Resource-Tracking Concurrent Games [⋆]

Aurore Alcolei, Pierre Clairambault, and Olivier Laurent

Université de Lyon, ENS de Lyon, CNRS, UCB Lyon 1, LIP
`{Aurore.Alcolei, Pierre.Clairambault, Olivier.Laurent}@ens-lyon.fr`

Abstract. We present a framework for game semantics based on concurrent games, that keeps track of *resources* as data modified throughout execution but not affecting its control flow. Our leading example is *time*, yet the construction is in fact parametrized by a *resource bimonoid* $\mathcal{R}$, an algebraic structure expressing resources and the effect of their consumption either sequentially or in parallel. Relying on our construction, we give a sound resource-sensitive denotation to $\mathcal{R}$-IPA, an affine higher-order concurrent programming language with shared state and a primitive for resource consumption in $\mathcal{R}$. Compared with general operational semantics parametrized by $\mathcal{R}$, our resource analysis turns out to be finer, leading to non-adequacy. Yet, our model is not degenerate as adequacy holds for an operational semantics specialized to time.
In regard to earlier semantic frameworks for tracking resources, the main novelty of our work is that it is based on a non-interleaving semantics, and as such accounts for *parallel* use of ressources accurately.

1 Introduction

Since its inception, *denotational semantics* has grown into a very wide subject. Its developments now cover numerous programming languages or paradigms, using approaches that range from the extensionality of *domain semantics* [24] (recording the input-output behaviour) to the intensionality of *game semantics* [1,17] (recording execution traces, formalized as *plays* in a 2-players game between the program ("Player") and its execution environment ("Opponent")). Denotational semantics has had significant influence on the theory of programming languages, with contributions ranging from program logics or reasoning principles to new language constructs and verification algorithms.

Most denotational models are *qualitative* in nature, meaning that they ignore efficiency of programs in terms of time, or other resources such as power or bandwith. To our knowledge, the first denotational model to cover time was Ghica's *slot games* [13], an extension of Ghica and Murawski's fully abstract model for a higher-order language with concurrency and shared state [14]. Slot games exploit the intensionality of game semantics and represent time via special moves called *tokens* matching the *ticks* of a clock. They are fully abstract *w.r.t.* the notion of observation in Sands' operational theory of *improvement* [26].

⋆ Supported by project Elica (ANR-14-CE25-0005) and 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).

More recently, there has been a growing interest in capturing quantitative aspects denotationally. Laird *et al* constructed [18] an enrichment of the relational model of Linear Logic [11], using weights from a *resource semiring* given as parameter. This way, they capture in a single framework several notions of resources for extensions of PCF, ranging from time to probabilistic weights. Two type systems with similar parametrizations were introduced simultaneously by, on the one hand, Ghica and Smith [15] and, on the other hand, Brunel, Gaboardi *et al* [4]; the latter with a quantitative realizability denotational model.

In this paper, we give a resource-sensitive denotational model for $\mathcal{R}$-*IPA*, an affine higher-order programming language with concurrency, shared state, and with a primitive for resource consumption. With respect to slot games our model differs in that our resource analysis accounts for the fact that resource consumption may combine differently in parallel and sequentially – simply put, we mean to express that $\mathbf{wait}(1) \parallel \mathbf{wait}(1)$ may terminate in 1 second, rather than 2. We also take inspiration from weighted relational models [18] in that our construction is parametrized by an algebraic structure representing resources and their usage. Our *resource bimonoids* $\langle \mathcal{R}, 0, ;, \parallel, \leq \rangle$ differ however significantly from their resource semiring $\langle \mathcal{R}, 0, 1, +, \cdot \rangle$: while $;$ matches $\cdot$, $\parallel$ is a new operation expressing the consumption of resources in parallel. We have no counterpart for the $+$, which agglomerates distinct non-deterministically co-existing executions leading to the same value: instead our model keeps them separate.

Capturing parallel resource usage is technically challenging, as it can only be attempted relying on a representation of execution where parallelism is explicit. Accordingly, our model belongs to the family of *concurrent* or *asynchronous* game semantics pioneered by Abramsky and Melliès [2], pushed by Melliès [20] and later with Mimram [22], and by Faggian and Piccolo [12]; actively developed in the past 10 years prompted by the introduction of a more general framework by Rideau and Winskel [25,7]. In particular, our model is a refinement of the (qualitative) truly concurrent interpretation of *affine IPA* described in [5]. Our methodology to record resource usage is inspired by game semantics for first-order logic [19,3] where moves carry first-order terms from a signature – instead here they carry explicit *functions*, *i.e.* terms up to a congruence (it is also reminiscent of Melliès' construction of the free dialogue category over a category [21]).

As in [5] we chose to interpret an affine language: this lets us focus on the key phenomena which are already at play, avoiding the technical hindrance caused by replication. As suggested by recent experience with concurrent games [6,10], we expect the developments presented here to extend transparently in the presence of *symmetry* [8,9]; this would allow us to move to the general (non-affine) setting.

Outline. We start Section 2 by introducing the language $\mathcal{R}$-IPA. We equip it first with an interleaving semantics and sketch its interpretation in slot games. We then present resource bimonoids, give a new parallel operational semantics, and hint at our truly concurrent games model. In Section 3, we construct this model and prove its soundness. Finally in Section 4, we show adequacy for an operational semantics specialized to time, noting first that the general parallel operational semantics is too coarse *w.r.t.* our model.

2 From $\mathcal{R}$-IPA to $\mathcal{R}$-Strategies

2.1 Affine IPA

Terms and Types. We start by introducing the basic language under study, *affine Idealized Parallel Algol* (IPA). It is an affine variant of the language studied in [14], a call-by-name concurrent higher-order language with shared state. Its **types** are given by the following grammar:

$$A, B ::= \mathbf{com} \mid \mathbf{bool} \mid \mathbf{mem}_W \mid \mathbf{mem}_R \mid A \multimap B$$

Here, $\mathbf{mem}_W$ is the type of *writeable* references and $\mathbf{mem}_R$ is the type of *readable* references; the distinction is necessary in this affine setting as it allows to share accesses to a given state over subprocesses; this should make more sense in the next paragraph with the typing rules. In the sequel, non-functional types are called **ground types** (for which we use notation $\mathbb{X}$). We define terms directly along with their typing rules in Figure 1. **Contexts** are simply lists $x_1 : A_1, \dots, x_n : A_n$ of variable declarations (in which each variable occurs at most once), and the exchange rule is kept implicit. Weakening is not a rule but is admissible. We comment on a few aspects of these rules.

Firstly, observe that the reference constructor $\mathbf{new}\, x, y\, \mathbf{in}\, M$ binds two variables x and y, one with a write permission and the other with a read permission. In this way, the permissions of a shared state can be distributed in different components of *e.g.* an application or a parallel composition, causing interferences despite the affine aspect of the language. Secondly, the assignment command, $M := \mathbf{tt}$, seems quite restrictive. Yet, the language is affine, so a variable can only be written to once, and, as we choose to initialize it to $\mathbf{ff}$, the only useful thing to write is $\mathbf{tt}$. Finally, many rules seem restrictive in that they apply only at ground type $\mathbb{X}$. More general rules can be defined as syntactic sugar; for instance we give (all other constructs extend similarly): $M;_{A \multimap B} N = \lambda x^A. \,(\, M;_B\, (N\, x))$.

Operational Semantics. We fix a countable set L of **memory locations**. Each location ℓ comes with two associated variable names ℓ_W and ℓ_R distinct from other variable names. Usually, stores are partial maps from L to $\{\mathbf{tt}, \mathbf{ff}\}$. Instead, we find it more convenient to introduce the notion of **state** of a memory location. A state corresponds to a history of memory actions (reads or writes) and follows the *state diagram* of Figure 2 (ignoring for now the annotations with α, β). We write $(\mathsf{M}, \leq_{\mathsf{M}})$ for the induced set of states and accessibility relation on it. For each $m \in \mathsf{M}$, its set of **available actions** is $\mathrm{act}(m) = \{W, R\} \setminus m$ (the letters not occurring in m, annotations being ignored); and its **value** (in $\{\mathbf{tt}, \mathbf{ff}\}$) is $\mathrm{val}(m) = \mathbf{tt}$ iff W occurs in m.

$$\begin{tikzcd}
& \varepsilon \arrow[dl] \arrow[dr] & \\
R^{\alpha} \arrow[d] & & W^{\alpha} \arrow[d] \\
R^{\alpha} W^{\beta} & & W^{\alpha} R^{\beta}
\end{tikzcd}$$

Fig. 2: State diagram

Finally, a **store** is a partial map $s : \mathsf{L} \to \mathsf{M}$ with finite domain, mapping each memory location to its current state. To each store corresponds a *typing context*

$$\Omega(s) = \{\ell_X : \mathbf{mem}_X \mid \ell \in \mathrm{dom}(s) \;\&\; X \in \mathrm{act}(s(\ell))\}.$$

$$
\frac{}{\Gamma \vdash \mathbf{skip} : \mathbf{com}} \qquad
\frac{}{\Gamma \vdash \mathbf{tt} : \mathbf{bool}} \qquad
\frac{}{\Gamma \vdash \mathbf{ff} : \mathbf{bool}} \qquad
\frac{}{\Gamma \vdash \bot : \mathbb{X}} \qquad
\frac{(x : A) \in \Gamma}{\Gamma \vdash x : A}
$$

$$
\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.\, M : A \multimap B} \qquad
\frac{\Gamma \vdash M : A \multimap B \qquad \Delta \vdash N : A}{\Gamma, \Delta \vdash M\, N : B} \qquad
\frac{\Gamma \vdash M : \mathbf{mem}_R}{\Gamma \vdash {!M} : \mathbf{bool}}
$$

$$
\frac{\Gamma \vdash M : \mathbf{com} \qquad \Delta \vdash N : \mathbb{X}}{\Gamma, \Delta \vdash M;\, N : \mathbb{X}} \qquad
\frac{\Gamma \vdash M : \mathbf{com} \qquad \Delta \vdash N : \mathbb{X}}{\Gamma, \Delta \vdash M \parallel N : \mathbb{X}} \qquad
\frac{\Gamma \vdash M : \mathbf{mem}_W}{\Gamma \vdash M := \mathbf{tt} : \mathbf{com}}
$$

$$
\frac{\Gamma \vdash M : \mathbf{bool} \qquad \Delta \vdash N_1 : \mathbb{X} \qquad \Delta \vdash N_2 : \mathbb{X}}{\Gamma, \Delta \vdash \mathbf{if}\, M\, N_1\, N_2 : \mathbb{X}} \qquad
\frac{\Gamma, x : \mathbf{mem}_W, y : \mathbf{mem}_R \vdash M : \mathbb{X}}{\Gamma \vdash \mathbf{new}\, x, y\, \mathbf{in}\, M : \mathbb{X}}
$$

Fig. 1: Typing rules for affine IPA

The operational semantics operates on **configurations** defined as pairs $\langle M, s\rangle$ with s a store and $\Gamma \vdash M : A$ a term whose free variables are all memory locations with $\Gamma \subseteq \Omega(s)$. This property will be preserved by our rather standard small-step, call-by-name operational semantics. We refrain for now from giving the details, they will appear in Section 2.2 in the presence of resources.

2.2 Interleaving Cost Semantics, and $\mathcal{R}$-IPA

Ghica and Murawski [14] have constructed a *fully abstract*(for may-equivalence) model for (non-affine) IPA, relying on an extension of Hyland-Ong games [17].

Their model takes an *interleaving* view of the execution of concurrent programs: a program is represented by the set of all its possible executions, as decided non-deterministically by the scheduler. In game semantics, this is captured by lifting the standard requirement that the two players alternate. For instance, Figure 3 shows a *play* in the interpretation of the program $x : \mathbf{com}, y : \mathbf{bool} \vdash x \parallel y : \mathbf{bool}$. The diagram is read from top to bottom, chronologically. Each line comprises one computational event ("move"), annotated with "$-$" if due to the execution environment ("Opponent") and with "$+$" if due to the program ("Player"); each move corresponds to a certain type component, under which it is placed. With the first move $\mathbf{q}^-$, the environment initiates the computation. Player then plays $\mathbf{run}^+$, triggering the evaluation of x. In standard game semantics, the control would then go back to the execution environment – Player would be stuck until Opponent plays. Here instead, due to parallelism Player can play a second move $\mathbf{q}^+$ immediately. At this point of execution, x and y are both running in parallel. Only when they have both returned (moves $\mathbf{done}^-$ and $\mathbf{tt}^-$) is Player able to respond $\mathbf{tt}^+$, terminating the

$$
\begin{array}{cccc}
x : \mathbf{com}, & y : \mathbf{bool} & \vdash & \mathbf{bool} \\
 & & & \mathbf{q}^- \\
\mathbf{run}^+ & & & \\
 & \mathbf{q}^+ & & \\
 & \mathbf{tt}^- & & \\
\mathbf{done}^- & & & \\
 & & & \mathbf{tt}^+
\end{array}
$$

Fig. 3: A non-alternating play

$$\begin{array}{rlrl}
\langle \mathbf{skip};\, M, s, \alpha\rangle & \to \langle M, s, \alpha\rangle & \langle (\lambda x.\, M)\, N, s, \alpha\rangle & \to \langle M[N/x], s, \alpha\rangle \\
\langle \mathbf{skip} \parallel M, s, \alpha\rangle & \to \langle M, s, \alpha\rangle & \langle !\ell_R, s, \alpha\rangle & \to \langle \mathrm{val}(s(\ell)), s[\ell \mapsto s(\ell).R^{\alpha}], \alpha\rangle \\
\langle M \parallel \mathbf{skip}, s, \alpha\rangle & \to \langle M, s, \alpha\rangle & \langle \ell_W := \mathbf{tt}, s, \alpha\rangle & \to \langle \mathbf{skip}, s[\ell \mapsto s(\ell).W^{\alpha}], \alpha\rangle \\
\langle \mathbf{if\,tt}\, N_1\, N_2, s, \alpha\rangle & \to \langle N_1, s, \alpha\rangle & \langle \mathbf{new}\, x, y\, \mathbf{in}\, M, s, \alpha\rangle & \to \langle M[\ell_W/x, \ell_R/y], s \uplus \{\ell \mapsto \varepsilon\}, \alpha\rangle \\
\langle \mathbf{if\,ff}\, N_1\, N_2, s, \alpha\rangle & \to \langle N_2, s, \alpha\rangle & \langle \mathbf{consume}(\beta), s, \alpha\rangle & \to \langle \mathbf{skip}, s, \alpha; \beta\rangle
\end{array}$$

Fig. 5: Operational semantics: basic rules

computation. The full interpretation of $x : \mathbf{com}, y : \mathbf{bool} \vdash x \parallel y : \mathbf{bool}$, its *strategy*, comprises numerous plays like that, one for each interleaving.

As often in denotational semantics, Ghica and Murawski's model is invariant under reduction: if $\langle M, s\rangle \to \langle M', s'\rangle$, both have the same denotation. The model adequately describes the result of computation, but not its *cost* in terms, for instance, of time. Of course this cost is not yet specified: one must, for instance, define a *cost model* assigning a cost to all basic operations (*e.g.* memory operations, function calls, *etc*). In this paper we instead enrich the language with a primitive for *resource consumption* – cost models can then be captured by inserting this primitive concomitantly with the costly operations (see for example [18]).

$$\frac{(\alpha \in \mathcal{R})}{\Gamma \vdash \mathbf{consume}(\alpha) : \mathbf{com}}$$

Fig. 4: Typing **consume**

$\mathcal{R}$-IPA. Consider a set $\mathcal{R}$ of **resources**. The language $\mathcal{R}$-IPA is obtained by adding to affine IPA a new construction, $\mathbf{consume}(\alpha)$, typed as in Figure 4. When evaluated, $\mathbf{consume}(\alpha)$ triggers the consumption of resource $\mathcal{R}$. Time consumption will be a running example throughout the paper. In that case, we will consider the non-negative reals $\mathbb{R}_+$ as set $\mathcal{R}$, and for $t \in \mathbb{R}_+$ we will use $\mathbf{wait}(t)$ as a synonym for $\mathbf{consume}(t)$.

To equip $\mathcal{R}$-IPA with an operational semantics we need operations on $\mathcal{R}$, they are introduced throughout this section. First we have $0 \in \mathcal{R}$, the null resource ; if $\alpha, \beta \in \mathcal{R}$, we have some $\alpha; \beta \in \mathcal{R}$, the resource taken by consuming α, then β – for $\mathcal{R} = \mathbb{R}_+$, this is simply addition. To evaluate $\mathcal{R}$-IPA, the **configurations** are now triples $\langle M, s, \alpha\rangle$ with $\alpha \in \mathcal{R}$ tracking resources already spent. With that, we give in Figure 5 the basic operational rules. The only rule affecting current resources is that for $\mathbf{consume}(\beta)$, the others leave it unchanged. However note that we store the current state of resources when performing memory operations, explaining the annotations in Figure 2. These annotations do not impact the operational behaviour, but will be helpful in relating with the game semantics in Section 3. As usual, these rules apply within call-by-name evaluation contexts – we omit the details here but they will appear for our final operational semantics.

Slot Games. In [13], Ghica extends Ghica and Murawski's model to *slot games* in order to capture resource consumption. Slot games introduce a new action called a *token*, representing an atomic resource consumption, and written $\textcircled{\$}$ – writing $\textcircled{n}$ for n successive occurrences of $\textcircled{\$}$. A model of $\mathbb{N}_+$-IPA using slot games would have for instance the play in Figure 6 in the interpretation of

$$H = (\mathbf{wait}(1);\, x;\, \mathbf{wait}(2)) \parallel (\mathbf{wait}(2);\, y;\, \mathbf{wait}(1))$$

in context $x : \mathbf{com}, y : \mathbf{bool}$, among with many others. Note, in examples, we use a more liberal typing rule for '$;$' allowing $y^{\mathbf{bool}}; z^{\mathbf{com}} : \mathbf{bool}$ to avoid clutter: it can be encoded as $\mathbf{if}\, y\, (z;\, \mathbf{tt})\, (z;\, \mathbf{ff})$. Following the methodology of game semantics, the interpretation of $(\lambda xy.\, H)\, \mathbf{skip}\, \mathbf{tt}$ would yield, by composition, the strategy with only maximal play $\mathbf{q}^{-}⑥\mathbf{tt}^{+}$, where ⑥ reflects the overall 6 time units (say "seconds") that have to pass in total before we see the result (3 in each thread). This seems wasteful, but it is indeed an adequate computational analysis, because both slot games and the operational semantics given so far implicitely assume a sequential operational model, *i.e.* that both threads compete to be scheduled on a *single* processor. Let us now question that assumption.

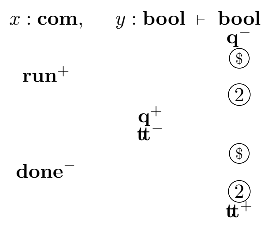

Fig. 6: A play with tokens

Parallel Resource Consumption. With a truly concurrent evaluation in mind, we should be able to prove that the program above may terminate in 3 *seconds*, rather than 6; as nothing prevents the threads from evaluating in parallel. Before we update the operational semantics to express that, we enrich our resource structure to allow it to express the effect of consuming resources in parallel.

We now introduce the full algebraic structure we require for resources.

Definition 1. *A* ***resource bimonoid*** *is* $\langle \mathcal{R}, 0, ;, \parallel, \leq \rangle$ *where* $\langle \mathcal{R}, 0, ;, \leq \rangle$ *is an ordered monoid,* $\langle \mathcal{R}, 0, \parallel, \leq \rangle$ *is an ordered commutative monoid,* 0 *is bottom for* $\leq$*, and* $\parallel$ *is* ***idempotent****,* i.e. *it satisfies* $\alpha \parallel \alpha = \alpha$*.*

A resource bimonoid is in particular a *concurrent monoid* in the sense of *e.g.* [16] (though we take $\leq$ in the opposite direction: we read $\alpha \leq_{\mathcal{R}} \alpha'$ as "α is *better/more efficient* than α'"). Our *Idempotence* assumption is rather strong as it entails that $\alpha \parallel \beta$ is the supremum of $\alpha, \beta \in \mathcal{R}$. This allows to recover a number of simple laws, *e.g.* $\alpha \parallel \beta \leq \alpha; \beta$, or the exchange rule $(\alpha; \beta) \parallel (\alpha'; \beta') \leq (\alpha \parallel \alpha'); (\beta \parallel \beta')$. Idempotence, which would not be needed for a purely functional language, is used crucially in our interpretation of state.

Our leading examples are $\langle \mathbb{N}_{+}, 0, +, \max, \leq \rangle$ and $\langle \mathbb{R}_{+}, 0, +, \max, \leq \rangle$ – we call the latter the *time bimonoid.* Others are the *permission bimonoid* $\langle \mathcal{P}(P), \emptyset, \cup, \cup, \subseteq \rangle$ for some set P of *permissions*: if reaching a state requires certain permissions, it does not matter whether these have been requested sequentially or in parallel; the bimonoid of *parametrized time* $\langle \mathcal{M}, 0, ;, \parallel, \leq \rangle$ with $\mathcal{M}$ the monotone functions from positive reals to positive reals, 0 the constant function, $\parallel$ the pointwise maximum, and $(f;\, g)(x) = f(x) + g(x + f(x))$: it tracks time consumption in a context where the time taken by $\mathbf{consume}(\alpha)$ might grow over time.

Besides time-based bimonoids, it would be appealing to cover resources such as *power*, *bandwith* or *heapspace.* Those, however, clearly fail idempotence of $\parallel$, and are therefore not covered. It is not clear how to extend our model to those.

Parallel Operational Semantics. Let us fix a resource bimonoid $\mathcal{R}$. To express parallel resource consumption, we use the many-step *parallel reductions* defined

$$\frac{}{\langle M, s, \alpha\rangle \rightrightarrows \langle M, s, \alpha\rangle} \qquad \frac{\langle M, s, \alpha\rangle \to \langle M', s', \alpha'\rangle}{\langle M, s, \alpha\rangle \rightrightarrows \langle M', s', \alpha'\rangle} \qquad \frac{\langle M, s, \alpha\rangle \rightrightarrows \langle M', s', \alpha'\rangle}{\langle C[M], s, \alpha\rangle \rightrightarrows \langle C[M'], s', \alpha'\rangle}$$

$$\frac{\langle M, s, \alpha\rangle \rightrightarrows \langle M', s', \alpha'\rangle \qquad \langle M', s', \alpha'\rangle \rightrightarrows \langle M'', s'', \alpha''\rangle}{\langle M, s, \alpha\rangle \rightrightarrows \langle M'', s'', \alpha''\rangle} \qquad \frac{\langle M, s, \alpha\rangle \rightrightarrows \langle M', s', \alpha'\rangle \qquad \langle N, s, \alpha\rangle \rightrightarrows \langle N', s'', \alpha''\rangle}{\langle M \parallel N, s, \alpha\rangle \rightrightarrows \langle M' \parallel N', s' \uparrow s'', \alpha' \parallel \alpha''\rangle}$$

Fig. 7: Rules for parallel reduction

in Figure 7, with **call-by-name evaluation contexts** given by

$$C[] ::= [] \mid []\,N \mid [];\,N \mid \mathbf{if}\,[]\,N_1\,N_2 \mid [] := \mathbf{tt} \mid ![] \mid ([] \parallel N) \mid (M \parallel [])$$

The rule for parallel composition carries some restrictions regarding memory: M and N can only reduce concurrently if they do not access the same memory cells. This is achieved by requiring that the *partial* operation $s \uparrow s'$ – that intuitively corresponds to "merging" two memory stores s and s' whenever there are no conflicts – is defined. More formally, the partial order $\leq_{\mathsf{M}}$ on memory states induces a partial order (also written $\leq_{\mathsf{M}}$) on stores, defined by $s \leq_{\mathsf{M}} s'$ iff $\mathrm{dom}(s) \subseteq \mathrm{dom}(s')$ and for all $\ell \in \mathrm{dom}(s)$ we have $s(\ell) \leq_{\mathsf{M}} s'(\ell)$. This order is a cpo in which s' and s'' are *compatible* (*i.e.* have an upper bound) iff for all $\ell \in \mathrm{dom}(s') \cap \mathrm{dom}(s'')$, $s'(\ell) \leq_{\mathsf{M}} s''(\ell)$ or $s''(\ell) \leq_{\mathsf{M}} s'(\ell)$ – so there has been no interference going to s' and s'' from their last common ancestor. When compatible, $s' \uparrow s''$ maps s' and s'' to their lub, and is undefined otherwise.

For $\vdash M : \mathbf{com}$, we set $M \Downarrow_\alpha$ if $\langle M, \emptyset, 0\rangle \rightrightarrows \langle \mathbf{skip}, s, \alpha\rangle$. For instance, instantiating the rules with the time bimonoid, we have

$$(\mathbf{wait}(1);\, \mathbf{wait}(2)) \parallel (\mathbf{wait}(2);\, \mathbf{wait}(1)) \Downarrow_3$$

2.3 Non-Interleaving Semantics

To capture this parallel resource usage semantically, we build on the games model for affine IPA presented in [5]. Rather than presenting programs as collections of *sequences* of moves expressing all observable sequences of computational actions, this model adopts a *truly concurrent* view using collections of *partially ordered* plays. For each Player move, the order specifies its *causal dependencies*, *i.e.* the Opponent moves that need to have happened before. For instance, ignoring the subscripts, Figure 8 displays a typical partially ordered play in the strategy for the term H of Section 2.2. One partially ordered play does not fully specify a sequential execution: that in Figure 8 stands for *many* sequential executions, one of which is in Figure 3. Behaviours expressed by partially ordered plays are deterministic *up to* choices of the scheduler irrelevant for the eventual result. Because $\mathcal{R}$-IPA is non-deterministic (via concurrency and shared state), our strategies will be *sets* of such partial orders.

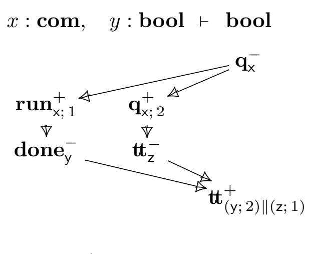
$x : \mathbf{com}, \quad y : \mathbf{bool} \vdash \mathbf{bool}$
$\mathbf{q}^-_{\mathsf{x}}$
$\mathbf{run}^+_{\mathsf{x};\,1}$
$\mathbf{q}^+_{\mathsf{x};\,2}$
$\mathbf{done}^-_{\mathsf{y}}$
$\mathbf{tt}^-_{\mathsf{z}}$
$\mathbf{tt}^+_{(\mathsf{y};\,2)\parallel(\mathsf{z};\,1)}$

Fig. 8: A parallel $\mathcal{R}$-play

To express resources, we leverage the causal information and indicate, in each partially ordered play and for each positive move, an $\mathcal{R}$-expression representing its *additional cost* in function of the cost of its negative dependencies. Figure 8 displays such a $\mathcal{R}$-*play*: each Opponent move introduces a fresh variable, which can be used in annotations for Player moves. As we will see further on, once applied to strategies for values **skip** and **tt** (with no additional cost), this $\mathcal{R}$-play will answer to the initial Opponent move $\mathbf{q}_{\mathsf{x}}^-$ with $\mathbf{tt}_{\mathsf{x};\,\alpha}^+$ where $\alpha = (1; 2) \parallel (2; 1) =_{\mathbb{R}_+} 3$, as prescribed by the more efficient parallel operational semantics.

We now go on to define formally our semantics.

3 Concurrent Game Semantics of IPA

3.1 Arenas and $\mathcal{R}$-Strategies

Arenas. We first introduce *arenas*, the semantic representation of types in our model. As in [5], an arena will be a certain kind of *event structure* [27].

Definition 2. *An* ***event structure*** *comprises* $(E, \leq_E, \#_E)$ *where* E *is a set of* events, $\leq_E$ *is a partial order called* causal dependency, *and* $\#_E$ *is an irreflexive symmetric binary relation called* conflict, *subject to the two axioms:*

$$\begin{array}{l} \forall e \in E, [e]_E = \{e' \in E \mid e' \leq_E e\} \text{ is finite} \\ \forall e_1 \#_E e_2, \forall e_1 \leq_E e_1', e_1' \#_E e_2 \end{array}$$

We will use some vocabulary and notations from event structures. A **configuration** $x \subseteq E$ is a down-closed, consistent (*i.e.* for all $e, e' \in x$, $\neg(e \#_E e')$) finite set of events. We write $\mathscr{C}(E)$ for the set of configurations of E. We write $\rightarrowtriangle_E$ for **immediate causality**, *i.e.* $e \rightarrowtriangle_E e'$ iff $e <_E e'$ with nothing in between – this is the relation represented in diagrams such as Figure 8. A conflict $e_1 \#_E e_2$ is **minimal** if for all $e_1' <_E e_1$, $\neg(e_1' \#_E e_2)$ and symmetrically. We write $e_1 \sim_E e_2$ to indicate that e_1 and e_2 are in minimal conflict.

With this, we now define arenas.

Definition 3. *An* ***arena*** *is* $(A, \leq_A, \#_A, \mathrm{pol}_A)$*, an event structure along with a* ***polarity function*** $\mathrm{pol}_A : A \longrightarrow \{-, +\}$ *subject to:* (1) $\leq_A$ *is forest-shaped,* (2) $\rightarrowtriangle_A$ *is alternating: if* $a_1 \rightarrowtriangle_A a_2$*, then* $\mathrm{pol}_A(a_1) \neq \mathrm{pol}_A(a_2)$*, and* (3) *it is race-free,* i.e. *if* $a_1 \sim_A a_2$*, then* $\mathrm{pol}_A(a_1) = \mathrm{pol}_A(a_2)$.

Arenas present the computational actions available on a type, following a call-by-name evaluation strategy. For instance, the observable actions of a closed term on **com** are that it can be ran, and it may terminate, leading to the arena $\mathbf{com} = \mathbf{run}^- \rightarrowtriangle \mathbf{done}^+$. Likewise, a boolean can be evaluated, and can terminate on **tt** or **ff**, yielding the arena on the right of Figure 9 (when drawing arenas, immediate causality is written with a dotted line, from top to bottom). We present some simple arena constructions. The **empty**

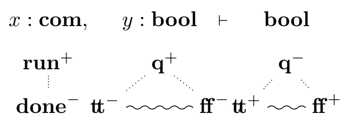

Fig. 9: An arena for a sequent

arena, written 1, has no events. If A is an arena, then its **dual** $A^\perp$ has the same components, but polarity reversed. The **parallel composition** of A and B, written $A \parallel B$, has as events the tagged disjoint union $\{1\} \times A \cup \{2\} \times B$, and all other components inherited. For $x_A \in \mathscr{C}(A)$ and $x_B \in \mathscr{C}(B)$, we also write $x_A \parallel x_B \in \mathscr{C}(A \parallel B)$. Figure 9 displays the arena $\mathbf{com}^\perp \parallel \mathbf{bool}^\perp \parallel \mathbf{bool}$.

$\mathcal{R}$-Augmentations. As hinted before, $\mathcal{R}$-strategies will be collections of partially ordered plays with resource annotations in $\mathcal{R}$, called *$\mathcal{R}$-augmentations.*

Definition 4. *An **augmentation** [5] on arena A is a finite partial order $\mathbb{q} = (|\mathbb{q}|, \leq_\mathbb{q})$ such that $\mathscr{C}(\mathbb{q}) \subseteq \mathscr{C}(A)$ (concerning configurations, augmentations are considered as event structures with empty conflict), which is **courteous**, in the sense that for all $a_1 \rightarrowtriangle_\mathbb{q} a_2$, if $\mathrm{pol}_A(a_1) = +$ or $\mathrm{pol}_A(a_2) = -$, then $a_1 \rightarrowtriangle_A a_2$.*

*A $\mathcal{R}$-**augmentation** also has (with $[a]^-_\mathbb{q} = \{a' \leq_\mathbb{q} a \mid \mathrm{pol}_A(a') = -\}$)*

$$\lambda_\mathbb{q} : (a \in |\mathbb{q}|) \quad \longrightarrow \quad \left(\mathcal{R}^{[a]^-_\mathbb{q}} \to \mathcal{R}\right)$$

such that if $\mathrm{pol}_A(a) = -$, then $\lambda_\mathbb{q}(a)(\rho) = \rho_a$, the projection on a of $\rho \in \mathcal{R}^{[a]^-_\mathbb{q}}$, and for all $a \in |\mathbb{q}|$, $\lambda_\mathbb{q}(a)$ is monotone w.r.t. *all of its variables.*

We write $\mathcal{R}$-Aug(A) for the set of $\mathcal{R}$-augmentations on A.

If $\mathbb{q}, \mathbb{q}' \in \mathcal{R}\text{-Aug}(A)$, $\mathbb{q}$ is **rigidly embedded** in $\mathbb{q}'$, or a **prefix** of $\mathbb{q}'$, written $\mathbb{q} \hookrightarrow \mathbb{q}'$, if $|\mathbb{q}| \in \mathscr{C}(\mathbb{q}')$, for all $a, a' \in |\mathbb{q}|$, $a \leq_\mathbb{q} a'$ iff $a \leq_{\mathbb{q}'} a'$, and for all $a \in |\mathbb{q}|$, $\lambda_\mathbb{q}(a) = \lambda_{\mathbb{q}'}(a)$. The *$\mathcal{R}$-plays* of Section 2.3 are formalized as $\mathcal{R}$-augmentations: Figure 8 presents an $\mathcal{R}$-augmentation on the arena of Figure 9. The functional dependency in the annotation of positive events is represented by using the free variables introduced alongside negative events, however this is only a symbolic representation: the formal annotation is a function for each positive event. In the model of $\mathcal{R}$-IPA, we will only use the particular case where the annotations of positive events only depend on the annotations of their immediate predecessors.

$\mathcal{R}$-Strategies. We start by defining $\mathcal{R}$-strategies on arenas.

Definition 5. *A $\mathcal{R}$-**strategy** on A is a non-empty prefix-closed set of $\mathcal{R}$-augmentations $\sigma \subseteq \mathcal{R}\text{-Aug}(A)$ which is **receptive** [5]: for $\mathbb{q} \in \sigma$ such that $|\mathbb{q}|$ extends with $a^- \in A$ (*i.e.* $\mathrm{pol}(a) = -$, $a \notin |\mathbb{q}|$, and $|\mathbb{q}| \cup \{a\} \in \mathscr{C}(A)$), there is $\mathbb{q} \hookrightarrow \mathbb{q}' \in \sigma$ such that $|\mathbb{q}'| = |\mathbb{q}| \cup \{a\}$.*

If σ is a $\mathcal{R}$-strategy on arena A, we write $\sigma : A$.

Observe that $\mathcal{R}$-strategies are fully described by their *maximal* augmentations, *i.e.* augmentations that are the prefix of no other augmentations in the strategy. Our interpretation of new will use the $\mathcal{R}$-strategy $\mathsf{cell} : [\![\mathbf{mem}_W]\!] \parallel [\![\mathbf{mem}_R]\!]$ (with arenas presented in Figure 10), comprising all the $\mathcal{R}$-augmentations rigidly included in either of the two from Figure 11. These two match the race when reading and writing simultaneously: if both $\mathbf{w}\mathbf{tt}^-$ and $\mathbf{r}^-$ are played the read may return $\mathbf{tt}^+$ or $\mathbf{ff}^+$, but it can only return $\mathbf{tt}^+$ in the presence of $\mathbf{w}\mathbf{tt}^-$.

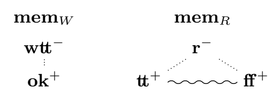

Fig. 10: $[\![\mathbf{mem}_W]\!]$ and $[\![\mathbf{mem}_R]\!]$

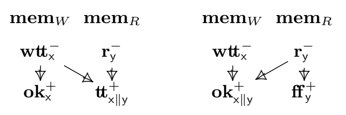
$\mathbf{mem}_W$ $\mathbf{mem}_R$
$\mathbf{wtt}_x^-$ $\mathbf{r}_y^-$
$\mathbf{ok}_x^+$ $\mathbf{tt}^+_{x\|y}$
$\mathbf{mem}_W$ $\mathbf{mem}_R$
$\mathbf{wtt}_x^-$ $\mathbf{r}_y^-$
$\mathbf{ok}^+_{x\|y}$ $\mathbf{ff}_y^+$

Fig. 11: Maximal $\mathcal{R}$-augmentations of cell

3.2 Interpretation of $\mathcal{R}$-IPA

Categorical Structure. In order to define the interpretation of terms of $\mathcal{R}$-IPA as $\mathcal{R}$-strategies, a key step is to show how to form a *category* of $\mathcal{R}$-strategies. To do that we follow the standard idea of considering **$\mathcal{R}$-strategies from A to B** to be simply $\mathcal{R}$-strategies on the compound arena $A^\perp \parallel B$. As usual, our first example of a $\mathcal{R}$-strategy between arenas is the *copycat $\mathcal{R}$-strategy*.

Definition 6. *Let A be an arena. We define a partial order $\leq_{\mathbb{C}_A}$ on $A^\perp \parallel A$:*

$$\begin{aligned}\leq_{\mathbb{C}_A} = (&\{((1,a),(1,a')) \mid a \leq_A a'\} \cup \{((2,a),(2,a')) \mid a \leq_A a'\} \cup \\ &\{((1,a),(2,a)) \mid \mathrm{pol}_A(a) = +\} \cup \{((2,a),(1,a)) \mid \mathrm{pol}_A(a) = -\})^+\end{aligned}$$

where $(-)^+$ denotes the transitive closure of a relation. Note that if $a \in A^\perp \parallel A$ is positive, it has a unique immediate predecessor $\mathrm{pred}(a) \in A^\perp \parallel A$ *for $\leq_{\mathbb{C}_A}$.*

If $x \parallel y \in \mathscr{C}(A^\perp \parallel A)$ is down-closed for $\leq_{\mathbb{C}_A}$ (write $\leq_{x,y}$ for the restriction of $\leq_{\mathbb{C}_A}$ to $x \parallel y$), we define an $\mathcal{R}$-augmentation $\mathbb{q}_{x,y} = (x \parallel y, \leq_{x,y}, \lambda_{x,y})$ where

$$\lambda_{x,y} : (a \in x \parallel y) \quad \longrightarrow \quad \left(\mathcal{R}^{[a]^-_{x\|y}} \to \mathcal{R}\right)$$

with $\lambda_{x,y}(a^-)(\rho) = \rho_a$, and $\lambda_{x,y}(a^+)(\rho) = \rho_{\mathrm{pred}(a)}$. Then, $\mathbb{c}_A$ is the $\mathcal{R}$-strategy comprising all $\mathbb{q}_{x,y}$ for $x \parallel y \in \mathscr{C}(A^\perp \parallel A)$ down-closed in A.

We first define *interactions* of $\mathcal{R}$-augmentations, extending [5].

Definition 7. *We say that $\mathbb{q} \in \mathcal{R}\text{-Aug}(A^\perp \parallel B)$, and $\mathbb{p} \in \mathcal{R}\text{-Aug}(B^\perp \parallel C)$ are **causally compatible** if $|\mathbb{q}| = x_A \parallel x_B$, $|\mathbb{p}| = x_B \parallel x_C$, and the preorder $\leq_{\mathbb{p}\circledast\mathbb{q}}$ on $x_A \parallel x_B \parallel x_C$ defined as $(\leq_\mathbb{q} \cup \leq_\mathbb{p})^+$ is a partial order.*

Say $e \in x_A \parallel x_B \parallel x_C$ is negative *if it is negative in $A^\perp \parallel C$. We define*

$$\lambda_{\mathbb{p}\circledast\mathbb{q}} : (e \in x_A \parallel x_B \parallel x_C) \quad \longrightarrow \quad \left(\mathcal{R}^{[e]^-_{\mathbb{p}\circledast\mathbb{q}}} \to \mathcal{R}\right)$$

as follows, by well-founded induction on $<_{\mathbb{p}\circledast\mathbb{q}}$, for $\rho \in \mathcal{R}^{[e]^-_{\mathbb{p}\circledast\mathbb{q}}}$:

$$\lambda_{\mathbb{p}\circledast\mathbb{q}}(e)(\rho) = \begin{cases} \lambda_\mathbb{p}(e)\left(\langle \lambda_{\mathbb{p}\circledast\mathbb{q}}(e')(\rho) \mid e' \in [e]^-_\mathbb{p}\rangle\right) & \textit{if } \mathrm{pol}_{B^\perp\|C}(e) = +, \\ \lambda_\mathbb{q}(e)\left(\langle \lambda_{\mathbb{p}\circledast\mathbb{q}}(e')(\rho) \mid e' \in [e]^-_\mathbb{q}\rangle\right) & \textit{if } \mathrm{pol}_{A^\perp\|B}(e) = +, \\ \rho_e & \textit{otherwise, } \text{i.e. } e \textit{ negative} \end{cases}$$

*The **interaction** $\mathbb{p} \circledast \mathbb{q}$ of compatible $\mathbb{q}, \mathbb{p}$ is $(x_A \parallel x_B \parallel x_C, \leq_{\mathbb{p}\circledast\mathbb{q}}, \lambda_{\mathbb{p}\circledast\mathbb{q}})$.*

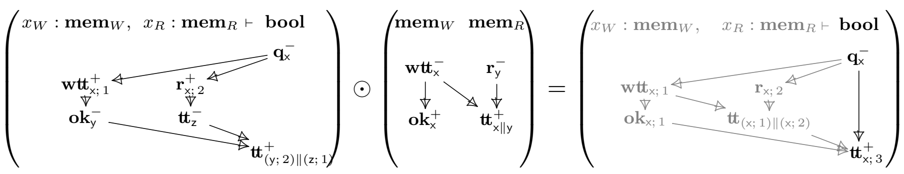
$x_W : \mathbf{mem}_W,\ x_R : \mathbf{mem}_R \vdash \mathbf{bool}$
$\mathbf{q}_{\mathsf{x}}^-$
$\mathbf{wtt}_{\mathsf{x};\,1}^+$
$\mathbf{r}_{\mathsf{x};\,2}^+$
$\mathbf{ok}_{\mathsf{y}}^-$
$\mathbf{tt}_{\mathsf{z}}^-$
$\mathbf{tt}_{(\mathsf{y};\,2)\|(\mathsf{z};\,1)}^+$
$\odot$
$\mathbf{mem}_W\ \ \mathbf{mem}_R$
$\mathbf{wtt}_{\mathsf{x}}^-$
$\mathbf{r}_{\mathsf{y}}^-$
$\mathbf{ok}_{\mathsf{x}}^+$
$\mathbf{tt}_{\mathsf{x}\|\mathsf{y}}^+$
$=$
$x_W : \mathbf{mem}_W,\ x_R : \mathbf{mem}_R \vdash \mathbf{bool}$
$\mathbf{q}_{\mathsf{x}}^-$
$\mathbf{wtt}_{\mathsf{x};\,1}$
$\mathbf{r}_{\mathsf{x};\,2}$
$\mathbf{ok}_{\mathsf{x};\,1}$
$\mathbf{tt}_{(\mathsf{x};\,1)\|(\mathsf{x};\,2)}$
$\mathbf{tt}_{\mathsf{x};\,3}^+$

Fig. 12: Example of interaction and composition between $\mathbb{R}_+$-augmentations

If $\sigma : A^\perp \parallel B$ and $\tau : B^\perp \parallel C$, we write $\tau \circledast \sigma$ for the set comprising all $\mathbb{p} \circledast \mathbb{q}$ such that $\mathbb{p} \in \tau$ and $\mathbb{q} \in \sigma$ are causally compatible. For $\mathbb{q} \in \sigma$ and $\mathbb{p} \in \tau$ causally compatible with $|\mathbb{p} \circledast \mathbb{q}| = x_A \parallel x_B \parallel x_C$, their **composition** is $\mathbb{p} \odot \mathbb{q} = (x_A \parallel x_C, \leq_{\mathbb{p}\odot\mathbb{q}}, \lambda_{\mathbb{p}\odot\mathbb{q}})$ where $\leq_{\mathbb{p}\odot\mathbb{q}}$ and $\lambda_{\mathbb{p}\odot\mathbb{q}}$ are the restrictions of $\leq_{\mathbb{p}\circledast\mathbb{q}}$ and $\lambda_{\mathbb{p}\circledast\mathbb{q}}$. Finally, the **composition** of $\sigma : A^\perp \parallel B$ and $\tau : B^\perp \parallel C$ is the set comprising all $\mathbb{p} \odot \mathbb{q}$ for $\mathbb{q} \in \sigma$ and $\mathbb{p} \in \tau$ causally compatible.

In Figure 12, we display an example composition between $\mathbb{R}_+$-augmentations – with also in gray the underlying interaction. The reader may check that the variant of the left $\mathbb{R}_+$-augmentation with $\mathbf{tt}$ replaced with $\mathbf{ff}$ is causally compatible with the other augmentation in Figure 11, with composition $\mathbf{q}_{\mathsf{x}}^- \rightarrow \mathbf{ff}_{\mathsf{x};\,4}^+$.

We also have a tensor operation: on arenas, $A \otimes B$ is simply a synonym for $A \parallel B$. If $\mathbb{q}_1 \in \mathcal{R}\text{-Aug}(A_1^\perp \parallel B_1)$ and $\mathbb{q}_2 \in \mathcal{R}\text{-Aug}(A_2^\perp \parallel B_2)$, their **tensor product** $\mathbb{q}_1 \otimes \mathbb{q}_2 \in \mathcal{R}\text{-Aug}((A_1 \otimes A_2)^\perp \parallel (B_1 \otimes B_2))$ is defined in the obvious way. This is lifted to $\mathcal{R}$-strategies element-wise. As is common when constructing basic categories of games and strategies, we have:

Proposition 1. *There is a compact closed category $\mathcal{R}$-Strat having arenas as objects, and as morphisms, $\mathcal{R}$-strategies between them.*

Negative Arenas and $\mathcal{R}$-Strategies. As a compact closed category, $\mathcal{R}$-Strat is a model of the linear λ-calculus. However, we will (as usual for call-by-name) instead interpret $\mathcal{R}$-IPA in a sub-category of *negative* arenas and strategies, in which the empty arena 1 is terminal, providing the interpretation of weakening. We will stay very brief here, as this proceeds exactly as in [5].

A partial order with polarities is **negative** if all its minimal events are. This applies in particular to arenas, and $\mathcal{R}$-augmentations. A $\mathcal{R}$-strategy is **negative** if all its $\mathcal{R}$-augmentations are. A negative $\mathcal{R}$-augmentation $\mathbb{q} \in \mathcal{R}\text{-Aug}(A)$ is **well-threaded** if for all $a \in |\mathbb{q}|$, $[a]_{\mathbb{q}}$ has exactly one minimal event; a $\mathcal{R}$-strategy is **well-threaded** iff all its $\mathcal{R}$-augmentations are. We have:

Proposition 2. Negative arenas *and* negative well-threaded $\mathcal{R}$-strategies *form a cartesian symmetric monoidal closed category $\mathcal{R}$-Strat_-, with* 1 *terminal.*

We also write $\sigma : A \rightarrow\!\!\!\!+\!\!\!\!\!\rightarrow B$ for morphisms in $\mathcal{R}$-Strat_-.

The closure of $\mathcal{R}$-Strat does not transport to $\mathcal{R}$-Strat_- as $A^\perp \parallel B$ is never negative if A is non-empty, thus we replace it with a negative version. Here we

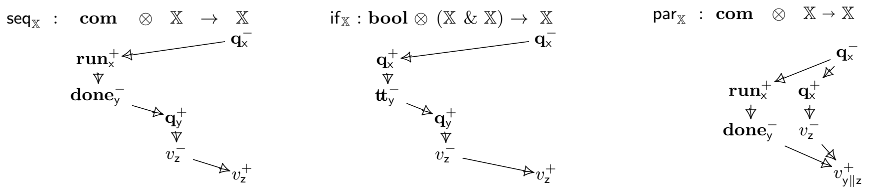
seq$_{\mathbb{X}}$: com ⊗ $\mathbb{X}$ → $\mathbb{X}$
$\mathbf{q}_{\mathsf{x}}^{-}$
$\mathbf{run}_{\mathsf{x}}^{+}$
$\mathbf{done}_{\mathsf{y}}^{-}$
$\mathbf{q}_{\mathsf{y}}^{+}$
v_{z}^{-}
v_{z}^{+}
if$_{\mathbb{X}}$: bool ⊗ ($\mathbb{X}$ & $\mathbb{X}$) → $\mathbb{X}$
$\mathbf{q}_{\mathsf{x}}^{-}$
$\mathbf{q}_{\mathsf{x}}^{+}$
$\mathbf{tt}_{\mathsf{y}}^{-}$
$\mathbf{q}_{\mathsf{y}}^{+}$
v_{z}^{-}
v_{z}^{+}
par$_{\mathbb{X}}$: com ⊗ $\mathbb{X}$ → $\mathbb{X}$
$\mathbf{q}_{\mathsf{x}}^{-}$
$\mathbf{run}_{\mathsf{x}}^{+}$
$\mathbf{q}_{\mathsf{x}}^{+}$
$\mathbf{done}_{\mathsf{y}}^{-}$
v_{z}^{-}
$v_{\mathsf{y}\|\mathsf{z}}^{+}$

Fig. 13: Maximal $\mathcal{R}$-augmentations of $\mathcal{R}$-strategies used in the interpretation

describe only a restricted case of the general construction in [5], which is however sufficient for the types of $\mathcal{R}$-IPA. If A, B are negative arenas and B is **well-opened**, *i.e.* it has exactly one minimal event b, we form $A \multimap B$ as having all components as in $A^\perp \parallel B$, with additional dependencies $\{((2, b), (1, a)) \mid a \in A\}$.

Using the compact closed structure of $\mathcal{R}$-Strat it is easy to build a copycat $\mathcal{R}$-strategy $\mathsf{ev}_{A,B} : (A \multimap B) \otimes A \stackrel{+}{\twoheadrightarrow} B$, and to associate to any $\sigma : C \otimes A \stackrel{+}{\twoheadrightarrow} B$ some $\Lambda(\sigma) : C \stackrel{+}{\twoheadrightarrow} A \multimap B$ providing the monoidal closure. The cartesian product of A and B is $A \,\&\, B$ with components the same as $A \parallel B$, except for $(1, a) \# (2, b)$ for all $a \in A, b \in B$. We write $\pi_i : A_1 \,\&\, A_2 \stackrel{+}{\twoheadrightarrow} A_i$ for the projections, and $\langle \sigma, \tau \rangle : A \stackrel{+}{\twoheadrightarrow} B \,\&\, C$ for the pairing of $\sigma : A \stackrel{+}{\twoheadrightarrow} B$, and $\tau : A \stackrel{+}{\twoheadrightarrow} C$.

Interpretation of $\mathcal{R}$-IPA. We set $[\![\mathbf{com}]\!] = \mathbf{run}^- \rightarrowtriangle \mathbf{done}^+$, $[\![\mathbf{bool}]\!]$ as in the right-hand side of Figure 9, $[\![\mathbf{mem}_W]\!]$ and $[\![\mathbf{mem}_R]\!]$ as in Figure 10, and $[\![A \multimap B]\!] = [\![A]\!] \multimap [\![B]\!]$ as expected. Contexts $\Gamma = x_1 : A_1, \ldots, x_n : A_n$ are interpreted as $[\![\Gamma]\!] = \otimes_{1 \le i \le n} [\![A_i]\!]$. Terms $\Gamma \vdash M : A$ are interpreted as $[\![t]\!] : [\![\Gamma]\!] \stackrel{+}{\twoheadrightarrow} [\![A]\!]$ as follows: $[\![\bot]\!]$ is the diverging $\mathcal{R}$-strategy (no player move), $[\![\mathbf{consume}(\alpha)]\!]$ has only maximal $\mathcal{R}$-augmentation $\mathbf{run}_{\mathsf{x}}^- \rightarrowtriangle \mathbf{done}_{\mathsf{x};\alpha}^+$, $[\![\mathbf{skip}]\!]$ is $[\![\mathbf{consume}(0)]\!]$, and $\mathbf{tt}$ and $\mathbf{ff}$ are interpreted similarly with the adequate constant $\mathcal{R}$-strategies. The rest of the interpretation is given on the left, using the two obvious isos $\mathsf{deref} : [\![\mathbf{mem}_R]\!] \stackrel{+}{\twoheadrightarrow} [\![\mathbf{bool}]\!]$ and $\mathsf{assign} : [\![\mathbf{mem}_W]\!] \stackrel{+}{\twoheadrightarrow} [\![\mathbf{com}]\!]$; the $\mathcal{R}$-strategy cell introduced in Figure 11; and additional $\mathcal{R}$-strategies with typical $\mathcal{R}$-augmentations in Figure 13. We omit the (standard) clauses for the λ-calculus.

$$
\begin{aligned}
[\![M; N : \mathbb{X}]\!] &= \mathsf{seq}_{\mathbb{X}} \odot ([\![M]\!] \otimes [\![N]\!]) \\
[\![M \parallel N : \mathbb{X}]\!] &= \mathsf{par}_{\mathbb{X}} \odot ([\![M]\!] \otimes [\![N]\!]) \\
[\![\mathbf{if}\, M\, N_1\, N_2 : \mathbb{X}]\!] &= \mathsf{if}_{\mathbb{X}} \odot ([\![M]\!] \otimes \langle [\![N_1]\!], [\![N_2]\!] \rangle) \\
[\![!M : \mathbf{bool}]\!] &= \mathsf{deref} \odot [\![M]\!] \\
[\![M := \mathbf{tt} : \mathbf{com}]\!] &= \mathsf{assign} \odot [\![M]\!] \\
[\![\mathbf{new}\, x, y\, \mathbf{in}\, M : \mathbb{X}]\!] &= [\![M]\!] \odot ([\![\Gamma]\!] \otimes \mathsf{cell})
\end{aligned}
$$

3.3 Soundness

Now that we have defined the game semantics of $\mathcal{R}$-IPA, we set to prove that it is sound with respect to the operational semantics given in Section 2.2.

We first introduce a useful notation. For any type A, $[\![A]\!]$ has a unique minimal event; write $(\!|A|\!)$ for the arena without this minimal event. Likewise, if $\Gamma \vdash M : A$, then by construction, $[\![M]\!] : [\![\Gamma]\!]^\perp \parallel [\![A]\!]$ is a negative $\mathcal{R}$-strategy whose augmentations all share the same minimal event $\mathbf{q}_{\mathsf{x}}^-$ where $\mathbf{q}^-$ is minimal in A. For $\alpha \in \mathcal{R}$, write $(\!|M|\!)_\alpha$ for $[\![M]\!]$ without $\mathbf{q}_\mathsf{x}^-$, with x replaced by α. Then we have $(\!|M|\!)_\alpha : [\![\Gamma]\!]^\perp \parallel (\!|A|\!)$ – one may think of $(\!|M|\!)_\alpha$ as "M started with consumed resource α".

Naively, one may expect soundness to state that for all $\vdash M : \mathbf{com}$, if $M \Downarrow_\alpha$, then $(\!|M|\!)_0 = \mathbf{done}_\alpha^+$. However, whereas the resource annotations in the semantics are always as good as permitted by the causal constraints, derivations in the operational semantics may be sub-optimal. For instance, we may derive $M \Downarrow_\alpha$ not using the parallel rule at all. So our statement is:

Theorem 1. *If $\vdash M : \mathbf{com}$ with $M \Downarrow_\alpha$, there is $\beta \leq_\mathcal{R} \alpha$* s.t. $(\!|M|\!)_0 = \mathbf{done}_\beta^+$.

Our proof methodology is standard: we replay operational derivations as augmentations in the denotational semantics. Stating the invariant successfully proved by induction on operational derivations requires some technology.

If s is a store, then write $\mathsf{cell}_s : [\![\Omega(s)]\!]$ for the memory strategy for store s. It is defined as $\otimes_{\ell \in \mathrm{dom}(s)} \mathsf{cell}_{s(\ell)}$ where $\mathsf{cell}_\varepsilon = \mathsf{cell}$, cell_{R^α} is the $\mathcal{R}$-strategy with only maximal $\mathcal{R}$-augmentation $\mathbf{wtt}_\mathsf{x}^- \twoheadrightarrow \mathbf{ok}_{\mathsf{x}\parallel\alpha}^+$, cell_{W^α} has maximal $\mathcal{R}$-augmentation $\mathbf{r}_\mathsf{y}^- \twoheadrightarrow \mathbf{tt}_{\alpha\parallel\mathsf{y}}^+$, and the empty $\mathcal{R}$-strategy for the other cases. If $s \leq_\mathsf{M} s'$, then s' can be obtained from s using memory operations and there is a matching $\mathcal{R}$-augmentation $\mathbb{q}_{s \rhd s'} \in \mathsf{cell}_s$ defined location-wise in the obvious way.

Now, if $\sigma : [\![\Omega(s)]\!]^\perp \parallel (\!|A|\!)$ is a $\mathcal{R}$-strategy and $\mathbb{q} \in \sigma$ with moves only in $[\![\Omega(s)]\!]^\perp$ is causally compatible with $\mathbb{q}_{s \rhd s'}$, we define the **residual** of σ after $\mathbb{q}$:

$$\sigma/(\mathbb{q} \circledast \mathbb{q}_{s \rhd s'}) : [\![\Omega(s')]\!]^\perp \parallel (\!|A|\!)$$

If $\mathbb{p} \in \sigma$ with $\mathbb{q} \hookrightarrow \mathbb{p}$, we write first $\mathbb{p}' = \mathbb{p}/(\mathbb{q} \circledast \mathbb{q}_{s \rhd s'})$ the $\mathcal{R}$-augmentation with $|\mathbb{p}'| = |\mathbb{p}| \setminus |\mathbb{q}|$, and with causal order the restriction of that of $\mathbb{p}$. For $e \in |\mathbb{p}'|$, we set $\lambda_{\mathbb{p}'}(e)$ to be $\lambda_\mathbb{p}(e)$ whose arguments corresponding to negative events e' in $\mathbb{q}$ are instantiated with $\lambda_{\mathbb{q} \circledast \mathbb{q}_{s \rhd s'}}(e') \in \mathcal{R}$. With that, we set $\sigma/(\mathbb{q} \circledast \mathbb{q}_{s \rhd s'})$ as comprising all $\mathbb{p}/(\mathbb{q} \circledast \mathbb{q}_{s \rhd s'})$ for $\mathbb{p} \in \sigma$ with $\mathbb{q} \hookrightarrow \mathbb{p}$.

Informally, this means that, considering some $\mathbb{q}$ which represents a scheduling of the memory operations turning s into s', we extract from σ its behavior after the execution of these memory operations. Finally, we generalize $\leq_\mathcal{R}$ to $\mathcal{R}$-augmentations by setting $\mathbb{q} \leq_\mathcal{R} \mathbb{q}'$ iff they have the same underlying partial order and for all $e \in |\mathbb{q}|$, $\lambda_\mathbb{q}(e) \leq_\mathcal{R} \lambda_{\mathbb{q}'}(e)$. With that, we can finally state:

Lemma 1. *Let $\Omega(s) \vdash M : A$, $\langle M, s_1, \alpha \rangle \rightrightarrows \langle M', s_1' \uplus s_2', \alpha' \rangle$ with* $\mathrm{dom}(s_1) = \mathrm{dom}(s_1')$, *and all resource annotations in s_1 lower than α. Then, there is $\mathbb{q} \in (\!|M|\!)_\alpha$ with events in $[\![\Omega(s)]\!]$, causally compatible with $\mathbb{q}_{s_1 \rhd s_1'}$, and a function*

$$\varphi : (\!|M'|\!)_{\alpha'} \circledast \mathsf{cell}_{s_2'} \quad \longrightarrow \quad (\!|M|\!)_\alpha/(\mathbb{q} \circledast \mathbb{q}_{s_1 \rhd s_1'})$$

preserving $\hookrightarrow$ and s.t. *for all $\mathbb{p} \circledast \mathbb{q}_{s_2'} \in (\!|M'|\!)_{\alpha'} \circledast \mathsf{cell}_{s_2'}$, $\varphi(\mathbb{p} \circledast \mathbb{q}_{s_2'}) \leq_\mathcal{R} \mathbb{p} \odot \mathbb{q}_{s_2'}$.*

This is proved by induction on the operational semantics – the critical cases are: assignment and dereferenciation exploiting that if $\alpha \leq_\mathcal{R} \beta$, then $\alpha \parallel \beta = \beta$ (which boils down to idempotence); and parallel composition where compatibility of s' and s'' entails that the corresponding augmentations of cell_s are compatible.

Lemma 1, instantiated with $\langle M, \emptyset, 0 \rangle \rightrightarrows \langle \mathbf{skip}, s, \alpha \rangle$, yields soundness.

Non-Adequacy. Our model is not adequate. To see why, consider:

$$\vdash \mathbf{new}\, x_W, x_R\, \mathbf{in} \left(\begin{array}{r} \mathbf{wait}(1); \\ x_W := \mathbf{tt}; \\ \mathbf{wait}(2) \end{array} \middle\| \begin{array}{l} \mathbf{wait}(2); \\ !x_R; \\ \mathbf{wait}(1) \end{array} \right) : \mathbf{bool}$$

Our model predicts that this may evaluate to $\mathbf{tt}$ in 3 seconds (see Figure 12) and to $\mathbf{ff}$ in 4 seconds. However, the operational semantics can only evaluate it (both to $\mathbf{tt}$ and $\mathbf{ff}$) in 4 seconds. Intuitively, the reason is that the causal shapes implicit in the reduction $\rightrightarrows$ are all series-parallel (generated with sequential and parallel composition), whereas the interaction in Figure 12 is not.

Our causal semantic approach yields a finer resource analysis than achieved by the parallel operational semantics. The operational semantics, rather than our model, is to blame for non-adequacy: indeed, we now show that for $\mathcal{R} = \mathbb{R}_+$ our model is adequate *w.r.t.* an operational semantics specialized for time.

4 Adequacy for Time

For time, we may refine the operational semantics by adding the following rule

$$\langle \mathbf{wait}(t_1 + t_2), s, t_0 \rangle \to \langle \mathbf{wait}(t_2), s, t_0 + t_1 \rangle$$

using which the program above evaluates to $\mathbf{tt}$ in 3 seconds. It is clear that the soundness theorem of the previous section is retained.

We first focus on adequacy for first-order programs without abstraction or application, written $\Omega(s) \vdash_1 M : \mathbf{com}$. For any $t_0 \in \mathbb{R}_+$ there is $\langle M, s, t_0 \rangle \rightrightarrows \langle M', s \uplus s', t_0 \rangle$ where $(\!| M |\!)_{t_0} = (\!| M' |\!)_{t_0} \odot \mathsf{cell}_{s'}$ and M' is in **canonical form**: it cannot be decomposed as $C[\mathbf{skip}; N]$, $C[\mathbf{skip} \parallel N]$, $C[N \parallel \mathbf{skip}]$, $C[\mathbf{if}\, \mathbf{tt}\, N_1\, N_2]$, $C[\mathbf{if}\, \mathbf{ff}\, N_1\, N_2]$, $C[\mathbf{wait}(0)]$ and $C[\mathbf{new}\, x, y\, \mathbf{in}\, N]$ for $C[]$ an evaluation context.

Consider $\Omega(s) \vdash_1 M : \mathbf{com}$, and $\mathbb{q} \in (\!| M |\!)_{t_0} \circledast \mathsf{cell}_s$ with a top element $\mathbf{done}^+_{t_\mathsf{f}}$ in $(\!| \mathbf{com} |\!)$, the **result** – *i.e.* $\mathbb{q}$ describes an interaction between $(\!| M |\!)_{t_0}$ and the memory leading to a successful evaluation to **done** at time t_f. To prove adequacy, we must extract from it a derivation from $\langle M, s, t_0 \rangle$, at time t_f.

Apart from the top $\mathbf{done}^+_{t_\mathsf{f}}$, $\mathbb{q}$ only records memory operations, which we must replicate operationally in the adequate order. A **minimal operation with timing** t is either the top $\mathbf{done}^+_t$ if it is the only event in $\mathbb{q}$, or a prefix $(m_t \rightarrowtriangle n_t) \hookrightarrow \mathbb{q}$ corresponding to a memory operation (for instance, in augmentations of Figure 14, the only minimal operation has timing 2). If $t = t_0$, this operation should be performed immediately. If $t > t_0$ we need to spend time to trigger it – it is then critical to spend time on *all available* **wait***s in parallel*:

Lemma 2. *For $\Omega(s) \vdash_1 M : \mathbf{com}$ in canonical form, $t_0 \in \mathbb{R}_+$, $\mathbb{q} \in (\!| M |\!)_{t_0} \circledast \mathsf{cell}_s$ with result $\mathbf{done}^+_{t_\mathsf{f}}$, if all minimal operations have timing strictly greater than t_0,*

$$\langle M, s, t_0 \rangle \rightrightarrows \langle M', s, t_0 + t \rangle$$

for some $t > 0$ and M' only differing from M by having smaller annotations in **wait** *commands and at least one* **wait** *changed to* **skip**.

Furthermore, there is $\mathbb{q} \leq_{\mathcal{R}} \mathbb{q}'$ with $\mathbb{q}' \in (\!| M' |\!)_{t_0+t} \circledast \mathsf{cell}_s$ with result $\mathbf{done}^+_{t_\mathsf{f}}$.

$$\left\langle \begin{array}{l}\mathbf{wait}(2);\\ \ell_W := \mathbf{tt}\end{array} \middle\| \begin{array}{l}\mathbf{wait}(1);\\ \mathbf{test}\,(!\ell_R)\end{array}, \ell \mapsto \epsilon, 0\right\rangle \rightrightarrows \left\langle \begin{array}{l}\mathbf{wait}(1);\\ \ell_W := \mathbf{tt}\end{array} \middle\| \begin{array}{l}\mathbf{wait}(0);\\ \mathbf{test}\,(!\ell_R)\end{array}, \ell \mapsto \epsilon, 1\right\rangle \rightrightarrows \left\langle \begin{array}{l}\mathbf{wait}(0);\\ \ell_W := \mathbf{tt}\end{array} \middle\| \begin{array}{l}\\ \mathbf{test}\,(!\ell_R)\end{array}, \ell \mapsto \epsilon, 2\right\rangle$$

$$\begin{tikzcd}
\mathbf{wtt}_2 \arrow[d] \arrow[rr] & & \mathbf{r}_1 \arrow[d] & & \mathbf{wtt}_2 \arrow[d] \arrow[rr] & & \mathbf{r}_1 \arrow[d] & & \mathbf{wtt}_2 \arrow[d] \arrow[rr] & & \mathbf{r}_2 \arrow[d] \\
\mathbf{ok}_2 \arrow[dr] & & \mathbf{tt}_2 \arrow[dl] & \leq_{\mathcal{R}} & \mathbf{ok}_2 \arrow[dr] & & \mathbf{tt}_2 \arrow[dl] & \leq_{\mathcal{R}} & \mathbf{ok}_2 \arrow[dr] & & \mathbf{tt}_2 \arrow[dl] \\
& \mathbf{done}_2^+ & & & & \mathbf{done}_2^+ & & & & \mathbf{done}_2^+ &
\end{tikzcd}$$

Fig. 14: Spending time adequately (where $\mathbf{test}\, M = \mathbf{if}\, M\, \mathbf{skip}\, \bot$)

Proof. As M is in canonical form, all delays in minimal operations are impacted by $\mathbf{wait}(t)$ commands in head position (*i.e.* such that $M = C[\mathbf{wait}(t)]$). Let $t_{\min}$ be the minimal time appearing in those $\mathbf{wait}(-)$ commands in head position. Using our new rule and parallel composition, we remove $t_{\min}$ to all such instances of $\mathbf{wait}(-)$; then transform the resulting occurrences of $\mathbf{wait}(0)$ to $\mathbf{skip}$.

A representative example is displayed in Figure 14. In the second step, though $!\ell_R$ is available immediately, we must wait to get the right result.

With that we can prove the key lemma towards adequacy.

Lemma 3. *Let $\Omega(s) \vdash_1 M : \mathbf{com}$, $t_0 \in \mathbb{R}_+$, and $\mathbb{q} \in (\!|M|\!)_{t_0} \circledast \mathsf{cell}_s$ with result $\mathbf{done}^+_{t_\mathsf{f}}$ in $(\!|\mathbf{com}|\!)$. Then, there is $\langle M, s, t_0\rangle \rightrightarrows \langle \mathbf{skip}, -, t_\mathsf{f}\rangle$.*

Proof. By induction on the size of M. First, we convert M to canonical form. If all minimal operations in $\mathbb{q} \in (\!|M|\!)_{t_0}$ have timing strictly greater than t_0, we apply Lemma 2 and conclude by induction hypothesis.

Otherwise, at least one minimal operation has timing t_0. If it is the result $\mathbf{done}^+_{t_0}$ in $(\!|\mathbb{X}|\!)$, then M is the constant $\mathbf{skip}$. Otherwise, it is a memory operation, say $\mathbb{p} \hookrightarrow \mathbb{q}$ with $\mathbb{p} = (\mathbf{r}_{t_0} \twoheadrightarrow b_{t_0})$ and write also $s' = s[\ell \mapsto s(\ell).R^{t_0}]$. It follows then by an induction on M that $M = C[!\ell_R]$ for some $C[]$, with

$$\mathbb{q}/(\mathbb{p} \circledast \mathbb{q}_{s \triangleright s'}) \in (\!|C[b]|\!)_{t_0} \circledast \mathsf{cell}_s$$

so $\langle M, s, t_0\rangle \rightrightarrows \langle C[b], s', t_0\rangle \rightrightarrows \langle \mathbf{skip}, -, t_\mathsf{f}\rangle$ by induction hypothesis.

Adequacy follows for higher-order programs: in general, any $\vdash M : \mathbf{com}$ can be β-reduced to first-order M', leaving the interpretation unchanged. By Church-Rosser, M' behaves like M operationally, up to weak bisimulation. Hence:

Theorem 2. *Let $\vdash M : \mathbf{com}$. For any $t \in \mathbb{R}_+$, if $\mathbf{done}^+_t \in (\!|M|\!)_0$ then $M \Downarrow_t$.*

5 Conclusion

It would be interesting to compare our model with structures used in timing analysis, for instance [23] relies on a concurrent generalization of control flow graphs that is reminiscent of event structures. In future work we also plan to investigate whether our annotated model construction could be used for other purposes, such as symbolic execution or abstract interpretation.

References

1. Abramsky, S., Jagadeesan, R., Malacaria, P.: Full abstraction for PCF. Inf. Comput. **163**(2), 409–470 (2000). https://doi.org/10.1006/inco.2000.2930
2. Abramsky, S., Melliès, P.: Concurrent games and full completeness. In: 14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999. pp. 431–442 (1999). https://doi.org/10.1109/LICS.1999.782638
3. Alcolei, A., Clairambault, P., Hyland, M., Winskel, G.: The true concurrency of Herbrand's theorem. In: 27th EACSL Annual Conference on Computer Science Logic, CSL 2018, September 4-7, 2018, Birmingham, UK. pp. 5:1–5:22 (2018). https://doi.org/10.4230/LIPIcs.CSL.2018.5
4. Brunel, A., Gaboardi, M., Mazza, D., Zdancewic, S.: A core quantitative coeffect calculus. In: Programming Languages and Systems - 23rd European Symposium on Programming, ESOP 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014, Proceedings. pp. 351–370 (2014). https://doi.org/10.1007/978-3-642-54833-8_19
5. Castellan, S., Clairambault, P.: Causality vs. interleavings in concurrent game semantics. In: Desharnais, J., Jagadeesan, R. (eds.) 27th International Conference on Concurrency Theory, CONCUR 2016, August 23-26, 2016, Québec City, Canada. LIPIcs, vol. 59, pp. 32:1–32:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2016). https://doi.org/10.4230/LIPIcs.CONCUR.2016.32
6. Castellan, S., Clairambault, P., Paquet, H., Winskel, G.: The concurrent game semantics of probabilistic PCF. In: Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018. pp. 215–224 (2018). https://doi.org/10.1145/3209108.3209187
7. Castellan, S., Clairambault, P., Rideau, S., Winskel, G.: Games and strategies as event structures. Logical Methods in Computer Science **13**(3) (2017). https://doi.org/10.23638/LMCS-13(3:35)2017
8. Castellan, S., Clairambault, P., Winskel, G.: The parallel intensionally fully abstract games model of PCF. In: 30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015. pp. 232–243 (2015). https://doi.org/10.1109/LICS.2015.31
9. Castellan, S., Clairambault, P., Winskel, G.: Thin games with symmetry and concurrent hyland-ong games. To appear in Logical Methods in Computer Science. (2019)
10. Clairambault, P., de Visme, M., Winskel, G.: Game semantics for quantum programming. PACMPL **3**(POPL), 32:1–32:29 (2019). https://doi.org/10.1145/3290345
11. Ehrhard, T.: The Scott model of linear logic is the extensional collapse of its relational model. Theor. Comput. Sci. **424**, 20–45 (2012). https://doi.org/10.1016/j.tcs.2011.11.027
12. Faggian, C., Piccolo, M.: Partial orders, event structures and linear strategies. In: Typed Lambda Calculi and Applications, 9th International Conference, TLCA 2009, Brasilia, Brazil, July 1-3, 2009. Proceedings. pp. 95–111 (2009). https://doi.org/10.1007/978-3-642-02273-9_9
13. Ghica, D.R.: Slot games: a quantitative model of computation. In: Proceedings of the 32nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2005, Long Beach, California, USA, January 12-14, 2005. pp. 85–97 (2005). https://doi.org/10.1145/1040305.1040313

14. Ghica, D.R., Murawski, A.S.: Angelic semantics of fine-grained concurrency. Ann. Pure Appl. Logic **151**(2-3), 89–114 (2008). https://doi.org/10.1016/j.apal.2007.10.005
15. Ghica, D.R., Smith, A.I.: Bounded linear types in a resource semiring. In: Programming Languages and Systems - 23rd European Symposium on Programming, ESOP 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014, Proceedings. pp. 331–350 (2014). https://doi.org/10.1007/978-3-642-54833-8_18
16. Hoare, T., Möller, B., Struth, G., Wehrman, I.: Concurrent Kleene algebra and its foundations. J. Log. Algebr. Program. **80**(6), 266–296 (2011). https://doi.org/10.1016/j.jlap.2011.04.005
17. Hyland, J.M.E., Ong, C.L.: On full abstraction for PCF: I, II, and III. Inf. Comput. **163**(2), 285–408 (2000). https://doi.org/10.1006/inco.2000.2917
18. Laird, J., Manzonetto, G., McCusker, G., Pagani, M.: Weighted relational models of typed lambda-calculi. In: 28th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), New Orleans, USA, Proceedings. pp. 301–310 (2013)
19. Laurent, O.: Game semantics for first-order logic. Logical Methods in Computer Science **6**(4) (2010). https://doi.org/10.2168/LMCS-6(4:3)2010
20. Melliès, P.: Asynchronous games 4: A fully complete model of propositional linear logic. In: 20th IEEE Symposium on Logic in Computer Science (LICS 2005), 26-29 June 2005, Chicago, IL, USA, Proceedings. pp. 386–395 (2005). https://doi.org/10.1109/LICS.2005.6
21. Melliès, P.: Game semantics in string diagrams. In: Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25-28, 2012. pp. 481–490 (2012). https://doi.org/10.1109/LICS.2012.58
22. Melliès, P., Mimram, S.: Asynchronous games: Innocence without alternation. In: CONCUR 2007 - Concurrency Theory, 18th International Conference, CONCUR 2007, Lisbon, Portugal, September 3-8, 2007, Proceedings. pp. 395–411 (2007). https://doi.org/10.1007/978-3-540-74407-8_27
23. Mittermayr, R., Blieberger, J.: Timing analysis of concurrent programs. In: Vardanega, T. (ed.) 12th International Workshop on Worst-Case Execution Time Analysis, WCET 2012, July 10, 2012, Pisa, Italy. OASICS, vol. 23, pp. 59–68. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2012). https://doi.org/10.4230/OASIcs.WCET.2012.59
24. Plotkin, G.D.: Post-graduate lecture notes in advanced domain theory (incorporating the "Pisa notes"). Dept. of Computer Science, Univ. of Edinburgh (1981)
25. Rideau, S., Winskel, G.: Concurrent strategies. In: Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science, LICS 2011, June 21-24, 2011, Toronto, Ontario, Canada. pp. 409–418 (2011). https://doi.org/10.1109/LICS.2011.13
26. Sands, D.: Operational theories of improvement in functional languages (extended abstract). In: Functional Programming, Glasgow 1991, Proceedings of the 1991 Glasgow Workshop on Functional Programming, Portree, Isle of Skye, UK, 12-14 August 1991. pp. 298–311 (1991)
27. Winskel, G.: Event structures. In: Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986, Part II, Proceedings of an Advanced Course, Bad Honnef, Germany, 8-19 September 1986. pp. 325–392 (1986). https://doi.org/10.1007/3-540-17906-2_31