Reduction rules have the form

```
(\langle M, s \rangle) \red (\langle M', s' \rangle)
```

where $\text{dom}(s) = \text{dom}(s')$; we write $\red^{*}$ for the reflexive transitive closure. If $\vdash M : \mathbb{X}$, we write $M \parallel \vdash M \parallel (M, \emptyset) \red^{*} (v, \emptyset)$ for some value $v$. Then we say that $M$ **converges**, else it **diverges**.

The detailed reduction rules are essentially as in [Ghica and Murawski 2008]. We postpone them to Appendix A, as they will be referred to only indirectly in this paper (via Theorem 4.12).
2.2 IPA-structures

We describe an abstract structure for the interpretation of IPA. The goal is not a categorical semantics with a general soundness theorem, but merely to structure the compositional interpretation. An IPA-structure is a category with just enough structure so that the interpretation may be defined following the standard lines of the semantics of functional languages into an adequately equipped model of Intuitionistic Linear Logic (say, a Seely category [Melliès 2009]). We call a precategory a structure with the data of a category, an associative composition but no identity laws.

Definition 2.1. An IPA-structure is a precategory $C$, with a set $C_*$ of well-opened objects – we use $A, B, C$ to range over the set $C_0$ of objects and $O$ to range over $C_*$ – and equipped with:

- **Constructions.** We have $U, B, N \in C_*$, $V, S \in C_0$, and constructions:
  - tensor: for any $A, B \in C_0$, there is $A \otimes B \in C_0$,
  - product: for any finite family $(A_x)_{x \in V}$ with $V \subseteq \text{Var}$, there is $\&_{x \in V} A_x \in C_0$,
  - linear arrow: for any $A \in C_0$ and $O \in C_*$, there is $A \rightarrow O \in C_*$,
  - exponential: for any $A \in C_0$, there is $!A \in C_0$,

  where we write $&\emptyset = \top$ for the product of the empty family.

- **Operations.** We have the following three constructions on morphisms:
  - tensor: $\otimes : C(A_1, B_1) \times C(A_2, B_2) \rightarrow C(A_1 \otimes A_2, B_1 \otimes B_2)$
  - currying: $\Lambda_{x:A,O}^\Gamma : C(!([\Gamma, x : A, \Delta]), O) \rightarrow C(!([\Gamma, \Delta]), !A \rightarrow O)$
  - promotion: $(\cdot) ! : C(![\Gamma], O) \rightarrow C(![\Gamma, !O])$

  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 $(A_x)_{x \in \{x_1, \ldots, x_n\}}$;

- **Primitives.** We have the basic morphisms listed in Figure 3, where $\otimes^0 A = 1$, $\otimes^1 A = A$, $\otimes^{n+2} A = A \otimes (\otimes^{n+1} A)$, and writing $w_A \in C(![A, 1])$ for $c_A^0$ and $c_A \in C(![A, !A \otimes !A])$ for $c_A^2$.

We shall soon define an interpretation of IPA in any IPA-structure – again, requiring no equation. Besides structuring the interpretation, it also provides a clean way to relate interpretations:

Definition 2.2. Consider $C$ and $D$ two IPA-structures, and $F : C \rightarrow D$ a functor.

Then, $F$ is a strict IPA-functor iff it preserves all structure on the nose.

2.3 Interpretation of IPA

The exact definition of the interpretation follows the standard lines of the interpretation of call-by-name languages into models of intuitionistic linear logic. Fix $C$ an IPA-structure.

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

To any typed term $\Gamma \vdash M : A$ we associate as interpretation a morphism $[\Gamma \vdash M : A] \in C(![\Gamma, [A]])$ sometimes shortened to $[M]$, following the clauses of Figure 2. Finally, we have:

Lemma 2.3. Consider $C, D$ two IPA-structures, and $F : C \rightarrow D$ an IPA-functor. Then,

1. for all type $A$ and context $\Gamma$, $F([A]_C) = [A]_D$ and $F([\Gamma]_C) = [\Gamma]_D$,
2. for all term $\Gamma \vdash M : A$, we have $F([M]_C) = [M]_D$.

This follows by induction on the definition of the interpretation, applying at each step the preservation property of $F$ – so strict IPA-functors list the proof obligations to relate two interpretations.

3 PETRI STRUCTURES FOR IPA

We start by describing the first and central IPA-structure of this paper, that of Petri structures.
3.1 Definition and Examples

Consider fixed sets $\text{Tok}$ and $\mathcal{M}$, respectively called tokens (which will serve as colours), and addresses (which will label certain transitions). Each $m \in \mathcal{M}$ has a polarity $\text{pol}(m) \in \{-, +\}$, specifying whether it is a label for actions by the program ($+$) or the environment ($-$). We shall not give just yet the definition of these data, which will not be necessary until later on.

We use $\cup$ to denote the usual set-theoretic union, when it is known to be disjoint.

**Definition 3.1.** Consider $M \subseteq \mathcal{M}$ a finite subset of addresses.

A Petri structure on $M$ is $\sigma = (\mathcal{L}, \mathcal{T} = \mathcal{T}^+ \cup \mathcal{T}^0 \cup \mathcal{T}^-, \vartheta, \text{pre}, \text{post}, \delta)$ where:

- $\mathcal{L}$ is a finite set of locations,
- $\mathcal{T}$ is a finite set of transitions sorted by polarity $+\mathrm{ or }-\mathrm{,}$
- $\vartheta$ is a labelling function $\mathcal{T}^+ \sqcup \mathcal{T}^- \rightarrow M$ such that $t \in \mathcal{T}^{\text{pol}(\vartheta(t))}$ for all $t \in \mathcal{T}^+ \sqcup \mathcal{T}^-$,
- $\text{pre}$ is a function $\mathcal{T} \rightarrow \mathcal{P}(\mathcal{L})$ of pre-conditions,
- $\text{post}$ is a function $\mathcal{T} \rightarrow \mathcal{P}(\mathcal{L})$ of post-conditions,

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

$$
\delta(t^0) : \text{cond}(\text{pre}(t)) \rightarrow \text{cond}(\text{post}(t)),
\delta(t^-) : \text{Tok} \rightarrow \text{cond}(\text{post}(t)),
\delta(t^+) : \text{cond}(\text{pre}(t)) \rightarrow \text{Tok}.
$$

where $\text{cond}(L) = L^L$ is the set of conditions with support $L$ – we write cond for all conditions.

In the terminology of coloured Petri nets, the set of colours is $\text{Tok}$, and is independent of the location. Locations are internal buffers that may contain one or several tokens. Neutral transitions correspond to internal computation: firing $t^0$ takes one token from each location of $\text{pre}(t)$, puts one token in each location of $\text{post}(t)$, acting on colours as prescribed by $\delta(t)$. Negative and Positive transitions, also called visible transitions, interact with the outside world. We shall see later on the rules that govern this interaction. For now we focus on closed Petri structures, defined as Petri structures on the set $M = \{\text{Q}^-, A^+\}$ – here $\text{Q}^-$ initiates computation, while $A^+$ terminates it.

For now we shall content ourselves with the informal description of the token game given above, and build up intuition by considering examples of closed Petri structures.
3.1.1 Closed Petri structures. We temporarily fix $\text{Tok} = \{\bullet\} \cup \mathbb{N} – \bullet$ is a token with no value, while natural numbers stand for the corresponding value (we shall later settle on a more expressive Tok).

We draw a Petri structure $\sigma$ following standard conventions from Petri nets: locations are circles, while transitions are boxes. The graph drawn carries the information of $\mathcal{L}, \mathcal{T}$, pre and post, while $\delta$ is given separately as a transition table. Whenever unambiguous, we use as name of visible transitions their label via $\delta$. If $L = \{l_1, \ldots, l_n\} \subseteq \mathcal{L}$, a condition $(t_{l_1} t_{l_2} \in \text{cond}(L))$ is written $\{t_{l_1}^{\oplus_1}, \ldots, t_{l_n}^{\oplus_n}\}$ where each $l_i \in L$ appears exactly once. An individual $t_i^{\oplus_1}$, where $t \in \text{Tok}$ and $l \in \mathcal{L}$, is called a token-in-location, or tokil for short – we write TokIL$(\sigma)$ for the set of tokils.

We start with the closed Petri structure for the constant 1, in Figure 4. Upon being triggered by $Q^-$, the net immediately prepares value 1 in location 1 by $\delta(Q^-)(\bullet) = \{1^{\oplus_1}\}$. This enables transition $A^+$, which outputs the value via $\delta(A^+)(\{n^{\oplus_1}\}) = n$. Figure 5 presents a parallel evaluation of 1 + 1. When triggered the net throws two tokils $\bullet^{\oplus_1}$ and $\bullet^{\oplus_2}$ corresponding to evaluation requests for the two constants. Both tokils are forwarded to an independent copy of the structure of Figure 4. Upon receiving the two values in locations 5 and 6, the last transition fires and outputs the sum 1 + 1. The example – or its closed variant as obtained by the interpretation – may be ran in the implementation here. It illustrates how Petri structures handle basic calling and returning mechanisms, and displays parallel computation. Those simple examples are not particularly original, and follow the usual folklore lines along which coloured Petri nets may be used to represent programs.

Another well-known idea is that Petri nets can represent shared state: Figure 6 shows the closed Petri structure for $\rhd$ coin : $\mathbb{B}$ as defined in Section 2.1.2. Upon initialization, the net throws three tokens: the tokil $0^{\oplus_2}$ initializes the variable to value 0; the tokil $1^{\oplus_1}$ is a write request for the value 1; and $\bullet^{\oplus_3}$ is a read request. There is a race between the read and write requests: if $r$ wins, the value in location 5 ends up being 0, while if $w$ then $r$ reads value 1 instead. The final transition $A^+$ waits for the write acknowledgment and the result of the read to return the value read. The example – or its closed variant as obtained by the interpretation – may be ran in the implementation here.

3.1.2 Open Petri structures. Petri structures also handle open or higher-order programs.

For open programs interacting with their execution environment, Petri structures will have a wider range of labels for visible transitions, reflecting the possible avenues of interaction. For instance, Petri structures corresponding to programs typed with $g : \mathbb{U} \rightarrow \mathbb{U} , x : \mathbb{U} \rhd \mathbb{U}$ will use

$$\mathcal{M} = \{\ell, \epsilon_x^g \ell, \ell_r Q^- , \ell, \epsilon_x^g \ell, A^+ , \ell, \epsilon_x^g \ell, r \ell Q^+, \ell, \epsilon_x^g \ell, r \ell A^- , \ell, \epsilon_x^g \ell, Q^+, \ell, \epsilon_x^g \ell, A^- , \ell, \epsilon_\ell Q^- , \ell, \epsilon_\ell A^+ \}$$

where the injections $\ell$ and $r$ indicate the two sides of $\rhd$; $\epsilon_x^g$ and $\epsilon_x^g$ point to a variable name; $\ell_r$ and $\ell_r$ point to either side of an arrow $\rhd$; and $Q$ and $A$ stand for “Question” (call) or “Answer” (return). Without $Q$ and $A$, each of the addresses above corresponds to an occurrence of a base type in $g : \mathbb{U} \rightarrow \mathbb{U} , x : \mathbb{U} \rhd \mathbb{U}$, and then each base type occurrence admits a call and a return.

More generally, we shall use in this paper the following addresses:
Definition 3.2. We first define the set $M$ of addresses as

$$M := \ell \cdot M \mid r_a \cdot M \mid \ell \cdot M \mid r_a \cdot M \mid r_a \cdot M \mid \ell \cdot M \mid w \cdot M \mid r_a \cdot M \mid q \cdot M \mid r_a \cdot M \mid Q \cdot A,$$

for $x \in \text{Var}$ – we use $m$ to range over addresses. These have a polarity defined as $\text{pol}(Q) = -$, $\text{pol}(A) = +$, $\text{pol}(\ell \cdot m) = -\text{pol}(m)$, $\text{pol}(r_a \cdot m) = -\text{pol}(m)$, and preserved in all other cases.

In this way, we show in Figure 7 (a close variant of) the open Petri structure interpreting $g : U \rightarrow U, x : U \mapsto g \cdot x : U$. Tokens are either $\bullet$ for a data request, or $\checkmark$ for the unique possible value on $U$. Upon initialization with $r_a Q^-$, the net interrogates the return value of $g$. If $g$ calls its argument with $\ell \cdot c_a^3 \cdot \ell \cdot Q^-$, the net interrogates the return value of $x$. If $x$ returns a value with $\ell \cdot c_a^3 A$, this value is propagated to the argument of $g$. Finally, if $g$ returns with $\ell \cdot c_a^3 A$, the value is forwarded to the right hand side. Readers familiar with proof nets will recognize the axiom links, readers familiar with game semantics will recognize pairs of Opponent moves and induced Player responses.

The implementation has no preset choice for this example, though one may obtain it by manually typing the program “$g \cdot x$” here – for clarity the implementation displays the hierarchical constraints between calls and returns, even though those are not part of the Petri structure.

3.1.3 Handling duplications. Next we introduce a crucial aspect of Petri structures: the need for thread indexing, leading to the exact definition of tokens. More precisely, we set:

**Definition 3.3.** The sets $E$ of exponential signatures and $D$ of data signatures are:

$$E := \ell \cdot E \mid r_a \cdot E \mid \langle E, E \rangle \mid \bullet \quad D := n \mid t \mid f \mid \checkmark \mid \bullet,$$

and we also write $E^*$ for finite lists of exponential signatures, called exponential stacks. We use e to range over exponential signatures, s to range over exponential stacks, and d for data signatures.

The set of tokens, written Tok, is simply defined as $E^* \times D$, ranged over by t.

Exponential stacks originate in standard Gt token machines [Danos and Regnier 1996]. An exponential signature uniquely identifies a precise resource occurrence within the net and allows us to route accesses to distinct resources via the same address. Figure 8 shows a net involving duplication. Upon receiving $r_a Q^-$, the net throws two tokens corresponding to the evaluations of $x$. Both tokens are redirected to location 3 enabling $\ell \cdot c_a^3 Q$, but with distinct exponential stacks. Subsequently, returns $\ell \cdot c_a^3 A$ are distributed according to their exponential signature. It requires
We now define the transition system, in two steps. For

3.1.4 The token game. Next, we formalize the token game on Petri structures.

As is familiar from the Petri net literature, a state of a Petri structure is called a marking:

Definition 3.4. Consider \( \sigma \) a Petri structure. A marking on \( \sigma \) is a finite subset of \( \text{Tok}_{IL} \).

The set of markings on \( \sigma \) is written \( \mathcal{M}(\sigma) \), and we use \( \alpha, \beta, \gamma \) as metavariables for markings.

We use the same notation for markings as for conditions, i.e. \( \alpha = \{([\ ]), [\ ]\}^{@1}, ([\ ]), [\ ]\}^{@2}, ([\ ]), [\ ]\}^{@2} \) is a (non-reachable) marking for the Petri structure of Figure 8. 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.

We define the token game, i.e. execution, as a walk on a labelled transition system on markings. While neutral transitions act on markings only, visible transitions send or receive tokens on addresses, regarded as channels. Together, an address and a token form a move – we define \( \text{Moves} = \mathcal{M} \times \mathcal{E}^* \times \mathcal{D} \simeq \mathcal{M} \times \text{Tok} \). We use \( m \) to range over moves – notice the different font from \( m \) for addresses. We now define the transition system, in two steps. For \( \sigma \) a Petri structure, we set:

Definition 3.5. We define the instantiated transitions (itransitions for short) as one of:

\[
\begin{align*}
t^\theta([\alpha]) & : \alpha \longrightarrow_\sigma \beta & \text{if } \alpha \in \text{cond}(\text{pre}(t)) \text{ and } \delta(t)(\alpha) = \beta, \\
t^+([\alpha]) & : \alpha \longrightarrow_\sigma \emptyset & \text{if } \alpha \in \text{cond}(\text{pre}(t)), \delta(t)(\alpha) = (s, d) \text{ and } m = (\partial(t), s, d), \\
t^-([s, d]) & : \emptyset \longrightarrow_\sigma \beta & \text{if } m = (\partial(t), s, d) \text{ with } \delta(t)(s, d) = \beta,
\end{align*}
\]

where \( t \in \mathcal{T} \) has the indicated polarity; we set \( \partial(t)[\emptyset] \) = \( m \) the move labelling a visible transition.

We write \( \text{IT}_\sigma \) the set of itransitions of \( \sigma \), and use \( t \) to range over those.

A visible iteration \( t \) is labelled with a move \( m = (m, s, d) = \partial(t) \). For negative iterations we read this as the token \((s, d)\) being received on \( m \), while for positive iterations, \((s, d)\) is sent on \( m \). The actual token game is played by instantiated transitions in context:

Definition 3.6. An instantiated transition in context (ictransition for short) is one of:

\[
\begin{align*}
t^\theta \cup \gamma & : \alpha \cup \gamma \longrightarrow_\sigma \beta \cup \gamma & \text{if } t : \alpha \longrightarrow_\sigma \beta, \\
t^+ \cup \gamma & : \alpha \cup \gamma \longrightarrow_\sigma \gamma & \text{if } t : \alpha \longrightarrow_\sigma \emptyset, \\
t^- \cup \gamma & : \gamma \longrightarrow_\sigma \gamma \cup \beta & \text{if } t : \emptyset \longrightarrow_\sigma \beta,
\end{align*}
\]

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

We write \( \text{ITC}_\sigma \) for the set of ictransitions of \( \sigma \), ranged over by \( t \) (note the different font).
Definition 3.7. A run in \( \sigma \) is a sequence \( \rho = t_1 \ldots t_n \) of ictransitions s.t. \( t_1 : a_i \xrightarrow{m} a_{i+1} \) with \( a_1 = \emptyset \). We write \( \rho : \emptyset \xrightarrow{a} a_{n+1} \) or \( \rho : \emptyset \xrightarrow{s} a_{n+1} \), where \( s = m_1 \ldots m_\rho \) lists the labels appearing in \( \rho \). We also write \( s = \text{play}(\rho) \) and call \( s \) the play of \( \rho \).

For example, the following is a run \( \rho \) for (the variant with full tokens of) Figure 5:

\[
\emptyset \xrightarrow{(Q,[\bullet])} \{([\bullet],[\bullet],[\cdot])\} \xrightarrow{([\bullet],[\bullet],[\cdot])} \{([\bullet],[\bullet],[\cdot])\} \xrightarrow{([\bullet],[\bullet],[\cdot])} \emptyset
\]

with play(\( \rho \)) = (Q, [], \bullet)(A, [], \bullet). It may be visualized as tokens walking through the Petri net as in Figure 9, ignoring the exponential stacks (which are always \([ \}] \) in this example).

Among other things, this lets us define a notion of \textit{may-convergence} for closed Petri structures:

Definition 3.8. Consider \( \sigma \) a closed Petri structure. We say \( \sigma \) may converge, written \( \sigma \parallel \), iff there is a run \( \rho : \emptyset \xrightarrow{a} \alpha \) such that play(\( \rho \)) = (Q, [], \bullet)(A, [], \bullet, d) for some \( d \neq \bullet \).

Next, we set toward defining the interpretation of IPA with Petri structures; i.e. following Section 2.2, constructing an IPA-structure with Petri structures as morphisms. The conceptual core of this endeavour is the definition of a precategory of Petri structures, and in particular their composition.

3.2 The Precategory PStruct

We build a precategory PStruct. Its objects are finite sets \( M \subseteq M \), considered as interfaces. A morphism from \( M \) to \( N \) is a Petri structure on \( M +^\circ N = \ell(M) \cup r(N) \), up to isomorphism:

Definition 3.9. Consider \( \sigma, \tau \) two Petri structures on \( M \subseteq M \). An isomorphism \( \phi : \sigma \equiv \tau \) consists of bijections \( \varphi_l : L_\sigma \cong L_\tau \) and \( \varphi_r : T_\sigma \cong T_\tau \) compatible with all structure.

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

Definition 3.10. Visible transitions \( t^\sigma \in \mathcal{T}_\sigma^+ \cup \mathcal{T}_\sigma^- \) and \( t^\tau \in \mathcal{T}_\tau^+ \cup \mathcal{T}_\tau^- \) are synchronizable if they have opposite polarities; and \( \partial_\sigma(t^\sigma) = r m \) while \( \partial_\tau(t^\tau) = \ell m \) for some \( m \in N \). We define the set

\[
\mathcal{T}_\tau \circ \mathcal{T}_\sigma = \{ t^\sigma(t^\tau) | t^\sigma \in \mathcal{T}_\sigma, t^\tau \in \mathcal{T}_\tau \text{ synchronizable} \}.
\]

with \( \mathcal{T}_\sigma^{p,i} \) transitions of polarity \( p \in \{-, +\} \) and label \( \partial(t) = \ell m \) for some \( m \in M \); idem for \( \mathcal{T}_\tau^{p,i} \).

Intuitively, \( \mathcal{T}_\tau \circ \mathcal{T}_\sigma \) imports an unsynchronized transition \( t \) from \( \sigma \) as \( t^\circ(t) \), an unsynchronized transition \( t \) from \( \tau \) as \( r^\circ(t) \), but also has a new transition \( t^\circ \circ t^\tau \) for every synchronizable pair.

We have used \( t^\circ \) and \( r^\circ \) to keep transitions from \( \sigma \) and \( \tau \) disjoint. From now on, if \( X \) and \( Y \) are sets, we write \( X +^\circ Y = t^\circ(X) \cup r^\circ(Y) \). We shall use the same convention with other tags later on - or simply write \( X + Y = \ell(X) \cup r(Y) \). Now, we may finally define the composition \( \tau \circ \sigma \) as:

Definition 3.11. We set \( L_{\tau \circ \sigma} = L_\sigma +^\circ L_\tau; \mathcal{T}_{\tau \circ \sigma} = \mathcal{T}_\sigma \circ \mathcal{T}_\tau; \mathcal{T}_{\tau \circ \sigma}^{p,i} = \mathcal{T}_\sigma^{p,i} +^\circ \mathcal{T}_\tau^{p,i} \) where \( p \in \{+, -, \} \); \( \partial_{\tau \circ \sigma}(t^\circ(t)) = \partial_\sigma(t) \) and \( \partial_{\tau \circ \sigma}(r^\circ(t)) = \partial_\tau(t) \). Conditions are in Figure 10, and:

\[
\begin{align*}
\delta_{\tau \circ \sigma}(t^\circ(\alpha)) &= t^\circ(\delta_\sigma(\alpha)) + t^\circ(\delta_\tau(\beta)), \\
\delta_{\tau \circ \sigma}(r^\circ(\beta)) &= r^\circ(\delta_\tau(\beta)) + r^\circ(\delta_\sigma(\alpha)), \\
\delta_{\tau \circ \sigma}(t^\circ \circ t^\tau(\alpha)) &= r^\circ((\delta_\tau(\alpha) \circ \delta_\sigma(t^\tau)))(\alpha), \\
\delta_{\tau \circ \sigma}(r^\circ \circ t^\tau(\beta)) &= t^\circ((\delta_\sigma(t^\tau) \circ \delta_\tau(\beta)))(\beta).
\end{align*}
\]

where \( t^\circ \) and \( r^\circ \) are applied to \( \alpha \in \text{cond}_\sigma \) and \( \beta \in \text{cond}_\tau \) by retagging locations, i.e. with \( t^\circ(\alpha) = \{(s,d)^{\circ l} \mid (s,d)^l \in \alpha \} \in \text{cond}_{\tau \circ \sigma} \) and \( r^\circ(\beta) = \{(s,d)^{\circ r} \mid (s,d)^r \in \beta \} \in \text{cond}_{\tau \circ \sigma} \).
We omit the transition tables in order to save space. Notice how the two copies of Figure 4 glue together the disconnected components – for calls and returns – of the left hand side operand.

Intuitively, runs on \( \tau \circ \sigma \) happen as follows: tokens first appear via negative transitions on either side. The token game is, at first, played independently in \( \sigma \) and \( \tau \). As soon as \( \sigma \) wishes to play a positive transition on the right (resp. \( \tau \) wishes to play a positive transition on the left), it synchronizes with the matching negative transition on the other side – if it exists – and the resulting (neutral) transition has transition function the composite as for the two compounds. The effect of that synchronization is that tokens “jump” between \( \sigma \) and \( \tau \), following the control flow.

### 3.2.2 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 3.12.** For \( M \subseteq M \) finite, we define the **copycat Petri structure** on \( M \), written \( \omega_M \).

Its **locations** are \( L_{\omega_M} = \{ l \} \), its **transitions** are \( T_{\omega_M} = \{ l, r \} \) with polarities as in

\[
T_{\omega_M}^+ = (M^+ \times \{ r \}) \cup (M^- \times \{ l \}) \quad T_{\omega_M}^- = (M^- \times \{ r \}) \cup (M^+ \times \{ l \})
\]

and no neutral transition. We set \( \partial(m, l) = l, m \) and \( \partial(m, r) = r, m \); we set \( \text{pre}(m^+, r), \text{pre}(m^-, l), \text{post}(m^-, r) \) and \( \text{post}(m^+, l) \) to \( \{ m \} \) and pre and post returning \( \emptyset \) elsewhere. Finally:

\[
\delta((m^+, r))((s, d)@m) = (s, d) \quad \delta((m^-, r))(s, d) = \{ (s, d)@m \}
\]

As an example, we show in Figure 12 the Petri structure \( \omega_{\mathbb{N} \rightarrow \mathbb{N}} \), writing \( \mathbb{N} \rightarrow \mathbb{N} \) for the set \( \{ \ell, Q^+, \ell, A^-, r, Q^-, r, A^+ \} \) which shall arise as the interpretation of \( \mathbb{N} \rightarrow \mathbb{N} \).

Associativity of composition up to iso is direct, which makes PStruct a precategory. Note that copycat is not neutral for composition up to iso: composing with copycat yields a structure with strictly more nodes – it is thus the reason why we based IPA-structures on precategories rather than categories. Copycat will be neutral for composition only w.r.t. unfolding, see Section 5.2.

### 3.3 PStruct as an IPA-structure: Constructions, Operations

Now, we introduce the IPA-structure operations for Petri structures, following Definition 2.1.

For PStruct we fix \( PStruct_w = PStruct_0 \): we do not need to distinguish well-opened objects.

---

3.3.1 Constructions. First of all, the constructions of Definition 2.1 are applied to finite sets of addresses simply by applying the corresponding injections from Definition 3.2. In other words we set $M \otimes N = M +^\oplus N$; $\&_{x \in V}M_x = \sqcup_{x \in V} \tau^x(M_x)$; $M \twoheadrightarrow N = M +^\rightarrow N$ and $1M = M$.

For basic types, we set $U, B$ and $N$ as $G = \{Q^-, A^+\}$. We postpone $V$ and $S$ to Section 3.5.

3.3.2 Tensor. Next, we define tensor of Petri structures. Fix $\sigma, \tau$ Petri structures:

**Definition 3.13.** Consider $\sigma \in \text{PStruct}(M_1, N_1)$ and $\tau \in \text{PStruct}(M_2, N_2)$.

We set $L_{\sigma \otimes \tau} = L_\sigma +_\otimes L_\tau$; $T_{\sigma \otimes \tau} = T_\sigma +_\otimes T_\tau$ with $T^{\otimes}_\sigma = T^p_\sigma +_\otimes T^p_\tau$ for $p \in \{+, -\}$;

\[
\begin{align*}
\delta_{\sigma \otimes \tau}(t^\otimes(t)) &= \ell, t_m (\delta_{\sigma}(t) = \ell, m) \\
\delta_{\sigma \otimes \tau}(t^\otimes(t)) &= \tau, t_m (\delta_{\sigma}(t) = \tau, m)
\end{align*}
\]

pre- and post-conditions in Figure 13, and for the transition table we set:

\[
\begin{align*}
\delta_{\sigma \otimes \tau}(t^\otimes(a)) = t^\otimes(\delta_{\sigma}(t)(a)) \\
\delta_{\sigma \otimes \tau}(t^\otimes(t)) = t^\otimes(\delta_{\sigma}(t)(t))
\end{align*}
\]

with $t^\otimes, r^\otimes$ applied on conditions as in Definition 3.11. This yields $\sigma \otimes \tau \in \text{PStruct}(M_1 \otimes M_2, N_1 \otimes N_2)$.

This simply puts $\sigma$ and $\tau$ side by side without interaction. As an example, the Petri structure on the right hand side of the composition symbol in Figure 11 is $\sigma \otimes \sigma$ for $\sigma$ in Figure 4.

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

**Definition 3.14.** Take $\sigma$ a Petri structure on $M$ and $f : M \rightarrow M$ s.t. $M \subseteq \text{dom}(f), f(M) \subseteq N$.

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

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

**Definition 3.15.** Consider $\sigma \in \text{PStruct}((![\Gamma, x : A, \Delta], O))$. We define the function $\Lambda_x : M_\epsilon \rightarrow M_\epsilon$ by $\Lambda_x(\tau_m) = \tau \cdot m \cdot \epsilon$, $\Lambda_x(\tau, \ell \cdot s \cdot m) = \epsilon \cdot \tau \cdot m$, and $\Lambda_x(m) = m$ otherwise.

Then, setting $\Lambda^\Delta_{x, A, O} (\sigma) = \sigma[\Lambda_x]$, we obtain $\Lambda^\Delta_{x, A, O} (\sigma) \in \text{PStruct}(![\Gamma[\Delta, A], t \rightarrow O])$.

This reassigns visible transitions corresponding to variable $x$ to the left hand side of $\rightarrow$ on the right hand side of $t$. 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.

3.3.4 Promotion. The final operation on Petri strategies is promotion, for a Petri structure $\sigma$:

**Definition 3.16.** Consider $\sigma \in \text{PStruct}(!M, N)$. We set $L^{\uparrow}_\sigma = L_\sigma$, $T^{\uparrow}_\sigma = T_\sigma$ with the same polarities, $\partial^{\uparrow}_\sigma = \partial_\sigma$, and pre- and post-conditions are also unchanged. Finally, the transition table is:

\[
\begin{align*}
\delta^{\uparrow}_\sigma (t^\alpha)(e :: \alpha) &= e :: \beta \quad \text{if } \delta_\sigma(t)(a) = \beta \\
\delta^{\uparrow}_\sigma (t^\alpha)(e :: a) &= (e :: s, d) \quad \text{if } \delta_\sigma(t)(a) = \tau \text{ and } \delta_\sigma(t)(a) = (s, d) \\
\delta^{\uparrow}_\sigma (t^\alpha)(e :: a) &= ((e, e') :: s, d) \quad \text{if } \delta_\sigma(t)(a) = \ell \text{ and } \delta_\sigma(t)(a) = (e' :: s, d) \\
\delta^{\uparrow}_\sigma (t^\alpha)(e :: s, d) &= e :: a \quad \text{if } \delta_\sigma(t)(a) = \tau \text{ and } \delta_\sigma(t)(s, d) = \alpha \\
\delta^{\uparrow}_\sigma (t^\alpha)(e :: s, d) &= e :: \alpha \quad \text{if } \delta_\sigma(t)(a) = \ell \text{ and } \delta_\sigma(t)(e' :: s, d) = \alpha
\end{align*}
\]

where $\delta_\sigma(t) = \epsilon$ means $\delta_\sigma(t) = \epsilon m$ for some $m \in M$, and $e :: \alpha$ is $\{(e :: s, d) | (s, d) \in \alpha\}$.

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

In contrast with currying, promotion only affects the transition table, not the other components: promotion only manipulates the exponential signatures, which are only present in the tokens.

Those rules are similar to those dealing with promotion in standard GóI token machines [Danos and Regnier 1996]; their behaviour is somewhat subtle. The central idea is that $\sigma^+$ has – implicitly – countably many copies of $\sigma$ running in parallel. In reality all tokens flow through the same net, but copies are kept apart by the exponential signature added as a new layer of the exponential stack (the rest of the exponential stack concerns promotions deeper within the net). Finally, when interacting with the outside world on the left, this new layer of the exponential stack must be merged with the previous top of the stack (an effect known as digging in linear logic jargon).

3.4 PStruct as an IPA-structure: Stateless Primitives

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

3.4.1 Variable, evaluation. First, the variable is simply a copycat set to component $\iota$ of the context.

**Definition 3.17.** Consider $M \subseteq M$ finite, and $x \in \operatorname{Var}$.

We define $\operatorname{Var}_x M$ as $\mathcal{L}_{\operatorname{Var}_x M} = \mathcal{L}_{\mathcal{M}}$, $\mathcal{T}_{\operatorname{Var}_x M} = \mathcal{T}_{\mathcal{M}}$, with the same pre- and post-conditions as for $\omega M$. We set $\partial_{\operatorname{Var}_x M} (m, \ell) = \ell, \iota^x m$ and $\partial_{\operatorname{Var}_x M} (m, r) = r m$. Finally, the transition table is:

$$
\begin{align*}
\delta((m^+, r))((s, d)^{\oplus m}) &= (s, d) & \delta((m^-, r))((s, d)^{\ominus m}) &= \{ (s, d)^{\ominus m} \\
\tilde{\delta}((m^-, \ell))((s, d)^{\ominus m}) &= (\vdash s, d) & \tilde{\delta}((m^+, \ell))((\vdash s, d) &= \{ (s, d)^{\ominus m} 
\end{align*}
$$

Recall that terms $\Gamma \vdash M : A$ are meant to be interpreted as morphisms $[M] \in C(\![](\Gamma), [A])$. The $!$ explains the $\vdash$ 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 3.18.** For $M, N \subseteq M$ finite sets, $\mathcal{E}_{M, N} = \omega_{M \rightarrow N} [\Omega]$, where $\Omega : M \rightarrow M$ is $\Omega(x, r, m) = r m$, $\Omega(x, \ell, m) = \ell, r^x m$, $\Omega(\ell, r, m) = \ell, r^x m$, $\Omega(\ell, \ell, m) = \ell, \ell, m$, and $\Omega(m) = m$ otherwise.

3.4.2 Other stateless primitives. For finite $M \subseteq M$, the (binary) contraction $c_M$ has $\mathcal{L}_{c_M} = M$, transitions $\mathcal{T}_{c_M} = M + ^+ (M \times M)$ with polarity as in Definition 3.2. The net (i.e. pre- and post-conditions) appears in Figure 15a, and the transition rules in Figure 14. The Petri structure behaves as in Figure 8, using the injections $\ell$ and $r$ from exponential signatures to record the routing back to the node from which a request originates. The $n$-ary contraction, defined likewise, is omitted – it may also be obtained by induction, composing binary contractions.

The fixpoint combinator $Y_M$ is similar: it has $\mathcal{L}_{Y_M} = M$, $\mathcal{T}_{Y_M} = (M + ^+ M) + ^+ M$, net in Figure 15b and transition rules in Figure 14. The net has no loops, but loops may arise by composition.

Constants, conditionals and operations are introduced in Figures 15d, 15e and 15c respectively, with transition rules in Figure 14. Hopefully their behaviour is clear at this point.

Finally, the let has net in Figure 15f and transition rules in Figure 14. We explain its behaviour, which is more subtle. Recall that Definition 2.1 requires $\mathcal{L}_{X Y} \in C(\![](X \rightarrow Y) \times Y, X, Y)$, and in this case let $\in \operatorname{PStruct}(\![](G \rightarrow G) \times G, G)$. Upon being called with $r Q^-$, let triggers the evaluation of its argument at $\ell, r^x Q$ – this evaluation is performed once, and will never happen again. Upon receiving a value with $\ell, r A^+$, let does two things: it gives the control to its function argument at $\ell, r^x A^+$. In parallel, it memoizes the value by storing it into location 3. This value is then read each time the function calls its argument. In essence, let is a read-only memory cell.

This naturally brings us to our final primitives, dealing with mutable state and semaphores.
Recall that Definition 2.1 requires primitives

\[ \text{newref} \in C(V \otimes N, U), \quad \text{deref} \in C(V, N), \quad \text{grab} \in C(S, U), \quad \text{release} \in C(S, U) \]

for reference and semaphore queries. Among these, we set \( \text{deref} = \alpha_N[\ell, m \mapsto \ell, \kappa, m] \), \( \text{grab} = \alpha_N[\ell, m \mapsto \ell, g, m, \kappa, m \mapsto \kappa, m] \) and \( \text{release} = \alpha_N[\ell, m \mapsto \ell, \kappa, m, \kappa, m \mapsto \kappa, m] \). Copycat strategies simply accessing the matching component of the reference or semaphore.

Fig. 14. Transition tables for IPA-structure primitives

(a) Contraction \( c_M \)

(b) Fixpoint combinator \( Y_M \)

(c) Operator \( \text{op} \)

(d) Constant

(e) Petri structure for if

(f) Petri structure for let

(g) Petri structure for newref

(h) Petri structure assign

(i) Petri structure for newsem

Fig. 15. The nets of Petri structures for IPA primitives

3.5 PStruct as an IPA-structure: Stateful Primitives

For types \( V \) and \( S \) we have addresses \( V = \{w, \kappa, Q, \omega, A, \kappa; Q, \omega; A\} \) and \( S = \{g, \xi, Q, g, A, \xi; Q, \xi; A\} \), where \( w \) is the address for write requests, \( \kappa \) for read requests, \( g \) for grab, \( \xi \) for release.
The Concurrent Games Abstract Machine

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 15h and its transition rules are trivial.

3.5.2 Initialization. The actual stateful behaviour is provided by newref ∈ C(!V → X, X) and newsem ∈ C(!S → X, X) with nets in Figures 15g and 15i, non-trivial transitions in Figure 14.

The Petri structure newref takes as an argument !V → X, i.e. a “program of type X” that may query a reference. Location 2 stores the current value. Upon initialization with 𝜓 ⊢ Q−, newref: (1) initializes the reference, setting a token with data 0 in location 2; (2) in parallel, gives control to the argument via ℓ, 𝜓 ⊢ Q+. Write and read requests arrive respectively in locations 3 and 5. The neutral transitions w and r perform the memory update and reading. In the normal operation, there is ever at most one token in location 2, forcing reads and writes to be handled in some sequential order.

The net for newsem is essentially the same. In normal operation, location 2 contains exactly one token with data either t 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.

3.6 Wrapping up the Interpretation
With the above, we have completed the construction for:

Corollary 3.19. PStruct is an IPA-structure.

Following Section 2.3, we obtain the interpretation of a type A of IPA as a finite set [A] ⊆ M of addresses, likewise for a context, and of a term Γ ⊢ M : A as [M] ∈ PStruct(!Γ, [A]) which we regard as the concurrent games abstract machine with M loaded. In particular, a closed program ⊢ M : X is interpreted as a Petri structure [M] on set of addresses {rQ−, rA+}, i.e. a closed Petri structure up to the obvious renaming. It will follow from the later results of this paper that:

Theorem 3.20 (Adequacy). For any closed program ⊢ M : U, we have M ⇓ iff [M] ⇓.

This is the standard way of stating that at ground type and with respect to may-convergence, the concurrent games abstract 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 interpreting M.

4 CONCURRENT STRATEGIES
Before we describe this unfolding, we must recall its target. So we include a brief reminder of the concurrent games model and its adequate semantics of IPA. The model and its interpretation of IPA is first described in [Castellan et al. 2019], with improvements and a detailed adequacy proof in [Castellan and Clairambault 2020]. With respect to this, the present model differs in two ways. Firstly, our moves follow Petri structures and use exponential signatures rather than natural numbers as copy indices. Secondly, unlike in [Castellan and Clairambault 2020], our games and strategies come without symmetry – we discuss this further in Appendix B.4.

4.1 Types and Contexts as Games
The first step is to introduce games: a game collects the moves one may encounter during an execution on a given type, and organizes them according to their causal dependencies and conflict.

In concurrent games, both games and strategies are certain event structures:

4.1.1 Event structures. Event structures are a well-known model from concurrency theory, presenting a system by specifying causal dependency and conflict between computational events:
Definition 4.1. An event structure (es) is a triple \( E = (|E|, \leq_E, \#_E) \), where \(|E|\) is a (countable) set of events, \( \leq_E \) is a partial order and \( \#_E \) is an irreflexive symmetric binary relation, satisfying:

- finite causes: \( \forall e \in |E|, [e]_E = \{ e' \in |E| \mid e' \leq e \} \) is finite
- conflict inheritance: \( \forall e_1 \#_E e_2, \forall e \leq e_1' e_2' \),

and we call \( \#_E \) and \( \leq_E \) respectively conflict and causal dependency.

We write \( \rightarrow_E \) for the immediate dependency relation induced by \( \leq_E \), i.e. \( e \rightarrow_E e' \) iff \( e \leq_E e' \) with no other event strictly in between. An event structure \( E \) naturally comes with a notion of state, its (finite) configurations: those are finite sets \( x \subseteq |E| \) which are down-closed for \( \leq_E \) and compatible, in the sense that if \( e, e' \in x \), then \( \neg (e \#_E e') \). We write \( \mathcal{C}(E) \) the set of configurations of \( E \). If \( x \in \mathcal{C}(E) \) and \( e \notin x \) satisfies \( x \cup \{ e \} \in \mathcal{C}(E) \), we say that \( x \) enables \( e \) and write \( x +_E e \).

4.1.2 Games and arenas. Games are event structures composed of moves:

Definition 4.2. A game is an \( A = (|A|, \leq_A, \#_A) \) s.t. \(|A| \subseteq \text{Moves} \) and satisfying:

- finite addresses: the set \( \text{mult}(A) = \{ m \in M \mid \exists (m, s, d) \in |A| \} \) is finite.

This is the usual notion of concurrent games, with the small difference that moves are taken in \( \text{Moves} \). Consequently, polarity is inherited and no longer part of the data – we still display polarities in diagrams. The condition ensures that \( \text{mult}(A) \) is an object of PStruct for any \( A \).

In Figure 16, we show the interpretation for ground types. In diagrams for games, the dotted lines indicate immediate causal dependency, flowing from top to bottom. Wiggly lines indicate conflict – in the diagram for N it is understood that all moves in the second row are in pairwise conflict, some being omitted to alleviate notations. As usual in game semantics for call-by-name languages, these diagrams formalize a protocol where the environment initiates computation by playing the negative move (a question), following which Player may respond a value (via an answer). All the moves in Figure 16 have a trivial address without injections: non-trivial addresses will arise only with constructors. Likewise, they all have trivial exponential stacks as the types do not contain a !. Finally, the data signature matches the return value whenever relevant, and is \( \bullet \) otherwise.

It will be helpful in the sequel to have a tighter grasp on the shape of games arising from types:

Definition 4.3. An arena is a game \((A, \leq_A, \#_A)\) satisfying:

- alternating: if \( a_1 \rightarrow_A a_2 \), then \( \text{pol}(a_1) \neq \text{pol}(a_2) \),
- forestial: if \( a_1 \leq_A a \) and \( a_2 \leq_A a \), then \( a_1 \leq_A a_2 \) or \( a_2 \leq_A a_1 \),
- locally conflicting: if \( a_1, a_2 \in |A| \) are in minimal conflict, then they are both minimal, or have the same (necessarily unique) predecessor.
- negative: if \( a \in \text{min}(A) \), then \( \text{pol}(a) = - \),

where \( \text{min}(A) \) comprises the minimal events of \( A \). Finally, \( A \) is well-opened if \( \text{min}(A) \) is a singleton.

This used 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, \( \neg (e'_1 \#_E e'_2) \) – so the conflict is not inherited. In that case we write \( e_1 \sim_E e_2 \); note that minimal conflict is what is represented throughout this paper in event structure diagrams.

Types and contexts shall be interpreted as arenas, via the constructions introduced next.
4.1.3 Basic game constructions. We write either 1 or \( T \) for the empty arena.

If \((m, s, d)\in\text{Moves}\), we write \(\ell_e((m, s, d)) = (\ell_e(m), s, d)\) 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. \(\ell_e(R) = \{(\ell_e(m), \ell_e(m')) | m R m'\}\).

**Definition 4.4.** Consider \(A, B\) games. Then we set games \(A \otimes B\) and \(A \rhd B\) with components:

\[
\begin{align*}
|A \otimes B| &= \ell_e(|A|) \cup \ell_e(|B|) \\
\leq_{A \otimes B} &= \ell_e(\leq_A) \cup \ell_e(\leq_B) \\
\#_{A \otimes B} &= \ell_e(\#_A) \cup \ell_e(\#_B)
\end{align*}
\]

\(A \otimes B\) is called the **tensor**, and \(A \rhd B\) is called the **hom-game**.

If \(A, B\) are arenas, so is \(A \otimes B\). In contrast, \(A \rhd B\) is never an arena unless \(A\) is empty. Formally, both constructions are defined in the same way, but with a distinct injection – remember from Definition 3.2 that \(\ell_e\) inverts polarity, so that in \(A \rhd B\) the polarity is inverted in \(A\).

From the definition, configurations of \(A \otimes B\) have a restricted shape: any \(x \in \mathcal{C}(A \otimes B)\) decomposes uniquely as \(x = x_A \otimes x_B\) with \(x_A \in \mathcal{C}(A)\) and \(x_B \in \mathcal{C}(B)\) – we write \(x = x_A \otimes x_B\). Likewise, any configuration \(x \in \mathcal{C}(A \rhd B)\) decomposes as \(x = x_A \rhd x_B = \ell_e(x_A) \rhd \ell_e(x_B)\).

4.1.4 Further arena constructions. We give the remaining constructions required by Definition 2.1.

First a new notation: for \(m = (m, s, d)\in\text{Moves}\) and \(e \in \mathcal{E}\), we write \(e \colon m = (m, e :: s, d)\), 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 4.5.** We define three other constructions on arenas:

- **linear arrow**: for \(A, O\) arenas with \(O\) well-opened, we define \(A \rightarrow O\) well-opened in Figure 17a.
- **product**: for \((A_x)_{x \in V}\) a family of arenas with \(V \subseteq \text{Var}\), we define \(\prod_{x \in V} A_x\) in Figure 17b.
- **bang**: for \(A\) an arena, we define \(!A\) in Figure 17c.

It remains to give the arenas for references and semaphores, in Figure 18.

The arrow \(A \rightarrow O\) 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 19. The representation is denoted from the exponentials in the construction, the full arena is infinite and comprises moves as in the diagram for all exponential signatures \(e, e' \in \mathcal{E}\). However, we have e.g. \((\ell_{\mathcal{E}}, \mathcal{Q}, [e], \bullet) \leq (\ell_{\mathcal{E}}, \mathcal{Q}, [e_1, e_2], \bullet)\) only when \(e = e_1\) – in an exponential stack, the first element corresponds to the outermost \(!(-)\).
4.2 The Category of Arenas and Strategies

4.2.1 Definition. In concurrent games, strategies are also event structures, labelled by the game:

Definition 4.6. A prestrategy $\sigma : A$ on game $A$ comprises an es $|\sigma|, \leq_{\sigma}, \#_{\sigma}$ with $\partial : |\sigma| \rightarrow |A|$ a function called the display map, subject to the following conditions:

- rule-abiding: for all $x \in E(\sigma), \partial(x) \in E(A),$
- locally injective: for all $s_1, s_2 \in |\sigma|, \text{if } \partial(s_1) = \partial(s_2) \text{ then } s_1 = s_2.$

We say that $\sigma$ is a strategy if it satisfies the further two conditions:

- courteous: for all $s_1 \Rightarrow_A s_2, \text{if } \text{pol}(s_1) = + \text{ or } \text{pol}(s_2) = - \text{ then } \partial(s_1) \Rightarrow_A \partial(s_2),$
- receptive: for all $x \in E(\sigma), \text{for all } \partial(x) \Rightarrow_A a^{-},$

there is a unique $x \Rightarrow_{\sigma} s \in E(\sigma)$ such that $\partial(s) = a,$

and additionally it is negative if for all $s \in |\sigma|, \text{if } s \text{ is minimal then } s \text{ is negative.}$

Rule-abiding and locally injective together amount to $\partial : \sigma \rightarrow A$ being a map of event structure. Events of $\sigma$ inherit a polarity from $\text{pol}_{\sigma}(s) = \text{pol}(\partial(s))$ – 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 (see Appendix B.4 for more).

The event structure $\sigma$ presents observable computational events along with their causal dependencies and conflicts. Events of $\sigma$ are not moves of the game, but they do correspond to moves via the action of $\partial.$ This permits an explicit representation of non-deterministic branching: several events of $\sigma$ 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 20 (part of) a concurrent strategy playing on the game in Figure 19. 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 $\partial$ – this means that two conflicting nodes may have the same label, as happens in Figure 20. Arrows $\Rightarrow_{\sigma}$ correspond to the immediate causal dependency in $\sigma,$ while dotted lines correspond as before to the causal dependency from the game. The diagram in Figure 20 is a part of the strategy for:

Example 4.7. We introduce the term $\triangleright$ strictness : $(U \rightarrow U) \rightarrow B,$ defined as

$\triangleright \lambda f:U\rightarrow U. \text{newref } x \text{ in } f \left( x := 1 \right); \text{not (iszero } !x) : (U \rightarrow U) \rightarrow B,$

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

Figure 20 reads as follows: Opponent initiates computation with $(r, Q, [\bullet], \triangleright)^{-}.$ This lets Player call his argument with $(\ell, r, Q, [\bullet], \bullet)^{+}.$ This lets Opponent return the call with $(\ell, r, A, [\bullet], \checkmark)^{-},$ or call its argument with $(\ell, \ell, Q, [\checkmark], e)^{-}$ for any $e \in E$ (in fact Opponent may call this argument arbitrarily many times with distinct exponential signatures, but Figure 20 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 20 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.

### 4.2.2 Isomorphic strategies

Strict equality of strategies is too strict to be useful; instead we use:

**Definition 4.8.** Consider \( \sigma, \tau : A \) two (pre)strategies on game \( A \).

An isomorphism \( \varphi : \sigma \cong \tau \) is an invertible map of es \( \varphi : \sigma \rightarrow \tau \) such that \( \partial_{\tau} \circ \varphi = \partial_{\sigma} \).

We write \( \sigma \cong \tau \) to mean that \( \sigma \) and \( \tau \) 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 \( \sigma \) and \( \tau \) are in one-to-one correspondence with order-isos \( \varphi : \mathcal{E}(\sigma) \cong \mathcal{E}(\tau) \) 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 \( x \in \mathcal{E}(\sigma) \) is covered if any \( m \in x \) maximal in \( x \) is positive; we write \( \mathcal{E}^+(\sigma) \) for the +-covered configurations of \( \sigma \). The action of an iso \( \varphi \) on +-covered configurations suffices to completely describe it:

**Lemma 4.9.** Consider \( \sigma, \tau : A \) two strategies, and \( \varphi : \mathcal{E}^+(\sigma) \cong \mathcal{E}^+(\tau) \) 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 \( x \in \mathcal{E}^+(\sigma) \).

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.

### 4.2.3 Composition

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

A strategy \( \sigma \) 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 \( \tau \circ \sigma : A \vdash C \).

Lemma 4.9 puts the emphasis on +-covered configurations, so we investigate what should be the +-covered configurations of \( \tau \circ \sigma \). It turns out that they correspond to pairs \( x^\sigma, x^\tau \in \mathcal{E}^+(\sigma) \) and \( x^\tau \in \mathcal{E}^+(\tau) \) whose synchronization of \( x^\sigma \) and \( x^\tau \) through \( B \) is “sound”, which we must now define.

We fix the convention that if \( x^\sigma \in \mathcal{E}(\sigma) \) and \( x^\tau \in \mathcal{E}(\tau) \), we write \( \partial_{\sigma} x^\sigma = x^\sigma_A + x^\sigma_B \) and \( \partial_{\tau} x^\tau = x^\tau_B + x^\tau_C \), where \( x^\sigma_A \in \mathcal{E}(A), x^\sigma_B \in \mathcal{E}(B) \), and \( x^\tau_C \in \mathcal{E}(C) \). We say \( x^\sigma \in \mathcal{E}(\sigma) \) and \( x^\tau \in \mathcal{E}(\tau) \) are matching if \( x^\sigma_B = x^\tau_B \) in which case it is unambiguous to write \( \partial_{\sigma} (x^\sigma) = x^\sigma_B \) and \( \partial_{\tau} (x^\tau) = x^\tau_B \). So \( x^\sigma \) and \( x^\tau \) reach the same state on \( B \), but it remains to see if they do it with compatible causal constraints. For that, we set \( x_A \parallel x_B \parallel x_C = \ell(x_A) \cup m(x_B) \cup r(x_C) \), and

\[
\partial_{\sigma}^\ell : x^\sigma \rightarrow x_A \parallel x_B \parallel x_C \\
\partial_{\tau}^m : x^\tau \rightarrow x_A \parallel x_B \parallel x_C
\]

\[
m \mapsto \ell(a) \quad \text{if } \partial_{\sigma}(m) = \ell(a), \quad n \mapsto m(b) \quad \text{if } \partial_{\tau}(n) = \ell(b), \\
m \mapsto m(b) \quad \text{if } \partial_{\sigma}(m) = r(c), \quad n \mapsto r(c) \quad \text{if } \partial_{\tau}(n) = r(c).
\]

are variants of the display maps set to embed \( x^\sigma \) and \( x^\tau \) 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 4.10.** Consider \( x^\sigma \in \mathcal{E}(\sigma) \) and \( x^\tau \in \mathcal{E}(\tau) \) matching configurations.

They are causally compatible if the relation \( \preceq = \preceq_\sigma \cup \preceq_\tau \) on \( x_A \parallel x_B \parallel x_C \) set with:

\[
\partial_{\sigma}^\ell(m) \preceq_\sigma \partial_{\sigma}^\ell(m') \quad \text{for } m \preceq_\sigma m' \\
\partial_{\tau}^m(n) \preceq_\tau \partial_{\tau}^m(n') \quad \text{for } n \preceq_\tau n'
\]
is acyclic. We also say that the pair \( x^\sigma, x^\tau \) is secured.
We show in Figures 21 and 22 examples of matching secured and non-secured pairs involved in computing the composition of the strategy of Figure 20 with $\lambda x^U. x$. In Figure 22, a deadlock directly arises from opposite causal constraints (highlighted in blue). This entails that the only result arising from this composition will be $\top$ from Figure 21, 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 4.11.** 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:

$$(- \odot -) : \{(x^\tau, x^\sigma) \in \mathcal{C}(\tau) \times \mathcal{C}(\sigma) \mid \text{causally compatible}\} \cong \mathcal{C}(\tau \odot \sigma)$$

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

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 (Lemma 5.11) which keeps the visible events only. In this paper, the above characterization suffices for our purposes.

Figure 23 shows an example composition which, ignoring exponentials, corresponds to:

$$[f : \mathbb{B} \to U \vdash f \text{ coin}] \circ \downarrow \lambda x^3. \text{ if } x \text{ skip skip} : 1 \vdash U.$$ 

Observe the resulting strategy has two distinct ways to converge, even though the two occurrences of $(xA, [], \sqrt{\_})^*$ 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.

### 4.3 An IPA-structure

The remaining structure required for the interpretation of IPA follows [Castellan and Clairambault 2020; Castellan et al. 2019]. We must omit the details for lack of space, but they can be found in Appendix B. Altogether, and with Theorem 4.40 of [Castellan and Clairambault 2020]:

**Theorem 4.12.** The category Strat forms an IPA-structure. Moreover, for any closed program $\vdash M : U$, $[M]_{\text{Strat}} \downarrow \text{iff } M \downarrow$.

In the statement above, we say that $\sigma : U$ converges, written $\sigma \downarrow$, iff it has a positive move.

Of course, Strat is most interesting not for programs of ground types, but for higher-order programs: there, it gives a complete account of the underlying causal structure behind observable computational actions – along with the non-deterministic branching information.

In the remainder of this paper we show that though \([M]_{\text{Strat}}\) is computed by definition denotationally by induction on terms following the corresponding operations on strategies, it may also be computed from the execution by unfolding the Petri structure interpreting \(M\).

5 PETRI STRATEGIES AND UNFOLDING

Whereas concurrent strategies follow the rules of a game, unrestricted Petri structures are too liberal; they may play moves that are not accessible yet, or even meaningless (e.g. a boolean on a unit type). Before we unfold, we must capture, among Petri structures, those that respect the rules.

5.1 Strategic Petri Structures

5.1.1 Definition. Fix \(A\) a game, and \(\sigma\) a Petri structure on \(\text{mult}(A)\). Intuitively, \(\sigma\) is a Petri strategy on \(A\) if its runs follow the rules specified by \(A\), i.e. form valid plays on \(A\):

- **valid:** for all \(1 \leq i \leq n\), \([s_1, \ldots, s_i] \in \mathcal{C}(A)\),
- **non-repetitive:** for all \(1 \leq i \leq j \leq n\), if \(s_i = s_j\) then \(i = j\),

we write \(\text{Plays}(A)\) for the set of plays on game \(A\).

Recall that any run \(\rho : \emptyset \rightarrow_{\sigma} \alpha\) on \(\sigma\) generates a sequence of moves \(\text{play}(\rho)\) obtained by collecting the labels of visible moves in \(\rho\). Requiring that \(\text{play}(\rho) \in \text{Plays}(A)\) for every run \(\rho\) is too brutal; we shall ask only that \(\sigma\) behaves as prescribed by the game as long as Opponent does. But we shall also need a safety condition, required for the forthcoming unfolding. For that, we set:

5.1.2 Definition. Consider \(t : \alpha \rightarrow_{\sigma} \beta\) an itransition of \(\sigma\). We call \(\text{pre}(t) = \alpha\) the pre-condition of \(t\), and \(\text{post}(t) = \beta\) the post-condition of \(t\). The set \(\text{new}(t) = \beta \setminus \alpha\) contains the tokils produced by \(t\); and \(\text{eat}(t) = \alpha \setminus \beta\) those consumed. Those extend to ictransitions in the obvious way.

The distinction between \(\text{post}(t)\) and \(\text{new}(t)\) matters for \(\text{let}\): its transition in Figure 14 requires \(([\text{[]}, \text{d}])^{@3}\) to fire, but leaves it unchanged. So \(([\text{[]}, \text{d}])^{@3}\) is both a pre-condition and a post-condition, but is not produced by the transition. Other than for \(\text{let}\), pre- and post-conditions are always disjoint.

Our safety constraint uses the notion of collection of the tokils encountered in a run:

5.1.3 Definition. Consider \(\rho = t_1 \ldots t_n : \emptyset \rightarrow_{\sigma} \alpha_{i+1}\), a run of \(\sigma\), with \(t_i : \alpha_i \rightarrow \alpha_{i+1}\).

The collection of \(\rho\) is \(\text{Coll}(\rho) = \bigcup_{1 \leq i \leq n+1} \alpha_i\). We say \(\alpha \in \text{cond}_\sigma\) is fresh in \(\rho\) iff \(\alpha \cap \text{Coll}(\rho) = \emptyset\).

This lets us finally give the definition of Petri strategies on a game \(A\):

5.1.4 Definition. We say that \(\sigma\) is a Petri strategy on \(A\), written \(\sigma : A\), if for all \(\rho : \emptyset \rightarrow_{\sigma} \alpha\):

- **valid:** if \(s \in \text{Plays}(A)\) and \(t^+ : \alpha \rightarrow_{\sigma} \beta\), then \(sa \in \text{Plays}(A)\),
- **receptive:** \(\forall s a^- \in \text{Plays}(A), \exists ! t^- \in \text{IT}_\sigma\) s.t. \(t^- : \emptyset \rightarrow_{\sigma} \beta\) for some \(\beta \cap \alpha = \emptyset\),
- **strongly safe:** if \(s \in \text{Plays}(A)\) and \(t : \alpha \rightarrow_{\sigma} \beta\), or \(t : \alpha \rightarrow_{\sigma} \beta\) with \(sa \in \text{Plays}(A)\), then \(\text{new}(t)\) is fresh in \(\rho\),

and additionally it is negative iff for all \(t^0 : \alpha \rightarrow_{\sigma} \beta\) or \(t^+ : \alpha \rightarrow_{\sigma} \emptyset\), we have \(\alpha \neq \emptyset\).

Strong safety entails that in a given execution, a tokil can occur at most once: it cannot appear, then be consumed, only to reappear later. The main consequence of this is that tokils in a run of a Petri strategy have a canonical causal partial ordering, which will be central in the unfolding of Petri strategies to actual strategies – we direct to Section 5.2 for more on this.

Examples of Petri strategies abound in this paper so far – since it shall follow from this section 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 \(1\) of Section 3.1.1 does not satisfy \(1 : U\), as there is a clear failure of condition valid.
5.1.2 An IPA-structure. This lets us refine PStruct as follows:

**Proposition 5.5.** There is an IPA-structure PStrat, with objects arenas (with the associated constructions), and morphisms from A to B the negative Petri strategies \( \sigma : A \to B \) up to iso.

There is a strict \( F : \text{PStrat} \to \text{PStruct} \) sending an arena \( A \) to mult(\( A \)) and preserving morphisms.

**Proof.** We must show that all primitives are Petri strategies, and that all operations preserve those. The main technical challenge is composition: there we show any \( \rho : \emptyset \to \star \cdot \alpha \cdot \sigma \) projects to

\[ \rho_\sigma : \emptyset \to_{\sigma} \alpha_\sigma, \quad \rho_\tau : \emptyset \to_{\tau} \alpha_\tau \]

with \( \alpha = \alpha_\sigma + \circ \alpha_\tau \). If \( \sigma : A \to B \) and \( \tau : B \to C \) and if additionally in play(\( \rho \)), the external Opponent respects the game, none of Opponent, \( \sigma \) and \( \tau \) can be the first to break the rules; and we get by induction play(\( \rho \)) \( \in \text{Plays}(A \to C) \), play(\( \rho_\sigma \)) \( \in \text{Plays}(A \to B) \) and play(\( \rho_\tau \)) \( \in \text{Plays}(B \to C) \). This is the crux from which all conditions follow. See details in Appendix C. \( \square \)

5.2 The Unfolding of a Petri Strategy

It is well-known that standard Petri nets unfold to event structures [Hayman and Winskel 2008b; Nielsen et al. 1981]. Usually, unfolding is an elaborate inductive unwinding process, yielding first an occurrence net which is then converted to an event structure. Here instead we leverage the presence of colours and in particular our strong safety condition to give a much more direct definition.

A strategy is a global object, putting together all possible executions with explicit causal and branching information. In this paper, we approach the construction of a strategy more locally: we perform a causal analysis of individual runs, obtaining a structure called a rigid family. We then apply an already existing construction to get an event structure from this rigid family.

5.2.1 On rigid families. 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 \( \rho = (|\rho|, \leq_\rho) \) finite partial orders. We say that \( q \) is rigidly included in \( \rho \), written \( q \hookrightarrow \rho \), if \( |q| \subseteq |\rho| \), and that that inclusion: (1) preserves down-closed sets, i.e. \( C(q) \subseteq C(\rho) \); and (2) preserves causality, i.e. for all \( e, e' \in |q| \), if \( e \leq_q e' \) then \( e \leq_\rho e' \) as well.

**Definition 5.6.** A rigid family \( F \) is a non-empty set of finite partial orders which is:

- rigid-closed: if \( \rho \in F \) and \( q \hookrightarrow \rho \), then \( q \in F \),
- binary-compatible: if \( X \subseteq_F F \), then \( X \uparrow \) iff for all \( q, \rho \in X, \{q, \rho\} \uparrow \).

where \( X \uparrow \) means that there is \( r \in F \) 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 \( F \) collects causal executions. Particularly interesting are the primes of \( F \), i.e. those \( q \in F \) with a top element \( \text{top}(q) = e \); 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 5.7.** For \( F \) a rigid family, the data \( \Pr(F) = (|\Pr(F)|, \leq_{\Pr(F)}, \#_{\Pr(F)}) \) defined by:

\[
|\Pr(F)| = \{ q \in F \mid q \text{ prime} \}
\]

\[
q \leq_{\Pr(F)} \rho \iff q \subseteq \rho
\]

\[
\neg(q \#_{\Pr(F)} \rho) \iff \{q, \rho\} \uparrow,
\]

is an event structure with \( \chi_F : C(\Pr(F)) \cong F \) an order-isomorphism.

The proof is routine – \( \chi_F \) takes \( x \in C(\Pr(F)) \) to its sup \( \vee x \in F \) obtained as the (necessarily) compatible union of all partial orders in \( x \), while \( \chi_F^{-1} \) takes \( q \in F \) to the set of primes below \( q \).
5.2.2 Causal analysis of runs. Now, fix a Petri strategy $\sigma : A \vdash B$. From $\sigma$ we shall extract a rigid family with one partial order for each run generating a valid play. Our first definition is:

**Definition 5.8.** Consider $\rho : \emptyset \longrightarrow_\sigma \gamma$ a valid run on $A \vdash B$, i.e. s.t. play($\rho$) $\in$ Plays$(A \vdash B)$. Write $IT_\rho$ the set of $t \in IT_\sigma$ s.t. $t \not\subseteq \gamma$ appears in $\rho$ for some $\gamma$. We set binary relations on $IT_\rho$:

- $t \prec_A t' \iff t : a \rightarrow_\beta t' : a' \rightarrow_\beta' \beta', a \rightarrow_A a'$,
- $t \prec_\sigma t' \iff \text{new}(t) \cap \text{pre}(t') \neq \emptyset \lor \text{post}(t) \cap \text{eat}(t') \neq \emptyset$

Then, we set $\prec_\rho = \prec_A \cup \prec_\sigma$.

It is obvious that $\prec_\rho$ is acyclic, because if $t \prec_\rho t'$, by definition $t$ must appear before $t'$ in $\rho$. Hence its reflexive transitive closure is a partial order, yielding a poset $\mathcal{T}(\rho) = (IT_\rho, \preceq_\rho)$.

The poset $\mathcal{T}(\rho)$ is generated by two kinds of basic causal dependencies. Firstly, $\prec_A$ imports the “static” immediate causal links enforced by the game. Secondly, $\prec_\sigma$ carries the “dynamic” immediate causal links imposed by the token game itself: $t \prec_\sigma t'$ if $t'$ expects some tokils produced by $t$; or if $t$ requires the presence of some tokils later destroyed by $t'$.

We show that the set of all $\mathcal{T}(\rho)$ forms a rigid family – this rests on a few observations. First:

**Lemma 5.9.** Consider $\rho, \rho'$ two valid runs on $\sigma$. If $IT_\rho = IT_{\rho'}$, then $\mathcal{T}(\rho) = \mathcal{T}(\rho')$.

This is immediate since Definition 5.8 does not depend on the order of transitions in $\rho$.

This lemma draws interest to the sets of itransitions arising from valid runs. If $x \subseteq_f IT_\sigma$, $x$ is a history of $\sigma$, written $x \in \text{Hist}(\sigma)$, if there is $\rho : \emptyset \longrightarrow_\sigma \alpha$ valid s.t. $x = IT_\rho$ (note the change of fonts, to distinguish $x$ from a configuration). By Lemma 5.9, Definition 5.8 yields a poset $\mathcal{T}(x) = (x, \preceq_x)$ for all $x \in \text{Hist}(\sigma)$ – as $\preceq_x$ is fully induced by $x$, we also refer to $\mathcal{T}(x)$ as the history.

**Proposition 5.10.** The set comprising all $\mathcal{T}(x)$ for $x \in \text{Hist}(\sigma)$, is a rigid family written $\mathcal{T}(\sigma)$. Moreover, $\mathcal{T}(\sigma)$ (ordered by rigid inclusion) is order-isomorphic to $\text{Hist}(\sigma)$ (ordered by inclusion).

See Appendix D.1. Propositions 5.7 and 5.10 give $\text{Pr}(\mathcal{T}(\sigma))$, with $\mathcal{C}(\text{Pr}(\mathcal{T}(\sigma))) \equiv \text{Hist}(\sigma)$.

5.2.3 Construction of a strategy. The above gives us an event structure $\text{Pr}(\mathcal{T}(\sigma))$ but not quite the right one for a strategy: its events correspond to all itransitions, but that includes neutral itransitions not accounted for by strategies. To remove them, we use projection:

**Lemma 5.11.** Consider $E$ an event structure, and $V \subseteq |E|$ any set of events.

Then the projection $E \downarrow V$ is an event structure, with components: events, $|E \downarrow V| = V$; causality, $e \leq_{E \downarrow V} e' \iff e \leq_E e'$; conflict, $e \#_{E \downarrow V} e' \iff e \#_E e'$. Moreover, we have an order-isomorphism $\mathcal{C}(E \downarrow V) \cong \mathcal{C}^V(E)$ where $\mathcal{C}^V(E)$ is the maximally visible configurations, i.e. the $x \in \mathcal{C}(E)$ with maximal events in $V$.

**Proof.** Direct; the order-iso sends $x \in \mathcal{C}(E \downarrow V)$ to $[x]_E = \{e' \in |E| \mid \exists e \in x, e' \leq_E e\}$. \hfill $\Box$

Say $x \in \mathcal{T}(\sigma)$ is visible if its top element is a visible itransition written $t : a \rightarrow_m \beta$ – in that case we write $t = \text{top}(x)$ (and recall $\partial_{\sigma}(t) = m$). We write $\mathcal{V}_\sigma$ the set of visible $x \in \mathcal{T}(\sigma)$.

We are finally in position to unfold a Petri strategy $\sigma : A \vdash B$:

**Proposition 5.12.** The event structure $\mathcal{U}(\sigma) = \text{Pr}(\mathcal{T}(\sigma)) \downarrow \mathcal{V}_\sigma$, equipped with the display map

$$
\partial_{\mathcal{U}(\sigma)} : |\mathcal{U}(\sigma)| \rightarrow |A \vdash B| \quad q \mapsto \partial_{\sigma}(\text{top}(q))
$$

is a strategy in the sense of Definition 4.6. Moreover, $\mathcal{U}(\sigma)$ is negative if $\sigma$ is.
We show unfolding preserves the interpretation of IPA
\[ K_\sigma : \mathcal{C}(\mathcal{U}(\sigma)) \cong \mathcal{T}^V(\sigma) \]  
(4)
an order-iso with \( \mathcal{T}^V(\sigma) \) the set of histories whose maximal elements are visible. It sends \( x \in \mathcal{C}(\mathcal{U}(\sigma)) \) to \( \forall x \in \mathcal{T}^V(\sigma) \), and its inverse sends \( q \in \mathcal{T}^V(\sigma) \) to \( \{ p \leftrightarrow q \mid p \in \mathcal{F}_\sigma \} \) prime.

Rule-abiding. For \( x \in \mathcal{C}(\mathcal{U}(\sigma)) \), \( \partial_{\mathcal{U}(\sigma)}(x) \) is easily characterized as the set of labels of visible itransitions in \( K_\sigma(x) \) – a configuration of \( A \rhd B \), since histories originate in valid runs.

Locally injective. Likewise, through \( K_\sigma \) this boils down to the fact that no two visible itransitions of \( K_\sigma(x) \) may have the same label, as that would contradict condition non-repetitive of plays.

Courteous, receptive, negative. See Appendix D.1 for details.

Finally, to prove its functoriality, the following straightforward lemma will be helpful – where \( \mathcal{T}^+(\sigma) \), the \( + \)-covered histories of \( \sigma \), are those histories whose maximal transitions are positive.

**Lemma 5.13.** The order-isomorphism (4) of Lemma 5.12 specializes to \( \mathcal{K}_\sigma^+ : \mathcal{C}(\mathcal{U}(\sigma)) \cong \mathcal{T}^+(\sigma) \) s.t. for all \( x \in \mathcal{C}(\mathcal{U}(\sigma)) \), we have \( \partial_{\mathcal{U}(\sigma)}(x) = \partial_{\mathcal{K}_\sigma^+}(x) \) the labels of visible transitions in \( K_\sigma^+(x) \).

### 5.3 Unfolding as an IPA-functor

We show unfolding preserves the interpretation of IPA – the main challenge is composition.

#### 5.3.1 Preservation of composition

Recall from Proposition 4.11 that given strategies \( \sigma : A \rhd B \) and \( \tau : B \rhd C \), \( + \)-covered configurations of \( \sigma \circ \tau \) correspond to causally compatible pairs of \( x^\sigma \in \mathcal{C}_+(\sigma) \) and \( x^\tau \in \mathcal{C}_+(\tau) \) such that \( x^\sigma \) and \( x^\tau \) match on \( B \) (through their respective display maps), and such that the induced synchronization is secured, i.e. deadlock-free. The crux of preservation of composition is a corresponding statement for histories of the composition of Petri strategies.

For Petri strategies \( \sigma : A \rhd B \) and \( \tau : B \rhd C \), we repeat the constructions of Section 4.2.3. Given \( x^\sigma \in \mathcal{T}^+(\sigma) \), write \( \partial_{\sigma}(x^\sigma) = x^\sigma_A \rhd x^\sigma_B \). Histories \( x^\sigma \in \mathcal{T}^+(\sigma) \) and \( x^\tau \in \mathcal{T}^+(\tau) \) are matching if \( x^\sigma_B = x^\tau_B \); so it is unambiguous to write \( \partial_{\sigma}(x^\sigma) = x^\sigma_A \rhd x^\sigma_B \) and \( \partial_{\tau}(x^\tau) = x^\tau_B \rhd x^\tau_C \). We set

\[
\partial_{\sigma} : x^\sigma \rightarrow x^\sigma_A \parallel x^\sigma_B \parallel x^\sigma_C \quad \text{and} \quad \partial_{\tau} : x^\tau \rightarrow x^\tau_A \parallel x^\tau_B \parallel x^\tau_C
\]

and undefined otherwise (i.e. for neutral itransitions). We define:

**Definition 5.14.** Consider \( x^\sigma \in \mathcal{T}^+(\sigma) \) and \( x^\tau \in \mathcal{T}^+(\tau) \) matching.

They are causally compatible if the relation \( \preceq = \preceq_{\sigma} \sqcup \preceq_{\tau} \) on \( x^\sigma_A \parallel x^\tau_B \parallel x^\tau_C \) set with:

\[
\begin{align*}
\partial_{\sigma}(t) & \preceq_{\sigma} \partial_{\sigma}(t') & \text{for } t \preceq_{x^\sigma} t' \\
\partial_{\tau}(t) & \preceq_{\tau} \partial_{\tau}(t') & \text{for } t \preceq_{x^\tau} t'
\end{align*}
\]

is acyclic. We also say that the pair \( x^\sigma, x^\tau \) is secured.

This is by design almost a copy of Definition 4.10. As for strategies, we have (see Appendix D.2):

**Proposition 5.15.** Consider \( \sigma : A \rhd B \) and \( \tau : B \rhd C \) Petri strategies. Then, there is an order-iso:

\[ (- \circ -) : \{ (x^\sigma, x^\tau) \in \mathcal{T}^+(\tau) \times \mathcal{T}^+(\sigma) \mid \text{causally compatible} \} \cong \mathcal{T}^+(\tau \circ \sigma) \]

such that for \( x^\sigma \in \mathcal{T}^+(\sigma), x^\tau \in \mathcal{T}^+(\tau) \) causally compatible, \( \partial_{\tau \circ \sigma}(x^\tau \circ x^\sigma) = x^\sigma_A \rhd x^\tau_B \).

**Proposition 5.16.** For \( \sigma : A \rhd B \) and \( \tau : B \rhd C \) Petri strategies, \( \mathcal{U}(\tau \circ \sigma) \cong \mathcal{U}(\tau) \circ \mathcal{U}(\sigma) \).
Proof. We may deduce preservation of composition simply by manipulating isos:

\[
\mathcal{C}^+ (\mathcal{U}(\tau \circ \sigma)) \cong \mathcal{T}^+ (\tau \circ \sigma) \\
\cong \{(x^\sigma, x^\tau) \in \mathcal{T}^+ (\sigma) \times \mathcal{T}^+ (\tau) \mid \text{causally compatible}\} \\
\cong \{(x^{\mathcal{U}(\sigma)}, x^{\mathcal{U}(\tau)}) \in \mathcal{C}^+ (\mathcal{U}(\sigma)) \times \mathcal{C}^+ (\mathcal{U}(\tau)) \mid \text{causally compatible}\} \\
\cong \mathcal{C}^+ (\mathcal{U}(\tau) \circ \mathcal{U}(\sigma))
\]

via Lemma 5.13, Proposition 5.15, and Lemma 5.13 – preservation of causal compatibility follows directly from the order-isomorphism. All these steps preserve displays to the game. By Proposition 4.11, it follows that \(\mathcal{U}(\tau \circ \sigma) \cong \mathcal{U}(\tau) \circ \mathcal{U}(\sigma)\) as required. \(\square\)

5.3.2 Wrapping up. For other operations (tensor, currying, promotion), their preservation by the unfolding is much more direct; primitives are shown to unfold to the adequate strategy via a characterization of their +-covered histories – details appear in Appendices D.3 and D.4. Altogether:

**Theorem 5.17.** Unfolding yields an IPA-functor \(\mathcal{U} : \text{PStrat} \rightarrow \text{Strat}\).

Following [Castellan and Clairambault 2020; Castellan et al. 2019], a formal description of the causal behaviour of any \(\Gamma \vdash M : A\) as an event structure can be obtained by its interpretation \([M]_{\text{Strat}} \in \text{Strat}([[\Gamma]], [[A]])\). Theorem 5.17 shows that it can also be obtained more directly by purely operational means, through the unfolding of the abstract machine initialized on \(M\), i.e. \([M]_{\text{PStrat}}\).

Finally, Theorem 3.20 follows from Theorems 4.12 and 5.17, Proposition 5.5, and Lemma 2.3.

6 IMPLEMENTATION

We implemented our translation from IPA to Petri strategies in an interactive web application available here with some documentation here. The interface lets the user enter a IPA program (or choose from a list of examples) and then displays the Petri strategy. The user can then simulate runs by firing available transitions and see the tokens flow through the net. The implementation only displays the data component of a token, but the exponential stack can be obtained by hovering the mouse above the token. Likewise, transition rules are displayed by hovering the mouse above transitions, though written in an undocumented stack language.

Our translation, following the categorical semantics, tends to generate large Petri strategies. To keep the nets at a reasonable size, we have implemented several optimisations. The first optimisation is to eliminate locations and transitions that are unreachable from a negative transition, or that never reach a positive transition. Such "dead code" can occur during the 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 on the right, where we merge \(a\) and \(b^0\) and remove the location \(\ell\). The new transition \(c\) has \(\text{pre}(c) = \text{pre}(a)\) and \(\text{post}(c) = \text{post}(a) \setminus \{\ell\} \cup \text{post}(c)\) and transition function: \(\delta(c)(\alpha) = \delta(a)(\alpha) \setminus \{t\} \cup \delta(b)(\{t\})\), where \(t\) denotes the tokil at \(\ell\) in \(\delta(a)(\alpha)\).

There are minor inconsistencies between examples given in the paper and the implementation: firstly, optimization 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 6 with this). Secondly, in some examples we simplify the exponential signatures used by a strategy rather than using directly those arising from the interpretation (compare e.g. Figure 20 with this).
7 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 languages.

Firstly, our translation cleanly 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, rather than getting entirely rid of the infinite behaviour, handling it symbolically or over-approximating via abstract interpretation – perhaps offering a new truly concurrent basis for the static analysis of higher-order concurrent programs.

Secondly, GoI has been proposed in the past as an approach for the compilation of functional languages, suggested in particular by connections with Lamping’s optimal reduction for the \(\lambda\)-calculus [Asperti et al. 1994; Gonthier et al. 1992; Lamping 1990]. Perhaps this work suggests a way to reap the rewards of this connection beyond purely functional languages.

Thirdly, another merit in using Petri strategies as an intermediate language is that the unfolding theorem (Theorem 5.17) provides clean and formal semantics which may serve as guide or semantic justification for optimizations, including those that rely on causal information.

IPA is of course hardly a realistic programming language, but we do not foresee any fundamental obstacle in generalizing this approach. In earlier work on GoI, handling call-by-value has sometimes proved a challenge, e.g. requiring repeating computation [Dal Lago et al. 2015], which breaks the expected cost model – this motivated the recent dynamic GoI of [Muroya and Ghica 2019]. In contrast, Petri strategies support a controlled evaluation order using memoization without repetition, as already illustrated in the let binding included in our version of IPA.

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


Basic red. for PCF

\[(\lambda x^A. M)\ N \rightsquigarrow M[N/x]\]
\[\text{if } b\ N_1\ N_2 \rightsquigarrow N_b\]
\[Y\ M \rightsquigarrow (M\ Y\ M)\]

let \(x = v\) in \(M\) \(\rightsquigarrow M[v/x]\)
\[f(v_1, v_2) \rightsquigarrow v\]
\[(\text{if } f(v_1, v_2) = v)\]

Basic reductions for references and semaphores

\[\text{newref } x\ in\ v \rightsquigarrow v\]
\[\text{newsem } x\ in\ v \rightsquigarrow v\]

Stateful context rules

\[
\begin{array}{c}
MN \rightsquigarrow M' N \\
\text{if } M\ N_1\ N_2 \rightsquigarrow \text{if } M'\ N_1\ N_2 \\
!M \rightsquigarrow !M' \\
M: N \rightsquigarrow M: N'
\end{array}
\]

\[
\begin{array}{c}
\text{grab}(M) \rightsquigarrow \text{grab}(M') \\
N \rightsquigarrow N' \\
M_1 \rightsquigarrow M_1'
\end{array}
\]

let \(x = N\) in \(M\) \(\rightsquigarrow\)
let \(x = N'\) in \(M\)

\[
\begin{array}{c}
f(M_1, M_2) \rightsquigarrow f(M_1', M_2) \\
f(M_1, M_2) \rightsquigarrow f(M_1, M_2')
\end{array}
\]

Stateful context rules

\[
\begin{array}{c}
\langle M[x], s \cup \{ \ell \mapsto n \} \rangle \rightsquigarrow \langle M'[x], s' \cup \{ \ell \mapsto n' \} \rangle \\
\text{(newref } x:=n\ \text{in } M, s) \rightsquigarrow \text{(newref } x:=n'\ \text{in } M', s')\]
\[\text{fresh}\]
\[
\begin{array}{c}
\langle M[x], s \cup \{ \ell \mapsto n \} \rangle \rightsquigarrow \langle M'[x], s' \cup \{ \ell \mapsto n' \} \rangle \\
\text{(newsem } x:=n\ \text{in } M, s) \rightsquigarrow \text{(newsem } x:=n'\ \text{in } M', s')\]
\[\text{fresh}\]
\[
M_1 \rightsquigarrow M_1' \\
\text{fresh}
\]

Fig. 24. Operational semantics of IPA

A OPERATIONAL SEMANTICS

We include in Figure 24 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 Strat

B.1 Categorical structure

Section 4.2 already contains the data of Strat, and composition. It remains to define:

B.1.1 Copycat. Copycat may be defined on any game, but it is slightly simpler on arenas:

Definition B.1. For each arena \(A\), the \textbf{copycat strategy} \(\epsilon_A : A \rightarrow A\) is defined as having:

\[
\begin{align*}
|\epsilon_A| &= |A \rightarrow A| \\
\epsilon_A(m) &= m \\
\epsilon(a) &\leq \epsilon_A \epsilon'(a) \iff a \leq_A a' \text{; or } a = a' \text{, } \text{pol}(\epsilon(a)) = - \text{ and } \text{pol}(\epsilon'(a)) = + \\
\epsilon(a) &\# \epsilon_A \epsilon'(a) \iff a \#_A a',
\end{align*}
\]

where \(\epsilon, \epsilon' \in \{\ell, \kappa\}\).
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 B.2.** Consider $A$ any arena. Then, we have $C^+(a_A) = \{x_A \vdash x_A \in C(A) \mid x_A \in C(A)\}$.

From Lemma 4.9 this characterizes $a_A$ uniquely up to iso, just like Proposition 4.11 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 B.3.** 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.

### B.2 Strat as an IPA-structure: Operations

We detail the different operations involved in the IPA-structure.

#### B.2.1 Tensor.

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

**Definition B.4.** We define $\sigma_1 \otimes \sigma_2 \in \text{Strat}(A_1 \otimes A_2, B_1 \otimes B_2)$ with:

\begin{align*}
|\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}(r(m)) &= \partial_{\sigma_1}(m)
\end{align*}

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

**Proposition B.5.** The strategy $\sigma_1 \otimes \sigma_2 \in \text{Strat}(A_1 \otimes A_2, B_1 \otimes B_2)$ satisfies:

$(- \otimes -) : C^+(\sigma_1) \times C^+(\sigma_2) \rightarrow C^+(\sigma_1 \otimes \sigma_2)$

such that $\partial_{\sigma_1 \otimes \sigma_2}(x_{A_1} \otimes x_{A_2}) = (x_{A_1} \otimes x_{A_2} \vdash (x_{B_1} \otimes x_{B_2}))$ for all $x_{A_1} \otimes x_{A_2} \in C^+(\sigma_1 \otimes \sigma_2)$.

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 4.9. $\square$

#### B.2.2 Currying.

As for Petri structures, we start with renaming.

**Definition B.6.** Consider $\sigma$ a strategy on game $A$, and $f : |A| \rightarrow |B|$.

Then, we define the renaming to be as $\sigma$ 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 B.7.** We say that $f : |A| \rightarrow |B|$ is valid if it is a map of es, additionally satisfying hypotheses receptive and courteous from Definition 4.6.

If $\sigma : A$ and $f$ is valid, then $\sigma[f]$ is a strategy on $B$.

**Proof.** Straightforward. $\square$

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

$\Lambda_{x ; A}^{\Gamma, A} : |! \& [\Gamma, x : A, \Delta] \vdash O| \rightarrow |! \& [\Gamma, \Delta] \vdash !A \rightarrow O|$

$(m, s, d) \mapsto (\Lambda^s(m), s, d)$
using $\Lambda_x$ from Definition 3.15, 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 B.8.** Consider $\sigma \in \text{Strat}(\Gamma & [\Gamma, x : A, \Lambda], O)$.
Then, there exists a unique $\Lambda(\sigma) \in \text{Strat}(\Gamma & [\Gamma, x : A, \Lambda] \vdash O)$ such that

$$\phi : \mathcal{C}^+(\sigma) \equiv \mathcal{C}^+(\Lambda(\sigma))$$

and satisfying that $\partial_{\Lambda(\sigma)}(\phi(x^\sigma)) = \Lambda_{x,A,O}^\Gamma(\partial_{\sigma}(x^\sigma))$ for all $x^\sigma \in \mathcal{C}^+(\sigma)$.

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

Uniqueness. Direct from Lemma 4.9.

**B.2.3 Promotion.** Next we define the promotion of $\sigma \in \text{Strat}([A,B])$.

First, for any arena $A$, we define the function

$$\text{dig}_{\Lambda_A} : \text{!}A \rightarrow \text{!}A$$

yielding a map of event structures. If $\sigma$ is a strategy, we write $\mathcal{C}^{+,\theta}(\sigma)$ the set of $\tau$-covered, non-empty configurations of $\sigma$. Finally, for $A$ a set we write $\text{Fam}(X)$ 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 B.9.** There is a strategy $\sigma^\dagger \in \text{Strat}([A,B])$, unique up to iso, such that there is

$$[\text{!}] : \text{Fam}(\mathcal{C}^{+,\theta}(\sigma)) \equiv \mathcal{C}^+(\sigma^\dagger)$$

satisfying that for all $(x^e)_{e \in E} \in \text{Fam}(\mathcal{C}^{+,\theta}(\sigma))$, we have

$$\partial_{\sigma^\dagger}[(x^e)_{e \in E}] = \text{dig} \left( \bigcup_{e \in E} e : x^e_A \bigcup_{e \in E} e : x^e_B \right)$$

writing $\partial_{\sigma}(x^e) = x^e_A \vdash x^e_B$ for all $e \in E$.

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

Uniqueness. Direct from Lemma 4.9.

**B.3 Strat as an IPA-Structure: Primitives**

**B.3.1 Copycat strategies.** We first address the three primitives arising as copycat-like strategies: variable, evaluation, and contraction.

**Definition B.10.** Consider $\Gamma, x : A, \Lambda$ a semantic context. Then we set $\text{Var}^{\Gamma,\Lambda}_{x,A}$ as $\omega_A[\text{Var}_x]$ where

$$\text{Var}_x : (A \vdash A) \rightarrow (\text{!} & [\Gamma, x : A, \Lambda] \vdash A)$$

$$\begin{align*}
\phi_m, s, d & \mapsto (\phi_m, s, d) \\
\xi m, s & \mapsto (\xi, s, m, \cdot : s, d) \\
\ell m, s & \mapsto (\ell, s, m, \cdot : s, d)
\end{align*}$$

Likewise, the evaluation morphism is simply by renaming.

**Definition B.11.** Consider $A, O$ arenas with $O$ well-opened.
Then we set $\text{ev}_{A,O} = \omega_{A\rightarrow O}[\Omega]$ where $\Omega$ is that of Definition 3.18 canonically extended to moves.

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

Definition B.12. Consider an arena. Then we set \( c_A = \omega_{A \otimes A} \mid A \in \text{Strat}(\! \! / A, ! A \otimes ! A) \) where

\[
\begin{align*}
\ell \triangleright (\ell, \ell' \cdot m, e : s, d) & \mapsto (\ell, m, (\ell' \cdot e) : s, d) \\
\ell \triangleright (\ell, \nu m, e : s, d) & \mapsto (\ell, m, (\nu m) : s, d) \\
(\nu m, s, d) & \mapsto (\nu m, s, d)
\end{align*}
\]

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

Proposition B.13. For any arena \( A, \mathcal{C}^*(c_A) = \{ (\ell(x_A) \cup \nu(y_A) \vdash x_A \otimes y_A \mid x_A, y_A \in \mathcal{C}(\! / A) \} \).

Proof. Immediate by Lemma B.2 and definition. \( \square \)

B.3.2 Constants, conditional, queries. The strategies are displayed in Figure 25. 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.

B.3.3 Let. We illustrate the strategy let in Figure 26. Note that there is a similar call to \( ! X \) for all exponential signature \( e \in E \).

B.3.4 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 \( F \mapsto (f : O \rightarrow O \vdash F f) \).

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).

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

\[
(\cdot) := \ell \triangleright \\
(e_{n+1}, e_n, \ldots, e_1) := \nu \langle e_{n+1}, (e_n, \ldots, e_1) \rangle,
\]

yielding \( (e_n, \ldots, e_1) \in E \) for each \( e_1, \ldots, e_n \in E \). With this convention, we draw the recursion combinator for \( U \) in Figure 27. Again the representation is symbolic, with similar branches for all \( e_1, \ldots, e_{n+1} \in E \). 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 27. The rest of the strategy is simple copycat behaviour; which may be simply described via the following direct characterization of the \( += \)-covered configurations of \( Y_O \):

Proposition B.14. For \( O \) a well-opened arena, the strategy \( Y_O \) has events a subset \( |Y_O| \subseteq |(! O \rightarrow O) \vdash O| \), and +=-covered configurations the compatible unions of configurations of the form

\[
\emptyset \rightarrow (\cdot) : x \vdash x
\]

\[
(e_{n+1} :: (e_n, \ldots, e_1) : x) \sim (e_{n+1}, (e_n, \ldots, e_1) : x) \vdash \emptyset,
\]

for \( n \in \mathbb{N}, e_1, \ldots, e_{n+1} \in E, x \in \mathcal{C}(O) \); compatible means that the union is in \( \mathcal{C}(\! / (\! / O \rightarrow O) \vdash O) \).

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 \( (\cdot) : E^* \rightarrow E \).

Lemma B.15. There is an order-isomorphism between \( \mathcal{C}^*(Y_O) \) and tuples \( (J \subseteq E^+, z \in \mathcal{C}(O), (x_z \in \mathcal{C}^J(O))_{z \in J}) \) such that \( J \) is suffix-closed, and empty if \( z \) is.\(^1\) The isomorphism sends \( (J, z, (x_z)) \) to

\[
(\emptyset \rightarrow (\cdot) : z) \cup (e :: (s) : x_{e \cdot s} \sim (e :: s) : x_{e \cdot s} \in E)_{z \in J} \vdash z.
\]

\(^1\)A stack in \( J \) represents a call stack: a list \( e_n \cdot e_{n-1} \cdot \ldots \cdot e_1 \) represents the calls made to the argument: \( e_1 \) is the first call made in the execution, and so on.
B.3.5 New reference. Next, we introduce the strategy for reference initialization.

The intuition is that newref\(X\) : !V \to X \vdash X applies its argument to cell : !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 cell\(n\)\(B\), defined as:

\[
\text{cell}^E_n = (w, Q, [e], [\cdot]) - (w, Q, [e], n^+ \cdot \text{cell}^E_{n+1}(e)) \mid (w, Q, [e], k)^- \cdot (w, A, [\cdot])^+ \cdot \text{cell}^E_k(e),
\]

with \(e \notin E\). Intuitively, words in cell\(n\)\(B\) 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\(n\)\(B\) are plays on !V in the sense of Definition 5.1. We may then define:

\[\text{deref} : V \vdash N\]
\[\text{grab} : S \vdash U\]
\[\text{release} : S \vdash U\]

Fig. 25. Basic strategies for IPA primitives
We define a strategy $\text{cell}$:

$$
\begin{align*}
|\text{cell}| & = \text{cell}_0 \\
\leq_{\text{cell}} & = \subseteq \\
\pi_{\text{cell}} & = s \#\text{cell} \overset{\equiv}{\iff} \neg(s \subseteq s' \lor s' \subseteq s) \\
\delta_{\text{cell}}(sa) & = a,
\end{align*}
$$

where $\subseteq$ 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 B.17. We define a strategy $\text{cell} : !V$ as: $\text{cell} = \omega_V \circ \text{precell} : !V$.

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 15g.

To obtain the strategy for $\text{newref}_X$, we must add a copycat behaviour on $X$:

Definition B.18. We define a strategy $\text{newref}_X : !V \multimap X \vdash X$ with components:

$$
\begin{align*}
|\text{newref}_X| & = |\text{cell}| + |\omega_X| \\
\leq_{\text{newref}_X} & = (\leq_{\text{cell}} + \leq_{\omega_X}) \\
& \cup \{ (r((\ell, \omega_X, [\cdot, \cdot]), \ell(s)) \mid m \in |\text{cell}|) \\
& \cup \{ (r((\ell, \omega_X, [\cdot, \cdot]), \ell(s)) \mid m \in |\text{cell}|) \\
\pi_{\text{newref}_X} & = \#\text{cell} + \#\omega_X \\
\delta_{\text{newref}_X}(\ell(m)) & = \ell_{\ell}, \delta_{\text{cell}}(m) \\
\delta_{\text{newref}_X}(r(\ell, m)) & = \ell_{r}, m \\
\delta_{\text{newref}_X}(r(\omega_X, m)) & = r_{\omega_X}, m
\end{align*}
$$

So cell is plugged after the first Player move of $\omega_X$. In other words, $\text{newref}_X$ 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:

\[\text{cell} \mid |\omega_X| \leq_{\text{cell}} |\text{newref}_X| \]

where $\leq_{\text{cell}}$ is the prefix ordering.
The Concurrent Games Abstract Machine

Proposition B.19. There is an order-isomorphism:

\[ \langle \_, \_ \rangle : \langle \cdot \rangle (\text{precell}) \times \langle \cdot \rangle (X) \cong \langle \cdot \rangle (\text{newref}) \]

such that \( \partial_{\text{newref}} (\langle s, x \rangle) = |s| \rightarrow x \vdash x \in \langle \cdot \rangle (X \rightarrow X \vdash X) \).

Proof. By Definition B.18, the characterization of confs. of composition, and Lemma B.2.

Above, we implicitly use the one-to-one correspondence between \( \langle \cdot \rangle (\text{precell}) \) and even-length words in \( \text{cell}_0 \); and if \( s \) is an even length word in \( \text{cell}_0 \), then \( |s| \in \langle \cdot \rangle (IV) \) is its set of events.

In a \(+\)-covered configuration of newref, all read and write requests have been successfully handled. Proposition B.19 shows that besides the almost independent copycat behaviour on \( X \), \(+\)-covered configurations of newref exactly correspond to some sequential ordering of these requests.

B.3.6 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:

\[ \begin{align*}
\text{lock}^E_0 &= (g_s Q, [e], \bullet)^- \cdot (g_s A, [e], \checkmark)^+ \cdot \text{lock}^E_0 (e) \quad e \notin E \\
\text{lock}^E_n &= (g_s Q, [e], \bullet)^- \cdot (g_s A, [e], \checkmark)^+ \cdot \text{lock}^E_0 (e) \quad e \notin E, n > 0
\end{align*} \]

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 B.20. We define a prestrategy \( \text{prelock} : !S \) with components:

\[ \begin{align*}
|\text{prelock}| &= |\text{lock}| + |\text{ax}_X| \\
\leq_{\text{prelock}} &= \subseteq \\
\#_{\text{prelock}} s' &\iff \neg (s \subseteq s' \lor s' \subseteq s) \\
\partial_{\text{prelock}} (s) &= a.
\end{align*} \]

Definition B.21. We define a strategy \( \text{lock} : !S \rightarrow X \) as \( \omega_S \circ \text{prelock} : !S \).

Definition B.22. We define a strategy \( \text{newsem}_X : !S \rightarrow X \rightarrow X \) with components:

\[ \begin{align*}
|\text{newsem}_X| &= |\text{lock}| + |\text{ax}_X| \\
\leq_{\text{newref}} &= (\leq_{\text{lock}} + \leq_{\text{ax}_X}) \\
\#_{\text{newsem}_X} &= \#_{\text{lock}} + \#_{\text{ax}_X} \\
\partial_{\text{newsem}_X} (\ell (m)) &= \ell, \ell, \partial_{\text{lock}} (m) \\
\partial_{\text{newsem}_X} (\tau (\ell, m)) &= \ell, \tau, m \\
\partial_{\text{newsem}_X} (\tau (\tau, m)) &= \tau, m
\end{align*} \]

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

Proposition B.23. There is an order-isomorphism:

\[ \langle \_, \_ \rangle : \langle \cdot \rangle (\text{prelock}) \times \langle \cdot \rangle (X) \cong \langle \cdot \rangle (\text{newsem}) \]

such that \( \partial_{\text{newsem}} (\langle s, x \rangle) = |s| \rightarrow x \vdash x \in \langle \cdot \rangle (X \rightarrow X \vdash X) \).

This concludes the description of the IPA-structure of Strat.
B.4 Complement: on symmetry

B.4.1 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 B.24. A event structure with symmetry is $E = (|E|, \leq_E, \#_E, \mathcal{S}(E))$ with $\mathcal{S}(E)$ 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 $\mathcal{S}(A)$ when $\theta$ 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 $\varphi : \sigma \approx \tau$ is an invertible map of event structure such that the triangle

commutes up to symmetry, defined as $\{ (\partial_\sigma s, \partial_\tau (\varphi(s))) \mid s \in x \} \in \mathcal{S}(A)$ for all $x \in \mathcal{C}^{\mathcal{S}}(\sigma)$. 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 $\mathcal{S}(\sigma)$ ensures that if Opponent changes their copy indices, $\sigma$ may change its copy indices accordingly, but not more. This makes $\approx$ a congruence, and strategies up to $\approx$ satisfy all the required equational laws to model higher-order languages. This issue is discussed at length in [Castellan et al. 2019].

B.4.2 Removing symmetry. The model developed in [Castellan and Clairambault 2020] is a structure $\text{StratSym}$ with two equivalences $\cong$ (standard isomorphism) and $\approx$ (weak isomorphism). Both are preserved by all constructions, but the laws of Seely categories are satisfied with respect to $\approx$ only.

Now, the first observation is that though $\approx$ is crucial in establishing adequacy for $\text{StratSym}$ (for instance, the $\beta$-law in IPA is validated by the interpretation only up to $\approx$), 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 $\approx$, and from [Castellan and Clairambault 2020] we get an IPA-structure $\text{StratSym}/\cong$ 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 $\approx$ is a congruence. As mere uniformity witnesses, they can be safely forgotten once $\approx$ 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 B.25.** There is a symmetry-forgetting IPA-functor $\text{StratSym} / \cong \to \text{Strat}$.  

From this, it follows by Lemma 2.3 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.

C THE IPA-STRUCTURE PStrat

C.1 The Precategory PStrat

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

C.1.1 Composition. Consider $A, B, C$ arenas, $\sigma : A \to B$ and $\tau : B \to C$ Petri strategies.

The idea is simple: as Petri strategies, both $\sigma$ and $\tau$ 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 $\rho$ on $\tau \circ \sigma$ to runs on $\sigma$ and $\tau$, and to an interaction in more familiar game-semantic terms:

**Definition C.1.** Consider the set $\text{MInt} = \ell (\text{Moves}) \cup m (\text{Moves}) \cup r (\text{Moves})$.

An interaction is a sequence $u \in \text{MInt}^*$. We write $\text{Int}$ for the set of all interactions.

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

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

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

$$u_\sigma = u \upharpoonright \ell, u_\tau = u \upharpoonright \ell, r, u_{\tau \circ \sigma} = u \upharpoonright \ell, r$$

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

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

**Lemma C.3.** Consider $\sigma$ and $\tau$ Petri structures.

Then, instantiated transitions of $\tau \circ \sigma$ are exactly as in Figure 28 – 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 29 partial functions $\text{lbl}_\sigma : \text{IT} \tau \circ \sigma \to \text{MInt}, \pi_\sigma : \text{IT} \tau \circ \sigma \to \text{IT} \sigma$ extracting various data from instantiated transitions, following the characterization of instantiated transitions of $\tau \circ \sigma$ given in Figure 28.

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

$$\text{lbl}_\sigma : \text{ITC} \tau \circ \sigma \to \text{MInt} \quad \pi_\sigma : \text{ITC} \tau \circ \sigma \to \text{ITC} \sigma$$

and symmetrically for $\pi_\tau : \text{ITC} \tau \circ \sigma \to \text{ITC} \tau$. Using these, from a run $\rho : \emptyset \to \tau \circ \sigma$ $\alpha$ we extract:

$$\text{Int}(\rho) = \rho \upharpoonright \text{lbl}_\sigma \quad \rho_\sigma = \rho \upharpoonright \pi_\sigma \quad \rho_\tau = \rho \upharpoonright \pi_\tau$$
which allow us to prove the following property:

**Lemma C.4.** Consider $\rho: \emptyset \rightarrow_{\tau \otimes \sigma} \alpha$. Then, $\alpha = \alpha_{\sigma} + \alpha_{\tau}$ and

$$
\rho_{\sigma} : \emptyset \rightarrow \alpha_{\sigma} , \quad \rho_{\tau} : \emptyset \rightarrow \alpha_{\tau}
$$

where $\text{play}(\rho) = \text{Int}(\rho)_{\tau \otimes \sigma}, \text{play}(\rho_{\sigma}) = \text{Int}(\rho)_{\sigma}$ and $\text{play}(\rho_{\tau}) = \text{Int}(\rho)_{\tau}$.

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

**Proof.** A lengthy and grindy induction on $\rho$. For $\rho$ empty this is clear, otherwise we reason by cases on the last instantiated transition in context of $\rho$, following the Figure 28.

Consider first that we have $\rho' = \rho(t^\circ(t^0)|t^\circ(\mu)) \cup \gamma: \emptyset \rightarrow_{\tau \otimes \sigma} \beta$ where $\rho: \emptyset \rightarrow_{\tau \otimes \sigma} \alpha,$

$$
\ell^\circ(t^0)(\ell^\circ(\alpha)) : \ell^\circ(\mu) \rightarrow_{\tau \otimes \sigma} \ell^\circ(\nu), \quad \ell^\circ(t^0)(\ell^\circ(\alpha)) \cup \gamma : \alpha \rightarrow_{\tau \otimes \sigma} \beta
$$

with necessarily $\alpha = \ell^\circ(\mu) \cup \gamma$ and $\beta = \ell^\circ(\nu) \cup \gamma$, and $t^0(\mu) : \mu \rightarrow_{\sigma} \nu$. By IH, $\alpha = \alpha_{\sigma} + \alpha_{\tau}$ with

$$
\rho_{\sigma} : \emptyset \rightarrow_{\sigma} \alpha_{\sigma}, \quad \rho_{\tau} : \emptyset \rightarrow_{\tau} \alpha_{\tau},
$$

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

$$
t^0(\mu) \cup \gamma_{\sigma} : \alpha_{\sigma} \rightarrow_{\sigma} \beta_{\sigma}
$$

writing $\beta_{\sigma} = \nu \cup \gamma_{\sigma}$. Writing $\beta_{\tau} = \alpha_{\tau} = \gamma_{\tau},$ we have

$$
(\rho(t^\circ(t^0)(\ell^\circ(\mu)) \cup \gamma))_{\sigma} = \rho_{\sigma}(t^0(\mu) \cup \gamma_{\sigma}) : \emptyset \rightarrow_{\sigma} \beta_{\sigma}
$$
and $(\rho(\ell^o(t))^o(\mu)) \cup \gamma = \rho_\gamma : \emptyset \rightarrow_\tau \beta_\gamma$. Moreover play($\rho'$) = play($\rho$), Int($\rho'$) = Int($\rho$), play($\rho'_\sigma$) = play($\rho_\sigma$) and play($\rho'_\tau$) = play($\rho_\tau$), making required properties obvious. Finally, Coll($\rho'$) = Coll($\rho$) $\cup$ $\ell^o(v)$, Coll($\rho'_\sigma$) = Coll($\rho_\sigma$) $\cup \nu$ and Coll($\rho'_\tau$) = Coll($\rho_\tau$), so Coll($\rho'$) = Coll($\rho'_\sigma$) $\lor$ Coll($\rho'_\tau$) follows from IH. The case of a neutral transition from $\tau$ is symmetric.

Next, consider that we have $\rho' = \rho(\ell^o(t))^o(\alpha) \cup \gamma : \emptyset \rightarrow_\tau \beta$, where $\rho : \emptyset \rightarrow_\tau \alpha$. $\ell^o(t)^o(\mu) : \ell^o(\mu)$ $\ell^o(t)^o(\alpha) : \emptyset \rightarrow_\tau \beta$, where necessarily $\alpha = \ell^o(\mu) \cup \gamma$ and $\beta = \gamma$, and $t^o(\mu) : \mu \rightarrow_\tau \emptyset$. Now, by IH, $\alpha = \alpha_\sigma + \gamma_\tau$ with $\rho_\sigma : \emptyset \rightarrow_\tau \alpha_\sigma$, $\rho_\tau : \emptyset \rightarrow_\tau \alpha_\tau$, so in particular that entails $\gamma = \gamma_\sigma + \gamma_\tau$ with $\alpha_\sigma = \mu \cup \gamma_\sigma$. Now, we have $t^o(\mu) \cup \gamma_\sigma : \alpha_\sigma \rightarrow_\tau \beta_\sigma$ writing $\beta_\sigma = \gamma_\sigma$. Writing $\beta_\tau = \alpha_\tau = \gamma_\tau$, we have $(\rho(\ell^o(t))^o(\mu)) \cup \gamma) : \emptyset \rightarrow_\tau \beta_\sigma$ and $(\rho(\ell^o(t))^o(\mu)) \cup \gamma) : \emptyset \rightarrow_\tau \beta_\tau$. Moreover, play($\rho'$) = play($\rho$)$, Int($\rho'$) = Int($\rho$)$\ell^o(m)$, play($\rho'_\sigma$) = play($\rho_\sigma$)$\ell(m)$ and play($\rho'_\tau$) = play($\rho_\tau$), making the required properties clear. Finally, Coll($\rho'$) = Coll($\rho$), Coll($\rho'_\sigma$) = Coll($\rho_\sigma$) and Coll($\rho'_\tau$) = Coll($\rho_\tau$), so Coll($\rho'$) = Coll($\rho'_\sigma$) $\lor$ Coll($\rho'_\tau$) follows from IH. The case of a positive transition from $\tau$ is symmetric.

Next, consider that we have $\rho' = \rho(\ell^o(t))^o(\mu)$ $\cup \gamma) : \emptyset \rightarrow_\tau \beta$, where $\rho : \emptyset \rightarrow_\tau \alpha$. $\ell^o(t)^o(\mu) : \ell^o(\mu)$ $\ell^o(t)^o(\alpha) : \emptyset \rightarrow_\tau \beta$, where necessarily, $\alpha = \gamma$ and $\beta = \ell^o(v) \cup \gamma$, and $t^o(s, d) : \emptyset \rightarrow_\tau \emptyset$. By IH, $\alpha = \alpha_\sigma + \gamma_\tau$ with $\rho_\sigma : \emptyset \rightarrow_\tau \alpha_\sigma$, $\rho_\tau : \emptyset \rightarrow_\tau \alpha_\tau$, so that $\gamma = \gamma_\sigma + \gamma_\tau$ with $\gamma_\sigma = \alpha_\sigma \cup \gamma_\tau = \alpha_\tau$. Writing $\beta_\sigma = \gamma_\sigma \cup \nu$ and $\beta_\tau = \gamma_\gamma = \alpha_\gamma$, we have $(\rho(\ell^o(t))^o(\mu)) \cup \gamma) : \emptyset \rightarrow_\tau \beta_\sigma$ and $(\rho(\ell^o(t))^o(\mu)) \cup \gamma) : \emptyset \rightarrow_\tau \beta_\tau$. Moreover, play($\rho'$) = play($\rho$)$, Int($\rho'$) = Int($\rho$)$\ell^o(m)$, play($\rho'_\sigma$) = play($\rho_\sigma$)$\ell(m)$ and play($\rho'_\tau$) = play($\rho_\tau$), making the required properties clear. Finally, Coll($\rho'$) = Coll($\rho$) $\cup \ell^o(v)$, with Coll($\rho'_\sigma$) = Coll($\rho_\sigma$) $\cup \nu$ and Coll($\rho'_\tau$) = Coll($\rho_\tau$), so Coll($\rho'$) = Coll($\rho'_\sigma$) $\lor$ Coll($\rho'_\tau$) follows from IH. Negative transitions from $\tau$ are symmetric.

Consider finally $\rho' = \rho(t^o \oplus t^o)(\mu) \cup \gamma) : \emptyset \rightarrow_\tau \beta$ where $\rho : \emptyset \rightarrow_\tau \alpha$. $t^o(\mu) : \ell^o(\mu)$ $t^o(\mu) : \emptyset \rightarrow_\tau \beta$, where necessarily, $\alpha = \ell^o(\mu) \cup \gamma$, $\beta = \ell^o(v) \cup \gamma$, and where we necessarily have $t^o(s, d) : \emptyset \rightarrow_\tau \emptyset$, with $\delta_\sigma(t^o)(\mu) = (s, d), \ell(m) = (\ell(m), s, d)$ where $\ell(m) = \delta_\sigma(t^o)$; and $\delta_\sigma(t^o)(s, d) = \emptyset$. Now,

$\rho_\sigma : \emptyset \rightarrow_\tau \alpha_\sigma$, $\rho_\tau : \emptyset \rightarrow_\tau \alpha_\tau$

for $\alpha = \alpha_\sigma + \gamma_\tau$ by IH, and necessarily $\gamma = \gamma_\sigma + \gamma_\tau$ with $\alpha_\sigma = \mu \cup \gamma_\sigma$ and $\alpha_\tau = \gamma_\tau$. Writing $\beta_\sigma = \gamma_\sigma$ and $\beta_\tau = \gamma_\tau$, we have $\beta = \beta_\sigma \cup \beta_\tau$. Hence, we can form the projected runs

$\rho'_\sigma = \rho'_\sigma(t^o(\mu) \cup \gamma) : \emptyset \rightarrow_\tau \beta_\sigma$, $\rho'_\tau = \rho'_\tau(t^o(\mu) \cup \gamma) : \emptyset \rightarrow_\tau \beta_\tau$,

satisfying play($\rho'$) = play($\rho$), Int($\rho'$) = Int($\rho$)$\ell^o(m)(m)$, play($\rho'_\sigma$) = play($\rho_\sigma$)$\ell(m)$ and play($\rho'_\tau$) = play($\rho_\tau$)$\ell(m)$ from which the required verifications are immediate. Finally, Coll($\rho'$) = Coll($\rho$) $\cup \ell^o(v)$ with Coll($\rho'_\sigma$) = Coll($\rho_\sigma$) and Coll($\rho'_\tau$) = Coll($\rho_\tau$) $\cup \nu$, so Coll($\rho'$) = Coll($\rho'_\sigma$) $\lor$ Coll($\rho'_\tau$) follows from IH. The case $t^o \oplus t^o$ is symmetric, concluding the proof. □
Using this lemma, we shall now prove that a valid run of $\tau \odot \sigma$ projects to valid runs on $\sigma$ and $\tau$. For this we will also exploit the following easy lemma.

**Lemma C.5.** Consider $A, B$ arenas, and $s \in [A \vdash B]^*$. Then, $s \in \text{Plays}(A \vdash B)$ iff $s \uparrow t^*_s \in \text{Plays}(A)$ and $s \uparrow r^*_s \in \text{Plays}(B)$.

**Proof.** Straightforward. \qed

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

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

\[ \ell^\circ(t^0)(\ell^\circ(\alpha)) : \ell^\circ(\mu) \rightarrow_{\tau \odot \sigma} \ell^\circ(\nu), \quad \ell^\circ(t^0)(\ell^\circ(\alpha)) \uplus \gamma : \alpha \rightarrow_{\tau \odot \sigma} \beta, \]

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

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

\[ \ell^\circ(t^+)(\ell^\circ(\mu)) : \ell^\circ(\mu) \rightarrow_{\tau \odot \sigma} \emptyset, \quad \ell^\circ(t^+)(\ell^\circ(\alpha)) \uplus \gamma : \alpha \rightarrow_{\tau \odot \sigma} \beta, \]

where necessarily $\alpha = \ell^\circ(\mu) \uplus \gamma$ and $\beta = \gamma$, and $t^+(\mu) : \mu \rightarrow_{\tau \odot \sigma} \emptyset$. By IH, we have

\[ \text{play}(\rho_{\sigma}) \in \text{Plays}(A \vdash B), \quad \text{play}(\rho_{\tau}) \in \text{Plays}(B \vdash C) \]

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

\[ \rho_{\sigma} : \emptyset \rightarrow_{\tau \odot \sigma} \alpha_{\sigma}, \quad t^+(\mu) \uplus \gamma_{\sigma} : \alpha_{\sigma} \rightarrow_{\tau \odot \sigma} \beta_{\sigma} \]

with components named as in the proof of Lemma C.4, and with $s = \text{play}(\rho_{\sigma}) \in \text{Plays}(A \vdash B)$. Hence by condition valid of Petri strategies, $s(t^+ m) = \text{play}(\rho'_\tau) \in \text{Plays}(A \vdash B)$, which concludes this case. The case of a positive transition from $\tau$ is symmetric.

Consider next that $\rho' = \rho(\ell^\circ(t^-)(\ell^\circ(\alpha)) \uplus \gamma) : \emptyset \rightarrow_{\tau \odot \sigma} \beta$, where $\rho : \emptyset \rightarrow_{\tau \odot \sigma} \alpha$, 

\[ \ell^\circ(t^-)(\ell^\circ(\mu)) : \emptyset \rightarrow_{\tau \odot \sigma} \ell^\circ(\nu), \quad \ell^\circ(t^-)(\ell^\circ(\alpha)) \uplus \gamma : \alpha \rightarrow_{\tau \odot \sigma} \beta, \]

in that case $\text{play}(\rho'_\alpha) = \text{play}(\sigma) \ell^\circ(m^-)$ and $\text{play}(\rho'_\tau) = \text{play}(\tau)$. By IH we have $\text{play}(\rho'_\alpha) \in \text{Plays}(A \vdash B)$ and $\text{play}(\rho'_\tau) \in \text{Plays}(B \vdash C)$. But by hypothesis, we have $\text{play}(\rho') = \text{play}(\rho) \ell^\circ(m) \in \text{Plays}(A \vdash C)$. By Lemma C.5, $\text{play}(\rho(\ell^\circ(m)) \uparrow \ell^\circ = \text{play}(\rho) \uparrow \ell^\circ m \in \text{Plays}(A)$. But play $\text{play}(\rho'_\alpha) \uparrow \ell^\circ = \text{play}(\rho) \uparrow \ell^\circ m \in \text{Plays}(A)$, and $\text{play}(\rho'_\alpha) \uparrow \ell^\circ = \text{play}(\rho_{\sigma}) \uparrow \ell^\circ$, so play $(\rho'_\alpha) \in \text{Plays}(A \vdash B)$ by Lemma C.5. The case of a negative transition from $\tau$ is symmetric.

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

\[ (t^+ \odot t^-)(\ell^\circ(\mu)) : \ell^\circ(\mu) \rightarrow_{\tau \odot \sigma} r^\circ(\nu), \quad (t^+ \odot t^-)(\ell^\circ(\mu)) \uplus \gamma : \alpha \rightarrow_{\tau \odot \sigma} \beta, \]

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

\[ t^+(\mu) : \mu \rightarrow_{\tau \odot \sigma} \emptyset, \quad t^-((s, d)) : \emptyset \rightarrow_{\tau \odot \sigma} r \nu, \]

with $\delta_\sigma(t^+)(\mu) = (s, d)$, $\ell^\circ(m) = (\ell^\circ(m), s, d)$ where $\ell^\circ(m) = \delta_\sigma(t^+)$; and $\delta_\tau(t^-)(s, d) = \nu$. By IH,

\[ \text{play}(\rho_{\sigma}) \in \text{Plays}(A \vdash B), \quad \text{play}(\rho_{\tau}) \in \text{Plays}(B \vdash C). \]

Summing up the situation on the side of $\sigma$, we have

\[ \rho_{\sigma} : \emptyset \rightarrow_{\tau \odot \sigma} \alpha_{\sigma}, \quad t^+(\mu) \uplus \gamma_{\sigma} : \alpha_{\sigma} \rightarrow_{\tau \odot \sigma} \beta_{\sigma} \]

The Concurrent Games Abstract Machine

Lemma C.4. The case of a negative transition from play

Lemma C.5; so by Lemma C.5 again we deduce play(\rho'_s) \in \text{Plays}(B+C) as required. The case of a synchronized transition \tau^- \otimes \tau^+ is symmetric.

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

Proposition C.7. If \sigma : A+B and \tau : B+C are Petri strategies, then so is \tau \circ \sigma : A+C.

Proof. Valid. Consider \rho : \emptyset \longrightarrow_{\tau \circ \sigma} \alpha with play(\rho) \in \text{Plays}(A+B), and \tau^+ : \alpha \xrightarrow{m'} \beta. \text{W.l.o.g., assume } m' = \ell, (m). By Lemma C.4, \alpha decomposes as \alpha_\sigma \oplus \alpha_\tau, and we have:

\rho_\sigma : \emptyset \longrightarrow_{\tau \circ \sigma} \alpha_\sigma, \quad \rho_\tau : \emptyset \longrightarrow_{\tau \circ \alpha} \alpha_\tau.

Now, \tau^+ must have the form \tau^+ = (\ell^\circ (t^+) \uplus \ell^\circ (\mu)) \cup \gamma with \tau^+ = \mu \xrightarrow{\ell^\circ (m)} \emptyset, and \beta = \gamma. Hence,

\tau^+ = \mu \xrightarrow{\ell^\circ (m)} \emptyset, and \beta = \gamma. Hence,

\tau^+ = \mu \xrightarrow{\ell^\circ (m)} \emptyset, and \beta = \gamma. Hence,

\tau^+ = \mu \xrightarrow{\ell^\circ (m)} \emptyset, and \beta = \gamma. Hence,
Finally, assume $t = (t^+ \oplus t^-) \| (t^0(\mu)) \cup \gamma$. Say that we have 
\[
(t^+ \oplus t^-)\| (t^0(\mu)) : t^0(\mu) \rightarrow_{\tau \circ \sigma} r^0(v),
\]
so that new(t) = $r^0(v)$ – assume $t^+ \| \mu : t^+(\mu) \emptyset$ and $t^- \| (s, d)) : \emptyset \rightarrow_{t^-(\mu)} v$. By Lemma C.5, 
play($\rho_\tau$)$\ell (m) \in \text{Plays}(B \vdash C)$. Therefore since $\tau$ is strongly safe, 
new(t$^-(\mu)$) = $v$ is fresh in $\rho_\tau$. But by Lemma C.4, Coll($\rho$) = Coll($\rho_\sigma$) $\oplus$ Coll($\rho_\tau$), so new(t) = $r^0(v)$ is fresh in $\rho$ as required. 

The other synchronized case is symmetric. 

**Negative.** Straightforward by inspection and negativity of $\sigma$ and $\tau$. \hfill $\square$

### C.1.2 Copycat

First, we characterize the markings of copycat reachable through a play:

**LEMMA C.8.** Consider $A$ an arena, and $\rho : \emptyset \rightarrow_{a_A} \alpha$ such that play($\rho$) $\in \text{Plays}(A \vdash A)$. 

Then, $|\text{play}(\rho)| = x \vdash y$ with $x, y \in C(A)$ and $y \sqsupset x \cap y \sqsubseteq x$; and $\alpha = (y^- \setminus x) \cup (x^+ \setminus y)$. 

Moreover, Coll($\rho$) = $y^- \cup x^+$, where $x^0$ is the subset of $x \in C(A)$ whose moves have polarity $p$.

**Proof.** By induction on $\rho$. For $\rho$ empty, this is clear. Consider $\rho' = \rho t : \emptyset \rightarrow_{a_A} \beta$ with $\rho : \emptyset \rightarrow_{a_A} \alpha$, and play($\rho'$) $\in \text{Plays}(A \vdash A)$. We also have play($\rho$) $\in \text{Plays}(A \vdash A)$, so by IH, 

\[
|\text{play}(\rho)| = x \vdash y, \quad y \sqsupset x \cap y \sqsubseteq x, \quad \alpha = (y^- \setminus x) \cup (x^+ \setminus y),
\]

we reason by cases depending on $t$. If $t = (m^+, r')\| ((s, d)^{(\alpha_m)}) \cup \gamma$, then $\alpha = ((s, d)^{(\alpha_m)}) \cup \gamma$, and 

\[
|\text{play}(\rho')| = x \vdash (y \cup \{(m, s, d)\}) \\
\beta = \alpha \cup \{(s, d)^{(\alpha_m)}\};
\]

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

If $t = (m^-, r')\| ((s, d)^{(\alpha_m)}) \cup \gamma$, then $\alpha = \gamma$, and 

\[
|\text{play}(\rho')| = x \vdash (y \cup \{(m, s, d)\}) \\
\beta = \alpha \cup \{(s, d)^{(\alpha_m)}\};
\]

and $(m, s, d)$ is negative. Since $|\text{play}(\rho')| = x \vdash (y \cup \{(m, s, d)\})$, $(m, s, d) \not\in y$. So, it cannot be in $x$. The invariant directly follows. The case $t = (m^+, \ell)\| ((s, d)^{(\alpha_m)}) \cup \gamma$ is symmetric. \hfill $\square$

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

**PROPOSITION C.9.** For any arena $A$, $a_A : A \vdash A$ is a negative Petri strategy.

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

\[
\rho : \emptyset \rightarrow_{s} \alpha, \quad t^+ : \alpha \rightarrow_{m} \beta,
\]

and say w.l.o.g. that $m = \kappa_a$ for $a \in |A|$. This means that $t^+ = (m^+, r')\| ((s, d)^{(\alpha_m)}) \cup \gamma$, where $\alpha = \gamma \cup \{(s, d)^{(\alpha_m)}\}$, and $m = (\kappa_m, s, d)$. We must show that $s(\kappa_m, s, d) \in \text{Plays}(A \vdash A)$.

By Lemma C.8, $\alpha = (y^- \setminus x) \cup (x^+ \setminus y)$ where $|s| = x \vdash y$. As $m \in \text{mult}^+(A)$, $(m, s, d) \in x^+ \setminus y$. As $(m, s, d) \not\in y$, we have condition non-repetitive. Next, we show $|sm| = x \vdash (y \cup \{a\})$ is down-closed. Consider $a' : \kappa_A a$. Since $A$ is alternating, $a'$ is negative. Since $x \in C(A)$ and $a \in x$, we must have $a' \in x$ as well. But by Lemma C.8 we have $y \sqsupset x \cap y \sqsubseteq x$, 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 $x \cap y \sqsubseteq x$ and $x \cap y \sqsubseteq y$, we have $x \cup y$ consistent, so in particular $y \cup \{a\} \in C(A)$. It follows that $sm \in \text{Plays}(A \vdash A)$ as needed.
We examine the operations involved in the \( \sigma \) (for polarity reasons), it follows that \( \alpha \)

C.2.1 Tensor. The preservation of Petri strategies by the tensor operation is a simplification of \( \text{PStrat} \)

Consider \( \rho : \emptyset \rightarrow \omega_{A} \alpha \) with play(\( \rho \)) = \( s \) \in \text{Plays}(A \vdash A) \) and \( sm^{-} \in \text{Plays}(A \vdash A) \). \( \text{W.l.o.g.} \) consider \( m = r_{\rho}(a) \). Decompose \( a = (m, s, d) \) for \( m^{-} \in \text{mult}(A) \) and token \( (s, d) \). By inspection there is a unique matching instantiated transition, namely \( (m^{-}, r_{\rho})(s, d) : \emptyset \rightarrow \{(s, d)^{\oplus m}\} \). Moreover, by Lemma C.8 we have \( \alpha = (y^{-} \setminus x) \cup (x^{+} \setminus y) \) where \( |s| = x \vdash y \). But as \( sm^{-} \in \text{Plays}(A \vdash A) \), by non-repetitive we have \( m^{-} \notin y \). It follows that \( (s, d)^{\oplus m} \notin \rho \).

Stronally safe. Consider \( \rho : \emptyset \rightarrow \omega_{A} \alpha \) with play(\( \rho \)) = \( s \) \in \text{Plays}(A \vdash A) \) extended with \( t \). If \( t \) is positive, then new(\( t \)) = \( \emptyset \) and there is nothing to prove. Hence, consider w.l.o.g. \( t = (m^{-}, r_{\rho})(s, d) \cup s(m) \), \( s(d)^{-} \in \text{Plays}(A \vdash A) \). Then, new(\( t \)) = \( \{(s, d)^{\oplus m}\} \). Write \( |s| = x \vdash y \in \Sigma(A \vdash A) \). By non-repetitive, \( (s, d) \notin y \). By Lemma C.8, Coll(\( \rho \)) = \( y^{-} \cup x^{+} \); but as \( (m, s, d) \notin y \) and \( (m, s, d) \notin x^{+} \) (for polarity reasons), it follows that \( \{(s, d)^{\oplus m}\} \) is fresh in \( \rho \).

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

### C.2 PStrat as an IPA-Structure: Operations

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

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

**Lemma C.11.** Consider \( \sigma \) and \( \tau \) Petri structures.

Then, instantiated transitions of \( \sigma \otimes \tau \) are exactly as in Figure 30 — 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 31 partial functions \( \pi_{\sigma}^{\oplus} : \text{IT}_{\sigma \otimes \tau} \rightarrow \text{IT}_{\sigma} \) and \( \pi_{\tau}^{\oplus} : \text{IT}_{\sigma \otimes \tau} \rightarrow \text{IT}_{\tau} \) extracting various data from instantiated transitions, following the characterization of instantiated transitions of \( \sigma \otimes \tau \) given in Figure 30.
we will also use for restriction the partial functions which allow us to prove the following property:

$$\begin{align*}
\ell^\sigma(t^0)(\ell^\sigma(\alpha)) & \mapsto t^0(\alpha) \\
\ell^\sigma(t^+)(\ell^\sigma(\alpha)) & \mapsto t^+(\alpha) \\
\ell^\sigma(t^-)(s, d) & \mapsto t^-(s, d) \\
\epsilon^\sigma(t^0)(\epsilon^\sigma(\alpha)) & \mapsto t^0(\alpha) \\
\epsilon^\sigma(t^+)(\epsilon^\sigma(\alpha)) & \mapsto t^+(\alpha) \\
\epsilon^\sigma(t^-)(s, d) & \mapsto t^-(s, d)
\end{align*}$$

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

$$\begin{align*}
\pi_\sigma^\sigma : & \text{ITC}_{\sigma \otimes \tau} \rightarrow \text{ITC}_\sigma \\
\pi_\tau^\sigma : & \text{ITC}_{\sigma \otimes \tau} \rightarrow \text{ITC}_\tau \\
\ell \upharpoonright (\gamma^+ \gamma') & \mapsto \pi_\sigma^\sigma(\ell) \upharpoonright \gamma \\
\ell \upharpoonright (\gamma^+ \gamma') & \mapsto \pi_\tau^\sigma(\ell) \upharpoonright \gamma'.
\end{align*}$$

Using these, from a run $\rho : 0 \rightarrow_{\sigma \otimes \tau} \alpha$ we extract:

$$\rho_\sigma = \rho \upharpoonright \pi_\sigma^\sigma, \quad \rho_\tau = \rho \upharpoonright \pi_\tau^\sigma,$$

we will also use for restriction the partial functions

$$\begin{align*}
u : & \text{Moves} \rightarrow \text{Moves} \\
d : & \text{Moves} \rightarrow \text{Moves}
\end{align*}$$

which allow us to prove the following property:

**Lemma C.12.** Consider $\rho : 0 \rightarrow_{\sigma \otimes \tau} \alpha$. Then, $\alpha = \alpha_\sigma^+ \alpha_\tau$ and

$$\rho_\sigma : 0 \rightarrow \alpha_\sigma, \quad \rho_\tau : 0 \rightarrow \alpha_\tau.$$

where play$(\rho_\sigma) = \text{play}(\rho) \upharpoonright \nu$ and play$(\rho_\tau) = \text{play}(\rho) \upharpoonright d$.

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

**Proof.** Exactly as for Lemma C.4, without synchronized transitions. \(\square\)

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

**Lemma C.13.** Consider $A, B$ arenas, and $s \in |A \otimes B|^*$. Then, $s \in \text{Plays}(A \otimes B)$ iff $s \uparrow t^*_s \in \text{Plays}(A)$ and $s \uparrow r^*_s \in \text{Plays}(B)$.

**Proof.** Straightforward. \(\square\)

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

**Lemma C.14.** Consider $\rho : 0 \rightarrow_{\sigma \otimes \tau} \alpha$ such that $\text{play}(\rho) \in \text{Plays}(A_1 \otimes A_2 \rightarrow B_1 \otimes B_2)$. Then, play$(\rho_\sigma) \in \text{Plays}(A_1 \rightarrow B_1)$ and play$(\rho_\tau) \in \text{Plays}(A_2 \rightarrow B_2)$.

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

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

**Proposition C.15.** If $\sigma : A_1 \rightarrow B_1$ and $\tau : A_2 \rightarrow B_2$ are Petri strategies, so is $\sigma \otimes \tau : A_1 \otimes B_1 \rightarrow A_2 \otimes B_2$. Moreover, if $\sigma$ and $\tau$ are negative, so is $\sigma \otimes \tau$.
C.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 $\text{Plays}_-(A)$ for the set of negative plays on $A$, i.e. those $s_1 \ldots s_n \in \text{Plays}(A)$ such that $\text{pol}(s_1) = -$. If $f : \text{Moves} \rightarrow \text{Moves}$ and $s = s_1 \ldots s_n \in \text{Plays}(A)$ such that $f$ is defined on $|A|$, then we write $f(s) = f(s_1) \ldots f(s_n)$. In the sequel, we should be particularly interested in such functions on moves that can be decomposed in $f$ and $(g_m)_{m \in \text{dom}(f)}$ where

$$f : M \rightarrow M,$$

$$g_m : \text{Tok} \rightarrow \text{Tok},$$

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

$$[f, (g_m)] : \text{Plays}_- \rightarrow \text{Plays}_- \quad (m, s, d) \mapsto (f(m), s', d')$$

where $(s', d') = g_m(s, d)$.

**Definition C.16.** Consider $A, B$ games, $f, (g_m)$ s.t. $[f, (g_m)] : \text{Plays}_- \rightarrow \text{Plays}_-$ partial injection. We say $h = [f, (g_m)]$ is a global renaming from $A$ to $B$, written $[f, (g_m)] : A \Rightarrow B$, if:

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

Global renamings are used to transport Petri strategies across games. The following definition, first applied simply on Petri structures, extends Definition 3.14 in that it also renames tokens rather than merely rerouting visible transitions.

**Definition C.17.** Consider $A, B$ games, $\sigma$ a Petri structure on $\text{mult}(A)$, and $h = [f, (g_m)] : A \Rightarrow B$. We define the renaming $\sigma[h]$ on $\text{mult}(B)$, with the same components as $\sigma$, except:

$$\begin{align*}
\partial_{\sigma[h]}(t) &= f(\partial_{\sigma}(t)) \\
\delta_{\sigma[h]}(t^+)(\alpha) &= g_m(\delta_{\sigma}(t^+)(\alpha)) & \text{for } m = \partial_{\sigma}(t^+)
\end{align*}$$

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

In order to use global renaming to transport Petri strategies, we must transport valid runs. Consider $A, B$ games, $h = [f, (g_m)] : A \Rightarrow B$, and $\sigma : A$ a Petri strategy. Then we set

$$[h] : I\text{I}_{\sigma} \rightarrow I\text{I}_{\sigma[h]}$$

$$t^0(\alpha) \mapsto t^0(\alpha)$$

$$t^-((s, d)) \mapsto t^-(g_m(s, d))$$

where $m = \partial_{\sigma}(t^-)$

extended to instantiated transitions in context with $(t \lor \gamma)[h] = t[h] \lor \gamma$. 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 $\rho : \emptyset \rightarrow_{\sigma} \alpha, \rho[h] : \emptyset \rightarrow_{\sigma[h]} \alpha$ is defined pointwise.

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

**Lemma C.18.** Consider $A$ a game, and $\sigma : A$ a negative Petri strategy.
Then, for all $\rho : \emptyset \rightarrow_{\sigma} \alpha$ s.t. $\text{play}(\rho) \in \text{Plays}(A)$, we have $\text{play}(\rho) \in \text{Plays}_-(A)$.

**Proof.** By negative, the first transition of $\rho$ cannot be positive or neutral (as those require at least one tokil). Thus, it is negative. □
LEMMA C.19. Consider a game, $\sigma : A$ negative, $h = [f, (g_m)] : A \bowtie B$, and $\rho : \emptyset \to\sigma \alpha$. If $\text{play}(\rho) \in \text{Plays}_-(A)$, then $\text{play}(\rho[h]) = h(\text{play}(\rho)) \in \text{Plays}_-(B)$.

Proof. Straightforward by induction on $\rho$. □

We shall also use a sort of reciprocal statement:

LEMMA C.20. Consider a game, $\sigma : A$ negative, $h = [f, (g_m)] : A \bowtie B$, and $\rho' : \emptyset \to\sigma[h] \alpha$. If $\text{play}(\rho') \in \text{Plays}(B)$, there is a unique $\rho : \emptyset \to\sigma \alpha$ s.t. $\text{play}(\rho) \in \text{Plays}(A)$ and $\rho' = \rho[h]$. Proof. Straightforward by induction on $\rho'$. □

PROPOSITION C.21. Consider a game, $\sigma : A$ negative, $h = [f, (g_m)] : A \bowtie B$. Then, $\sigma[h] : B$ is a negative Petri strategy.

Proof. Valid. Consider $\rho' : \emptyset \to\sigma[h] \alpha$ such that $\text{play}(\rho') \in \text{Plays}(B)$. Consider $t^+ = \alpha \xrightarrow{b} \sigma[h] \beta$, write $t = t^+[\mu] \uplus y$. By Lemma C.20, there is a unique $\rho : \emptyset \to\sigma \alpha$ such that $\text{play}(\rho) \in \text{Plays}(A)$ and $\rho' = \rho[h]$. By definition of transitions of $\sigma[h]$, we have $b = h(a)$ with

$$t^+[\mu] \uplus y : \alpha \xrightarrow{\sigma} \beta$$

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

Receptive. Consider $\rho' : \emptyset \to\sigma[h] \alpha$ such that $s' = \text{play}(\rho') \in \text{Plays}(B)$. Consider $sb^- \in \text{Plays}(B)$. By Lemma C.20, there is a unique $\rho : \emptyset \to\sigma \alpha$ such that $s = \text{play}(\rho) \in \text{Plays}(A)$ and $h(s) = s'$. By condition receptivity of global renamings, there is $sa^- \in \text{Plays}(A)$ such that $h(a) = b$. As $\sigma$ is receptive, there is a unique $t^- \in \Pi\sigma$ such that $t^- : \emptyset \to\sigma[h] \beta$ for some $\beta$. By definition of $\sigma[h]$, $t^- : \emptyset \to\sigma[h] \beta$ as required. Uniqueness follows immediately from uniqueness for $\sigma$.

Strongly safe. Consider $\rho' : \emptyset \to\sigma[h] \alpha$ such that $s' = \text{play}(\rho') \in \text{Plays}(B)$. By Lemma C.20, there is a unique $\rho : \emptyset \to\sigma \alpha$ such that $s = \text{play}(\rho) \in \text{Plays}(A)$ and $h(s) = s'$. If $t' : \alpha \to\sigma[h] \beta$ then also $t' : \alpha \to\sigma \beta$, and new$(t')$ is fresh in $\rho$, so fresh in $\rho'$. If $t' : \alpha \xrightarrow{sb} \sigma[h] \beta$ with $s'b \in \text{Plays}(B)$, then again by Lemma C.20, $t' = t[h]$ and $b = h(a)$ for $t : \alpha \xrightarrow{a} \sigma \beta$ with $sa \in \text{Plays}(A)$. As $\sigma$ is strongly safe, it follows that new$(t)$ is fresh in $\rho$, but new$(t) = $ new$(t')$ so new$(t')$ is fresh in $\rho'$ as required.

Negative. Straightforward from the fact that $\sigma$ is negative. □

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

LEMMA C.22. Consider $\Gamma, x : A, \Lambda$ a list of variable/arena declarations, and $O$ well-opened. Then,

$$(\Lambda x, (\text{id})) : (!([\Gamma, x : A, \Delta]) \vdash O) \bowtie (!([\Gamma, \Delta]) \vdash A \Rightarrow O)$$

where $\Lambda x$ is defined in Definition 3.15.

Proof. Immediate verification. □

COROLLARY C.23. Consider $\sigma : ![\Gamma, x : A, \Delta]) \vdash O$ a negative Petri strategy. Then, $\Lambda x : A, O \sigma : ![\Gamma, \Delta]) \vdash !A \Rightarrow O$ is a negative Petri strategy.

C.2.4 Functorial promotion. Rather than directly dealing with Definition 3.16, we decompose it: first, a functorial promotion, and secondly, a renaming corresponding to digging.

We first define functorial promotion on Petri structures:

we will also use for restrictions the partial functions where 
\[ e \mapsto \beta \]
for all \( e \) and undefined otherwise – the abuse of notations should not create confusion.

\[ t^0([e : \alpha]) : e \mapsto_\sigma \beta \quad t^-((s, d)) : \emptyset \mapsto_\sigma \beta \quad t^+([e : \alpha]) : e \mapsto_\sigma \emptyset \]
\[ t^0([e :: \alpha]) : e :: \alpha \mapsto_\sigma \beta \quad t^-((e :: s, d)) : \emptyset \mapsto_\sigma e :: \beta \quad t^+(e :: \alpha) : e :: \alpha \mapsto_\sigma \emptyset \]

Fig. 32. Description of instantiated transitions of \( !\sigma \)

**Definition C.24.** Consider \( \sigma \in \text{PStruct}(M, N) \). We set \( L_\sigma = L, T_\sigma = T \) with the same polarities, \( \partial_\sigma = \partial \), and pre- and post-conditions are also unchanged. Finally, the **transition table** is:

\[
\begin{align*}
\delta_\sigma(t^0)(e :: \alpha) &= e :: \beta & \text{if } \delta_\sigma(t)(\alpha) &= \beta \\
\delta_\sigma(t^*)(e :: \alpha) &= (e :: s, d) & \text{if } \delta_\sigma(t)(\alpha) &= (s, d) \\
\delta_\sigma(t^-)(e :: s, d) &= e :: \alpha & \text{if } \delta_\sigma(t)(s, d) &= \alpha
\end{align*}
\]

where \( e :: \alpha \) is \( \{ (e :: s_i, d_i) @^l_i \mid (s_i, d_i) @^l_i \in \alpha \} \).

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

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 C.25.** Consider \( \sigma \) a Petri structure.

Then, instantiated transitions of \( !\sigma \) are exactly as in Figure 32 – 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 for each \( e \in E \) a partial function

\[ \pi_\sigma^e : \text{IT}_\sigma \rightarrow \text{IT}_\sigma \]

\[
\begin{align*}
\pi_\sigma^e(t^0([e :: \alpha])) &= t^0([e :: \alpha]) \\
\pi_\sigma^e(t^-((e :: s, d))) &= t^-((e :: s, d)) \\
\pi_\sigma^e(t^+(e :: \alpha)) &= t^+(e :: \alpha)
\end{align*}
\]

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

\[ \bigoplus_{e \in E} \alpha_e = \bigoplus_{e \in E} e :: \alpha_e \]

for \( (\alpha_e)_{e \in E} \); a family of conditions empty almost everywhere. We may then set:

\[ \pi_\sigma^e : \text{ITC}_\sigma \rightarrow \text{ITC}_\sigma \\
\pi_\sigma^e(t (\bigoplus_{e \in E} \gamma_e)) \mapsto \pi_\sigma^e(t) \cup \gamma_e .
\]

Using these, from a run \( \rho : \emptyset \rightarrow_{\sigma} \alpha \) we extract, for all \( e \in E \):

\[ \rho_e = \rho \upharpoonright \pi_\sigma^e , \]

we will also use for restrictions the partial functions

\[ e \begin{array}{cc}
\text{Moves} & \rightarrow \text{Moves} \\
(m, e :: s, d) & \mapsto (m, s, d)
\end{array} \]

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

Now, as for the tensor we can prove:

**Lemma C.26.** Consider \( \rho : \emptyset \rightarrow_{\sigma} \alpha \). Then, \( \alpha = \bigoplus_{e \in E} \alpha_e \) and

\[ \rho_e : \emptyset \rightarrow_{\sigma} \alpha_e \]

for all \( e \in E \), where \( \text{play}(\rho_e) = \text{play}(\rho) \upharpoonright e \).

Moreover, \( \text{Coll}(\rho) = \bigoplus_{e \in E} \text{Coll}(\rho_e) \).
The proof is the same as for Lemma C.4, without synchronized transitions. □

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

**Lemma C.27.** Consider an arena and \( s \in |!A|^\ast \).
Then, \( s \in \text{Plays}(!A) \) iff \( s \uparrow e \in \text{Plays}(A) \) for all \( e \in \mathcal{E} \).

**Proof.** Straightforward. □

Consider now \( \sigma : A \vdash B \) a Petri strategy.
Using Lemmas C.27 and C.26, we show that projections preserve valid runs.

**Lemma C.28.** Consider \( \rho : \emptyset \rightarrow \rightarrow_{!\sigma} \alpha \) such that \( \text{play}(\rho) \in \text{Plays}(!A \vdash !B) \).
Then, for all \( e \in \mathcal{E} \), \( \text{play}(\rho_e) \in \text{Plays}(A \vdash B) \).

**Proof.** As for the proof of Lemma C.6 (without synchronization), using condition valid of Petri strategies along with Lemma C.27. □

Using Lemmas C.26, C.27 and C.28, we prove as for Proposition C.7:

**Proposition C.29.** If \( \sigma : A \vdash B \) is a Petri strategy, then so is \( !\sigma : !A \vdash !B \).
Moreover, if \( \sigma \) is negative then so is \( !\sigma \).

### C.2.5 Local renamings

To match Definition 3.16, we must also rename following digging.
Recall that digging is the following map:

\[
\text{dig} : \text{Moves} \rightarrow \text{Moves} \\
(m, e :: e' :: s, d) \mapsto (m, (e, e') :: l, d)
\]

and undefined otherwise. To rename a strategy following this, it is convenient to introduce:

**Definition C.30.** Consider \( A, B \) arenas.

A (local) renaming from \( A \) to \( B \) is a partial injection \( f : \text{Moves} \rightarrow \text{Moves} \) defined on \(|A|\), s.t.:

- **validity:** for all \( x \in \mathcal{C}(A) \), \( fx \in \mathcal{C}(B) \),
- **polarity-preserving:** for all \( a \in |A| \), \( \text{pol}(f(a)) = \text{pol}(a) \),
- **receptivity:** for all \( x \in \mathcal{C}(A) \), if \( f(x) \vdash_B b^- \),
  then there is \( x \vdash_A a \) such that \( f(a) = b \),
- **courtesy:** for all \( a \vdash_A a' \), either \( f(a) \vdash_B f(a') \) or \( (\text{pol}(a), \text{pol}(a')) = (\neg, +) \).

We write \( f : A \bowtie B \) to mean that \( f \) is a renaming from \( A \) to \( B \).

It is clear in particular that \( \text{dig} : !!A^\perp \rightarrow !!A \) is a renaming, for any arena \( A \). This is a variant of Definition C.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 \( f : A^\perp \bowtie B^\perp \) and \( g : A' \bowtie B' \) are local renaming, then it is obvious that \( f \upharpoonright g : A \vdash B \bowtie A' \vdash B' \) (defined in the obvious way) is still a local renaming – this is not always the case for global renamings for non-negative games.

**Definition C.31.** Consider \( \sigma : A \vdash B \) a negative Petri strategy and \( f : A^\perp \bowtie A'^\perp, g : B \bowtie B' \).
Then, we define \( g \cdot \sigma \cdot f = \sigma[f \upharpoonright g] : A' \vdash B' \).

This yields a negative Petri strategy by Proposition C.21.
C.2.6 Digging. We may finally perform digging and deduce the correctness of promotion:

**Proposition C.32.** Consider \( \sigma : !A \vdash B \) a negative Petri strategy. Then, \( \sigma^+ : !A \vdash !B \) is a negative Petri strategy.

**Proof.** It is a direct verification that \( \sigma^+ = \langle \sigma \rangle \) [\( \text{dig} \) \( \text{id} \)], which is a negative Petri strategy by Propositions C.29 and Definition C.31.

C.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 \( \sigma \) and a play \( s \) of a game \( A \), we say that \( s \) is reachable by \( \sigma \) when there exists a run \( \rho \) of \( \sigma \) with \( \text{play}(\rho) = s \). Given a game \( A \), we define the Scott order on \( \mathcal{P}(A) \) as follows: \( x \sqsubseteq_A y := (x \sqsubseteq \sqsubseteq^+ y) \) which was already encountered in Lemma C.8 for copycat.

C.3.1 Variable and Evaluation. For variable, we notice that \( \text{var}_{x:M} = \mathcal{A}[\ell^+_x \vdash \text{id}] \) and we conclude easily by C.21 since \( \ell^+_x : M^\perp \sim \Gamma, x : M, A \downarrow \) is a local renaming. For the evaluation, the map \( \Omega \) defined in Section 3.4 is a global renaming ((\( M \to N \)) \( \to (M \to N) \) \( \sim \) (\( M \to N \)) \( \otimes \) \( M \vdash N \)).

C.3.2 Contraction. We now show that the Petri structure \( \rho_A \) is a Petri strategy on \( !A \vdash !A \otimes !A \).

Given a move \( e \), we write \( \ell(e) \) for \( (m, \ell(e) \vdash s, d) \) for \( a = (m, e :: s, d) \), and similarly for \( \kappa(e) \). It is not defined on moves with an empty stack.

**Lemma C.33.** Consider \( s \in \text{play}(!A \vdash !A \otimes !A) \) reachable by \( \rho_A \).

Then, \( |s| = (\ell(x_1) \cup \ell(x_2) \vdash y_1 \otimes y_2) \) 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 \( \sigma \) and \( s \) has length \( n \). We apply the induction hypothesis to \( s \) (which is reachable by \( \sigma \)) and obtain that, writing \( |s| = (\ell(x_1) \cup \ell(x_2) \vdash y_1 \otimes y_2) \), we have \( y_i \sqsubseteq x_i \).

- If \( a \) is negative, and on \( \ell \), then the inequality for \( s' \) holds by definition of \( \sqsubseteq \).
- If \( a \) is negative, and on \( \kappa \), 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 \( \ell(e) :: s \) or \( \kappa(e) :: s \) and we can conclude.
- If \( a \) is positive, and on the left, ie. \( m = \ell(m_0) \) 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 \( \kappa_\ell m_0 \) or \( \kappa_\ell m_0^{-} \) - assume the former. This directly shows that \( s = \ell(e) :: s' \) for some \( s' \), and that \( (\ell_\ell m_0, e :: s, d) \in |s| \), which implies that \( (m_0, e :: s, d) \in y \). As a result \( |s| = (x \cup \{(m_0, \ell(e) :: s', d)\}) \vdash y \otimes z \) satisfies the desired property.
- If \( a \) is positive and on the right for instance \( m = \ell_\ell m_0 \). Then a similar line of reasoning shows that \( s = e :: s' \) and we must have \( (\ell(m_0, e :: s', d) \in |s| \) which entails the desired property. \( \square \)

**Lemma C.34.** \( \rho_A \) is a negative Petri strategy on \( !A \vdash !A \otimes !A \).

**Proof.** **Negative.** Simple inspection of the net.

Strong safety. Consider \( \rho : \emptyset \rightarrow a \) with \( s \) a play, and \( t : a \rightarrow^a \beta \) with \( s \alpha \) 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 \( \rho : \emptyset \rightarrow^a a \) a run of \( \rho_A \) and \( s \) a play of the game. Assume that \( \rho \) can extend by \( t^+ : a \rightarrow^a \beta \). By Lemma C.33, we know that \( |s| = (\ell(x_1) \cup \ell(x_2) \vdash y_1 \otimes y_2) \) with \( y_i \sqsubseteq x_i \). There are
three transitions, hence three cases. We detail the case for \( a = \ell, a_0 \): 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 \( \rho : \emptyset \xrightarrow{s} \alpha \). If \( a \) is on the right of \( s \), then we can use the corresponding transition whose function domain is total on stacks of \( a \). As a result, \( a \) will be accepted by the transition corresponding to its address. \( \square \)

C.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 B.3.4.

**Lemma C.35.** Let \( \rho : \emptyset \xrightarrow{s} \alpha \) be a run of \( Y_O \) such that \( s \) is a play.

Then there exists suffix-closed \( J \subseteq E^+ \), configurations \( z, y_e \in C(O), (y_s)_{s \in J} \) and \( (z_s \in C^{\neq 0}(O))_{s \in J} \) with \( J \) empty if \( y_e \) is \( z \subseteq y_e \) and \( z_s \subseteq y_s \) for all \( s \) and

\[
|s| = (\emptyset \leadsto (\cdot : y_e) \cup (e :: (s) :: z_{e,s} \leadsto (e \cdot s) :: y_s) \vdash z,
\]

plus if \( y_e = \emptyset \) then \( J = \emptyset \), and if \( e \cdot s \in J \), then \( y_s \neq \emptyset \).

**Proof.** A direct induction over the run, using the transition table. \( \square \)

**Lemma C.36.** \( Y_O \) is a negative Petri strategy \(!O \rightarrow O \vdash O \).

**Proof.** Negativity and receptivity are easily verified.

**Validity.** Consider \( \rho : \emptyset \xrightarrow{s} \alpha \) be a run of \( Y_O \) such that \( s \) is a play of the game. Consider now an extension of \( \rho \) by the positive transition \( \ell : \alpha \xrightarrow{a} \beta \). 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 C.35

**Strong-safety.** All negative transitions have injective transition functions, and the two negative transitions \( r^m \) and \( \ell, \ell^m \) which have a common postcondition \( (m^{-}) \), have disjoint codomains, hence \( Y_O \) is strongly safe. \( \square \)

C.3.4 **Queries, Conditional, Constants.** For these IPA structures defined on linear games, a simple inspection shows that they define they are IPA strategies.

C.3.5 **Let bindings.** We now move on to showing that \( \text{let} \) is a Petri strategy on \(!X \rightarrow Y \otimes X \vdash Y\).

**Lemma C.37.** Let \( \rho : \emptyset \xrightarrow{s} \alpha \) be a valid run for \( \text{let} \). Then \( |s| = ((\forall e \in I : x_e) \rightarrow y) \otimes z \vdash w \) with:

1. if \( y \neq \emptyset \) then \( z \) is maximal in \( C(X) \);
2. \( w \subseteq y \) and \( x_e \subseteq z \) for all \( e \in I \).

**Proof.** By induction on \( \rho \). \( \square \)

**Lemma C.38.** \( \text{let} \) is a Petri strategy on \(!X \rightarrow Y \otimes X \vdash Y\).

**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 \( eat(s) \), 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 C.37. \( \square \)
C.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 $\Sigma := E \times \{ r, w \} \times D$. There is a a partial function $\pi : \Pi_{\text{newref}} \to \Sigma$ as follows:

$$\pi(w\{(\{e\}, d)^{\ominus_3}\}) = (e, w, d) \quad \pi(r\{(\{e\}, \_)^{\ominus_5}, (\_, d)^{\ominus_2}\}) = (e, r, d),$$

and undefined everywhere else. We write $\text{Tr}(\rho) = \rho \vdash \pi$.

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 C.39. Consider a run $\rho : \emptyset \rightarrow \alpha$ of newref such that $s$ is aplay. Then:

- $\text{Tr}(\rho)$ is a consistent memory trace.
- If $\rho$ is not empty, then there is a unique tokil $(s, d)^{\ominus_2}$ in $\alpha$ such that: if $\text{Tr}(\rho)$ is empty then $s = \emptyset$ and $d = 0$, otherwise $s = \{e\}$ with $e$ and $d$ the components of the last operation in $\text{Tr}(\rho)$.
- If $s$ has the shape $((\varnothing \cup e :: x_e) \rightarrow y) \uplus z$ with $z \subseteq y$ and $x_e \in \mathcal{E}(V)$ such that:
  - if $x_e$ is non-empty then $e$ occurs in $\text{Tr}(\rho)$ and the value coincides in the case of a read.
  - For every signature $e$ occurring in $\text{Tr}(\rho)$, $x_e$ is non-empty.

Proof. We proceed by induction on $\rho$, the base case being trivial. We assume $\rho = \rho' \cdot t$ with $t : \alpha \rightarrow \beta$. For the visible transitions $r_{x(-)}$ and $\ell_{x(-)}$, this is a proof similar to copycat.

- If $t$ is on $\ell_{x \cdot w \cdot y \cdot Q^{-}}$ or $\ell_{x \cdot r \cdot Q^{-}}$, there is nothing to add to the induction hypothesis.
- If $t = y \cup w\{(\{e\}, d)^{\ominus_2}, (\{e'\}, d')^{\ominus_3}\}$: then $\text{Tr}(\rho) = \text{Tr}(\rho')(\{e'\}, w, d')$ is still a consistent trace. Moreover, from the tokil $(\{e'\}, d')^{\ominus_3}$, we deduce that in $\rho'$ there must be a visible transition with move $(\ell_{x \cdot w \cdot y \cdot Q^{-}}, [e], d')$.
- If $t = y \cup w\{(\{e\}, d)^{\ominus_2}, (\{e\}, \_)^{\ominus_5}\}$: the same line of reasoning works, except that $\text{Tr}(\rho) = \text{Tr}(\rho')(\{e'\}, r, d)$ is no longer automatically consistent. However, by induction we know that in $\alpha$ 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 $\text{Tr}(\rho)$ is indeed consistent.
- If $t = y \cup w\{(\{e\}, d)^{\ominus_4}\}$: the first two conditions are trivially true. Moreover, since $(\{e\}, d)^{\ominus_4}$ belongs to $\alpha$, there must have been a transition $w$ in $\alpha$ before that put it there. That shows that there must be an element in $\text{Tr}(\rho)$ with exponential token $e$ as desired. $\square$

Lemma C.40. newref is a Petri strategy on $\forall \mathcal{V} \rightarrow X \vdash X$.

Proof. Negativity and receptivity follow by inspection of the net and transition tables.

Strong-safety. Consider a run $\rho : \emptyset \rightarrow \alpha$ such that $s$ is a play, that can be extended by a transition $t : \alpha \rightarrow \beta$ that is negative or neutral, with play($\rho t$) being a play.

Initial question. If $t$ is a negative transition $\alpha \rightarrow \beta$ on the address $xQ^{-}$. Then necessarily $\rho = e$ and so $\text{Coll}(\rho) = \emptyset$.

Final return. If $t$ is a negative transition $\alpha \rightarrow \beta$ on the address $\ell_{x^2 \cdot \lambda^2}$: trivial since the function of this transition is simply the identity, it follows from $s a$ being a play hence non-repetitive.

Request. If $t$ is a negative transition $\alpha \rightarrow \beta$ on the address $\ell_{\ell_{\_} \cdot w \cdot y \cdot Q^{-}}$ or $\ell_{\ell_{\_} \cdot r \cdot Q^{-}}$. In both cases, the transition function is again the identity, so we can conclude by the same argument.

Atomic operation. If $t$ arises from $w$ or $r$. The two cases being symmetric, we only show for $w$. From the Petri structure, we get that $\alpha = y \cup \{(\{e\}, d)^{\ominus_3}, (s, d')^{\ominus_2}\}$ and $\beta = y \cup \{(\{e\}, d)^{\ominus_2}, (\{e\}, \_)^{\ominus_6}\}$.

We show that new($t$) is fresh in $\rho$. For the token in location 6, which is always in new($t$), only the transition $w$ writes to 6, so if the tokil $(\{e\}, \_)^{\ominus_6}$ appeared before in $\rho$, it means that
there would be already a tokil \([e], d''\) in \(\text{Coll}(\rho)\). This is not possible because 3 is only fed via the negative transition on \(\ell, \ell, w, Q^-\). This means that Opponent would have played \((\ell, \ell, w, Q^-, [e], d'')\) which violates the fact that \(s\) is non-repetitive (if \(d = d''\)) or that \(s\) is a play (if \(d \neq d''\) as those moves are in conflict).

Finally, if \([e], d\) is in \(\text{Coll}(\rho)\), then it means that a previous instance of \(w\) or \(r\) produced it, which means that there must have been a tokil \([e], d\) for \(w\) or \([e], \bullet\) for \(r\). That implies there has been two Opponent moves on addresses of the form \(\ell, \ell, (-)\) with the same exponential address, which is not allowed by the game as they are all in conflict.

**Valid.** Consider a run \(\rho : \emptyset \xrightarrow{\sigma} \alpha\) with \(s\) a play, and a positive extension \(t : \alpha \xrightarrow{\sigma} \beta\). There are several cases depending on the address of \(a\):

- If \(a\) is on \(r_A\): easy since \(\rho\) 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 \(\ell, r_A\).
- If \(a\) is on \(\ell, r_Q\): same reasoning.
- If \(a\) is for instance on \(\ell, \ell, w, A\) (the case for \(r_i\) is similar). This means that in location 4, there must be a tokil \([e], \sqrt{'}\)\(^{\text{Q}}\). That tokil proves that, there must be an entry \((e, w, d)\) (for some \(d\)) in \(\text{Tr}(\rho)\). By Lemma C.39, we know that in \(s\) there must be justifying move \((\ell, \ell, w, [e], d)\). 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 \(\text{Tr}(\rho)\) cannot repeat twice the same exponential token.

\[\Box\]

**D. THE UNFOLDING**

We provide some detailed proofs of the unfolding to strategies.

**D.1 Construction of the Unfolding**

Fix a game \(A\), and a Petri strategy \(\sigma : A\). First, for a valid run \(\rho : \emptyset \xrightarrow{\sigma} \alpha\), we write \(\text{post}(\rho) = \alpha\).

If \(x \in \text{Hist}(\sigma)\), we write \(\text{post}(x) = \text{post}(\rho)\) for any \(\rho\) such that \(x = \text{IT}_\rho\). This is justified by:

**Lemma D.1.** Consider \(\rho : \emptyset \xrightarrow{\sigma} \alpha\) and \(\rho' : \emptyset \xrightarrow{\sigma} \alpha'\) valid runs such that \(\text{IT}_\rho = \text{IT}_{\rho'}\).

Then, \(\alpha = \alpha'\).

**Proof.** Exploiting strong safety, it is immediate by induction on \(\rho\) that:

\[\alpha = (\wp\{\text{post}(t) \mid t \in \text{IT}_\rho\}) \setminus (\wp\{\text{pre}(t) \mid t \in \text{IT}_\rho\})\]

from which the result immediately follows. \[\Box\]

We aim to prove that valid runs exactly correspond to linearizations of histories. The first step is:

**Lemma D.2.** Consider \(\rho\) a valid run of \(\sigma\) of the form \(\rho = \rho_0 \cdot (t \cup \alpha) \cdot (t' \cup \alpha')\).

If \(t\) is maximal in \(\text{IT}_\rho\), then \(\rho_0 \cdot (t' \cup \beta') \cdot (t \cup \beta)\) is a valid run for some \(\beta, \beta'\).

**Proof.** First, we show that for \(\beta' = \text{post}(\rho_0)\) \(\setminus \text{pre}(t')\), \(\rho_0\) extends by \(t' \cup \beta'\) which means showing:

\[
\begin{align*}
(1) \quad \beta' \cap \text{pre}(t') &= \emptyset \\
(2) \quad \text{pre}(t') &\subseteq \text{post}(\rho_0) \\
(3) \quad \beta' \cap \text{post}(t') &= \emptyset
\end{align*}
\]

First, (1) is by construction of \(\beta'\). For (2), consider \(e \in \text{pre}(t')\). Then \(e\) must either be in \(\alpha \subseteq \text{post}(\rho_0)\), or in \(\text{post}(t)\). If it is in \(\text{post}(t)\), then it cannot be a token produced by \(t\) (i.e. in new(t)) as \(t\) and \(t'\) are incomparable. So it must be in \(\text{pre}(t) \subseteq \text{post}(\rho_0)\) as desired. For (3), consider \(e \in \beta' \cap \text{post}(t')\). The tokil \(e\) must be in new(t'), which implies since \(t\) and \(t'\) are incomparable
that e does not appear in pre(t). Since e ∈ post(ρ₀), it must be that e ∈ α. Since α’ must be disjoint from post(t’), we have e ∈ pre(t’) which is absurd. Hence, ρ₁ = ρ₀’ ∪ (t’ ⊔ β’) is indeed a valid run.

We now let β = post(ρ₁) \ pre(t) and must prove (1) pre(t) ⊆ post(ρ₁); and (2) β ∩ post(t) = ∅. For (1), if e ∈ pre(t), then e is in post(ρ₀). As a result, either e is not in pre(t’), which implies that e ∈ β’ hence e ∈ post(ρ₁) (as desired), or e ∈ pre(t’) as well. In the second case, we have then e ∈ pre(t) ∩ pre(t’). Because, in ρ, t comes before t’, this implies that e cannot be eaten by t, in other words e ∈ post(t). This implies e ̸∈ eat(t’) as the two transitions are incomparable, i.e. e ∈ post(t’) ⊆ post(ρ₁). For (2), consider e ∈ β ∩ post(t), i.e. in particular e ∈ new(t). As e ∈ post(ρ₁), e either belongs to β’ or post(t’). In the first case, this means that e ∈ post(ρ₀) ∩ post(t), which can only be if e ∈ pre(t) which is absurd. In the second case, it means that e ∈ post(t’), which in turn means that e ∈ pre(t’) as e is produced by t so it cannot be produced by t’ as well by strong safety. But that is not possible either as it would imply a dependency from t to t’.

**Lemma D.3.** Consider x ∈ Hist(σ) and t a maximal element of x.

Then, there exists a valid run ρ ending in t ⊔ α (for some context α) such that ITₜ = x.

**Proof.** Consider a run ρ₀ spanning x, which must have the shape:

ρ₀ = ρ₁ ∪ (t ⊔ α) ∪ (t₁ ⊔ α₁) ∪ ⋯ ∪ (tₙ ⊔ αₙ).

We proceed by induction on n. If n = 0, then t 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 ITₓ = x \ {tₙ₊₁} and χ ends with t. By Lemma D.1, we have post(χ) = post(ρ’); from that it is immediate that χ ∪ (tₙ₊₁ ⊔ αₙ₊₁) is a valid run, and we conclude by Lemma D.2.

**Lemma D.4.** Consider x ∈ Hist(σ).

Then, valid runs ρ such that x = ITₜ exactly correspond to linearizations of x.

**Proof.** Clearly, all runs preserve ≤ₓ. For the converse, for any transition t maximal in T (x), we obtain a run where it is played last by taking any valid run ρ such that x = ITₜ, and pushing t to the end via local permutations – maximality of t ensures that there is no obstruction – see Lemma D.3. Iterating this process, we can indeed obtain any linearization.

**Proposition 5.10.** The set comprising all T (x) for x ∈ Hist(σ), is a rigid family written T (σ).

Moreover, T (σ) (ordered by rigid inclusion) is order-isomorphic to Hist(σ) (ordered by inclusion).

**Proof.** First, the claimed order-isomorphism is clear by construction.

**Rigid-closed.** Now if ρ ∈ T (σ) and q → ρ, by Lemma D.4 there is a valid run ρ playing q first. Truncating ρ after q, we get ρ’ such that T (ρ’) = q by construction.

**Binary-compatible.** Take X ≤₉ Hist(σ). Clearly, if (1) there are x, y ∈ X, visible instantiated transitions t in x and t’ in y labelled by conflicting events of A + B; or (2) there are x, y ∈ X, t : α →ₐ β in x and t’ : α’ →ₐ β’ in y such that α ∩ α’ ≠ ∅; then there cannot be a valid run witnessing ∪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 (ρₓ)ₓ∈X may be directly “zipped” into a valid run witnessing ∪X ∈ Hist(σ).

This concludes the proof, as it brings compatibility of X ≤₉ T (σ) to pairwise compatibility.

**Proposition 5.12.** The event structure U(σ) = Pr(T (σ)) ↓ Y, equipped with the display map

∂U(σ) : |U(σ)| → |A + B|

q → ∂ₐ(top(q))

is a strategy in the sense of Definition 4.6. Moreover, U(σ) is negative if σ is.
Theorem 5.15. Consider \( \sigma : A \vdash B \) and \( \tau : B \vdash C \) Petri strategies. Then, there is an order-isomorphism:

\[
\bigodot \circ (-) : \{ (x^\sigma, x^\tau) \in \mathcal{T}^+(\tau) \times \mathcal{T}^+(\sigma) \mid \text{causally compatible} \} \cong \mathcal{T}^+(\tau \circ \sigma)
\]

such that for \( x^\sigma \in \mathcal{T}^+(\sigma) \) and \( x^\tau \in \mathcal{T}^+(\tau) \) causally compatible, \( \partial_{\tau \circ \sigma}(x^\sigma \circ x^\tau) = x^\sigma_A \rightarrow x^\tau_C \).

Proof. For \( x^\sigma \in \mathcal{T}^+(\sigma) \) and \( x^\tau \in \mathcal{T}^+(\tau) \), we set \( x^\sigma \circ x^\tau \) as the set of instantiated transitions obtained from \( x^\sigma \) and \( x^\tau \) by the rules of Figure 28 (following Lemma C.3). We prove by induction that for all \( x^\sigma \in \mathcal{T}^+(\sigma) \) and \( x^\tau \in \mathcal{T}^+(\tau) \) causally compatible, then \( x^\sigma \circ x^\tau \in \mathcal{T}^+(\tau \circ \sigma) \), and

\[
\text{post}(x^\sigma \circ x^\tau) = \text{post}(x^\sigma) + \circ \text{post}(x^\tau).
\]

If \( x^\sigma \circ x^\tau \) is empty, there is nothing to prove. If \( x^\sigma \) or \( x^\tau \) have a maximal neutral instantiated transition, say \( x^\sigma \) with maximal \( t = \ell^\sigma(\mu) \in x^\sigma \). Then, setting \( y^\sigma = x^\sigma \setminus \{ t \} \) yields \( y^\sigma \in \mathcal{T}^+(\sigma) \) by Proposition 5.10; and with also \( y^\tau = x^\tau \), it is direct that \( y^\sigma \) and \( y^\tau \) are still causally compatible. By IH, we have \( y^\tau \circ y^\sigma \in \mathcal{T}^+(\tau \circ \sigma) \) and \( \text{post}(y^\tau \circ y^\sigma) = \text{post}(y^\tau) + \circ \text{post}(y^\sigma) \). This means that there is a run \( \rho : \emptyset \rightarrow_{\tau \circ \sigma} \alpha \) projecting to \( \rho_{\sigma} : \emptyset \rightarrow_{\tau} \sigma \circ \alpha \) with \( \alpha = \text{post}(y^\sigma) \). Since \( t \) is enabled in \( \text{post}(y^\sigma) \) it follows that \( t^\circ(\ell^\sigma(\mu)) \) is enabled in \( y^\tau \circ y^\sigma \), and

\[
\rho(t^\circ(\ell^\sigma(\mu))) : \emptyset \rightarrow_{\tau \circ \sigma} \beta
\]
where by construction $\beta = \rho = \rho(x^\sigma) +^\omicron \rho(x^\tau)$ as needed. The symmetric reasoning applies if $x^\tau$ has a maximal neutral instantiated transition – so assume all maximal transitions in $x^\sigma, x^\tau$ visible.

Now, by causal compatibility of $x^\sigma$ and $x^\tau$, there is an element of $x_A \parallel x_B \parallel x_C$ (following the notations of Section 5.3.1) which is maximal for $\prec$. If it is in $x_A$, it has the form $\delta^\eta(t)$ for $t \in x^\sigma$ positive or negative. In both cases, the same argument is 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

$$t^+(\mu) : \mu \overset{\xi^\sigma}{\rightarrow} \emptyset, \quad t^-(\langle s, d \rangle) : \emptyset \overset{\xi^\tau}{\rightarrow} v,$$

or the dual – symmetric – situation, respectively maximal in $x^\sigma$ and $x^\tau$; and by necessity $m = (m, s, d)$ where $\beta(t^+) = \rho m$, $\beta(t^-) = \eta m$, $\delta(t^+) = (s, d)$ and $\delta(t^-) = (s, d) = v$. Setting $y^\sigma \setminus \{t^+(\mu)\}$ and $y^\tau \setminus \{t^-(\langle s, d \rangle)\}$, it is straightforward that they are still causally compatible histories. By IH, $y^\sigma \circ y^\tau \in \mathcal{F}(\tau \circ \sigma)$ with $\rho(y^\sigma \circ y^\tau) = \rho(y^\sigma) +^\omicron \rho(y^\tau)$. It follows that there is a run $\rho : \emptyset \rightarrow \tau \circ \sigma \rho(y^\sigma \circ y^\tau)$.

Since $x^\sigma \in \mathcal{F}(\sigma)$ with $t^+(\mu)$ maximal and $x^\tau \in \mathcal{F}(\sigma)$ with $t^-(\langle s, d \rangle)$ maximal, there are

$$\xi^\sigma(t^+(\mu) \cup y^\sigma) : \emptyset \rightarrow \rho(y^\sigma), \quad \xi^\tau(t^-(\langle s, d \rangle) \cup y^\tau) : \emptyset \rightarrow \tau \rho(y^\tau),$$

valid runs by Lemma D.3, with $x^\sigma = \Pi \xi^\sigma$ and $x^\tau = \Pi \xi^\tau$. By Lemma D.1, $\rho(y^\sigma) = \rho(y^\sigma) = \rho(y^\sigma)$ and $\rho(y^\tau) = \rho(y^\tau)$. It follows that the transition

$$(t^+ \circ t^-)(\rho(y^\sigma)) : \rho(y^\sigma) \rightarrow \tau \circ \sigma \rho(y^\sigma)$$

is enabled in $\rho(y^\sigma \circ y^\tau)$, hence it can be appended to $\rho$, witnessing $x^\tau \circ x^\sigma \in \mathcal{F}(\tau \circ \sigma)$.

In the other direction, given $y \in \mathcal{F}(\tau \circ \sigma)$, consider a valid run $\rho : \emptyset \rightarrow \tau \circ \sigma \alpha$ such that $y = \Pi \rho$. By Lemmas C.4 and C.6, we then have $\alpha = \alpha_\sigma +^\omicron \alpha_\tau$ with

$$\rho_\sigma : \emptyset \rightarrow \alpha_\sigma \rho_\tau : \emptyset \rightarrow \tau \alpha_\tau$$

valid runs. Recall that $\rho_\sigma = \rho \uparrow \pi_\sigma$ and $\rho_\tau = \rho \uparrow \pi_\tau$ – hence, setting $x^\sigma = \Pi \rho_\sigma$ and $x^\tau = \Pi \rho_\tau$, we have $x^\sigma = \Pi \rho_\sigma$ and $x^\tau = \Pi \rho_\tau$, so that $x^\sigma \in \mathcal{F}(\sigma)$ and $x^\tau \in \mathcal{F}(\tau)$. Causal compatibility is direct as $\rho$ provides a linearization of $\prec$.

It is direct that these constructions are inverse; it remains to show that they preserve $\dagger$-covered histories. If $x^\sigma$ and $x^\tau$ causally compatible are $\dagger$-covered, then consider t maximal in $x^\sigma \odot x^\tau$. If $t = \rho(t)(\rho(y^\sigma))$, this directly contradicts $\dagger$-coveredness of $x^\sigma$, and likewise for $\rho(y^\sigma)$. If $t = \rho(t)(\rho(y^\tau))$, then $t^{-}(\langle s, d \rangle)$ is maximal in $x^\sigma$, contradiction – likewise for $\rho(y^\tau)$. If $t = (t^+ \circ t^-)(\rho(y^\sigma))$ with $t^+(\mu) \in x^\sigma$ and $t^-((s, d)) \in x^\tau$, then $t^-(\langle s, d \rangle)$ is maximal in $x^\tau$, contradiction – likewise, $t = \rho(t)(\rho(y^\sigma))$ leads to a contradiction. So, $x^\sigma \odot x^\tau$ is $\dagger$-covered.

Reciprocally, assume $x^\sigma \odot x^\tau$ $\dagger$-covered. Consider $t \in x^\sigma$ maximal. If $t = \rho(t)(\rho(y^\sigma))$, then since $x^\sigma \odot x^\tau$ is $\dagger$-covered, there is $t^{-}(\langle s, d \rangle) \rightarrow x^\sigma \odot x^\tau \rho(t^-)$ and a direct case analysis shows that $\pi_\sigma t'$ is defined with $t^-((s, d)) \rightarrow x^\sigma \pi_\sigma t'$, contradicting the maximality of $t$. The last case has $t$ positive; and symmetrically, $x^\tau$ is $\dagger$-covered. □

As detailed in Proposition 5.16, it follows that unfolding preserves composition up to iso.

Next, we show the same for copyscat. If $A$ is an arena, then we have obvious bijections

$$\IT_{\times A}^+ \cong [A \vdash A]^+ \quad \IT_{\times A}^- \cong [A \vdash A]^-, \quad \IT_{\times A}^+ \cong [A \vdash A]^+ \quad \IT_{\times A}^- \cong [A \vdash A]^-, \quad \IT_{\times A}^+ \cong [A \vdash A]^+ \quad \IT_{\times A}^- \cong [A \vdash A]^-, \quad \IT_{\times A}^+ \cong [A \vdash A]^+ \quad \IT_{\times A}^- \cong [A \vdash A]^-, \quad \IT_{\times A}^+ \cong [A \vdash A]^+ \quad \IT_{\times A}^- \cong [A \vdash A]^-$,$$

and coercing silently through these, we have:

**Lemma D.5.** Consider $A$ an arena, and $\rho : \emptyset \rightarrow \aleph_A \alpha$ a valid run.

Then, $\IT_{\rho} = \play(\rho)$, with negative maximal transitions in bijection with $\alpha$. 

PROOF. Straightforward by induction on $\rho$. □

LEMMA D.6. Consider $A$ an arena. Then, we have the order-isomorphism

$$\mathcal{T}^+(\mathcal{C}_A) \cong \{ x \vdash x \mid x \in \mathcal{C}(A) \}$$

with $\partial_{\mathcal{C}_A}(x \vdash x) = x \vdash x$.

PROOF. The isomorphism simply applies the bijection $\text{IT}_{\mathcal{C}_A} = |A|$. From left to right, recall first that by Lemma C.8, for $\rho : \emptyset \longrightarrow_{\mathcal{C}_A} \alpha$ a valid run, we have

$$|\text{play}(\rho)| = x \vdash y, \quad y \supseteq x \cap y \supseteq x, \quad \alpha = (y^\top \setminus x) \cup (x^\top \setminus y).$$

By Lemma D.5, the history $\text{IT}_\rho$ is +-covered iff $\alpha = \emptyset$, i.e. $y^\top \supseteq x$ and $x^\top \subseteq y$. But as $y \supseteq x \cap y \supseteq x$ this entails $x = y$. In that case, $\partial_{\mathcal{C}_A}(\text{IT}_\rho) = |\text{play}(\rho)| = x \vdash x$ as needed. Reciprocally, for any $x \in \mathcal{C}(A)$, it is straightforward to build a valid run $\rho : \emptyset \longrightarrow_{\mathcal{C}_A} \emptyset$ s.t. $\text{IT}_\rho = x \vdash x$ as required. □

From that, preservation of copycat follows:

PROPOSITION D.7. Consider $A$ an arena. Then, $\mathcal{U}(\mathcal{C}_A) \cong \mathcal{C}_A$.

PROOF. We compose label-preserving order-isomorphisms:

$$\mathcal{C}^+(\mathcal{U}(\mathcal{C}_A)) \cong \mathcal{T}^+(\mathcal{C}_A) \cong \mathcal{C}(A) \cong \mathcal{C}^+(\mathcal{C}_A)$$

by Lemmas 5.13 and D.6. From this it follows that $\mathcal{U}(\mathcal{C}_A) \cong \mathcal{C}_A$ by Lemma 4.9. □

COROLLARY D.8. We have a functor of precategories $\mathcal{U} : \text{PStrat} \rightarrow \text{Strat}$.

D.3 The Unfolding Preserves Operations

Next, we prove that the unfolding preserves all operations of the IPA-structure.

D.3.1 Tensor. Preservation of the tensor operation is easy via the following observation:

LEMMA D.9. Consider $\sigma : A_1 + B_1, \tau : A_2 + B_2$ Petri strategies. Then, we have an order-isomorphism

$$(- \otimes -) : \mathcal{T}^+(\sigma) \times \mathcal{T}^+(\tau) \cong \mathcal{T}^+(\sigma \otimes \tau)$$

s.t. $\partial_{\sigma \otimes \tau}(x^\sigma \otimes x^\tau) = (x_{A_1} \otimes x_{A_2}) \vdash (x_{B_1} \otimes x_{B_2})$ where $\partial_{\sigma}(x^\sigma) = x_{A_1} \vdash x_{B_1}$ and $\partial_{\tau}(x^\tau) = x_{A_2} \vdash x_{B_2}$.

PROOF. Consider $x^\sigma \in \mathcal{T}^+(\sigma)$ and $x^\tau \in \mathcal{T}^+(\tau)$. By definition, there are valid runs

$$\rho^\sigma : \emptyset \longrightarrow_{\sigma} \alpha_{\sigma}, \quad \rho^\tau : \emptyset \longrightarrow_{\tau} \alpha_{\tau}$$

such that $x^\sigma = \text{IT}_{\rho^\sigma}$ and $x^\tau = \text{IT}_{\rho^\tau}$. We define the history $x^\sigma \otimes x^\tau$ as

$$x^\sigma \otimes x^\tau = \{ t^\sigma(1^0)(\ell^\sigma(\mu)) \mid t^\sigma(1^0)(\mu) \in x^\sigma \}$$
$$\cup \{ t^\sigma(1^-)(\ell^\sigma(s,d)) \mid t^-((s,d)) \in x^\sigma \}$$
$$\cup \{ t^\sigma(1^0)(\ell^\sigma(s,d)) \mid t^0((s,d)) \in x^\sigma \}$$
$$\cup \{ t^\tau(1^-)(\ell^\tau(s,d)) \mid t^-((s,d)) \in x^\tau \}$$

This must be the history of a valid run – to show that, we build

$$t^\sigma(\rho^\sigma) : \emptyset \longrightarrow_{\sigma \otimes \tau} t^\sigma(\alpha_{\sigma}), \quad t^\tau(\rho^\tau) \cup t^\sigma(\alpha_{\sigma}) : t^\sigma(\alpha_{\sigma}) \longrightarrow_{\sigma \otimes \tau} t^\sigma(\alpha_{\sigma}) \cup t^\sigma(\alpha_{\tau})$$

which by concatenation (and Lemma C.13) yields a valid run $\rho^\sigma \otimes \rho^\tau : \emptyset \longrightarrow_{\sigma \otimes \tau} \alpha_{\sigma} \cup \alpha_{\tau}$; and it is immediate that $x^\tau \otimes x^\sigma = \text{IT}_{\rho^\tau \otimes \rho^\sigma}$. By definition of the causal ordering of instantiated transitions, it is also immediate that $x^\tau \otimes x^\sigma$ is +-covered; and that this preserves the labelling.
Reciprocally, for any \( x \in \mathcal{T}^+(\sigma \otimes \tau) \) we consider the projections
\[
x^\sigma = \pi_\sigma(x), \quad x^\tau = \pi_\tau(x),
\]
and it follows from Lemma C.14 that \( x^\sigma \in \mathcal{T}(\sigma) \) and \( x^\tau \in \mathcal{T}(\tau) \). From the definition of the causal ordering of instantiated transitions, \( x^\sigma \) and \( x^\tau \) are still \(+\)-covered.

Finally, these two transformations are inverses as required. \( \square \)

Again, from this we can conclude that the unfolding preserves the tensor.

**Corollary D.10.** Consider \( \sigma : A_1 \vdash B_1, \tau : A_2 \vdash B_2 \) Petri strategies.
Then, we have \( \mathcal{U}(\sigma \otimes \tau) \cong \mathcal{U}(\sigma) \otimes \mathcal{U}(\tau) \).

**Proof.** We compose label-preserving isomorphisms:
\[
\mathcal{C}^+(\mathcal{U}(\sigma \otimes \tau)) \cong \mathcal{T}^+(\sigma \otimes \tau)
\cong \mathcal{T}^+(\sigma) \times \mathcal{T}^+(\tau)
\cong \mathcal{C}^+(\mathcal{U}(\sigma)) \times \mathcal{C}^+(\mathcal{U}(\tau))
\cong \mathcal{C}^+(\mathcal{U}(\sigma) \otimes \mathcal{U}(\tau))
\]
by Lemmas 5.13, D.9, Lemma 5.13 again, and Proposition B.5. \( \square \)

### D.3.2 Renaming

Before detailing the unfolding of currying and promotion, we show that it preserves renaming. We have already established in Lemma C.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 D.11.** Consider \( A, B \) games, \( h = [f, (g_m)] : A \rightsquigarrow B, \sigma : A, \) and \( \rho : \emptyset \rightarrow_{\sigma} \alpha \) valid.
For all \( t, t' \in \mathcal{T}_\rho \), we have \( t \leq_\rho t' \iff t[h] \leq_\rho[h] t'[h] \).

**Proof.** Consider \( t \rightarrow_\rho t' \). W.l.o.g., we assume that this dependency cannot be deduced otherwise by transitivity. By Lemma D.4, we can assume that \( t \) and \( t' \) appear subsequently in \( \rho \).

Assume first \( t \rightarrow_A t' \). By definition,
\[
t : \alpha \xrightarrow{a}_{\sigma} \beta, \quad t' : \alpha' \xrightarrow{a'}_{\sigma} \beta'.
\]
with \( a \rightarrow_A a' \), while by construction, \( t[h] : \alpha \xrightarrow{ha}_{\sigma[h]} \beta \) and \( t'[h] : \alpha' \xrightarrow{ha'}_{\sigma[h]} \beta' \). Now, we distinguish cases depending on the polarity of \( a, a' \). If \( \text{pol}_A(a) = + \) or \( \text{pol}_A(a') = - \), then by courtesy we have \( ha \rightarrow_B ha' \), so that \( t[h] \rightarrow_B t'[h] \) by Definition 5.8. If \( \text{pol}_A(a) = - \) and \( \text{pol}_A(a') = + \), then
\[
t = t^-((s, \Delta)) : \emptyset \xrightarrow{a}_{\sigma} \beta, \quad t' = t^+(\alpha') : \alpha' \xrightarrow{a'}_{\sigma} \emptyset.
\]
Assume, seeking a contradiction, that \( \beta \cap \alpha' = \emptyset \), and consider the prefix of \( \rho \):
\[
\rho'(t^-((s, \Delta)) \cup \gamma)(t^+(\alpha') \cup \gamma') : \emptyset \rightarrow_{\sigma} v,
\]
but if indeed \( \beta \cap \alpha' = \emptyset \), then \( t \) and \( t' \) permute as in
\[
\rho'(t^-((\alpha') \cup \mu)(t^-(s, \Delta)) \cup \mu') : \emptyset \rightarrow_{\sigma} v
\]
and by valid, this entails play(\( \rho' \alpha' \in \text{Plays}(A) \)), contradicting \( a \rightarrow_A a' \). Assume now \( t \rightarrow_\rho t' \). This comes either from new(\( t \) \cap \text{pre}(t') \neq \emptyset, or post(\( t \) \cap \text{eat}(t') \neq \emptyset. But the renaming of instantiated transitions does not change pre- and post-conditions, so \( t[h] \rightarrow_{\rho[h]} t'[h] \) still.

Reciprocally, assume \( t[h] \leq_{\rho[h]} t'[h] \). Seeking a contradiction, assume \( \neg(t \leq_\rho t') \). By Lemma D.4, we can assume that \( t' \) appears before \( t \) in \( \rho \). Hence, \( t'[h] \) appears before \( t[h] \) in \( \rho[h] \). But by Lemma C.19 \( \rho[h] \) is valid, so by Lemma D.4 \( t[h] \) must appear before \( t'[h] \), contradiction. \( \square \)

Next, we need to show that valid runs are also reflected by renamings.
Lemma D.12. Consider $A, B$ games, $\sigma : A \Rightarrow [f, (g_m)] : A \rightsquigarrow B$, and $\rho' : \emptyset \rightarrow_{\sigma} [h] \Rightarrow \alpha$ valid.
Then, there is a unique $\rho : \emptyset \rightarrow_{\alpha} \alpha$ valid such that $\rho' = \rho[h]$.

Proof. By induction on $\rho'$. If it is empty, this is clear. Consider $\rho'[\emptyset] \rightarrow_{\sigma} [h]$ and $t^0 = t^0(\emptyset] \uplus \gamma : \alpha \rightarrow_{\sigma} [h] \Rightarrow \beta$, with $t^0(\emptyset] : \mu \rightarrow_{\sigma} [h] \Rightarrow v$. By IH, there is $\rho : \emptyset \rightarrow_{\alpha} \alpha$. By definition, we still have $t^0(\emptyset] \uplus \gamma : \alpha \rightarrow_{\sigma} \beta$, so $\rho[t^0] : \emptyset \rightarrow_{\alpha} \beta$ and as required, $(\rho[t^0])[h] = \rho'[\emptyset]$.

Next, consider $\rho'[t^*] : \emptyset \rightarrow_{\sigma} [h] \Rightarrow \alpha$ and $t^* = t^*(\emptyset] \uplus \gamma : \alpha \rightarrow_{\sigma} [h] \Rightarrow \beta$, with $t^*(\emptyset] : \mu \rightarrow_{\sigma} [h] \Rightarrow 0$. Necessarily, $b' = (f[m], d', s')$ for $m = \partial_{\sigma}(t^*)$ and $(d', s') = g_m(d, s)$ for $(d, s) = \delta(t^*(\emptyset)] - s - b = h(b)$ for $b = (m, d, s)$. But then, by definition, $t^*(\emptyset] : \mu \rightarrow_{\sigma} \emptyset$ as well, so $t^* : \alpha \rightarrow_{\sigma} \beta$. By IH, there is $\rho' : \emptyset \rightarrow_{\alpha} \alpha$ such that $\rho[h] = \rho'$. By valid, $\rho t^*$ is still valid, and $(\rho t^*)[h] = \rho'[t^*]$.

Finally, consider $\rho'[t] \rightarrow_{\sigma} \alpha$ with $t' \in \partial_{\sigma} \alpha$ and $t = t'((s', d')) \uplus \gamma : \alpha \rightarrow_{\sigma} [h] \Rightarrow$, with $\partial_{\sigma}(t) = \emptyset \rightarrow_{\sigma} [h] \Rightarrow$. Here, necessarily, $b' = (m', s', d')$ where $m' = f(m)$, $m = \partial_{\sigma}(t^-)$, $(s', d') = g_m(s, d)$. In other words, $b' = h(b)$ with $b = (m, s, d)$. Now, by IH we have $\rho : \emptyset \rightarrow_{\alpha} \alpha$ with $\rho[h] = \rho'$. Besides, we have the transition $t^-((s, d)) : \emptyset \rightarrow_{\sigma} \alpha$, so that $(t^-((s, d)) \uplus \gamma) : \alpha \rightarrow_{\sigma} \beta$. Its validity follows immediately from receptivity of global renamings (and the fact that they are injective). It is clear that $(\rho(t^-((s, d)) \uplus \gamma))[h] = \rho'[t^-]$, as required.

Finally, uniqueness is immediate by induction and injectivity of $h$. \hfill $\square$

Lemma D.13. Consider $A, B$ games, $\sigma : A \Rightarrow \text{a Petri strategy}$, and $h = [f, (g_m)] : A \rightsquigarrow B$. Then, $\mathcal{U}(\sigma[h]) \cong \mathcal{U}(\sigma)[h]$.

Proof. We construct an order-isomorphism

$$-[h] : \mathcal{F}^+(\sigma) \cong \mathcal{F}^+(\sigma[h])$$

such that $\partial_{\sigma[h]}(x) = h(\partial_{\sigma}(x))$ for all $x \in \mathcal{F}^+(\sigma)$. Given $x \in \mathcal{F}^+(\sigma)$, there is some $\rho$ a valid run in $\sigma$ such that $x = \mathcal{F}(\rho)$. By Lemma C.19, $\rho[h]$ is a valid run of $\sigma[h]$, so we may consider $x[h] = \mathcal{F}(\rho[h])$. By Lemma D.11, $-[h] \Rightarrow$ on instantiated transitions is an order-isomorphism $-[h] : x \Rightarrow x[h]$ on $\mathcal{F}^+(\sigma[h])$. Together with Lemma D.12, this easily entails that we get $-[h] : \mathcal{F}(\sigma) \cong \mathcal{F}(\sigma[h])$ an order-isomorphism. By definition, for each $x \in \mathcal{F}(\sigma)$ we have an order-iso $x \cong x[h]$ 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 5.13, we also obtain an order-isomorphism

$$-[h] : \mathcal{E}^+(\mathcal{U}(\sigma)) \cong \mathcal{E}^+(\mathcal{U}(\sigma[h]))$$

such that $\partial_{\mathcal{U}(\sigma[h])}(x[h]) = h(\partial_{\mathcal{U}(\sigma)}(x))$ for any $x \in \mathcal{E}^+(\mathcal{U}(\sigma))$, so, from Definition 3.14, an order-isomorphism $-[h] : \mathcal{E}^+(\mathcal{U}(\sigma)[h]) \cong \mathcal{E}^+(\mathcal{U}(\sigma[h]))$ such that $\partial_{\mathcal{U}(\sigma[h])}(x[h]) = h(\partial_{\mathcal{U}(\sigma)}(x))$ for all $x \in \mathcal{E}^+(\mathcal{U}(\sigma)[h])$. By Lemma 5.13, it follows that $\mathcal{U}(\sigma)[h] \cong \mathcal{U}(\sigma)[h]$. $\square$

D.3.3 Currying. Follows from Lemma D.13, as currying is obtained with the same global renaming both in PStrat and in Strat.

D.3.4 Promotion. Let us start with characterizing $+$-covered traces of the functorial promotion.

Lemma D.14. Consider $\sigma : A \Rightarrow B$ a Petri strategy. Then, we have an order-iso

$$-[\cdot] : \text{Fam}(\mathcal{F}^+, x^0(\sigma)) \cong \mathcal{F}^+(\mathcal{E}(\sigma))$$

satisfying that for all $(x^e)_{e \in E} \in \text{Fam}(\mathcal{F}^+, x^0(\sigma))$, we have

$$\partial_{\sigma}([x^e]_{e \in E}) = \left(\biguplus_{e \in E} x^e_A \equiv x^e_{A_e} \right) \uplus \left(\biguplus_{e \in E} x^e_B \right)$$

writing $x^e = x^e_A \uplus x^e_B$ for all $e \in E$. 

Proof. This is a $n$-ary adaptation of the proof of Lemma D.9. Consider $(x^e)_{e \in E} \in \text{Fam}(\mathcal{T}^+,(\mathcal{T}^+)^\top(\sigma))$. By definition, there is a valid run
\[ \rho^e : \emptyset \xrightarrow{\alpha^e} \]
for all $e \in E$ such that $x^e = \text{IT}_{\rho^e}$. We define the history $[(x^e)_{e \in E}]$ as
\[ [x^e \mid e \in E] = \{ t^0(\{ e :: \alpha^e \} \mid e \in E, t^0(\{ \alpha^e \} \in x^e) \}
\cup \{ t^+(\{ e :: \alpha^e \} \mid e \in E, t^+(\{ \alpha^e \} \in x^e) \}
\cup \{ t^-(\{ (e :: s, d) \} \mid e \in E, t^-(\{ (s, d) \} \in x^e) \} .
\]

We construct a run $\rho$ obtained by concatenating all $\rho^e$s in the obvious way as in Lemma D.9. Exploiting Lemma C.27, it is a valid run and $[x^e \mid e \in E] = \text{IT}_{\rho}$ by construction. By definition of the causal ordering of instantiated transitions, it is also immediate that $[x^e \mid e \in E]$ is $+$-covered; and that this preserves the labelling. Reciprocally, for any $x \in \mathcal{T}^+(!\sigma)$, we consider the projections
\[ x^e = \pi_x^e(x) \]
and it follows from Lemma C.28 that $x^e \in \mathcal{T}^+(!\sigma)$ for all $e \in E$. From the definition of the causal ordering of instantiated transitions, each $x^e$ is still $+$-covered.

Finally, these two transformations are inverses as required. \hfill $\square$

Corollary D.15. Consider $\sigma : !A + B$ a Petri strategy. Then, we have $\mathcal{U}(\sigma^\top) \cong \mathcal{U}(\sigma)^\top$.

Proof. We exploit the following sequence of label-preserving order-isomorphisms:
\[ \mathcal{C}^+(\mathcal{U}(\sigma^\top)) \cong \mathcal{T}^+(\sigma^\top) \]
\[ = \mathcal{T}^+((!\sigma)[\text{dig}+\text{id}]) \]
\[ = \mathcal{T}^+(!\sigma) \]
\[ \cong \text{Fam}(\mathcal{T}^+,(\mathcal{T}^+)^\top(\sigma)) \]
\[ \cong \text{Fam}(\mathcal{T}^+,(\mathcal{U}(\sigma)^\top)) \]
using first Lemma 5.13, then via a direct verification as in Proposition C.32, then applying (5), followed by Lemma D.14, and then Lemma 5.13 – 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 $\mathcal{U}(\sigma^\top) \cong \mathcal{U}(\sigma)^\top$ from Proposition B.9. \hfill $\square$

D.4 The Unfolding Preserves Primitives


D.4.2 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.

D.4.3 Fixpoint. We prove the following proposition:

Proposition D.16. For any well-opened arena $O$, $\mathcal{U}(Y_O) \cong Y_O$.

Proof. Using Lemmas 4.9 and 5.13, the required isomorphism boils down to an order-iso
\[ \mathcal{T}^+(Y_O) \cong \mathcal{C}^+(Y_O) \]

preserving display maps. We build it using Lemmas C.35 and B.15. 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 $(J, z, (y_h)_{h \in J})$. We can construct a set of itransitions realising $x$ as follows:
• For each $((m, s, d)^- \in z$ we include the itransition $x_m\{(s, d)\} = (\ell, \rho, \{((\ell :: s, d)^{-m^-}) \})$
• For each \((m, s, d)^+ \in z\), we include the transitions \(\ell \cdot_m \cdot (\ell \cdot_m : s, d)\) and \(\nu_m \cdot_m ((\ell \cdot_m : s, d)^{\otimes m^*})\).
• For each \((m, s_0, d)^- \in y_{e-s}\), we include the transitions \(\nu_m \cdot_m ((e \cdot (s) : s_0, d))\) and \(\nu_m \cdot_m (((e \cdot (s) :: s_0, d)^{\otimes m^*}))\).
• For each \((m, s_0, d)^+ \in y_{e-s}\), we include the transitions \(\nu_m \cdot_m (\nu (e :: s) : s_0, d))\) and \(\nu_m \cdot_m (\nu (e :: s) :: s_0, d)^{\otimes m^*})\).

and it is easy to see that this set of transitions is reachable by a valid run of \(Y_{\O} \). □

\[ \]

\section*{D.4.4 Contraction} The reasoning follows a similar and simpler route as for the fixpoint operator.

\section*{D.4.5 Let bindings} We first characterise the configurations of the strategy interpreting lets.

\textbf{Lemma D.17.} The \(+\)-covered configurations of let are order-isomorphic to tuples \((l \subseteq \E, x \in \C(X), y, y' \in \C(Y))\) such that:

1. \(y \neq \emptyset\) if \(x \neq \emptyset\)
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 \(((\nu_{e=I}(e :: x)) \rightarrow y') \otimes x \rightarrow y\).

\textbf{Lemma D.18.} We have \(\mathcal{U}(\text{let}) \cong \text{let}\).

\textbf{Proof.} As before we rely on Lemmas 4.9 and 5.13 to build the isomorphism. Lemma C.37 together with Lemma D.17 induce an injective map from \(\mathcal{T}^+(\text{let})\) into \(\C^+(\text{let})\). We show it is surjective by constructing a set of transitions of let from \(x \in \C^+(\text{let})\) corresponding to a \(\langle I, x, y, y' \rangle\).

- If \(y \neq \emptyset\), then we have transitions \(\nu \cdot_m \cdot (\nu \cdot_m : ([], \bullet))\) and \(\ell \cdot_m \cdot (\nu \cdot_m : (\nu \cdot_m : ([], \bullet)^{\otimes 1}))\).
- If \(x\) has an event \((\nu \cdot_m : ([], d))\) maximal in \(X\), then we have the two transitions \(\ell \cdot_m \cdot (\nu \cdot_m : (\nu \cdot_m : ([], d))\) and \(\ell \cdot_m \cdot (\nu \cdot_m : (\nu \cdot_m : ([], d)^{\otimes 2}))\).
- For every \(e \in I\), we have three transitions \(\ell \cdot_m \cdot (\nu \cdot_m : (\nu \cdot_m : ([e], \bullet))\), \(s \cdot_m (([e], \bullet)^{\otimes 4}, ([e], d)^{\otimes 5})\), and \(\ell \cdot_m \cdot (\nu \cdot_m : (([e], d))^{\otimes 6})\) where \(d\) is the value of the maximal event in \(x\).
- If \(y'\) has a maximal \((\nu \cdot_m : ([], d))\), we have transitions \(\ell \cdot_m \cdot (\nu \cdot_m : (\nu \cdot_m : ([], d))\) and \(\nu \cdot_m \cdot (\nu (e :: x)) \rightarrow w) \rightarrow z\). □

\section*{D.4.6 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 C.3.6 correspond to \(+\)-covered configurations of newref:

\textbf{Lemma D.19.} \(\text{There is an order-isomorphism between } \C^+(\text{precell})\) and the set of consistent memory traces ordered by prefix.

\textbf{Proof.} Direct from the definition of precell. □

We now show the main result:

\textbf{Proposition D.20.} \(\mathcal{U}(\text{newref}) \cong \text{newref}\).

\textbf{Proof.} By Lemmas 4.9 and 5.13, this amounts to building an order-iso:

\[
\mathcal{T}^+(\text{newref}) \cong \C^+(\text{newref}).
\]

From left-to-right. We focus on non-empty histories and configurations and use characterisation of \(\C^+(\text{newref})\) from Proposition B.19.

Consider a non-empty history \(y \in \mathcal{T}^+(\text{newref})\). It is reached by a run \(\rho : \emptyset \rightarrow x \rightarrow a\). By Lemma C.39, we know that there \(\text{Tr}(\rho)\) is a consistent memory trace, and that \(s\) must have the shape \(((\nu_{e=I}(e :: x_e)) \rightarrow w) \rightarrow z\). Since \(y\) is \(+\)-covered, we observe that \(w = z\). We can thus map \(y\) to \(\langle \text{Tr}(\rho), w \rangle \in \C^+(\text{newref})\) using the isomorphism of Proposition B.19. Note that by the side
conditions of Lemma C.39, the family \((x_e)_{e \in I}\) is entirely determined by \(\text{Tr}(\rho)\): \(I\) matches the length of \(\text{Tr}(\rho)\) and each \(x_e\) is a two-event configuration corresponding to the memory operation \(\text{Tr}(\rho)_e\).

The last check it to show that this does not depend on the particular run \(\rho\) chosen. Clearly \(x\) only depends on \(|s|\). For \(\text{Tr}(\rho)\), 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 \((w, e, d)\) if \(w\{([e], d)@3, \_\} \in y\) and \((r, e, d)\) if \(r\{([e], \_\} @5, (\_, d)@2\} \in y\). 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 \(\text{Tr}(\rho)\).

From right-to-left. Consider now a \(\langle \rho, x \rangle \in \mathcal{C}^+\) (newref) where \(\rho\) is a consistent memory trace. We can build a history \(y\) containing the following instantiated transitions:

- The initial negative itransition on \(\nu\mathcal{Q}([\_], \_\)\).
- The positive itransition \(\ell, \ell\mathcal{Q}([\_], \_\)\).
- If \(x\) contains a move \((A, [\_], d)\), then the transitions \(\ell, \nu\mathcal{A}^-([\_], d)\) and \(\nu\mathcal{A}^+([\_], d)@7\).
- If \(\rho_i = (w, e, d)\) then the transitions \(\ell, \ell, \omega, \mathcal{Q}^-([e], d)\), \(w\{([e], d) @3, \{[e'], d'\}@2\}\) where:
  - \(e'\) is the exponential token of \(\rho_{i-1}\) (or \([\_]\) if \(i = 0\)), and \(d'\) the value observed by \(\rho_{i-1}\) (or zero if \(i = 0\)); and \(\ell, \ell, \omega, \mathcal{A}^+([e], \_\)@4\).
- And similarly if \(\rho_i = (r, e, d)\).

From this description, it is easy to build a valid run reaching \(y\) establishing that \(y \in \mathcal{T}\) (newref). An easy verification shows that all maximal itransitions in \(y\) are positive. □