A Game Semantics Study of Virtual Effects

HAMZA JAAFAR and *, Inria / Nantes Université, France

We introduce the first fully abstract game semantics model for a typed language with algebraic effects, effect handlers, and dynamically generated effect instances. Central to this work is the study of effect propagation — a transitory in which an effect is performed but not yet handled, suspended between the program and its environment, which reveals subtle interactive behaviors, prompting a refinement of the standard game semantics interface.

To account for this additional form of interaction, we extend the standard *pure* interaction interface of game semantics consisting of questions and answers with *effectful* moves that involve the propagation of effects and the yielding of *delimited* continuations. Crucially, we introduce a novel trace equivalence that matches contextual equivalence by abstracting away from unobservable forwarding steps.

To support this finer analysis, we develop handling structures that explicitly track how effects are handled and how continuations are invoked. In parallel, a view-based semantics is introduced to model the program's evolving perspective throughout interaction. These contributions allow us to revisit and generalize key semantic notions—such as innocence and visibility—in the setting of virtual effects.

Together, these developments culminate in a compositional and fully abstract model that illuminates the semantics of effectful computation under dynamic effect generation, offering a refined understanding of interaction in the presence of algebraic effects and handlers.

CCS Concepts: • **Theory of computation** → **Denotational semantics**; *Control primitives*; **Operational semantics**.

Additional Key Words and Phrases: Algebraic Effects, Handlers, Game Semantics

HAMZA JAAFAR and [11], Inria / Nantes Université, France

ACM Reference Format:
. 2018. A Game Semantics Study of Virtual Effects. 1, 1 (July 2018), 68 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Algebraic effects were originally introduced by Power and Plotkin [Plotkin and Power 2001, 2002] to provide a directdenotational account for syntactic entities that *perform* effects such as **get**, **set** for global state, **raise** for exceptions, or **chose** for non-deterministic choice, *etc.*

The combination of algebraic operations with effect handlers [Plotkin and Pretnar 2013] is a more recent and powerful abstraction for programming with effects. Effect handlers generalize exception handlers by giving access not only to the raised effect but also to its delimited continuation. This enables a modular and compositional approach to effectful programming, supports user-defined effects, and facilitates effect combination in a seamless way.

Authors' Contact Information: Hamza JAAFAR;, Inria / Nantes Université, Nantes, France.

ACM XXXX-XXXX/2018/7-ART
https://doi.org/XXXXXXX.XXXXXXX

A central challenge with this abstraction lies in managing multiple occurrences of the same effect while preserving modularity. Two main approaches have been proposed: the *lexical approach* and the *generative approach*, as described in [de Vilhena and Pottier 2023]. In the lexical approach [Biernacki et al. 2019b; Brachthäuser et al. 2020; Zhang and Myers 2019], one introduces effect instances locally, and the handler responsible for these instances is determined statically, preventing instance leakage. This static discipline enforces a form of *effect safety*, ensuring that all performed effects are handled. In contrast, the generative approach provides the possibility of generating dynamically fresh instances of an effect [Bauer and Pretnar 2015], that can be referred by their names, and be passed around as values. While this increases flexibility, it introduces challenges such as aliasing between names of effect instances, that needs to be tamed in order to enforce effect safety via a type-and-effect-system [de Vilhena and Pottier 2023].

In this paper, we adopt the generative approach, but allow for *unsafe interaction*: effect instances may remain unhandled, and aliasing is not prevented. Our goal is to model the interaction between a program and its environment in such a language. Trace models for higher-order languages capture program behavior as sequences of observable actions -calls and returns- where the arguments of these actions may themselves be (abstractions of) functions. We follow the methodology of *operational game semantics* (OGS) to define such a trace model.

The central idea of OGS is to represent programs as labelled transition systems (LTS), derived from the operational semantics of the language, in combination with a decomposition of normal forms into observable and interacting parts. This approach is particularly well suited to languages with generative effects, as demonstrated by Laird's seminal work on ML-style higher-order references [Laird 2007]. OGS also naturally accommodates control operators such as `call/cc` or the μ-binder of the $\lambda\mu$-calculus [Jaber and Murawski 2021a]. In the absence of such control features, it enforces a *well-bracketing* discipline on the interaction—matching calls with returns—as is standard in game semantics [Abramsky et al. 2000; Hyland and Ong 2000].

To extend OGS to a language with algebraic effect handlers, we incorporate:

- actions representing the execution of an effect;
- a fine-grained representation of the control flow between the program and its environment, to account for the exchange of delimited continuations triggered by handled effects.

Our long-term objective is to obtain a fully abstract trace model, capturing contextual equivalence of our language. As a first step in this direction, we prove that trace equivalence of our model is *sound* with respect to contextual equivalence. Moreover, we show that a well-bracketing condition can be imposed on the environment's interaction without compromising soundness. However, we exhibit counterexamples to full abstraction: trace equivalence turns out to be too discriminating to characterize contextual equivalence.

To keep the model simple, we forbid the exchange of effect instances between the program and its environment. This avoids the need to dynamically track which effect instances have been disclosed—information that would otherwise be required for reasoning about interactions.

2 Background material on Operational Game Semantics (OGS)

We begin by recalling the definition of the standard call-by-value λ-calculus with basic types: the unit value and Booleans. This calculus forms the foundation for our development of Operational Game Semantics (OGS).

$$\begin{array}{lrcl} \text{Values} & \mathsf{v} & := & \langle\rangle \mid \mathbb{ff} \mid \mathbb{tt} \mid x \in \mathit{Vars} \mid \lambda x.\mathsf{t} \\ \text{Terms} & \mathsf{t} & ::= & \mathsf{v} \mid \mathsf{t}\ \mathsf{t} \mid \textbf{if } \mathsf{t} \textbf{ then } \mathsf{t} \textbf{ else } \mathsf{u} \\ \text{Evaluation Ctx} & \mathsf{E} & ::= & [] \mid \mathsf{E}\ \mathsf{v} \mid \mathsf{t}\ \mathsf{E} \mid \textbf{if } \mathsf{E} \textbf{ then } \mathsf{t} \textbf{ else } \mathsf{u} \end{array}$$

(a) Term Syntax

$$\begin{array}{lcl} \mathsf{E}[(\lambda x.\mathsf{t})\ \mathsf{v}] & \to_v & \mathsf{E}[\mathsf{t}\{\mathsf{v}/x\}] \\ \mathsf{E}[\textbf{if } \mathbb{tt} \textbf{ then } \mathsf{t}_1 \textbf{ else } \mathsf{t}_2] & \to_v & \mathsf{E}[\mathsf{t}_1] \\ \mathsf{E}[\textbf{if } \mathbb{ff} \textbf{ then } \mathsf{t}_1 \textbf{ else } \mathsf{t}_2] & \to_v & \mathsf{E}[\mathsf{t}_2] \end{array}$$

(b) Operational Semantics

Fig. 1. CBV λ-calculus syntax and semantics

Normal Forms. The considered evaluation strategy is the left variant of call-by-value: only values are substituted during function application. This induces a natural definition of irreducible terms.

$$\text{Normal forms} \quad \mathsf{n} := \mathsf{v} \mid \mathsf{E}[x\ \mathsf{v}]$$

Operational Game Semantics (OGS)

OGS models terms by observing their interactions with all compatible environments (program contexts). These interactions are expressed as traces in a labelled transition system (LTS) guided by the operational semantics. The model is inspired by traditional game semantics [Abramsky and McCusker 1997; Laird 2007], where programs are strategies in an interaction game with the environment.

Intuition. A program (Propponent) is executed against an abstract environment (Opponent) until the evaluation reaches a normal form. A returned value represent an *answer* over a communication channel, whereas an open-stuck expression $\mathsf{E}[x\ \mathsf{v}]$ to a *question*, *i.e.* an interactive call of x with v as input that expects an answer in E. The control is then passed to Opponent, who *continues* the evaluation and initiates the next interactive move. These moves form a trace.

The LTS is bipartite: states are either *active* (under the control of the Proponent) or *passive* (waiting on Opponent). Transitions are labelled by *actions* initiated alternately by each player.

Label Grammar.

$$\begin{array}{lrcll} \text{Abstract values} & a & := & x \mid \langle\rangle \mid \mathbb{ff} \mid \mathbb{tt} & \\ \text{Labels} & \ell & := & a & \text{(return value)} \\ & & \mid & \alpha[x\ a] & \text{(function call)} \end{array} \tag{1}$$

These abstract values and call labels represent the observable interface between term and context. When a function is evaluated to a λ-abstraction, it is abstracted into a fresh variable.

Passive state:

$$I := \langle \pi, \gamma \rangle$$

- π: a continuation stack, keeping track of evaluation context variables
- γ: a mapping from abstract names to concrete functions and contexts

Active state:

$$\langle \mathsf{t}; I \rangle$$

The abstract machine starts in an active state and generates a trace by alternating control with the passive state.

$$\textsc{Eval}\ \frac{\mathsf{t} \longrightarrow_v \mathsf{u}}{\langle \mathsf{t}; \gamma \rangle \longrightarrow_{\mathsf{ci}} \langle \mathsf{u}; \gamma \rangle}$$

(a) Internal Evaluation

$$\textsc{Px}\ \frac{\mathbf{abstract}(\mathsf{v}) = (a, \gamma_a)}{\langle \mathsf{v}; \gamma \rangle \xrightarrow{\mathbf{ret}\, a}_{\mathsf{ci}} \langle \gamma \cdot \gamma_a \rangle}$$

$$\langle \gamma \rangle \xrightarrow{\alpha[\mathbf{ret}\, a]}_{\mathsf{ci}} \langle \gamma(\alpha)[a]; \gamma \rangle\ \textsc{Ox}$$

(b) Pure Moves and Responses

Fig. 2. Transitions of the abstract interaction LTS

Interaction Example

Let $\mathsf{t} = \lambda x.\mathtt{ff}$ and let the context $\mathsf{E} = \lambda y.\mathsf{if}\ y\ ()\ \mathsf{then}\ \mathtt{tt}\ \mathsf{else}\ \mathtt{ff}$.

- Proponent state: $\langle \mathsf{t}; \varnothing \rangle$
- Opponent state: $\langle \alpha, \gamma \rangle$ where $\gamma = \{\alpha \mapsto \mathsf{E}\}$

Together, they form a configuration:

$$\langle I \parallel \mathsf{t}, I' \rangle$$

Transitions interleave, modeling the evaluation of $\mathsf{E}[\mathsf{t}]$ without syntactic substitution. The LTS captures all possible interaction traces.

Abstract Continuations as Channels

Abstract continuations α are treated as communication channels. For example:

- $\alpha[z\ \mathtt{ff}]$: a question on function z with argument $\mathtt{ff}$
- $\alpha[\mathbf{ret}\ \mathtt{tt}]$: a response via continuation α

$$
\begin{array}{lcl}
\mathsf{t} & ::= & \cdots \mid \mathbf{op}\,\mathsf{v} \mid \{\mathsf{t}\}\,\mathbf{with}\,\mathsf{h} \\
\mathsf{h} & ::= & \{\mathbf{ret}\,x \mapsto \mathsf{t};\ \mathbf{op}\,x \mapsto \mathsf{v}\} \\
\mathsf{E} & ::= & \cdots \mid \{\mathsf{E}\}\,\mathbf{with}\,\mathsf{h}
\end{array}
$$

(a) Term Syntax

$$
\begin{array}{lcl}
\mathsf{E}[\{\mathbf{ret}\,\mathsf{v}\}\,\mathbf{with}\,\mathsf{h}] & \to_v & \mathsf{E}[\mathsf{t}\{\mathsf{v}/x\}] \\
\mathsf{E}[\{\mathsf{K}^{\mathbf{op}}[\mathbf{op}\,\mathsf{v}]\}\,\mathbf{with}\,\mathsf{h}] & \to_v & \mathsf{w}\,\mathsf{v}\,(\lambda y.\{\mathsf{K}[y]\}\,\mathbf{with}\,\mathsf{h})
\end{array}
$$

(b) Operational Semantics

Fig. 3. λ_v^{eff} calculus: syntax and operational semantics

Constraints on Opponent Behavior: Visibility, Well-Bracketing, and Innocence

While the OGS framework models a wide range of contextual interactions through its abstract environment and trace-based semantics, it often becomes necessary to restrict the capabilities of the Opponent to ensure meaningful and tractable notions of program equivalence. Three central constraints used in game semantics to this end are: (along three axes/lines)

Well-bracketing. This constraint enforces a stack discipline: every call must be answered before another call is completed. Violations of well-bracketing occur in the presence of control features like continuations or exceptions. In the absence of such features, Proponent and Opponent are expected to alternate moves in a properly nested fashion.

Visibility. Visibility requires that each move (e.g., a question or answer) be justified by a visible part of the history. In other words, a move must relate only to currently open calls. This prevents the Opponent from acting on information that is not locally available in the trace, reflecting a kind of information locality.

Innocence. The innocence constraint states that the strategy's behavior at a given point should depend only on the current view (the visible portion of the interaction history), not the entire history. This is a stricter form of locality and ensures that Proponent strategies are memoryless with respect to unseen branches of the interaction.

These constraints are crucial in many settings, particularly for proving full abstraction results or ensuring that strategies correspond to definable, type-respecting program behaviors.

In our setting, the lack of control features like `call/cc` allows us to inline evaluation contexts and manage them structurally without needing to track explicit continuation stacks. However, when extending the model with effects or control operators, these constraints become active axes of semantic expressivity. Their presence or absence fundamentally changes the kinds of behaviors the Opponent can manifest in a given trace.

Extending to Algebraic Effects and Handlers

We extend the calculus with algebraic effect operators and effect handlers, yielding λ_v^{eff}. A handler h manages a single operation **op**:

New normal forms include stuck effect calls:

$$\mathsf{n} ::= \cdots \mid \mathsf{E}^{\mathbf{op}}[\mathbf{op}\ \mathsf{v}]$$

Extending the OGS. Intuitively, on may want to add the following transition:

$$\langle \mathsf{E}[\mathsf{K}^{\mathbf{op}}[\mathbf{op}\ \mathsf{v}]], \gamma\rangle \xrightarrow{\alpha[\beta[\mathbf{op}\ a]]}_{\mathsf{ci}} \langle \gamma \cdot \gamma_a \cdot \{\alpha \mapsto \mathsf{E}, \beta \mapsto \mathsf{K}\}\rangle \tag{2}$$

When evaluating with abstract environment β, the label $\beta[\alpha^{\mathbf{op}}[\mathbf{op}\ a]]$ denotes both an answer and a question. If Opponent handles **op**, it may bind the delimited continuation.

Conclusion and Roadmap

We have presented the OGS framework by first constructing a CBV LTS and extending it to accommodate abstract environments, algebraic effects, and interaction traces. The rest of this document formalizes the transition system for `dEff` and explores properties such as well-bracketing and visibility in effectful settings.

3 The language Λ_{eff}

3.1 Syntax of Λ_{eff}

We consider a fine-grained call-by-value λ-calculus [Levy 2004] with typed algebraic effects and handlers. Its syntax is given in Figure 4 and its type system in Figure **??**.
The computations of the shape **op** v are responsible for triggering effects. They are given by an operation symbol **op** associated to an effect $\mathbb{E}$ and a particular instance ι of this effect. An effect $\mathbb{E}$ is described by its signature $\{(\mathbf{op}_i : \tau_i \twoheadrightarrow v_i)_i\}$.
For a handler $\mathsf{h} = \uplus_i(\{\mathbf{op}_i\ x_i\ \kappa_i \mapsto \mathsf{u}_i\})_i \uplus \{\mathbf{ret}\ x \mapsto \mathsf{t}\}$, we define $\mathsf{h}^{\mathbf{ret}} := \{\mathbf{ret}\ x \mapsto \mathsf{t}\}$ and $\mathsf{h}^{\mathbf{op}_i} := \{\mathbf{op}_i\ x_i\ \kappa_i \mapsto \mathsf{u}_i\}$, and $\mathbf{hdl}(\mathsf{h}) := \{(\mathbf{op}_i)_i\}$ (*i.e.* the exact set of effects handled by h).

In Fig. 4, we only highlight the additional names that would be absent in a standard presentation. They are given by *function names* $\mathcal{L}$ whose elements we range over by f, g, *delimited* and *undelimited continuation names* range over by $\kappa, \varkappa \in \mathcal{K}$ and $c, d \in C$ respectively, as well as *effect names* $e \in \mathcal{E}$. Further details about their semantic role will be postponed to later sections.

3.2 Operational semantics

The propagation of effects to abstract continuations is to be seen as a form of interaction (in a *game semantics* sense) whereby the *program* inquires whether the *environment*'s code (corresponding to the abstract continuation) handles some specific effect or not. For this reason, we privilege in Fig. 5 an abstract machine presentation of the *operational semantics* (*à la* [Hillerström et al. 2020]) whose reduction rules make this flow of information explicit through the *structural rules* $\mapsto_{\mathsf{struct}} := \mapsto_{\mathsf{fwd}} \cup \mapsto_{\mathsf{psh}}$ and the structure of its configurations.

The *reduction relation* $\mapsto_{\mathsf{op}}$ is defined over pairs `M` formed by a *running computation* `M` and an *effect instance context* `I` crucial to provide a semantics to **new** $\mathbb{E}$. The former has the shape $\langle \mathsf{t} \mid \mathsf{S} \circ \mathsf{T}\rangle$ consisting of three components: the *active term* `t`, the *active evaluation stack* `S`, and the *forwarded evaluation stack* `T`. We will often use $\langle \mathsf{t} \mid \mathsf{S}\rangle$ as *syntactic sugar* for $\langle \mathsf{t} \mid \mathsf{S} \circ [] \rangle$ and conflate a term `t` with $\langle \mathsf{t} \mid [] \rangle$, *i.e.* its initial embedding in running computations. We will also write $\Mapsto_{\mathsf{eval}}$ for $\mapsto_{\mathsf{eval}} \cup \mapsto_{\mathsf{psh}}$.

$$\begin{array}{llll} \text{types} & \tau, \upsilon & ::= & \mathbb{1} \mid \mathbb{B} \mid \mathbb{Z} \mid \tau \times \upsilon \mid \tau \to \upsilon \\ \text{signatures} & \mathbb{E} & ::= & \{(\mathbf{op}_i : \tau_i \twoheadrightarrow \upsilon_i)_i\} \end{array}$$

(a) types

$$\begin{array}{llll} \text{values} & \mathsf{v}, \mathsf{w} & := & \langle\rangle \mid \mathbb{ff} \mid \mathbb{tt} \mid \mathsf{n} \\ & & \mid & x \mid \lambda x.\mathsf{t} \mid \boxed{f} \\ \text{effects} & \mathsf{e} & ::= & \mathbf{op}\,\mathsf{w} \mid \boxed{e} \\ \text{terms} & \mathsf{t}, \mathsf{u} & ::= & \mathbf{ret}\,\mathsf{v} \mid \mathbf{op}\,\mathsf{w} \mid \boxed{e} \mid \mathbf{new}\ \mathbb{E} \\ & & \mid & \mathsf{w}\,\mathsf{v} \mid \mathbf{if}\ \mathsf{v}\ \mathbf{then}\ \mathsf{t}\ \mathbf{else}\ \mathsf{u} \\ & & \mid & \mathsf{S}[\mathsf{t}] \end{array}$$

$$\begin{array}{llll} \text{handlers} & \mathsf{h} & := & \{\mathbf{ret}\,x \mapsto \mathsf{t}\} \mid \{\mathbf{op}\,x\,y \mapsto \mathsf{t}\} \uplus \mathsf{h} \\ \text{eval frames} & \mathsf{E} & ::= & \mathbf{let}\,x = [\,]\ \mathbf{in}\ \mathsf{t} \mid \{[\,]\}\ \mathbf{with}\ \mathsf{h} \\ \text{stack frames} & \mathsf{S}, \mathsf{T} & ::= & [\,] \mid \boxed{a}\,[\,] \mid \mathsf{T}[\mathsf{S}] \end{array}$$

(b) expressions

$$\mathsf{M}, \mathsf{N} ::= \langle \mathsf{t} \mid \mathsf{S} \circ \mathsf{T} \rangle \in \overline{\Lambda}_{\text{eff}}$$

(c) running computations

Fig. 4. Λ_{eff} syntax.

$$\begin{array}{rl} \langle (\lambda x.\mathsf{t})\,\mathsf{v} \mid \mathsf{S} \rangle & \mapsto_{\text{eval}} \langle \mathsf{t}\{\mathsf{v}/x\} \mid \mathsf{S} \rangle \\ \langle \mathbf{ret}\,\mathsf{v} \mid \mathsf{S}[\mathbf{let}\,x = [\,]\ \mathbf{in}\ \mathsf{t}] \rangle & \mapsto_{\text{eval}} \langle \mathsf{t}\{\mathsf{v}/x\} \mid \mathsf{S} \rangle \\ \langle \mathbf{ret}\,\mathsf{v} \mid \mathsf{S}[\{[\,]\}\ \mathbf{with}\ \mathsf{h}] \rangle & \mapsto_{\text{eval}} \langle \mathsf{t}\{\mathsf{v}/x\} \mid \mathsf{S} \rangle \qquad \text{when } \mathsf{h}^{\mathbf{ret}} = \{\mathbf{ret}\ x \mapsto \mathsf{t}\} \\ \langle \mathbf{op}\,\mathsf{v} \mid \mathsf{S}[\{[\,]\}\ \mathbf{with}\ \mathsf{h}] \circ \mathsf{T} \rangle & \mapsto_{\text{eval}} \langle \mathsf{t}\{\mathsf{v}/x\}\{\lambda z.\{\mathsf{T}[\mathbf{ret}\,z]\}\ \mathbf{with}\ \mathsf{h}/y\} \mid \mathsf{S} \rangle \\ & \text{when } \mathsf{h}^{\mathbf{op}} = \{\mathbf{op}\,x\,y \mapsto \mathsf{t}\} \end{array}$$

$$\begin{array}{rll} \langle \mathbf{op}\,\mathsf{v} \mid \mathsf{S}[\mathsf{E}] \circ \mathsf{T} \rangle & \mapsto_{\text{fwd}} \langle \mathbf{op}\,\mathsf{v} \mid \mathsf{S} \circ \mathsf{E}[\mathsf{T}] \rangle & \text{when } \mathbf{op} \notin \mathbf{hdl}(\mathsf{E}) \\ \langle e \mid \mathsf{S}[\mathsf{E}] \circ \mathsf{T} \rangle & \mapsto_{\text{fwd}} \langle e \mid \mathsf{S} \circ \mathsf{E}[\mathsf{T}] \rangle & (\star_1) \\ \langle \mathsf{E}[\mathsf{t}] \mid \mathsf{S} \rangle & \mapsto_{\text{psh}} \langle \mathsf{t} \mid \mathsf{S}[\mathsf{E}] \rangle & \\ \langle a[\mathsf{t}] \mid \mathsf{S} \rangle & \mapsto_{\text{psh}} \langle \mathsf{t} \mid \mathsf{S}[a\,[\,]] \rangle & (\star_2) \end{array}$$

Fig. 5. operational semantics

The only non-standard rules are the starred ones ($\star_1$) and ($\star_2$) involving the propagation of effect in presence of *abstract codata*. Notice that we did not add the unsound rule ($\times$) which may seem like the innocuous dual to ($\star_1$).

$$\langle \mathbf{op}\,\mathsf{v} \mid \mathsf{S}[\kappa\,[\,]] \circ \mathsf{T} \rangle \mapsto_{\text{fwd}} \langle \mathbf{op}\,\mathsf{v} \mid \mathsf{S} \circ \kappa[\mathsf{T}] \rangle \qquad (\times)$$

Indeed, as it will be evident in section 4, the ability to capture delimited continuations means that the *environment* can capture fragments of the *program*'s evaluation stack, and

as a result, gain control of a *program*'s handler. Therefore, the possibility of $\kappa\ []$ handling the effect (even though $\mathbf{op}\ \mathtt{v}$ is abstract from the environment's perspective) cannot *a priori* be ruled out.

3.3 Metatheory

3.3.1 Interactive Expressions. In the following, we will mostly be interested in the *interactive* subset of terms and computations on which our OGS semantics operate.

Definition 3.1. A term $\mathtt{t}$ (resp. a running computation $\mathtt{M}$) is said to be *interactive* when it is well-typed and admits a typing of the form $\mathtt{I}; \Gamma; \emptyset \vdash_c \mathtt{t} : \bot$ (resp. $\mathtt{I}; \Gamma \vdash_c \mathtt{M}$).

3.3.2 Normal forms. Since *normal forms* describe the interaction interface between the program and its environment, we will privilege a useful presentation where we identify the *interaction patterns* that are causing a computation to be stuck. By expliciting these patterns, the normal forms are written as $\mathtt{P}(a)$; they involve a *co-pattern* $\mathtt{P}$ (described in fig. 7) representing the program controlled code and a *name* a abstracting an environment controlled codata. The token $\square$ represents a *codata hole* whose *filling*, *i.e.* the syntactic substitution thereof, is given by the application $(\mathtt{P}, x) \mapsto \mathtt{P}(x)$ of *co-patterns* to *codata* defined as $\mathtt{P}(x) := \mathtt{P}\{\square \mapsto x\}$.

$$
\begin{array}{rcll}
\mathtt{Nf} & ::= & \langle \mathbf{ret}\ \mathtt{v} \mid c\ [\,]\ \rangle \mid \langle \mathbf{ret}\ \mathtt{v} \mid \mathtt{S}[\kappa\ [\,]] \rangle & \textit{evaluated returners} \\
& \mid & \langle \mathtt{e} \mid \mathtt{S}[\kappa\ [\,]] \circ \mathtt{T} \rangle \mid \langle \mathtt{e} \mid c\ [\,] \circ \mathtt{T} \rangle & \textit{unhandled effects} \\
& & \text{where } \mathtt{e} ::= e \mid \mathbf{op}\ \mathtt{v} & \\
& \mid & \langle f\ \mathtt{v} \mid \mathtt{S} \rangle & \textit{open applications}
\end{array}
$$

Fig. 6. interactive normal forms of Λ_{eff}.

Borrowing from the terminology of [Biernacki et al. 2020], we identify in fig. 6 *open-stuck* terms in which the reduction of the term depends on an abstract function f or an abstract delimited continuation $\kappa\ []$ and where the continuation is controlled by the program, as well as *context-stuck* terms corresponding to unhandled effects and fully-evaluated returners inside an abstract continuation $c\ []$. The two classes of normal forms correspond to different forms of interaction; the former calls for a *question* whereas the latter for an *answer*.

Lemma 3.2 (normal forms shape). *For all interactive computations* $\mathtt{I}; \Gamma \vdash_c \mathtt{M}$ *such that* $\mathtt{M} \not\mapsto_{\text{op}}$, *there exists a co-pattern* $\mathtt{P}$ *and a name* $a \in \mathrm{dom}(\Gamma)$ *s.t.* $\mathtt{M} = \mathtt{P}(a)$.

Our notion of program equivalence builds on the *Closed Instantiations of Uses* (CIU) equivalence, introduced by Mason and Talcott [Mason and Talcott 1991]. Whereas Morris-style contextual equivalence [Morris Jr 1969] quantifies over general contexts—including those placing holes under λ-abstractions to handle open terms—CIU-equivalence restricts attention to evaluation contexts and handles open terms by quantifying over substitution maps as well.

In our setting, we refine this idea by leveraging the distinction between variables and names: we define equivalence only for interactive terms $\mathtt{I}; \Gamma \vdash_c \mathtt{M}$, where $\mathtt{M}$ contains no free variables, and all names are drawn from Γ. We assume that Γ contains exactly one

$$\begin{array}{lll} \mathsf{P} & ::= \langle \mathbf{ret}\ \mathsf{v} \mid \Box \rangle \mid \langle \mathsf{e} \mid \Box \circ \mathsf{S} \rangle & \textit{context-stuck} \\ & \mid \langle \mathbf{ret}\ \mathsf{v} \mid \mathsf{S}[\Box] \rangle \mid \langle \mathsf{e} \mid \mathsf{S}[\Box] \circ \mathsf{T} \rangle & \textit{open-stuck} \\ & \mid \langle \Box\ \mathsf{v} \mid \mathsf{S} \rangle & \\ \text{where } \mathsf{e} & ::= e \mid \mathbf{op}\ \mathsf{v} & \end{array}$$

(a) syntax of co-patterns

$$\textsc{control-stuck}\ \frac{\textsc{cont-val}\ \dfrac{\mathtt{I};\Gamma \vdash_{\mathsf{v}} \mathsf{v} : \tau}{\mathtt{I};\Gamma \vdash \langle \mathbf{ret}\ \mathsf{v} \mid \Box \rangle : \neg\neg\tau}}{} \qquad \frac{\textsc{cont-eff}\ \dfrac{\mathtt{I};\Gamma;\vdash_{\mathsf{c}} \mathsf{e} : \tau \qquad \mathtt{I};\Gamma \vdash_{\mathsf{s}} \mathsf{S} : \tau \rightsquigarrow \upsilon}{\mathtt{I};\Gamma \vdash \langle \mathsf{e} \mid \Box \circ \mathsf{S} \rangle : \neg\neg\upsilon}}{}$$

$$\textsc{open-stuck}\ \frac{\textsc{lambda}\ \dfrac{\mathtt{I};\Gamma \vdash_{\mathsf{v}} \mathsf{v} : \tau \qquad \mathtt{I};\Gamma \vdash \mathsf{S} : \neg\upsilon}{\mathtt{I};\Gamma \vdash \langle \Box\ \mathsf{v} \mid \mathsf{S} \rangle : \neg(\tau \to \upsilon)}}{} \qquad \frac{\textsc{delim-cont}\ \dfrac{\mathtt{I};\Gamma \vdash \mathsf{P} : \neg\neg\tau \qquad \mathtt{I};\Gamma \vdash \mathsf{S} : \neg\upsilon}{\mathtt{I};\Gamma \vdash \mathsf{P}(\mathsf{S}[\Box]) : \neg(\tau \rightsquigarrow \upsilon)}}{}$$

(b) typing judgements for co-patterns

Fig. 7. co-patterns: syntax and types.

continuation name, no delimited continuation names, and that all types in Γ are free of effect signatures $\mathbb{E}$. This last restriction ensures that no effect instances can be exchanged between the program and its environment. We refer to such typing contexts as initial, since these constraints do not generally hold as interaction proceeds. Equivalence between $\mathtt{M}_1$ and $\mathtt{M}_2$ can then be defined as each approximating the other. Quantifying over name assignment for Γ, the continuation name d of Γ is then mapped to an evaluation context. This provides a definition of CIU equivalence quantifying solely on name assignment.

Definition 3.3. **(ciu-approximation)** We consider two interactive terms $\mathtt{M}_1, \mathtt{M}_2$ such that both $\mathtt{I};\Gamma \vdash_{\mathsf{c}} \mathtt{M}_i$ (for $i \in \{1,2\}$), with Γ initial. Then $\mathtt{M}_1$ is said to be *ciu-approximated* by $\mathtt{M}_2$, written $\mathtt{I};\Gamma \vdash \mathtt{M}_1 \preceq_{ciu} \mathtt{M}_2$, when for all instance contexts $\mathtt{I}' \supseteq \mathtt{I}$, for all continuation names c_{f} and for all name assignments γ such that $\mathtt{I}'; c_{\mathsf{f}} : \neg\mathbb{1} \vdash \gamma : \Gamma$, if $\mathtt{M}_1\{\gamma\} \Downarrow_{\mathrm{op}} c_{\mathsf{f}}[\langle\rangle]$ then $\mathtt{M}_2\{\gamma\} \Downarrow_{\mathrm{op}} c_{\mathsf{f}}[\langle\rangle]$.

Equivalence between $\mathtt{M}_1$ and $\mathtt{M}_2$ is then defined via mutual approximation. Quantifying over all name assignments for Γ, the continuation name d is interpreted as an evaluation context. This yields a form of CIU-equivalence based solely on quantification over name assignments.

We note that this definition excludes terms with names of positive types such as $\mathbb{B}$ or $\mathbb{Z}$, which should rather be handled using free variables. A mild generalization of our approach could accommodate such terms by also quantifying over variable substitutions. However, for the sake of simplicity, we leave this extension to future work.

4 Abstract Interaction

The abstract interaction LTS is a bi-labelled transition system. It describes the *interactive execution* of a term (*Proponent*) inside and abstract *program context* (*Opponent*). Throughout the execution, both parties play the role of an *active term* as well as that of an *active evaluation stack*, which is reflected in *Proponent*'s alternating between *active* and *passive*

states, hence the bipartite part. The transitions mark this interactive role swapping and their corresponding labels correspond to the active *Player's* move. The internal operational evaluation of *Proponent* (in an active state) are accounted for by silent transitions.

4.1 Nominal abstraction

When an *active term* reaches a *normal form*, it discloses the information needed for the *environment* to continue computing a value. This is done through the *abstraction process* whereby the *codata* occurring in the active *normal form* are abstracted away and transformed into *names* that are handed over to the *environment* as part of various *nominal patterns*, capturing the fact that the latter's behaviour can only be parametric *w.r.t.* these *codata* (unlike *data* that could be inspected to trigger a *data-dependent* reaction).

In a dual manner, upon receiving a *nominal pattern*, the *passive* player (who knows its type) infers the types of the underlying disclosed names through a *type inference system*.

In the following, we will define the *nominal abstraction* of a few syntactic categories, where we will introduce, for each, their *abstract* counterpart, the *type inference system* as well as the generating *focussing process* that transforms a *syntactic object* into an ultimate pattern [Lassen and Levy 2008] (its positive skeleton) together with a corresponding name assignment map (its negative filling).

4.1.1 Abstract values. The values exchanged between *Proponent* and *Opponent* are abstracted away, by transforming a value into a *nominal ultimate pattern* (nup) $\mathtt{A}$ defined over a set of *function names* $f \in \mathcal{L}$:

$$\mathtt{V}, \mathtt{W} := \langle\rangle \mid \mathtt{n} \mid \mathtt{ff} \mid \mathtt{tt} \mid f \mid \langle \mathtt{V}, \mathtt{W} \rangle$$

We define in fig. 8 the *abstraction process* of values, where $\nearrow$ denotes the corresponding *focussing process*.

4.1.2 Abstract generalised values. Not only the structure of values is exposed in the interaction of *Opponent* and *Proponent*, but so are some fully-evaluated computations. These *generalised values*[1] are in turn abstracted away into a *nominal computation pattern* (ncp) which is defined as:

$$\underline{\mathtt{A}}, \underline{\mathtt{B}} := \mathbf{ret}\ \mathtt{A} \mid e \mid r[e]$$

where $e \in \mathcal{E}$ is an abstract *effect name* and $r \in \mathtt{List}(\mathcal{K})$ is a *abstract (delimited) evaluation stack* defined over the set $\mathcal{K}$ of *delimited continuations names*.

Analogously to abstract values, we lift the *focusing process* to one that transforms a computation in normal form into an abstract computation. We will denote it by the symbol $\nearrow\!\!\!\!\nearrow$.

4.1.3 Abstract normal forms. Continuing with our nominal abstraction of expressions, we introduce *abstract normal forms* and *abstract copatterns* (acp) and for the sake of a uniform treatment in-keeping with the classification of copatterns of fig. 7, we will express them in terms of *abstract computations* in order to exhibit the underlying *generalized* questions and answers structure.

[1] computations that *are* but cannot *do*.

$$\frac{\emptyset \vdash_{\mathsf{v}} \mathsf{v} : \tau \qquad \tau \in \{\mathbb{1}, \mathbb{Z}, \mathbb{B}\}}{\mathsf{v} \nearrow (\mathsf{v}, \varepsilon)} \qquad \frac{}{\lambda x.\mathtt{t} \nearrow (f; [f \mapsto \lambda x.\mathtt{t}])}$$

$$\frac{\mathsf{v} \nearrow (\mathsf{V}; \gamma) \qquad \mathsf{w} \nearrow (\mathsf{W}; \gamma')}{\langle \mathsf{v}, \mathsf{w} \rangle \nearrow (\langle \mathsf{V}, \mathsf{W} \rangle, \gamma \cdot \gamma')} \qquad \frac{}{f \nearrow (g; [g \mapsto f])}$$

$$\frac{\emptyset \vdash_{\mathsf{v}} \mathsf{v} : \tau \qquad \tau \in \{\mathbb{1}, \mathbb{Z}, \mathbb{B}\}}{\Gamma \Vdash_{\mathsf{v}} \mathsf{v} : \tau \triangleright \emptyset} \qquad \frac{}{\Gamma \Vdash_{\mathsf{v}} f : \tau \to \upsilon \triangleright [f \mapsto (\tau \to \upsilon)]}$$

(a) abstraction of values.

$$\frac{\mathsf{v} \nearrow (\mathsf{A}; \gamma)}{\mathbf{ret}\, \mathsf{v} \nearrow\!\!\nearrow (\mathbf{ret}\, \mathsf{A}; \gamma)} \qquad \frac{}{\mathbf{op}\, \mathsf{v} \nearrow\!\!\nearrow (e; [e \mapsto \mathbf{op}\, \mathsf{v}])}$$

$$\frac{\Gamma \Vdash_{\mathsf{v}} \mathsf{A} : \tau \triangleright \Delta}{\Gamma \Vdash_{\mathsf{c}} \mathbf{ret}\, \mathsf{A} : \tau \triangleright \Delta; \emptyset} \qquad \frac{}{\Gamma \Vdash_{\mathsf{c}} e : ? \triangleright [e \mapsto ?]; \emptyset} \qquad \frac{\Gamma \vdash_{\mathsf{c}} r[e] : \tau}{\Gamma \Vdash_{\mathsf{c}} \kappa :: r[e] : \upsilon \triangleright \emptyset; [\kappa \mapsto \tau \rightsquigarrow \upsilon]}$$

(b) abstraction of generalised values.

Fig. 8. nominal abstraction process.

$$\begin{array}{rrcll} \text{CoPatterns} & \ni & \mathsf{p}, \mathsf{q} & ::= & \langle \underline{\mathsf{A}} \mid \Box \rangle & \textit{context-stuck} \\ & & & \mid & \langle \Box\, \underline{\mathsf{A}} \mid d \rangle \mid \langle \Box\, \mathsf{A} \mid d \rangle & \textit{open-stuck} \end{array}$$

where $\langle r[e] \mid \Box \rangle$, $\langle \Box[r[e]] \mid d \rangle$ and $\langle \Box[\mathbf{ret}\, \mathsf{A}] \mid d \rangle$ are *syntactic sugar* for $\langle e \mid \Box \circ r \rangle$, $\langle e \mid d[\Box] \circ r \rangle$ and $\langle \mathbf{ret}\, \mathsf{A} \mid d[\Box] \rangle$ respectively, and where the *hole filling* is given by the application $(\mathsf{p}, a) \mapsto \mathsf{p}[a]$ where $\mathsf{p}[a] := \mathsf{p}\{\Box := a\}$.

In fig. 24 we introduce the abstraction process of *copatterns*. Naturally, it produces an *abstract copattern*, a name assignment γ, in addition to a component ξ, the *abstract forward*, carrying information about the *forward evaluation stack*. It is either of the shape $\varnothing$ (indicating the absence of effects) or (e, r, δ) (indicating a performed effect e to be forwarded or handled) where r represents the *abstract (delimited) stack* through which said effect has been propagated and δ is another name assignment for abstracting the effects and forward evaluation frames.

Example 4.1. Perhaps the simplest example is a fully-evaluated returner in the form of $c[\mathbf{ret}\, \mathsf{v}]$ that can be written as $(\Box\, \mathbf{ret}\, \mathsf{v})(c)$ and whose abstraction can be expressed as an abstract copattern $\mathsf{p} = \langle \mathbf{ret}\, \mathsf{A} \mid \Box \rangle$ and the abstract codata c.

Remark 1. *Notice how the abstraction process of normal forms is not total. It is only defined on effectful normal forms of a specific shape (involving* forward frames F*) that reflects the order of effect propagation and the incrementation of the* abstract forward *mirrors this gradual interactive process. The choice of keeping γ and δ (of ξ) and to combine abstract codata belonging to both the program and the environment in r has a technical benefit as well as a conceptual one; as it maintains the distinction between passive interactions and interactions with observation.*

$$\textsc{context-stuck}\ \frac{\mathsf{v} \nearrow (\mathsf{A}, \gamma_{\mathsf{A}})}{\langle \mathbf{ret}\, \mathsf{v} \mid \square \rangle \mathrel{/\!/\!\nearrow} (\langle \mathbf{ret}\, \mathsf{A} \mid \square \rangle, \gamma_{\mathsf{A}}, \varnothing)}\ \textsc{return}$$

$$\frac{\mathsf{v} \nearrow (\mathsf{A}, \delta_{\mathsf{A}})}{\langle \mathbf{op}\, \mathsf{v} \mid \square \circ \mathsf{F} \rangle \mathrel{/\!/\!\nearrow} (\langle \kappa[\mathbf{op}\, \mathsf{A}\ \mathbf{as}\ e] \mid \square \rangle, \varepsilon, (e, [\kappa], \delta_{\mathsf{A}} \cdot [e \mapsto \mathbf{op}\, \mathsf{v}] \cdot [\kappa \mapsto \mathsf{F}]))}\ \textsc{perform}$$

$$\frac{}{\langle e \mid \square \circ \mathsf{F}[r] \rangle \mathrel{/\!/\!\nearrow} (\langle \kappa :: r[e] \mid \square \rangle, \varepsilon, (e, \kappa :: r, [\kappa \mapsto \mathsf{F}]))}\ \textsc{forward}$$

$$\textsc{open-stuck}\ \frac{\mathsf{v} \nearrow (\mathsf{A}, \gamma_{\mathsf{A}})}{\langle \square\, \mathsf{v} \mid \mathsf{S} \rangle \mathrel{/\!/\!\nearrow} (\langle \square\, \mathsf{A} \mid c \rangle, \gamma_{\mathsf{A}} \cdot [c \mapsto \mathsf{S}], \varnothing)} \qquad \frac{\mathsf{P} \mathrel{/\!/\!\nearrow} (\mathsf{p}, \gamma, \xi)}{\mathsf{C}[\mathsf{S}[\square]] \mathrel{/\!/\!\nearrow} (\mathsf{p}[c[\square]]], \gamma \cdot [c \mapsto \mathsf{S}], \xi)}$$

(a) abstracting copatterns

$$\textsc{context-stuck}\ \frac{\Gamma \Vdash_{\mathsf{c}} \underline{\mathsf{A}} : \tau \triangleright \Gamma_{\underline{\mathsf{A}}}; \Delta_{\underline{\mathsf{A}}}}{\Gamma \Vdash \langle \underline{\mathsf{A}} \mid \square \rangle : \neg\neg\tau \triangleright \Gamma_{\underline{\mathsf{A}}}; \Delta_{\underline{\mathsf{A}}}}$$

$$\textsc{open-stuck}\ \frac{\Gamma \Vdash_{\mathsf{c}} \underline{\mathsf{A}} : \tau \triangleright \Gamma_{\underline{\mathsf{A}}}; \Delta_{\underline{\mathsf{A}}}}{\Gamma \Vdash \langle \square\, \underline{\mathsf{A}} \mid d \rangle : \neg(\tau \rightsquigarrow v) \triangleright \Gamma_{\underline{\mathsf{A}}} \cdot [d \mapsto \neg v]; \Delta_{\underline{\mathsf{A}}}} \qquad \frac{\Gamma \Vdash_{\mathsf{v}} \mathsf{A} : \tau \triangleright \Gamma_{\mathsf{A}}}{\Gamma \Vdash \langle \square\, \mathsf{A} \mid d \rangle : \neg(\tau \to v) \triangleright \Gamma_{\mathsf{A}} \cdot [d \mapsto \neg v]; \emptyset}$$

(b) copatterns typing rules.

Fig. 9. abstraction process of copatterns.

4.2 Abstract Interactive LTS

In the *abstract interaction*, the executing term (*Proponent*) will be represented by an LTS configuration in order to be evaluated against an *abstract environment* (*Opponent*) that models all compatible *program environments* while exhibiting the *interaction* between the two that gets obscured by plain *syntax substitution*. To this effect, the configurations will carry an *information* component that keeps track of the history of the *interaction*. It consists of a map between the names and the *concrete* code of codata that *Proponent* has disclosed and rendered accessible to *Opponent*.

We define next the abstract interactive LTS $\mathcal{L}_{\mathsf{AI}}$ that is given by $(\mathcal{A}, \mathcal{M}, \xrightarrow{\mathbf{m}}_{\mathsf{I}})$.

4.2.1 *configurations, moves and transitions.*

configurations	$\mathcal{A}$	$\ni$	$\mathbb{I}, \mathbb{J}$	$::=$	$\langle \mathtt{I}; \gamma; \xi \rangle$	*active state*
				$\mid$	$\langle \mathtt{M}; \gamma; \delta \rangle$	*passive state*
moves	$\mathcal{M}$	$\ni$	$\mathbf{m}$	$:=$	$\boldsymbol{a}.\mathsf{p}^{\oplus} \mid \boldsymbol{a}.\mathsf{p}^{\ominus}$	

The moves are given by the previously described *abstract copatterns* where the action $\boldsymbol{a}.\mathsf{p}$ is to be understood as the combination of an *interaction handle* a and an input p: the abstract codata a belonging to the *passive* player is being probed with the *active* player's abstract copatern p.

The superscript $\oplus$ (resp. $\ominus$) is a convenient notation to indicate that *Proponent* (resp. *Opponent*) is the *active* player performing the *move*. We will sometimes range over these polarized moves by $\mathbf{p}$ and $\mathbf{o}$ respectively.
Breaking down all the possible *interactive transitions* witnessed by $\boldsymbol{a}$.p, we have:

$\boldsymbol{c}.\langle \mathbf{ret}\, \mathsf{A} \mid \Box \rangle^{\oplus}$	returning a value to the abstract context c [].
$\boldsymbol{c}.\langle r[e] \mid \Box \rangle^{\oplus}$	propagating e and the delimiting continuation r to the abstract context c [].
$\boldsymbol{f}.\langle \Box\, \mathsf{A} \mid d \rangle^{\oplus}$	requesting the result of the application $f\, \mathsf{A}$ in d [].
$\boldsymbol{\kappa}.\langle \Box[\mathbf{ret}\, \mathsf{A}] \mid d \rangle^{\oplus}$	requesting the result of the computation $\kappa[\mathbf{ret}\, \mathsf{A}]$ in d [].
$\boldsymbol{\kappa}.\langle \Box[r[e]] \mid d \rangle^{\oplus}$	delegating the handling or propagation of e with its delimiting continuation r to κ and requesting the result in d [].

After a series of internal reduction steps $\mapsto^{*}_{\mathsf{op}}$, the *active* configuration reaches a *normal form*. It performs a transition into an *passive state* where the *codata* in the corresponding copattern is *abstracted* then disclosed or forwarded to the environment via the information components.

A transition from a *passive configuration* admits *branching*. Indeed, since the *program environment* is abstract and not fixed, it can use any of the disclosed information to perform a *move* triggering a transition into an *active configuration* and the *concretization* of a running computation.
The transitions as well as the underlying abstraction and concretization processes are defined in fig. 11 and fig. 10, respectively.

For a compact presentation, we will abuse the notation of maps and substitution on ξ when it is the constituting component δ that is intended instead.

Remark 2. *As mentioned in section 3.2 where we have introduced* $\mapsto_{\mathsf{op}}$*, the reduction* $\mapsto_{\mathsf{fwd}}$ *captures a flow of information; that of effect propagation. A series of* $\mapsto_{\mathsf{fwd}}$ *reductions not followed by an* $\mapsto_{\mathsf{eval}}$ *reduction results in an unhandled effect forwarded to the environment, which in turn could potentially handle it, capturing the associated delimited continuation in the process.*

This information flow is a form of interaction between the program and its environment consisting in inquiring whether the environment handles the effect or not. The abstract effect and its enclosing evaluation context are provisionally forwarded as a fragment of the "would-be" captured delimited continuation in case the environment cacthes the effect and handles it. In case it does not, a pseudo copy-cat action (forwarding back the same effect but with an extended evaluation conext) is expected instead.

By way of example, we will give an informal presentation of the *interaction* of a term with a *concrete* program context, which will map to one possible *interactive execution path* against an *abstract environment* in $\mathcal{L}_{\mathsf{AI}}$.

5 From traces to handling structures

Consider the interaction of the term $\mathsf{u} := \mathbf{let}\, y = \mathbf{new}\, \mathbb{E}\ \mathbf{in}\, \{f\ (\lambda x.\mathbf{op}\ \langle\rangle)\}\ \mathbf{with}\, \mathsf{h}^{y}$ (from example **??**) with the environment given by $\mathsf{C} := \mathbf{let}\, f = (\lambda g.g\ \mathsf{v}; \mathbf{ret}\, \mathbb{t})\ \mathbf{in}\ []$ was represented

$$\text{PURE}\quad \frac{\mathsf{P} \nRightarrow (\mathsf{p}, \gamma, \varnothing)}{\mathbf{abstract}(\mathsf{P}(a), \varnothing, \_) \nRightarrow (\boldsymbol{a}.\mathsf{p}, \gamma, \varnothing)}$$

$$\text{EFFECTFUL}\quad \frac{\langle \mathbf{op}\ \mathsf{v} \mid \mathsf{S}[\Box] \circ \mathsf{F}\rangle \nRightarrow (\mathsf{p}, \gamma, \xi)}{\mathbf{abstract}(\langle \mathbf{op}\ \mathsf{v} \mid \mathsf{S}[a\ []] \circ \mathsf{F}\rangle, \varnothing, v) \nRightarrow (\boldsymbol{a}.\mathsf{p}, \gamma, \xi)}\ \text{PERFORM}$$

$$\frac{\langle e \mid \mathsf{S}[\Box] \circ \mathsf{F}[r]\rangle \nRightarrow (\mathsf{p}, \gamma, \xi)}{\mathbf{abstract}(\langle \mathsf{e} \mid \mathsf{S}[a\ []] \circ \mathsf{F}[r]\rangle, (e, r, \_), \_) \nRightarrow (\boldsymbol{a}.\mathsf{p}, \gamma, \xi)}\ \text{FORWARD}$$

(a) abstracting normal forms

$$\text{PURE/PERFORM}\quad \frac{}{\mathbf{concretize}(\boldsymbol{a}.\mathsf{p}, \gamma, \varnothing) = \mathsf{p}(\gamma(a))}$$

$$\text{FORWARD}\quad \frac{\xi = (e, r, \delta) \qquad \mathsf{p} = \langle \kappa :: r[e] \mid \_\rangle \qquad \mathsf{P} = \mathsf{p}\{\delta(e)/e\}}{\mathbf{concretize}(\boldsymbol{a}.\mathsf{p}, \gamma, \xi) = \mathsf{P}(\gamma(a))}$$

(b) concretizing abstract normal forms

Fig. 10. the dualizing process.

by the following trace

$$\mathsf{s_u} = \boldsymbol{f}.\langle \Box\, g \mid d\rangle^{\oplus}\ \boldsymbol{g}.\langle \Box\, \mathsf{A} \mid c\rangle^{\ominus}\ \boldsymbol{c}.\langle \kappa[e] \mid \Box\rangle^{\oplus}\ \boldsymbol{d}.\langle \kappa_c :: \kappa[e] \mid \Box\rangle^{\ominus}\ \boldsymbol{\kappa_c}.\langle \Box[\mathbf{ret}\ 5] \mid d'\rangle^{\oplus}$$

Now considering the term $\mathsf{t} := f\ (\lambda x.5)$, that is contextually equivalent to u, and observing how it interacts with the same environment C, we get:

$$\begin{array}{lll}
\langle c_{\mathsf{f}}[\mathsf{t}], \varepsilon, \varnothing\rangle & \xrightarrow{\boldsymbol{f}.\langle \Box\, g \mid d\rangle^{\oplus}}_{\mathsf{I}} & \langle [g \mapsto (\lambda x.5);\ d \mapsto c_{\mathsf{f}}[[]]];\ \varnothing\rangle \\
& \xrightarrow{\boldsymbol{g}.\langle \Box\, \mathsf{A} \mid c\rangle^{\ominus}}_{\mathsf{I}} & \langle c[(\lambda x.5)]\ \mathsf{A};\ [g \mapsto (\lambda x.5);\ d \mapsto c_{\mathsf{f}}[[]]];\ \varnothing\rangle \\
& \xrightarrow{\boldsymbol{c}.\langle \mathbf{ret}\ 5 \mid \Box\rangle^{\oplus}}_{\mathsf{I}} & \langle [g \mapsto (\lambda x.5);\ d \mapsto c_{\mathsf{f}}[[]]];\ \varnothing\rangle \\
& \xrightarrow{\boldsymbol{d}.\langle \mathbf{ret}\ \mathtt{tt} \mid \Box\rangle^{\ominus}}_{\mathsf{I}} & \langle c_{\mathsf{f}}[\mathbf{ret}\ \mathtt{tt}], [g \mapsto (\lambda x.5);\ d \mapsto c_{\mathsf{f}}[[]]];\ \varnothing\rangle \\
& \xrightarrow{\boldsymbol{c_{\mathsf{f}}}.\langle \mathbf{ret}\ \mathtt{tt} \mid \Box\rangle^{\oplus}}_{\mathsf{I}} & \langle [g \mapsto (\lambda x.5);\ d \mapsto c_{\mathsf{f}}[[]]];\ \varnothing\rangle
\end{array}$$

that is an interaction witnessed by the trace $\mathsf{s_t}$

$$\mathsf{s_t} = \boldsymbol{f}.\langle \Box\, g \mid d\rangle^{\oplus}\ \boldsymbol{g}.\langle \Box\, \mathsf{A} \mid c\rangle^{\ominus}\ \boldsymbol{c}.\langle \mathbf{ret}\ 5 \mid \Box\rangle^{\oplus}\ \boldsymbol{d}.\langle \mathbf{ret}\ \mathtt{tt} \mid \Box\rangle^{\ominus}\ \boldsymbol{c_{\mathsf{f}}}.\langle \mathbf{ret}\ \mathtt{tt} \mid \Box\rangle^{\oplus}$$

It is clear that $\mathsf{s_u}$ is not a trace generated by t, and $\mathsf{s_t}$ is not a trace generated by u, so that their respective set of complete traces are incomparable. Therefore, a complete OGS model requires a notion of equivalence coarser than equality of traces, that would identify

$$\textsc{Eval}\ \frac{\mathsf{M} \mapsto_{\mathsf{eval}} \mathsf{N}}{\langle \mathsf{M};\ \gamma\rangle \xrightarrow{\mathsf{eval}}_{\mathsf{I}} \langle \mathsf{N};\ \gamma\rangle}$$

(a) Internal evaluation

$$\textsc{Px}\ \frac{\mathbf{abstract}(\mathsf{Nf}) \nearrow\!\!\!\nearrow (\boldsymbol{a}.\mathsf{p},\ \gamma_{\mathsf{p}})}{\langle \mathsf{Nf};\ \gamma\rangle \xrightarrow{\boldsymbol{a}.\mathsf{p}^{\oplus}}_{\mathsf{I}} \langle \mathsf{I};\ \gamma{\cdot}\gamma_{\mathsf{p}}\rangle} \qquad \textsc{Ox}\ \frac{}{\langle \mathsf{I};\ \gamma\rangle \xrightarrow{\boldsymbol{a}.\mathsf{p}^{\ominus}}_{\mathsf{I}} \langle \mathsf{p}[\gamma(a)];\ \gamma\rangle}$$

(b) Pure moves

$$\textsc{Forward}\ \frac{\mathbb{I} \overset{\mathsf{s}}{\Rightarrow}_{\mathbf{fwd}} \mathbb{J}}{\mathbb{I} \overset{\mathsf{s}}{\Rightarrow}_{\mathsf{I}} \mathbb{J}}$$

(c) Effect forwarding sequences

$$\textsc{Forward}\ \frac{\mathsf{M} \mapsto_{\mathsf{fwd}} \mathsf{N}}{\langle \mathsf{M};\ \gamma;\ \xi\rangle \rightarrow_{\mathbf{fwd}} \langle \mathsf{N};\ \gamma;\ \xi\rangle}$$

$$\textsc{PHandle}\ \frac{\mathsf{M} \mapsto_{\mathsf{eval}}}{\langle \mathsf{M};\ \gamma;\ !\delta\rangle \xrightarrow{hdl^{\oplus}}_{\mathbf{fwd}} \langle \mathsf{M}\{\gamma\};\ \gamma\rangle} \qquad \textsc{OHandle}\ \frac{}{\langle \mathsf{I};\ \gamma;\ ?\xi\rangle \xrightarrow{hdl^{\ominus}}_{\mathbf{fwd}} \langle \mathsf{I};\ \gamma{\cdot}\xi\rangle}$$

(d) Internal effect manipulation

$$\textsc{Peff}\ \frac{\mathbf{abstract}(\mathsf{Nf},\ \delta) \nearrow\!\!\!\nearrow (\boldsymbol{a}.\mathsf{p},\ \gamma_{\mathsf{p}},\ \xi)}{\langle \mathsf{Nf};\ \gamma;\ \delta\rangle \xrightarrow{\boldsymbol{a}.\mathsf{p}^{\oplus}}_{\mathbf{fwd}} \langle \mathsf{I};\ \gamma{\cdot}\gamma_{\mathsf{p}};\ \xi\rangle} \qquad \frac{\mathbf{concretize}(\boldsymbol{a}.\mathsf{p},\ \gamma,\ \xi) = (\mathsf{M},\ \delta)}{\langle \mathsf{I};\ \gamma;\ \xi\rangle \xrightarrow{\boldsymbol{a}.\mathsf{p}^{\ominus}}_{\mathbf{fwd}} \langle \mathsf{M};\ \gamma;\ \delta\rangle}\ \textsc{Oeff}$$

(e) Effectful moves

Fig. 11. transitions of the Abstract Interaction LTS

s_{t} and s_{u}.

$$\boldsymbol{f}.\langle \square\, g \mid c\rangle^{\oplus}\ \boldsymbol{g}.\langle \square\, \mathsf{A} \mid d\rangle^{\ominus}\ \boldsymbol{d}.\langle \mathbf{ret}\ 5 \mid \square\rangle^{\oplus}\ \boldsymbol{c}.\langle \mathbf{ret}\ \mathrm{tt} \mid \square\rangle^{\ominus}$$
$$\simeq$$
$$\boldsymbol{f}.\langle \square\, g \mid c\rangle^{\oplus}\ \boldsymbol{g}.\langle \square\, \mathsf{A} \mid d\rangle^{\ominus}\ \boldsymbol{d}.\langle \kappa[e] \mid \square\rangle^{\oplus}\ \boldsymbol{c}.\langle \kappa_d :: \kappa[e] \mid \square\rangle^{\ominus}\ \boldsymbol{\kappa_d}.\langle \square[\mathbf{ret}\ 5] \mid c'\rangle^{\oplus}\ \boldsymbol{c'}.\langle \mathbf{ret}\ \mathrm{tt} \mid \square\rangle^{\ominus}$$

$$
\begin{array}{lll}
\langle c_{\mathsf{f}}[\mathsf{t}_2], \varepsilon, \varnothing\rangle & \xrightarrow{\text{eval}}_{\mathsf{I}} & \langle c_{\mathsf{f}}[\{f\ (\lambda x.\mathbf{op}\,\langle\rangle)\}\ \mathbf{with}\ \mathsf{h}^{\iota}];\ \varepsilon;\ \varnothing\rangle \\
& \xrightarrow{f.\langle \square\, g | d\rangle^{\oplus}}_{\mathsf{I}} & \langle [\iota \mapsto \mathbb{E}];\ [g \mapsto (\lambda x.\mathbf{op}\,\langle\rangle),\ d \mapsto c_{\mathsf{f}}[\{[]\}\ \mathbf{with}\ \mathsf{h}^{\iota}]];\ \varnothing\rangle \\
& \xrightarrow{g.\langle \square\, \mathsf{A} | c\rangle^{\ominus}}_{\mathsf{I}} & \langle c[(\lambda x.\mathbf{op}\langle\rangle)\ \mathsf{A}];\ \gamma;\ \varnothing\rangle \\
& \xrightarrow{\text{eval}}_{\mathsf{I}} & \langle c[\mathbf{op}\,\langle\rangle];\ \gamma;\ \varnothing\rangle \\
& \xrightarrow{c.\langle \kappa[e] | \square\rangle^{\oplus}}_{\mathsf{I}} & \langle [\iota \mapsto \mathbb{E}];\ \gamma;\ (!e,\ \mathbf{op}\,\langle\rangle,\ \kappa,\ [\kappa \mapsto []])\rangle \\
& \xrightarrow{d.\langle \kappa_c \cdot \kappa[e] | \square\rangle^{\ominus}}_{\mathsf{I}} & \langle \gamma(d)[\kappa_c :: \kappa[\mathbf{op}\langle\rangle]];\ \gamma;\ (!e,\ \mathbf{op}\,\langle\rangle,\ \kappa_c :: \kappa,\ [\kappa \mapsto []])\rangle \\
& \xrightarrow{hdl^{\oplus}}_{\mathsf{I}} & \langle c_{\mathsf{f}}[\{\kappa_c \circ\ []\ [\mathbf{op}\langle\rangle]\}\ \mathbf{with}\ \mathsf{h}^{\iota}];\ \gamma;\ \varnothing\rangle \\
& \xrightarrow{\text{eval}}_{\mathsf{I}} & \langle c_{\mathsf{f}}[\{\kappa_c[\mathbf{ret}\ 5]\}\ \mathbf{with}\ \mathsf{h}^{\iota}];\ \gamma;\ \varnothing\rangle \\
& \xrightarrow{\kappa_c.\langle \square[\mathbf{ret}\ 5] | d'\rangle^{\oplus}}_{\mathsf{I}} & \langle [\iota \mapsto \mathbb{E}];\ \gamma \cdot [d' \mapsto c_{\mathsf{f}}[\{[]\}\ \mathbf{with}\ \mathsf{h}^{\iota}]];\ \varnothing\rangle \\
& \xrightarrow{d'.\langle \mathbf{ret}\ \mathsf{tt} | \square\rangle^{\ominus}}_{\mathsf{I}} & \langle \gamma(d')[\mathbf{ret}\ \mathsf{tt}];\ \gamma';\ \varnothing\rangle \\
& \xrightarrow{\text{eval}}_{\mathsf{I}} & \langle c_{\mathsf{f}}[\mathbf{ret}\ \mathsf{tt}];\ \gamma';\ \varnothing\rangle \\
& \xrightarrow{c_{\mathsf{f}}.\langle \mathbf{ret}\ \mathsf{tt} | \square\rangle^{\oplus}}_{\mathsf{I}} & \langle [\iota \mapsto \mathbb{E}];\ \gamma';\ \varnothing\rangle
\end{array}
$$

5.1 Trace equivalence is too fine

5.1.1 *Analysis: handling structure.*

- forward moves introduce a lot of redundancy (new names for the same continuations)
- obscure the underlying handling structure
- it is worth restoring this handling structure in order to better understand what happens

Let us scrutinize the previous example and observe what is really happening at each step:

$$
\begin{array}{lll}
\mathsf{t}_{\mathsf{t}_2} = & f.\langle \square\, g \mid d\rangle^{\oplus} & \\
& g.\langle \square\, \mathsf{A} \mid c\rangle^{\ominus} & (fig.\ step1) \\
& c.\langle \kappa[e] \mid \square\rangle^{\oplus} & (fig.\ step2) \\
& d.\langle \kappa_c :: \kappa[e] \mid \square\rangle^{\ominus} & (fig.\ step3) \\
& \kappa_c.\langle \square[\mathbf{ret}\ 5] \mid d'\rangle^{\oplus} & (fig.\ step4) \\
& d'.\langle \mathbf{ret}\ \mathsf{tt} \mid \square\rangle^{\ominus}\ c_{\mathsf{f}}.\langle \mathbf{ret}\ \mathsf{tt} \mid \square\rangle^{\oplus} & (fig.\ step5)
\end{array}
$$

We observe that at (step 3), Opponent has no choice but to forward the effect e and that the move it makes is invisible from its perspective. Additionally, the continuation starting correspondin to c and delimited by d. Therefore, κ_c is essentially nothing but this delimited continuation repackaged, and if interrogated, it answers exactly what c would answer d had it been interrogated instead.

Now if we squash the steps 2 and 3, into one unobservable move, we can rewrite this trace as follows:

$$\begin{array}{rcll} \mathrm{t}_{\mathrm{t}_2} & = & f.\langle \square\, g \mid d\rangle^{\oplus} & \\ & & \boldsymbol{g}.\langle \square\, \mathsf{A} \mid c\rangle^{\ominus} & (fig.\ step1) \\ & & \mathbf{fwd}(c, \kappa_c, d) & \\ & & \boldsymbol{\kappa_c}.\langle \square[\mathbf{ret}\ 5] \mid d'\rangle^{\oplus} & (fig.\ step4) \\ & & \boldsymbol{d'}.\langle \mathbf{ret}\ \mathtt{tt} \mid \square\rangle^{\ominus}\, \boldsymbol{c_f}.\langle \mathbf{ret}\ \mathtt{tt} \mid \square\rangle^{\oplus} & (fig.\ step5) \end{array}$$

Knowing that if κ_c is interrogated in context given by d', it should react the same way like c, it becomes clear why this trace is equivalent to $\mathrm{t}_{\mathrm{t}_1}$, given that $\mathbf{fwd}(c, \kappa_c, d)$ cannot be observed, that $\kappa_c n$ plays the role of c and that d' plays the role of d.

In the following, we make this observation formal through a coarser trace equivalence.

*Definition 5.1 (**fwd**-transitions).* We call a **fwd**-transition any two consecutive *OGS* moves playing the same effect name. The second player's move consists of forwarding the effect of its opponent and giving them access to their continuation through a delimited continuation name.

We will denote **Ofwd**-transitions and **Pfwd**-transitions by $\mathbf{fwd}(\kappa)$ and $\overline{\mathbf{fwd}}(\kappa)$ respectively, where κ is the delimited continuation name it involves.

The grammar of **fwd**-transition.

$$\mathbf{fwd}(\kappa) ::= \mathbf{fwd}(c, d, \kappa) \mid \mathbf{fwd}(c, (\kappa_d, c'), \kappa) \mid \mathbf{fwd}(\kappa_c, d, \kappa)$$

$$\begin{array}{lcl} \mathbf{fwd}(c, d, \kappa_c) & ::= & \boldsymbol{c}.\langle r[e] \mid \square\rangle^{\ominus}\, \boldsymbol{d}.\langle \kappa_c :: r[e] \mid \square\rangle^{\oplus} \\ \mathbf{fwd}(c, (\kappa_d, c'), \kappa_c) & ::= & \boldsymbol{c}.\langle r[e] \mid \square\rangle^{\ominus}\, \boldsymbol{\kappa_d}.\langle \square[\kappa_c :: r[e]] \mid c'\rangle^{\oplus} \\ \mathbf{fwd}(\kappa, d, \kappa') & ::= & \boldsymbol{\kappa}.\langle \square[r[e]] \mid d\rangle^{\ominus}\, \boldsymbol{d}.\langle \kappa' :: r[e] \mid \square\rangle^{\oplus} \end{array}$$

*Definition 5.2 (**fwd**-sequences).* We will call a **fwd**-sequence any sequence of *OGS* moves starting with a sequence of **fwd** moves and ending with a non-**fwd** move, i.e a sequence of the form:

$$\mathbf{fwd}(\kappa_0)\ \cdots\ \mathbf{fwd}(\kappa_n)\ \mathbf{m}$$

For $X \in \{O, P\}$, an X-ending sequence will be called an **Xfwd** sequence.

Definition 5.3 (Trace well-parenthesisedness). Given two OGS tracest and s, we say that t is conditionally well-parenthesised given s and write $\mathcal{WP}(\mathrm{t}\,|\,\mathrm{s})$ when:

$$\langle [] \mid \varnothing \mid \emptyset\rangle \overset{\mathrm{s}}{\Rightarrow}_{wb} \langle \sigma \mid \eta \mid \phi\rangle \overset{\mathrm{t}}{\Rightarrow}_{wb} \langle \sigma \mid \eta' \mid \phi'\rangle \qquad \text{for some } \sigma, \eta, \eta', \phi \text{ and } \phi'.$$

Remark 3. *TODO: how this is different than the standard notion of a complete trace.*

In order to reason modulo **Ofwd**-transitions, we will start by characterizing the shape of traces differing by *one and only one* **Ofwd**-transition. For this we introduce the following move substitution that captures how the suffix of a trace is changed if one **Ofwd**-transition is suitably removed. It depends on a move $\mathbf{o} = \langle \kappa[\underline{\mathsf{A}}] \mid \overline{d}\rangle$ that features the resumption name κ that *Proponent* has introduced in the **Ofwd**-transition $\mathbf{fwd}(\kappa)$, because if $\mathbf{fwd}(\kappa)$ is to be removed from the trace then so should κ and d.

Definition 5.4 (Move substitution). We define the trace-dependent move substition $\partial(\mathbf{fwd}(\kappa),\ \mathbf{o})$ by pattern-matching on $\mathbf{fwd}(\kappa)$:

$$\begin{array}{lcl}
\partial(\mathbf{fwd}(c,d,\kappa_c),\ \langle \kappa_c[\underline{A}] \mid \overline{d'}\rangle) & := & [\langle \kappa_c[\underline{A}] \mid \overline{d'}\rangle \mapsto \langle \underline{A} \mid c\rangle] \cdot [d' \mapsto d] \\
\partial(\mathbf{fwd}(c,(\kappa_d,c'),\kappa_c),\ \langle \kappa_c[\underline{A}] \mid \overline{d'}\rangle) & := & [\langle \kappa_c[\underline{A}] \mid \overline{d'}\rangle \mapsto \langle \underline{A} \mid c\rangle] \cdot [\langle \underline{B} \mid \overline{d'}\rangle \mapsto \langle \overline{\kappa_d}[\underline{B}] \mid c'\rangle] \\
\partial(\mathbf{fwd}(\kappa,d,\kappa'),\ \langle \kappa'[\underline{A}] \mid \overline{d}\rangle) & := & [\langle \kappa'[\underline{A}] \mid \overline{d}\rangle \mapsto \langle \kappa[\underline{A}] \mid \overline{d}\rangle]
\end{array}$$

The next definition will relate two environments whose *intensional* behaviours are different; in that one captures an additional fragment of the program's continuation while the other does not, but at the same time this difference has no bearing on their *observable* behavior.

Definition 5.5 (O-canonical form). With the understading that writing $\mathsf{t}_1(\mathsf{s}\{\delta\})\mathsf{t}_2$ means that the substitution δ acts on the subtrace s, we define the O-canonical form of a trace t as its normal form w.r.t. the following rewriting rule:

$$\frac{\mathcal{WP}(\mathsf{t}_1 \,|\, \mathsf{t}) \qquad \kappa \notin \mathrm{supp}(\mathsf{t}_1) \cup \mathrm{supp}(\mathsf{t}_2) \qquad \delta = \partial(\mathbf{fwd}(\kappa),\ \langle \kappa[\underline{A}] \mid \overline{d}\rangle)}{\mathsf{t}\ \mathbf{fwd}(\kappa)\ \mathsf{t}_1\ \langle \kappa[\underline{A}] \mid \overline{d}\rangle\ \mathsf{t}_2\ \ \rightarrow_o\ \ \mathsf{t}\ \mathsf{t}_1\ (\langle \kappa[\underline{A}] \mid \overline{d}\rangle\ \mathsf{t}_2\{\delta\})}$$

We will write $[\mathsf{t}]_o$ for t' s.t $\mathsf{t} \rightarrow_o^* \mathsf{t}' \not\rightarrow_o$.

This rewriting rule relates two equivalent terms, in which the same handler has been installed in two different places. We can illustrate this by taking the environment's perspective, where K_i's are the environment's evaluation contexts, h is its handler such that $\mathbf{op} \in \mathbf{hdl}(\mathsf{h})$ and the κ_i's are the program's fragments.

$$(\{\mathsf{K}_1\}\ \mathbf{with}\ \mathsf{h}) \circ \kappa_1 \circ \mathsf{K}_2 \circ \kappa_2 \circ \mathsf{K}[\mathbf{op}\ \mathsf{v}] \xrightarrow{\simeq_{ctx}}_o \mathsf{K}_1 \circ \kappa_1 \circ (\{\mathsf{K}_2\}\ \mathbf{with}\ \mathsf{h}) \circ \kappa_2 \circ \mathsf{K}[\mathbf{op}\ \mathsf{v}] \quad (1)$$

If $\mathsf{h}^{\mathbf{op}} = \{\mathbf{op}\ x\ y \mapsto y\ \mathsf{w}\}$, for example, then the normal form w.r.t. $\rightarrow_o$ would correspond to the following term:

$$\mathsf{K}_1 \circ \kappa_1 \circ \mathsf{K}_2 \circ \kappa_2 \circ \mathsf{K}[\mathbf{ret}\ \mathsf{w}\{x \mapsto \mathsf{v}\}]$$

The condition $\kappa \notin \mathrm{supp}(\mathsf{t}_2) \cup \mathrm{supp}(\mathsf{t}_2')$ along with the presence of the move $\langle \kappa[\underline{A}] \mid \overline{d}\rangle$ guarantees that κ is used *linearly*. The condition $\mathcal{WP}(\mathsf{t}_1 \mid \mathsf{t})$ ensures that if $\mathbf{fwd}(\kappa)$ appears in a **Ofwd**-sequence propagating the same effect e, then it is the outermost one as illustrated in (1) and that it is used in a well-scoped manner.

*Definition 5.6 (**fwd**-commutation).*

$$\frac{\mathcal{WP}(\mathsf{s} \,|\, \mathsf{t}_1) \qquad \kappa \notin \mathrm{supp}(\mathsf{s})}{\mathsf{t}_1\ \mathsf{s}\ \mathbf{fwd}(\kappa)\ \mathsf{t}_2 \quad \mathscr{R}_o \quad \mathsf{t}_1\ \mathbf{fwd}(\kappa)\ \mathsf{s}\ \mathsf{t}_2}$$

Let $\mathscr{R}_o{}^*$ be the transitive closure of $\mathscr{R}_o$.

If the previous relation $\rightarrow_o$ is concerned with the placement of handlers, the relation $\mathscr{R}_o$, on the other hand, is concerned with the place where the effect has been performed.

To illustrate this, consider the equivalent terms $\mathsf{T}_1 = \mathsf{E}[\mathsf{w}_1\ \langle\rangle]$ and $\mathsf{T}_2 = \mathsf{E}[\mathsf{w}_2\ \langle\rangle]$ sucht that $\mathtt{I}; \Gamma, f : \mathbb{1} \rightarrow \mathbb{Z} \vdash_{\mathsf{c}} \mathsf{T}_1,\ \mathsf{T}_2$ and $\mathtt{I}(\iota) = \{\mathbf{get} : \mathbb{1} \rightarrow \mathbb{Z};\ \mathbf{set} : \mathbb{Z} \rightarrow \mathbb{1}\}$, where

$$\mathsf{w}_1 = \lambda\_.\ \mathbf{let}\ x = f\ \langle\rangle\ \mathbf{in}\ \mathbf{let}\ y = \mathbf{get}\ \langle\rangle\ \mathbf{in}\ \mathbf{ret}\ x + y$$

and

$$\mathtt{w}_2 = \lambda\_.\ \mathbf{let}\ x = \mathbf{get}\ \langle\rangle\ \mathbf{in}\ \mathbf{let}\ y = f\ \langle\rangle\ \mathbf{in}\ \mathbf{ret}\ x{+}y$$

If we fix an observing environment and consider the two corresponding interaction traces, we observe that:

$$[\![\mathtt{T}_1]\!]_{\mathrm{ogs}} \ni \quad \mathtt{t}\, \boldsymbol{f}.\langle \square\ \langle\rangle \mid d\rangle^{\oplus}\, \mathtt{s}'\boldsymbol{d}.\langle \mathbf{ret}\ \mathtt{A} \mid \square\rangle^{\ominus}\, \overline{\mathbf{fwd}}(\kappa)\ \mathtt{q}$$

if and only if

$$[\![\mathtt{T}_2]\!]_{\mathrm{ogs}} \ni \quad \mathtt{t}\, \overline{\mathbf{fwd}}(\kappa)\, \boldsymbol{f}.\langle \square\ \langle\rangle \mid d\rangle^{\oplus}\, \mathtt{s}'\boldsymbol{d}.\langle \mathbf{ret}\ \mathtt{A} \mid \square\rangle^{\ominus}\ \mathtt{q}$$

By writing s for the t-*well-parenthesised* sequence $\boldsymbol{f}.\langle \square\ \langle\rangle \mid d\rangle^{\oplus}\, \mathtt{s}'\boldsymbol{d}.\langle \mathbf{ret}\ \mathtt{A} \mid \square\rangle^{\ominus}$ we see how the commuting of s and $\overline{\mathbf{fwd}}(\kappa)$ does not affect the extensional behavior.

*Definition 5.7 (***fwd***-equivalence).* Two traces t and t′ are said to be **Ofwd**-equivalent, i.e equivalent up-to **Ofwd**-transitions, and written $\mathtt{t} \simeq_o \mathtt{t}'$ when $[\mathtt{t}]_o\ \mathscr{R}_o{}^*\ [\mathtt{t}']_o$.

Dually, we say that two traces t and t′ are **Pfwd**-equivalent, and write $\mathtt{t} \simeq_p \mathtt{t}'$ when $[\mathtt{t}^{\perp}]_o\ \mathscr{R}_o{}^*\ [\mathtt{t}'^{\perp}]_o$.

Definition 5.8 (Trace preorder).

$$\mathbf{Tr}_{\mathrm{ogs}}(\mathbb{G}) \preceq_{tr} \mathbf{Tr}_{\mathrm{ogs}}(\mathbb{H}) \ :\Longleftrightarrow\ \forall \mathtt{t} \in \mathbf{Tr}_{\mathrm{ogs}}(\mathbb{G}).\ \exists \mathtt{t}' \in \mathbf{Tr}_{\mathrm{ogs}}(\mathbb{H})\ \text{s.t.}\ \mathtt{t} \simeq_p \mathtt{t}'$$

5.2 Handling structures

As we have seen in the definition of trace equivalence.

- forward moves introduce a lot of redundancy (new names for the same continuations)
- obscure the underlying handling structure
- it is worth restoring this handling structure in order to better understand what happens

In that section, we proceeded by eliminating a only a certain subset of **fwd**-transitions and their corresponding delimited continuation invokations. That the had the benefit that the resulting squences remain *legal* traces (*i.e.* sequences that could potentially be generated by $\mathcal{L}_{\mathrm{OGS}}$).

Continuing our effort to eliminate the redundancy introduced by forward transitions, we revisit the role of delimited continuation names. These names are merely semantic artefacts—they are not part of the actual surface language. Once we remove their occurrences from the generated traces, a more natural and intrinsic structure begins to emerge. This underlying organization, which was previously obscured by syntactic overhead, is what we term *handling structures*. In this sense, the approach of transforming traces by relaxing or removing the linearity and well-parenthesisedness side conditions, in order to obtain canonical forms is, without question, the right direction. However, it comes at a cost: the resulting sequences, while more canonical in spirit, no longer conform to the *legality* constraints of OGS traces (they are not necessarily generated by $\mathcal{L}_{\mathrm{OGS}}$). To account for this, we introduce handler structures as a new formal device. These structures allow us to represent the essential phenomenon of capture—a central aspect of effectful computation—while simultaneously reducing redundancy. More importantly, they recover the underlying semantics of handling, which had been obfuscated by the machinery of

delimited continuation names and the complexity of the the justification structure they introduce.

Remark 4. *Algebraic effects and handlers are expressive enough to capture paradigms such as asynchronicity and cooperative green threads. Given this, it is perhaps unsurprising that the resulting semantic structures naturally exhibit patterns where "events" give rise to—or enable—other events, and these in turn can be observed as independent. This emergent structure, arising from the justification structure, is reminiscent of event structures commonly used to model concurrency and parallelism. It stands in contrast to the traditional approach of modeling interaction via interleaving or strictly alternating sequences of moves, as seen in conventional game semantics.*

We motivate the structure by revisiting the previous example.

$$
\begin{array}{lll}
\mathtt{t}_{t_2} = & f.\langle \square\, g \mid d\rangle^{\oplus} & \\
& g.\langle \square\, \mathsf{A} \mid c\rangle^{\ominus} & (fig.\ step1) \\
& c.\langle \kappa[e] \mid \square\rangle^{\oplus} & (fig.\ step2) \\
& d.\langle \kappa_c :: \kappa[e] \mid \square\rangle^{\ominus} & (fig.\ step3) \\
& \kappa_c.\langle \square[\mathbf{ret}\ 5] \mid d'\rangle^{\oplus} & (fig.\ step4) \\
& d'.\langle \mathbf{ret}\ \mathtt{tt} \mid \square\rangle^{\ominus}\ c_{\mathbf{f}}.\langle \mathbf{ret}\ \mathtt{tt} \mid \square\rangle^{\oplus} & (fig.\ step5)
\end{array}
$$

Ultimately, trace equivalence amounts to identifying handling structures modulo flattening.

Definition 5.9 (h-struct). We define the grammar of handling structures as follows:

$$
\begin{array}{rcll}
\mathbb{t} & ::= & \mathtt{t} & \text{ogs trace} \\
& \mid & \mathbb{t}\, [\![\mathbb{t}]\!]\, \mathbb{t} & \text{capturing} \\
& \mid & \mathbb{t}\, \langle \mathbb{t}, \ldots, \mathbb{t} \rangle & \text{branching}
\end{array}
$$

We will often write these structures vertically as trees (execution top to bottom).

* The first constructor $[\![\mathbb{t}]\!]$ is to be understood as a captured delimited trace, and that, if $[\![\mathtt{t}]\!]$ occurs in a structure like so $t_1 [\![t]\!] t_2 = s$
then s is a second-order trace in which $[\![t]\!]$ is a second-order move. Higher-order traces are to be understood similarly.

These structures represent extended trace forms. The $[\![\mathbb{t}]\!]$ construct models a capture of a delimited continuation corresponding to $\mathbb{t}$, and the $\langle \mathbb{t}_0, ..., \mathbb{t}_n \rangle$ form models branching execution where multiple interactions stem from a common prefix. We frequently render these handling structures vertically as trees, representing execution from top to bottom, and left to right for branching. The constructor $[\![\mathbb{t}]\!]$ is interpreted as a captured delimited trace. When this appears within a structure, say $\mathbb{t} = \mathtt{t}_0\, [\![\mathtt{t}]\!]\, \mathtt{t}_1$, we interpret the entire sequence $\mathbb{t}$ as a second-order trace, in which $[\![\mathtt{t}]\!]$ plays the role of a second-order move. This reasoning generalizes naturally to higher-order handling structures, where h-structs themselves become the content of moves at increasingly nested levels. The branching form $\mathbb{t}\, \langle \mathbb{t}_0, ..., \mathbb{t}_n \rangle$ captures the idea of an execution trace $\mathbb{t}$ followed by several "playouts" or invocations of the last captured structure, where each one is a trace or a complex handling structure in its own right. These branches represent observationally distinct, but semantically related, and ordered continuation invocations. We sometimes visualize this structure as a tree:

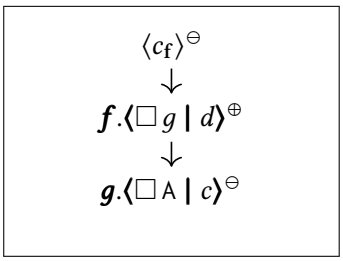


Fig. 12. subfigure step 1

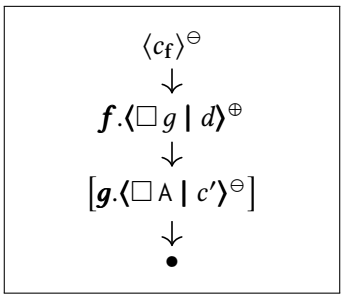


Fig. 13. subfigure effect forwarding 1

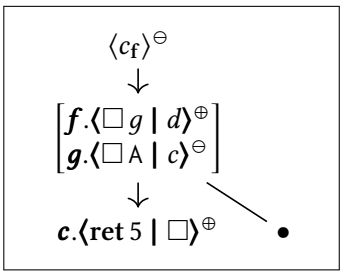


Fig. 14. subfigure effect forwarding 2

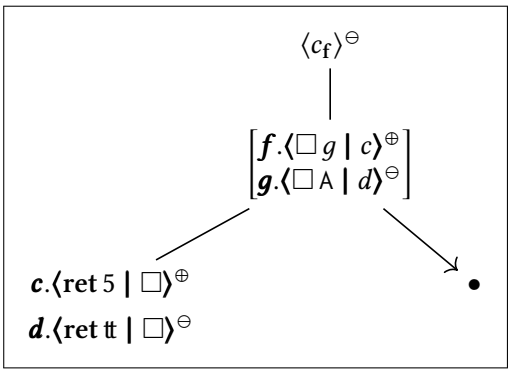


Fig. 15. subfigure invoking the continuation

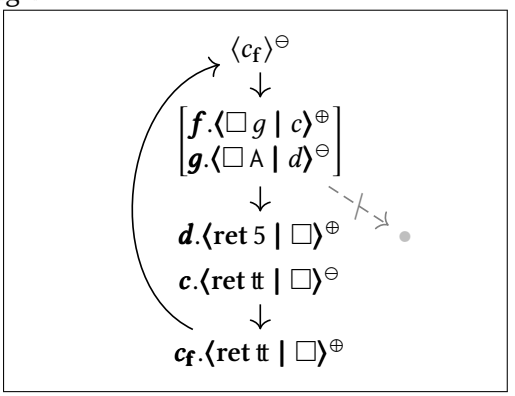
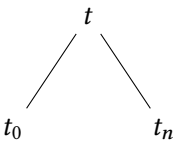


Fig. 16. subfigure exiting the handling scope

Fig. 17. Five subfigures arranged in 3-top/2-bottom layout

* A branching $t\langle t_0, \ldots, t_n \rangle$ will be sometimes represented as follows:



Each branch (e.g., t_0 through t_n) denotes an independent attempt or path of interaction that stems from the same prefix t.

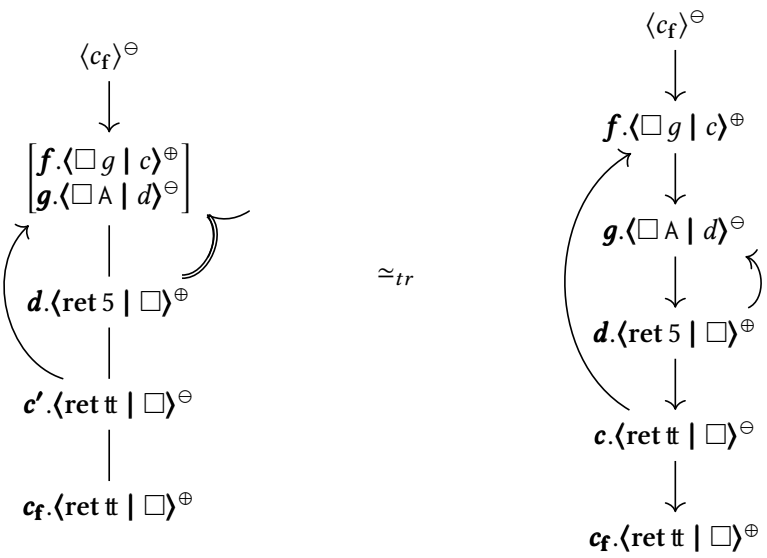
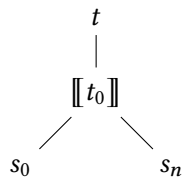


Fig. 18. trace equivalence, revisited.

Example 5.10. A basic example is of the shape:



where $t, t_0, s_0, \ldots, s_n$ are OGS sequences.

representing the interaction in which Opponent captured the delimited sequence t_0 via a handler and played it n times before resuming the continuation. This structure models an interaction in which the Opponent captures a delimited subtrace t_0 via a handler and then invokes it multiple times—resulting in trace sequences s_0 through s_n—before resuming the overall continuation.

concatenation. As we will see the branching is only an observationally branching structure from the point of view of the player that did not trigger it (observational independence), on the other hand, the player responsible for it (the player that handled the effect) these branches are sequential and ordered.

We chose to order them from left to right. Concatenation is therefore denoted $t_1 - t_2$ or simply $t_1 t_2$ is attaching t_2 to the left-most branch. It is defined recursively as follows:

It is important to note that branching, as represented in handler structures, is observational. That is, it appears as nondeterministic or parallel branching only from the perspective of the player not responsible for triggering the handler. For the handling player, the one who performed andhandled the effect, the branches are seen as ordered

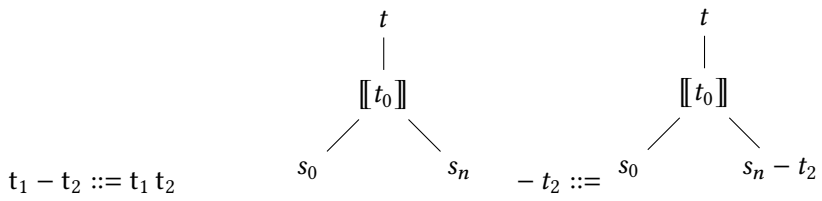


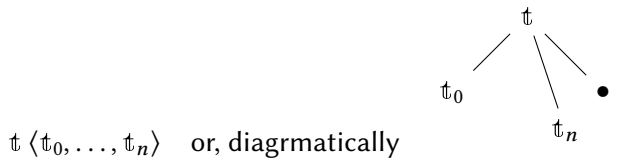
Fig. 19. concatenation of handling structures.

and sequential. This asymmetry reflects the inherent observational structure of algebraic effect handling.

Remark 5. *Notice that the concatenation operation proceeds within the handler structure—it traverses downward into its interior rather than simply grafting a new subtree at a leaf. This behavior is reminiscent of the zipper data structure, which enables navigation and insertion deep within hierarchical trees. A subtree of the form* $[\![t]\!]$ *does not necessarily correspond to an actual trace t that was captured at runtime via suspension or effect capture. Instead, it should be viewed as a static artifact—a structural component obtained purely through the rewriting process. G-note: The outcome of the rewriting process is a tree structure with a canonical form, up to permutation of its branches. This means that while the ordering of parallel branches may vary, the overall computational structure remains semantically invariant.*

Open handling structure. Once forward transitions and resumptions are eliminated from the trace, what remains is the handler structure—a tree capturing the essential control flow. This structure offers a refined view of effectful programs: rather than treating them as sequences, we model them as open handling structures—rooted trees in which leaves may be unbound branches or dangling prompts. These represent suspended computations or points of interaction that have yet to be resolved or resumed.

In order to represent computations/traces whereby the handling is still ongoing, we introduce an open branching structure



to indicate that the possibility of re-playing the captured delimited continuation by the handler is still there.

Just as it is standard in game semantics to refer to incomplete traces — interactions where some questions remain unanswered, open handling structures encode not only such interactional incompleteness, but also the partial handling of effects. Specifically, they signal that the handling of the last performed effect has not been fully resolved, and that the possibility of opening a new branch, *i.e.* of initiating a further resumption invokation, remains enabled.

Definition 5.11. We will define the opening operator $(\square)^\bullet$ as follows:

$$(\mathrm{t})^\bullet := \mathrm{t}\,\bullet \quad \text{i.e.} \quad \mathrm{t}_{\bullet} \tag{1}$$

$$(\mathtt{t}\,\langle \mathtt{t}_0, \ldots, \mathtt{t}_n \rangle)^\bullet := \mathtt{t}\langle \mathtt{t}_0, \ldots, \mathtt{t}_n, \bullet \rangle \tag{2}$$

$$(\mathtt{t}\,\mathbb{s})^\bullet := \mathtt{t}(\mathbb{s})^\bullet \tag{3}$$

Pointer structure/notation. We now turn our attention to the underlying pointer structure, which plays a critical role in characterizing what it means to close an open handler structure. This will be the final conceptual ingredient required to formally relate OGS traces to their corresponding handler structures.

Pointer annotations within the trace serve as semantic breadcrumbs: they allow us to reconstruct the nesting structure of effect handlers. Using these annotations, we can identify which portions of the computation have been captured—corresponding to subtrees—and how resumptions link back to the precise handler contexts in which effects were handled.

We will lift the pointer notation from moves, to traces then handling structures.

Definition 5.12 (pointer notation). The following definition is nothing but handy and less verbose notation that exposes the increasingly nested justification structure:

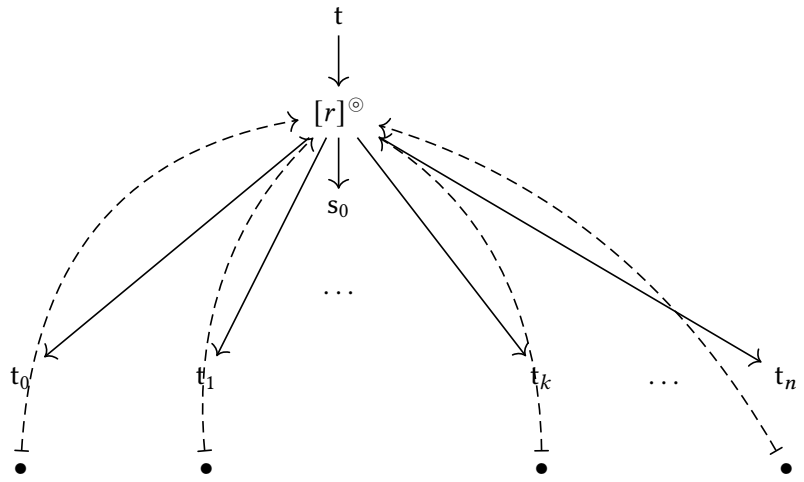
$$\frac{\mathbf{o} = \boldsymbol{a}.\mathrm{p} \qquad \mathbf{p} = \boldsymbol{b}.\mathrm{q} \qquad b \in \mathrm{supp}(\mathrm{p}) \setminus \mathrm{supp}(\mathrm{t}_0)}{\mathrm{t}_0\,\mathbf{o}\,\mathrm{t}_1\,\mathbf{p}\,\mathrm{t}_2}$$

$$\frac{\mathbf{m}_0 = \boldsymbol{a}.\mathrm{p} \qquad \mathbf{m}_1 = \boldsymbol{b}.\mathrm{q} \qquad b \in \mathrm{supp}(\mathrm{p}) \setminus \mathrm{supp}(\mathtt{t}_0) \qquad \mathtt{t}_1 = \mathbf{m}_0\,\mathbb{s}_1 \qquad \mathtt{t}_2 = \mathbb{s}_0\,\mathbf{m}_1}{\mathtt{t}_0\,\mathtt{t}_1\,\mathtt{t}_2\,\mathtt{t}_3}$$

$$\frac{\mathbf{m}_0 = \boldsymbol{a}.\mathrm{p} \qquad \mathbf{m}_1 = \boldsymbol{b}.\mathrm{q} \qquad b \in \mathrm{supp}(\mathrm{p}) \setminus \mathrm{supp}(\mathtt{t}_0) \qquad \mathtt{t}_0\,\mathtt{t}_1\,\mathtt{t}_2\,\mathtt{t}_3}{\mathtt{t}_0\,[\![\mathtt{t}_1]\!]\,\mathtt{t}_2\,\mathtt{t}_3}$$

$$\frac{\mathtt{t}\,[\![\mathbb{r}]\!]\,\mathtt{t}_0\,\mathbb{s}_i\,\mathtt{t}_{i+1}\,\mathtt{t}_{n+1}}{\mathtt{t}\,[\![\mathbb{r}]\!]\,\mathtt{t}_0\,\langle \mathbb{s}_0\,\mathtt{t}_1, \ldots, \mathbb{s}_i\,\mathtt{t}_{i+1}, \ldots, \mathbb{s}_n\,\mathtt{t}_{n+1} \rangle}$$

Example 5.13 (archetypical h-struct).



From traces to handling structures

To recover the handling structure representation of a flat trace, we introduce a contextual rewriting relation $\hookrightarrow_{\mathbf{fwd}}$ that systematically eliminates **fwd**-sequences and delimited continuation names from handling structures. In doing so, we expose the latent structure of effect handling.

h-redex. An *h-redex* is an h-struct of the form

$$\mathbb{r}\,\vec{\mathbf{fwd}}(r)\,\mathbb{t}_0\,\mathbb{s}_0\,\mathbb{t}_1\,\ldots\,\mathbb{s}_n\,\mathbb{t}_{n+1}$$

that satisfies the following conditions:

- $\forall i.\overset{\curvearrowleft}{\mathbb{t}_i} \quad \wedge\, \forall \kappa \in \mathrm{supp}(r),\ \kappa \in \mathrm{supp}(\mathbb{t}_i).$
- $r = \kappa \circ r' \circ \varkappa$ and $\forall i.\, \exists \mathrm{p}, \mathrm{q}, \mathbb{s}_i^0 \mathbb{s}_i^1.\quad \mathbb{s}_i = \boldsymbol{\kappa}.\mathrm{p}\ \mathbb{s}_i^0\ \boldsymbol{\varkappa}.\mathrm{q}\ \mathbb{s}_i^1$

We often use the notation $\mathbb{r}\,\vec{\mathbf{fwd}}(r)\,\mathbb{t}_0\,(\mathbb{s}_0; \mathbb{t}_1 \mid \cdots \mathbb{s}_n; \mathbb{t}_{n+1})$ for the redex $\mathbb{r}$ given by $\mathbb{r}\,\vec{\mathbf{fwd}}(r)\,\mathbb{t}_0\,(\mathbb{s}_0; \mathbb{t}_1 \mid \cdots \mid \mathbb{s}_n; \mathbb{t}_{n+1})$.

reducible contexts. Since the rewriting could occur in any arbitrary sub-structure, we will define rewriting contexts akin to lean contexts in λ-calculus. These are nothing but handling struct. with a single hole. Because rewriting may take place at arbitrary depths within a handler structure, we define rewriting contexts in the spirit of evaluation contexts from λ-calculus. These contexts are simply handler structures with a single hole, *i.e.* places where rewriting can be applied without loss of generality. What sets the apart from standard evaluation contexts, is that the rewriting rule is not defined as root relation on redexes which is then lifted to arbitrary h-structs via reduction contexts inside which the reduction occurrs and that remain unchanged. The h-struct *reducible context* may also be affected by the elimination of the **fwd**-transition they enclose.

Remark 6. *This is explained by the fact that the considered* h-redexes *capture a well-behaved use of continuations corresponding to* well-scoped *handlers described in [Xie et al.*

2020] . Since we are dealing with arbitrary handlers, the captured (delimited) continuation may escape its original scope by being "smuggled" inside a returned value, e.g. under a λ-abstraction. This non-scoped use occurrs outside the h-redex, hence the rewriting of contexts in order to eliminate delimited continuation names from the h-struct.

$$\mathcal{H} ::= [] \quad | \quad \mathtt{t}\,[\![\mathcal{H}]\!]\,\mathtt{t} \quad | \quad \mathtt{t}\,\langle \mathtt{t}_0, \ldots, \mathtt{t}_i, \mathcal{H}, \mathtt{t}_{i+1}, \ldots, \mathtt{t}_n \rangle \quad (n \in \mathbb{N})$$

Forward elimination rule. Reductions are labeled with a move substitution δ and are given by:

$$\text{ROOT}\ \frac{\mathtt{t} = \mathcal{B}r([\![\mathtt{r}]\!]\,\bullet, \mathtt{s}') \qquad \mathtt{s}' = \mathtt{s}\{\delta\}}{\mathtt{r}\,\vec{\mathbf{fwd}}(r)\,\mathtt{s} \overset{\delta}{\hookrightarrow}_{\mathbf{fwd}} [\![\mathtt{r}]\!]\,\mathtt{t}} \qquad \text{LIFT}\ \frac{\mathtt{t} \overset{\delta}{\hookrightarrow}_{\mathbf{fwd}} \mathtt{t}' \qquad \mathcal{H}' = \mathcal{H}\{\delta\}}{\mathcal{H}[\mathtt{t}] \overset{\delta}{\hookrightarrow}_{\mathbf{fwd}} \mathcal{H}'[\mathtt{t}']}$$

$$\text{LIFT}\ \frac{\mathtt{r} \overset{\delta}{\hookrightarrow}_{\mathbf{fwd}} \mathtt{r}' \qquad \mathcal{H}' = \mathcal{H}\{\delta\}}{\mathcal{H}[\mathtt{r}] \overset{\delta}{\hookrightarrow}_{\mathbf{fwd}} \mathcal{H}'[\mathtt{r}']}$$

$$\mathcal{B}r([\![\mathtt{r}]\!]\,(\mathtt{t})^{\bullet}, \mathbf{m}\,\mathtt{s}) := \begin{cases} \mathcal{B}r(\mathtt{r}\,(\mathtt{t})^{\bullet}\,\mathbf{m}, \mathtt{t}) & \text{if } \mathtt{r}\,(\mathtt{t})^{\bullet}\,\mathbf{m} \\ \mathcal{B}r(\mathtt{r}\,(\mathtt{t}[\mathbf{m}])^{\bullet}, \mathtt{t}) & \text{if } \mathtt{r}\,(\mathtt{t})^{\bullet}\,\mathbf{m} \\ \mathcal{B}r(\mathtt{r}\,(\mathtt{t}[\mathbf{m}])^{\bullet}, \mathtt{t}) & \text{if } \mathtt{r}\,(\mathtt{t})^{\bullet}\,\mathbf{m} \end{cases}$$

Confluence and normalization: This rewriting system enjoys both confluence and strong normalization: no matter the order in which rewrite steps are applied, the process always terminates in a unique normal form. This normal form is canonical—up to permutations of parallel branches—capturing the essential computational structure while abstracting from syntactic artifacts. This establishes the key result: every handler structure admits a canonical representative that can be systematically reconstructed from traces through the rewriting process. This form serves as a normal representative in the space of equivalent traces.

Definition 5.14. Given an OGS trace t, we will write $\mathtt{t}^{\dagger}$ for the **fwd**-free handling structure such that:

$$\mathtt{t} \hookrightarrow^{*}_{\mathbf{fwd}} \mathtt{t}^{\dagger} \not\hookrightarrow_{\mathbf{fwd}}$$

Remark 7. *Unless stated otherwise, whenever we say handling structure, we assume that it is an OGS one, i.e., it can be obtained from an OGS trace via the rewriting rule* $\hookrightarrow_{\mathbf{fwd}}$.

Player's behaviour, semantically: Fibers and Interaction Trees

As we will see in later sections, branches arising from a player-controlled handlers are considered independent from the perspective of its opponent. Each such branch reflects a distinct point of disclosure or interaction initiated by this player. Accordingly, it is

natural to view a handling structure as comprising the aggregate behavior of opponent — assembled from the separate behaviors associated with each independent branch. The partial semantics of Opponent's behaviour (exposed by the/its interaction witnessed by $\mathbb{t}$) are thus captured through these constituent threads of interaction.

In scenarios where *actual* effects are absent, we refer to these branching substructures as fibers—in analogy to lightweight threads or "green threads" in concurrent programming. Each fiber represents a potential line of interaction initiated by the Opponent.

We first present a simple illustrative example to build intuition, before formalizing the notion of fibers and their composition.

Fibers

We define a function $\mathcal{F}ib : \mathbf{Tr}_{\mathrm{ogs}} \to \wp(\mathbf{Tr}_{\mathrm{ogs}})$ which maps each handling structure to the set of its constituing fibers—that is, its independent substructures corresponding to all the possible interaction it "subsumes". $\mathcal{F}ib(\mathbb{t})$ then denotes the set of traces embedded in the handler structure $\mathbb{t}$. These are obtained by linearizing each branch or subtree, and collectively represent all possible sequential executions contained within $\mathbb{t}$.

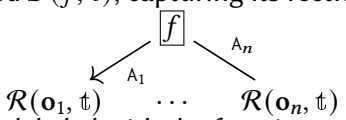
Definition 5.15 (O-fibers). O-fibers associated to an h-struct is given inductively by:

$$\mathcal{F}ib(\mathcal{H}[[\![\mathbb{r}]\!]\ \mathbb{t}_0\ (\mathbb{s}_0; \mathbb{t}_1 \parallel \cdots \parallel \mathbb{s}_{n+1})]) := \bigcup \{\mathcal{H}[\mathcal{F}ib(\mathbb{t}_0)[\![\mathbb{r}]\!]\mathbb{s}_i] \mid 0 \leq i \leq n\}$$

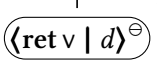
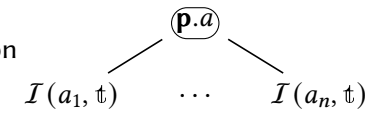
Interaction trees

Interaction trees are defined from handling structures and that *partially* capture the behaviour of a player — that is the behaviour of all code(s) it disclosed to it opponent.

A function f disclosed as an abstract name f, its behaviour in the abstract will be described by the the *introduction tree*, denoted $\mathcal{I}(f, \mathbb{t})$, capturing its rection to different inputs from the interaction represented by $\mathbb{t}$:



where the edges coming out of the node are labeled with the function arguments A_i *s.t.* $\boldsymbol{f}.\langle\square\,\mathsf{A}_i \mid \_\rangle \in \mathbb{t}$ and $\mathcal{R}(\mathbf{o}_i, \mathbb{t})$ corresponds to the tree representing the reaction of f to this question.

$\mathcal{R}(\mathbf{o}_i, \mathbb{t})$ is either a leaf of the shape  where v is a positive value or or a reaction that introduces new abstract names to the interaction 

where, for $i \in \{1, \ldots, n\}$, $a_i \in \mathrm{supp}(\mathrm{p})$

In the presence of *actual* state effects, such a tree with unordered branches becomes insufficient, since a a name a may respond differently to the same input, this violating functionality.

In our model, while observational innocence—the inability to distinguish between certain interactions externally, can be broken, *virtual* innocence is preserved. This means that the

internal structure remains deterministic and principled, even when observabl (external) behavior diverges.

We give a formal definition in Figure 20.

Reaction trees	$\mathcal{R}(\mathbf{p}.a, \mathbb{h})$	:=	$\{\mathcal{I}(b, \mathbb{h}) \mid b \in \mathrm{supp}(\mathrm{p})\}$
Introduction trees	$\mathcal{I}(f, \mathbb{h})$	:=	$\{\langle \mathsf{A}, \mathcal{R}(\mathbf{o}, \mathbb{h})\rangle \mid \boldsymbol{f}.\ \langle \square\, \mathsf{A} \mid c\rangle^{\oplus}\ \mathbf{o} \sqsubset \mathbb{h}\}$
	$\mathcal{I}(c, \mathbb{h})$	:=	$\{\langle \mathsf{A}, \mathcal{R}(\mathbf{o}, \mathbb{h})\rangle \mid \langle \mathbf{ret}\, c \mid \mathsf{A}\rangle^{\ominus}\ \mathbf{q}.b^{\ominus} \sqsubset \mathbb{h}\}$ $\bigcup\{\langle \eta, \mathcal{F}(\eta, \mathbb{t})\rangle \mid \ldots\}$
Handling Forests	$\mathcal{F}(\eta, \mathbb{h})$	:=	$\langle \mathcal{R}(\mathbf{o}_0, \mathbb{h}), \boldsymbol{\kappa}.\ \langle \mathbf{ret}\, n_0 \mid [\square]\rangle, \ldots \boldsymbol{\kappa}.\ \langle \mathbf{ret}\, n_k \mid [\square]\rangle, \mathcal{R}(\mathbf{o}_{k+1}, \mathbb{h})\rangle$
where $[\![\mathbb{h}]\!]^{\eta}\mathbb{t}_0(\boldsymbol{\kappa}.\ \langle \mathbf{ret}\, n_0 \mid [\square]\rangle\ \mathbb{s}_0; \mathbb{t}_1 \parallel \cdots \parallel \boldsymbol{\kappa}.\ \langle \mathbf{ret}\, n_0 \mid [\square]\rangle\ \mathbb{s}_k; \mathbb{t}_{k+1}) \sqsubseteq \mathbb{h}$			

(a) reaction trees

Fig. 20. Definition of Interaction Trees

Definition 5.16 (OGS tree). Given a definable OGS h-struct $\mathbb{h}$, we define the corresponding definability tree as follows:

$$\mathcal{T}[\mathbb{h}] := \{\mathcal{I}(a, \mathbb{h}) \mid \mathbf{p}.b^{\ominus} \sqsubset \mathbb{h} \wedge a \in \mathrm{supp}(\mathrm{p})\}$$

We will use the powerset notation $\wp(\mathcal{T})$ to refer to the set of (reaction and introduction) sub-trees of $\mathcal{T}$. Given a definability tree $\mathcal{S} = \mathcal{T}[\mathbb{t}]$, we will denote by $\mathcal{S}_{/a}$ and $\mathcal{S}_{/\mathbf{o}}$ the sub-trees $\mathcal{I}(a, \mathbb{t})$, $\mathcal{R}(\mathbf{o}, \mathbb{t}) \in \wp(\mathcal{S})$, respectively.

Lemma 5.17 (Innocent trees). *Given an* O*-innocent definable handling structure* $\mathbb{h}$*, then for any introduction sub-tree* $\mathcal{I} \in \wp(\mathcal{T}[\mathbb{h}])$*:*

Definition 5.18 (discordant trees). Two interaction trees $\mathcal{T}, \mathcal{S}$ are said to be *discordant* when there exists a name a, an absract value A and two introduction sub-trees $\mathcal{T}_{/a}$ and $\mathcal{S}_{/a}$ such that:

$$\langle \mathsf{A}, \mathcal{R}\rangle \in \mathcal{T}_{/a} \wedge \langle \mathsf{A}, \mathcal{R}'\rangle \in \mathcal{S}_{/a} \wedge \mathcal{R} \neq \mathcal{R}'$$

Concordant trees have a lattice-like structure, we exhibit this through the following partial order and the following operation on trees that constructs an upper bound.

Definition 5.19 (dependent-approximation). where $\preceq_{\mathcal{T}}$ is defined inductively on introduction and reaction trees:

$$\begin{array}{lll} \mathcal{I}_1 \preceq_{\mathcal{T}} \mathcal{I}_2 & :\Longleftrightarrow & \exists \mathcal{I}_1' \subseteq \mathcal{I}_2,\ \mathcal{I}_1 = \{\langle \mathsf{A}, \mathcal{R}\rangle \mid \langle \mathsf{A}, \mathcal{R}'\rangle \wedge \mathcal{R} \preceq_{\mathcal{T}} \mathcal{R}'\} \\ \mathcal{R}_1 \preceq_{\mathcal{T}} \mathcal{R}_2 & :\Longleftrightarrow & \mathcal{R}_1 = \{\langle a, \mathcal{I}\rangle \mid \exists\, \langle a, \mathcal{I}'\rangle \in \mathcal{R}_2.\ \mathcal{I} \preceq_{\mathcal{T}} \mathcal{I}'\} \end{array}$$

Definition 5.20 (tree grafting). We define the tree grafting operator on concordant trees $\circledast$ as follows:

$$\mathcal{T}_1 \circledast \mathcal{T}_2 := \begin{cases} \mathcal{T} := \mathcal{T}_{/a} \circledast \mathcal{S} & \text{if } \mathcal{S} = \mathcal{R}(a, \_) \\ \mathcal{T}_1 \cup \mathcal{T}_2 & \text{if } \exists a.\ \mathcal{T}_i = \mathcal{T}_{/a} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Lemma 5.21. *Given an intrinsically visible and innocent h-struct* $\mathbb{t}$ *we have:*

(1) $\forall \mathbb{s}_1, \mathbb{s}_2 \in \mathcal{F}ib(\mathbb{t}), \quad \mathbb{s}_1$ *and* $\mathbb{s}_2$ *are concordant*

(2) $\mathcal{T}[\![\mathbb{t}]\!] = \underset{\mathbb{s}\in\mathcal{F}ib(\mathbb{t})}{\circledast} \mathcal{T}[\![\mathbb{s}]\!]$

This property states the behaviour of a player interacting with another player that uses effect handlers can be seen as a sum of behaviours against independent yet concordant players (each represented by an uniplexed fiber).

This paves the way to the next property on traces / A-statutes of $\sqsubseteq$-saturation which entails that in the absence of actual effects, it is not possible to single out a single execution truth.

Definition 5.22 (Environment dependent approximation). Given two OGS traces $t_1, t_2 \in \mathbf{Tr}_{\mathrm{ogs}}$, we say that t_1 approximates t_2 and we write:

$$t_1 <_o t_2 :\Longleftrightarrow \mathcal{T}[\![t_1]\!] \preceq_{\mathcal{T}} \mathcal{T}[\![t_2]\!]$$

Lemma 5.23 (O-saturation). *Given an OGS configuration* $\mathbb{G}$ *and a trace* $t \in \mathbf{Tr}(\mathbb{G})$*, we have:*

(1) $s <_o t$ *implies* $s \in \mathbf{Tr}(\mathbb{G})$
(2) $s \simeq_o s$ *implies* $s \in \mathbf{Tr}(\mathbb{G})$

Lemma 5.24 (Double equivalence). *Let* t, t' *and* s *be traces satisfying* $t \simeq_o s \simeq_p t'$*. There exists a trace* s' *such that* $t \simeq_p s' \simeq_o t'$*.*

6 The OGS model

The OGS LTS $(\mathcal{O}, \mathcal{M}, \rightarrow_{\mathrm{ogs}})$ is the transition system that can generate the interaction traces[2] compatible with what our language $(\Lambda_{\mathrm{eff}}, \mapsto_{\mathrm{op}})$ permits.
It is defined as the product of $\mathcal{L}_{\mathrm{AI}}$, $\mathcal{L}_{\mathcal{T}}$ and $\mathcal{L}_{wb}$ where the underlying labelled transitions systems are synchronised on $\mathcal{M} \cup \{hdl^{\oplus}, hdl^{\ominus}\}$. The configurations in $\mathcal{O}$ satisfy some conditions that guarantee the well-typedness of $\mathcal{L}_{\mathrm{AI}}$; in that *Opponent* and *Proponent* agree on types during their interaction, in addition to ensuring that *Opponent*'s use of continuations is well-bracketed.

6.1 The OGS LTS

In the following we will denote $\langle \mathbb{I} \parallel \mathbb{T} \parallel \mathbb{W} \rangle \in \mathcal{A} \times \mathcal{T} \times \mathcal{W}$ any triple of configurations satisfying $\mathbb{I} \wr \mathbb{T}$ and $\mathbf{Compat}(\mathbb{I}, \mathbb{W})$, where the second proposition morally means that the control-flow history in $\mathbb{W}$ is *compatible* with the interactive information in $\mathbb{I}$. The formal defintion of this predicate will be later introduced in section H.2.

6.2 Typing Constraints

Definition 6.1. Type configurations $\mathbb{T}; \mathbb{S}$ are of the shape $\langle \Gamma_o; \Delta_o \mid \Gamma_p; \Delta_p \rangle$ where:

- Γ_p, Γ_o are two *disjoint* type contexts for *Program names* and *Environment names*, respectively.

[2]For a reader familiar with *game semantics*, this LTS specifies the *plays*.

$$\textsc{passive conf.}\quad \frac{\mathrm{dom}(\gamma) = \Gamma_p \qquad \Gamma_o \vdash \gamma : \Gamma_p \qquad \Gamma_p \vdash \xi : \Delta_o}{\langle \mathtt{I};\, \gamma;\, \xi\rangle \Subset \langle \Gamma_o;\, \Delta_o \mid \Gamma_p;\, \Delta_p\rangle}$$

$$\textsc{active conf.}\quad \frac{\mathrm{dom}(\gamma) = \Gamma_p \qquad \mathtt{I}; \Gamma_o \cdot \Delta_o \vdash_{\mathtt{c}} \mathtt{M} \qquad \Gamma_o \vdash \delta : \Delta_p}{\langle \mathtt{M};\, \gamma;\, \delta\rangle \Subset \langle \Gamma_o;\, \Delta_o \mid \Gamma_p;\, \Delta_p\rangle}$$

Fig. 21. configuration typing.

- Δ_p, Δ_o are two *disjoint* type contexts for effects and forwarded *abstract delimited continuations*.
- For $y \in \{o, p\}$. Γ_y and Δ_y are disjoint.

The typing judgements $\mathbb{E} \Subset \mathbb{T}$ and $\mathbb{P} \Subset \mathbb{S}$ denotes that the configurations $\mathbb{E}$ and $\mathbb{P}$ are of type $\mathbb{T}$ and $\mathbb{S}$ respectively. We define the corresponding typing relation fig. 21.

We will denote by $\mathbb{T}^{\perp}$ the type configuration dual to $\mathbb{T}$ defined by:

$$\langle \Gamma_o; \Delta_o \mid \Gamma_p; \Delta_p\rangle^{\perp} := \langle \Gamma_p; \Delta_p \mid \Gamma_o; \Delta_o\rangle$$

The *Type LTS* $\mathcal{L}_{\mathcal{T}}$ is defined as $(\mathcal{T}; \mathcal{M}; \xrightarrow{\mathbf{m}}_{\mathcal{T}})$ with $\xrightarrow{\mathbf{m}}_{\mathcal{T}}$ defined in figure 22, where $\mathcal{T}$ is the set of type configurations.

$$\textsc{PAction}\quad \frac{\Gamma_o(a) = \tau \qquad \Gamma_o \cdot \Delta_p \cdot \Delta_o \Vdash \mathsf{q} : \neg\tau \triangleright \Gamma_{\mathsf{q}}; \Delta_{\mathsf{q}}}{\langle \Gamma_o;\, \Delta_o \mid \Gamma_p;\, \Delta_p\rangle \xrightarrow{\boldsymbol{a}.\mathsf{q}^{\oplus}}_{\mathcal{T}} \langle \Gamma_o;\, \Delta_o \mid \Gamma_p \cdot \Gamma_{\mathsf{q}};\, \Delta_p \cdot \Delta_{\mathsf{q}}\rangle}$$

$$\textsc{PHandle}\quad \frac{}{\langle \Gamma_o;\, \Delta_o \mid \Gamma_p;\, \Delta_p\rangle \xrightarrow{hdl^{\oplus}}_{\mathcal{T}} \langle \Gamma_o \cdot \Delta_o;\, \emptyset \mid \Gamma_p;\, \emptyset\rangle}$$

(a) proponent transitions.

$$\frac{\mathbb{T} \xrightarrow{\mathbf{m}}_{\mathcal{T}} \mathbb{S}}{\mathbb{T}^{\perp} \xrightarrow{\mathbf{m}^{\perp}}_{\mathcal{T}} \mathbb{S}^{\perp}}$$

(b) symmetrical transitions.

Fig. 22. Definition of $\mathcal{L}_{\mathcal{T}}$

6.3 Well-bracketing Constraints

The interactive configurations $\mathbb{I} \in \mathcal{A}$ we have considered thus far represent *Proponent*'s one-sided perspective on the computation. The *information component* contains all of its *codata* that had been disclosed up to the current stage of the computation, namely the *continuations*. However, this component (and by extant any *passive configuration*), as it is defined, is history-agnostic whereas the language Λ_{eff} only permits a constrained usage of these *abstract continuations* that respects a certain history-sensitive *bracketing* discipline.

In this section, we define the well-bracketing LTS that addresses this issue by constraining the moves of *Opponent* in $\mathcal{L}_{\mathsf{AI}}$, so that the generated *well-bracketed* behaviours are correspond to actual concrete interactions.

6.3.1 The control-flow of delimited continuations. When an effect is performed by one *Player* and gets propagated beyond its immediate controlled scope, it initiates a sequence of effectful moves until the effect is handled or reaches *top-level*.

To visualise this part of the interaction, we will zoom in at the computation underlying it in the concrete example of **??**.

In figure.**??**, we only highlight the player responsible for triggering the effect (in its active and passive states). The resulting effectful sequence is $\boldsymbol{d_1}.\langle \kappa[e] \mid \Box\rangle^{\oplus}\boldsymbol{c_0}.\langle \varkappa :: \kappa[e] \mid \Box\rangle^{\ominus}$, which can be understood as a *pseudo-copycat* sequence in which *Opponent* forwards the computation $\langle e \mid d_1\ []\ \circ \kappa\ []\ \rangle$ as is to the abstract continuation c_0 $[]$.

Now if we wish to generalize this sequence to one of arbitrary length n, $\mathbb{I} \overset{s}{\Rightarrow}_{\mathbf{fwd}} \mathbb{J}$, the generated configuration will always be of the following shape:

$$\frac{\mathbb{I} \overset{s}{\Rightarrow}_{\mathbf{fwd}} \mathbb{J} \qquad \mathbb{J} \xrightarrow{\mathbf{hdl}}_{\mathbf{fwd}}}{\mathbb{J} = \langle\langle \mathbf{op}\ \mathsf{v} \mid \mathsf{T}[\{[]\}\ \mathbf{with}\ \mathsf{h}]\ \circ\ \mathsf{S}_{n+1}[\kappa_n[\mathsf{S}_n[\cdots \kappa_0[\mathsf{S}_0]\cdots]]]\rangle\rangle}$$

Consequently, the captured continuation will be of the shape:

$$\lambda x.\{\mathsf{S}_{n+1}[\kappa_n[\mathsf{S}_n[\cdots \kappa_0[\mathsf{S}_0[\mathbf{ret}\ x]]]\cdots]]\}\ \mathbf{with}\ \mathsf{h}$$

In the case the continuation is discarded and not used, the control-flow stack remains intact. Otherwise, every call to the captured delimited continuation entails exactly one query addressed to each one of the n environment controlled fragments; *i.e.* a call to each of the abstract stack frames κ_i in the order in which they have been disclosed.

6.3.2 Well-bracketing LTS. We define the well-bracketing LTS as $\mathcal{L}_{wb} := (\mathcal{W}, \mathcal{M}, \rightarrow_{wb})$ where the configurations are described below and the transitions $\overset{\mathbf{m}}{\longrightarrow}_{wb}$ are defined in figure 23.

effect forwarding info.	$\mathcal{F} \times \mathcal{E}$	$\ni$	η	$::=$	$\varnothing$	*(absence of effect)*
				$\mid$	$(f,\ e)$	*(effect e forwarded through f)*
configurations	$\mathcal{W}$	$\ni$	$\mathbb{W}, \mathbb{U}$	$:=$	$\langle \sigma \mid \eta \mid \phi \rangle$	

where $\sigma \in \mathtt{List}(\mathcal{C} \cup \mathcal{K})$, $f \in \mathcal{F} = \mathtt{List}(\mathcal{K})$ denote a *control-flow stack* and a *call frame*, respectively, and $\phi \in \wp(\mathcal{F})$ denotes the set of all captured call frames.

6.3.3 Embedding the interactive language. Finally, we define an interactive term embedding into an initial *active* $\mathcal{L}_{\mathrm{OGS}}$ configuration and ciu-environment embedding into an initial *passive* $\mathcal{L}_{\mathrm{OGS}}$ configuration.

Definition 6.2. **(term embedding)** Given an interactive term $\mathtt{I}; \Gamma \vdash_{\mathtt{c}} \mathtt{M}$, we define its embedding

$$\langle \mathtt{I}; \Gamma \vdash_{\mathtt{c}} \mathtt{M} \rangle^{\oplus}_{\mathrm{ogs}} := \langle \langle \mathtt{M};\ \varepsilon \rangle \parallel \langle \Gamma; \emptyset \mid c_{\mathtt{f}} : \neg\mathbb{1}; \emptyset \rangle \parallel \langle [d];\ \varnothing;\ \emptyset \rangle \rangle$$

with d the unique continuation name of Γ.

Definition 6.3. **(environment embedding)** Given a name assignment $\mathtt{I}; c_{\mathtt{f}} : \neg\mathbb{1} \vdash \gamma : \Gamma$, we define its embedding

$$\langle \mathtt{I}; c_{\mathtt{f}} : \neg\mathbb{1} \vdash \gamma : \Gamma \rangle^{\ominus}_{\mathrm{ogs}} := \langle \langle \emptyset; \gamma \rangle \parallel \langle c_{\mathtt{f}} : \neg\mathbb{1}; \emptyset \mid \Gamma; \emptyset \rangle \parallel \langle [c_{\mathtt{f}}];\ \varnothing;\ \emptyset \rangle \rangle$$

$$\begin{array}{lcl}
\mathbf{forward}(\varnothing,\ \boldsymbol{c}.\ \langle e \mid \square \circ \varkappa\rangle^{\ominus}) & := & (e,\ []) \\
\mathbf{forward}(\varnothing,\ \boldsymbol{\kappa}.\ \langle e \mid d[\square] \circ \varkappa\rangle^{\ominus}) & := & (e,\ []) \\
\mathbf{forward}(\eta,\ \mathbf{o}) & := & \eta
\end{array}$$

$$\begin{array}{lcl}
\mathbf{call/ret}(a :: \sigma,\ _,\ \boldsymbol{a}.\mathrm{p}^{\ominus}) & := & \sigma \\
\mathbf{call/ret}(\sigma,\ \phi,\ \boldsymbol{\kappa}.\ \langle e \mid d[\square] \circ r\rangle^{\ominus}) & := & f +\!\!+ \sigma \\
\mathbf{call/ret}(\sigma,\ \phi,\ \boldsymbol{\kappa}.\ \langle \mathbf{ret}\ \mathsf{A} \mid d[\square]\rangle^{\ominus}) & := & f +\!\!+ \sigma \\
\quad \text{when} \quad \kappa :: f \in \phi & &
\end{array}$$

$$\frac{\eta' = \mathbf{forward}(\eta,\ \mathbf{o}) \qquad \sigma' = \mathbf{call/ret}(\sigma,\ \phi,\ \mathbf{o})}{\langle \sigma \mid \eta \mid \phi\rangle \xrightarrow{\mathbf{o}}_{wb} \langle \sigma' \mid \eta' \mid \phi\rangle}$$

$$\frac{}{\langle \sigma \mid (e, f) \mid \phi\rangle \xrightarrow{hdl^{\ominus}}_{wb} \langle \sigma \mid \varnothing \mid \phi \cup \{f\}\rangle}$$

(a) opponent transitions.

$$\begin{array}{lcl}
\mathbf{forward}((e, f),\ \boldsymbol{d}.\ \langle e \mid \square \circ \kappa :: r\rangle^{\oplus}) & : \\
\mathbf{forward}((e, f),\ \boldsymbol{\varkappa}.\ \langle e \mid c[\square] \circ \kappa :: r\rangle^{\oplus}) & : \\
\mathbf{forward}(\varnothing,\ \mathrm{p}) & :
\end{array}$$

$$\begin{array}{lcl}
\mathbf{call/ret}(\sigma,\ _,\ \boldsymbol{\varkappa}.\ \langle \mathbf{ret}\ \mathsf{A} \mid c[\square]\rangle^{\oplus}) & := & c \\
\mathbf{call/ret}(\sigma,\ _,\ \boldsymbol{\varkappa}.\ \langle e \mid c[\square] \circ _\rangle^{\oplus}) & := & c \\
\mathbf{call/ret}(\sigma,\ _,\ \boldsymbol{f}.\ \langle \square\ \mathsf{A} \mid c\rangle^{\oplus}) & := & c \\
\mathbf{call/ret}(\sigma,\ _,\ \mathrm{p}) & := & \sigma
\end{array}$$

$$\frac{\sigma' = \mathbf{call/ret}(\sigma,\ \mathrm{p}) \qquad \eta' = \mathbf{forward}(\eta,\ \mathrm{p}}{\langle \sigma \mid \eta \mid \phi\rangle \xrightarrow{\mathrm{p}}_{wb} \langle \sigma' \mid \eta' \mid \phi\rangle}$$

$$\frac{}{\langle \sigma \mid \varnothing \mid \phi\rangle \xrightarrow{hdl^{\oplus}}_{wb} \langle \sigma \mid \varnothing \mid \phi\rangle}$$

(b) pro

Fig. 23. $\mathcal{L}_{wb}$ transitions.

6.4 Interpretation of expressions

Following our equivalence of programs introduced in definition 3.3, we only observe covergence, *i.e.* a full evaluation into a value returner of the shape **ret** v. We introduce a notion of *complete trace* in order to capture this *complete* evaluation.

Definition 6.4. **(complete traces):** A trace $\mathrm{t}\,\mathbf{m} \in \mathbf{Tr}_{\mathrm{ogs}}$ is said t be *complete* and written $\mathcal{CP}(\mathrm{t}\,\mathbf{m})$ when:

$$\mathcal{CP}(\mathrm{t}\,\mathbf{m}) := \begin{cases} \langle [] \mid \varnothing \mid \emptyset\rangle \overset{\mathrm{t}\,\mathbf{m}}{\Longrightarrow}_{wb} \langle [] \mid \eta \mid \phi\rangle \ \text{ for some } \eta \text{ and } \phi \\ \mathbf{m} = \boldsymbol{c}.\langle \mathbf{ret}\ \mathsf{A} \mid \square\rangle^{\ominus} \ \text{ for some } \mathsf{A} \text{ and } c. \end{cases}$$

Accordingly, we define the set of complete *traces* of a configuration:

$$\mathbf{CTr}_{\mathrm{ogs}}(\mathbb{G}) := \{\mathrm{t} \in \mathbf{Tr}_{\mathrm{ogs}}(\mathbb{G}) \mid \mathcal{CP}(\mathrm{t})\}$$

6.5 Visibility and Innocence

Intuitively, Opponent's behaviour cannot be constrained to be innocent nor visible since he has the full capability of encoding a high-order state.

However, as explained in the introduction, the fact that we are dealing with *virtual* effects entails that progams are *virtually pure* suggesting that there might exist corresponding notions of *virtual visibility* and *virtual innocence*. As a matter of fact, the intensionality of traces with respect to effects makes it so that breaking *actual innocence* or *actual visibility* will always be done using *unobservable* private effects, the propagation of which appears on the traces.

In the following, unless specified we use the term innocence and visibility to refer to the virtual notions. The *actual* innocence (resp. visibility) will be termed *strict* or *observational* innocence (resp. visibility).

In the previous section, we have shown how distinct traces can denote the same underlying observable behaviour. Thus a third variation on these constraints up to trace equivalence naturally arises, which we term *lax-visibility* and *lax-innocence*. In this section, we will justify not constraining Opponent's behaviour by showing that all traces breaking *observational* O-*innocence* (resp. O-*visibility*) are actually O-lax-innocent (resp. O-lax-visible) up-to.

The View, naively

The view in game semantics is the semantic counterpart to the syntactic scope. Typically, in the absence of a state effect, a Player's action can only be justified by moves falling within this view. This condition is typically referred to as *visibility*.

The view of a given strategy is then defined as ... following the pointer structure, thereby associating to each step of the interaction the accessible/visible part of the history of interaction. However, this restriction is lifted when dealing with a stateful environment, which can, in principle, store and keep updating the history of interaction – thus maintaining access to it.

A state effect can be achieved with algebraic effects and handlers, it is justified to ask whether is required. It is worth defining the view on traces naively and observe how far we can go.

Definition 6.5 (pointer structure).

$$\frac{\mathbf{o} = \boldsymbol{a}.\mathrm{p} \qquad \mathbf{p} = \boldsymbol{b}.\mathrm{q} \qquad b \in \mathrm{supp}(\mathrm{p}) \setminus \mathrm{supp}(\mathrm{t}_0)}{\mathrm{t}_0\, \mathbf{o}\, \mathrm{t}_1\, \mathbf{p}\, \mathrm{t}_2}$$

Opponent's *strict* view is given at any stage of the interaction by the recursive function $\ulcorner\urcorner_{\ominus} : \mathbf{Tr}_{\mathrm{ogs}} \rightharpoonup \mathbf{Tr}_{\mathrm{ogs}}$ defined on P-ending traces.

Definition 6.6 (Opponent's view). The O-view associated to a P-ending trace t, denoted $\ulcorner \mathrm{t} \urcorner_{\ominus}$ is defined as:

$$\ulcorner \mathrm{t}\, \mathbf{o}\, \mathrm{s}\, \mathbf{p} \urcorner_{\ominus} := \ulcorner \mathrm{t} \urcorner_{\ominus}\, \mathbf{o}\, \mathbf{p}$$

We will resort to the handling structures to illustrate this definition of the view.

Visibility is a predicate on traces that guarantees that the usage by a *player* of names introduced by its *opponent* is coherent with its current *view* of past interaction.

Definition 6.7 (visibility). Given a trace $\mathrm{t} \in \mathbf{Tr}_{\mathrm{ogs}}$, we say that t is O-visible when:

$$\forall \mathrm{s} \in (\mathrm{X}, \mathrm{P})\text{-}\mathbf{Tr}_{\mathrm{ogs}}. \forall \mathbf{o}.\, \mathrm{s}\, \mathbf{o} \sqsubseteq \mathrm{t}.\ \ \mathrm{supp}(\mathbf{o}) \cap \mathrm{supp}(\mathrm{s}) \subseteq \mathrm{supp}(\ulcorner \mathrm{s} \urcorner_{\ominus}).$$

Dually, a trace $\mathrm{t} \in \mathbf{Tr}_{\mathrm{ogs}}$ is P-visible *if and only if* $\mathrm{t}^{\perp}$ is O-visible.

Definition 6.8 (O-innocence). Given a trace $t \in \mathbf{Tr}_{ogs}$, we say that t is O-innocent when Opponent's moves are invariant w.r.t. their corresponding views; that is:

$$\forall s_1\ \mathbf{o}_1, s_2\ \mathbf{o}_2 \in (X, O)\text{-}\mathbf{Tr}_{ogs}.s_1\ \mathbf{o}_1, s_2\ \mathbf{o}_2 \sqsubseteq t.$$
$$\forall \pi \in \mathrm{Perm}(\mathcal{N}).\ \pi \cdot \ulcorner s_1 \urcorner_{\ominus} = \ulcorner s_2 \urcorner_{\ominus} \implies \pi \cdot \mathbf{o}_1 \equiv \mathbf{o}_2$$

Consider the term:

$$t := d[\lambda x.(x\ \mathsf{v};\ x\ \mathsf{v};\ \mathbf{ret}\ \langle\rangle)] \tag{4}$$

A possible interaction is witnessed by the following trace, we denote t_1:

$$\overbrace{\underbrace{\boldsymbol{d_f}.\langle \mathbf{ret}\ f \mid \square\rangle^{\oplus}\ \boldsymbol{f}.\langle \square\ g \mid d\rangle^{\ominus}\ \boldsymbol{g}.\langle \square\ \mathsf{A} \mid c_1\rangle^{\oplus}\ \mathbf{tt}.\langle \mathbf{ret}\ c_1 \mid \square\rangle^{\ominus}\ \boldsymbol{g}.\langle \square\ \mathsf{A} \mid c_2\rangle^{\oplus}}_{s_2}\ \boldsymbol{c_2}.\langle \mathbf{ret}\ \mathbf{ff} \mid \square\rangle^{\ominus}\ \boldsymbol{d_f}.\langle \mathbf{ret}\ \langle\rangle \mid \square\rangle}^{s_1}$$

This trace is not innocent, because $s_1 \equiv s_2$ but $\mathbf{tt}.\langle \mathbf{ret}\ c_1 \mid \square\rangle^{\oplus} \not\equiv \mathbf{ff}.\langle \mathbf{ret}\ c_2 \mid \square\rangle^{\oplus}$.

However, if we consider the environment given by E, where

$$\mathsf{E} := (\{[]\ \ (\lambda x.\mathbf{op}\ \langle\rangle)\}\ \mathbf{with}\ \mathsf{h})\ \mathbf{tt}$$

and

$$\mathsf{h} := \mathsf{h}^{s}_{\mathsf{state}} \uplus \{\mathbf{op}\ \langle\rangle\ k \mapsto \mathbf{let}\ x = \mathbf{get}\ \langle\rangle\ \mathbf{in}\ \mathbf{if}\ x\ \mathbf{then}\ \mathbf{set}\ \mathbf{ff}\ \mathbf{else}\ \mathbf{set}\ \mathbf{tt};\ k\ x\}$$

Their interaction is witnessed exactly by the trace t_2.

$$t_2 := \overbrace{\underbrace{\boldsymbol{d_f}.\langle \mathbf{ret}\ f \mid \square\rangle^{\oplus}\ \boldsymbol{f}.\langle \square\ g \mid d_0\rangle^{\ominus}\ \boldsymbol{g}.\langle \square\ \mathsf{A} \mid c_1\rangle^{\oplus}\ \boldsymbol{d_1}.\langle \kappa[e] \mid \square\rangle^{\ominus}\ \boldsymbol{c_f}.\langle \kappa_1 \cdot \kappa[e] \mid \square\rangle^{\oplus}}_{s_2}\ \boldsymbol{\kappa_1}.\langle \square[\mathbf{ret}\ \mathbf{tt}] \mid d_1\rangle^{\ominus}}^{s_1}$$

$$t_1 := \overbrace{\underbrace{\boldsymbol{d_f}.\langle \mathbf{ret}\ f \mid \square\rangle^{\oplus}\ \boldsymbol{f}.\langle \square\ g \mid d_0\rangle^{\ominus}\ \boldsymbol{g}.\langle \square\ \mathsf{A} \mid c_1\rangle^{\oplus}\ \mathbf{fwd}(\kappa_1)}_{q_2}\ \boldsymbol{\kappa_1}.\langle \square[\mathbf{ret}\ \mathbf{tt}] \mid d_1\rangle^{\ominus}\ \boldsymbol{g}.\langle \square\ \mathsf{A} \mid c_2\rangle^{\oplus}\ \mathbf{fwd}(\kappa_2)\ \boldsymbol{\kappa}}^{q_1}$$

It is clear that these two traces are equivalent up to $\simeq_o$.

From Propponent's perspective these two executions are indistinguishable, therefore even though Opponent's behaviour is not constrained to be *innocent*, there exists an observationally equivalent behaviour that is. This is a case of lax-innocence.

Two natural questions arise:

- Is there a general definition of view, under which t_2 is innocent?
- Supposing there is one, does lax-innocence always hold? Can the same be said about lax-visibility?
- If so, then t_1 represents a case of lax-innocence. Does lax-innocence always hold? Is it the same for lax-visibility?

In the following section, we will turn our attention to the interaction of the underlying structure of effect forwarding and handling of traces and that of justification, this will allow us to recover the adequate structure necessary to appreciate and answer the questions above.

In the following, we will extend this to all OGS traces.

Notice how Opponent's point-of-view is not mentioned in the definition of O-visibility. The idea is that every point-of-view is associated to a set of P-names, i.e a fixed scope or an O-view. And every Opponent function name is associated to a point of view. That is how O-visibility can be .. at the level of the OGS LTS in

In the presence of algebraic effects and handlers, variables can escape their scope when used as parameters of an algebraic operation **op**. The scope corresponding to the evaluation stack that handles **op** is therefore dynamic and not a fixed set.

We will capture this notion of dynamic visibility, by giving a more relaxed definition of the view by considering **fwd**-transitions.

Definition 6.9 (Lax Opponent visibility). Given a trace t, we say that it is O-lax-visible and write $\mathbf{Visible}^{\mathsf{lax}}_{\mathsf{O}}(\mathsf{t})$ when, all the P-names in an Opponent move **o** are in its view; that is in the view of the corresponding O-ending trace:

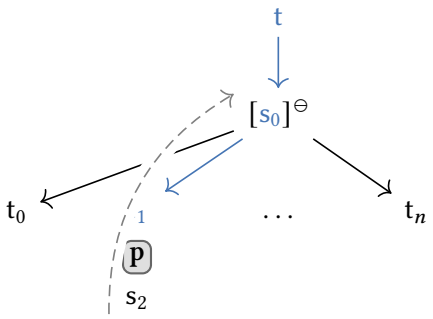
$$\forall \mathsf{s}, \mathbf{o}.\ \mathsf{s}\,\mathbf{o} \sqsubseteq \mathsf{t} \implies \mathsf{supp}(\mathbf{o}) \subseteq \mathcal{S}_{\ominus}(^{)}\mathsf{lax}(\mathsf{s})$$

6.6 The View, Revisited

We will introduce a few handy notations to express operations on sets of traces. If $\mathcal{V}$ is a set of traces in $\mathbf{Tr}_{\mathsf{ogs}}$ and s is a sequence in $\mathbf{Tr}_{\mathsf{ogs}}$. Then we will write $\mathcal{V}\,\mathsf{s}$ for the point-wise appending of s to the elements of $\mathcal{V}$, that is:

$$\mathcal{V}\,\mathsf{s} := \{\mathsf{t}\,\mathsf{s} \mid \mathsf{t} \in \mathcal{V}\}$$

Definition 6.10 (Opponent's view). Opponent's view is given at any stage of the interaction by the mutually recursive functions $\ulcorner\urcorner_{\mathsf{OFwd}}, \ulcorner\urcorner_{\ominus}, \urcorner\urcorner_{\ominus} : \mathbf{Tr}_{\mathsf{ogs}} \rightharpoonup \wp(\mathbf{Tr}_{\mathsf{ogs}})$ defined on P-ending traces and $\llcorner\lrcorner_{\ominus} : \mathbf{Tr}_{\mathsf{ogs}} \rightharpoonup \wp(\mathbf{Tr}_{\mathsf{ogs}})$ defined on O-ending traces:

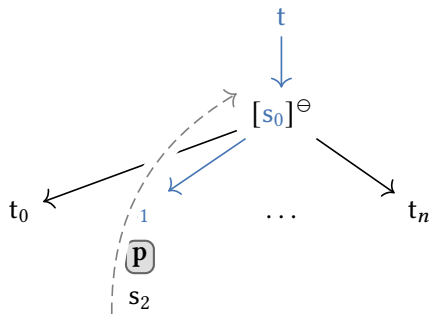


$$\urcorner \mathsf{t} \urcorner_{\ominus} := \ulcorner \mathsf{t} \urcorner_{\ominus} \cup \boxed{\ulcorner \mathsf{t} \urcorner_{\mathsf{ofwd}}} \tag{5}$$

$$
\begin{array}{lcl}
\ulcorner \mathsf{t}\ \mathbf{fwd}(\kappa_0,\cdots,\kappa_\ell)\urcorner_{\mathrm{ofwd}} & := & \urcorner \mathsf{t} \urcorner_{\ominus} \mathbf{fwd}(\kappa_0,\cdots,\kappa_\ell) \\
\ulcorner \mathsf{t} \urcorner_{\mathrm{ofwd}} & := & \emptyset
\end{array}
$$

$$
\begin{array}{lcll}
\ulcorner \mathsf{t}\ \boldsymbol{a}.\mathsf{p}^{\ominus}\ \mathsf{s}\ \boldsymbol{f}.\langle \square\, \mathsf{A} \mid c\rangle^{\oplus}\urcorner_{\ominus} & := & \urcorner \mathsf{t} \urcorner_{\ominus}\ \boldsymbol{a}.\mathsf{p}^{\ominus}\ \boldsymbol{f}.\langle \square\, \mathsf{A} \mid c\rangle^{\oplus} & \text{where } f \in \mathrm{supp}(\mathsf{p}) \\
\ulcorner \mathsf{t}\ \boldsymbol{c'}.\langle \kappa :: r[e] \mid \square\rangle^{\ominus}\ \mathsf{s}\ \boldsymbol{\kappa}.\mathsf{p}^{\oplus}\urcorner_{\ominus} & := & \urcorner \mathsf{t} \urcorner_{\ominus}\ \boldsymbol{c'}.\langle \kappa :: r[e] \mid \square\rangle^{\ominus}\ \boldsymbol{\kappa}.\mathsf{p}^{\oplus} & \\
\ulcorner \mathsf{t}\ \boldsymbol{\kappa'}.\langle \square[\kappa :: r[e]] \mid d\rangle^{\ominus}\ \mathsf{s}\ \boldsymbol{\kappa}.\mathsf{p}^{\oplus}\urcorner_{\ominus} & := & \urcorner \mathsf{t} \urcorner_{\ominus}\ \boldsymbol{\kappa'}.\langle \square[\kappa :: r[e]] \mid d\rangle^{\ominus}\ \boldsymbol{\kappa}.\mathsf{p}^{\oplus} & \\
\ulcorner \mathsf{t}\ \langle \underline{\mathsf{B}} \mid \overline{d}\rangle\ \mathsf{s}\ \boldsymbol{d}.\mathsf{p}^{\oplus}\urcorner_{\ominus} & := & \boxed{\llcorner \mathsf{t}\ \langle \underline{\mathsf{B}} \mid \overline{d}\rangle \lrcorner_{\ominus}}\ \boldsymbol{d}.\mathsf{p}^{\oplus} &
\end{array}
$$

$$
\begin{array}{lcl}
\llcorner \mathsf{t}\ \boldsymbol{g}.\langle \square\, \mathsf{B} \mid d\rangle^{\ominus}\lrcorner_{\ominus} & := & \urcorner \mathsf{t} \urcorner_{\ominus}\ \boldsymbol{g}.\langle \square\, \mathsf{B} \mid d\rangle^{\ominus} \\
\llcorner \mathsf{t}\ \mathbf{fwd}(\kappa_0,\cdots,\kappa_\ell)\ \mathsf{s}\ \boldsymbol{\kappa_\ell}.\mathsf{p}^{\oplus}\lrcorner_{\ominus} & := & \urcorner \mathsf{t}\ \mathbf{fwd}(\kappa_0,\cdots,\kappa_\ell)\urcorner_{\ominus}\ \boldsymbol{\kappa_\ell}.\mathsf{p}^{\oplus} \\
\llcorner \mathsf{t}\ \mathbf{fwd}(\kappa_0,\cdots,\kappa_\ell)\ \mathsf{s}\ \boldsymbol{\kappa_i}.\mathsf{p}^{\oplus}\lrcorner_{\ominus} & := & \ulcorner \mathsf{t}\ \mathbf{fwd}(\kappa_0,\cdots,\kappa_i)\urcorner_{\ominus}\ \mathbf{fwd}(\kappa_{i+1},\cdots,\kappa_\ell)\ \boldsymbol{\kappa_i}.\mathsf{p}^{\oplus}
\end{array}
$$



Proposition 6.11 (P-innocence). *Given any trace* $\mathsf{t} \in \mathbf{Tr}_{\mathrm{ogs}}$*, we have that* t *is* P*-innocent.*

Lemma 6.12 (Lax O-innocence). *Given any trace* $\mathsf{t} \in \mathbf{Tr}_{\mathrm{ogs}}$*, there exists a trace* t_{in} *such that* $\mathsf{t} \simeq_o \mathsf{t}_{in}$ *and* t_{in} *is* O*-innocent.*

Proposition 6.13 (P-visibility). *Given any trace* $\mathsf{t} \in \mathbf{Tr}_{\mathrm{ogs}}$*, we have that* t *is* P*-visible.*

Lemma 6.14 (Lax O-visibility). *Given any trace* $\mathsf{t} \in \mathbf{Tr}_{\mathrm{ogs}}$*, there exists a trace* t_{vis} *such that* $\mathsf{t} \simeq_o \mathsf{t}_{vis}$ *and* t_{vis} *is* O*-visible.*

The previous property captures the property that an environment of a term is not stateful.

If an environment is to behave in a way that is not observationally innocent, it has to be stateful, and this can be achieved, in our setting, through an encoding of a state using algebraic operations and a handler. However, supposing it is not innocent w.r.t to some function w, then this function must have the following *intensional* behaviour: performing an effect to read the state, which in is propagated to some evaluation stack K that handles it and encodes the state.

Semantically, at the level of traces this effect propagation is observed in the form of **fwd**-transitions and the handler corresponds to an Opponent point-of-view. therefore

instead of the strict notion O-innocence whereby Opponent is expected to have the exact same *observational* behaviour and output the same result as in def. 6.8, we can relax this notion of O-innocence to capture this *intensional* behaviour.

7 Soundness of the model

Theorem 7.1 (Adequacy). *Given an interactive term* $\mathtt{I};\Gamma \vdash_c \mathtt{M}$ *and a compatible ciu-substitution* $\mathtt{I}';c_f : \neg\mathbb{1} \vdash \gamma : \Gamma$*, then by writing* $\mathbb{P}$ *and* $\mathbb{E}$ *for their respective* ogs *embeddings, we have:*

$$\exists \mathtt{I}'' \supseteq \mathtt{I}'\cdot\mathtt{I}.\ \mathtt{M}\{\gamma\} \Downarrow_{\mathrm{op}} [c_f]\ \mathbf{ret}\ \langle\rangle$$

$$\Longleftrightarrow$$

$$\exists t \in \mathbf{CTr}_{\mathrm{ogs}}(\mathbb{P}).\ t^{\perp}\, \boldsymbol{c_f}.\langle \mathbf{ret}\ \langle\rangle \mid \square\rangle^{\oplus} \in \mathbf{CTr}_{\mathrm{ogs}}(\mathbb{E})$$

Proof. The proof of the previous adequacy theorem 7.1 is detailed in section F. It relies on the introduction of another LTS (*cf.* section E) that formalizes the notion of *concrete interaction* of example **??**. This allows to define semantic notions of composition and observation (at the level of configurations and complete traces) which are then shown, in sections G and H, to coincide with their syntactic/operational counterpats. □

Theorem 7.2 (Soundness). *Taking* $\mathtt{M}_1, \mathtt{M}_2$ *two interactive terms such that both* $\mathtt{I};\Gamma \vdash_c \mathtt{M}_i$ *(for* $i \in \{1,2\}$*), suppose that* $\mathbf{CTr}(\langle \mathtt{I};\Gamma \vdash \mathtt{M}_1\rangle) \subseteq \mathbf{CTr}(\langle \mathtt{I};\Gamma \vdash \mathtt{M}_2\rangle)$*. Then* $\mathtt{I};\Gamma \vdash \mathtt{M}_1 \preceq_{ciu} \mathtt{M}_2$*.*

Proof. We take a continuation name c_f, an instance context $\mathtt{I}' \supseteq \mathtt{I}$ and a name assignment γ such that $\mathtt{I}';c_f : \neg\mathbb{1} \vdash \gamma : \Gamma$. We write $\mathbb{G}_{P,i}$ for $\langle \mathtt{I};\Gamma \vdash \mathtt{T}_i\rangle$, and $\mathbb{G}_O$ for $\langle \mathtt{I}';c_f : \neg\mathbb{1} \vdash \gamma : \Gamma\rangle$.

Suppose that $\mathtt{T}_1\{\gamma\} \Downarrow_{\mathrm{op}} [c_f]\langle\rangle$. Then from Theorem 7.1, we get that $\mathbb{G}_{P,1} \between \mathbb{G}_O \Downarrow$, and from Theorem **??** that there exists a trace $t_1 \in \mathbf{CTr}(\mathbb{G}_{P,1})$ such that $t_1^{\perp}\boldsymbol{c_f}.\langle \mathbf{ret}\ \langle\rangle \mid \square\rangle^{\oplus} \in \mathbf{CTr}(\mathbb{G}_O)$. From $\mathbf{CTr}(\mathbb{G}_{P,1}) \preceq_{tr} \mathbf{CTr}(\mathbb{G}_{P_2})$, we get the existence of a trace $t_2 \simeq_p t_1$ such that $t_2 \in \mathbf{CTr}(\mathbb{G}_{P,2})$. From the O-saturation property (Lemma 5.23), we get that $t_2^{\perp}\boldsymbol{c_f}.\langle \mathbf{ret}\ \langle\rangle \mid \square\rangle^{\oplus} \in \mathbf{CTr}(\mathbb{G}_O)$. So from Theorem **??** in the other direction, we get that $\mathbb{G}_{P,2} \between \mathbb{G}_O \Downarrow$. Finally, from Theorem 7.1, we get that $\mathtt{T}_2\{\gamma\} \Downarrow_{\mathrm{op}} [c_f]\langle\rangle$, as we wanted. □

8 Completeness of the model

Definable handling structures:

These are essentially abstract whose captured substructures are annotated with an identifying component making them easily amenable to ···

These ingredients are necessary to simulate non-innocence and non-visibility.

Breaking observational innocence:

As alluded to in ···, the nuance that is actually strictly invisible yet spawns an effect, and the stateful reaction actually handled by the effect handler.

As the model stands, every effect is associated to a fresh name, i.e., there exist **effectful** moves that break internal innocence. We call such traces **degenerate traces**.

In this model, every effect invocation is tied to a fresh name, giving rise to **effectful** moves that may break internal innocence. Traces that exhibit such behavior; *i.e.* that

$$\frac{\mathtt{t} = \mathtt{s}_0\, \vec{\mathbf{fwd}}(e^{\mathtt{A}}, r)\mathtt{s}_1 \qquad \mathtt{s}_0 \downarrow^{f}_{vis} \mathtt{s}'_0 \qquad f \in \mathrm{supp}(\ulcorner \mathtt{t} \urcorner_{\ominus}) \setminus (\mathrm{supp}(\ulcorner \mathtt{t} \urcorner_{\ominus}) \cup \mathrm{supp}(\mathtt{A}))}{\mathtt{t} \downarrow^{f}_{vis} \mathtt{s}'_0\, \vec{\mathbf{fwd}}(e^{\langle \mathtt{A}, f\rangle}, r)\, \mathtt{t}_1}$$

$$\mathtt{X} = \bigcup \{\mathrm{supp}(\mathtt{p}) \mid \exists \mathtt{t}'.\, \mathtt{t}'\, \mathbf{p}.a^{\oplus}\} \qquad \mathtt{t} \downarrow^{\mathtt{X}}_{vis} \mathtt{s}$$

break determinism through non-local naming or escaping effects—are termed **degenerate traces**.

We define a translation turning an degenerate trace into an internally innocent one. In order to do so, we define a marking on the $\mathcal{H}$-generated structure backwards and use it to ....

To mitigate degeneracy, we define a translation that maps degenerate traces to internally innocent ones. This involves annotating the HH-generated structure in a backward pass, effectively tracing dependencies in reverse and restoring structural regularity.

In the following, we assume that all examined are non-degenerate.

Breaking observational visibility:

Function names: these can "escape" their scope through effect propagation, by being a parameter of an effect operator.

Thus, given a **non-visible** trace, every name is either strictly visible, or dependent on the use of effect and handlers, which can be read off the trace / structure.

Suppose we have $\mathtt{s}\,\mathbf{p}.f^{\ominus} \sqsubset \mathtt{t}$ where $f \notin \mathrm{supp}(\ulcorner \mathtt{s} \urcorner_{\ominus})$, the idea is to follow the handling structure (**fwd**-transitions) backwards and to annnotate the effect names e in s that responsible for rendering f visible. We use a subscript A on the corresponding effect names e.

We define coniductively the visibility annotation using a rewriting rule $\downarrow^{\mathtt{X}}_{vis}$ labeled with a set of function names X.

Definable reaction trees:

We formalize these behavioral trees in terms of definable handling structures. These serve as the counterpart to strictly innocent and visible traces, offering a structured view of possible responses based on observational structure.

They are given by the following grammar, and can be obtained ...

$$\mathbb{h} ::= \mathtt{t} \mid \mathbb{h}\, [\![\, \mathtt{t} \,]\!]^{\eta}\, \mathbb{h} \mid \mathbb{h}\langle \mathbb{h}, \ldots, \mathbb{h} \rangle$$

where $\eta ::= \langle e, i, \mathtt{A} \rangle$

- A is an abstract value, such that supp(A) is the set of names that escape their original scope through the propagation of the effect e — this one is crucial to determine properly underlying effect operation **op** in the underlying definability proof, which we will suppose that it should have been used as follows **op** v with v $\nearrow$ A.
- m on the other hand indicates the number of times the effect e (*i.e.* **op**v) has been replayed so far. It is crucial to maintain an internal clock, especially if the effect

is handled by the same handler, as it can be key element that enables a stateful behaviour that breaks of observational innocence.

Definition 8.1 (definability tree). Given a definable h-struct $\mathbb{h}$, we define the corresponding definability tree as follows:

$$\mathcal{T}[\mathbb{h}] := \{\mathcal{I}(a, \mathbb{h}) \mid \mathbf{p}.b^{\ominus} \sqsubset \mathbb{h} \wedge a \in \mathrm{supp}(\mathrm{p})\}$$

We will use the powerset notation $\wp(\mathcal{T})$ to refer to the set of (reaction and introduction) sub-trees of $\mathcal{T}$. Given a definability tree $\mathcal{S} = \mathcal{T}[\mathbb{t}]$, we will denote by $\mathcal{S}_{/a}$ and $\mathcal{S}_{/\mathbf{o}}$ the sub-trees $\mathcal{I}(a, \mathbb{t})$, $\mathcal{R}(\mathbf{o}, \mathbb{t}) \in \wp(\mathcal{S})$, respectively.

Lemma 8.2 (Innocent trees). *Given an* O*-innocent definable handling structure* $\mathbb{h}$*, then for any introduction sub-tree* $\mathcal{I} \in \wp(\mathcal{T}[\mathbb{h}])$*:*

Definability

We now turn to a key definability result concerning OGS traces. This property underpins the completeness of the semantic model, ensuring that every well-formed trace has a corresponding generating configuration. The core idea is this: given a complete trace t produced by some initial configuration, we construct a CIU-environment given by γ that precisely captures the observable behavior exhibited in t—excluding any dissonant behaviors, that is:

$$\mathbf{Tr}(\langle\gamma\rangle) = \{\mathrm{s} \mid \mathrm{s} \leq_o \mathrm{t}^{\perp}\, \langle\rangle.\langle \mathbf{ret}\, c \mid \square\rangle^{\ominus}\}$$

Proof Sketch. We proceed as follows:

(1) We prove this result for O-pure traces.
(2) By induction on the length and **fwd**-structure of s, we write $\mathrm{s} = \mathrm{t}_i^0\, \mathrm{q}_i\, t_j^1$ where $\forall i < j, \mathrm{t}_i^0 \sqsubseteq \mathrm{t}_j^0$ and q_i is a **fwd**-starting sequence (precisely, an h-struct redex).

We show that there exist a path of traces:

$$s = t_0 \xrightarrow{fwd} t_1 \xrightarrow{fwd} \ldots \xrightarrow{fwd} t_m = s$$

and $\mathcal{T}(t) \cup \mathcal{T}(s)$ induces a CLI env δ that generates t_i and rejects all discordant traces.

$$\mathcal{H} ::= \square \mid \mathcal{F} \cdot t \mid \mathcal{F} \cdot \langle t, \ldots, t\rangle$$

□

Lemma 8.3.

$$\langle \mathtt{ret}\ \ A, \mathcal{R}(q.p, \bar{s})\rangle \in \bar{S}_d$$

Lemma 8.4 (O-pure definability). *Given a* O*-pure trace* t*, there exists a CIU-environment, i.e. a name assignment* γ *such that:*

$$\mathbf{Tr}_{\mathrm{ogs}}(\langle\gamma\rangle) = \{\mathrm{s} \mid \exists \mathrm{t}'.\ \mathrm{s} <_o \mathrm{t}' \wedge \mathrm{t}' \sqsubseteq \mathrm{t}\}$$

Proof. We proceed by exhibiting such a name assignment γ (1), then we show that it generates $\mathrm{t}^{\dagger}$, then we show that for any $\mathrm{s}'\,\mathbf{o} \sqsubseteq \mathrm{t}$ and $\mathbf{o}'$ such that $\mathrm{s}'\,\mathbf{o}'$ and $\mathrm{s}'\,\mathbf{o}$ are discordant, then:

$$\mathrm{s}'\,\mathbf{o}'^{\perp} \notin \mathbf{Tr}_{\mathrm{ogs}}(\langle\gamma\rangle)2$$

Let $\mathcal{T} = \mathcal{T}[\mathsf{t}^\dagger]$, we define the corresponding γ, denoted $\delta(\mathcal{T})$, inductively by constructing one out of name assignemts $\delta(\mathcal{T}_{/a})$ corresponding to each introduction sub-tree $\mathcal{T}_{/a} \in \wp(\mathcal{T})$.

construction:

$$\gamma := \delta(\mathcal{T}) = \delta(\mathcal{T}_{/\star})$$

$$\delta(\mathcal{T}_{/f}) := f \mapsto \lambda x.\,\mathtt{match}\ x\ \mathtt{with}\ \{\mathsf{A} \Rightarrow \mathbf{o}\{\delta(\mathcal{T}_{/\mathbf{o}})\}\}_{\langle \mathsf{A}, \mathcal{T}_{/\mathbf{o}}\rangle \in \mathcal{T}_{/f}}$$

$$\delta(\mathcal{T}_{/c}) := c \mapsto \mathtt{let}\ x = \square\ \mathtt{in}\ \mathtt{match}\ x\ \mathtt{with}\ \{\mathsf{A} \Rightarrow \mathbf{o}\{\delta(\mathcal{T}_{/\mathbf{o}})\}\}_{\langle \mathsf{A}, \mathcal{T}_{/\mathbf{o}}\rangle \in \mathcal{T}_{/c}}$$

$$\delta(\mathcal{T}_{/\mathbf{o}}) := a \in \mathrm{supp}(\mathbf{o})\ \delta(\mathcal{T}_{/a})$$

Case t is P-pure:

It is straightforward to verify that the lifted trace $\mathsf{t}^\dagger$ is generated by the passive state $\langle\gamma\rangle$. Specifically, if an interaction fragment $\mathtt{t}\,\mathbf{p}.a^{\ominus}\,\mathbf{p}$ occurrs in $\mathsf{t}^\dagger$, then we can proceed by induction on the length of $\mathsf{t}^\dagger$ to show that.

$$\langle\gamma\rangle \overset{\mathsf{s}\,\mathbf{p}.a^{\ominus}}{\Longrightarrow}_{\mathrm{ogs}} \langle\langle \mathsf{p}[\gamma'(a) \mid \mathbf{o}\ [] \rangle, \gamma'\rangle \xrightarrow{\mathrm{eval}}_{\mathrm{ogs}} \langle \mathsf{Nf}, \mathtt{I}'', \gamma'\rangle$$

such that

$$\mathbf{abstr}_{\mathrm{nf}}((, \mathsf{Nf})) = (\mathbf{p}, \_) \quad \text{and} \quad \langle \mathsf{p}, \mathcal{T}_{/\mathbf{p}}\rangle \in \mathcal{T}_{/a}$$

Case t is P-effectful:

In this case we have to account for Proponent moves of the form:

$$\boldsymbol{\kappa}.\langle \square[\mathsf{A}] \mid c\rangle^{\oplus}$$

We proceed by induction on the length of $t^{\uparrow}$ and the forward structure.

Suppose $\mathsf{s}\,\boldsymbol{\kappa}.\langle\square[\mathsf{A}] \mid c\rangle^{\oplus}\ \mathsf{q}\,\mathbf{o} \in \mathsf{t}^\dagger$

Then we can write $\mathsf{s} = \mathsf{s}_0\,\overline{\mathbf{fwd}}(\kappa)\,\mathsf{s}_1$. We treat here the subcase where $\overline{\mathbf{fwd}}(\kappa) = \overline{\mathbf{fwd}}(d, c; \kappa)$ only. The others are similar.

By induction hypothesis:

$$\begin{aligned}
\langle\gamma\rangle \overset{\mathsf{s}_0}{\Longrightarrow}_{\mathrm{ogs}} \mathbb{G}_1 \quad & \xrightarrow{\boldsymbol{d}.\langle\kappa[e]\mid\square\rangle^{\oplus}}_{\mathrm{ogs}} && \langle\gamma(d)[\kappa[e]], \gamma_1', \delta\rangle \\
& \xrightarrow{\boldsymbol{c}.\langle\varkappa\circ\kappa[e]\mid\square\rangle^{\ominus}}_{\mathrm{ogs}} && \langle\gamma_1', \delta.[\kappa \mapsto \mathsf{F}]\rangle \\
& && \mathbb{G}_1 = \langle\gamma''\rangle \\
& \xrightarrow{\boldsymbol{\kappa}.\langle\square[\mathsf{A}]\mid c\rangle^{\oplus}}_{\mathrm{ogs}} && \langle \mathsf{F}[\delta(d)][\mathbf{ret}\,\mathsf{A}], \gamma''\rangle = \mathbb{G}_2
\end{aligned}$$

□

Theorem 8.5 (Completeness). *Taking* $\mathsf{T}_1, \mathsf{T}_2$ *two named terms such that both* $\mathtt{I}; \Gamma \vdash_{\mathsf{c}} \mathsf{T}_i$ *(for* $i \in \{1, 2\}$*), suppose that* $\mathtt{I}; \Gamma \vdash \mathsf{T}_1 \sqsubseteq_{sub} \mathsf{T}_2$*. Then* $\mathbf{CTr}(\langle \mathtt{I}; \Gamma \vdash \mathsf{T}_1\rangle) \preceq_{tr} \mathbf{CTr}(\langle \mathtt{I}; \Gamma \vdash \mathsf{T}_2\rangle)$.

Proof. We write $\mathbb{G}_{\mathrm{P},i}$ for $\langle \mathtt{I}; \Gamma \vdash \mathsf{T}_i\rangle$. Taking $\mathsf{t}_1 \in \mathbf{CTr}(\mathbb{G}_{\mathrm{P},1})$ and c_{f} a continuation name, from the definability property (Lemma **??**), we get the existence of a trace $\mathsf{s} \simeq_o \mathsf{t}_1$ and an OGS configuration $\mathbb{G}_{\mathrm{O}}$ composable with $\mathbb{G}_{\mathrm{P},1}$ such that $\mathbf{CTr}(\mathbb{G}_{\mathrm{O}}) = \{\mathsf{t} \mid \mathsf{t} \simeq_o \mathsf{s}^{\perp}\overline{c_{\mathsf{f}}}(\langle\rangle)\}$. Then from Lemma H.14, we get that $\mathbb{G}_{\mathrm{O}}$ can be written as $\langle \mathtt{I}'; c_{\mathsf{f}} : \neg\mathbb{1} \vdash \gamma : \Gamma\rangle$, with $\mathtt{I}'$ an instance context satisfying $\mathtt{I}' \supseteq \mathtt{I}$, and γ a name assignment such that $\mathtt{I}'; c_{\mathsf{f}} : \neg\mathbb{1} \vdash \gamma : \Gamma$.

From Theorem 7.1, we get that $\mathbb{G}_{\mathrm{P},2} \between \mathbb{G}_{\mathrm{O}} \Downarrow$. Applying Theorem **??**, we deduce that $\mathtt{T}_1\{\gamma\} \Downarrow_{\mathrm{op}} c_{\mathrm{f}}[\langle\rangle]$. So from $\mathtt{I};\Gamma \vdash \mathtt{T}_1 \sqsubseteq_{sub} \mathtt{T}_2$, we deduce that $\mathtt{T}_2\{\gamma\} \Downarrow_{\mathrm{op}} c_{\mathrm{f}}[\langle\rangle]$. From Theorem **??** and Theorem 7.1, we get that there exists a trace t_2 such that $\mathrm{t}_2 \in \mathbf{CTr}(\mathbb{G}_{\mathrm{P},1})$ and $\mathrm{t}_2^{\perp}\overline{c_{\mathrm{f}}}(\langle\rangle) \in \mathbf{CTr}(\mathbb{G}_{\mathrm{O}})$. By definition of $\mathbb{G}_{\mathrm{O}}$, we get that $\mathrm{t}_2^{\perp}\overline{c_{\mathrm{f}}}(\langle\rangle) \simeq_o \mathrm{s}\ \overline{c_{\mathrm{f}}}(\langle\rangle)$.

Given that $\mathrm{t}_1 \simeq_o \mathrm{s} \simeq_p \mathrm{t}_2$, the Lemma 5.24 ensures the existence of t_2' s.t $\mathrm{t}_1 \simeq_p \mathrm{t}_2' \simeq_o \mathrm{t}_2$. We thus have $\mathrm{t}_2' \in \mathbf{Tr}_{\mathrm{OGS}}(\mathbb{G}_2)$ (by Lemma 5.23) s.t $\mathrm{t}_1 \simeq_p \mathrm{t}_2'$, as wanted. □

9 Conclusion and related work

We have provided a sound interactive model for a language with dynamically-scoped effect handlers and fresh generation of effect instances. Our model does not allow so far exchange of effect instances between the program and its environment, that would necessitate to keep track of such disclosed instances in a specific set, to enforce a non-omniscience property as in [Ghica and Tzevelekos 2012]. The well-bracketing LTS would also have to keep track of the exchanged delimited continuations when the program performs a public effect (using a disclosed instance that is) because the environment, in this case, would be able to handle it, and thus its behaviour must be bound accordingly.

In future work, we will present a notion of equivalence between set of complete traces, coarser than equality, that we conjecture to be fully-abstract.

Game semantics has a long history of providing a fully-abstract interactive models for languages with control operators, starting from the relaxation of the well-bracketed condition by Laird to capture control operators like call/cc [Laird 1997] and exceptions [Laird 2001]. Game models for both statically bound, first-class continuations and locally declared, dynamically bound prompts were presented in [Laird 2017], via a monadic presentation of such effects. In this work, prompts and exceptions cannot be referred by their names and passed around.

To represent them as values, nominal game semantics has been developed as a versatile framework to handle generative effects like dynamic name creation [Abramsky et al. 2004], higher-order references [Murawski and Tzevelekos 2011] and generative exceptions and handlers [Murawski and Tzevelekos 2014].

Operational techniques like applicative, normal-form and environmental bisimulations has also been developed for higher-order language with fine-grained control operators like static delimited continuations via shift and reset operations [Biernacki et al. 2019a], dynamic ones via prompt and reset [Aristizábal et al. 2017], and algebraic effects and handlers were considered in [Biernacki et al. 2020]. This last work is the most relevant to us, however they do not consider fresh generation of instances. Using a notion of Kripke normal-form bisimulations as introduced in [Hirschkoff et al. 2023; Jaber and Murawski 2021b], that is directly derived from an OGS model, it would be interesting to explore the development of normal form bisimulations for such fresh generation of effect instances.

More Related Work

Algebraic effects and handlers have emerged as a powerful and flexible abstraction for structuring computational effects in programming languages. Their foundation in algebraic operations provides an elegant and modular framework for reasoning about programs. In such a setting, effects are typically described via equational theories, and handlers are expected to interpret or realize these operations in a manner consistent with the underlying algebra.

However, the very generality and expressiveness of user-defined handlers introduce significant semantic complexity. When handlers are allowed to operate in arbitrary ways—without being constrained by a fixed algebraic specification—the traditional reasoning principles begin to erode. For example, handlers may intercept operations out of order, selectively ignore them, or invoke continuations in unconventional ways. This threatens

key properties like equational reasoning, compositionality, and even type safety, especially when combined with effects like state or nondeterminism.

Recent work has responded to these challenges in two broad ways. On one side are efforts to restrict the language of handlers—e.g., by disallowing non-algebraic operations or enforcing shallow handlers—to preserve soundness and compositionality. A representative of this line is [Xie et al. 2020], which proposes a restricted subset of handlers for which equational reasoning and semantic well-behavior can be recovered. We show that this restriction corresponds to in our semantics model, although the considered language in that work is not the same and some of the underlying issues also do not arise when considering generality.

- It is worth studying these fragments and investigate what our model can add to the understanding thereof.
- Scoped effects can be seen as a good programming practice since it induces a well-disciplined semantics, but at the same time it rules out uninvited behaviours in arbitrary/adversarial environments.
- It is worth studying whether these mis-handling of effects persists when we introduce generality and private instance.

On the other side, however, is an emerging body of research that seeks to embrace the generality of effect handlers and give it rigorous semantic footing []. Rather than enforcing equational constraints syntactically, these approaches provide a semantic account of handler behavior. This includes non-equational reasoning, where the meaning of a program is characterized not by equations but by the structure of its interactions—including the control flow, branching, and observability of effects. Our work aligns with the latter perspective. Inspired by efforts such as Kiselyov's Not by Equations Alone, we propose a model where arbitrary handlers can be studied and reasoned about semantically, even when no clear equational theory exists to govern their behavior. The use of structured traces and handling trees allows us to capture and reflect the complex dynamics of control and effect invocation, including delimited continuations, resumption, and dynamic branching.

—

Shallow Handlers

It is worth noting, the model's completeness result does not depend on the presence of catamorphic handlers. Indeed any effectful interaction assumes that the effect is different — unless assuming so breaks virtual innocence. Even when considering only shallow handlers, we conjecture that the definability result and its proof remain valid, confirming the expressibility result of [Hillerström et al. 2020] We also conjecture that there is no required change in our model to account for shallow handlers.

Event Structures

Our model considers sequential traces that are sequential and alternating.

However, the point structure hides/veils an intricate control-flow structure, which, coupled with effect information, can be used to uncover a richer structure that underpins

the well-bracketing conditions and induces a generalization of view that can be understood as the sum of standard views.

The emergence of these nuanced structures is unsurprising. The study of delimited continuations—particularly when managed via effect handlers—exposes a control-flow landscape that straddles two extremes: strict stack discipline (with exactly one valid continuation at each step), and fully liberalized nondeterminism (where multiple continuations may coexist). Our setting reveals a middle ground: several "answering events" may be enabled concurrently, yet they remain compatible. This naturally leads to a structure akin to event structures in concurrency theory.

In our setting, “answering events” enable several others (in general) which may be understood as independent events but compatible, so it is not surprising that this structure emerges again akin to “event structure”.

Delimited continuations, especially as exposed via effect handlers, offer a fine-grained mechanism for suspending and resuming computation. Unlike classical stack-based control, delimited continuations allow multiple resumable points to exist simultaneously, and effect handlers provide the means to inspect, discard, or resume them selectively. This introduces a form of control flow that is inherently branching and context-sensitive.

Interestingly, this behavior parallels key constructs in concurrency theory, particularly in the framework of event structures. Event structures model computations as partially ordered sets of events, where causality, independence, and conflict are explicitly represented. In this view, an "event" may depend on others before it can occur, or it may exclude the occurrence of other conflicting events. These structures are central in modeling nondeterminism, concurrent execution, and branching time.

The handler structures studied in our work exhibit similar features. A captured continuation (e.g., $[\![\mathbb{t}]\!]$) corresponds to a suspended computation — akin to an enabled event that is not yet triggered. The branching constructs (e.g., $\mathbb{t}\langle \mathbb{t}_0, \ldots, \mathbb{t}_m \rangle$) resemble a set of enabled events, where each branch represents a distinct potential continuation. These resumptions may or may not interfere with each other, depending on how the handler behaves — mirroring the notion of conflict or compatibility in event structures.

More broadly, our semantic model lives in an intermediate space between two extremes. On one end, strict stack-based control allows only one valid continuation at a time — enforcing a linear, sequential control structure. On the other, fully nondeterministic or concurrent semantics allow multiple continuations or events to coexist and evolve independently. Delimited continuations with handlers navigate this space by allowing several suspended computations to be present at once, while maintaining causal dependencies and structural coherence through the trace and pointer system.

The structured rewriting in our model, that is guided the justification structure, reinforces this analogy. These tools capture how branches relate to each other temporally and causally, much like the partial order in an event structure or the unfolding of a concurrent process. Consequently, our framework not only accommodates structured control and effectful interactions but also aligns with core principles from concurrency semantics — providing a semantic account that bridges programming language theory with event-based models of computation.

This analogy to event structures has not been widely explored in the study of effect handlers, and we believe it sheds new light on the control-theoretic and semantic properties of these constructs. It also opens the door to importing established reasoning tools from concurrency theory into the semantics of delimited control.

References

Samson Abramsky, Dan R. Ghica, Andrzej S. Murawski, C.-H. Luke Ong, and Ian David Bede Stark. 2004. Nominal Games and Full Abstraction for the Nu-Calculus. In *19th IEEE Symposium on Logic in Computer Science (LICS 2004), 14-17 July 2004, Turku, Finland, Proceedings*. IEEE Computer Society, 150–159. doi:10.1109/LICS.2004.1319609

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

Samson Abramsky and Guy McCusker. 1997. Game Semantics. In *Handbook of Logic in Computer Science*. Vol. 5. Oxford University Press.

Beniamino Accattoli, Pablo Barenbaum, and Damiano Mazza. 2014. Distilling Abstract Machines. In *Proceedings of the 19th ACM SIGPLAN International Conference on Functional Programming (ICFP '14)*. Association for Computing Machinery, New York, NY, USA, 363–376. doi:10.1145/2628136.2628154

Andrés Aristizábal, Dariusz Biernacki, Sergueï Lenglet, and Piotr Polesiuk. 2017. Environmental Bisimulations for Delimited-Control Operators with Dynamic Prompt Generation. *Log. Methods Comput. Sci.* 13, 3 (2017). doi:10.23638/LMCS-13(3:27)2017

Andrej Bauer and Matija Pretnar. 2015. Programming with algebraic effects and handlers. *Journal of Logical and Algebraic Methods in Programming* 84, 1 (2015), 108–123. doi:10.1016/j.jlamp.2014.02.001 Special Issue: The 23rd Nordic Workshop on Programming Theory (NWPT 2011) Special Issue: Domains X, International workshop on Domain Theory and applications, Swansea, 5-7 September, 2011.

Dariusz Biernacki, Sergueï Lenglet, and Piotr Polesiuk. 2019a. Bisimulations for Delimited-Control Operators. *Log. Methods Comput. Sci.* 15, 2 (2019). doi:10.23638/LMCS-15(2:18)2019

Dariusz Biernacki, Sergueï Lenglet, and Piotr Polesiuk. 2020. A complete normal-form bisimilarity for algebraic effects and handlers. In *Formal Structures for Computation and Deduction*.

Dariusz Biernacki, Maciej Piróg, Piotr Polesiuk, and Filip Sieczkowski. 2019b. Binders by day, labels by night: effect instances via lexically scoped handlers. *Proceedings of the ACM on Programming Languages* 4, POPL (2019), 1–29.

Jonathan Immanuel Brachthäuser, Philipp Schuster, and Klaus Ostermann. 2020. Effects as capabilities: effect handlers and lightweight effect polymorphism. *Proc. ACM Program. Lang.* 4, OOPSLA (2020), 126:1–126:30. doi:10.1145/3428194

Vincent Danos, Hugo Herbelin, and Laurent Regnier. 1996. Game semantics and abstract machines. In *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*. IEEE, 394–405.

Paulo Emílio de Vilhena and François Pottier. 2023. A type system for effect handlers and dynamic labels. In *Programming Languages and Systems: 32nd European Symposium on Programming, ESOP 2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2023, Paris, France, April 22–27, 2023, Proceedings*. Springer Nature Switzerland Cham, 225–252.

Dan R Ghica and Nikos Tzevelekos. 2012. A system-level game semantics. *Electronic Notes in Theoretical Computer Science* 286 (2012), 191–211.

Daniel Hillerström, Sam Lindley, and Robert Atkey. 2020. Effect Handlers via Generalised Continuations. 30 (2020), e5. doi:10.1017/S0956796820000040

Daniel Hirschkoff, Guilhem Jaber, and Enguerrand Prebet. 2023. Deciding Contextual Equivalence of ν-Calculus with Effectful Contexts. In *Foundations of Software Science and Computation Structures - 26th International Conference, FoSSaCS 2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2023, Paris, France, April 22-27, 2023, Proceedings (Lecture Notes in Computer Science, Vol. 13992)*, Orna Kupferman and Pawel Sobocinski (Eds.). Springer, 24–45. doi:10.1007/978-3-031-30829-1_2

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

Guilhem Jaber and Andrzej S Murawski. 2021a. Complete trace models of state and control. *Programming Languages and Systems* 12648 (2021), 348.

Guilhem Jaber and Andrzej S. Murawski. 2021b. Compositional relational reasoning via operational game semantics. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*. IEEE, 1–13. doi:10.1109/LICS52264.2021.9470524

James Laird. 1997. Full abstraction for functional languages with control. In *Proceedings of Twelfth Annual IEEE Symposium on Logic in Computer Science*. 58–67. doi:10.1109/LICS.1997.614931

James Laird. 2001. A Fully Abstract Game Semantics of Local Exceptions. In *16th Annual IEEE Symposium on Logic in Computer Science, Boston, Massachusetts, USA, June 16-19, 2001, Proceedings*. IEEE Computer Society, 105–114. doi:10.1109/LICS.2001.932487

James Laird. 2007. A Fully Abstract Trace Semantics for General References. In *Proceedings of the 34th International Conference on Automata, Languages and Programming* (Wrocław, Poland) *(ICALP'07)*. Springer-Verlag, Berlin, Heidelberg, 667–679.

James Laird. 2017. Combining control effects and their models: Game semantics for a hierarchy of static, dynamic and delimited control effects. *Ann. Pure Appl. Log.* 168, 2 (2017), 470–500. doi:10.1016/J.APAL.2016.10.011

Søren B. Lassen and Paul Blain Levy. 2008. Typed normal form bisimulation for parametric polymorphism. In *2008 23rd Annual IEEE Symposium on Logic in Computer Science*. IEEE, 341–352.

Paul Blain Levy. 2004. *Call-By-Push-Value: A Functional/Imperative Synthesis*. Semantics Structures in Computation, Vol. 2. Springer.

Ian Mason and Carolyn Talcott. 1991. Equivalence in functional languages with effects. *Journal of Functional Programming* 1, 3 (1991), 287–327. doi:10.1017/S0956796800000125

James Hiram Morris Jr. 1969. *Lambda-calculus models of programming languages.* Ph. D. Dissertation. Massachusetts Institute of Technology.

Andrzej S Murawski and Nikos Tzevelekos. 2011. Game semantics for good general references. In *2011 IEEE 26th Annual Symposium on Logic in Computer Science*. IEEE, 75–84.

Andrzej S Murawski and Nikos Tzevelekos. 2014. Game semantics for nominal exceptions. In *Foundations of Software Science and Computation Structures: 17th International Conference, FOSSACS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014, Proceedings 17*. Springer, 164–179.

Gordon Plotkin and John Power. 2001. Semantics for algebraic operations. *Electronic Notes in Theoretical Computer Science* 45 (2001), 332–345.

Gordon Plotkin and John Power. 2002. Computational effects and operations: An overview. (2002).

Gordon D Plotkin and Matija Pretnar. 2013. Handling algebraic effects. *Logical methods in computer science* 9 (2013).

Ningning Xie, Jonathan Immanuel Brachthäuser, Daniel Hillerström, Philipp Schuster, and Daan Leijen. 2020. Effect Handlers, Evidently. *Proc. ACM Program. Lang.* 4, ICFP (Aug. 2020), 99:1–99:29. doi:10.1145/3408981

Yizhou Zhang and Andrew C. Myers. 2019. Abstraction-safe effect handlers via tunneling. *Proc. ACM Program. Lang.* 3, POPL (2019), 5:1–5:29. doi:10.1145/3290318

$$\textsc{context-stuck}\ \frac{\mathsf{v} \nearrow (\mathsf{A}, \gamma_{\mathsf{A}})}{\langle \mathbf{ret}\, \mathsf{v} \mid \Box \rangle \nrightarrow (\langle \mathbf{ret}\, \mathsf{A} \mid \Box \rangle, \gamma_{\mathsf{A}}, \varnothing)}\ \textsc{return}$$

$$\textsc{perform}\ \frac{\iota \notin v}{\langle \mathbf{op}\, \mathsf{v} \mid \Box \circ \mathsf{F} \rangle \nrightarrow (\langle \kappa[e] \mid \Box \rangle, \varepsilon, (e, [\kappa], [e \mapsto \mathbf{op}\, \mathsf{v}] \cdot [\kappa \mapsto \mathsf{F}]))}\ \textsc{private}$$

$$\frac{\iota \in v \qquad \mathsf{v} \nearrow (\mathsf{A}, \delta_{\mathsf{A}})}{\langle \mathbf{op}\, \mathsf{v} \mid \Box \circ \mathsf{F} \rangle \nrightarrow (\langle \kappa[\mathbf{op}\, \mathsf{A}\ \mathbf{as}\ e] \mid \Box \rangle, \varepsilon, (e, [\kappa], \delta_{\mathsf{A}} \cdot [e \mapsto \mathbf{op}\, \mathsf{v}] \cdot [\kappa \mapsto \mathsf{F}]))}\ \textsc{public}$$

$$\textsc{forward}\ \frac{}{\langle e \mid \Box \circ \mathsf{F}[r] \rangle \nrightarrow (\langle \kappa :: r[e] \mid \Box \rangle, \varepsilon, (e, \kappa :: r, [\kappa \mapsto \mathsf{F}]))}$$

$$\textsc{open-stuck}\ \frac{\mathsf{v} \nearrow (\mathsf{A}, \gamma_{\mathsf{A}})}{\langle \Box\, \mathsf{v} \mid \mathsf{S} \rangle \nrightarrow (\langle \Box\, \mathsf{A} \mid c \rangle, \gamma_{\mathsf{A}} \cdot [c \mapsto \mathsf{S}], \varnothing)} \qquad \frac{\mathsf{P}(\Box) \nrightarrow (\mathsf{p}, \gamma, \xi)}{\mathsf{P}(\mathsf{S}[\Box]) \nrightarrow (\mathsf{p}(c[\Box]]), \gamma \cdot [c \mapsto \mathsf{S}], \xi)}$$

(a) abstracting copatterns

$$\textsc{context-stuck}\ \frac{\Gamma \Vdash_{\mathsf{c}} \underline{\mathsf{A}} : \tau \triangleright \Gamma_{\underline{\mathsf{A}}}; \Delta_{\underline{\mathsf{A}}}}{\Gamma \Vdash \langle \underline{\mathsf{A}} \mid \Box \rangle : \neg\neg\tau \triangleright \Gamma_{\underline{\mathsf{A}}}; \Delta_{\underline{\mathsf{A}}}}$$

$$\textsc{open-stuck}\ \frac{\Gamma \Vdash_{\mathsf{c}} \underline{\mathsf{A}} : \tau \triangleright \Gamma_{\underline{\mathsf{A}}}; \Delta_{\underline{\mathsf{A}}}}{\Gamma \Vdash \langle \Box\, \underline{\mathsf{A}} \mid d \rangle : \neg(\tau \rightsquigarrow v) \triangleright \Gamma_{\underline{\mathsf{A}}} \cdot [d \mapsto \neg v]; \Delta_{\underline{\mathsf{A}}}} \qquad \frac{\Gamma \Vdash_{\mathsf{v}} \mathsf{A} : \tau \triangleright \Gamma_{\mathsf{A}}}{\Gamma \Vdash \langle \Box\, \mathsf{A} \mid d \rangle : \neg(\tau \rightarrow v) \triangleright \Gamma_{\mathsf{A}} \cdot [d \mapsto \neg v]; \emptyset}$$

(b) copatterns typing rules.

Fig. 24. abstraction process of copatterns.

A Disclosure of effect instances

A.0.1 Abstract values.

$$\mathsf{A}, \mathsf{B} := \langle\rangle \mid \mathsf{n} \mid \mathbb{ff} \mid \mathbb{tt} \mid f \mid \boxed{\iota}$$

$$\frac{}{\iota \nrightarrow (\iota, \varepsilon)}$$

B Concrete Interaction

In concrete interaction, we have named terms…effects and toplevel explanation + concrete example + composing a term and its concrete environment

B.0.1 configurations.

CI confs	$\mathbb{A}, \mathbb{B}$	:=	$\langle \mathbb{J} \dashv\vdash \mathbb{I} \rangle$	$\mathfrak{e}$:=	$\bot$	(no effect)
Env. conf	$\mathbb{J}$	:=	$\langle \mathtt{I}_{\mathbb{I}}; \gamma_{\mathbb{I}}; \mathfrak{e}_{\mathbb{I}}; \boxed{v_{\mathbb{I}}} \rangle$	|	$e e \delta$	(Abstract effect to be forwarded)
Prog. conf	$\mathbb{I}$	:=	$\langle \mathsf{M}; \gamma_{\mathbb{J}}; \mathfrak{e}_{\mathbb{J}}; \boxed{v_{\mathbb{J}}} \rangle$	|	$e\delta$	(Explicit effect to be handled or fowarded)

$$\textsc{Handling}\ \frac{\mathsf{M} \not\mapsto_{\mathsf{eval}}}{\langle \mathtt{I}_a;\ \gamma_p;\ (e, \mathsf{e}, \delta_p);\ \nu_p \dashv\vdash \mathsf{M};\ \gamma_a;\ \delta_a;\ \nu_a\rangle \overset{\mathsf{hdl}}{\mapsto}_{\mathsf{ci}} \langle \mathtt{I}_p;\ \gamma_p;\ \bot;\ \nu_a \dashv\vdash \mathsf{M}\{\delta_p\};\ \gamma_a \cdot \delta_p;\ \bot;\ \nu_p\rangle}$$

$$\textsc{Evaluating}\ \frac{\mathsf{M} \mapsto_{\mathsf{eval}} \mathsf{N}}{\langle \mathtt{I}_a;\ \gamma_p;\ \bot;\ \nu_p \dashv\vdash \mathsf{M};\ \gamma_a;\ \bot;\ \nu_a\rangle \overset{\mathsf{eval}}{\mapsto}_{\mathsf{ci}} \langle \mathtt{I}_a;\ \gamma_p;\ \bot;\ \nu_p \dashv\vdash \mathsf{N};\ \gamma_a \cdot \delta_a;\ \bot;\ \nu_a\rangle}$$

(a) Internal evaluation

$$\textsc{Dualizing}\ \frac{\mathbf{abstract}(\mathsf{Nf}, \mathsf{e}_{\mathsf{X}}, \nu_p) \nearrow\!\!\nearrow (\langle \underline{\mathtt{A}} \mid c\rangle, \gamma_{\mathsf{abs}}, \mathsf{e}'_{\mathsf{X}^\perp}, \nu'_p) \qquad \mathbf{concretize}(\langle \underline{\mathtt{A}} \mid c\rangle^\perp, \gamma_a, \mathsf{e}_{\mathsf{X}}) \searrow\!\!\searrow \mathsf{E}^\perp[\mathtt{t}]}{\langle \mathtt{I}_a; \gamma_p; \mathsf{e}_{\mathsf{X}^\perp};\ \nu_p \dashv\vdash \mathsf{Nf}; \gamma_a; \mathsf{e}_{\mathsf{X}}; \nu_a\rangle \overset{\mathsf{dual}}{\mapsto}_{\mathsf{ci}} \langle \mathtt{I}_p; \gamma_a; \mathsf{e}_{\mathsf{X}}; \nu_a \dashv\vdash \mathsf{E}^\perp[\mathtt{t}]; \gamma_p \cdot \gamma_{\mathsf{abs}}; \mathsf{e}_{\mathsf{X}^\perp} \cdot \mathsf{e}'_{\mathsf{X}^\perp}; \nu_p \cup \nu'_p\rangle}$$

(b) Interaction interface

Fig. 25. Concrete Interaction LTS

$$\textsc{Pure}\ \frac{\mathsf{P} \nearrow\!\!\nearrow (\mathsf{p}, \gamma, \varnothing)}{\mathbf{abstract}(\mathsf{P}(a), \varnothing, \_) \nearrow\!\!\nearrow (\boldsymbol{a}.\mathsf{p}, \gamma, \varnothing)}$$

$$\textsc{Effectful}\ \frac{\langle \mathbf{op}\ \mathsf{v} \mid \mathsf{S}[\Box] \circ \mathsf{F}\rangle \nearrow\!\!\nearrow (\mathsf{p}, \gamma, \xi)}{\mathbf{abstract}(\langle \mathbf{op}\ \mathsf{v} \mid \mathsf{S}[a\ []] \circ \mathsf{F}\rangle, \varnothing, v) \nearrow\!\!\nearrow (\boldsymbol{a}.\mathsf{p}, \gamma, \xi)}\ \textsc{Perform}$$

$$\frac{\langle e \mid \mathsf{S}[\Box] \circ \mathsf{F}[r]\rangle \nearrow\!\!\nearrow (\mathsf{p}, \gamma, \xi)}{\mathbf{abstract}(\langle \mathsf{e} \mid \mathsf{S}[a\ []] \circ \mathsf{F}[r]\rangle, (e, r, \_), \_) \nearrow\!\!\nearrow (\boldsymbol{a}.\mathsf{p}, \gamma, \xi)}\ \textsc{Forward}$$

(a) abstracting normal forms

$$\textsc{Pure/Perform}\ \frac{}{\mathbf{concretize}(\boldsymbol{a}.\mathsf{p}, \gamma, \varnothing) = \mathsf{p}(\gamma(a))}$$

$$\textsc{Forward}\ \frac{\xi = (e, r, \delta) \qquad \mathsf{p} = \langle \kappa :: r[e] \mid \_\rangle \qquad \mathsf{P} = \mathsf{p}\{\delta(e)/e\}}{\mathbf{concretize}(\boldsymbol{a}.\mathsf{p}, \gamma, \xi) = \mathsf{P}(\gamma(a))}$$

(b) concretizing abstract normal forms

Fig. 26. the dualizing process.

C Abstract Interaction

configurations and moves. The information exchanged $\mathbb{J}$ will be captured by a two components; $\gamma : \mathcal{N} \rightharpoonup \text{Terms} \times \text{ECxts}$ (codata + handles to use/probe) and ξ that captures effect information.

$$\mathbb{J} := \langle \mathtt{I};\ \gamma;\ \xi;\ \boxed{\nu}\rangle$$

$$\textsc{Eval}\ \frac{\mathtt{M} \mapsto_{\mathsf{eval}} \mathtt{N}}{\langle \mathtt{M};\ \gamma;\ \nu\rangle \xrightarrow{\mathsf{eval}}_{\mathtt{I}} \langle \mathtt{N};\ \gamma;\ \nu\rangle}$$

(a) Internal evaluation

$$\textsc{Px}\ \frac{\mathbf{abstract}(\mathtt{Nf}) \nearrow\!\!\!\nearrow (\boldsymbol{a}.\mathtt{p},\ \gamma_{\mathtt{p}}) \qquad \rho = \mathrm{supp}(\mathtt{p})}{\langle \mathtt{Nf};\ \gamma;\ \nu\rangle \xrightarrow{\boldsymbol{a}.\mathtt{p}^{\oplus}}_{\mathtt{I}} \langle \mathtt{I};\ \gamma{\cdot}\gamma_{\mathtt{p}};\ \nu{\cdot}\rho\rangle} \qquad \frac{}{\langle \mathtt{I};\ \gamma;\ \nu\rangle \xrightarrow{\boldsymbol{a}.\mathtt{p}^{\ominus}}_{\mathtt{I}} \langle \mathtt{p}(\gamma(a));\ \gamma;\ \nu\rangle}\ \textsc{Ox}$$

(b) Pure moves

$$\textsc{Forward}\ \frac{\mathbb{I} \overset{\mathsf{s}}{\Rightarrow}_{\mathbf{fwd}} \mathbb{J}}{\mathbb{I} \overset{\mathsf{s}}{\Rightarrow}_{\mathtt{I}} \mathbb{J}}$$

(c) Effect forwarding sequences

$$\textsc{Forward}\ \frac{\mathtt{M} \mapsto_{\mathsf{fwd}} \mathtt{N}}{\langle \mathtt{M};\ \gamma;\ \square\ \xi;\ \nu\rangle \rightarrow_{\mathbf{fwd}} \langle \mathtt{N};\ \gamma;\ \square\ \xi;\ \nu\rangle}$$

$$\textsc{PHandle}\ \frac{\mathtt{M} \mapsto_{\mathsf{eval}}}{\langle \mathtt{M};\ \gamma;\ !\xi;\ \nu\rangle \xrightarrow{hdl^{\oplus}}_{\mathbf{fwd}} \langle \mathtt{M}\{\xi\};\ \gamma;\ \nu\rangle} \qquad \textsc{OHandle}\ \frac{}{\langle \mathtt{I};\ \gamma;\ ?\xi;\ \nu\rangle \xrightarrow{hdl^{\ominus}}_{\mathbf{fwd}} \langle \mathtt{I};\ \gamma{\cdot}\xi;\ \nu\rangle}$$

(d) Internal effect manipulation

$$\textsc{Peff}\ \frac{\mathbf{abstract}(\mathtt{Nf},\ \xi,\ \nu) \nearrow\!\!\!\nearrow (\boldsymbol{a}.\mathtt{p},\ \gamma_{\mathtt{p}},\ \xi_{\mathtt{p}}) \qquad \rho = \mathrm{supp}(\mathtt{p})}{\langle \mathtt{Nf};\ \gamma;\ \square\ \xi;\ \nu\rangle \xrightarrow{\boldsymbol{a}.\mathtt{p}^{\oplus}}_{\mathbf{fwd}} \langle \mathtt{I};\ \gamma{\cdot}\gamma_{\mathtt{p}};\ \square\ \xi{\cdot}\xi_{\mathtt{p}};\ \nu{\cdot}\rho\rangle}$$

$$\frac{\mathbf{concretize}(\boldsymbol{a}.\mathtt{p},\ \gamma,\ \square\ \xi) = (\mathtt{M},\ \delta)}{\langle \mathtt{I};\ \gamma;\ \square\ \xi;\ \nu\rangle \xrightarrow{\boldsymbol{a}.\mathtt{p}^{\ominus}}_{\mathbf{fwd}} \langle \mathtt{M};\ \gamma;\ \xi;\ \nu\rangle}\ \textsc{Oeff}$$

(e) Effectful moves

Fig. 27. transitions of the Abstract Interaction LTS

$$\begin{array}{rcll} \mathbb{G}, \mathbb{H} & := & \langle \mathtt{M}; \mathbb{J}\rangle & \text{(Active State)} \\ & | & \mathbb{J} & \text{(Passive State)} \end{array} \qquad \begin{array}{rcll} \xi & := & \varnothing \mid (?e,\ r,\ \delta) \mid \boxed{(?\mathrm{e},\ e,\ r,\ \delta)} & \text{(may perform/forward effect)} \\ & | & (!\mathrm{e},\ e,\ r,\ \delta) & \text{(must forward effect)} \end{array}$$

The moves are given by the previously described *named abstract computations*.

$$\mathbf{m} := \underline{\mathbf{a}} \mid \underline{\mathbf{a}}^{\perp}$$

C.0.1 Definition of $\mathcal{L}_{\mathsf{AI}}$.

D Well-bracketing LTS

D.0.1 configurations and moves. $\mathcal{L}_{wb}(\mathsf{L}_{\mathrm{op}})$ is defined as $(\mathcal{W}; \mathcal{M}; \xrightarrow{\mathbf{m}}_{wb})$ with $\xrightarrow{\mathbf{m}}_{wb}$ defined in Figure 23 and $\mathcal{W}$ defined in the following subsection D.0.1.

Call Stack	π	::=	$[] \mid c :: \sigma \mid \kappa :: \sigma$
Frames	f	::=	$[] \mid \kappa :: f$
Current Frame	η	::=	$\varnothing$
		$\mid$	(p, e, f) where $p \in \{\ominus, \oplus\}$
Captured Cont.	ϕ	:=	$\wp(\{\ominus, \oplus\} \times F)$
WB configurations	$\mathcal{W}$	:=	$\Pi \times F \times K$

where Π, F, Φ and K denote the set of all call stacks, the set of frames, the set of current effectful frames and the set of all captured continuations, respectively.

For uniformity of treatment, we will consider $\varnothing \in \Phi \times Names_{\perp_e}$ and write $\varnothing := [], \underline{\perp_e}$.

D.0.2 transitions.

$$\begin{array}{rcl}
\mathbf{forward}(\varnothing, \boldsymbol{a}.\ \langle e \mid _[\square] \circ \varkappa \rangle^{\ominus}) & := & (\ominus, e, [\,]) \\
\mathbf{forward}(\eta, \mathbf{o}) & := & \eta \\
\\
\mathbf{call/ret}(a :: \sigma, _, \boldsymbol{a}.\mathrm{p}^{\ominus}) & := & \sigma \\
\mathbf{call/ret}(\sigma, \phi, \boldsymbol{\kappa}.\ \langle \underline{\mathsf{B}} \mid d[\square] \circ _ \rangle^{\ominus}) & := & f +\!\!+ \sigma \quad \text{when } (\circledast, \kappa :: f) \in \phi \\
\mathbf{call/ret}(\sigma, \phi, \boldsymbol{\kappa}.\ \langle \mathbf{ret}\,\mathsf{B} \mid d[\square] \rangle^{\ominus}) & := & f +\!\!+ \sigma \quad \text{when } (\ominus, \kappa :: f) \in \phi
\end{array}$$

(a) opponent moves

$$\begin{array}{rcl}
\mathbf{forward}(\varnothing, \boldsymbol{a}.\ \langle e \mid _[\square] \circ \kappa \rangle^{\oplus}) & := & (\oplus, e, [\kappa]) \\
\mathbf{forward}((p, e, f), \boldsymbol{a}.\ \langle e \mid _[\square] \circ \kappa :: _ \rangle^{\oplus}) & := & (p, e, f +\!\!+ [\kappa]) \\
\mathbf{forward}(\varnothing, \mathbf{p}) & := & \varnothing \\
\\
\mathbf{call/ret}(\sigma, _, \boldsymbol{\varkappa}.\ \langle \underline{\mathsf{A}} \mid c[\square] \circ _ \rangle^{\oplus}) & := & c :: \sigma \\
\mathbf{call/ret}(\sigma, _, \boldsymbol{g}.\langle \square\,\mathsf{A} \mid c \rangle^{\oplus}) & := & c :: \sigma \\
\mathbf{call/ret}(\sigma, _, \mathbf{p}) & := & \sigma
\end{array}$$

(b) proponent moves

$$\frac{\eta' = \mathbf{forward}(\eta, \mathbf{m}) \qquad \sigma' = \mathbf{call/ret}(\sigma, \phi, \mathbf{m})}{\langle \sigma \mid \eta \mid \phi \rangle \xrightarrow{\mathbf{m}}_{wb} \langle \sigma' \mid \eta' \mid \phi \rangle}$$

$$\frac{}{\langle \sigma \mid (p, e, f) \mid \phi \rangle \xrightarrow{hdl^{\ominus}}_{wb} \langle \sigma \mid \varnothing \mid \phi \cup \{f\} \rangle} \qquad \frac{}{\langle \sigma \mid (p, e, f) \mid \phi \rangle \xrightarrow{hdl^{\oplus}}_{wb} \langle \sigma \mid \varnothing \mid \phi \rangle}$$

(c) transitions

Fig. 28. transitions

E Concrete Interaction

$\mathcal{L}_{\mathrm{ci}}$ can be seen as an abstract machine that performs a variant of the linear head reduction, which is known to correspond to interaction in game semantics [Danos et al. 1996], (having a global environment γ, it is similar to the Milner Abstract Machine [Accattoli et al. 2014]), computing the interaction between a program configuration and an environment configuration that are *composable*.

The purpose of the concrete interaction is to *evaluate* a term inside an environment while exhibiting the *interaction* between the two that gets obscured by plain *syntax substitution*. To this effect, the program being evaluated and its environment will remain decoupled, but will each carry an *information* component that will be threaded through their interaction. This component keeps track of the history of the interaction and it consists of a map

between the names and the *concrete* code of codata that each party has disclosed and rendered accessible to the other.

E.1 Concrete interaction LTS

The concrete interaction describes how a term and a complementary *concrete* environment interact by disclosing information and alternate between passive and active states. It emphasizes the interaction points between the two by highlighting the information they disclose to each other. At every stage of the interaction, the passive player is described by an accumulative component tracking its past contribution to the interaction, whereas the active player description contains in addition an interactive term representing the current *active* computation.

It is a polarized abstract machine whose configurations involve a *passive* and a *decoupled* complementary *passive* $\mathcal{L}_{\mathsf{AI}}$ configurations (*cf.* 4.2).

We formalize this notion of complementarity by the following definition.

Definition E.1. **(Complementarity)** Let $\mathbb{P} = \langle \mathtt{I}_a ; \gamma_a ; \xi_a \rangle$, $\mathbb{A} = \langle \mathtt{M} ; \gamma_p ; \delta_p \rangle \in \mathcal{A}$ such that $\mathtt{M} = \langle \mathtt{t} \mid \mathtt{S} \circ \mathtt{T} \rangle$ and $c_{\mathtt{f}} \in \mathcal{C}$. We say that $\mathbb{P}$ and $\mathbb{A}$ are $c_{\mathtt{f}}$-complementary and write $\mathbb{P} \dashv\vdash_c \mathbb{A}$ when:

- $\mathtt{I}_a \cap \mathtt{I}_p = \emptyset$ and $\mathrm{dom}(\gamma_a) \cup \mathrm{dom}(\gamma_p) = \emptyset$
- $\exists y \in \{a, p\}.\, \exists d \in \mathrm{dom}(\gamma_y).\ c_{\mathtt{f}} \in \mathrm{supp}(\gamma_y(d))$
- $\forall y \in \{a, p\}.\, \forall d \in \mathrm{dom}(\gamma_y).\ (\mathrm{supp}(\gamma_y(d)) \setminus \{c_{\mathtt{f}}\}) \subseteq \mathrm{dom}(\gamma_{y^\perp})$
- $\mathrm{supp}(\mathrm{codom}(\xi_a)) \subseteq \mathrm{dom}(\gamma_p)$ and $\mathrm{supp}(\mathrm{codom}(\delta_p)) \subseteq \mathrm{dom}(\gamma_a)$
- $\mathrm{supp}(\mathtt{t}), \mathrm{supp}(\mathtt{S}) \subseteq \mathrm{dom}(\gamma_a)$ and $\mathrm{supp}(\mathtt{T}) \subseteq \mathrm{dom}(\gamma_a) \cup \mathrm{dom}(\xi_a)$.
- There exists a type configuration $\mathbb{T} = \langle \Gamma_o ; \Delta_o \mid \Gamma_p ; \Delta_p \rangle \in \mathcal{T}$ and a type τ *s.t.* either: $\mathbb{P} \Cat \langle \Gamma_o, c_{\mathtt{f}} : \neg\tau ; \Delta_o \mid \Gamma_p ; \Delta_p \rangle$ and $\mathbb{A} \Cat \mathbb{T}^\perp$ or $\mathbb{P} \Cat \mathbb{T}$ and $\mathbb{A} \Cat \langle \Gamma_p, c_{\mathtt{f}} : \neg\tau ; \Delta_p \mid \Gamma_o ; \Delta_o \rangle$.

We will sometimes omit the index c on the relation $\dashv\vdash_c$ in contexts that do not require us to be explicit and just write $\mathbb{P} \dashv\vdash \mathbb{A}$.
In the following, we will use the notation $\langle \mathbb{P} \dashv\vdash \mathbb{A} \rangle$ to denote the couple $(\mathbb{P}, \mathbb{A}) \in \mathcal{A}^2$ satisfying $\mathbb{P} \dashv\vdash \mathbb{A}$.

E.1.1 configurations and transitions.

$$\text{configurations} \qquad \mathrm{Confs}_{\mathrm{ci}} \quad \ni \quad \mathbb{C}, \mathbb{D} \quad ::= \quad \langle \mathbb{P} \dashv\vdash \mathbb{A} \rangle$$

The CI transitions are given by $\mapsto_{\mathrm{ci}} := \overset{\mathsf{eval}}{\mapsto}_{\mathrm{ci}} \cup \overset{\mathsf{dual}}{\mapsto}_{\mathrm{ci}} \cup \overset{\mathsf{hdl}}{\mapsto}_{\mathrm{ci}}$ and are defined in fig. 29. Similar to $\mathcal{L}_{\mathsf{AI}}$, $\overset{\mathsf{eval}}{\mapsto}_{\mathrm{ci}}$ corresponds to evaluating a term through $\mapsto_{\mathrm{eval}}$ down to an normal form, while $\overset{\mathsf{hdl}}{\mapsto}_{\mathrm{ci}}$ marks the end of effect propagation whereby the *active* player gets to capture the fragments of the *forward stack* that belong to the environment.

On the other hand, $\overset{\mathsf{dual}}{\mapsto}_{\mathrm{ci}}$ acts on configurations in normal form by dualizing the perspective of the computation; i.e taking the perspective of the environment.

The dualizing process. It is defined as the composition of two processes **abstract** and **concretize**; The polarity of the *abstract normal form* $\boldsymbol{a}$.p produced by **abstract** is switched in order to take the environment's perspective, then the concretization is carried out on $\boldsymbol{a}$.p as explained above.

EVALUATING

$$\frac{\mathtt{M} \mapsto_{\mathsf{op}} \mathtt{N}}{\langle \mathtt{I}_p;\ \gamma_p \dashv\vdash \mathtt{M};\ \gamma_a\rangle \overset{\mathsf{eval}}{\mapsto}_{\mathsf{ci}} \langle \mathtt{I}_p;\ \gamma_p \dashv\vdash \mathtt{N};\ \gamma_a\rangle}$$

HANDLING

$$\frac{}{\langle \mathtt{I}_a;\ \gamma_p;\ ?(\_,\ \_,\ \delta_p) \dashv\vdash \mathtt{M};\ \gamma_a;\ \delta_a\rangle \overset{\mathsf{hdl}}{\mapsto}_{\mathsf{ci}} \langle \mathtt{I}_a;\ \gamma_p{\cdot}\delta_p \dashv\vdash \mathtt{M};\ \gamma_a\rangle}$$

(a) Internal evaluation

DUALIZING

$$\frac{\mathbf{abstract}(\mathtt{Nf},\ \delta_p) \nrightarrow (\boldsymbol{a}.\mathsf{p},\ \gamma_{\mathsf{abs}},\ \xi_{\mathsf{abs}}) \qquad \mathbf{concretize}(\boldsymbol{a}.\mathsf{p},\ \gamma_p,\ \xi_p) = \mathtt{M}}{\langle \mathtt{I}_a;\ \gamma_p;\ \xi_p \dashv\vdash \mathtt{Nf};\ \gamma_a;\ \delta_a\rangle \overset{\mathsf{dual}}{\mapsto}_{\mathsf{ci}} \langle \mathtt{I}_p;\ \gamma_a{\cdot}\gamma_{\mathsf{abs}};\ \xi_a[\xi_{\mathsf{abs}}] \dashv\vdash \mathtt{M};\ \gamma_p;\ \xi_p\rangle}$$

(b) Interaction interface

Fig. 29. transitions of the Concrete Interaction LTS

Abstracting a *co-pattern* into the shapes $\langle r[e] \mid \mathtt{S}[\Box]\rangle$ or $\langle r[e] \mid \Box\rangle$; that is an abstract effect e with its delimiting abstract evaluation stack r is a gradual process. The fragments κ making up r cannot be computed from the syntax alone, an interaction has to force the environment to perform the necessary "abstract" $\mapsto_{\mathsf{fwd}}$ reductions on stuck terms such as $\langle \mathbf{op}\ \mathtt{v} \mid \mathtt{S}[\kappa\ []] \circ \mathtt{T}\rangle$ or $\langle e \mid c\ [] \circ \mathtt{T}\rangle$ in oder to progressively compute r. The exchanged delimited continuations are gradually put in δ, whose disclosure and concretization is delayed until the effect has been effectively handled (cf. handling rule in Fig. 29).

E.2 Properties of the CI LTS

E.2.1 *Correctness.*

LEMMA E.2 (COMPLEMENTARITY PRESERVATION). *Let* $\mathbb{P} \in \mathsf{Confs}^{\ominus}_{\mathsf{ci}}$, $\mathbb{A} \in \mathsf{Confs}^{\oplus}_{\mathsf{ci}}$ *such that* $\mathbb{P} \dashv\vdash \mathbb{A}$. *If* $\langle \mathbb{P} \dashv\vdash \mathbb{A}\rangle \mapsto_{\mathsf{ci}} \langle \mathbb{P}' \dashv\vdash \mathbb{A}'\rangle$, *then* $\mathbb{P}' \dashv\vdash \mathbb{A}'$.

PROOF. Let's write $\mathbb{A} = \langle \mathtt{M};\ \gamma_p;\ \delta_p\rangle$, $\mathbb{P} = \langle \mathtt{I}_p;\ \gamma_a;\ \xi_a\rangle$ and $\mathtt{M} = \langle \mathtt{t} \mid \mathtt{S} \circ \mathtt{T}\rangle$.

Case $\langle \mathbb{A} \dashv\vdash \mathbb{P}\rangle \overset{\mathsf{op}}{\mapsto}_{\mathsf{ci}} \langle \mathbb{A}' \dashv\vdash \mathbb{P}'\rangle$.
Then $\xi_a = \varnothing$ and $\delta_p = \varnothing$ and there exists $\mathtt{N} = \langle \mathtt{u} \mid \mathtt{S}' \circ \mathtt{T}'\rangle$ and $\mathtt{I}'_a$ s.t. $\mathtt{M} \mapsto_{\mathsf{op}} \mathtt{N}$, $\mathbb{A}' = \langle \mathtt{N};\ \gamma_p;\ \varnothing\rangle$ and $\mathbb{P}' = \mathbb{P}$.
Subcase $\mathtt{I}'_a \cap \mathtt{I}_p = \emptyset$.
It is immediate that $\mathsf{supp}(\mathtt{T}') \subseteq \mathsf{supp}(\mathtt{M}) \subseteq \mathsf{dom}(\gamma_a) \cup \mathsf{dom}(\delta_p)$. We have $\mathsf{supp}(\mathtt{u}), \mathsf{supp}(\mathtt{S}') \subseteq \mathsf{supp}(\mathtt{t}) \cup \mathsf{supp}(\mathtt{S})$, then by hypothesis we get $\mathsf{supp}(\mathtt{u}), \mathsf{supp}(\mathtt{S}') \subseteq \mathsf{supp}(\mathsf{dom}(\gamma_a))$.
W.l.o.g. suppose $\exists\, \mathbb{T} = \langle \Gamma_o; \emptyset \mid \Gamma_p; \emptyset\rangle$ and $\subseteq \mathsf{dom}(\gamma_a)$. $\mathbb{P} \between \langle \Gamma_o; \emptyset \mid \Gamma_p{\cdot}[c \mapsto \neg\tau]; \emptyset\rangle$ and $\mathbb{A} \between \mathbb{T}^{\perp}$, then we have $\mathtt{I}_a; \Gamma_o \vdash_{\mathsf{c}} \mathtt{M}$ and from lemma **??**, we have $\mathtt{I}'_a; \Gamma_o \vdash_{\mathsf{c}} \mathtt{N}$, thus $\mathbb{A}' \between \mathbb{T}^{\perp}$. The remaining complementarity conditions are trivially satisfied.

Case $\langle \mathbb{A} \dashv\vdash \mathbb{P}\rangle \overset{\mathsf{hdl}}{\mapsto}_{\mathsf{ci}} \langle \mathbb{A}' \dashv\vdash \mathbb{P}'\rangle$.
Then $\mathbb{A}' = \langle \mathtt{M};\ \gamma_p;\ \varnothing\rangle$ and $\mathbb{P}' = \langle \mathtt{I}_p;\ \gamma_a{\cdot}\delta_a\ \varnothing\rangle$ where $\xi_a = (\_, \_, \delta_a)$.
Since $\mathsf{supp}(\mathtt{M}) \subseteq \mathsf{dom}(\gamma_a) \cup \mathsf{dom}(\delta_a)$ then $\mathsf{supp}(\mathtt{M}) \subseteq \mathsf{dom}(\gamma_a{\cdot}\delta_a)$.

W.l.o.g. let $\mathbb{T} = \langle \Gamma_o; \Delta_o \mid \Gamma_p; \Delta_p \rangle$ and suppose $\mathbb{P} \between \langle \Gamma_o; \Delta_o \mid \Gamma_p \cdot [c \mapsto \neg\tau]; \Delta_p \rangle^\perp$ and $\mathbb{A} \between \mathbb{T}$., Taking $\mathbb{S} = \langle \Gamma_o \cdot \Delta_o; \emptyset \mid \Gamma_p; \emptyset \rangle$, since $\mathtt{I}_a; \Gamma_o \cdot \Delta_o \vdash_{\mathtt{c}} \mathtt{M}$, we can verify that $\mathbb{A}' \between \mathbb{S}$ and $\mathbb{P} \between \langle \Gamma_o \cdot \Delta_o; \emptyset \mid \Gamma_p \cdot [c \mapsto \neg\tau]; \emptyset \rangle^\perp$.

Case $\langle \mathbb{A} \dashv\vdash \mathbb{P} \rangle \overset{\text{dual}}{\mapsto_{\text{ci}}} \langle \mathbb{A}' \dashv\vdash \mathbb{P}' \rangle$.

By writing $\textbf{abstract}(\mathtt{M}, \delta_p) \nearrow\!\!\!\nearrow (\boldsymbol{a}.\mathrm{p}, \gamma_{\text{abs}}, \xi_p)$ and $\textbf{concretize}(\boldsymbol{a}.\mathrm{p}, \gamma_a; \xi_a) = (\mathtt{N}, \delta_a)$ with $\mathtt{N} = \langle\!\langle \mathtt{u} \mid \mathtt{S}' \circ \mathtt{T}' \rangle\!\rangle$, we have $\mathbb{A}' = \langle \mathtt{N}; \gamma_a; \delta_a \rangle$ and $\mathbb{P}' = \langle \mathtt{I}_a; \gamma_p \cdot \gamma_{\text{abs}}; \delta_p \cdot \xi \rangle$.

We have $\text{supp}(\text{codom}(\delta_a)) = \text{supp}(\text{codom}(\xi_a))$ and $\text{supp}(\text{codom}(\xi)) \subseteq \text{supp}(\mathtt{t})$, thus by hypothesis, we have $\text{supp}(\text{codom}(\delta_a)) \subseteq \text{dom}(\gamma_p) \subseteq \text{dom}(\gamma_p \cdot \gamma_{\text{abs}})$, and we also have $\text{supp}(\text{codom}(\xi \cdot \delta_p)) \subseteq \text{supp}(\gamma_a)$.

W.l.o.g. let $\mathbb{T} = \langle \Gamma_o; \Delta_o \mid \Gamma_p; \Delta_p \rangle$ and suppose $\mathbb{P} \between \langle \Gamma_o; \Delta_o \mid \Gamma_p \cdot [c \mapsto \neg\tau]; \Delta_p \rangle^\perp$ and $\mathbb{A} \between \mathbb{T}$., Since $\mathtt{I}_a; \Gamma_o \cdot \Delta_o \vdash_{\mathtt{c}} \mathtt{M}$ and $a \in \text{dom}(\Gamma_o)$ (because $a \in \text{supp}(\mathtt{t}) \cup \text{supp}(\mathtt{S})$), then there exists τ s.t. $\Gamma_o(a) = \tau$.

By writing $\Gamma_o \cdot \Delta_o \Vdash \mathrm{p} : \neg\tau \triangleright \Gamma_{\mathrm{p}}; \Delta_{\mathrm{p}}$ and taking $\mathbb{S} = \langle \Gamma_p \cdot \Gamma_{\mathrm{p}}; \Delta_p \cdot \Delta_{\mathrm{p}} \mid \Gamma_o; \Delta_o \rangle$, it is easy to verify that $\mathbb{A}' \between \mathbb{S}$ and $\mathbb{P}' \between \langle \Gamma_o; \Delta_o \mid \Gamma_p \cdot \Gamma_{\mathrm{p}} \cdot [c \mapsto \neg\tau]; \Delta_p \cdot \Delta_{\mathrm{p}} \rangle$.

□

E.2.2 Concrete observation. If an observation amounts to evaluating an observable value, an environment configuration can only do this after a switch of perspective, i.e through a $\overset{\text{dual}}{\mapsto_{\text{ci}}}$ transition. For this reason we will define the obsrevation transition $\Downarrow_{\text{ci}}$ a closure on $\overset{\text{dual}}{\mapsto_{\text{ci}}}$ (up-to $\overset{\text{eval}}{\mapsto_{\text{ci}}}$ and $\overset{\text{hdl}}{\mapsto_{\text{ci}}}$) such that $\mathbb{C} \Downarrow_{\text{ci}} \mathbb{D}$ means $\mathbb{C} \mapsto_{\text{ci}}^* \mathbb{D}$ where $\mathbb{C}$ and $\mathbb{D}$ are of opposite perspective.

Definition E.3. Taking $\mapsto\!\!\!\!=_{\text{dual}}$ to be $(\overset{\text{hdl}}{\mapsto_{\text{ci}}} \cup \overset{\text{eval}}{\mapsto_{\text{ci}}})^* \overset{\text{dual}}{\mapsto_{\text{ci}}} (\overset{\text{hdl}}{\mapsto_{\text{ci}}} \cup \overset{\text{eval}}{\mapsto_{\text{ci}}})^*$ we define:

$$\mathbb{C} \Downarrow_{\text{ci}} \mathbb{D} \iff \exists k \in \mathbb{N}. \mathbb{C} \overset{2*k+1}{\mapsto\!\!\!\!=_{\text{dual}}} \mathbb{D} \wedge \mathbb{D} \not\mapsto_{\text{ci}}$$

where, for $n \in \mathbb{N}$, $\overset{\text{n}}{\mapsto\!\!\!\!=_{\text{dual}}} := \overbrace{\mapsto\!\!\!\!=_{\text{dual}} \cdots \mapsto\!\!\!\!=_{\text{dual}}}^{\text{n times}}$.

E.2.3 Relating $\mathcal{L}_{\text{AI}}$ and $\mathcal{L}_{\text{ci}}$. We observe that, by definition, the *interaction* between a program and its environment given in the form of *concrete interaction* hides the *parallel abstract composition* of the underlying complementary passive and active $\mathcal{L}_{\text{AI}}$ configurations.

The following proposition captures how the $\mapsto_{\text{ci}}$ transitions, can be equivalently re-expressed in terms of *complementarity* and and the transition $\rightarrow_{\text{I}}$.

Proposition E.4 (Parallel composition).

Evaluating

$$\frac{\mathbb{J} \dashv\vdash \mathbb{I} \qquad \mathbb{I} \overset{\text{eval}}{\longrightarrow_{\text{I}}} \mathbb{J}}{\langle \mathbb{J} \dashv\vdash \mathbb{I} \rangle \overset{\text{eval}}{\mapsto_{\text{ci}}} \langle \mathbb{J} \dashv\vdash \mathbb{J} \rangle}$$

Handling

$$\frac{\mathbb{J} \dashv\vdash \mathbb{I} \qquad \mathbb{J} \overset{hdl^{\ominus}}{\longrightarrow_{\text{I}}} \mathbb{K} \qquad \mathbb{I} \overset{hdl^{\oplus}}{\longrightarrow_{\text{I}}} \mathbb{J}}{\langle \mathbb{J} \dashv\vdash \mathbb{I} \rangle \overset{\text{hdl}}{\mapsto_{\text{ci}}} \langle \mathbb{J} \dashv\vdash \mathbb{J} \rangle}$$

Dualizing

$$\frac{\mathbb{J} \dashv\vdash \mathbb{I} \qquad \mathbb{J} \overset{\mathbf{m}}{\longrightarrow_{\text{I}}} \mathbb{J} \qquad \mathbb{I} \overset{\mathbf{m}^{\perp}}{\longrightarrow_{\text{I}}} \mathbb{K}}{\langle \mathbb{J} \dashv\vdash \mathbb{I} \rangle \overset{\text{dual}}{\mapsto_{\text{ci}}} \langle \mathbb{K} \dashv\vdash \mathbb{J} \rangle}$$

Corollary E.5. *Given* $\mathbb{I} \in \text{Confs}_{\text{act}}$ *and* $\mathbb{J} \in \text{Confs}_{\text{pas}}$ *such that* $\mathbb{I} \dashv\vdash \mathbb{J}$*, we have:*

$$\langle \mathbb{J} \dashv\vdash \mathbb{I} \rangle \mapsto_{\text{ci}}^* \langle \mathbb{J}_0 \dashv\vdash \mathbb{J}_1 \rangle \iff \exists \mathrm{t} \in \mathbf{Tr}_{\text{AI}},\ i \in \{0, 1\}.\ \mathbb{J} \overset{\mathrm{t}}{\Rightarrow_{\text{I}}} \mathbb{J}_0 \wedge \mathbb{I} \overset{\mathrm{t}^{\perp}}{\Longrightarrow_{\text{I}}} \mathbb{J}_1$$

$$\textit{where}\ \overset{\mathbf{m}}{\Longrightarrow}_{\mathsf{I}} := \xrightarrow{\textit{hdl}}_{\mathsf{I}} \xrightarrow{\text{eval}}_{\mathsf{I}}{}^{*} \xrightarrow{\mathbf{m}}_{\mathsf{I}} \xrightarrow{\textit{hdl}}_{\mathsf{I}} \xrightarrow{\text{eval}}_{\mathsf{I}}{}^{*}$$

F Adequacy

Now that all the necessary definitions are available, we will state the main intermediate lemmas for proving adequacy: lemmas F.1 and F.2, and we will prove them in the following sections G and H, respectively.

Lemma F.1 (Bisimilarity). *Taking an interactive term* $\mathtt{I}_1; \Gamma \vdash_{\mathrm{c}} \mathtt{M}$ *and a compatible name assignment* $\mathtt{I}_1'; d : \neg\mathbb{1} \vdash \gamma : \Gamma$*, then:*

$$\langle \mathtt{I}_1',\ \gamma;\ \bot \dashv\vdash \mathtt{M};\ \varepsilon,\ \bot\rangle \Downarrow_{\mathrm{ci}} \langle \mathtt{I}_2;\ \gamma_2;\ \bot \dashv\vdash d[\mathbf{ret}\,\langle\rangle];\ \gamma_1;\ \bot\rangle$$
$$\Longleftrightarrow$$
$$\mathtt{M}\{\gamma\} \Downarrow_{\mathrm{op}} d[\mathbf{ret}\,\langle\rangle]$$

Lemma F.2 (Full observation). *Taking an interactive term* $\mathtt{I}; \Gamma \vdash_{\mathrm{c}} \mathtt{M}$ *and a compatible name assignment* $\mathtt{I}'; d : \neg\mathbb{1} \vdash \gamma : \Gamma$*, then by writing* $\mathbb{G} = \langle \mathbb{I} \parallel \mathbb{T} \parallel \mathbb{W}\rangle$ *and* $\mathbb{H} = \langle \mathbb{J} \parallel \mathbb{S} \parallel \mathbb{U}\rangle$ *for their respective* ogs *embeddings:*

$$\exists \mathbb{K}, \gamma'.\ \langle \mathbb{J} \dashv\vdash \mathbb{I}\rangle \Downarrow_{\mathrm{ci}} \langle \mathbb{K} \dashv\vdash d[\mathbf{ret}\,\langle\rangle];\ \gamma';\ \bot\rangle$$
$$\Longleftrightarrow$$
$$\exists \mathrm{t} \in \mathbf{CTr}_{\mathrm{ogs}}(\mathbb{G}).\ \mathrm{t}^{\perp}\, \boldsymbol{d}.\langle \mathbf{ret}\,\langle\rangle \mid \Box\rangle^{\oplus} \in \mathbf{CTr}_{\mathrm{ogs}}(\mathbb{H})$$

Theorem F.3 (Adequacy). *Given an interactive term* $\mathtt{I}; \Gamma \vdash_{\mathrm{c}} \mathtt{M}$ *and a compatible name assignment* $\mathtt{I}'; d : \neg\mathbb{1} \vdash \gamma : \Gamma$*, then by writing* $\mathbb{P}$ *and* $\mathbb{E}$ *for their respective* ogs *embeddings, we have:*

$$(\mathtt{M}\{\gamma\} \Downarrow_{\mathrm{op}} d[\,\mathbf{ret}\,\langle\rangle]$$
$$\Longleftrightarrow$$
$$\exists \mathrm{t} \in \mathbf{CTr}_{\mathrm{ogs}}(\mathbb{P}).\ \mathrm{t}^{\perp}\, \boldsymbol{d}.\langle \mathbf{ret}\,\langle\rangle \mid \Box\rangle^{\oplus} \in \mathbf{CTr}_{\mathrm{ogs}}(\mathbb{E})$$

Proof. Follows from the two preceding lemmas. □

G Bisimilarity (proof of lemma F.1)

$(\mathcal{L}_{\text{ci}}, \mapsto_{\text{ci}})$ and $(\Lambda_{\text{eff}}, \mapsto_{\text{op}})$ are weakly bisimilar

We start by stating the main result of this section; that is modulo $\overset{\text{dual}}{\mapsto}_{\text{ci}}$ and $\overset{\text{hdl}}{\mapsto}_{\text{ci}}$ reductions, the transition systems $(\mathcal{L}_{\text{ci}}, \mapsto_{\text{ci}})$ and $(\Lambda_{\text{eff}}, \mapsto_{\text{op}})$ are realted by a weak bisimulation. Afterwards we will introduce the necessary definitions and lemmas in order to prove it.

The proof relies on a notion of *telescoped substitution* to avoid cycles in the concatenation of mappings coming from the composition of two interactive configurations. This enforces the absence of livelocks in the interaction.

Definition G.1. A *telescoped substitution* δ is a substitution seen as a stack of mappings $[a_0 \mapsto \mathsf{v}_0, \dots, a_k \mapsto \mathsf{v}_k]$ such that for all $i \in \{1, \dots, k\}$, we have $\text{supp}(\mathsf{v}_i) \cap \{a_i, \dots, a_k\} = \varnothing$.

Proposition G.2. *Given a* telescoped substitution $[a_0 \mapsto \mathsf{v}_0, \dots, a_k \mapsto \mathsf{v}_k]$, *for all* $i \in \{1, \dots, k\}$, *we have* $\text{supp}(\mathsf{v}_i) \subseteq \{a_0, \dots, a_i\}$.

A telescoped substitution δ can be transformed into a substitution δ^* such that for any $a \in \text{dom}(\delta)$ $\text{supp}(\delta(a)) \subseteq \text{supp}(\mathsf{v}_0)$.

Definition G.3. Taking a telescoped substitution δ and $k \in \mathbb{N}^*$, we define the iterated telescoped substitution δ^i as:

$$\delta^1 := \delta \qquad \delta^{i+1} := \Pi_{\mathsf{n} \in \text{dom}(\delta)} [\mathsf{n} \mapsto \delta(\mathsf{n})\{\delta^i\}]$$

We then define δ^* as δ^k with k the size of the domain of δ. Then if $\Gamma \cdot \Delta \vdash \delta : \Gamma$ and $\text{dom}(\Delta) = \text{supp}(\mathsf{v}_0)$, we have $\Delta \vdash \delta^* : \Gamma$.

Lemma G.4. *Taking a telescoped substitution* δ *and* $a \in \text{dom}(\delta)$, *then there exists a name* $a \in \text{dom}(\delta)$ *and a natural number* $k \in \mathbb{N}^*$ *such that* $\text{supp}(\delta^k(a)) \cap \text{dom}(\delta) = \emptyset$.

We define a bisimulation between $\mapsto_{\text{ci}}$ and $\mapsto_{\text{op}}$ by collapsing $\mathcal{L}_{\text{ci}}$ configurations into operational ones.

Lemma G.5. *Let* $\langle \mathbb{E} \dashv \mathbb{P} \rangle \in \text{Confs}_{\text{ci}}$ *such that* $\mathbb{E} \dashv\!\vdash_d \mathbb{P}$. *If we write* $\mathbb{E} = \langle \mathtt{I}_p;\ \gamma_p;\ \xi_p \rangle$ *and* $\mathbb{P} = \langle \mathtt{M};\ \gamma_a;\ \delta_a \rangle$ *and* $\mathbb{P} \between \langle \Gamma_o; \Delta_o \mid \Gamma_p; \Delta_p \rangle$, *then* $\delta = \gamma_a \cdot \gamma_p$ *is a telescoped substituition, and there exists* $\mathtt{I}_a \cdot \mathtt{I}_p;\ d : \neg\mathbb{1} \vdash \delta^* : \Gamma_p \cdot \Gamma_o$.

Definition G.6. We define the function $\Phi^{\text{ci}}_{\text{op}} : \text{Confs}_{\text{ci}} \to \Lambda_{\text{eff}}$ as the function:

$$\langle \mathtt{I}_p;\ \gamma_p;\ \xi_p \dashv\!\vdash \mathtt{M};\ \gamma_a;\ \delta_a \rangle \mapsto \mathtt{M}\{(\gamma_p \cdot \gamma_a)^* \cdot \delta_e\}$$

with

$$\delta_e := \begin{cases} \varepsilon & \text{if } \xi_p = \bot \\ \delta_a \cdot \delta_p & \text{if for some } \mathsf{Y} \in \{a, p\},\ \xi_p = !(e, \_, \delta_p) \end{cases}$$

We write $\mathscr{B}^{\text{ci}}_{\text{op}}$ for the functional relation corresponding to $\Phi^{\text{ci}}_{\text{op}}$.

Lemma G.7. *If* $\mathbb{C} \overset{\text{dual}}{\mapsto}_{\text{ci}} \mathbb{D}$ *or* $\mathbb{C} \overset{\text{hdl}}{\mapsto}_{\text{ci}} \mathbb{D}$, *then* $\Phi^{\text{ci}}_{\text{op}}(\mathbb{C}) = \Phi^{\text{ci}}_{\text{op}}(\mathbb{D})$.

Proof. We proceed by case analysis knowing that $\mathbb{C} = \langle \mathtt{I}_p;\ \gamma_p;\ \xi_p \dashv\!\vdash \mathtt{P}(a);\ \gamma_a;\ \delta_a \rangle$ and that $\mathtt{P}(a)$ is in normal form. □

Lemma G.8. *Taking a configuration* $\mathbb{C} \in \mathrm{Confs}_{\mathrm{ci}}$, *there exists a configuration* $\mathbb{D}$ *such that* $\mathbb{C}(\overset{\mathrm{dual}}{\mapsto_{\mathrm{ci}}})^*\mathbb{D}$ *and* $\mathbb{D} \overset{\mathrm{dual}}{\not\mapsto_{\mathrm{ci}}}$.

Proof. Let's write $\mathbb{C}$ as $\langle \mathtt{I}_p; \gamma_p; \xi_p \dashv\vdash \mathtt{M}; \gamma_a; \delta_a\rangle$ and $\gamma = [a_0 \mapsto \mathtt{v}_0, \ldots, a_\mathtt{n} \mapsto \mathtt{v}_\mathtt{n}]$ for $\gamma_a \cdot \gamma_p$.

If $\mathtt{M}$ is not in normal form then $\mathbb{C} \overset{\mathrm{dual}}{\not\mapsto_{\mathrm{ci}}}$, thus $\mathbb{D} = \mathbb{C}$. Otherwise, there exists an index $k \in \{0, \ldots, \mathtt{n}\}$ and a co-pattern, whose shape is either $\langle \Box \_ \mid \mathtt{K}\rangle$ or $\langle \_ \mid \mathtt{K}[\Box] \circ \_\rangle$, such that $\mathtt{M} = \mathtt{P}(a_k)$.
We proceed by induction on k and on the length of $\mathtt{K}$.

case $\mathtt{M} = \langle f\ \mathtt{v} \mid \mathtt{K}\rangle$ with $f \in \mathrm{dom}(\gamma)$.

Subcase $\gamma(f) \in \mathrm{dom}(\gamma)$.

Since γ is a telescoped substitution, there exists $\ell \in \{k+1, \ldots, \mathtt{n}\}$ such that $\gamma(f) = a_\ell$ Thus there exists a configuration $\mathbb{D}$ such that $\mathbb{C} \overset{\mathrm{dual}}{\mapsto_{\mathrm{ci}}} \mathbb{C}'$ and whose active component can be written as $\langle d[a_\ell\ \mathtt{A}]; \gamma_p \cdot \gamma_\mathtt{A}; \varnothing\rangle$ By induction hypothesis, there exists $\mathbb{D}$ such that $\mathbb{C}'(\overset{\mathrm{dual}}{\mapsto_{\mathrm{ci}}})^*\mathbb{D}$, and in particular $\mathbb{C}'(\overset{\mathrm{dual}}{\mapsto_{\mathrm{ci}}})^*\mathbb{D}$, and $\mathbb{D} \overset{\mathrm{dual}}{\not\mapsto_{\mathrm{ci}}}$.

Subcase $\gamma(f) \notin \mathrm{dom}(\gamma)$.

$\gamma(f)$ is a λ-abstraction. Then there exists a configuration $\mathbb{D}$ such that $\mathbb{C} \overset{\mathrm{dual}}{\mapsto_{\mathrm{ci}}} \mathbb{D}$ and whose active component can be written as $\langle d[(\lambda x.\mathtt{t})\ \mathtt{A}]; \gamma_p \cdot \gamma_\mathtt{A}; \varnothing\rangle$. From which we can deduce that $\mathbb{D} \overset{\mathrm{op}}{\mapsto_{\mathrm{ci}}}$ and in particular that $\mathbb{D} \overset{\mathrm{dual}}{\not\mapsto_{\mathrm{ci}}}$.

case $\mathtt{M} ::= \langle \mathbf{ret}\ \mathtt{v} \mid \mathtt{K}[\kappa\ []]\rangle \mid \langle \mathtt{e} \mid \mathtt{K}[\kappa\ []] \circ \mathtt{S}\rangle$ with $\kappa \in \mathrm{dom}(\gamma)$.

Subcase $\gamma(\kappa) = \mathtt{T}[\mathtt{E}]$.

Then there exists a configuration $\mathbb{D}$ such that $\mathbb{C} \overset{\mathrm{dual}}{\mapsto_{\mathrm{ci}}} \mathbb{D}$ and whose active term $\mathtt{N}$ can be written as $\langle \mathbf{ret}\ \mathtt{A} \mid \mathtt{K}[\mathtt{T}[\mathtt{E}]]\rangle$ in which case $\mathtt{N} \mapsto_{\mathrm{eval}}$, or as $\langle \mathtt{e}\{\delta_p\} \mid \mathtt{K}[\mathtt{T}[\mathtt{E}]] \circ \mathtt{S}\rangle$ in which case $\mathtt{N} \mapsto_{\mathrm{fwd}}$. In either case, we have $\mathbb{D} \overset{\mathrm{op}}{\mapsto_{\mathrm{ci}}}$ and in particular that $\mathbb{D} \overset{\mathrm{dual}}{\not\mapsto_{\mathrm{ci}}}$.

Subcase $\gamma(\kappa) = \Box$.

Suppose $\mathtt{M} = \langle \mathbf{ret}\ \mathtt{v} \mid \mathtt{K}[\kappa\ []]\rangle$, we have

$$\begin{aligned} \mathbb{C} \quad &\overset{\mathrm{dual}}{\mapsto_{\mathrm{ci}}} \quad \langle \mathtt{I}_a; \gamma_a; \varnothing \dashv\vdash \langle \mathbf{ret}\ \mathtt{A} \mid d\ [\,]\ \rangle; \gamma_p \cdot \gamma_\mathtt{A} \cdot [d \mapsto \mathtt{K}]; \varnothing\rangle \\ &\overset{\mathrm{dual}}{\mapsto_{\mathrm{ci}}} \quad \langle \mathtt{I}_p; \gamma_p \cdot \gamma_\mathtt{A} \cdot \gamma_d; \varnothing \dashv\vdash \langle \mathbf{ret}\ \mathtt{B} \mid \mathtt{K}\rangle; \gamma_a \cdot \gamma_\mathtt{B}; \varnothing\rangle = \mathbb{C}' \end{aligned}$$

We conclude by applying the induction hypothesis on $\mathbb{C}'$.
The same reasoning applied to the other case.

case $\mathtt{M} ::= \langle \mathbf{ret}\ \mathtt{v} \mid c\ []\ \rangle \mid \langle \mathtt{e} \mid c\ []\ \circ \mathtt{S}\rangle$.

Subcase $c \notin \mathrm{dom}(\gamma)$ Then $\mathbb{D} = \mathbb{C}$ since $\mathbb{C} \overset{\mathrm{dual}}{\not\mapsto_{\mathrm{ci}}}$.

Subcase $\gamma(c) ::= \gamma(c) = a_\ell\ []\mid \mathtt{K}[a_\ell\ []]$ with $\ell \in \{k+1, \ldots, \mathtt{n}\}$.

There exists a configuration $\mathbb{C}'$ such that $\mathbb{C} \overset{\mathrm{dual}}{\mapsto_{\mathrm{ci}}} \mathbb{C}'$ and whose active component can be written as $\langle \mathtt{Q}(a_\ell); \gamma_p \cdot \gamma_\mathtt{Q}; \varnothing\rangle$ where $\mathtt{Q}$ is a co-pattern. We conclude using the induction hypostesis on $\mathbb{C}'$.

□

Lemma G.9. *Let* $\mathtt{I};\Gamma \vdash_c \mathtt{M}$ *be an interactive computation and let* $\mathtt{I};\Delta \vdash \gamma : \Gamma$ *be a name assignment.*
If $\mathtt{M} \mapsto_{op} \mathtt{N}$ *then* $\mathtt{M}\{\gamma\} \mapsto_{op} \mathtt{N}\{\gamma\}$.

Lemma G.10. *Let* $\mathtt{I};\Gamma \vdash_c \mathtt{M}$ *be a computation s.t.* $\mathtt{M} \mapsto_{op}$ *and let* $\mathtt{I};\Delta \vdash \gamma : \Gamma$ *be a name assignment.*
If $\mathtt{M}\{\gamma\} \mapsto_{op} \mathtt{N}\{\gamma\}$ *and* $\mathrm{supp}(\mathtt{M}) = \mathrm{supp}(\mathtt{N})$, *then* $\mathtt{M} \mapsto_{op} \mathtt{N}$.

Proof. By case analysis knowing that any such term $\mathtt{M}$ is given by the syntax:

$$\begin{array}{rcl} \mathtt{Nf}_{\mathrm{fwd}} & ::= & \langle \mathbf{ret}\ \mathtt{v} \mid \mathtt{K}[\mathtt{E}] \rangle \mid \langle (\lambda x.\mathtt{t})\ \mathtt{v} \mid \mathtt{K} \rangle \\ & \mid & \langle \mathbf{op}\ \mathtt{v} \mid \mathtt{K}[\{[]\}\ \mathbf{with}\ \mathtt{h}]\ \circ\ \mathtt{S} \rangle \quad (\iota \in \mathbf{hdl}(\mathtt{h})) \\ & \mid & \langle e \mid \mathtt{K}[\mathtt{E}]\ \circ\ \mathtt{S} \rangle \end{array}$$

□

Lemma G.11. *Taking a configuration* $\mathbb{C} \in \mathrm{Confs}_{ci}$ *s.t.* $\Phi^{ci}_{op}(\mathbb{C}) \mapsto_{op} \mathtt{N}$, *then there exists a configuration* $\mathbb{D} \in \mathrm{Confs}_{ci}$ *s.t.* $\mathbb{C} \Mapsto_{ci} \mathbb{D}$ *and* $\Phi^{ci}_{op}(\mathbb{D}) = \mathtt{N}$.

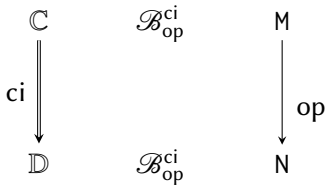
Proof. Let's write $\mathbb{C} = \langle \mathtt{I}_p; \gamma_p; \xi_p \dashv\vdash \mathtt{M}; \gamma_a; \delta_a \rangle$.

Case $\mathbb{C} \overset{dual}{\not\mapsto}_{ci}$; that is $\mathtt{M} \notin \mathtt{Nf}(\Lambda_{\mathrm{eff}})$.
By hypothesis, we have $\Phi^{ci}_{op}(\mathbb{C}) = \mathtt{M}\{\gamma\} \mapsto_{op} \mathtt{N}'\{\gamma\}$ where $\gamma = (\gamma_p \cdot \gamma_a)^*$ and $\mathtt{N}'$ is such that $\mathrm{supp}(\mathtt{M}) = \mathrm{supp}(\mathtt{N}')$ and $\mathtt{M} = \mathtt{N}'\{\gamma\}$. Then from lemma G.10, we can deduce that $\mathbb{C} \overset{op}{\mapsto}_{ci} \langle \mathtt{I}_p; \gamma_p; \varnothing \dashv\vdash \mathtt{N}'; \gamma_a; \varnothing \rangle = \mathbb{D}$ where $\mathtt{I} = \mathtt{I}_p \cdot \mathtt{I}'_a$, which entails that $\Phi^{ci}_{op}(\mathbb{D}) = \mathtt{N}'\{\gamma\} = \mathtt{N}$.

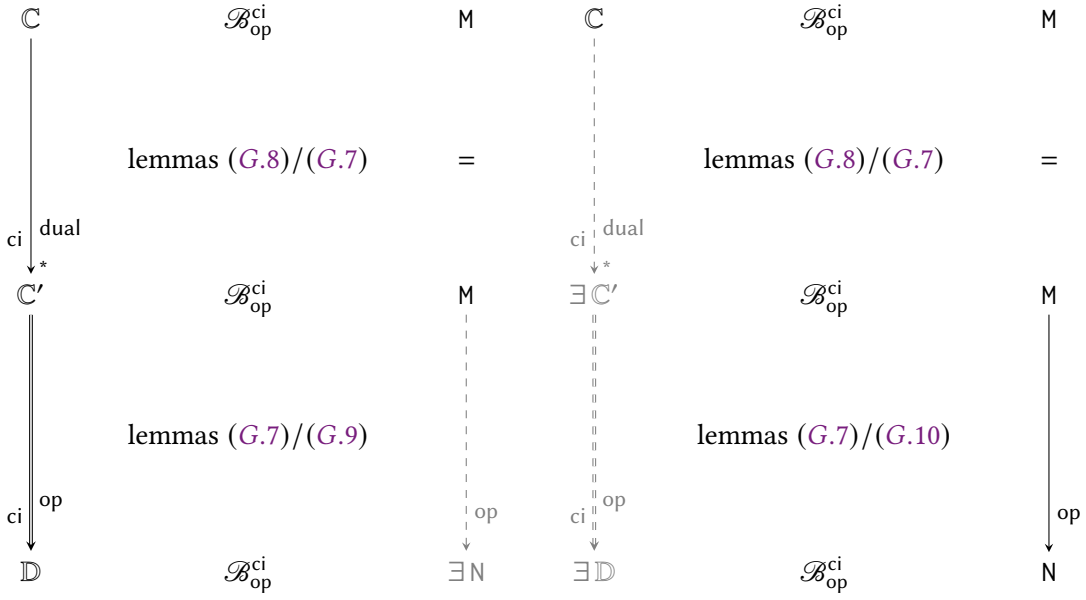
Case $\mathbb{C} \overset{dual}{\mapsto}_{ci}$**.**
By lemma G.8, there exists $\mathbb{C}'$ s.t. $\mathbb{C}(\overset{dual}{\mapsto}_{ci})^*\mathbb{C}'$ and $\mathbb{C}' \overset{dual}{\not\mapsto}_{ci}$. Since $\Phi^{ci}_{op}(\mathbb{C}) = \Phi^{ci}_{op}(\mathbb{C}')$ (by lemma G.7), then by applying the above argument to $\mathbb{C}'$ we can prove what we want.

□

Lemma G.12 (bisimulation). *Taking* $\overset{op}{\Mapsto}_{ci} := (\overset{hdl}{\mapsto}_{ci})^* \overset{op}{\mapsto}_{ci}$ *and* $\Mapsto_{ci} := (\overset{dual}{\mapsto}_{ci})^* \overset{op}{\Mapsto}_{ci}$, *there exists a bisimulation relating* $(\mathrm{Confs}_{ci}, \Mapsto_{ci})$ *and* $(\Lambda_{\mathrm{eff}}, \mapsto_{op})$, *i.e there exists a relation* $\mathscr{B}^{ci}_{op} \in \wp(\mathrm{Confs}_{ci} \times \Lambda_{\mathrm{eff}})$ *s.t. the following diagram holds.*



Proof. We show that $\mathscr{B}^{\mathsf{ci}}_{\mathsf{op}}$ and its converse are simulations.



□

Corollary G.13. *Taking a nominal term* $\mathtt{I}; \Gamma; \emptyset \vdash_{\mathsf{c}} \mathtt{t} : \tau$*, a compatible evaluation context* $\mathtt{I}'; \Delta \vdash \mathtt{K} : \tau \rightsquigarrow \mathbb{1}$ *and a assignment* $\mathtt{I}'; \Delta \vdash \gamma : \Gamma$*, then:*

$$\langle \mathtt{I}'_1,\, \gamma \cdot [c \mapsto d[\mathtt{K}]];\, \bot \dashv\vdash c[\mathtt{t}];\, \varepsilon,\, \bot \rangle \Downarrow_{\mathsf{ci}} \langle \mathtt{I}_2;\, \gamma_2;\, \bot \dashv\vdash d[\mathbf{ret}\,\langle\rangle];\, \gamma_1;\, \bot \rangle$$

$$\Longleftrightarrow$$

$$\mathtt{K}[\mathtt{t}\{\gamma\}] \Downarrow_{\mathsf{op}} d[\mathbf{ret}\,\langle\rangle]$$

H Semantic observation (proof of lemma F.2)

H.1 Relating $(\mathcal{L}_{\mathcal{T}}, \rightarrow_{\mathcal{T}})$ to $(\mathcal{L}_{\mathsf{AI}} \rightarrow_{\mathsf{I}})$

Lemma H.1. *Taking a well-typed interactive term in normal form* $\mathtt{I};\Gamma \vdash_{\mathsf{c}} \mathtt{Nf}$ *such that* $\mathbf{abstract}(\mathtt{Nf}, \delta) \nearrow\!\!\!\nearrow (\boldsymbol{a}.\mathsf{q}, \gamma, \delta\cdot\xi)$ *and* $\Gamma(a) = \tau$ *and* $\Gamma \Vdash \mathsf{q} : \neg\tau \triangleright \Gamma_{\mathsf{q}}; \Delta_{\mathsf{q}}$, *we have:* $\Gamma \vdash \gamma : \Gamma_{\mathsf{q}}$ *and* $\Gamma \vdash \xi : \Delta_{\mathsf{q}}$.

Lemma H.2. *Given a well-typed interactive term* $\mathtt{I};\Delta\cdot\Gamma \vdash_{\mathsf{c}} \mathtt{M}$ *and a substitution* $\Gamma' \vdash \delta : \Delta$ *such that* $\Gamma' \subseteq \Gamma$ *and* $\mathrm{dom}(\Delta) \subseteq \mathrm{dom}(\delta)$, *then:* $\mathtt{I};\Gamma \vdash_{\mathsf{c}} \mathtt{M}\{\delta\}$

Lemma H.3 (Subject reduction). *Taking* $\mathbb{I}, \mathbb{J} \in \mathcal{A}$ *and* $\mathbb{T} \in \mathcal{T}$ *such that* $\mathbb{I} \between \mathbb{T}$ *and* $\mathbb{I} \xrightarrow{\alpha}_{\mathsf{I}} \mathbb{J}$, *then there exists* $\mathbb{T} \in \mathcal{T}$ *such that* $\mathbb{T} \xrightarrow{\alpha}_{\mathcal{T}} \mathbb{S}$ *and* $\mathbb{J} \between \mathbb{S}$.

Proof. We write $\mathbb{T} = \langle \Gamma_o; \Delta_o \mid \Gamma_p; \Delta_p \rangle$ and proceed by case analysis on α.

Case $\alpha = hdl^{\oplus}$.

Then if we take $\mathbb{I} = \langle \mathtt{M};\ \gamma;\ \mathbf{!}\delta \rangle$, we have $\mathbb{I} \xrightarrow{hdl^{\oplus}}_{\mathsf{I}} \langle \mathtt{M};\ \gamma;\ \varnothing \rangle = \mathbb{J}$. It is immediate to verify that $\mathbb{S} = \langle \Gamma_o\cdot\Delta_o; \emptyset \mid \Gamma_p; \emptyset \rangle$ is s.t. $\mathbb{T} \xrightarrow{hdl^{\oplus}}_{\mathcal{T}} \mathbb{S}$ and $\mathbb{J} \between \mathbb{S}$.

Case $\alpha = hdl^{\ominus}$.

Then if we take $\mathbb{I} = \langle \mathtt{I};\ \gamma;\ \mathbf{?}\xi \rangle$ where $\xi = (\_, \_, \delta)$, we have $\mathbb{I} \xrightarrow{hdl^{\ominus}}_{\mathsf{I}} \langle \mathtt{I};\ \gamma\cdot\delta;\ \varnothing \rangle = \mathbb{J}$.

Similarily, it is immediate to verify that $\mathbb{S} = \langle \Gamma_o; \emptyset \mid \Gamma_p\cdot\Delta_p; \emptyset \rangle$ is s.t. $\mathbb{T} \xrightarrow{hdl^{\ominus}}_{\mathcal{T}} \mathbb{S}$ and $\mathbb{J} \between \mathbb{S}$.

Case $\alpha = \boldsymbol{a}.\mathsf{p}^{\oplus}$.

Then if we write $\mathbb{I} = \langle \mathtt{Nf};\ \gamma;\ \delta \rangle$, we have $\mathbb{I} \xrightarrow{\boldsymbol{a}.\mathsf{p}^{\oplus}}_{\mathsf{I}} \langle \mathtt{I};\ \gamma\gamma_{\mathsf{abs}};\ \xi \rangle = \mathbb{J}$ whith $\mathbf{abstract}(\mathtt{Nf}, \delta) \nearrow\!\!\!\nearrow (\gamma_{\mathsf{abs}},\ \delta\cdot\xi)$.

Since $\mathbb{I} \between \mathbb{T}$, we have $\mathtt{I};\Gamma_o\cdot\Delta_o \vdash_{\mathsf{c}} \mathtt{Nf}$, and since $a \in \mathrm{supp}(\mathtt{Nf})$ then there exists τ, Γ_{p}, Δ_{p} such that $\Gamma_o(a) = \tau$ and $\Gamma_o\cdot\Delta_p\cdot\Delta_o \Vdash \mathsf{p} : \neg\tau \triangleright \Gamma_{\mathsf{p}}; \Delta_{\mathsf{p}}$, so by lemma H.1, we have $\Gamma_o \vdash \gamma_{\mathsf{abs}} : \Gamma_{\mathsf{p}}$ and $\Gamma_o\cdot\Delta_o \vdash \xi : \Delta_{\mathsf{p}}$.

Thus by taking $\mathbb{S} = \langle \Gamma_o; \Delta_o \mid \Gamma_p\cdot\Gamma_{\mathsf{p}}; \Delta_p\cdot\Delta_{\mathsf{p}} \rangle$, it is immediate that $\mathbb{T} \xrightarrow{\boldsymbol{a}.\mathsf{p}^{\oplus}}_{\mathcal{T}} \mathbb{S}$ and $\mathbb{J} \between \mathbb{S}$, knowing that $\mathtt{I};\Gamma_o \vdash \gamma : \Gamma_p$, and $\mathtt{I};\Gamma_o \vdash \delta : \Delta_p$.

Case $\alpha = \boldsymbol{a}.\mathsf{p}^{\ominus}$.

Then if we take $\mathbb{I} = \langle \mathtt{I};\ \gamma;\ \xi \rangle$ where $\xi = (\_, \_, \delta)$, we have $\mathbb{I} \xrightarrow{\boldsymbol{a}.\mathsf{p}^{\ominus}}_{\mathsf{I}} \langle \mathtt{M};\ \gamma;\ \delta \rangle = \mathbb{J}$ where $\mathtt{M} = \mathsf{p}\{\delta\}(\gamma(a))$.

Let τ, Γ_{p}, Δ_{p} such that $\Gamma_p(a) = \tau$ and $\Delta_p\cdot\Delta_o \Vdash \mathsf{p} : \neg\tau \triangleright \Gamma_{\mathsf{p}}; \Delta_{\mathsf{p}}$, then we have $\mathbb{T} \xrightarrow{\boldsymbol{a}.\mathsf{p}^{\oplus}}_{\mathcal{T}} \langle \Gamma_o\cdot\Gamma_{\mathsf{p}}; \Delta_o\cdot\Delta_{\mathsf{p}} \mid \Gamma_p; \Delta_p \rangle = \mathbb{S}$.

Since $\mathbb{I} \between \mathbb{T}$, we have $\mathtt{I};\Gamma_o \vdash \gamma : \Gamma_p$ and $\mathtt{I};\Gamma_o \vdash \delta : \Delta_p$ and in particular $\mathtt{I};\Gamma_o\cdot\Gamma_{\mathsf{p}} \vdash \gamma : \Gamma_p$ and $\mathtt{I};\Gamma_o\cdot\Gamma_{\mathsf{p}} \vdash \delta : \Delta_p$.

Moreover, since $\mathtt{I};[a \mapsto \tau]\cdot\Gamma_{\mathsf{p}}\cdot\Delta_p\cdot\Delta_o\cdot\Delta_{\mathsf{p}} \vdash_{\mathsf{c}} \mathsf{p}(a)$ and $\mathrm{dom}(\Delta_o) = \mathrm{dom}(\delta)$, then by lemma H.2 we have $\mathtt{I};[a \mapsto \tau]\cdot\Gamma_{\mathsf{p}}\cdot\Delta_o\cdot\Delta_{\mathsf{p}} \vdash_{\mathsf{c}} \mathsf{p}\{\delta\}(a)$. Finally, given that $\mathtt{M} = \mathsf{p}\{\delta\}(a)$ and $\Gamma_o \vdash \gamma(a) : \tau$ then we can deduce that $\mathtt{I};\Gamma_o\cdot\Gamma_{\mathsf{p}}\cdot\Delta_o\cdot\Delta_{\mathsf{p}} \vdash_{\mathsf{c}} \mathtt{M}$, and by extent $\mathbb{J} \between \mathbb{S}$.

□

H.2 Compatibility of $(\mathcal{L}_{\mathsf{Al}}, \rightarrow_{\mathsf{I}})$ and $(\mathcal{L}_{wb}, \rightarrow_{wb})$

Definition H.4. We define the predicate $\between_{\square}$ on $\{?, !\} \times \mathcal{N} \times F$, relating an *abstract forward stack* to a *call frame*, depending on which *player* has performed the effect.

$$\frac{\cdot \between_{?} [] \qquad r \between_{?} f}{r :: \kappa' :: \kappa \between_{?} \kappa :: f} \qquad \frac{\cdot \between_{!} [] \qquad r \between_{!} f}{r :: \kappa' :: \kappa \between_{!} \kappa' :: f}$$

Definition H.5 ($(\mathcal{L}_{\mathsf{Al}}, \mathcal{L}_{wb})$-compatibility). Let $\mathbb{I}, \mathbb{J} \in \mathcal{A}$ and $\mathbb{W} \in \mathcal{W}$ s.t. $\mathbb{I} = \langle \gamma; \xi \rangle$, $\mathbb{J} = \langle \langle \mathsf{t} \mid \mathsf{K} \circ \mathsf{S} \rangle; \gamma'; \delta \rangle$ and $\mathbb{W} = \langle \sigma \mid \eta \mid \phi \rangle$.

We say that $\mathbb{I}$ and $\mathbb{W}$ are compatible when $\mathbf{Compat}(\mathbb{I}, \mathbb{W})$ holds, where

$$\mathbf{Compat}(\mathbb{I}, \mathbb{W}) :\Longleftrightarrow \forall \mathfrak{f} \in \phi.\ \mathrm{set}(\mathfrak{f}) \subseteq \mathrm{dom}(\gamma) \ \wedge\ \xi \sim_{\mathsf{fwd}} \eta$$

Similarily, we say that $\mathbb{J}$ and $\mathbb{W}$ are compatible when $\mathbf{Compat}(\mathbb{J}, \mathbb{W})$ holds, where

$$\mathbf{Compat}(\mathbb{J}, \mathbb{W}) :\Longleftrightarrow \forall \mathfrak{f} \in \phi.\ \mathrm{set}(\mathfrak{f}) \subseteq \mathrm{dom}(\gamma') \ \wedge\ \delta \sim_{\mathsf{fwd}} \eta$$

with

$$\varnothing \sim_{\mathsf{fwd}} \varnothing \qquad !(e, \_, \_) \sim_{\mathsf{fwd}} ([], e) \qquad ?(e, r, \_) \sim_{\mathsf{fwd}} (f, e) \text{ when } r \between_{?} f$$

$$!\delta^{e} \sim_{\mathsf{fwd}} ([], e) \qquad ?\delta \sim_{\mathsf{fwd}} (f, e) \text{ when } r \between_{?} f \text{ and } \mathsf{S}\{\delta^{-1}\} = \mathsf{T} \oplus r$$

Lemma H.6 (preservation of compatibility). *Let $\mathbb{I} \in \mathcal{A}$ and $\mathbb{W} \in \mathcal{W}$ s.t. $\mathbf{Compat}(\mathbb{I}, \mathbb{W})$. If $\mathbb{I} \xrightarrow{\alpha}_{\mathsf{I}} \mathbb{J}$ and $\mathbb{W} \xrightarrow{\alpha}_{wb} \mathbb{U}$, then $\mathbf{Compat}(\mathbb{J}, \mathbb{U})$.*

Proof. We write $\mathbb{W} = \langle \sigma \mid \eta \mid \phi \rangle$ with $\eta = (e, f)$ then proceed by case analysis on $\mathbb{I}$ and α.

Case $\mathbb{I} = \langle \mathtt{I}; \gamma; \square\, \xi \rangle$ with $\square \in \{?, !\}$ and $\xi = (e, r, \delta)$.

Subcase $\alpha = hdl^{\ominus}$.

By hypothesis, we have $r \between_{?} f$ and $\forall f' \in \phi.\ \mathrm{set}(f') \subseteq \mathrm{dom}(\gamma)$. Moreover, we have $\mathbb{J} = \langle \mathtt{I}; \gamma \cdot \delta; \varnothing \rangle$ and $\mathbb{U} = \langle \sigma \mid \varnothing \mid \phi \cup \{f\} \rangle$, thus, since $\mathrm{set}(f) \subseteq \mathrm{dom}(\delta)$ we can deduce that $\forall f' \in \phi \cup \{f\}.\ \mathrm{set}(f') \subseteq \mathrm{dom}(\gamma \cdot \delta)$ and $\varnothing \sim_{\mathsf{fwd}} \varnothing$, from which we can conclude that $\mathbf{Compat}(\mathbb{J}, \mathbb{U})$.

Subcase $\alpha ::= \boldsymbol{c}.\langle \mathbf{ret}\, \mathtt{A} \mid \square \rangle^{\ominus} \mid \boldsymbol{\kappa}.\langle \square[\mathbf{ret}\, \mathtt{A}] \mid d \rangle^{\ominus} \mid \boldsymbol{f}.\langle \square\, \mathtt{A} \mid d \rangle^{\ominus}$.

In either subcase, there exist $\mathtt{M}$ and σ' s.t. $\mathbb{J} = \langle \mathtt{M}; \gamma; \varnothing \rangle$ and $\mathbb{U} = \langle \sigma' \mid \varnothing \mid \phi \rangle$. Moreover, by defintion $\varnothing \sim_{\mathsf{fwd}} \varnothing$, and by hypothesis we have $\forall f \in \phi.\ \mathrm{set}(f) \subseteq \mathrm{dom}(\gamma)$, thus $\mathbf{Compat}(\mathbb{J}, \mathbb{U})$.

Subcase $\alpha ::= \boldsymbol{c}.\langle \kappa[e] \mid \square \rangle^{\ominus} \mid \boldsymbol{\kappa'}.\langle \square[\kappa[e]] \mid d \rangle^{\ominus}$ (*performing an effect*).

In either subcase, there exist K, σ' s.t. $\mathbb{J} = \langle \langle e \mid \mathsf{K} \circ \kappa \rangle; \gamma; \varepsilon \rangle$ and $\mathbb{U} = \langle \sigma' \mid (e, []) \mid \phi \rangle$.

We have then, by definition, $\kappa \between_{?} []$ and $?\varepsilon \sim_{\mathsf{fwd}} (e, [])$, and by hypothesis, we have that $\forall f \in \phi.\ \mathrm{set}(f) \subseteq \mathrm{dom}(\gamma)$. Thus, we can conclude that $\mathbf{Compat}(\mathbb{J}, \mathbb{U})$.

Subcase $\alpha ::= \boldsymbol{c}.\langle \kappa :: r[e] \mid \square \rangle^{\ominus} \mid \boldsymbol{\kappa'}.\langle \square[\kappa :: r[e]] \mid d \rangle^{\ominus}$ (*forwarding an effect*).

In either subcase, there exist K, σ' such that $\mathbb{J} = \langle \langle e\{\delta\} \mid \mathsf{K} \circ \kappa[r\{\delta\}] \rangle; \gamma; \square\, \delta \rangle$ and $\mathbb{U} = \langle \sigma' \mid \eta \mid \phi \rangle$.

By hypothesis, we have that $\forall f \in \phi.\ \mathrm{set}(f) \subseteq \mathrm{dom}(\gamma)$ and $r \between_{\square} f$.

If $\boxdot = \mathbf{?}$, then r is even-length, and by taking $\mathsf{S} = \kappa[r\{\delta\}]$, we have $\mathsf{S}\{\delta^1\} = \kappa :: r$ and $\kappa :: r \between_? f$. If otherwise $\boxdot = \mathbf{!}$, then $\eta = \varnothing$.

Thus in both cases, we have $\boxdot\ \delta \sim_{\mathsf{fwd}} \eta$, and we can conclude that **Compat**$(\mathbb{J}, \mathbb{U})$.

Case $\mathbb{I} = \langle \langle \mathsf{t} \mid \mathsf{K} \circ \mathsf{S} \rangle; \gamma; \delta \rangle$.

Subcase $\alpha ::= \boldsymbol{d}.\langle \mathbf{ret}\ \mathsf{A} \mid \Box \rangle^{\oplus} \mid \boldsymbol{\kappa}.\langle \Box[\mathbf{ret}\ \mathsf{A}] \mid c \rangle^{\oplus} \mid \boldsymbol{f}.\langle \Box\ \mathsf{A} \mid c \rangle^{\oplus}$.

In either subcase, there exist γ' and σ' s.t. $\mathbb{J} = \langle \mathtt{I}; \gamma \cdot \gamma'; \varnothing \rangle$ and $\mathbb{U} = \langle \sigma' \mid \varnothing \mid \phi \rangle$. By defintion, we have $\varnothing \sim_{\mathsf{fwd}} \varnothing$, and by hypothesis we have $\forall f \in \phi.\ \mathrm{set}(f) \subseteq \mathrm{dom}(\gamma)$, and in particular $\forall f \in \phi.\ \mathrm{set}(f) \subseteq \mathrm{dom}(\gamma \cdot \gamma')$, thus **Compat**$(\mathbb{J}, \mathbb{U})$.

Subcase $\alpha ::= \boldsymbol{d}.\langle \kappa :: r[e] \mid \Box \rangle^{\oplus} \mid \boldsymbol{\kappa'}.\langle \Box[\kappa :: r[e]] \mid c \rangle^{\oplus}$.

In either subcase, there exist $\gamma', \sigma', \eta', \mathsf{S}'$ and T s.t. $\mathbb{J} = \langle \mathtt{I}; \gamma \cdot \gamma'; \boxdot\ \xi \rangle$ and $\mathbb{U} = \langle \sigma' \mid \eta' \mid \phi \rangle$, where $\mathsf{S} = \mathsf{T} \oplus \mathsf{S}'$, $\mathsf{S}\{\delta^{-1}\} = r$ and $\xi = (e, r, \delta \cdot [\kappa \mapsto \mathsf{T}])$.

By hypothesis, we have that $\forall f \in \phi.\ \mathrm{set}(f) \subseteq \mathrm{dom}(\gamma)$ and $r \between_{\boxdot} f$.

If $\boxdot = \mathbf{?}$, then r is odd-length and $\eta' = (f +\!\!+ [\kappa], e)$, thus $\kappa :: r \between_? f +\!\!+ [\kappa]$. If otherwise $\boxdot = \mathbf{!}$, then $\eta = \eta' = \varnothing$.

Thus in both cases, we have $\boxdot\ \xi \sim_{\mathsf{fwd}} \eta$, and we can conclude that **Compat**$(\mathbb{J}, \mathbb{U})$.

□

H.3 Semantic ciu-observation

Given a term $\mathtt{I};\Gamma;\emptyset \vdash_c \mathtt{t} : \tau$, an *observer* of $\mathtt{t}$ is an environment given by an evaluation context and a name valuation $(\mathtt{K}, v)$ such that $\mathtt{I}';\emptyset \vdash \mathtt{K} : \tau \rightsquigarrow \mathbb{1}$ and $\mathtt{I}';\emptyset \vdash v : \Gamma$.

We want to give a semantic characterization of "observing $\langle\rangle$" but first we need to introduce the notion of *closed composability* of OGS configurations, which is the semantic counterpart of $\mathtt{t}$ being composable with $(\mathtt{K}, v)$ and $\mathtt{K}[\mathtt{t}\{v\}]$ being a closed term of type $\mathbb{1}$.

H.3.1 Closed composability of OGS *configurations.* If *complementarity* guarantees the "closedness" part of *closed composability*, we still require non-intial *program* and *environment* configurations to agree on their history of interaction. We will capture this *coherence of trajectory* by the following *synchronicity relation*.

First, we will the notion of *return frame* and auxiliary definitions.

Definition H.7. **(return frames)** We will call a *return frame* any control-flow stack of the shape $f +\!\!+ [c]$;*i.e.* that it is made up of a call frame ($f \in \mathtt{List}(\mathcal{K})$) to which is appended a single abstract undelimited continuation ($c \in C$).

We will also define the function F that, given an evaluation stack, computes the associated *return frame*.

$$F(c[\mathtt{S}]) := F(\mathtt{S}) +\!\!+ [c] \qquad F(\mathtt{T}[\kappa[\mathtt{S}]]) := F(\mathtt{S}) +\!\!+ [\kappa] \qquad F(\mathtt{T}) := []$$

Definition H.8. **(return frames decomposition)** For any a non-empty control-flow stack σ generated by $\mathcal{L}_{wb}$, there corresponds a natural number $k \in \mathbb{N}^*$ and k *return frames* $\sigma_1, \cdots, \sigma_k$ such that $\sigma = \sigma_1 +\!\!+ \cdots +\!\!+ \sigma_k$.

We will write $\sigma_1 \oplus \cdots \oplus \sigma_k$ for this unique decomposition of σ.

Definition H.9. **(Synchronicity)** We define synchronicity as a quarternary relation $\mathcal{S} \subseteq \mathcal{A} \times \mathcal{W} \times \mathcal{A} \times \mathcal{W}$.

Let $\mathbb{I} = \langle \mathtt{M};\ \gamma_{\mathbb{I}};\ \blacksquare\delta_{\mathbb{I}}\rangle$ in $\mathrm{Confs}_{\mathrm{act}}$, $\mathbb{J} = \langle \gamma_{\mathbb{J}},\ \square(\_,\ r,\ \_)\rangle$ in $\mathrm{Confs}_{\mathrm{pas}}$ where $\mathtt{M} = \langle \mathtt{t} \mid c_0[\mathtt{S}_0] \circ \mathtt{S}\rangle$ and $\blacksquare, \square \in \{?, !\}$, Let $\mathbb{W} = \langle \sigma_{\mathbb{W}} \mid (f_{\mathbb{W}},\ e_{\mathbb{W}}) \mid \phi_{\mathbb{W}}\rangle$, $\mathbb{U} = \langle \sigma_{\mathbb{U}} \mid (f_{\mathbb{U}}, e_{\mathbb{U}}) \mid \phi_{\mathbb{U}}\rangle$ in $\mathcal{W}$ where $\sigma_0 \oplus \cdots \oplus \sigma_k$ and $\sigma'_0 \oplus \cdots \oplus \sigma'_p$ (for some $k, p \in \mathbb{N}^*$) are the respective decompositions of $\sigma_{\mathbb{W}}$ and $\sigma_{\mathbb{U}}$ into *return frames*.

We say that $\langle \mathbb{I}, \mathbb{W}\rangle$ and $\langle \mathbb{J}, \mathbb{U}\rangle$ are *in sync*, and write $\langle \mathbb{I}, \mathbb{W}\rangle\ \mathcal{S}\ \langle \mathbb{U}, \mathbb{J}\rangle$ when the following holds:

- $\mathbb{I} \dashv\vdash \mathbb{J}$ and for all $i \geq 0$, we can write $\gamma_{\mathbb{J}}(c_i) = d_{i+1}[\mathtt{S}'_i]$ and $\gamma_{\mathbb{I}}(d_{i+1}) = c_{i+1}[\mathtt{S}_{i+1}]$.
- $r \between \blacksquare f_{\mathbb{U}}$ and $r \between \square f_{\mathbb{J}}$
- $\forall 0 \leq i.\ F(c_i[\mathtt{S}_i]) \simeq_{\phi_{\mathbb{U}}} \sigma'_i$ and $F(\gamma_{\mathbb{J}}(c_i)) \simeq_{\phi_{\mathbb{W}}} \sigma_i$

where $\simeq_\phi$ is given by

$$\textsc{Refl}\ \frac{\pi = \pi' \qquad \phi \in K}{\pi \simeq_\phi \pi'} \qquad\qquad \frac{\pi = \pi' \qquad \phi \in K \qquad f \in \phi}{\pi \simeq_\phi f +\!\!+ \pi'}\ \textsc{Up-to}$$

Next we show how the *synchronicity* relation indeed captures the *coherence* of both the *program's* and the *environment's* trajectories, in the sense that whatever interaction is triggered by the *active* player is in fact compatible with what the *passive* player is expecting, and that this property is preserved by interaction.

This is formalised in the next lemma, which proves that $\mathcal{S}$ is a simulation.

Lemma H.10 (O-simulation). *Given* $\mathbb{K} \in \mathrm{Confs}_{\mathrm{pas}}$, $\mathbb{I} \in \mathrm{Confs}_{\mathrm{act}}$, $\mathbb{W}$, $\mathbb{U} \in \mathcal{W}$ *such that* $\langle \mathbb{I}, \mathbb{W} \rangle \, \mathcal{S} \, \langle \mathbb{K}, \mathbb{U} \rangle$, *we have:*

$$\frac{\mathbb{I} \xrightarrow{\mathbf{m}}_{\mathsf{I}} \mathbb{J} \qquad \mathbb{W} \xrightarrow{\mathbf{m}}_{wb} \mathbb{W}'}{\exists \mathbb{J}, \mathbb{U}'.\ \mathbb{K} \xrightarrow{\mathbf{m}^\perp}_{\mathsf{I}} \mathbb{J} \qquad \mathbb{U} \xrightarrow{\mathbf{m}^\perp}_{wb} \mathbb{U}' \qquad \langle \mathbb{J}, \mathbb{W}' \rangle \, \mathcal{S} \, \langle \mathbb{J}, \mathbb{U}' \rangle}$$

Proof. We write $\mathbb{I} = \langle \langle \mathsf{t} \mid c_0[\mathsf{S}_0] \circ \mathsf{S} \rangle ; \gamma_{\mathbb{I}}, \delta_{\mathbb{I}} \rangle$, $\mathbb{W} = \langle \sigma_{\mathbb{W}} \mid \eta_{\mathbb{W}} \mid \phi_{\mathbb{W}} \rangle$, $\mathbb{W}' = \langle \sigma_{\mathbb{W}'} \mid (\_, r_{\mathbb{W}'}, \_) \mid \phi_{\mathbb{W}'} \rangle$, $\mathbb{U} = \langle \sigma_{\mathbb{U}} \mid \eta_{\mathbb{U}} \mid \phi_{\mathbb{U}} \rangle$ and $\mathbb{K} = \langle \mathrm{I}_{\mathbb{K}}; \gamma_{\mathbb{K}}; \xi_{\mathbb{K}} \rangle$ We will also write $\sigma_0 \oplus \cdots \oplus \sigma_k$ and $\sigma'_0 \oplus \cdots \oplus \sigma'_p$ (for some $k, p \in \mathbb{N}^*$) for the respective decompositions of $\sigma_{\mathbb{W}}$ and $\sigma_{\mathbb{U}}$, as well as $\gamma_{\mathbb{K}}(c_i) = d_{i+1}[\mathsf{S}'_i]$ and $\gamma_{\mathbb{I}}(d_{i+1}) = c_{i+1}[\mathsf{S}_{i+1}]$ for $i \geq 0$.

We proceed by case analysis on $\mathbf{m}$.

Case $\mathbf{m} = \boldsymbol{c_0}.\langle \mathbf{ret}\, \mathsf{A} \mid \Box \rangle^{\oplus}$.

We have $\mathsf{S}_0 = c_0[[]]$, $\mathsf{S} = []$, $\mathsf{t} = \mathbf{ret}\, \mathsf{V}$, $\delta_{\mathbb{I}} = \varnothing$ and $\eta_{\mathbb{W}} = \varnothing$, $r_{\mathbb{W}'} = \cdot$

$\mathbb{I},\ \mathbb{W} \xrightarrow{\mathbf{m}} \langle \mathrm{I}_{\mathbb{I}}; \gamma_{\mathbb{I}}; \varnothing \rangle,\ \langle \sigma_{\mathbb{W}'} \mid \varnothing \mid \phi_{\mathbb{W}'} \rangle$

Since $\langle \mathbb{I}, \mathbb{W} \rangle \, \mathcal{S} \, \langle \mathbb{K}, \mathbb{U} \rangle$, then $\sigma'_0 = F(c_0[[]]) = [c_0]$ and $\xi_{\mathbb{K}} = \varnothing$. Thus $\mathbb{U} \xrightarrow{\mathbf{m}^\perp}_{wb} \langle \sigma_{\mathbb{U}'} \mid \varnothing \mid \phi_{\mathbb{U}} \rangle = \mathbb{U}'$ and $\mathbb{K} \xrightarrow{\mathbf{m}^\perp}_{\mathsf{I}} \langle \langle \mathbf{ret}\, \mathsf{A} \mid \gamma_{\mathbb{K}}(c_0) \rangle ; \gamma_{\mathbb{K}}; \varnothing \rangle = \mathbb{J}$ with $\sigma_{\mathbb{U}'} = \sigma'_1 \oplus \cdots \oplus \sigma'_p$.

First, it is clear that $r_{\mathbb{W}'} \between f'_{\mathbb{U}}$, $r_{\mathbb{W}'} \between f'_{\mathbb{W}}$. Second, by synchronicity, we have $F(\gamma_{\mathbb{K}}(c_\ell)) \simeq_{\phi_{\mathbb{W}}} \sigma_\ell$ and $F(c_\ell[\mathsf{S}_\ell]) \simeq_{\phi_{\mathbb{U}}} \sigma'_\ell$, then if we take, forall $\ell \geq 0$, $c'_\ell = d_{\ell+1}$ and write $\gamma_{\mathbb{K}}(c_0) = c'_0[\mathsf{S}'_0]$, then forall $\ell \geq 0$, we get $F(c'_\ell[\mathsf{S}'_\ell]) \simeq_{\phi_{\mathbb{W}'}} \sigma_\ell$ and $F(\gamma_{\mathbb{I}}(c'_\ell)) \simeq_{\phi_{\mathbb{U}'}} \sigma'_{\ell+1}$, i.e . $\langle \mathbb{J}, \mathbb{W}' \rangle \, \mathcal{S} \, \langle \mathbb{J}, \mathbb{U}' \rangle$ since $\sigma_{\mathbb{W}} = \sigma_{\mathbb{W}'}$ and $\sigma_{\mathbb{U}'} = \sigma'_1 \oplus \cdots \oplus \sigma'_p$.

Case $\mathbf{m} = \boldsymbol{\kappa}.\langle \Box[\mathbf{ret}\, \mathsf{A}] \mid d_0 \rangle^{\oplus}$.

In this case, S_0 must be of the shape $\mathsf{S}[\kappa[[]]]$, $\mathsf{t} = \mathbf{ret}\, \mathsf{V}$, $\xi_{\mathbb{I}} = \varnothing$ and $\eta_{\mathbb{W}} = \varnothing$ and

$\mathbb{I},\ \mathbb{W} \xrightarrow{\mathbf{m}} \langle \mathrm{I}_{\mathbb{I}}; \gamma_{\mathbb{I}} \cdot [d_0 \mapsto c_0[\mathsf{S}]]; \varnothing \rangle,\ \langle [d_0] \oplus \sigma_{\mathbb{W}} \mid \varnothing \mid \phi_{\mathbb{W}} \rangle$

Since $\langle \mathbb{I}, \mathbb{W} \rangle \, \mathcal{S} \, \langle \mathbb{K}, \mathbb{U} \rangle$, then $F(c_0[\mathsf{S}[\kappa[[]]]]) \simeq_{\phi_{\mathbb{U}}} \sigma'_0$ and $\xi_{\mathbb{K}} = \varnothing$.

Subcase $F(c_0[\mathsf{S} \circ \kappa]) = \sigma'_0$.

There exists $\sigma''_0 = F(c_0[\mathsf{S}])$ s.t. $\sigma'_0 = \kappa :: \sigma''_0$,

We have $\mathbb{U} \xrightarrow{\mathbf{m}^\perp}_{wb} \langle \sigma''_0 +\!\!+ \sigma'_{\mathbb{U}} \mid \varnothing \mid \phi_{\mathbb{U}} \rangle = \mathbb{U}'$ that is $\sigma_{\mathbb{U}'} = \sigma''_0 \oplus \sigma'_1 \oplus \cdots \oplus \sigma'_p$ and $\mathbb{K} \xrightarrow{\mathbf{m}^\perp}_{\mathsf{I}} \langle \langle \mathbf{ret}\, \mathsf{A} \mid d_0[\gamma_{\mathbb{K}}(\kappa)] \rangle ; \gamma_{\mathbb{K}}; \varnothing \rangle = \mathbb{J}$.

Subcase $F(c_0[\mathsf{S} \circ \kappa]) \neq \sigma'_0$.

There exists $f \in \phi_{\mathbb{U}}$ s.t. $F(c_0[\mathsf{S} \circ \kappa]) = f +\!\!+ \sigma'_0$.

We have $\mathbb{U} \xrightarrow{\mathbf{m}^\perp}_{wb} \langle \sigma''_0 +\!\!+ \sigma'_{\mathbb{U}} \mid \varnothing \mid \phi_{\mathbb{U}} \rangle = \mathbb{U}'$ that is $\sigma_{\mathbb{U}'} = \sigma''_0 \oplus \sigma'_1 \oplus \cdots \oplus \sigma'_p$

where $f = \kappa :: f'$ and $\sigma''_0 = f' +\!\!+ \sigma'_0$ and $\mathbb{K} \xrightarrow{\mathbf{m}^\perp}_{\mathsf{I}} \langle \langle \mathbf{ret}\, \mathsf{A} \mid d_0[\gamma_{\mathbb{K}}(\kappa)] \rangle ; \gamma_{\mathbb{K}}; \varnothing \rangle = \mathbb{J}$.

In either subcase, let's write $d_0[\mathsf{S}''_0]$ for $d_0[\gamma_{\mathbb{K}}(\kappa)]$ and, forall $\ell \geq 0$, $\mathsf{S}''_{\ell+1} = \mathsf{S}'_\ell$.

First, it is clear that $r_{\mathbb{W}'} \between f'_{\mathbb{U}}$, $r_{\mathbb{W}'} \between f'_{\mathbb{W}}$ and that $F(d_0[\mathsf{S}''_0]) \simeq_{\phi_{\mathbb{W}'}} [d_0]$ and that $F(\gamma_{\mathbb{I}}(d_0)) \simeq_{\phi_{\mathbb{U}'}} \sigma''_0$, since $\phi_{\mathbb{U}} = \phi_{\mathbb{U}'}$, $\phi_{\mathbb{W}} = \phi_{\mathbb{W}'}$ and $\gamma_{\mathbb{I}}(d_0) = \gamma_{\mathbb{I}}(d_0) = c_0[\mathsf{S}]$.

Then we have, by synchronicity, forall $\ell \geq 0$, $F(\gamma_{\mathbb{K}}(c_\ell)) \simeq_{\phi_{\mathbb{W}}} \sigma_\ell$ and $F(c_\ell[\mathsf{S}_\ell]) \simeq_{\phi_{\mathbb{U}}} \sigma'_\ell$, that is, forall $\ell \geq 1$, $F(d_\ell[\mathsf{S}''_\ell]) \simeq_{\phi_{\mathbb{W}'}} \sigma_\ell$ and $F(\gamma_{\mathbb{I}}(d_\ell)) \simeq_{\phi_{\mathbb{U}'}} \sigma'_\ell$,

Thus we can conclude that $\langle \mathbb{J}, \mathbb{W}' \rangle \, \mathcal{S} \, \langle \mathbb{J}, \mathbb{U}' \rangle$.

Case $\mathbf{m} = \boldsymbol{c_0}.\langle \kappa :: r[e] \mid \Box \rangle^{\oplus}$.

In this case, $\langle \mathsf{t} \mid c_0[\mathsf{S}_0] \circ \mathsf{S} \rangle$ must be of the shape $\langle \mathsf{e} \mid c_0[[]] \circ \mathsf{T} \oplus \mathsf{S} \rangle$,

Subcase $\mathsf{e} = \mathbf{op}\, \mathsf{V}$.

$\mathbb{I},\ \mathbb{W} \xrightarrow{\mathbf{m}} \langle \mathtt{I}_{\mathbb{I}};\ \gamma_{\mathbb{I}};\ \mathbf{!}(e, \kappa :: r, \delta_{\mathbb{I}} \cdot [\kappa \mapsto \mathsf{T}]) \rangle,\ \langle \sigma_{\mathbb{W}} \mid (f_{\mathbb{W}} +\!\!+ [\kappa], e) \mid \phi_{\mathbb{W}} \rangle$.
Since $\langle \mathbb{I}, \mathbb{W} \rangle\ \mathcal{S}\ \langle \mathbb{K}, \mathbb{U} \rangle$, then $\mathcal{F}(c_0[[]]) \simeq_{\phi_{\mathbb{U}}} \sigma'_0$, i.e $\sigma'_0 = [c_0]$ and $\xi_{\mathbb{K}} = \mathbf{?}(e,\ r,\ \delta_{\mathbb{K}})$.
We have then $\mathbb{K} \xrightarrow{\mathbf{m}^{\perp}}_{\mathsf{I}} \langle \langle e \mid \gamma_{\mathbb{K}}(c_0) \circ \kappa[r\{\delta_{\mathbb{K}}\}] \rangle;\ \gamma_{\mathbb{K}};\ \mathbf{?}(e, \kappa :: r, \delta_{\mathbb{K}}) \rangle = \mathbb{J}$ and $\mathbb{U} \xrightarrow{\mathbf{m}^{\perp}}_{wb} \langle \sigma'_1 \oplus \cdots \oplus \sigma'_p \mid (f_{\mathbb{U}}, e) \mid \phi_{\mathbb{U}} \rangle = \mathbb{U}'$
By synchronicity, we have $\mathcal{F}(\gamma_{\mathbb{K}}(c_\ell)) \simeq_{\phi_{\mathbb{W}}} \sigma_\ell$ and $\mathcal{F}(c_\ell[\mathtt{S}_\ell]) \simeq_{\phi_{\mathbb{U}}} \sigma'_\ell$, $r \between\ !f_{\mathbb{W}}$ and $r \between\ ?f_{\mathbb{U}}$ then if we take, forall $\ell \geq 0$, $c'_\ell = d_{\ell+1}$ and write $\gamma_{\mathbb{K}}(c_0) = c'_0[\mathtt{S}'_0]$, we get $\kappa :: r \between\ !f'_{\mathbb{U}}$, $\kappa :: r \between\ ?f_{\mathbb{W}'}$ and forall $\ell \geq 0$ we get $\mathcal{F}(c'_\ell[\mathtt{S}'_\ell]) \simeq_{\phi_{\mathbb{W}'}} \sigma_\ell$ and $\mathcal{F}(\gamma_{\mathbb{I}}(c'_\ell)) \simeq_{\phi_{\mathbb{U}'}} \sigma'_{\ell+1}$, i.e . $\langle \mathbb{J}, \mathbb{W}' \rangle\ \mathcal{S}\ \langle \mathbb{J}, \mathbb{U}' \rangle$ since $\sigma_{\mathbb{W}} = \sigma_{\mathbb{W}'}$ and $\sigma_{\mathbb{U}'} = \sigma'_1 \oplus \cdots \oplus \sigma'_p$.

Subcase $\mathsf{e} = e.\ \mathbb{I},\ \mathbb{W} \xrightarrow{\mathbf{m}} \langle \mathtt{I}_{\mathbb{I}};\ \gamma_{\mathbb{I}};\ \mathbf{?}(e, \kappa :: r, \delta_{\mathbb{I}} \cdot [\kappa \mapsto \mathsf{T}]) \rangle,\ \langle \sigma_{\mathbb{W}} \mid (f_{\mathbb{W}} +\!\!+ [\kappa], e) \mid \phi_{\mathbb{W}} \rangle$.
Since $\langle \mathbb{I}, \mathbb{W} \rangle\ \mathcal{S}\ \langle \mathbb{K}, \mathbb{U} \rangle$, then $\mathcal{F}(c_0[[]]) \simeq_{\phi_{\mathbb{U}}} \sigma'_0$, i.e $\sigma'_0 = [c_0]$ and $\xi_{\mathbb{K}} = \mathbf{!}(e, r, \delta_{\mathbb{K}})$.
We have then $\mathbb{K} \xrightarrow{\mathbf{m}^{\perp}}_{\mathsf{I}} \langle \langle \mathsf{e}\{\delta_{\mathbb{K}}\} \mid \gamma_{\mathbb{K}}(c_0) \circ \kappa[r\{\delta_{\mathbb{K}}\}] \rangle;\ \gamma_{\mathbb{K}};\ \mathbf{?}(e, \kappa :: r, \delta_{\mathbb{K}}) \rangle = \mathbb{J}$ and $\mathbb{U} \xrightarrow{\mathbf{m}^{\perp}}_{wb} \langle \sigma'_1 \oplus \cdots \oplus \sigma'_p \mid (f_{\mathbb{U}}, e) \mid \phi_{\mathbb{U}} \rangle = \mathbb{U}'$
By synchronicity, we have $\mathcal{F}(\gamma_{\mathbb{K}}(c_\ell)) \simeq_{\phi_{\mathbb{W}}} \sigma_\ell$ and $\mathcal{F}(c_\ell[\mathtt{S}_\ell]) \simeq_{\phi_{\mathbb{U}}} \sigma'_\ell$, $r \between\ ?f_{\mathbb{W}}$ and $r \between\ !f_{\mathbb{U}}$ then if we take, forall $\ell \geq 0$, $c'_\ell = d_{\ell+1}$ and write $\gamma_{\mathbb{K}}(c_0) = c'_0[\mathtt{S}'_0]$, we get $\kappa :: r \between\ ?f'_{\mathbb{U}}$, $\kappa :: r \between\ !f_{\mathbb{W}'}$ and forall $\ell \geq 0$ we get $\mathcal{F}(c'_\ell[\mathtt{S}'_\ell]) \simeq_{\phi_{\mathbb{W}'}} \sigma_\ell$ and $\mathcal{F}(\gamma_{\mathbb{I}}(c'_\ell)) \simeq_{\phi_{\mathbb{U}'}} \sigma'_{\ell+1}$, i.e . $\langle \mathbb{J}, \mathbb{W}' \rangle\ \mathcal{S}\ \langle \mathbb{J}, \mathbb{U}' \rangle$ since $\sigma_{\mathbb{W}} = \sigma_{\mathbb{W}'}$ and $\sigma_{\mathbb{U}'} = \sigma'_1 \oplus \cdots \oplus \sigma'_p$.

Case $\mathbf{m} = \boldsymbol{\kappa}.\langle \Box[r[e]] \mid d_0 \rangle^{\oplus}$. Similar reasoning applies.

□

Corollary H.11. *Given* $\mathbb{J} \in \mathrm{Confs}_{\mathrm{pas}}$, $\mathbb{I} \in \mathrm{Confs}_{\mathrm{act}}$, $\mathbb{W}$, $\mathbb{U} \in \mathcal{W}$ *and a trace* t *such that* $\langle \mathbb{J}, \mathbb{W} \rangle\ \mathcal{S}\ \langle \mathbb{I}, \mathbb{U} \rangle$, $\mathsf{t} \in \mathbf{Tr}_{\mathcal{L}_{\mathsf{AI}}}(\mathbb{I})$ *and* $\mathbf{Tr}_{\mathcal{L}_{\mathsf{AI}}}(\mathbb{J})$ *then:*

$$\mathsf{t} \in \mathbf{Tr}_{\mathcal{L}_{wb}}(\mathbb{W}) \wedge \mathsf{t}^{\perp} \in \mathbf{Tr}_{\mathcal{L}_{wb}}(\mathbb{U})$$

Proof. By induction on the length of t using Lemma H.10. □

Definition H.12. **(Composability)** Given a passive configuration $\mathbb{G} \in O^{\ominus}$ and an active one $\mathbb{H} \in O^{\oplus}$ such that $\mathbb{G} = \langle \mathbb{I} \parallel \mathbb{T} \parallel \mathbb{W} \rangle$ and $\mathbb{H} = \langle \mathbb{J} \parallel \mathbb{S} \parallel \mathbb{U} \rangle$, we say that $\mathbb{G}$ and $\mathbb{H}$ are *closed composable* and write $\mathbb{G} \perp\!\!\!\perp \mathbb{H}$ when:

$$\frac{\mathbb{G} = \langle \mathbb{I} \parallel \mathbb{T} \parallel \mathbb{W} \rangle \qquad \mathbb{H} = \langle \mathbb{J} \parallel \mathbb{S} \parallel \mathbb{U} \rangle \qquad \langle \mathbb{I}, \mathbb{W} \rangle\ \mathcal{S}\ \langle \mathbb{J}, \mathbb{U} \rangle}{\mathbb{G} \perp\!\!\!\perp \mathbb{H}}$$

Remark 8. *The preservation of* closed composability *by configuration synchronization follows from lemmas E.2 and H.10.*

H.3.2 OGS *Observation.* Now we can semantically capture $\mathsf{K}[\mathsf{t}\{v\}] \Downarrow_{\mathrm{op}} \langle\rangle$ by the following predicate.

Definition H.13. **(Observation predicate)** Taking a passive configuration $\mathbb{G} \in O^{\ominus}$ and an active one $\mathbb{H} \in O^{\oplus}$ such that $\mathbb{G} \perp\!\!\!\perp \mathbb{H}$, we define the observation of $\mathbb{G}$ on $\mathbb{H}$ as the predicate:

$$\langle \mathbb{G} \perp\!\!\!\perp \mathbb{H} \rangle \Downarrow [c]\,\mathbf{ret}\,\langle\rangle \iff \exists \mathbb{K}, \gamma.\ \langle \mathbb{J} \dashv\vdash \mathbb{I} \rangle \Downarrow_{\mathsf{ci}} \langle \mathbb{K} \dashv\vdash c[\mathbf{ret}\,\langle\rangle];\ \gamma;\ \perp \rangle$$

The soundness of this observation predicate follows from Theorem F.1, Lemmas E.2, H.3, H.10. Its completeness is a consequence of the following result: the definability of configurations that are composable with an initial active configuration.

Lemma H.14 (OGS definability). *Taking an initial active* OGS *configuration* $\mathbb{G} = \langle \mathtt{I}; \Gamma \vdash d[\mathtt{t}] : \bot \rangle_{\mathrm{ogs}}$, *then for all passive* OGS *configuration* $\mathbb{H}$ *such that* $\mathbb{G} \perp\!\!\!\perp \mathbb{H}$, *there exist:*

- *a continuation name:* $c_{\mathtt{f}}$
- *an evaluation context:* $c_{\mathtt{f}} : \neg v \vdash \mathsf{K} : \neg\tau$,
- *a name valuation:* $\mathtt{I}'; \emptyset \vdash v : \Gamma$,

such that $\mathbb{H} = \langle \mathtt{I}'; c_{\mathtt{f}} : \neg v \vdash [d \mapsto \mathsf{K}] \cdot v \rangle_{\mathrm{ogs}}$

Proof. We will write $\mathbb{G} = \langle \mathbb{I} \parallel \mathbb{T} \parallel \mathbb{W} \rangle$ with $\mathbb{I} = \langle d[\mathtt{t}]; \varepsilon; \varnothing \rangle$, $\mathbb{T} = \langle \Gamma; \emptyset \mid \emptyset; \emptyset \rangle$ and $\mathbb{W} = \langle [] \mid \varnothing \mid \emptyset \rangle$. We will also write $\mathbb{H} = \langle \mathbb{J} \parallel \mathbb{S} \parallel \mathbb{U} \rangle$ with $\mathbb{J} = \langle \mathtt{I}'; \gamma; \xi \rangle$ and $\mathbb{U} = \langle \sigma \mid \eta \mid \phi \rangle$.

By complementarity, we have (1) $\mathrm{supp}(d[\mathtt{t}]) \subseteq \mathrm{dom}(\gamma)$, and (2) there exist a type v and continuation name $c_{\mathtt{f}} \in \mathrm{codom}(\gamma)$ s.t. $\mathbb{S} = \langle [c_{\mathtt{f}} \mapsto \neg v]; \emptyset \dashv\vdash \Gamma; \emptyset \rangle$.

(1) ensures the existence of a name valuation v s.t. $\mathrm{dom}(v) = \mathrm{supp}(\mathtt{t})$ and an interactive context K s.t. $\gamma = [d \mapsto \mathsf{K}]v$, whereas (2) entails $\mathtt{I}'; [c_{\mathtt{f}} \mapsto \neg v] \vdash \gamma : \Gamma$ and $\mathrm{dom}(\gamma) = \mathrm{dom}(\Gamma)$. $\mathtt{I}'; c_{\mathtt{f}} : \neg v \vdash \mathsf{K} : \Gamma(d)$ and $\mathtt{I}'; \emptyset \vdash v : \Gamma$.

By compatibility of configurations $\mathbb{J}$ and $\mathbb{U}$, we have $\mathrm{supp}(\phi) \subseteq \mathrm{dom}(v) \cap \mathcal{K}$, and since $\mathrm{dom}(v) = \mathrm{supp}(\mathtt{t})$ and $\mathrm{supp}(\mathtt{t}) \cap \mathcal{K} = \emptyset$, then $\phi = \emptyset$.

By synchronicity, we have $\mathcal{F}(\gamma(d)) \simeq_{\emptyset} \sigma$; that is $\mathcal{F}(\mathsf{K}) = [c_{\mathtt{f}}] = \sigma$, and we also have $\xi = \varnothing$, which again by compatibility entails that $\varnothing \sim_{\mathrm{fwd}} \eta$ and $\eta = \varnothing$.

Finally, we have shown that $\mathbb{H} = \langle \langle \mathtt{I}'; \gamma; \varnothing \rangle \parallel \langle [c_{\mathtt{f}} \mapsto \neg v]; \emptyset \dashv\vdash \Gamma; \emptyset \rangle \parallel \langle [c_{\mathtt{f}}] \mid \varnothing \mid \emptyset \rangle \rangle$, i.e. $\mathbb{H} = \langle \mathtt{I}'; c_{\mathtt{f}} : \neg v \vdash [d \mapsto \mathsf{K}] \cdot v \rangle_{\mathrm{ogs}}$.

□

Corollary H.15 (completeness of observation). *Taking two initial active* OGS *configurations* $\mathbb{G}_1 = \langle \mathtt{I}; \Gamma \vdash_{\mathrm{c}} \mathtt{M}_1 \rangle_{\mathrm{ogs}}$ *and* $\mathbb{G}_1 = \langle \mathtt{I}; \Gamma \vdash_{\mathrm{c}} \mathtt{M}_2 \rangle_{\mathrm{ogs}}$ *such that* $\mathtt{M}_1 \simeq_{ciu} \mathtt{M}_2$, *then:*

$$\forall \mathbb{H} \in \mathcal{O} \text{ s.t. } \mathbb{H} \perp\!\!\!\perp \mathbb{G}_1 \text{ and } \mathbb{H} \perp\!\!\!\perp \mathbb{G}_2.$$
$$\langle \mathbb{H} \perp\!\!\!\perp \mathbb{G}_1 \rangle \Downarrow [c]\mathbf{ret}\,\langle\rangle \text{ if and only if } \langle \mathbb{H} \perp\!\!\!\perp \mathbb{G}_2 \rangle \Downarrow [c]\mathbf{ret}\,\langle\rangle$$

H.4 Interpretation of observation

We have given a sensible notion of observation at the level of OGS configurations, it remains to show what does this predicate mean at the level of OGS interpretations.

Theorem H.16 (full observation). *Given two composable configurations* $\mathbb{G}$ *and* $\mathbb{H}$, *we have:*

$$\langle \mathbb{H} \perp\!\!\!\perp \mathbb{G} \rangle \Downarrow [c]\mathbf{ret}\,\langle\rangle \text{ if and only if } \exists \mathtt{t} \in \mathbf{CTr}_{\mathrm{ogs}}(\mathbb{G}).\ \mathtt{t}^{\perp}\, \boldsymbol{c}.\langle \mathbf{ret}\,\langle\rangle \mid \Box \rangle^{\oplus} \in \mathbf{CTr}_{\mathrm{ogs}}(\mathbb{H})$$

Proof. We write $\mathbb{G} = \langle \mathbb{I} \parallel \mathbb{T} \parallel \mathbb{W} \rangle$ and $\mathbb{H} = \langle \mathbb{J} \parallel \mathbb{S} \parallel \mathbb{U} \rangle$ and $\mathbb{I} = \langle \mathtt{t}; \gamma_{\mathbb{I}}; \xi \rangle$

Left to right. Suppose there exist $\mathbb{K}, \mathbb{J} \in \mathcal{A}$ and γ such that $\langle \mathbb{J} \dashv\vdash \mathbb{I} \rangle \Downarrow_{\mathrm{ci}} \langle \mathbb{K} \dashv\vdash \mathbb{J} \rangle$ with $\mathbb{J} = \langle c[\mathbf{ret}\,\langle\rangle]; \gamma; \varnothing \rangle$.

By corollary E.5 and definition E.3, there exists an odd-length trace t such that $\mathbb{I} \overset{\mathtt{t}}{\Rightarrow}_{\mathrm{I}} \mathbb{K}$ and $\mathbb{J} \overset{\mathtt{t}^{\perp}}{\Longrightarrow}_{\mathrm{I}} \mathbb{J} \overset{\boldsymbol{c}.\langle \mathbf{ret}\,\langle\rangle \mid \Box \rangle^{\oplus}}{\Longrightarrow}_{\mathrm{I}} \langle \gamma; \varnothing \rangle = \mathbb{J}'$. By lemma H.3, there exist $\mathbb{T}', \mathbb{S}', \mathbb{S}'' \in \mathcal{T}$

s.t. $\mathbb{T} \xrightarrow{\mathsf{t}}_{\mathcal{T}} \mathbb{T}'$, $\mathbb{S} \xrightarrow{\mathsf{t}^\perp}_{\mathcal{T}} \mathbb{S}'$ $\xrightarrow{\boldsymbol{c}.\langle \mathbf{ret}\,\langle\rangle | \Box\rangle^\oplus}_{\mathcal{T}} \mathbb{S}''$, $\mathbb{K} \wr \mathbb{T}'$, $\mathbb{J}' \wr \mathbb{S}'$ and $\mathbb{J}' \wr \mathbb{S}''$. Similarily, corollary H.11 ensures the existence of $\mathbb{W}', \mathbb{U}', \mathbb{U}'' \in \mathcal{W}$ s.t. $\mathbb{W} \xrightarrow{\mathsf{t}}_{\mathcal{WB}} \mathbb{W}'$, $\mathbb{U} \xrightarrow{\mathsf{t}^\perp}_{\mathcal{WB}}$ $\mathbb{U}' \xrightarrow{\boldsymbol{c}.\langle \mathbf{ret}\,\langle\rangle | \Box\rangle^\oplus}_{\mathcal{WB}} \mathbb{U}''$ whereas lemma H.10 ensures that synchronicity is preserved and that in particular $\langle \mathbb{J}, \mathbb{U}' \rangle \mathcal{S} \langle \mathbb{K}, \mathbb{W}' \rangle$.

Thus we have shown that $\mathbb{G} \overset{\mathsf{t}}{\underset{\text{ogs}}{\Rightarrow}} \langle \mathbb{K} \parallel \mathbb{T}' \parallel \mathbb{W}' \rangle$ and $\mathbb{H} \overset{\mathsf{t}^\perp}{\underset{\text{ogs}}{\Longrightarrow}} \langle \mathbb{J} \parallel \mathbb{S}' \parallel \mathbb{U}' \rangle \overset{\boldsymbol{c}.\langle \mathbf{ret}\,\langle\rangle | \Box\rangle^\oplus}{\underset{\text{ogs}}{\Longrightarrow}} \langle \mathbb{J}' \parallel \mathbb{S}'' \parallel \mathbb{U}'' \rangle$, from which we can deduce that $\mathsf{t} \in \mathbf{Tr}_{\text{ogs}}(\mathbb{G})$ and $\mathsf{t}^\perp \boldsymbol{c}.\langle \mathbf{ret}\,\langle\rangle \mid \Box\rangle^\oplus \in \mathbf{Tr}_{\text{ogs}}(\mathbb{H})$.

Right to left. Let $\mathsf{t} \in \mathbf{Tr}_{\text{ogs}}$ such that $\mathsf{t} \in \mathbf{Tr}_{\text{ogs}}(\mathbb{G})$ and $\mathsf{t}^\perp \boldsymbol{c}.\langle \mathbf{ret}\,\langle\rangle \mid \Box\rangle^\oplus \in \mathbf{Tr}_{\text{ogs}}(\mathbb{H})$.

If we write $\mathbb{G} \overset{\mathsf{t}}{\underset{\text{ogs}}{\Rightarrow}} \langle \mathbb{K} \parallel \mathbb{T}' \parallel \mathbb{W}' \rangle$ and for some $\mathbb{W}' \in \mathcal{W}$ with $\mathbb{K} = \langle \mathtt{I}_{\mathbb{K}}; \gamma_{\mathbb{K}}; \varnothing \rangle \in \mathcal{A}$, and write $\mathbb{H} \overset{\mathsf{t}^\perp}{\underset{\text{ogs}}{\Longrightarrow}} \langle \mathbb{J} \parallel \mathbb{S}' \parallel \mathbb{U}' \rangle \xrightarrow{\boldsymbol{c}.\langle \mathbf{ret}\,\langle\rangle | \Box\rangle^\oplus}_{\text{ogs}} \mathbb{E}$ for some $\mathbb{S}' \in \mathcal{T}, \mathbb{U}' \in \mathcal{W}$ and $\mathbb{E} \in \mathcal{O}$ with $\mathbb{J} = \langle c[\mathbf{ret}\,\langle\rangle]; \gamma; \varnothing \rangle$, then by corollary E.5, we have $\langle \mathbb{J} \dashv\vdash \mathbb{I} \rangle \mapsto^*_{\text{ci}} \langle \mathbb{J} \dashv\vdash \mathbb{K} \rangle$ and again by lemma E.2, we have $\mathbb{J} \dashv\vdash_c \mathbb{K}$, and in particular $c \notin \text{dom}(\gamma_{\mathbb{J}})$, thus $\langle \mathbb{J} \dashv\vdash \mathbb{K} \rangle \not\mapsto_{\text{ci}}$. Finally, we have shown that $\langle \mathbb{J} \dashv\vdash \mathbb{I} \rangle \Downarrow_{\text{ci}} \langle \mathbb{J} \dashv\vdash \mathbb{K} \rangle$ and consequently $\langle \mathbb{H} \amalg \mathbb{G} \rangle \Downarrow [c]\,\mathbf{ret}\,\langle\rangle$.

□