Two Sides of the Same Coin: Session Types and Game Semantics

A Synchronous Side and an Asynchronous Side

SIMON CASTELLAN, Imperial College London, United Kingdom
NOBUKO YOSHIDA, Imperial College London, United Kingdom

Game semantics and session types are two formalisations of the same concept: message-passing open *programs* following certain *protocols*. Game semantics represents protocols as *games*, and programs as *strategies*; while *session types* specify protocols, and well-typed π*-calculus processes* model programs. Giving faithful models of the π-calculus *and* giving a precise description of strategies as a programming language are two difficult problems. In this paper, we show how these two problems can be tackled at the same time by building an accurate game semantics model of the session π-calculus.

Our main contribution is to fill a semantic gap between the synchrony of the (session) π-calculus and the asynchrony of game semantics, by developing an event-structure based game semantics for synchronous concurrent computation. This model supports the first truly concurrent fully abstract (for barbed congruence) interpretation of the synchronous (session) π-calculus. We further strengthen this correspondence, establishing finite definability of asynchronous strategies by the internal session π-calculus. As an application of these results, we propose a faithful encoding of synchronous strategies into asynchronous strategies by call-return protocols, which induces automatically an encoding at the level of processes. Our results bring session types and game semantics into the same picture, proposing the session calculus as a programming language for strategies, and strategies as a very accurate model of the session calculus. We implement a prototype which computes the interpretation of session processes as synchronous strategies.

CCS Concepts: • **Theory of computation → Process calculi**; **Denotational semantics**;

Additional Key Words and Phrases: π-calculus, session types, event structures, game semantics

ACM Reference Format:
Simon Castellan and Nobuko Yoshida. 2019. Two Sides of the Same Coin: Session Types and Game Semantics: A Synchronous Side and an Asynchronous Side. *Proc. ACM Program. Lang.* 3, POPL, Article 27 (January 2019), 57 pages. https://doi.org/10.1145/3290340

1 INTRODUCTION

Over the last 25 years, game semantics [Abramsky et al. 2000, 1994; Hyland and Ong 1994, 2000] has been used as a versatile framework for constructing the first syntax-independent (often fully-abstract) models for a variety of programming languages, exhibiting a wide range of computational effects such as, *e.g.*., state [Abramsky et al. 1998; Abramsky and McCusker 1999], control [Laird 1997], concurrency [Laird 2001], nondeterminism [Harmer and McCusker 1999] and probabilities [Danos and Harmer 2002]. This versatility arises from an *interactive* interpretation of computation. An open (higher-order) program is modelled by a *process* (called *strategy*) interacting with its

Authors' addresses: Simon Castellan, Imperial College London, London, United Kingdom, s.castellan@ic.ac.uk; Nobuko Yoshida, Imperial College London, London, United Kingdom, n.yoshida@ic.ac.uk.

2475-1421/2019/1-ART27
https://doi.org/10.1145/3290340

environment according to a *protocol* (described by a *game*). In particular, the game only depends on the *type* of the program (the interface between the program and the environment), and not on the programming language it is written in.

It has also been 25 years since another framework, *session types* [Honda et al. 1998; Takeuchi et al. 1994], was proposed for codifying communication structures using protocols in concurrent, message-passing processes. This suggests the following analogy:

	Game semantics	**Session π-calculus**
protocols	games	types
open programs	strategies	processes

In both cases, protocols are two-party and described from the point of view of one of the participant. Actions are thus polarised: output actions correspond to Player moves, and input actions to Opponent moves. Reversing this polarity leads in both cases to a central notion of duality: two agents can only interact on dual types/games. According to [Hyland and Ong 1995], this analogy captures "every essential aspect of the dialogue game paradigm so precisely that the π-calculus presentation may as well be taken to be a formal *definition*". In particular, this insight led to, *e.g.*, the verification of type soundness for various programming constructs [Disney and Flanagan 2015]; and a typing discipline characterising sequential computation in the π-calculus [Berger et al. 2001].

In spite of these well-known conceptual similarities, there is no work building a *precise* connection between any dialect of the π-calculus and game semantics. Having an exact semantic correspondence between these two worlds would be mutually beneficial: session processes could provide a syntactic description of strategies as message-passing programs (an open problem for concurrent game semantics); and game semantics could offer a canonical denotational (categorical) semantics of session processes and give semantic proofs of properties (such as deadlock-freedom).

To have a close operational correspondence between processes and strategies, traditional play-based strategies do not fit: they forget the nondeterministic branching point, a feature necessary to obtain full abstraction for observational equivalence (barbed congruence) [Honda and Yoshida 1995; Milner and Sangiorgi 1992]. This prompts us to base our work on *concurrent games*, a framework for games and causal strategies based around ideas from concurrency theory. Initiated by [Abramsky and Melliès 1999; Melliès 2005; Melliès and Mimram 2007], this family of game semantics has been actively developed recently, prompted by new foundations using event structures introduced in [Rideau and Winskel 2011]. This representation of strategies in terms of event structures is able to remember the branching point, and, as an added benefit, also represents games as event structures, which support a natural interpretation of session types unlike traditional arena-based games [Hyland and Ong 1994]. This interpretation of session types explicits a striking connection between games and session types, describing protocols using the same constructs (sequencing, parallel composition and duality).

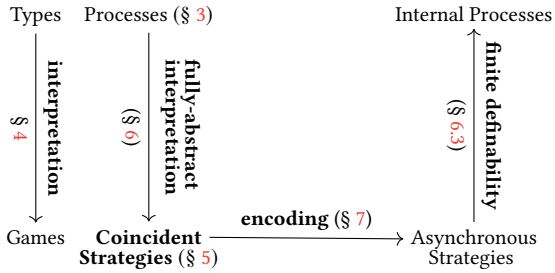
However, to build a formal relationship between the session π-calculus and game semantics based on event structures, there is still a conceptual gap to overcome. One the one hand, traditional session processes are *synchronous* where two processes communicate via hand-shakes, synchronising immediately. On the other hand, concurrent game semantics is inherently *asynchronous*. In game semantics, a ubiquitous agent, *copycat*, plays a key role to compositionally interpret programs. In the π-calculus, copycat corresponds to the *forwarder* or *link agent* [Honda and Yoshida 1995; Sangiorgi 1996]) which identifies (or links) two channels by forwarding data back and forth.

For the interpretation to be sound, copycat must be a categorical identity which implies that strategies must be invariant under composition by this agent. Since composition with copycat breaks syntactic dependences from an output, or to an input on different channels (forcing stronger asynchrony properties than the asynchronous π-calculus [Honda and Tokoro 1991] which only

forbids dependences from an output), strategies cannot represent synchronous processes. This phenomenon is due to the delay that copycat adds between the reception of a move, and its forwarding. We say that this copycat is *asynchronous*.

Our main contribution is to create a model where there also exists a *synchronous* copycat where the forwarding is instantaneous. In this model, strategies can play several moves *at the same time* (rather than in an unspecified order) in a ***coincidence***. Traditional models of true concurrency are coincidence-free: two events can always be separated by a partial execution containing one but not the other. Our main contribution is ***coincident event structures***, an extension of prime event structures [Winskel 1986] allowing coincidences, obtained by relaxing the causal order to a *causal preorder*. We show that coincident event structures support an intensionally fully-abstract game semantics interpretation of the session π-calculus, interpretation devised using the techniques of [Castellan et al. 2018; Rideau and Winskel 2011].

Contribution and outline. In § 2, the paper starts by an illustration of the correspondence and of the difficulties of bringing these two worlds together. We then introduce the synchronous session π-calculus in § 3. Our contributions start in § 4 with an interpretation from session types to games, which will be driving the interpretation of processes. We show that a large class of games are in the image of the translation, hence session types can be seen a game description language. In § 5, we introduce coincident event structures, which are then used to define coincident strategies. They extend the strategies of [Castellan et al. 2018] and form a category without requiring any asynchrony conditions on the strategies. In § 6, we model the session π-calculus inside coincident strategies and show that this interpretation is intensionally fully abstract (Theorem 6.5). We then show that *finite* strategies of [Castellan et al. 2018] are definable using *internal session processes* (processes that do not send free names), hence exhibiting a natural programming language for them (Theorem 6.10). Finally, in § 7, we show that our category of coincident strategies is isomorphic to a subcategory of the concurrent strategies of [Castellan et al. 2018]. Via the result of finite definability, this yields a sound translation from session processes to internal session processes, which are invariant under composition with asynchronous forwarders. This outline is summarised by the diagram on the left, where text in bold denotes our contributions.



The translation from processes (without recursion) to coincident event structures has been implemented. A version is available at http://sessiontypesandgames.github.io/

All the proofs and more examples can be found in appendix.

2 OVERVIEW OF THE MODEL

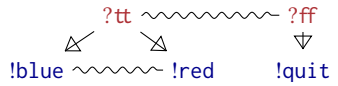
This section informally explains the problem of relating session types and game semantics, and the reason why we introduce ***coincident strategies***. This section does not require advanced knowledge of session types, event structures or game semantics. We first illustrate the correspondence (§ 2.1) and then explain the challenges to overcome to faithfully interpret processes as strategies (§ 2.2).

2.1 Illustration of the Correspondence

2.1.1 Selection and Branching. In (two-party) session types and game semantics, protocols are described from the viewpoint of one particular participant. For instance, a simple protocol "receive

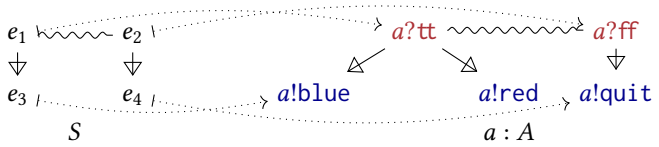
a boolean; if received true, then send a colour; else send a quit message" is represented in the session π-calculus, by the type T (left); or in game semantics by the game A:

$$T = ?\mathtt{tt}.\,(!\mathtt{blue} \oplus !\mathtt{red}) \,\&\, ?\mathtt{ff}.\,!\mathtt{quit} \qquad A =$$



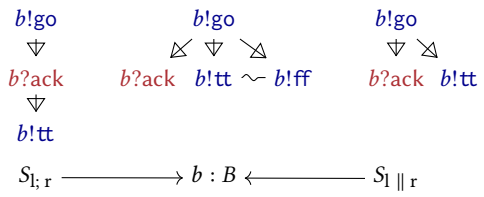
In the session type, tt, ff, … denote the *labels*, representing the messages exchanged. The symbols !/? denote the ***polarity*** of the action (input/output); ⊕/&, the polarity of the choice (internal/external); and dots, sequencing ("then"). On the other side, the game describes every message exchanged as a move, and organises them as an ***event structure***, which consists in a partial order $\rightarrowtriangle$, *causality*, and a relation $\sim$, *conflict*. Causality of the game describes the sequencing in the protocol; and conflict the possible choices of messages for one particular action. Moves of a game can be decorated by any relevant information, *e.g.* in this case A is decorated by messages inherited from the protocol.

In the π-calculus, processes exchange messages and synchronise on *channels*, while in game semantics strategies play and synchronise on *moves*. To implement this protocol as a process, we bind the type T to a channel a, and write a well-typed process on the typing environment $a : T$, for instance: $\vdash P := (a?\mathtt{tt}.\,a!\mathtt{blue}) \,\&\, (a?\mathtt{ff}.\,a!\mathtt{quit}) \triangleright a : T$, reading as: if tt is received on a, then send blue on a, or (&) if ff is received, then send quit. In game semantics based on event structures, a strategy on a game will be an event structure S along with a labelling function to the game. In this case, the interpretation of P will play on the interpretation of the context $a : T$ obtained from A by decorating the moves with the channel a as depicted on the right of the diagram below. The diagram on the left represents the strategy corresponding to P along with its labelling function.



In general, we will omit the labelling function and draw the events as their labels.

2.1.2 Parallelism and Multiple Sessions. The previous protocol has such a simple structure that implementations of it must be sequential. To allow for concurrent implementations, the protocols must leave the order between some actions unspecified. For instance in the protocol "send go, and then receive an acknowledgement and send a boolean in any order". There are two independent subparts: receiving the acknowledgement, and sending the boolean. The corresponding game will have two disjoint subgames for these, giving rise to a game B. The diagram on the left depicts the game $b : B$ (middle) as well as two strategies on it: one where the boolean is only sent after reception of the acknowledgement (left); and one where the boolean is sent without waiting for the acknowledgement (right). In $b : B$, the event $b?\mathtt{ack}$ is not comparable, and not in conflict with the events $b!\mathtt{tt}$ and $b!\mathtt{ff}$: they are *concurrent*.



In the session π-calculus, two independent parts of the protocols will occur on different channels. However, at the beginning of the session there is only one channel, say b. Since after the output of the go message, there are now two independent subprotocols, there must also be two channels. In the π-calculus, the second name is sent as a payload of the message go. This gives two possible

representations of this protocol as a session type (with the same game representation, namely B), depending on which part is fulfilled by the payload channel:

$$T = !\mathsf{go}\langle !\mathsf{ack}\rangle.(!\mathtt{tt} \oplus !\mathtt{ff}) \qquad T' = !\mathsf{go}\langle ?\mathtt{tt} \,\&\, ?\mathtt{ff}\rangle.(?\mathsf{ack}).$$

The syntax $!\mathsf{go}\langle S_1\rangle.S_2$ means send a message go along with a channel on which the *receiver* will perform S_1, and continue as S_2. Because S_1 is from the point of view of the receiver, there is a natural duality in the type of the payload channel for outputs. The strategies depicted above are the interpretations of the processes:

$$P_{\mathrm{l};\,\mathrm{r}} = (\nu c)(b!\mathsf{go}\langle\overline{c}\rangle.c!\mathsf{ack}.\, b!\mathtt{tt}) \qquad P_{\mathrm{l}\,\|\,\mathrm{r}} = (\nu c)(b!\mathsf{go}\langle\overline{c}\rangle.(c!\mathsf{ack} \mid b!\mathtt{tt})) \qquad \text{on context } b : T$$

These two processes start with a restriction (νc), creating a pair of connected channels c and $\overline{c}$ with dual types !ack and ?ack respectively, and proceed to send $\overline{c}$ on b and continue.

In both these processes, the name c sent is created just before the sent. In the π-calculus, such processes are called ***internal*** [Sangiorgi 1996] and sometimes abbreviated as:

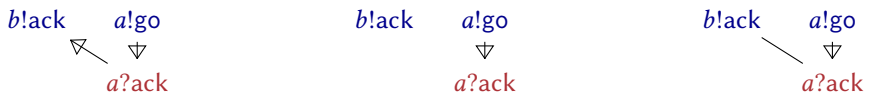
$$P_{\mathrm{l};\,\mathrm{r}} = b!\mathsf{go}(c).c!\mathsf{ack}.\, b!\mathtt{tt} \qquad P_{\mathrm{l}\,\|\,\mathrm{r}} = b!\mathsf{go}(c).(c!\mathsf{ack} \mid b!\mathtt{tt}) \qquad \text{on context } b : T$$

In § 6.3, we show that internal processes define a large class of event structures.

2.2 Discrepancy between Processes and Strategies

We now detail the obstacles in the way of giving a fully abstract semantics of session processes in terms of strategies. The main obstacle is the gap between synchrony and asynchrony detailed in § 2.2.1. We then explain why capturing both causal dependences (§ 2.2.2) and the nondeterministic branching point (§ 2.2.3) is also essential, and the solutions given by [Rideau and Winskel 2011] and [Castellan et al. 2018] respectively.

2.2.1 Synchrony and Asynchrony: Copycat. A key feature of the π-calculus is *free* name passing. In § 2.1.2, we only mentioned *bound* name passing. For instance, the process $a!\mathsf{go}\langle b\rangle$ is a well-typed session process on the typing environment $b : !\mathsf{ack}, a : !\mathsf{go}\langle !\mathsf{ack}\rangle$. The typing environment contains two protocols "send an acknowledgement" (on b), and "send go, and then receive ack" (on a), whose game interpretation is depicted in (1b) below. The meaning of $a!\mathsf{go}\langle b\rangle$ should be that the acknowledgement on b is sent exactly when the receiver of the go message writes on the payload channel. This leads to the strategy in (1a), which is the way that traditional game semantics interprets free name passing [Laird 2005]. The forwarding between b and a is *asynchronous*: intuitively, there can be an arbitrary delay between the input on a and the output on b. In fact, this strategy is also the interpretation of the internal process $(\nu c)(a!\mathsf{go}\langle\overline{c}\rangle.\, c?\mathsf{ack}.\, b!\mathsf{ack})$. The expression $c?\mathsf{ack}.\, b!\mathsf{ack}$ is the *asynchronous forwarder* between b and c for type !ack.



(1a) Asynchronous forwarding (1b) Game for $b : ?\mathsf{ack}, a : !\mathsf{go}\langle ?\mathsf{ack}\rangle$ (1c) Synchronous forwarding

In the session π-calculus, the two processes $(\nu c)(a!\mathsf{go}\langle\overline{c}\rangle.\, c?\mathsf{ack}.\, b!\mathsf{ack})$ and $a!\mathsf{go}\langle b\rangle$ are not observationally equivalent (see Example E.1 in Appendix for more details), therefore it is not possible to devise a fully abstract interpretation this way. In our model, we keep the same idea (to represent free name passing via a forwarder), and use a *synchronous forwarder* instead which makes sure that the input on a will occur at the same time (observationally) than the output on x: these two moves are *coincident*, depicted with the line between them. (1c) depicts our interpretation of $a!\mathsf{go}\langle b\rangle$.

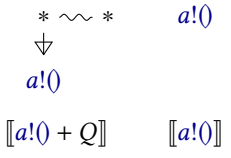
2.2.2 Causality of Processes. Because our model is based on event structures, it computes the causality between the actions of processes. This is important to obtain a fully abstract model, because the processes $a!\mathtt{tt} \mid b!\mathtt{tt}$ and $a!\mathtt{tt}.\, b!\mathtt{tt} + b!\mathtt{tt}.\, a!\mathtt{tt}$ have the same traces but are not barbed congruent. A causal model is necessary to distinguish them.

The main difficulty in building a causal model of the session π-calculus is interpreting the restriction operator. Without it, the event structures expressible by processes are simply forests. With it, every causal pattern becomes expressible (see § 6.3). We give here a little example of a non tree-like causal behaviour. Consider the simple session type $\mathbf{1} = !()$ (send a dummy message ()) and its dual $\bot = ?()$ (receive a dummy message), whose interpretation in term of games are simply the one-event games $!()$ and $?()$; and consider the process $J = (\nu d)(a?().\, \overline{d}!() \mid b?().\, d?().\, c!())$, well-typed in the environment $a : \bot, b : \bot, c : 1$. This process implements a simple causal behaviour: wait for input on a and b, and only then output on c. Our interpretation of J exactly expresses this:

$$[\![(\nu d)(a?().\, \overline{d}!() \mid b?().\, d?().\, c!())]\!] = \quad \begin{array}{ccc} a?() & & b?() \\ & \searrow \quad \swarrow & \\ & c!() & \end{array} \quad \text{on the game} \quad a?() \quad b?() \quad c!()$$

One of the main contributions of [Rideau and Winskel 2011] is to introduce the ***interaction*** of strategies, which is crucial to define the interpretation of the restriction operator (see § 5.3).

2.2.3 Nondeterministic Branching Point. Causality is not enough in presence of nondeterminism. Indeed, the model of [Rideau and Winskel 2011] although being causal only supports *angelic nondeterminism*, which means that nondeterministic branches that deadlock are forgotten.



For instance, define $P = a!()$ and $Q = (\nu c)(\nu d)(c?().\, d!().\, \overline{a}!() \mid \overline{d}?().\, \overline{c}!())$ (a deadlocking process). Then $P + Q$ and P are both typable by the same typing environment but they are not observationally equivalent. A model based on [Rideau and Winskel 2011] would however equate these processes by interpreting both by the strategy $a!()$. Using the recent methodology in [Castellan et al. 2018], we can define strategies with *internal actions* (written $*$) which can remember that some branch may deadlock. By remembering the nondeterministic choice, the interpretations of $P + Q$ and P, given on the left, are not weakly bisimilar even though the interpretation of Q is empty.

3 SESSION TYPES AND SESSION PROCESSES

This section gives a simplification of the most widely studied synchronous session calculus [Honda et al. 1998; Yoshida and Vasconcelos 2007]. Since our main focus is on session communications, we only allow exchanges of linear channels. The calculus is identical to the synchronous calculus presented in [Chen et al. 2017], extended with a nondeterministic choice operator. Despite only sharing linear names, because of recursive types and recursive processes, the language is expressive enough to contain the encoding of nondeterministic PCF.

3.1 Session Processes

The syntax of processes is given in Table 1. We use the following sets: *channel variables*, ranged over by $x, y, z \ldots$; *linear channels*, ranged over by $a, b, c, \overline{a}, \overline{b}, \overline{c} \ldots$; *process variables*, ranged over by $X, Y, \ldots$. We write $\mathcal{N}$ for the set of *identifiers* (channel variables and linear channels), ranged over by $u, u', \ldots$; and $\mathscr{L}$ for the *finite* set of possible labels, ranged over by $l, l', \ldots$ The notation $\widetilde{v}$ denotes a vector $(v^1, \ldots, v^n)$ of values; similarly for others, for instance $\widetilde{T}$ denotes a sequence of types. We write $\overline{a}$ is a *co-channel* of a, which represents a dual end-point of a; we assume $\overline{\overline{a}} = a$.

Table 1. Syntax of processes and types.

u	::=	**Identifiers**				
		$a, b, c, \overline{a}, \overline{b}, \overline{c} \ldots$	channel	$\mid$	$x, y, z, \ldots$	variable
P	::=	**Process**				
		$\mathbf{0}$	nil	$\mid$	X	variable
	$\mid$	$\&_{i\in I}\, u?l_i(\widetilde{x_i}).P_i$	input	$\mid$	$u!l\langle\widetilde{v}\rangle.\, P$	output
	$\mid$	$P \mid P$	parallel	$\mid$	$\sum_{i\in I}\, P_i$	choice
	$\mid$	$\mu X.\, P$	recursion	$\mid$	$(va)P$	restriction

S, T	::=	**Types**		
		end		end
	$\mid$	$\mathbf{t}$		variable
	$\mid$	$\&_{i\in I}?l_i(\widetilde{S_i}).\, T_i$		branching
	$\mid$	$\oplus_{i\in I}!l_i\langle\widetilde{S_i}\rangle.\, T_i$		selection
	$\mid$	$\mu\mathbf{t}.\, T$		recursion

Table 2. Reduction and structural rules.

$$[\text{R-COM}]\quad \frac{k \in I}{(va)(a!l_k\langle\widetilde{c}\rangle.\, P \mid \&_{i\in I}\, \overline{a}?l_i(\widetilde{x_i}).\, Q_i) \to (va)(P \mid Q_k\{\widetilde{c}/\widetilde{x_k}\})}$$

$$[\text{R-CHOICE}]\quad \frac{k \in I}{\sum_{i\in I} P_i \to P_k} \qquad [\text{R-RES}]\quad \frac{P \to P'}{(vc)P \to (vc)P'} \qquad [\text{R-PAR}]\quad \frac{P \to P'}{P \mid Q \to P' \mid Q} \qquad [\text{R-STRUCT}]\quad \frac{P \equiv P' \to Q' \equiv Q}{P \to Q}$$

$$\begin{aligned}
\mathbf{0} \mid P &\equiv P \\
P \mid Q &\equiv Q \mid P \\
P \mid (Q \mid R) &\equiv (P \mid Q) \mid R \\
\mu X.\, P &\equiv P\{\mu X.\, P/X\} \\
(va)\mathbf{0} &\equiv \mathbf{0} \\
(va)P \mid Q &\equiv (va)(P \mid Q) \quad (a \notin fn(Q)) \\
(va)(vb)P &\equiv (vb)(va)P
\end{aligned}$$

Session communications are performed between an output process and an input process where the former sends a channel choosing one of the branches offered by the latter. The choice process $\sum_{k\in I} P_k$ internally chooses one of P_k with $k \in I$. The process $(va)P$ is a restriction which binds two dual channels, a and $\overline{a}$ in P, making them co-channels, i.e. allowing them to communicate (see rule [R-COM] in Table 2). This restriction is commonly used in the recent literature of session types, e.g. [Gay and Hole 2005; Mostrous and Yoshida 2015; Yoshida and Vasconcelos 2007]. The rest of the syntax is standard. We often omit $\mathbf{0}$ from the tail of processes. The *bindings* for variables are in inputs and those for channels are in restrictions, and those for process variables are in recursions. The derived notions of bound and free channels, channel variables, identifiers, process variables, alpha equivalence, $\equiv_\alpha$, and substitution are standard. We use the Barendregt convention that no bound identifier can occur free or in two different bindings. By $fn(P)$/$\text{fv}(P)$ we denote the set of *free channels* and *variables* in P. We write $a!\langle\widetilde{u}\rangle$ and $a?(\widetilde{u})$ when the labels sent or received are irrelevant; and $a!l$ and $a?l$ when no channel is sent or received.

3.2 Operational Semantics

Table 2 defines the reduction relation and the structural rules. [R-COM] is the main communication rule between input and output at two co-channels a and $\overline{a}$, where the label l_k is selected and channels $\widetilde{c}$ are instantiated into the k-th input branch. [R-CHOICE] internally chooses the k-th process, which makes computations non-deterministic. Other rules are standard. $\to^*$ denotes the reflexive and transitive closure of $\to$.

3.3 Typing Session Processes

The syntax of *session types*, ranged over by T and S, is given in Table 1. The *branching type* $\&_{i\in I}?l_i(\widetilde{S_i}).\, T_i$ describes a channel willing to branch on an incoming label l_i, receive channels of type $\widetilde{S_i}$, and then continue its interaction as prescribed by T_i. The *selection type* $\oplus_{i\in I}!l_i\langle\widetilde{S_i}\rangle.\, T_i$ is its dual: it describes a channel willing to send a label l_i with channels of type $\widetilde{S_i}$, and then continue its

Table 3. Session typing rules.

$$[\text{T-IDLE}]\frac{\Delta \text{ end only}}{\Gamma \vdash \mathbf{0} \triangleright \Delta} \qquad [\text{T-VAR}]\frac{\Delta' \text{ end only}}{\Gamma, X{:}\Delta \vdash X \triangleright \Delta, \Delta'}$$

$$[\text{T-INPUT}]\frac{\forall i \in I, \Gamma \vdash P_i \triangleright \Delta, u : T_i, \widetilde{x_i} : \widetilde{S_i}}{\Gamma \vdash \&_{i\in I}\, u?l_i(\widetilde{x_i}).\, P_i \triangleright \Delta, u : \&_{i\in I}?l_i(\widetilde{S_i}).\, T_i} \qquad [\text{T-OUT}]\frac{\Gamma \vdash P \triangleright \Delta, u : T_k \quad k \in I}{\Gamma \vdash u!l_k\langle\widetilde{v}\rangle.\, P \triangleright \Delta, u : \oplus_{i\in I}!l_i\langle\widetilde{S_i}\rangle.\, T_i, \widetilde{v} : \widetilde{S_k}}$$

$$[\text{T-PAR}]\frac{\Gamma \vdash P_1 \triangleright \Delta_1 \qquad \Gamma \vdash P_2 \triangleright \Delta_2}{\Gamma \vdash P_1 \mid P_2 \triangleright \Delta_1, \Delta_2} \qquad [\text{T-CHOICE}]\frac{\Gamma \vdash P_i \triangleright \Delta}{\Gamma \vdash \sum_{i\in I} P_i \triangleright \Delta} \qquad [\text{T-REC}]\frac{\Gamma, X{:}\Delta \vdash P \triangleright \Delta}{\Gamma \vdash \mu X.\, P \triangleright \Delta} \qquad [\text{T-NEW}]\frac{\Gamma \vdash P \triangleright \Delta, a : T, \overline{a} : \overline{T}}{\Gamma \vdash (\nu a)P \triangleright \Delta}$$

interaction as prescribed by T_i. In branching and in selection types the labels are pairwise distinct; and the types of the exchanged channels are closed. We omit & and $\oplus$ and labels when there is only one branch. We use $\mathbf{t}$ to range over type variables. The type $\mu\mathbf{t}.\, T$ is a *recursive type*. We assume that recursive types are *contractive* (guarded), i.e. $\mu\mathbf{t}_1.\, \mu\mathbf{t}_2 \ldots \mu\mathbf{t}_n.\, \mathbf{t}_1$ is not a type. The type end represents the termination of a session and it is often omitted. We take an equi-recursive view of types considering two types unfolding to the same regular tree as equal [Chen et al. 2017]. We use the same convention as for processes, omitting the label when it is () or the channels when no channels should be sent: *e.g.* $!\mathtt{tt}$ is a shorthand for $!\mathtt{tt}\langle\rangle$, and $!\langle!\mathtt{quit}\rangle$ for $!()\langle!\mathtt{quit}\rangle$. In $\oplus_{i\in I}l_i\langle\widetilde{T_i}\rangle.\, S_i$ and $\&_{i\in I}l_i(\widetilde{T_i}).\, S_i$, S_i is the *continuation type*, while the $\widetilde{T_i}$ are the *argument types* of label l_i.

Session duality [Honda et al. 1998] ensures compatibility of communications. The function $\overline{T}$, defined below, yields the dual of the session type T.

$$\overline{\&_{i\in I}?l_i(\widetilde{S_i}).\, T_i} = \oplus_{i\in I}!l_i\langle\widetilde{S_i}\rangle.\overline{T_i} \quad \overline{\oplus_{i\in I}!l_i\langle\widetilde{S_i}\rangle.\, T_i} = \&_{i\in I}?l_i(\widetilde{S_i}).\, \overline{T_i} \quad \overline{\mathbf{t}} = \mathbf{t} \quad \overline{\mu\mathbf{t}.T} = \mu\mathbf{t}.\, \overline{T} \quad \overline{\text{end}} = \text{end}$$

The session type representing the boolean type is $\mathbb{B} = !\mathtt{tt} \oplus !\mathtt{ff}$, and the unit type $1 = !()$.

The typing judgements take the form: $\Gamma \vdash P \triangleright \Delta$ where Γ is the *recursion environment* which associates process variables to sequences of session types and Δ is the *session environment* which associates identifiers to session types. They are defined by:

$$\Gamma ::= \emptyset \mid \Gamma, X{:}\Delta \qquad \Delta ::= \emptyset \mid \Delta, u{:}T$$

We write $\Delta, u{:}T$ for $\Delta \cup \{u{:}T\}$ if $u \notin dom(\Delta)$ and Δ_1, Δ_2 for $\Delta_1 \cup \Delta_2$ when $dom(\Delta_1) \cap dom(\Delta_2) = \emptyset$. We say that Δ is *end-only* if $u : T \in \Delta$ implies $T = \text{end}$.

Table 3 gives the typing rules. They are essentially the same as [Chen et al. 2017]. Rule [T-IDLE] is the introduction rule for the nil process. To type an input process, rule [T-INPUT] requires the type S_i^j for variable x_i^j and the type T_i of channel u for the continuation P_i. In the resulting session environment, the type u has the branching type in which u receives S_i^j and then continues with T_i for each label l_i. The rule for typing output processes is dual. In rule [T-PAR], the session environment of $P_1 \mid P_2$ is the disjoint union of the environments Δ_1 and Δ_2 for the two processes, reflecting the linear nature of channels. In contrast, in rule [T-CHOICE], all of the processes have the same session environment because at most one of them will be executed. Rules [T-VAR] and [T-REC] type recursive processes, where recursive variables are typed by a session context. Rule [T-NEW] is a standard rule for name binding, where we ensure the co-channels have dual types. The type system enjoys

the standard subject reduction property. Notice that session environments are unchanged since reduction only occurs under a restriction.

Theorem 3.1 (Theorem 2.2 in [Chen et al. 2017]). *If* $\Gamma \vdash P \triangleright \Delta$ *and* $P \longrightarrow^* Q$, *then* $\Gamma \vdash Q \triangleright \Delta$.

3.4 Behavioural Theory

We present now the most used behavioural equivalence, *reduction-based barbed congruence* [Honda and Yoshida 1995; Milner and Sangiorgi 1992], for synchronous session processes. This congruent relation is defined as the reduction-based bisimulation closure with the minimum observability (*barbs*). A process $\vdash P \triangleright \Delta$ has a barb on b [Milner and Sangiorgi 1992], written $P \downarrow_b$ when:

$$P \equiv (\nu \widetilde{a})(b!l\langle \widetilde{v} \rangle.\, P' \mid Q) \qquad \text{with } b, \overline{b} \notin \{\widetilde{a}\}.$$

The side condition $\overline{b} \notin \{\widetilde{a}\}$ ensures that Q does not contain a process which must interact with $b!l\langle \widetilde{v} \rangle.\, P'$ (note that b is linear: if the dual process at $\overline{b}$ is composed, the observer cannot observe b even b is unrestricted [Kouzapas and Yoshida 2015]). We define $\vdash P \triangleright \Delta \Downarrow_a$ if there exists Q such that $P \longrightarrow^* Q$ and $\vdash Q \triangleright \Delta \downarrow_a$. The set of contexts is defined as:

$$C \quad ::= \quad \_ \mid P \mid (C \mid P) \mid (P \mid C) \mid \&_{i \in I}\, v?l_i(\widetilde{x_i}).C_i \mid u!l\langle \widetilde{v} \rangle.C \mid \sum_{i \in I} C_i \mid \mu X.C \mid (\nu a)C$$

$C[P]$ substitutes process P to each hole $(\_)$ in context C. We define the reduction-closed congruence based on the definition of barb and [Honda and Yoshida 1995; Kouzapas and Yoshida 2015] as a *typed relation*: a relation $\mathcal{R}$ on processes relating closed terms (*i.e.*, $\mathrm{fv}(P_i) = \emptyset$), typed on the same environment. We write $\vdash P_1 \mathcal{R} P_2 \triangleright \Delta$ for $(P_1, P_2) \in \mathcal{R}$ and $\vdash P_i \triangleright \Delta$.

Definition 3.2 (Reduction-closed congruence). A typed relation $\mathcal{R}$ is a *reduction-closed congruence* if it satisfies the following conditions for each $\vdash P_1 \mathcal{R} P_2 \triangleright \Delta$:

(1) $\vdash P_1 \triangleright \Delta \Downarrow_u$ iff $\vdash P_2 \triangleright \Delta \Downarrow_u$.
(2) • $P_1 \longrightarrow^* P_1'$ implies that there exists P_2' such that $P_2 \longrightarrow^* P_2'$ and $\vdash P_1' \mathcal{R} P_2' \triangleright \Delta$
    • the symmetric case.
(3) For all closed context C and all Δ' such that $\vdash C[P_1], C[P_2] \triangleright \Delta'$, we have $\vdash C[P_1]\ \mathcal{R}\ C[P_2] \triangleright \Delta'$.

The union of all reduction-closed congruence relations is denoted as $\simeq$.

3.5 Internal Session Processes

We also study a subset of the session calculus called the internal π-calculus [Sangiorgi 1995] where the output only passes private (bound) names. This means that a process is *internal* when for every output $a!l\langle \widetilde{b} \rangle$ of P, $\widetilde{b}$ is a sequence of names restricted immediately before the output. We write $u!l(\widetilde{b}).\, P$ for $(\nu \widetilde{b})u!l\langle \widetilde{b} \rangle.\, P$. Using structural congruence, the reduction rule [R-Com] instantiates to:

$$\text{[R-Com-I]} \quad (\nu a)(a!l_k(\widetilde{c}).\, P \mid \&_{i \in I}\, \overline{a}?l_i(\widetilde{x_i}).\, Q_i) \longrightarrow (\nu a\widetilde{c})(P \mid Q_k\{\widetilde{c}/\widetilde{x_k}\}) \quad (k \in I)$$

Internal processes will correspond to the standard (pre-)strategies in game semantics (§ 6.3).

Example 3.3. An example of well-typed internal process that will be useful to us later, is the ***asynchronous copycat***. Given a type T (without recursion here), one can define a process $\vdash [u = v]_T \triangleright u : T, v : \overline{T}$ by induction on T (where $[\widetilde{u} = \widetilde{v}]_{\widetilde{T}}$ means $[u^1 = v^1]_{T^1} \mid \cdots \mid [u^n = v^n]_{T^n}$):

$$[u = v]_{\&_{i \in I}?l_i(\widetilde{S}_i).T_i} = \sum_{i \in I} u?l_i(\widetilde{u_i}).\, v!l_i(\widetilde{v_i}).\, ([\widetilde{u_i} = \widetilde{v_i}]_{\widetilde{S}_i} \mid [u = v]_{T_i})$$

$$[u = v]_{\mathrm{end}} = \mathbf{0} \qquad [u = v]_T = [v = u]_{\overline{T}} \text{ (for selection types)}$$

4 TYPES AND GAMES

In this section, we define and study the interpretation of types as the games of our model. In § 4.1, we define event structures which are the basis for the notion of games introduced in [Rideau and Winskel 2011]. In § 4.2, we define games and the interpretation of session types inside them.

4.1 Event Structures

Our model interprets processes and types as causal structures, which allows us to compute the dependences between actions. Due to (internal and external) choice, the causal structure needs to account for nondeterminism as well, representing the possible outcomes of these choices. This leads us to Winskel's event structures [Winskel 1986]. In this paper, we use ***prime event structures with binary conflict***, simply called event structures.

Definition 4.1. An ***event structure*** is a triple $(E, \leq_E, \#_E)$ where $(E, \leq_E)$ is a partial order and $\#_E \subseteq E \times E$ is a binary relation which is symmetric and irreflexive, such that:

Finite causes For $e \in E$, the set $[e] = \{e' \in E, e' \leq e\}$ is finite.
Conflict inheritance If $e \#_E e'$ and $e \leq e_0$ then $e_0 \#_E e'$.

The partial order indicates causal relationships between events: $e \leq e'$ when e' causally depends on e. The conflict relation indicates which events are ***incompatible***, *i.e.*, cannot occur together in the same execution. As a result, event structures take a global view of the program: the set E is the set of all events of the program. This feature is crucial for the construction of the model and differs from other main models of concurrency such as pomsets [Pratt 1984] and presheaves [Cattani and Winskel 1997].

Notations. Given a subset $X \subseteq E$ we write $[X] = \{e' \in E \mid \exists e \in X, e' \leq e\}$. A subset X is downclosed when $X = [X]$. We also write $[X) = [X] \setminus X$, and in particular, $[e) = [e] \setminus \{e\}$. If $e < e'$ with no events in between, we write $e \rightarrowtriangle e'$ and say that e' ***immediately depends*** on e. We say that X is ***consistent*** if for all $e, e' \in X$, $(e, e') \notin \#_E$. The second axiom of event structures forces conflict to propagate upwards: if $e \leq e'$, then e' inherits all the conflicts of e (and on top of those, may have some more of its own). A conflict $e \#_E e'$ is ***minimal*** if there is no $e_0 \leq e$ and $e_0' \leq e'$ with $e_0 \#_E e_0'$. In this case we write $e \sim\!\!\sim e'$. In diagrams, we only represent $\rightarrowtriangle$ and $\sim\!\!\sim$ as they are enough to recover the event structure. A ***configuration*** (representing a partial execution, or a history) of E is a consistent and downclosed subset $x \subseteq E$. The set of *finite* configurations is written $\mathscr{C}(E)$. A configuration x makes a transition to y with event e, written $x \overset{e}{\text{—}\!\subset} y$ (or simply $x – \!\subset y$) when $e \notin x$ and $y = x \cup \{e\}$.

Labelled event structures. When trying to model languages with event structures, each element represents a computational event which may have some visible information (*e.g.*, for an event representing a channel output, the channel and message sent). This information is traditionally represented by *labels*, gathered in a set Σ. In this setting, to account faithfully for divergences and nondeterminism, we also keep track of internal choices. Processes are thus modelled as ***(partially) Σ-labelled event structures***, *i.e.*, an event structure E along with a *partial* map $lbl : E \rightharpoonup \Sigma$.

The interpretation of types will rely on standard operations on event structures. The ***prefixing*** of a Σ-labelled event structure E by a label ℓ has for event $\{\bot\} \cup E$ where $\bot \notin E$, for causal ordering $\leq_{\ell \cdot E} = \leq_E \cup \{\bot\} \times (\ell \cdot E)$, for conflict $\#_{\ell \cdot E} = \#_E$, and for labelling the extension of lbl_E with $\bot \mapsto \ell$. Prefixing by an internal action, $* \cdot E$, is defined similarly except that labelling is undefined on $\bot$.

The ***parallel composition*** $E_0 \parallel E_1$ of two event structures E_0 and E_1, has for events the disjoint union of E_0 and E_1, coded as $\{0\} \times E_0 \cup \{1\} \times E_1$, and for the ordering and the conflict relation:

$$\leq_{E_0 \parallel E_1} = \{(i, e), (i, e') \mid e \leq_{E_i} e'\} \qquad \#_{E_0 \parallel E_1} = \{(i, e), (i, e') \mid e \#_{E_i} e'\}.$$

Labelling functions on E and F induce naturally a labelling function on $E \parallel F$. The ***nondeterministic sum*** $E_0 + E_1$ is defined as $E_0 \parallel E_1$ except that events of E and F are in conflict:

$$\#_{E_0+E_1} = \{(i,e),(j,e') \mid (i \neq j) \vee (i = j \wedge e \#_{E_i} e')\}$$

We extend this notation to arbitrary sums: $\sum_{i \in I} E_i$.

Given an event structure E and $x \in \mathscr{C}(E)$, we define the ***remainder*** of E after x, written E/x as follows. Its events are those $e \in E \setminus x$ such that $x \cup [e] \in \mathscr{C}(E)$, and the partial order and conflict are inherited from E. Configurations of E/x are in bijection with extensions of x, that is with configurations $y \in \mathscr{C}(E)$ such that $x \subseteq y$.

Confusion-free Event Structures. Event structures support a wide notion of nondeterminism. In our source language, nondeterminism is localised at sums. This restricts the shape of nondeterminism expressible in the language. An event structure E is ***confusion-free*** when *(1)* $e \sim\!\!\sim_E e'$ implies $[e] = [e']$ and *(2)* if $e \sim\!\!\sim_E e' \sim\!\!\sim_E e''$ then $e = e''$ or $e \sim\!\!\sim_E e''$. As a result, the relation $(=_E \cup \sim\!\!\sim_E)$ becomes an equivalence relation whose equivalence classes are called ***cells***. Confusion-freeness is preserved under the operations defined in the previous paragraph.

Event Structures and Recursion. To interpret recursive types, we use the standard denotational technique of endowing event structures with an ω-CPO structure, representing types with open variables as continuous maps, and then interpreting recursive types as least fixpoints of these maps. We recall here the standard ω-CPO structure of event structures [Winskel 1982].

Definition 4.2. A Σ-labelled event structure E is ***included*** in Σ-labelled event structure F (written $E \hookrightarrow F$), when *(1)* $E \subseteq F$; *(2)* for $e, e' \in E$, $e <_E e'$ iff $e <_F e'$; *(3)* E is downclosed in F; *(4)* for $e, e' \in E$, $e \#_E e'$ iff $e \#_F e'$; and *(5)* for $e \in E$, $lbl_E(e) = lbl_F(e)$.

We write $\mathbf{ES}(\Sigma)$ for event structures ordered by inclusion. This order has a minimal element, the empty event structure $\bot$. It is well-known that it is also an ω-CPO:

Lemma 4.3. $\mathbf{ES}(\Sigma)$ *is an ω-CPO, meaning that every infinite ascending chain $E_0 \hookrightarrow E_1 \hookrightarrow \ldots$ has a least upper bound written* $\lim E_i$.

Recall that a monotonic function $f : \mathcal{A} \to \mathcal{B}$ between ω-CPOS is ***continuous*** when it preserves least upper bounds of ω-chains. Continuous maps in ω-CPOs always have a least fixpoint.

Lemma 4.4. *If $F : \mathcal{A} \to \mathcal{A}$ is a continuous map between ω-CPOs, then it has a least fixpoint* $\mathrm{fix}(F)$.

Lemma 4.5. *For any $a \in \Sigma$, the operations $B \mapsto a \cdot B$, $(A,B) \mapsto A \parallel B$ and $(A,B) \mapsto A + B$ are continuous maps. Moreover, if $f : \mathcal{A} \times \mathcal{B} \to \mathcal{B}$ is continuous, then the operation $a \mapsto \mathrm{fix}(b \mapsto f(a,b))$ defines a continuous map $\mathcal{A} \to \mathcal{B}$.*

4.2 Types as Games

We now recall games from [Castellan et al. 2018] and establish a formal link with session types.

Definition 4.6. Given a set Σ, a Σ-***game*** is a $\Sigma \times \{-,+\}$-labelled event structure A such that the labelling function lbl_A is total, and if $e \sim\!\!\sim_A e'$, then $\pi_2(lbl(e)) = \pi_2(lbl(e'))$.

A game A is a set of *moves*, representing the possible messages of a protocol. Each move $e \in A$ comes with a *polarity* $pol_A(e)$ in $\{-,+\}$ indicating if the message must be sent or received. Given a Σ-game A, its ***dual*** $A^\perp$ is obtained by reversing the polarity of moves. Games of [Castellan et al. 2018] are Σ-games for $\Sigma = \{*\}$, but Σ will be useful to make the link with the syntax (messages, names). The last condition ensures that nondeterministic choices belong to one participant.

$$[\![\mathsf{end}]\!](\rho) = \emptyset \qquad [\![X]\!](\rho) = \rho(X) \qquad [\![\oplus_{i\in I} !l_i\langle \widetilde{S_i}\rangle.T_i]\!](\rho) = \sum_{i\in I} !l_i \cdot \left([\![\widetilde{S_i}]\!](\rho)^\perp \parallel [\![T_i]\!](\rho)\right)$$
$$[\![\mu X.\, P]\!](\rho) = \mathrm{fix}(E \mapsto [\![P]\!](\rho[X := E])) \qquad [\![\&_{i\in I} ?l_i(\widetilde{S_i}).T_i]\!](\rho) = \sum_{i\in I} ?l_i \cdot \left([\![\widetilde{S_i}]\!](\rho) \parallel [\![T_i]\!](\rho)\right)$$

Fig. 1. Interpretation of session types into games

The causal order and conflict relation on a game encode the *rules of the game* and specify which event can be played at a particular point of the protocol. In an event structure, two events e and e' can relate in three ways: they can be causally related, in conflict, or concurrent. These three possibilities make sense from a session type perspective:

- *Causality* represents the sequentiality constraint in a session type. For instance in $!\mathtt{tt}.?\mathtt{ff}$, the message $\mathtt{tt}$ must be sent before $\mathtt{ff}$ can be received.
- *Conflict* represents when two messages cannot occur together in an execution. For instance in $!\mathtt{tt} \oplus !\mathtt{ff}$, a process must choose between sending $\mathtt{tt}$ or $\mathtt{ff}$.
- *Concurrency* represents when two parts of the protocol can be performed independently. For instance in $?\mathtt{start}(!\mathtt{tt}).!\mathtt{ff}$, the messages $\mathtt{tt}$ and $\mathtt{ff}$ can be sent in any order or in parallel.

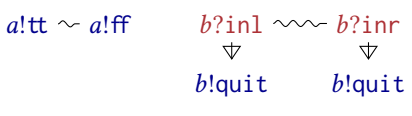
Using this analogy, we devise an interpretation of session types inside games.

Interpreting session types. We define the interpretation of session types into event structures. The events of the resulting event structure will correspond to actions allowed by the type, *i.e.*, will be labelled by extra information about the message sent and the channel on which it is sent. A type will thus be interpreted as a $\mathscr{L}$-game and a context as a $(\mathscr{N} \times \mathscr{L})$-game. To denote labels of moves, we use process-like notations: $!l$ for $(l, +)$, $?l$ for $(l, -)$ in $\mathscr{L} \times \{-, +\}$; and $a!l$ for $(a, l, +)$, $a?l$ for $(a, l, -)$ in $\mathscr{N} \times \mathscr{L} \times \{-, +\}$.

Because of recursive types, we will also have to interpret open types. Given a type T, we write $\mathrm{fv}(T)$ for the free variables occurring in it. Write $\mathbf{Games}(\mathscr{L})$ for the partial order of $\mathscr{L}$-games and inclusions, which is easily seen to be a sub-ω-CPO of $\mathbf{ES}(\mathscr{L} \times \{-, +\})$. An open type T will be interpreted by a continuous map $\mathbf{Games}(\mathscr{L})^{\mathrm{fv}(T)} \to \mathbf{Games}(\mathscr{L})$. Elements of $\mathbf{Games}(\mathscr{L})^{\mathrm{fv}(T)}$ can be seen as environments ρ mapping free variables of T to $\mathscr{L}$-games. The inductive definition of $[\![T]\!]$ is given in Figure 1. We write $[\![\widetilde{T}]\!]$ for $(\rho \mapsto [\![T^1]\!](\rho) \parallel \ldots \parallel [\![T^n]\!](\rho))$. The dual operator in the interpretation of selections is there to ensure that the interpretation commutes with duality:

Lemma 4.7. *If T is a type, $[\![\overline{T}]\!](\rho) = ([\![T]\!](\overline{\rho}))^\perp$ where $\overline{\rho}(X) = \rho(X)^\perp$.*

Given a $\mathscr{L}$-game A, and a name $a \in \mathscr{N}$, we write $a : A$ for the $(\mathscr{N} \times \mathscr{L})$-game obtained from A by relabelling: $lbl_{a:A}(e) = (a, lbl_A(e))$. Contexts are then simply interpreted as parallel composition of their (relabelled) components: $[\![a_1 : T_1, \ldots, a_n : T_n]\!] = a_1 : [\![T_1]\!], \ldots, a_n : [\![T_n]\!]$ assuming that all the types occurring in the context are closed. The game corresponding to the context $a : \mathbb{B}, b : (?\mathtt{inl}(!\mathtt{quit}) \,\&\, ?\mathtt{inr}.!\mathtt{quit})$ is depicted on the left. The labelling need not be injective:



in the picture on the left there are two events with label $b!\mathtt{quit}$. This is not a problem since the corresponding moves of the games are still distinct. We give a strategy on this game in § 5.2.

We now study two properties of this interpretation: (1) which games are denoted by types, and (2) which information is lost when interpreting a type.

4.2.1 Image of the Interpretation. Types and contexts, due to their inductive nature, give rise to event structures which are forests. An event structure E is ***forest-like*** when for all $e \in E$, $[e]$ is

linearly ordered by $\leq_E$. Moreover, the nondeterminism of $[\![T]\!]$ is due to branching/selection, so $[\![T]\!]$ will be confusion-free. This is actually enough to characterise the image without recursion (provided one restricts to finite event structures). However, the interpretation of session types does not reach all the infinite forests in $\mathbf{Games}(\mathscr{L})$ – only the regular ones. An event structure E is ***regular*** [Thiagarajan 2002] when *(1)* the equivalence relation $\{(x, y) \in \mathscr{C}(E) \mid E/x \cong E/y\}$ has a finite number of equivalence classes and *(2)* any configuration can only have a finite number of immediate extensions. When E is a forest, E/x can be seen as the sub-forest rooted at x. Regularity ensures that E contains only a finite number of sub-forests. If Δ is a typing environment, the game $[\![\Delta]\!]$ can be regarded as a $\mathscr{L}$-game by discarding the information on names.

Lemma 4.8. *A $\mathscr{L}$-game is isomorphic (as $\mathscr{L}$-games) to a game of the form $[\![\Delta]\!]$ if and only if it is is forest-like, regular and confusion-free.*

4.2.2 An Equational Theory on Types. Finally, we look at the equational theory induced by the model. As illustrated in § 2.1.2, the interpretation equates some types, *e.g.* $?(S).T$ and $?(T).S$ whose interpretation is $?() \cdot ([\![S]\!] \parallel [\![T]\!])$. We capture these equalities by the equivalence relation on session types, generated by the following rules, plus closure under context:

$$\&_{i\in I}?l_i(\widetilde{S_i}).T_i \equiv \&_{i\in I}?l_i(\sigma(\widetilde{S_i}, T_i, \mathrm{end})).\mathrm{end} \qquad \oplus_{i\in I}\, !l_i\langle\widetilde{S_i}\rangle.T_i \equiv \oplus_{i\in I}!l_i\langle\sigma(\widetilde{S_i}, \overline{T}_i, \mathrm{end})\rangle.\mathrm{end}$$

where σ denotes any permutation of the sequence of types. By applying repeatedly these rules, a number of end can be inserted or removed from the argument types. The equivalence $\equiv$ states that the order of argument types does not matter, and that argument types and continuation types can be permuted. Remark that to move the continuation to the argument type in the case of an output, it is necessary to take its dual. (This is related to the interpretation of types in games).

Proposition 4.9. *For session types S, T without recursion, $S \equiv T$ if and only if $[\![S]\!] \cong [\![T]\!]$.*

5 COINCIDENT STRATEGIES

We now introduce our notion of strategies, the semantic counterpart of processes. As discussed in § 3.5, strategies in the existing models of concurrency in game semantics only feature an asynchronous copycat which makes difficult to give meaning to free name passing. In this section, we generalise event structures to allow several moves to be played *at the same time*, creating ***coincidences***. In § 5.1, we introduce coincident event structures, obtained from event structures by moving from a causal partial order to a causal *preorder*. In § 5.2, we generalise the strategies of [Castellan et al. 2018] using these coincident event structures. Then, in § 5.3, we study the interaction of coincident strategies, crucial to define the composition of strategies and interpret the restriction operator. This composition is introduced in § 5.4, where we show that coincident strategies naturally organise themselves in a (compact-closed) category. We conclude this section in § 5.5, by showing weak bisimulation and observational equivalence coincide on strategies applying the technique in [Hennessy 2007]. This result will be used to prove that our model is intensionally fully abstract in § 6.2.

5.1 Coincident Event Structures

Traditional prime event structures cannot express the fact that two events must occur at the same time: concurrent (*i.e.*, causally incomparable) events may occur at the same time, but may also occur in a particular order. This is due to event structures being ***coincidence-free***: for each pair of event e, e' there exists a configuration x which separates them, *i.e.* contains one but not the other. Each configuration x can thus be reached (possibly in many ways) by a series of atomic steps:

$$\emptyset \xrightarrow{e_1} \{e_1\} \dots \xrightarrow{e_n} \{e_1, \dots, e_n\} = x$$

In prime event structures, this coincidence-freedom follows directly from the anti-symmetry of the causal relation $\leq_E$. By removing this axiom, *coincident events* become possible, as cycles for the causal order.

Definition 5.1. A Σ-labelled ***coincident event structure*** is a triple $(E, \leq_E, \#_E, lbl_E : E \rightharpoonup \Sigma)$ where $(E, \leq_E)$ is a preorder, $\#_E \subseteq E \times E$ is a symmetric irreflexive binary relation on E such that *(1)* $[e]$ is finite for $e \in E$ and *(2)* if $e \#_E e'$ and $e \leq e_0$ then $e_0 \#_E e'$.

We use similar notations as for event structures: $\mathscr{C}(E)$ denotes the set of finite, downclosed and conflict-free subsets of E, called ***configurations***. Minimal conflict is defined as before. We define $e \rightarrowtriangle e'$ in this new setting as $e < e'$, $\neg(e' < e)$ and there are no events in between e and e'. An event $e \in E$ is said to be ***visible*** when $lbl_E(e)$ is defined; ***invisible*** or ***internal*** otherwise. We write E_* for the set of internal events of E and E_υ for the set of visible events of E, so that the set E splits into $E_* \cup E_\upsilon$. This decomposition applies to configurations: if $x \in \mathscr{C}(E)$ we write x_* (resp. x_υ) for the set of internal (resp. visible) events of x.

We write $\equiv$ for the equivalence relation induced by the preorder $\leq$ (*i.e.* $e \equiv e'$ when $e \leq e'$ and $e' \leq e$). The equivalence classes for $\equiv$ are called ***coincidences***. Two coincident $e \equiv e'$ must occur *at the same time*, as they cannot be separated by a configuration: $e \in x$ iff $e' \in x$. We write $E_{/\equiv}$ for the set of coincidences of E. We tend to use $X, Y, \ldots$ for coincidences. A coincidence is ***trivial*** when it is a singleton. Given any subset $X \subseteq E$, we write $[X]$ for the downclosure of X in E and $[X) = [X] \setminus X$ as for event structures. We say that $x \overset{X}{\subset\!\!-} y$ when $y = x \cup X$, the union is disjoint, and if $x \subseteq z \subseteq y$, then $x = z$ or $z = y$ (with $x, y, z \in \mathscr{C}(E)$). In that case, X must be a coincidence.

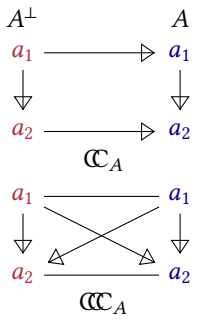
To illustrate coincidences, we give the definition of the ***(asynchronous) copycat*** $\mathfrak{C}_A$ used in traditional concurrent game semantics, and the ***coincident copycat***, a new feature of our model.

Definition 5.2. Given a game A, we define the coincident event structures $\mathfrak{C}_A$ (copycat) and $\mathfrak{CC}_A$ (coincident copycat) as follows. Their event set is $A^\perp \parallel A$, and their causal preorder:

$$\leq_{\mathfrak{C}_A} = (\leq_{A^\perp \parallel A} \cup \{((i,a),(1-i,a)) \mid i \in \{0,1\} \wedge pol_{A^\perp \parallel A}(i,a) = -\})^*$$
$$\leq_{\mathfrak{CC}_A} = (\leq_{A^\perp \parallel A} \cup \{((i,a),(1-i,a)) \mid i \in \{0,1\}\})^*$$

Conflict is then inherited from the game: two events e, e' are in conflict in $\mathfrak{C}_A$ (resp. $\mathfrak{CC}_A$) when $[\{e,e'\}]_{\mathfrak{C}_A}$ (resp. $[\{e,e'\}]_{\mathfrak{CC}_A}$) is not a configuration of $A^\perp \parallel A$. The identity function turns $\mathfrak{CC}_A$ and $\mathfrak{C}_A$ into $(A^\perp \parallel A)$-labelled event structures.

For the game $A = a_1 \rightarrowtriangle a_2$ (two positive moves in sequence), $\mathfrak{C}_A$ and $\mathfrak{CC}_A$ are depicted on the right. Coincidences are drawn with a straight line, rather than as causal loops. Parallel composition and sums extend to coincident event structures. Event structures come with a notion of map $f : E \rightharpoonup E'$ used to represent a (partial) simulation of E within E'. Maps allow to characterise some operations on event structures with universal properties, *e.g.*, the synchronised product of event structures as a categorical product.



We extend the traditional maps of event structures to coincident event structures as follows. For $f : E \rightharpoonup E'$ to be a valid simulation, when $e \equiv e'$ in E and both $f\,e$ and $f\,e'$ are defined then $f\,e$ and $f\,e'$ must be coincident or concurrent. This is justified by the fact that strategies will be maps of coincident event structures, and the interaction of coincident strategies satisfies a universal property relative to this class of maps (Theorem 5.8).

Definition 5.3. A ***map of coincident event structures*** $f : E \rightharpoonup E'$ is a partial function from E to E' satisfying: *(1)* if $x \in \mathscr{C}(E)$ then $f\,x \in \mathscr{C}(E')$; *(2)* if $e, e' \in x \in \mathscr{C}(E)$ and $f\,e = f\,e'$ then $e = e'$;

(3) if $e \equiv_E e'$, and $f\ e$ and $f\ e'$ are both defined, then they are either concurrent or coincident; and *(4)* for $e \in E$, $lbl_E(e)$ and $lbl_{E'}(f\ e)$ are both undefined, or both defined and equal.

The first two axioms are the same as for usual event structure maps. As a result, a map of coincident event structures $E \rightharpoonup E'$ when E and E' are event structures is simply a map of event structures as usually defined. A map $f : E \rightharpoonup E'$ is ***total*** when the underlying function is; an ***isomorphism*** when it has an inverse $g : E \to E'$: in this case we write $E \cong E'$.

The operation E/x can be easily generalised to coincident event structures, and is compatible with the notion of maps:

Lemma 5.4. *Let $f : E \rightharpoonup E'$ be a map of coincident event structures. For $x \in \mathscr{C}(E)$, the function $E/x \rightharpoonup E'/f\ x$ obtained by restricting f is well-defined and a map of coincident event structures.*

5.2 Coincident Strategies

We introduce coincident strategies on games, generalising the strategies of [Castellan et al. 2018]. Consider a game A.

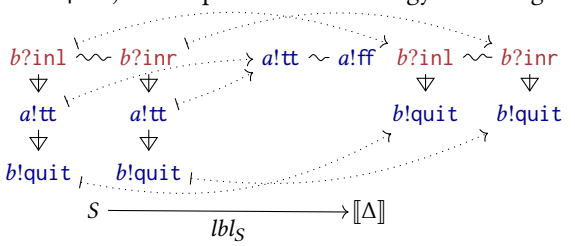
Definition 5.5. An A-labelled coincident event structure S is a ***coincident strategy*** on A when:

(1) Labelling is a map of coincident event structures $S \rightharpoonup A$.
(2) If $s \frown\!\!\!\sim_S s'$, then s and s' are either both internal, or both visible. Moreover, if they are visible, s is coincident to negative s_0 and s' to negative s'_0 such that $lbl(s_0) \frown\!\!\!\sim_A lbl(s'_0)$.
(3) If $X \in S_{\equiv}$ is non-trivial, then $X = \{s, s'\}$ where $lbl(s)$ and $lbl(s')$ are both defined and of distinct polarity in A.

The first condition ensures that strategies respect the rules of A and play every move at most once in a configuration. The second condition, **secrecy**, generalises the secrecy condition of [Castellan et al. 2018], forcing conflict between visible events to arise from a negative conflict in the game. The third condition severely restricts the shape of coincidences in S: the only non-trivial coincidences allowed are those with two visible events of opposite polarities, as in coincident copycat. Note that we do not impose any receptivity or courtesy conditions as in standard concurrent game semantics. This implies that the asynchronous copycat will not be an identity, but the coincident copycat will.

The assertion that S is a coincident strategy on A will often be written $S : A$. We write $\mathbf{CG}_S(A)$ for the set of coincident strategies on a game A. An A-labelled coincident event structure where the labelling function $lbl_S : S \rightharpoonup A$ is a map is called a ***coincident pre-strategy***. Isomorphism of labelled event structures gives a notion of isomorphism of strategies. Two strategies are isomorphic when they only differ by the "codes" (in an informal sense) of the events, but not by the structure, for instance $S = \{\alpha\}$ with $lbl_S(\alpha) = a$ and $S' = \{\beta\}$ with $lbl_{S'}(\beta) = a$ must be identified even if $\alpha \neq \beta$. In particular, the categorical laws (cf. § 5.4) will only hold up to isomorphism.

Remember the context in the example of § 4, $\Delta = a : \mathbb{B}, b : (?\mathtt{inl}(!\mathtt{quit}) \,\&\, ?\mathtt{inr}.\,!\mathtt{quit})$. The process $P = (b?\mathtt{inl}(x).\, a!\mathtt{tt}.\, x!\mathtt{quit}) \,\&\, (b?\mathtt{inr}.\, a!\mathtt{tt}.\, b!\mathtt{quit})$ is interpreted as the strategy on the right. In most cases, the information about the labelling to the game can be recovered so we will omit it, only drawing the left picture, as explained in § 2.1. Note that in this case, the labelling is not injective since $a!\mathtt{tt}$ is played twice, but the two events occur in incompatible branches.



Weak bisimulation. As coincident strategies are event structures, we can equip them with a labelled transition system using the remainder operation defined above. Note that if $S : A$ and

$x \in \mathscr{C}(S)$, it is easy to see that the map $S/x \rightharpoonup A/lbl_S(x)$ induced by Lemma 5.4 turns S/x into a coincident strategy on $A/lbl_S(x)$. Given a coincident pre-strategy S on A, we say that:

- $S \xrightarrow{\tau} S'$ when there exists an internal minimal event $s \in S$ such that $S/[s] \cong S'$
- $S \xrightarrow{Y} S'$ when there exists a visible coincidence X containing minimal events such that $lbl_S(X) = Y$ and $S/X \cong S'$ (as strategies on A/Y) (Y must be a set of minimal events of A).

Similarly, we can define weak transitions, in order to define weak bisimulation:

$$\overset{\tau}{\Rightarrow} := (\xrightarrow{\tau})^* \qquad \overset{Y}{\Rightarrow} := (\xrightarrow{\tau})^* \xrightarrow{Y}$$

Definition 5.6. A weak bisimulation is family of *equivalence* relations $(\mathcal{R}_A)_{A \in \mathbf{Games}}$ where $\mathcal{R}_A$ relates coincident pre-strategies on A, such that

- If $S \mathcal{R}_A T$, and $S \xrightarrow{\tau} S'$, then there exists a coincident strategy $T' : A$ with $T \overset{\tau}{\Rightarrow} T'$ and $S' \mathcal{R}_A T'$;
- If $S \mathcal{R}_A T$ and $S \xrightarrow{Y} S'$, then there exists $T \overset{Y}{\Rightarrow} T'$ and $S' \mathcal{R}_{A/Y} T'$.

The largest weak bisimulation is called ***weak bisimilarity*** and written $\approx$. Since isomorphism is clearly a weak bisimulation, $S \cong T$ implies $S \approx T$. As in process calculi, weak bisimulation offers a more concrete characterisation of observational equivalence (§ 5.5).

5.3 Interaction of Coincident Strategies

We have most of the ingredients to interpret the session π-calculus, except the restriction operator. Syntactically, the composition of $\vdash P \triangleright \Delta, \Delta_1$ with $\vdash Q \triangleright \overline{\Delta}, \Delta_2$ on the interface Δ is the process

$$\vdash Q \circledcirc_\Delta P := (\nu \widetilde{a})(P \mid Q) \triangleright \Delta_1, \Delta_2 \qquad \text{where } \Delta = a_1 : T_1, \ldots, a_n : T_n.$$

In game semantics, composition is primitive, and implemented in two steps. First P and Q ***interact*** over Δ, and then the actions on Δ in the resulting process are hidden. In this section, we start by defining the interaction of $S : A \parallel B$ and $S' : A^\perp \parallel C$, resulting in a $A \parallel B \parallel C$-labelled event structure $S *_A S'$. It is not a strategy, because of the disagreement over the polarity of moves of A.

We refer the reader to [Castellan et al. 2018, 2017] for detailed intuition about the behaviour of interaction. For lack of space, we give here an axiomatic definition, phrased as a universal property. The explicit definition can be found in Appendix B.1. Intuitively, the interaction $S * S'$ should only exhibit behaviour that S and S' have. To formulate a common behaviour between S and S', *i.e.* a behaviour that can be both simulated by S and S', we use maps of coincident event structures.

Definition 5.7. A ***common behaviour*** of S and S' is a triple (T, α, β) of a $(A \parallel B \parallel C)$-labelled coincident event structure T and two maps $\alpha : T \rightharpoonup S$ and $\alpha : T \rightharpoonup S'$ such that:

(1) If α and β are both defined on $t \in T$, then $lbl(t)$ is defined.
(2) There are no coincident $s_1, \ldots, s_n \in T$ with $\alpha\, s_1 \equiv \alpha s_2$, $\beta s_2 \equiv \beta s_3$, $\alpha s_3 \equiv \alpha s_4$, …, $\beta s_n \equiv \beta s_1$.

A common behaviour is a coincident event structure that both S and S' can simulate. We also add two conditions: the first one is taken from [Castellan et al. 2018], and amounts to saying that two processes cannot synchronise on neutral events: the neutral events of S are invisible to S' and vice versa. The second one prevents cycles between coincidences, which is necessary to prove functoriality of the encoding of session π-calculus in § 7.

The interaction should not just be any common behaviour, but the largest: it should be able to simulate any other common behaviour. A ***simulation*** from (T, α, β) to (T', α', β') is a partial map $\varphi : T \rightharpoonup T'$ with $\alpha' \circ \varphi = \alpha$ and $\beta' \circ \varphi = \beta$.

Theorem 5.8. *There exists a common behaviour of S and S' $(S *_A S', \Pi_S, \Pi_{S'})$ which is the largest common behaviour,* i.e. *for every other common behaviour (T, α, β) there exists a unique simulation $(T, \alpha, \beta) \to (S *_A S', \Pi_S, \Pi_{S'})$.*

5.4 The Category of Coincident Strategies

Using the interaction defined in the previous section, we now define the composition of strategies and show it defines a compact-closed category of strategies. A coincident strategy from a game A to a game B will be, as usual in game semantics, a strategy on $A^\perp \parallel B$. As a result, $\mathfrak{CC}_A$ is a strategy from A to A and will be the identity of our category. Composition will be obtained in a standard manner by interaction plus hiding.

Given a coincident strategy S on $A^\perp \parallel B$ and R on $B^\perp \parallel C$, we wish to build $R \circledcirc S$ to be a coincident strategy on $A^\perp \parallel C$. Via the previous subsection, we can form their interaction $S *_B R$, written $R \circledast_B S$, which is a $(A^\perp \parallel B \parallel C)$-labelled event structure. To obtain a strategy on $A^\perp \parallel C$, we need to hide the behaviour on B. A solution would be to simply relabel $R *_B S$ to consider events on B as being internal. On the one hand, most events on B do not represent anything meaningful about the *external* behaviour of S and T interacting together. On the other hand, hiding everything forgets nondeterministic branches without observable behaviour. In [Castellan et al. 2018], a compromise is found by keeping only the essential events:

Definition 5.9. An event of a Σ-coincident event structure S is ***essential*** when it is visible; or internal, and *(1)* there exists $s' \in S$ such that $s \frown\!\frown s'$ and *(2)* s is not coincident to a visible event.

The definition of essential events had to be updated to take into account coincidences. In particular, internal events coincident to visible ones are never useful since the visible events remember the conflict. This is essential to ensure that the composition will indeed be a coincident strategy, as $R \circledast_B S$ is only a pre-strategy (it may contain arbitrary large coincidences involving events synchronised on B). For S a Σ-labelled coincident event structure, we write $\mathscr{E}(S)$ for the coincident event structure consisting in the essential events of S, and causality and conflict inherited from S. As in [Castellan et al. 2018], hiding inessential events is sound up to weak bisimulation:

Lemma 5.10. *For any coincident pre-strategy S on A, we have $S \approx \mathscr{E}(S)$.*

Let us define $T \circledcirc S$ as $\mathscr{E}(T \circledast_B S)$ along with the labelling function $\mathscr{E}(T \circledast_B S) \rightharpoonup A \parallel B \parallel C \rightharpoonup A \parallel C$ (note that, as far as labelling functions are concerned, polarity on the games does not matter). We obtain the desired coincident strategy:

Lemma 5.11. *$T \circledcirc S$ is a coincident strategy on $A^\perp \parallel C$.*

Unlike in traditional asynchronous game semantics, the coincident copycat is an identity without any further asynchrony assumption on strategies.

Proposition 5.12. *For a coincident strategy S on A, $\mathfrak{CC}_A \circledcirc S \cong \mathscr{E}(S)$.*

Define a coincident strategy S to be ***essential*** when $\mathscr{E}(S) = S$ (*i.e.*, all its events are essential). From Proposition 5.12, it is easy to construct a compact-closed category:

Theorem 5.13. *We obtain a compact-closed category $\mathbf{CG}_S$ whose objects are games, and morphisms are essential coincident strategies up to isomorphism.*

To close the section, we remark that the strategies of [Castellan et al. 2018] arise as a special case of coincident strategies, which are asynchronous. A ***coincidence-free strategy*** is a coincident strategy S without non-trivial coincidences (*i.e.* S is an event structure).

Definition 5.14. A coincidence-free strategy S on A is ***asynchronous*** when:

Courtesy if $s \rightarrowtriangle s'$ but $lbl(s)$ and $lbl(s')$ are incomparable, then s is negative and s' positive.

Receptivity If $x \in \mathscr{C}(S)$ is such that $lbl(x) \xrightarrow{a} \subset$ in A, with a negative, then there exists a unique $s \in S$ such that $lbl(s) = a$ and $x \xrightarrow{s} \subset$.

Theorem 5.15 ([Castellan et al. 2018]). *Games and essential asynchronous strategies up to isomorphism form a compact-closed category* $\mathbf{CG}_A$.

The composition of asynchronous strategies defined in [Castellan et al. 2018] arises as a special case of the composition defined here. However, the coincident copycat is obviously not asynchronous, thus $\mathbf{CG}_A$ is not a subcategory of $\mathbf{CG}_S$.

Prefixing and neutral events. We have seen that if we have $S : A$, then we can prefix both S and A by a common event e (outside a) to obtain a game $e \cdot A$ and a strategy $e \cdot A$. In particular, we have the lifting strategy $L_e := (1, e) \cdot \mathfrak{CC}_A$ playing on $A^\perp \parallel (e \cdot A)$. In the next section, we need to prefix a strategy by an event, by composing with L_e. However, $L_e \circledcirc S$ is not isomorphic to $e \cdot S$ in general. This is only true if S has no minimal internal events. Indeed such minimal events will induce minimal internal events in $L_e \circledcirc S$. For this reason, we introduce the construction $e \rightarrowtriangle S$ as follows. The set of events and conflict is the same as $e \cdot S$ but the causality is given by the transitive closure of $\leq_S \cup \{e\} \times S_v$ (only visible events of S may depend immediately on e). We have:

Lemma 5.16. *For $S : A$, we have $L_e \circledcirc S \cong e \rightarrowtriangle S$.*

Recursion. As before, we can define inclusions between coincident event structures. A coincident event structure E is included in a coincident event structure F when $E \subseteq F$, E is downclosed in F and causality, and conflict from E and F coincide on events of E.

Lemma 5.17. $\mathbf{CG}_S(A)$ *along with inclusion forms an ω-CPO.*

As before, the main operations used to build strategies are continuous, in particular composition (remember that $\mathbf{CG}_S(A, B) = \mathbf{CG}_S(A^\perp \parallel B)$):

Lemma 5.18. *For a coincident strategy $S : A^\perp \parallel B$, the operation $R : B^\perp \parallel C \mapsto R \circledcirc_B S$ is a continuous map $\mathbf{CG}_S(B, C) \to \mathbf{CG}_S(A, C)$.*

5.5 Behavioural Equivalence

Before interpreting the session π-calculus, we study the behavioural theory of our model. We define on strategies a semantic counterpart to the reduction-closed barbed congruence [Honda and Yoshida 1995; Milner and Sangiorgi 1992], and show it coincides with weak bisimulation. Since behavioural equivalences depend on the possible set of contexts, we will consider a parameter $\mathscr{M}$ called a ***submodel*** which is a subset $\mathscr{M}$ of games and for every $A, B \in \mathscr{M}$ a subset $\mathscr{M}(A, B) \subseteq \mathbf{CG}_S(A, B)$ such that:

(1) $\emptyset \in \mathscr{M}$ and $\mathscr{M}$ is stable under duality, parallel composition, prefix and remainder.
(2) If $S \in \mathscr{M}(A)$ and $x \in \mathscr{C}(S)$ contains only internal events, then S/x is in $\mathscr{M}(A)$.
(3) $\mathscr{M}$ is closed under composition.
(4) For every $S \in \mathscr{M}(A)$ and label e, $e \rightarrowtriangle S \in \mathscr{M}(e \cdot A)$.

where $\mathscr{M}(A)$ stands for $\mathscr{M}(\emptyset, A)$. We define the ***barbs*** of a coincident strategy $S : A$ to be the set $barb(S)$ of positive $e \in min(A)$ such that $S \overset{e}{\Rightarrow}$.

Definition 5.19. A family of equivalence relations $(\mathcal{R}_A \subseteq \mathscr{M}(A) \times \mathscr{M}(A))_{A \in \mathscr{M}}$ is a reduction-closed $\mathscr{M}$-congruence, when:

- If $S \mathcal{R}_A S'$ then $barb(S) = barb(S')$.
- If $S \mathcal{R}_A R$, and $S \xrightarrow{\tau} S'$, then there exists $R \overset{\tau}{\Rightarrow} R'$ with $S' \mathcal{R}_A R'$.
- If $S \mathcal{R}_A S'$ then for all game $B \in \mathscr{M}$ and strategy $U : A^\perp \parallel B \in \mathscr{M}(A, B)$, $(U \circledcirc S) \mathcal{R}_B (U \circledcirc S')$.
- If $S \in \mathscr{M}(A)$ and $S' \in \mathscr{M}(A)$ are such that $(e \rightarrowtriangle S) \mathcal{R}_{e \cdot A} (e \rightarrowtriangle S')$, then $S \mathcal{R}_A S'$.

This definition is very similar to the definition in § 3.4, except for the last condition, which comes for free if $\mathscr{M}$ contains sufficiently many strategies. We include it so that Lemma 5.24 holds in full generality. Given a submodel $\mathscr{M}$, we write $\simeq_{\mathscr{M}}$ for the largest $\mathscr{M}$-congruence, and in particular $\simeq$ will be a shorthand for $\simeq_{\mathbf{CG}_S}$. The larger the submodel is, the more fine-grained the equivalence is, *i.e.* if $\mathscr{M} \subseteq \mathscr{M}'$, then $S \simeq_{\mathscr{M}'} S'$ implies $S \simeq_{\mathscr{M}} S'$ when S, S' belong to $\mathscr{M}$. To relate weak bisimulation and barbed congruence, we start by showing that weak bisimulation is a congruence.

Lemma 5.20. *If $S \approx S' : A^\perp \parallel B$, then for all $R : B^\perp \parallel C$, then $R \circledcirc S \approx R \circledcirc S'$.*

We deduce that weak bisimulation is a reduction-closed congruence for any class of observers:

Lemma 5.21. *For any submodel $\mathscr{M}$ of $\mathbf{CG}_S$ and $S, R \in \mathscr{M}(A)$, if $S \approx T$ then $S \simeq_{\mathscr{M}} S'$.*

5.5.1 Towards Full Abstraction. To show the converse, we apply a method based on the action tester from [Hennessy 2007]. Actions on a game A are either minimal events of A or pairs $\{a^-, b^+\}$ of minimal events of A. We will write $\alpha, \beta, \ldots$ for actions. Given a game A, and an action α, we will write $\mathsf{test}(A, \alpha)$ for the game $(\mathsf{succ}^+ \cdot A/\alpha) \parallel \mathtt{fail}^+$.

Definition 5.22. A submodel $\mathscr{M}$ ***defines an action*** α of $A \in \mathscr{M}$ if there exists a strategy $U \in \mathscr{M}(A, \mathsf{test}(A, \alpha))$ such that:

- For all $S, S' \in \mathscr{M}(A)$ such that $S \xrightarrow{\alpha} S'$, $\mathit{barb}(U \circledcirc S) = \{\mathsf{succ}^+\}$ and $U \circledcirc S \overset{\tau}{\Rightarrow} (\mathsf{succ}^+ \twoheadrightarrow S')$.
- For $S \in \mathscr{M}(A)$, if $\mathit{barb}(U \circledcirc S) \overset{\tau}{\Rightarrow} S_0$ with $\mathit{barb}(U \circledcirc S) = \mathit{barb}(S_0) = \{\mathsf{succ}^+\}$ then $S_0 \overset{\tau}{\Rightarrow} (\mathsf{succ}^+ \twoheadrightarrow S')$ with $S \overset{\alpha}{\Rightarrow} S'$.

Definition 5.23. A submodel $\mathscr{M}$ is ***testing-closed*** if it defines all actions of games in $\mathscr{M}$.

Lemma 5.24. *If $\mathscr{M}$ is a testing-closed submodel, then for $S, R \in \mathscr{M}(A)$, we have: $S \approx R$ iff $S \simeq_{\mathscr{M}} T$.*

5.5.2 Definability of Actions. As a first application of Lemma 5.24, we show that weak bisimulation and barbed congruence coincide in $\mathbf{CG}_S$ by showing that $\mathbf{CG}_S$ is testing-closed.

Consider a game A and an action α of A. We construct a coincident strategy on the game $A^\perp \parallel \mathsf{test}(A, \alpha)$, as follows:

- If α is a singleton e, we define $U_e = (0, e) \cdot (1, \mathsf{succ}) \cdot \mathbb{CC}_{A/e} : A^\perp \parallel \mathsf{succ}^+ \cdot (A/\alpha) \subseteq A^\perp \parallel \mathsf{test}(A, \alpha)$
- If α is a set $\{a, b\}$, we define

$$\begin{aligned} U_{\{a,b\}} = {} & (* \cdot ((0, a) \parallel (0, b)) \cdot (1, \mathsf{succ}^+) \cdot \mathbb{CC}_{A/\alpha}) \\ & + (* \cdot (0, a) \cdot (0, b) \cdot (1, \mathtt{fail}^+)) \\ & + (* \cdot (0, b) \cdot (0, a) \cdot (1, \mathtt{fail}^+)) \end{aligned}$$

The usage of `fail` is crucial for testing coincidences, since playing a and b in parallel does not rule out strategies starting with a and b in parallel, rather than coincidentally. Hence we need the other two branches that test a then b and b then a. If the strategy plays a and b concurrently rather than coincidentally, then in the composition this matches with all branches, hence triggering the fail.

Theorem 5.25 (Semantic full abstraction – $\mathbf{CG}_S$). *The coincident strategy U_α defines the action α for $\mathbf{CG}_S$. As a result, $\mathbf{CG}_S$ is testing-closed, and $S \simeq R$ iff $S \approx R$ for $S, R \in \mathbf{CG}_S(A)$.*

This result is fundamental to prove that our interpretation is fully-abstract. It also shows that this semantic space is well-behaved from a concurrency theory point of view. Interestingly, we can also apply this lemma to show that $\mathbf{CG}_A$ is fully abstract:

Theorem 5.26 (Semantic full abstraction – $\mathbf{CG}_A$). *The submodel $\mathbf{CG}_A$ is testing-closed, hence for any asynchronous strategies $S, R : A$, we have:*

$$S \simeq_{\mathbf{CG}_S} R \quad \textit{iff} \quad S \approx R \quad \textit{iff} \quad S \simeq_{\mathbf{CG}_A} R.$$

$$[\![\textstyle\sum_{i\in I} P_i]\!]\,(\gamma) = \sum_{i\in I} * \cdot [\![P_i]\!](\gamma) \quad [\![P \mid P']\!]\,(\gamma) = [\![P]\!](\gamma) \parallel [\![P']\!](\gamma) \quad [\![X]\!]\,(\gamma) = \gamma(X)$$
$$[\![\&_{i\in I}\, u?l_i(\widetilde{x_i}).P_i]\!]\,(\gamma) = \textstyle\sum_{i\in I} u?l_i \cdot [\![P_i]\!](\gamma)$$
$$[\![u!l_k\langle\widetilde{v}\rangle.P]\!]\,(\gamma) = u!l_k \cdot ([\![P]\!](\gamma) \parallel \mathfrak{CC}_{[\![\widetilde{T}]\!]}) \text{ where } \widetilde{v} : \widetilde{T}$$
$$[\![\mu X.P]\!]\,(\gamma) = \mathrm{fix}(S \mapsto [\![P]\!](\gamma[X := S])) \quad [\![(\nu a{:}T)P]\!]\,(\gamma) = [\![P]\!](\gamma) \circledcirc_{[\![\overline{a}:\overline{T},a:T]\!]} \mathfrak{CC}_{[\![T]\!]} \quad [\![\mathbf{0}]\!]\,(\gamma) = \emptyset$$

Fig. 2. Interpretation of session π-calculus as coincident strategies

The proof is by taking the same testing strategies and replacing occurrences of the coincident copycat with the asynchronous copycat. This result shows that coincident strategies do not add any observational power to asynchronous strategies. However, the submodel of coincidence-free strategies is not testing-closed because there is no identity. This is related to Example E.1.

6 SESSION PROCESSES AND COINCIDENT STRATEGIES

In this section, we give an interpretation of session processes in terms of coincident strategies: given a closed process $\vdash P \triangleright \Delta$, we construct a strategy $[\![P]\!]$ on the game $[\![\Delta]\!]$ (§ 6.1). We then show that this interpretation is fully abstract (§ 6.2), *i.e.* the interpretation preserves and reflects barbed congruence. Our proof of full abstraction does not use the standard finite definability argument as it is not enough to define finite tests in this nondeterministic setting, but instead relies on the development of § 5.5, showing the action testers in § 5.5.2 are definable in the syntax. Nevertheless, finite definability is an interesting result on its own and we conclude this section by showing a result of finite definability for the internal session π-calculus (§ 6.3), hence showing that internal processes form a programming language for coincidence-free strategies.

6.1 Interpreting Session π-Calculus

We now have all the ingredients to interpret the session π-calculus. Recursive processes are interpreted similarly to recursive types, as fixpoints of continuous maps. We assume all types in recursion contexts are closed. Given a context $\Gamma = (X_1{:}\Delta_1), \ldots, (X_n{:}\Delta_n)$ we map it to the category of embeddings $[\![\Gamma]\!] = \mathbf{CG}_S([\![\Delta_1]\!]) \times \ldots \times \mathbf{CG}_S([\![\Delta_n]\!])$. Then a judgement $\Gamma \vdash P \triangleright \Delta$ is interpreted as a continuous map $[\![\Gamma]\!] \to \mathbf{CG}_S(\Delta)$. The interpretation is defined on Figure 2. For branching processes, the continuation is prefixed with an event encoding the message received. When interpreting the selection, the synchronous forwarder links the subgame for u, and the subgames for $\widetilde{v}$. For a nondeterministic sum, we prefix the strategies with internal events in order to remember the branching point. A parallel composition is mapped to a parallel composition (with no interaction) since a reduction only occurs under a name restriction. The interpretation of restriction is obtained by composed by copycat: here, $\mathfrak{CC}_{[\![T]\!]}$ is a map from the empty game to $([\![T]\!]^\perp \parallel [\![T]\!])$, composed with the interpretation of P viewed as a coincident strategy from $([\![T]\!]^\perp \parallel [\![T]\!])$ to $[\![\Delta]\!]$.

Lemma 6.1. *Consider $\Gamma \vdash P, Q \triangleright \Delta$. If $P \equiv Q$ then $[\![P]\!](\gamma) \cong [\![Q]\!](\gamma)$ for $\gamma \in [\![\Gamma]\!]$.*

We notice that composition of processes, and asynchronous forwarders are definable.

Lemma 6.2. *Consider a context $\Delta = a_1 : A_1, \ldots, a_n : A_n$, and $\vdash P \triangleright \Delta, \Delta_P$ and $\vdash Q \triangleright \overline{\Delta}, \Delta_Q$. We have*

$$[\![Q]\!] \circledcirc_{[\![\Delta]\!]} [\![P]\!] \cong [\![Q \circledcirc_\Delta P]\!] \; (= [\![(\nu \widetilde{a})(P \mid Q)]\!]).$$

Moreover, for a type T without recursion, we have $[\![[u = v]_T]\!] = \mathfrak{C}_{[\![T]\!]}$.

6.2 Full Abstraction

In this subsection, we prove that our interpretation is fully abstract. First we state a correspondence between the operational semantics and the model:

Lemma 6.3 (Soundness and adequacy). *Let $\vdash P \triangleright \Delta$ be a well-typed process. Then we have: (1) if $P \to^* Q$, then $[\![P]\!] \stackrel{\tau}{\Rightarrow} [\![Q]\!]$; and (2) if $[\![P]\!] \stackrel{\tau}{\Rightarrow} S$ then $S = [\![Q]\!]$ with $P \to^* Q$.*

This result induces a correspondence at the level of barbs. However, in the syntax, recall that a process P that contains both a name and its coname does not have visible barbs on these names, yet in the current semantics, actions on these names are visible. To solve this mismatch, from $[\![P]\!]$, we define another interpretation $(\!|P|\!)$ which hides actions on names that should not be observable. Given $\vdash P \triangleright \Delta$, consider the decomposition of Δ of the form $\Delta_0, \overline{\Delta_0}, \Delta_1$ where Δ_1 does not contain a name and its coname. We write $(\!|P|\!) = [\![P]\!] \circledcirc_{[\![\Delta_0, \overline{\Delta_0}]\!]} \mathbb{C}_{[\![\Delta_0]\!]}$, which is a strategy on $[\![\Delta_1]\!]$.

Lemma 6.4. *Let $\vdash P \triangleright \Delta$. Then $P \Downarrow_a$ if and only if $(\!|P|\!) \stackrel{a!l}{\Longrightarrow}$ for some $l \in \mathscr{L}$.*

To derive full abstraction of the model, we reuse Lemma 5.24. Define $\mathscr{C}_\pi$, the submodel comprising of the games of the form $[\![\Delta]\!]$, and given two games $[\![\Delta]\!]$, $[\![\Delta']\!]$, $\mathscr{C}_\pi([\![\Delta]\!], [\![\Delta']\!])$ comprises all $(\!|P|\!)$ when $\vdash P \triangleright \overline{\Delta}, \Delta'$. It is a routine verification to show that $\mathscr{C}_\pi$ is indeed a submodel. By finding process equivalents of the strategies used in the previous section, we obtain:

Theorem 6.5. *$\mathscr{C}_\pi$ is testing-closed. As a result, for $\vdash P, Q \triangleright \Delta$, $P \simeq Q$ if and only if $(\!|P|\!) \simeq (\!|Q|\!)$.*

Via this result, we can show that the interpretation of a process is asynchronous if and only if it is invariant under composition with the asynchronous forwarder.

Lemma 6.6. *Consider a context $\Delta = a_1 : A_1, \ldots, a_n : A_n$ which does not contain a name and its coname. For $\vdash P \triangleright \Delta$, $[\![P]\!]$ is an asynchronous strategy if and only if $(\nu \widetilde{a})(P \mid [\widetilde{a} = \widetilde{b}]) \simeq P\{\widetilde{b}/\widetilde{a}\}$.*

In this case, we say that P is ***latency-independent***.

6.3 Finite Definability and Internal Processes

One of the goals of the correspondence between session types and game semantics is to be able to use session processes as a syntax for strategies. However not all coincident strategies are definable by a session process due to arbitrary coincidences. In this section, we turn our attention to the coincidence-free fragment of the model, corresponding to pre-strategies of [Castellan et al. 2018].

As for types, we now try to understand which conditions on coincident strategies are necessary for the converse to hold. A confusion-free coincidence-free strategy S on a game A is ***matching-exhaustive*** when if $\mathfrak{c}$ is a visible cell of S, then $lbl(\mathfrak{c})$ is a cell of A. This ensures that a strategy does not forget branches in an external choice. A coincidence-free strategy $S : A$ is ***internal*** when S is confusion-free and matching-exhaustive.

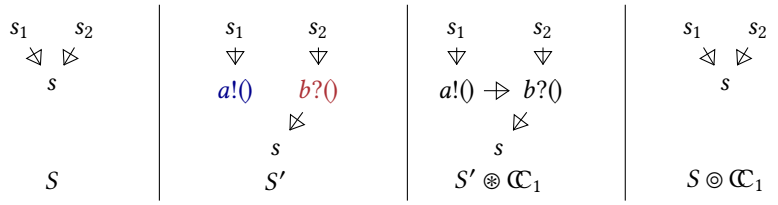
Lemma 6.7. *Let $\vdash P \triangleright \Delta$ be an internal session process. Then $[\![P]\!]$ is an internal strategy.*

Moreover, for internal strategies which are forests it is easy to define them by induction:

Lemma 6.8. *Let $S : [\![\Delta]\!]$ be a* finite *internal strategy such that S is a forest. Then there exists an internal process P such that $[\![P]\!] = S$.*

The main difficulty is to deal with joins. We need to define a strategy whose causal pattern can be arbitrary. First, let us define the causal complexity of a strategy. Given $s \in S$, we write $pred(s)$ for the set of its immediate predecessors, *i.e.*, the set of events $s' \in S$ such that $s' \rightarrowtriangle s$. If S is a forest, then $pred(s)$ is at most one for all $s \in S$. Then, we define the ***causal complexity*** of an event $s \in S$:

(1) $\mathrm{cc}(s) = 0$ if $\mathit{pred}(s)$ has cardinality ≤ 1; (2) else $\mathrm{cc}(s) = \sum_{s' \to_{\mathrm{P}} s} |[s']|$. We then define the ***causal complexity*** $\mathrm{cc}(S)$ of a finite internal strategy $S : A$ to be the sum of the causal complexity of all its events. Note that $\mathrm{cc}(S) = 0$ if and only if S is forest-like. We show how to inductively decrease the causal complexity of a strategy. Consider a join in $S : A$ as in the left of the diagram below. We then rewrite this strategy into S', which now plays on a larger arena $[\![a : \bot, b : 1]\!]^{\bot} \parallel A$. Here S' has a lower causal complexity than S but not $s_1 \leq s$. This dependence can be recovered by composition with asynchronous copycat, as illustrated below:



Lemma 6.9. *Let $S : [\![\Delta]\!]$ be an internal strategy. If S has causal complexity > 0, then there exists $S' : [\![a : \bot, b : 1]\!]^{\bot} \parallel A$ such that $\mathrm{cc}(S') < \mathrm{cc}(S)$ and $S \cong S' \circledcirc_{[\![a:\bot,b:1]\!]} \mathbb{CC}_{[\![1]\!]}$.*

Since asynchronous copycat and composition are definable, we obtain by induction:

Theorem 6.10. *If $S : [\![\Delta]\!]$ is finite internal, there exists an internal process P with $[\![P]\!] \cong S$.*

Since asynchronous strategies are automatically internal, we conclude that finite asynchronous strategies are all definable by processes. Hence there is a two-way correspondence between latency-independent processes and asynchronous strategies.

On the one hand, it is not clear the class of infinite event structures that internal processes (with recursion) define: it is more than regular event structures. On the other, augmenting the syntax with infinitary parallel composition and restriction is enough to define all infinite event structures.

7 FROM COINCIDENT STRATEGIES TO ASYNCHRONOUS STRATEGIES

This section presents an encoding of coincident strategies into asynchronous strategies, which explains how to represent synchronous strategies as particular asynchronous strategies. We encode synchronous actions as a pair of asynchronous actions of dual polarities, a request, followed by an acknowledgement. As an application of our correspondence, this encoding can be lifted at the level of types and processes.

Syntactically, a type T (resp. a game A) is unfolded into a protocol $\uparrow T$ (resp. a game $\uparrow A$) where every action is duplicated into a request and an acknowledgement. For instance, $\uparrow\mathbb{B} = (!\mathsf{tt}.?\mathsf{ack}) \oplus (!\mathsf{ff}.?\mathsf{ack})$.

Processes and strategies on T or A are lifted on $\uparrow T$ or $\uparrow A$. For instance, consider the process $P = a!\mathsf{tt}.b!\mathsf{ff}$ on the context $\Delta = a : \mathbb{B}, b : \mathbb{B}$. It is latency-*dependent* due to the direct syntactic dependence between two positive actions on ***unrelated*** channels, hence requires to know when its send has been received to be able to continue. It can be unfolded into a latency-independent process respecting the unfolded types:

$$\vdash P' := a!\mathsf{tt}.a?\mathsf{ack}.b!\mathsf{ff}.b?\mathsf{ack} \triangleright a : \uparrow\mathbb{B}, b : \uparrow\mathbb{B},$$

P' need not to know when its outputs are received since it can simply wait for the acknowledgement.

We first start by explaining how protocols are unfolded. Then § 7.1 defines the encoding of strategies, and § 7.2 characterises the image of the encoding. Finally, we study properties of the encoding in § 7.3 and show how it can be used to build a syntactic translation of (finite) processes.

Definition 7.1. Let A be a $\mathscr{L}$-game. We define $\uparrow A$ as follows:

Events $A \times \{\mathsf{r}, \mathsf{a}\}$
Causality generated by $(e, \mathsf{r}) \rightarrowtriangle (e, \mathsf{a})$ and $(e, \mathsf{a}) \rightarrowtriangle (e', \mathsf{r})$ when $e \rightarrowtriangle_A e'$.
Conflict $(e, \alpha) \# (e', \beta)$ when $e \# e'$.
Labelling (to $\mathscr{L} \times \{-, +\}$): $lbl(e, \mathsf{r}) = lbl(e)$ and $lbl(e, \mathsf{a}) = (\mathsf{ack}, -pol_A(e))$.

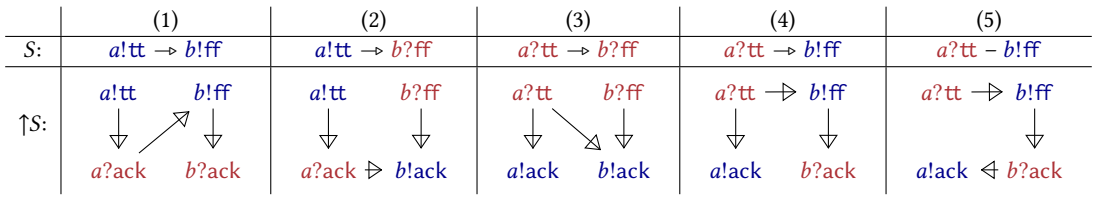
The translation preserves most properties and constructions on games. Moreover, through the results of § 4, this induces a translation on session types $T \mapsto \uparrow T$ that can be described syntactically:

$$\uparrow \mathsf{end} = \mathsf{end} \quad \uparrow \mathbf{t} = \mathbf{t} \quad \uparrow(\mu \mathbf{t}.T) = \mu \mathbf{t}.\uparrow T$$

$$\uparrow \&_{i \in I} ?l_i(\widetilde{S_i}).\, T_i = \&_{i \in I} ?l_i.\, !\mathsf{ack}\langle \widetilde{\uparrow S_i} \rangle.\, \uparrow T_i \qquad \uparrow \oplus_{i \in I} !l_i \langle \widetilde{S_i} \rangle.\, T_i = \oplus_{i \in I} !l_i.\, ?\mathsf{ack}(\widetilde{\uparrow S_i}).\, \uparrow T_i$$

7.1 Encoding of Strategies

We define an encoding from a coincident strategy S on A into an asynchronous strategy $\uparrow S$ on $\uparrow A$. Contrary to the translation of games, which can be easily carried at the syntax level for types, this translation relies on the causal structure offered by the model. The difficulty of this encoding is to translate causal patterns which are not necessarily courteous into courteous ones (*i.e.* $\ominus \rightarrowtriangle \oplus$). Before formally defining the encoding, we list the possible causal patterns in coincident strategies and what their encoding will be below.



In (1), there is an immediate causality between two positive actions which is translated to a strategy below. (4) is a translation of a courteous causality, while (5) is of a coincidence. They only differ in how the acknowledgement is handled: when translating a coincidence $a?\mathtt{tt} - b!\mathtt{ff}$, the acknowledgement of $a?\mathtt{tt}$ depends on the acknowledgement of $b!\mathtt{ff}$, which is not the case for the translation of $a?\mathtt{tt} \rightarrowtriangle b!\mathtt{ff}$. This translation was guided by a principle: there should be a bijection between downclosed subsets of the source partial order, and *complete* downclosed subsets of the target partial order, *i.e.* those closed under acknowledgements.

Defining the encoding in one step would technically involved, so we proceed in two steps. First, given a coincident strategy S on A, we build a coincidence-free strategy $\overline{S}$ on the game $\uparrow A$. Remember that S splits in $S = S_* \cup S_v$ where S_v is the set of visible events and S_* the set of internal events. Consider the set $\overline{S} := S_* \times \{*\} \cup S_v \times \{\mathsf{r}, \mathsf{a}\}$, along with labelling function $\overline{lbl_S} : \overline{S} \rightharpoonup \uparrow A$, mapping (s, α) to $(lbl_S(s), \alpha)$ when defined; and undefined otherwise. A subset $X \subseteq \overline{S}$ is consistent when its first projection $\pi_1(X)$ is consistent in S. The preorder $\leq_{\overline{S}}$ is generated by the transitive and reflexive closure of:

$$(s, \mathsf{r}) \prec (s, \mathsf{a}) \text{ for } s \in S$$
$$(s, \mathsf{r}) \prec (t, \mathsf{r}), (t, \mathsf{a}) \prec (s, \mathsf{a}) \text{ when } s^- \equiv t^+$$
$$(s, \mathsf{a}) \prec (t, \mathsf{r}) \text{ when } s \rightarrowtriangle t,\ s, t \in S_v$$
$$(s, \mathsf{a}) \prec (t, *) \quad \text{when } s \rightarrowtriangle t \text{ and } s \in S_v, t \in S_*$$
$$(s, *) \prec (t, \mathsf{r}) \quad \text{when } s \rightarrowtriangle t \text{ and } s \in S_*, t \in S_v$$
$$(s, *) \prec (t, *) \quad \text{when } s \rightarrowtriangle t \text{ and } s, t \in S_*$$

The resulting preorder still contains non-courteous causal links, but it is antisymmetric:

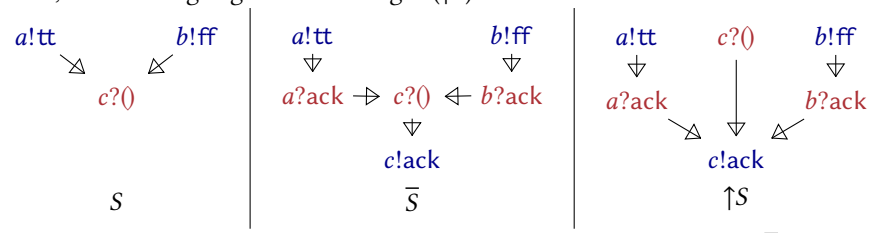
Lemma 7.2. *$\overline{S}$ is a coincidence-free strategy on $\uparrow A$.*

To turn $\overline{S}$ into an asynchronous strategy, we borrow this result from [Castellan et al. 2018]:

Lemma 7.3 ([Castellan et al. 2018]). *If S coincidence-free, then $\alpha_A \circledcirc_A S$ is asynchronous.*

As a result, we define $\uparrow S$ as the asynchronous strategy $\alpha_{\uparrow A} \circledcirc_{\uparrow A} \overline{S}$.

Example 7.4. Consider the process $P = (\nu d)(a!\mathtt{tt}.\overline{d}! \mid b!\mathtt{ff}.d?.\,c?)$. Its interpretation is on the left (S). Via $\overline{S}$, its encoding is given on the right ($\uparrow S$):



The strategy on the right is the interpretation of $(\nu dd')(a!\mathtt{tt}.\,a?\mathrm{ack}.\,\overline{d}! \mid b!\mathtt{ff}.\,b?\mathrm{ack}.\,\overline{d'}! \mid c?.\,d?.\,d'?.\,c!\mathrm{ack})$. It illustrates that the translation is not inductive on the syntax of processes but relies on the causal structure.

Because composing with copycat removes all non courteous causal links, $\uparrow S \cong \uparrow T$ does not imply $\overline{S} \cong \overline{T}$. However, by adding redundancy to the step from S to $\overline{S}$, the whole process is injective. Moreover, the encoding also preserves the categorical structure:

Theorem 7.5. *The operation $\uparrow$ defines a faithful functor $\mathbf{CG}_S \to \mathbf{CG}_A$,* i.e.,

$$(1)\ \uparrow \mathbb{CC}_A \cong \mathbb{C}_{\uparrow A} \qquad (2)\ \uparrow(R \circledcirc S) \cong \uparrow R \circledcirc \uparrow S \qquad (3)\ \uparrow R \cong \uparrow S \Rightarrow S \cong R$$

$\mathbf{CG}_S$ is thus isomorphic to a subcategory of $\mathbf{CG}_A$. We now characterise this subcategory.

7.2 Acknowledging Strategies

Consider an asynchronous strategy $S : \uparrow A$. Define $\downarrow S$ to be the set of $s \in S$ which are either internal events, or acknowledged requests, *i.e.*, requests such that there exists $s' \in S$ with $\sigma\, s \rightarrowtriangle \sigma\, s'$. Define the relation $\preceq_S$ on $\downarrow S$ as follows (reflexive but not transitive in general):

$$s \preceq_S s' := \begin{cases} s \leq_S s' & s' \text{ neutral} \\ s \leq_S s'_0 & s'_0 \text{ an acknowledgement of } s' \end{cases}$$

Lemma 7.6. *Let $\sigma : S \rightharpoonup A$ be a coincident strategy. There exists a function $\varphi : \downarrow\uparrow S \to S$ such that $s \preceq_S s'$ if and only if $\varphi\, s \leq \varphi\, s'$.*

Definition 7.7. An asynchronous strategy $\sigma : S \rightharpoonup \uparrow A$ is ***well-acknowledging*** when

(1) $X \in \mathrm{Con}_S$ iff $[X \setminus \{s \in X \mid \sigma\, s \text{ defined and an acknowledgement}\}] \in \mathrm{Con}_S$.
(2) The relation $\preceq_S$ is a preorder on $\downarrow S$ (*i.e.* it is transitive).
(3) If $s \preceq_S s'$ and $s' \preceq_S s$, then s and s' have distinct polarities.
(4) If s is a non-maximal request, then it is acknowledged.
(5) There is no internal s_0 with $s < s_0 < s'$ when s is a request and $\sigma\, s \rightarrowtriangle \sigma\, s'$.

Lemma 7.8. *For any $\sigma : S \rightharpoonup A$, the strategy $\uparrow\sigma$ is well-acknowledging.*

Given S an asynchronous strategy on $\uparrow A$, we write $\downarrow S$ for the triple $(\downarrow S, \preceq_S, \mathrm{Con}_S \cap \mathscr{P}(\downarrow S))$, with labelling mapping $s \in \downarrow S$ to a whenever $lbl_S(s) = (a, \mathsf{a})$.

Proposition 7.9. *If S is well-acknowledging, then $\downarrow S$ is a coincident strategy. Moreover $\uparrow\downarrow S \cong S$.*

Since the construction $\uparrow\cdot$ is easily seen to be injective on games, we obtain:

Theorem 7.10. *The category $\mathbf{CG}_S$ is isomorphic to the subcategory of $\mathbf{CG}_A$ whose objects are games of the form $\uparrow A$ and morphisms well-acknowledging (asynchronous) strategies.*

7.3 Properties of the Encoding

We show that our encoding is *sound*: it reflects weak bisimulation (hence barbed congruence)

Lemma 7.11. *For $S, R : A$, if $\uparrow S \approx \uparrow R$ then $S \approx R$.*

However, as the completeness direction of the encoding from synchrony into asynchrony fails in the π-calculus [Yoshida 1996], the converse direction does not hold(cf. Example E.1 in Appendix).

The semantic encoding can be lifted to finite processes via Theorem 6.10.

Definition 7.12. Given a *finite* process $\vdash P \triangleright \Delta$, define $\uparrow P$ to be a process defining the strategy $\uparrow[\![P]\!]$ on the game $\uparrow[\![\Delta]\!] \cong [\![\uparrow\Delta]\!]$ via Theorem 6.10.

By construction, $\vdash \uparrow P \triangleright \uparrow\Delta$. Using the full abstraction result, we deduce immediately that the encoding is sound:

Lemma 7.13. *For finite $\vdash P, Q \triangleright \Delta$, if $\uparrow P \simeq \uparrow Q$ then $P \simeq Q$.*

This allows any process which expects a synchronous network to be lifted to a process that has the same behaviour on a network where synchronising actions are not available. Unlike the traditional untyped encoding of the synchronous π-calculus into the asynchronous π-calculus, this encoding does not rely on name-passing but simply on passing acknowledgements.

8 RELATED WORK AND CONCLUSION

Related work. Recently, Disney and Flanagan [2015] applied game semantics to prove the soundness of typing system for extensions of the λ-calculus by translating them as strategies represented by processes. Honda and Yoshida [1999] recast the traditional encoding of the call-by-value λ-calculus into the π-calculus by Milner [1992] into a game semantics model for call-by-value PCF. Our paper applies the proof technique from the π-calculus (definability of *action testers* in [Hennessy 2007]) to game semantics to obtain full abstraction where the standard finite definability (which works only for may equivalence) is insufficient due to reduction-closedness of the barbed congruence.

On concurrent game semantics, Laird [2005] was the first to devise a model of the asynchronous π-calculus. The nondeterminism of the model being angelic, the model is only fully abstract for may equivalence. The model rests on plays as traces and thus is unable to account precisely for causality between actions. The departure from trace-based games model to truly concurrent ones was initiated by [Abramsky and Melliès 1999] using closure operator to represent deterministic computation. The first model using explicitly causal structures is due to [Melliès and Mimram 2007], which introduces the asynchronous copycat, as well as the local condition, *courtesy*, ensuring that it is an identity. This causal structure is used to prove positionality of innocent strategies and deduce a fully complete model of linear logic [Melliès 2005]. Eberhart et al. [2013] have a model of the synchronous π-calculus using similar ideas, fully abstract for fair convergence. However, their model does not support any notion of hiding, which makes the representation of programs very large; and they do not study the link between types and games. The first model of a π-calculus dialect based on this causal approach in game semantics is due to Sakayori and Tsukada [2017] using pomsets to devise a model of the asynchronous π-calculus, fully abstract for may. More recently, concurrent game semantics has been applied to provide a semantic proof of soundness for concurrent separation logic [Melliès and Stefanesco 2018].

The idea of representing coincidences as causal loops is first presented in [Ghica and Menaa 2011], which outlines how to build a sequential and deterministic model in game semantics of synchronous computation. It is also proposed in [Melliès 2019], to which our model is related. Since the domain of configurations of a coincident event structure can be seen as a category of positions,

our games and strategies arise as a special case of the abstract setting put forward in [Melliès 2019]. With this embedding, our synchronous copycats coincide, but it is not entirely clear to us what the status of composition is. Domains of configuration are not composed using pullbacks, but as a subjobject of the pullback where deadlocks have been removed. It would be interesting to investigate whether our category can be recovered as a subcategory of Melliès', maybe by changing the base category of polarities, which is the main parameter of the construction of [Melliès 2019].

The line of work [Crafa et al. 2007; Cristescu et al. 2015] also models processes by event structures, however in these models, labels carry information about the names that are passed or received, unlike in ours. This representation of actions makes the models combinatorially more involved and does not offer fully abstract or definability results.

Our definability result is related to [van Glabbeek and Vaandrager 2003], tackling the problem of definability of event structures by expressions of CCS or CSP. We define event structures using a name-passing calculi, the internal session π-calculus, rather than a value-passing one, which relies on the game structure.

The analysis of synchrony and asynchrony at the level of strategies is reminiscent to Selinger [Selinger 1997], where asynchrony properties on labelled transition systems are defined in terms of equations satisfied by the composition of the transition system with specific agents, representing buffers or queues, just like we do with asynchronous copycat. Our setting using games allows for more general LTSs than those simply generated by sets of input and output labels. We leave for future work understanding the exact relationship between the two frameworks.

Implementation. We have implemented a prototype for the model, available online at http://sessiontypesandgames.github.io/. The implementation is mainly an illustration of the model, and intended to help readers to familiarise themselves with the model. It computes the interpretation of a (recursion-free) process, and draws the coincident event structure. Coincident event structures are not represented using a graph representation as drawn in the paper, but as a labelled transition system enhanced with causal information. This allows us to represent infinite event structures lazily (even though we do not take advantage of this in the implementation). The transition system can then be explored (with possibly a bound) to recover the event structure, as a graph.

Because of this representation, the implementation does actually list the traces (where each event of the trace lists the events depended on causally), which only differ due to nondeterministic choices, and not due to the choice of the scheduler. For this reason, our implementation *directly* gives the minimal set of traces of a process, and as such is related to partial order reduction tools which approximate this minimal set (in more general contexts).

Future work. We believe that this model can be applied to other areas related to synchronous computation: for instance to the geometry of synthesis and hardware circuits [Ghica and Menaa 2011], or the Applied π-calculus [Abadi et al. 2017] where a truly concurrent model could help with checking cryptographic properties related to trace-equivalence [Baelde et al. 2015].

We would also like to extend the correspondence, to support: (1) races between different actions, in order to represent nonlinear channels or shared memory and (2) nonlinear "symmetric" actions (corresponding to unrestricted channels in the π-calculus and copy indices in strategies).

From the session types perspective, a link with causal game semantics could offer a causal account of asynchronous buffered session semantics [Gay and Vasconcelos 2010] and multiparty session types [Honda et al. 2008]; a correspondence with a relational model of a linear logic based session calculus [Atkey 2017]; a semantic proof of deadlock-freedom; a generalisation of binary session types to types that do not denote forest-like games.

ACKNOWLEDGEMENT

This work has been partially sponsored by: EPSRC EP/K034413/1, EP/K011715/1, EP/L00058X/1, EP/N027833/1, and EP/N028201/1. We would like to thank Pierre Clairambault, Sung-Shik Jongmans, and Hugo Paquet for helpful comments and discussions, as well as the POPL reviewers for their feedback.

REFERENCES

Martín Abadi, Bruno Blanchet, and Cédric Fournet. 2017. The Applied Pi Calculus: Mobile Values, New Names, and Secure Communication. *J. ACM* 65, 1, Article 1 (Oct. 2017), 41 pages. https://doi.org/10.1145/3127586

Samson Abramsky, Kohei Honda, and Guy McCusker. 1998. A Fully Abstract Game Semantics for General References *(LICS'98)*. IEEE Computer Society, Washington, DC, USA.

Samson Abramsky, Radha Jagadeesan, and Pasquale Malacaria. 2000. Full Abstraction for PCF. 163 (2000), 409–470.

Samson Abramsky, Pasquale Malacaria, and Radha Jagadeesan. 1994. Full Abstraction for PCF. In *TACS (Lecture Notes in Computer Science)*, Vol. 789. Springer, 1–15.

Samson Abramsky and Guy McCusker. 1999. Full Abstraction for Idealized Algol with Passive Expressions. Vol. 227. 3–42. https://doi.org/10.1016/S0304-3975(99)00047-X

Samson Abramsky and Paul-André Melliès. 1999. Concurrent Games and Full Completeness. In *14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999*. 431–442. https://doi.org/10.1109/LICS.1999.782638

Robert Atkey. 2017. Observed Communication Semantics for Classical Processes. In *ESOP*. 56–82.

David Baelde, Stéphanie Delaune, and Lucca Hirschi. 2015. Partial Order Reduction for Security Protocols. In *Proceedings of the 26th International Conference on Concurrency Theory (CONCUR'15) (Leibniz International Proceedings in Informatics)*, Luca Aceto and David de Frutos-Escrig (Eds.), Vol. 42. Leibniz-Zentrum für Informatik, Madrid, Spain, 497–510. https://doi.org/10.4230/LIPIcs.CONCUR.2015.497

Martin Berger, Kohei Honda, and Nobuko Yoshida. 2001. Sequentiality and the π-Calculus. In *Typed Lambda Calculi and Applications (Lecture Notes in Computer Science)*, Samson Abramsky (Ed.), Vol. 2044. Springer Berlin Heidelberg, 29–45. https://doi.org/10.1007/3-540-45413-6_7

Simon Castellan, Pierre Clairambault, Jonathan Hayman, and Glynn Winskel. 2018. Non-angelic Concurrent Game Semantics. In *Foundations of Software Science and Computation Structures - 21st International Conference, FOSSACS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings*. 3–19.

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

Gian Luca Cattani and Glynn Winskel. 1997. Presheaf models for concurrency. In *Computer Science Logic*, Dirk van Dalen and Marc Bezem (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 58–75.

Tzu-Chun Chen, Mariangiola Dezani-Ciancaglini, Alceste Scalas, and Nobuko Yoshida. 2017. On the Preciseness of Subtyping in Session Types. *LMCS* 13 (2017), 1–62. Issue 2.

Silvia Crafa, Daniele Varacca, and Nobuko Yoshida. 2007. Compositional Event Structure Semantics for the Internal π-calculus. In *Proceedings of the 18th International Conference on Concurrency Theory (CONCUR'07)*. Springer-Verlag, Berlin, Heidelberg, 317–317. http://dl.acm.org/citation.cfm?id=2392200.2392224

Ioana Domnina Cristescu, Jean Krivine, and Daniele Varacca. 2015. Rigid Families for CCS and the π-calculus. In *Theoretical Aspects of Computing - ICTAC 2015 - 12th International Colloquium Cali, Colombia, October 29-31, 2015, Proceedings*. 223–240.

Vincent Danos and Russell Harmer. 2002. Probabilistic Game Semantics. *ACM Trans. Comput. Logic* 3, 3 (July 2002), 359–382. https://doi.org/10.1145/507382.507385

Tim Disney and Cormac Flanagan. 2015. Game Semantics for Type Soundness. In *LICS (LICS'15)*. IEEE, 104–114.

Clovis Eberhart, Tom Hirschowitz, and Thomas Seiller. 2013. Fully-abstract concurrent games for pi. *CoRR* abs/1310.4306 (2013). http://arxiv.org/abs/1310.4306

Simon Gay and Malcolm Hole. 2005. Subtyping for Session Types in the Pi Calculus. *Acta Informatica* 42, 2/3 (2005), 191–225.

Simon Gay and Vasco T. Vasconcelos. 2010. Linear Type Theory for Asynchronous Session Types. *Journal of Functional Programming* 20, 1 (2010), 19–50.

Dan R. Ghica and Mohamed N. Menaa. 2011. Synchronous Game Semantics via Round Abstraction. In *Foundations of Software Science and Computational Structures*, Martin Hofmann (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 350–364.

Russell Harmer and Guy McCusker. 1999. A Fully Abstract Game Semantics for Finite Nondeterminism. In *14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999*. 422–430. https://doi.org/10.1109/LICS.1999.782637

Matthew Hennessy. 2007. *A Distributed Pi-Calculus.* CUP.
Kohei Honda and Mario Tokoro. 1991. An Object Calculus for Asynchronous Communication. In *ECOOP'91 (LNCS)*, Vol. 512. 133–147.
Kohei Honda, Vasco T. Vasconcelos, and Makoto Kubo. 1998. Language Primitives and Type Disciplines for Structured Communication-based Programming. In *ESOP (LNCS)*, Vol. 1381. Springer, 122–138.
Kohei Honda and Nobuko Yoshida. 1995. On Reduction-Based Process Semantics. *Theorical Computer Science* 151, 2 (1995), 437–486.
Kohei Honda and Nobuko Yoshida. 1999. Game-Theoretic Analysis of Call-by-Value Computation. *TCS* 221 (1999), 393–456.
Kohei Honda, Nobuko Yoshida, and Marco Carbone. 2008. Multiparty Asynchronous Session Types. In *POPL*. ACM, 273–284.
J. Martin E. Hyland and C. H. Luke Ong. 1994. *On Full Abstraction for PCF.* Technical Report. Imperial College, Department of Computing. 130pp pages.
J. Martin E. Hyland and C. H. Luke Ong. 2000. On Full Abstraction for PCF. *Inf. & Comp.* 163 (2000), 285–408.
J. M. E. Hyland and C.-H. Luke Ong. 1995. Pi-Calculus, Dialogue Games and PCF. In *FPCA*. ACM, 96–107.
Dimitrios Kouzapas and Nobuko Yoshida. 2015. Globally Governed Session Semantics. *LMCS* 10 (2015). Issue 4.
James Laird. 1997. Full Abstraction for Functional Languages with Control. In *Proceedings, 12th Annual IEEE Symposium on Logic in Computer Science, Warsaw, Poland, June 29 - July 2, 1997*. 58–67. https://doi.org/10.1109/LICS.1997.614931
James Laird. 2001. A Game Semantics of Idealized CSP. *Electr. Notes Theor. Comput. Sci.* 45 (2001), 232–257. https://doi.org/10.1016/S1571-0661(04)80965-4
Jim Laird. 2005. A Game Semantics of the Asynchronous *pi*-Calculus. In *CONCUR 2005 - Concurrency Theory, 16th International Conference, CONCUR 2005, San Francisco, CA, USA, August 23-26, 2005, Proceedings*. 51–65. https://doi.org/10.1007/11539452_8
Paul-André Melliès. 2005. Asynchronous Games 4: A Fully Complete Model of Propositional Linear Logic. In *20th IEEE Symposium on Logic in Computer Science (LICS 2005), 26-29 June 2005, Chicago, IL, USA, Proceedings*. 386–395. https://doi.org/10.1109/LICS.2005.6
Paul-André Melliès. 2019. Categorical combinatorics of scheduling and synchronization for games and strategies. In *Accepted for publication at POPL'19*.
Paul-André Melliès and Samuel Mimram. 2007. Asynchronous Games: Innocence Without Alternation. In *CONCUR 2007 - Concurrency Theory, 18th International Conference*. 395–411.
Paul-André Melliès and Léo Stefanesco. 2018. An Asynchronous Soundness Theorem for Concurrent Separation Logic. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*. 699–708. https://doi.org/10.1145/3209108.3209116
Robin Milner. 1992. Functions as Processes. *MSCS* 2, 2 (1992), 119–141.
Robin Milner and Davide Sangiorgi. 1992. Barbed Bisimulation. In *ICALP (LNCS)*, Vol. 623. Springer, 685–695.
Dimitris Mostrous and Nobuko Yoshida. 2015. Session Typing and Asynchronous Subtyping for the Higher-Order π-Calculus. *Inform. Comput.* 241 (2015), 227–263.
Vaughan R. Pratt. 1984. The Pomset Model of Parallel Processes: Unifying the Temporal and the Spatial. In *Seminar on Concurrency, Carnegie-Mellon University, Pittsburg, PA, USA, July 9-11, 1984*. 180–196. https://doi.org/10.1007/3-540-15670-4_9
Silvain Rideau and Glynn Winskel. 2011. Concurrent Strategies. In *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science, LICS 2011, June 21-24, 2011, Toronto, Ontario, Canada*. 409–418. https://doi.org/10.1109/LICS.2011.13
Ken Sakayori and Takeshi Tsukada. 2017. A Truly Concurrent Game Model of the Asynchronous Pi-Calculus. In *FoSSaCs*. 389–406.
Davide Sangiorgi. 1995. Internal Mobility and Agent Passing Calculi. In *Proc. ICALP'95*.
D. Sangiorgi. 1996. π-calculus, internal mobility and agent-passing calculi. *TCS* 167, 2 (1996), 235–274.
Peter Selinger. 1997. First-Order Axioms for Asynchrony. In *Proc. CONCUR '97*.
Kaku Takeuchi, Kohei Honda, and Makoto Kubo. 1994. An Interaction-based Language and its Typing System. In *PARLE'94 (LNCS)*, Vol. 817. 398–413.
P. S. Thiagarajan. 2002. Regular Event Structures and Finite Petri Nets: A Conjecture. In *Formal and Natural Computing - Essays Dedicated to Grzegorz Rozenberg [on occasion of his 60th birthday, March 14, 2002]*. 244–256. https://doi.org/10.1007/3-540-45711-9_14
Rob van Glabbeek and Frits Vaandrager. 2003. Bundle Event Structures and CCSP. In *CONCUR 2003 - Concurrency Theory*, Roberto Amadio and Denis Lugiez (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 57–71.
Glynn Winskel. 1982. Event structure semantics for CCS and related languages. 561–576. See also DAIMI Report PB-159, Computer Science Department, Aarhus University, 1983.
Glynn Winskel. 1986. Event Structures. In *Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986, Part II, Proceedings of an Advanced Course, Bad Honnef, 8.-19. September 1986*. 325–392. https://doi.org/10.1007/3-540-17906-2_31
Nobuko Yoshida. 1996. Graph Types for Monadic Mobile Processes.. In *FSTTCS (LNCS)*, Vol. 1180. Springer, 371–386.

Nobuko Yoshida and Vasco Thudichum Vasconcelos. 2007. Language Primitives and Type Discipline for Structured Communication-Based Programming Revisited: Two Systems for Higher-Order Session Communication. *Electr. Notes Theor. Comput. Sci.* 171, 4 (2007), 73–93.

A PROOFS OF SECTION 4

A.1 Link between games and types

A.1.1 Definability of games. Define the relation $\leq$ on type as the transitive and reflexive closure of the following rules:

- $\oplus_{i\in I}!l_i\langle\widetilde{T_i}\rangle.\, S \leq S$ and $\oplus_{i\in I}!l_i\langle\widetilde{T_i}\rangle.\, S \leq \overline{T_i^j}$
- $\&_{i\in I}?l_i(\widetilde{T_i}).\, S \leq S$ and $\&_{i\in I}?l_i(\widetilde{T_i}).\, S \leq T_i^j$
- If $T[\mu\mathbf{t}.\, T/\mathbf{t}] \leq S$ then $\mu\mathbf{t}.\, T \leq S$.

Lemma A.1. *If T is a type, there is a finite number of type T' such that $T \leq T'$.*

Proof. We can only unfold the rules a finite number of time and the syntactic tree is finitely branching. □

Lemma A.2. *Let T be a closed type and $x \in \mathscr{C}(\llbracket T\rrbracket)$. Then there exists a subtype T' of T such that $\llbracket T'\rrbracket \cong \llbracket T\rrbracket/x$.*

Proof. Easy induction. □

Lemma A.3. *If Δ is a typing environment, $\llbracket\Delta\rrbracket$ is confusion-free, regular, and forest-like.*

Proof. That $\llbracket\Delta\rrbracket$ is confusion-free and forest-like is trivial. We show that $\llbracket\Delta\rrbracket$ is regular. To do so it is enough to show that given a closed type T, $\llbracket T\rrbracket$ is regular. If there were a infinite number of configurations of $\llbracket T\rrbracket$ with non-isomorphic futures, then by Lemma A.1, there would be an infinite number of subtype T' of T contradicting Lemma A.2. □

To prove the converse, we will have to reconstruct the recursive variables, exploiting the fact that regular event structures are cyclic. Let A be a forest-like, confusion-free and regular $\mathscr{L}$-game. We write $\alpha, \beta, \ldots$ for cells of A. Note that by regularity they are all finite. They are naturally ordered: $\alpha \leq \beta$ when there exists $e \in \alpha$ and $e' \in \beta$ such that $e \leq e'$. Moreover if α is a cell of A, we write A/α for $A/[e)$ for any $e \in \alpha$. Let us consider a set C of cells of A such that:

(1) C is down-closed, if $\alpha \leq \beta \in C$ then $\alpha \in C$.
(2) C is exhaustive: if α is a cell of A, there exists a cell $\mathsf{nf}(\alpha) \leq \alpha$ in C.

By regularity of A, there exists a finite such C. Given a cell α of A, we write $[\alpha)$ for the set of cells strictly below α and $[\alpha]$ for the cells below α so that $[\alpha] = \{\alpha\} \cup [\alpha)$. In the following, we will consider the set $\mathbf{Games}(\mathscr{L})^{[\alpha)}$ for $\alpha \in C$, that we see as functions $\rho : [\alpha) \to \mathbf{Games}(\mathscr{L})$. By induction on α, we define:

$$F_\alpha : \mathbf{Games}(\mathscr{L})^{[\alpha)} \to \mathbf{Games}(\mathscr{L})$$
$$\rho \mapsto \mathrm{fix}(B \mapsto \sum_{e_i \in \alpha} msg(e_i)^{pol(e_i)} \cdot (\|_{e_i \twoheadrightarrow \beta}\ \xi_{\alpha,\mathsf{nf}\beta}))$$
$$\xi_{\alpha,\gamma} = \begin{cases} \rho(\gamma) & \text{if } \gamma < \alpha \\ B & \text{if } \gamma = \alpha \\ F_\gamma(\rho \cup \{\alpha \mapsto B\}) & \text{if } \alpha < \gamma \end{cases}$$

This inductive definition is well-founded since F_α only refers to F_β for $\alpha < \beta$ with $\beta \in C$, and C is finite. The parallel composition ranges over cells β which are successors to e_i (That is, β such that forall $e' \in \beta$, $e_i \twoheadrightarrow e'$). By regularity, there is a finite number of such β. Finally, the last clause of $\xi_{\alpha,\gamma}$ relies on the fact that if $e_i \twoheadrightarrow \beta$ then $[\beta) = [\alpha) \cup \{\alpha\}$. Let us write $\rho_\alpha : \beta \in [\alpha) \mapsto A/\beta$.

Lemma A.4. *For $\alpha \in C$, we have: $F_\alpha(\rho_\alpha) = A/\alpha$.*

Proof. First, let us notice that for any cell α, we have, because A is a forest:

$$(*) \qquad A/\alpha \cong \sum_{e_i \in \alpha} msg(e_i)^{pol(e_i)} \cdot (\|_{e_i \twoheadrightarrow \beta} A/\beta).$$

We proceed in two steps:

(1) We construct a family of embedding $(\iota_\alpha : F_\alpha(\rho_\alpha) \to A/\alpha)$ by induction on $\alpha \in C$
(2) By induction on $n \in \mathbb{N}$, we show that for every element $\alpha \in C$ and every element $e \in A/\alpha$ of depth n, e is in the image of ι_α.

From there we conclude that ι_α is surjective, hence an iso and we conclude.

(1) We proceed by induction on $\alpha \in C$. Let us define $G_\alpha : \mathbf{Games}(\mathscr{L})^{[\alpha]} \to \mathbf{Games}(\mathscr{L})$ as follows:

$$G_\alpha(\rho) = \sum_{e_i \in \alpha} msg(e_i)^{pol(e_i)} \cdot \|_{e_i \twoheadrightarrow \beta} \begin{pmatrix} \rho(\beta) & \text{if } \beta \leq \alpha \\ F_\beta(\rho) & \text{otherwise} \end{pmatrix}$$

so that $F_\alpha(\rho) = \mathrm{fix}(B \mapsto G_\alpha(\rho[\alpha := B]))$. By induction hypothesis, using $(*)$ we have $G_\alpha(\rho_\alpha[\alpha := A/\alpha]) \cong A/\alpha$. This means that A/α is a fixpoint of $B \mapsto G_\alpha(\rho_\alpha[\alpha := B])$, hence there is an embedding $\iota_\alpha : F_\alpha(\rho_\alpha) \hookrightarrow A/\alpha$.
(2) Let $n \in \mathbb{N}$ and consider $e \in A/\alpha$ of depth n. There are two cases: if $e \in \alpha$, then e is one of the e_i and is clearly in the domain of ι_α. Otherwise, there exists $e_i \in \alpha$ such that $e_i < e$. As a result, there exists a cell β such that $e \in A/\beta$ and $e_i \twoheadrightarrow \beta$. Clearly, e is going to be of depth $n-1$ in A/β. Since $A/\beta \cong A/\mathrm{nf}\beta$, there exists $e' \in A/\mathrm{nf}\beta$ corresponding to e. It is also of depth $n-1$, hence by induction it is in the image of $\iota_{\mathrm{nf}\beta}$. From there, it is easy to see that e is in the image of ι_α.

□

Lemma 4.8. *A $\mathscr{L}$-game is isomorphic (as $\mathscr{L}$-games) to a game of the form $[\![\Delta]\!]$ if and only if it is is forest-like, regular and confusion-free.*

Proof. Consider a game A satisfying the assumption. We use the notation of the previous lemma. Let us fix a family of type variable $(X_\beta)_{\beta \in C}$. For $\alpha \in C$, we define

$$T_\alpha = \mu X_\alpha . \sum_{e_i \in \alpha} !msg(e_i)\langle \overline{\xi_{\alpha,\mathrm{nf}\beta_1^i}}, \ldots, \overline{\xi_{\alpha,\mathrm{nf}\beta_n^i}} \rangle . \xi_{\alpha,\mathrm{nf}\beta_0} \qquad (\alpha \text{ positive cell})$$

$$T_\alpha = \mu X_\alpha . \sum_{e_i \in \alpha} ?msg(e_i)(\xi_{\alpha,\mathrm{nf}\beta_1^i}, \ldots, \xi_{\alpha,\mathrm{nf}\beta_n^i}) . \xi_{\alpha,\mathrm{nf}\beta_0} \qquad (\alpha \text{ negative cell})$$

where $\beta_0^i, \ldots, \beta_{n_i}^i$ are the cells succeeding to e_i, and

$$\xi_{\alpha,\beta} = \begin{cases} X_\beta & \text{if } \beta \leq \alpha \\ T_\beta & \text{otherwise} \end{cases}$$

By induction, we see that $\mathrm{fv}(T_\alpha) \subseteq \{X_\beta \mid \beta < \alpha\}$. Moreover, it is easy to derive that

$$[\![T_\alpha]\!](\rho) \cong F_\alpha(\beta \mapsto \rho(X_\beta)).$$

Then if α is a minimal cell of A, T_α is a closed type and $[\![T_\alpha]\!] \cong A/\alpha$ by Lemma A.4. As a result, writing $\alpha_1, \ldots, \alpha_n$, for the minimal cells, we have $[\![u_1 : T_{\alpha_1}, \ldots, u_n : T_{\alpha_n}]\!] = A$ as A is a forest. □

A.1.2 Equational theory on types.

Proposition 4.9. *For session types S, T without recursion, $S \equiv T$ if and only if $[\![S]\!] \cong [\![T]\!]$.*

Proof. By induction on $\equiv$, it is immediate to derive that $T \equiv T'$ implies $[\![T]\!] \cong [\![T']\!]$.

Conversely assume that there is an isomorphism $\varphi : [\![T]\!] \cong [\![T']\!]$. Because $\equiv$ allows us to pull the continuation inside the argument types, we can assume up to $\equiv$ that the subtypes in T have no continuation. Secondly, the rules in $\equiv$ allow us to remove the end types in argument types, so we will assume that the arguments types appearing are never empty. Write $\alpha = \{e_1, \dots, e_n\}$ for the initial cell of $[\![T]\!]$. For $1 \leq i \leq n$, write $(\beta_i^j)_j$ the successor cells of e_i. Assume for instance that α is a positive cell (the negative case is simlar). Then, we must have $T = \oplus_{1 \leq i \leq n} l_i \langle \widetilde{T_i} \rangle$. Since φ preserves polarities and messages, we must have $T' = \oplus_{1 \leq i \leq n} l_i \langle \widetilde{T_i'} \rangle$. Moreover, φ must restrict to an iso $[\![\widetilde{T_i}]\!]^\perp \cong [\![\widetilde{T_i'}]\!]^\perp$ for all i. From there, we deduce easily a permutation σ on $\{1, \dots, n_i\}$ such that $[\![T_i^j]\!] \cong [\![T_i'^{\sigma(j)}]\!]$ which by induction implies $T_i^j \equiv T'^{\sigma(j)}_{\sigma(i)}$. Then we conclude by the rule for $\oplus$ in $\equiv$. □

B PROOFS OF SECTION 5

Lemma 5.4. *Let $f : E \rightharpoonup E'$ be a map of coincident event structures. For $x \in \mathscr{C}(E)$, the function $E/x \rightharpoonup E'/f\,x$ obtained by restricting f is well-defined and a map of coincident event structures.*

Proof. We first show it is well-defined. If $e \in E/x$, then $f\,x \cup [f\,e] \subseteq f\,x \cup f[e] = f(x \cup [e]) \in \mathscr{C}(F)$. Moreover, since e is consistent with x, $f\,e$ cannot be in $f\,x$ by local injectivity. The other conditions are inherited from f. □

Lemma B.1. *Let E, F be coincident event structures and $\varphi : (\mathscr{C}(E), \subseteq) \cong (\mathscr{C}(F), \subseteq)$ be an order isomorphism, which preserves cardinality of configurations. Then there exists an isomorphism $f : E \cong F$ such that for all $x \in \mathscr{C}(E)$, we have $f\,x = \varphi\,x$.*

Moreover for any partial maps $g : E \rightharpoonup G$ and $h : F \rightharpoonup G$, such that $h(\varphi(x)) = g(x)$ for $x \in \mathscr{C}(E)$, then f can be chosen so that $h(f(e)) = g(e)$ for $e \in E$.

Proof. We construct $f(e)$ by induction on e. Write $X = \{e_1, \dots, e_n\}$ the coincidence of e inside $[e]$. Remember that $[e) = [e] \setminus X$. Assume f is defined on $[e)$.

Since φ is an isomorphism we have that $\varphi([e)) \overset{Y}{\text{—}\!\subset} \varphi([e])$ for some set Y. Because φ preserves cardinality, X and Y must be in bijection. If g and h are provided, pick any bijection $\alpha : X \simeq Y$ that respects g and h (using local injectivity); otherwise pick any bijection.

Now, augment f with the mappings in α.

This concludes the construction of f. Now we show that $f\,x = \varphi\,x$ for all $x \in \mathscr{C}(E)$ from which the result of the lemma follows. We prove this by induction on x; the base case being trivial.

If $x \overset{X}{\text{—}\!\subset} x'$, then we can consider the extension $[X)\text{–}\!\subset [X]$. As φ preserves extension squares (it is an isomorphism), $\varphi\,x' = \varphi\,x \cup (\varphi\,[X] \setminus \varphi\,[X))$. By induction, we have $\varphi\,x = f\,x$, and $\varphi\,[X] \setminus \varphi\,[X) = f\,[X)$ by construction of f. □

Lemma B.2. *Let $S, T : A$ be coincident strategies. If there exists an order-isomorphism $\varphi : (\mathscr{C}(S), \subseteq) \cong (\mathscr{C}(T), \subseteq)$ such that $lbl_T \circ \varphi = lbl_S : \mathscr{C}(S) \to \mathscr{C}(A)$, then $S \cong T$.*

Proof. The result follows from Lemma B.1 after having shown that φ must preserves cardinality. We proceed by induction on $x \in \mathscr{C}(S)$. If $x = \emptyset$, then $\varphi\,x$ must be the minimal element of $\mathscr{C}(S')$ hence $\varphi\,x = \emptyset$.

Otherwise, assume that $x \overset{X}{\text{—}\subset} y$. Because S is a coincident strategy, either X contains only visible events, and then

$$|\varphi\, y| - |\varphi\, x| = |lbl_{S'}(\varphi\, y)| - |lbl_{S'}(\varphi\, x)| = |lbl_S\, y| - |lbl_S\, x| = |y| - |x|$$

by local injectivity of S and S', or X contains a unique neutral element and we conclude easily. □

In some proofs, we will use an equivalent formulation of weak bisimulation on configurations rather than strategies:

Lemma B.3. *Given two A-labelled event structures S, S' whose labelling functions are maps. if there exists a relation $\mathcal{R} \subseteq \mathscr{C}(S) \times \mathscr{C}(S')$ such that:*

- $(\emptyset, \emptyset) \in \mathcal{R}$
- *If $x \mathcal{R} y$ then $lbl_S(x) = lbl_{S'}(y)$*
- *If $x \overset{s}{\text{—}\subset} x'$ with s neutral, then $y \subseteq y'$ for some internal extension y' of y and $x' \mathcal{R} y'$ (symmetric condition for y)*
- *If $x \overset{s}{\text{—}\subset} x'$ with $lbl_S(s) = a$, then $y \subseteq y_0 \overset{s'}{\text{—}\subset} y'$ for some internal extension y' of y and $x' \mathcal{R} y'$ (symmetric condition for y)*

Such a $\mathcal{R}$ will be called a **per-configuration weak bisimulation between S and S'**.

Proof. Assume that $S \approx S'$. Then define the relation $\mathcal{R} = \{(x, y) \in \mathscr{C}(S) \times \mathscr{C}(S') \mid S/x \approx S'/y\}$. It is easy to show that it satisfies the conditions of the lemma.

Conversely, consider a relation $\mathcal{R}$ satisfying the assumption of the lemma. Given $B \in \mathbf{Games}$, write $\mathcal{S}_B$ to be the least equivalence relation containing the pairs $(S/x, S'/y)$ for $(x, y) \in \mathcal{R}$ when $A/x = B$. It is then easy to show that this is a weak bisimulation, implying that $S \approx S'$ since $(\emptyset, \emptyset) \in \mathcal{R}$. □

Lemma B.4. *Consider $S, S' : A$ and $\mathcal{R}$ a per configuraiton weak bisimulation between them. Consider $x \mathcal{R} y$. If x has n concurrent visible extensions $X_1, \ldots, X_n$, leading to x', then there exists concurrent coincidences $Y_1, \ldots, Y_n$ such that $y \overset{Y_i}{\text{—}\subset}$ and $y \cup \bigcup Y_i = y'$.*

Proof. Let $x_i \in \mathscr{C}(S)$ such that $x \overset{X_i}{\text{—}\subset} x_i$. Moreover, we know that $y \subseteq y'$ with $x' \mathcal{R} y'$. Write Y_i for the correspondence of S' corresponding to X_i via the bijection $x' \simeq y'$. Assume that the Y_i are not all concurrent, eg. there is a minimal Y_1 below Y_2. Then $y \subseteq y \cup [Y_1) = y_1$. Because $x \mathcal{R} y$, we know that x extends to x_1 by neutral events such that $x_1 \mathcal{R} y$, and x_1 can do X_1. Because $Y_1 < Y_2$, y can only do Y_1 but not Y_2. Moreover, since X_2 is visible, and S a coincident strategy, the extension $x \subseteq x_1$ must be compatible with the extension $x \overset{X_2}{\text{—}\subset}$. Hence x_1 can do X_2, absurd since y cannot do Y_2. □

B.1 Construction of the interaction

We now show how to construct the interaction of $S : A \parallel B$ and $S' : A \parallel C$. It is defined through the notion of **secured bijections**.

Remember that any configuration x of $\mathscr{C}(S)$, we write x_* for the set of internal events of x. A **synchronised configuration** is a bijection $\varphi : x \parallel y_* \simeq x_* \parallel y$ (seen as its graph) where $x \in \mathscr{C}(S)$, $y \in \mathscr{C}(S')$ with $lbl_S(x) = lbl'_S(y)$ such that:

(1) if $s \in x$ is visible, then $\varphi(0, s) = (0, t)$ with $g\, s = f\, t$
(2) if $s \in x$ is internal, then $\varphi(0, s) = (1, s)$
(3) if $t \in y_*$, then $\varphi(1, t) = (0, t)$

We see such bijections as graphs, ie. as subsets of $(S \parallel S') \times (S \parallel S')$. We define $\pi_S : (S \parallel S') \times (S \parallel S') \rightharpoonup S$ as follows:

$$\pi_S((0,e),\_) = e \quad \pi_S(\_,\_) = \mathsf{undef} \qquad \pi_{S'}(\_,(1,f)) = f \quad \pi_{S'}(\_,\_) = \mathsf{undef}$$

Notice that for $\varphi : x \parallel y_* \simeq x_* \parallel y$, $\pi_S(\varphi) = x$ and $\pi_{S'}(\varphi) = y$.

Each synchronised configuration φ comes equipped with preorder defined as

$$(s,t) \leq_\varphi (s',t') := (s \leq_S s') \vee (t \leq_{S'} t').$$

We write $\simeq_\varphi$ for the equivalence relation generated by $\leq_\varphi$, and $max(\varphi)$ for the set of maximal events of φ for $\leq_\varphi$, ie. those events $e \in \varphi$ such that $e \leq_\varphi e'$ implies $e' \simeq_\varphi e$. A synchronised configuration is **prime** when two maximal events for $\leq_\varphi$ are equivalent. (In other terms, "φ has a top coincidence")

We say that $\varphi \overset{X}{\text{—}\subset} \varphi'$ when:

- We have $\varphi \subsetneq \varphi'$ and $X = \varphi' \setminus \varphi$.
- $\pi_S X$ splits in concurrent coincidences $Y_1, \ldots, Y_n \in S_\equiv$, and similarly for $\pi_{S'} X$.
- There is no cycle $\pi_S e_1 \equiv \pi_S e_2$, $\pi_{S'} e_2 \equiv \pi_{S'} e_3$, ..., $\pi_{S'} e_{n-1} \equiv \pi_{S'} e_1$ in X.

Say that φ is **reachable** when it exists a chain $\emptyset \overset{X_1}{\text{—}\subset} \varphi_1 \ldots \overset{X_n}{\text{—}\subset} \varphi_n = \varphi$. We can now define $S *_A S'$ as follows:

- Events: Pairs (φ, e) where φ is a prime reachable synchronised configurations and $e \in max(\varphi)$.
- Causality: $(\varphi, e) \leq (\varphi', e')$ when $\varphi \subseteq \varphi'$
- Conflict: $(\varphi, e)\#(\varphi', e')$ when there exists no reachable synchronised configuration (not necessarily prime) ψ such that $\varphi \cup \varphi' \subseteq \psi$.
- Labelling (to $A \parallel B \parallel C$): $lbl(\varphi, (s, s'))$ is equal to $lbl(s)$ or $lbl(s')$ (up to reindexing), if any is defined. If both are defined, note that $lbl_S(s) = lbl_{S'}(s')$ by definition of synchronised configurations.

Lemma B.5. *Suppose that $\varphi \overset{X}{\text{—}\subset} \varphi_1$ and $\varphi \overset{X'}{\text{—}\subset} \varphi_2$. If $\varphi_1 \cup \varphi_2 \subseteq \psi$ for some synchronised configuration ψ, then $\varphi_1 \overset{X'}{\text{—}\subset} \varphi_1 \cup \varphi_2$.*

Proof. Simple observation. □

Define $\Pi_S : S *_C S' \rightharpoonup S$ mapping (φ, e) to $\pi_S(e)$ and similarly for $\Pi_{S'}$.

Lemma B.6. *$S *_A S'$ is a coincident event structure and there is an order-isomorphism:*

$$\mathscr{C}(S *_C S') \cong \{\varphi \mid \varphi \text{ is a reachable synchronised configuration}\}$$

Proof. *$S *_A S'$ is a coincident event structure:* It is easy to see that since φ is finite $[\varphi, e]$ can only have a finite number of elements in $S *_A S'$. Moreover, assume that $(\varphi, e)\#(\varphi', e')$ and $(\varphi, e) \leq (\varphi'', e'')$. If there were a reachable synchronised configuration ψ containing $\varphi'' \cup \varphi'$, it would also contain $\varphi \cup \varphi'$, absurd.

Order-isomorphism. Consider a configuration z of $S *_A S'$. Write $\psi = \bigcup_{\varphi \in z} \varphi$, we show it is a reachable synchronised configuration. First, since z is a configuration, the set $\{\Pi_S \varphi\}_{\varphi \in z}$ is a pairwise-compatible set of configurations of S. As a result, its union is a configuration x of z (as S has binary conflict). Similarly, the union of the $\Pi_{S'} \varphi$ is a configuration y of S', and it easy to see that φ is indeed a synchronised configuration. Reachability follows by an easy induction using Lemma B.5. □

Lemma B.7. *$(S *_A S', \Pi_S, \Pi_{S'})$ is a common behaviour of S and S' on C.*

Proof. The functions Π_S and $\Pi_{S'}$ preserves labelling by construction of the labelling. As we have seen $\pi_S\varphi$ and $\pi_{S'}\varphi$ are configurations of S and S' respectively, and local injectivity is clear from the definition of Π_S and $\Pi_{S'}$. Moreover, if two coincident events $(\varphi, e) \equiv (\varphi, e')$, then $e \simeq_\varphi e'$. Suppose that $\pi_S e$ and $\pi_S e'$ are neither concurrent nor coincident , eg. $\pi_S e < \pi_S e'$. Then since $e \simeq_\varphi e'$, we must have $\pi_{S'} e' < \pi_{S'} e$. This directly contradicts reachability of φ. As a result Π_S and $\Pi_{S'}$ are indeed maps of coincident event structures from $S *_A S'$ to S and S' respectively.

We have already established condition (1) of common behaviours. Condition (2) arises again from reachability of φ and the definition of $\multimap$. □

Theorem 5.8. *There exists a common behaviour of S and S' $(S *_A S', \Pi_S, \Pi_{S'})$ which is the largest common behaviour,* i.e. *for every other common behaviour (T, α, β) there exists a unique simulation $(T, \alpha, \beta) \to (S *_A S', \Pi_S, \Pi_{S'})$.*

Proof. We are left to show the universal property. Consider a common behaviour (Z, α, β).

Existence. By local injectivity of α, given $x \in \mathscr{C}(Z)$, and property (1) of cones, we have a bijection $\varphi_x : \alpha x \parallel (\beta x)_* \simeq (\alpha\, x)_* \parallel \beta x$ which is synchronised. Moreover if $x \multimap x'$, write the $\varphi_1, \ldots, \varphi_n$ the immediate extensions of φ_x inside $\varphi_{x'}$, and write $X_i = \varphi_i \setminus \varphi_x$. The coincidence $x' \setminus x$ of Z splits in concurrent coincidences $Y_1, \ldots, Y_n$ of S (via α) and in concurrent coincidences $Z_1, \ldots, Z_p$ of S' (via β). Two elements of X are connected when they project via α or β inside the same coincidence. Then it is easy to see that each X_i corresponds to a connected component in this graph (acyclic by definition of common behaviours). This shows that $\varphi \overset{X_i}{\multimap} \varphi_i$ and the coincidences $X_1, \ldots, X_n$ are concurrent. From this result, we deduce that φ_x is reachable by iterating over a covering chain of x.

As a result, the map $Z \to S *_A S'$ defined by

$$e \mapsto \left(\varphi_{[e]}, \begin{cases} ((0, \alpha(e)), (1, \beta(e))) & \text{if both are defined} \\ ((0, \alpha(e)), (0, \alpha(e))) & \text{if only } \alpha \text{ defined} \\ ((1, \beta(e)), (1, \beta(e))) & \text{if only } \beta \text{ defined} \\ \mathsf{undef} & \text{if both are undefined} \end{cases} \right)$$

preserves configuration and make the triangles commute. Now if X is a coincidence of Z, we have seen above that it is mapped to concurrent coincidences of $S *_A S'$, thus proving that this map is inded a map of coincident event structures.

Uniqueness. It follows from joint injectivity of Π_S and $\Pi_{S'}$. □

We finish by some properties of the interaction that will be useful later on. Given a coincident event structure E and two maps $f : E \rightharpoonup F$ and $g : E \rightharpoonup F'$, and X a coincidence of E, we define a graph (X, f, g) whose nodes are elements of X. Two elements $e, e' \in X$ are connected when $f\, e \equiv f\, e'$ or $g\, e \equiv g\, e'$.

Lemma B.8. *For $z, w \in \mathscr{C}(S *_A S)$ with $z \subseteq w$, $z \multimap w$ if and only if the graph $(w \setminus z, f, g)$ is connected.*

Proof. We prove both implications.

($\Rightarrow$) Assume $z \multimap w$. By definition, we know that $\Pi_s(w \setminus z)$ splits as concurrent coincidences of E, $Y_1 \cup \ldots \cup Y_n$, and similarly $\Pi_{S'}(w \setminus z)$ splits as $Z_1 \cup \ldots \cup Z_p$, and let X' be a connected component of $(y \setminus x, f, g)$.
Since X' is connected, $\Pi_S X'$ is some $Y_{i_1} \cup \ldots \cup Y_{i_k}$. As a result, $x := \Pi_S(z \cup X') = \Pi_S z \cup Y_{i_1} \cup \ldots \cup Y_{i_n} \in \mathscr{C}(S)$. Similarly, $y := \Pi_{S'}(z \cup X') \in \mathscr{C}(S')$. This implies that $z \cup X'$ is the graph of a synchronised configuration. Moreover, since w is reachable, so is $z \cup X'$, hence $z \cup X' \in \mathscr{C}(S *_C S')$. This implies that $X = X'$ and X is connected.

($\Leftarrow$) We prove the result by contradiction: assume that $x \mathrel{-\!\!\subset} x' \subsetneq z$ for some $x' \in \mathscr{C}(S *_A S')$. Since the graph for $z \setminus x$ is connected, there must exist $e \in x' \setminus x$ and $e \in z \setminus x'$ such that, say $\Pi_S e \equiv_{\Pi_S z} \Pi_S e'$. This is absurd, as $\Pi_S x'$ is a subconfiguration of $\Pi_S z$ and contains $\Pi_S e$, it should also contain $\Pi_S e'$. Hence we must have $x \mathrel{-\!\!\subset} z$.

□

We now characterise minimal conflict in the interaction:

Lemma B.9. *Let $x \in \mathscr{C}(S *_A S')$ such that x has two incompatible extensions of X and X'. Then either there exist $Y \subseteq \Pi_1 X$ and $Y' \subseteq \Pi_1 X'$, such that Y and Y' are incompatible extensions of $\Pi_1 x$, or the same thing for $\Pi_2 x, \Pi_2 X$ and $\Pi_2 X'$.*

Proof. Write φ for the synchronised configuration associated to x and ψ, ψ' those associated to $x \cup X$ and $x \cup X'$. Assume that the result does not hold. This implies that $\psi \cup \psi'$ is a synchronised configuration. Since a coincidence of $\psi \cup \psi'$ is a coincidence of ψ or of ψ', it is reachable, hence shows that $x \cup X \cup X' \in \mathscr{C}(S *_A S')$ which is absurd. □

B.2 Properties of hidng

We need to show a few properties of the hiding of coincident event structures before moving on. Remember that given a coincident event structure E and a subset $V \subseteq E$, the coincident event structure $E \downarrow V$ has event V and causality and conflict inherited from E.

Lemma B.10 (Unique witness). *Let E be a coincident event structure and $x, y \in \mathscr{C}(E)$ such that every maximal coincidence of x and y intersect V. Then $x \cap V = y \cap V$ implies $x = y$.*

Proof. Since $x \cap V = y \cap V$, every maximal coincidence of x intersects a coincidence of y and vice versa, hence $max(x) = max(y)$. Since $x = [max(x)]$ and similarly for y, we get $x = y$. □

Lemma B.11. *Let E be a coincident event structure and $V \subseteq E$. For $x \in E$, there exists a unique $y \subseteq x$ such that $x \cap V = y \cap V$ and every maximal coincidence of y intersects V.*

Proof. We proceed by induction on x. If all maximal coincidences of x intersect V, then we are done. Otherwise, assume there is a maximal coincidence X of x disjoint from V. By induction, there exists $y \subseteq x \setminus X \subseteq x$ such that $y \cap V = (x \setminus X) \cap V = x \cap V$ whose maximal coincidence intersect V as desired. Unicity comes from Lemma B.10 □

Lemma B.12. *Configurations x of $E \downarrow V$ are in one-to-one correspondence with configurations $y \in \mathscr{C}(E)$ whose maximal coincidences all intersect V.*

Proof. Consider $x \in \mathscr{C}(E \downarrow V)$. Define $y := [x]$ to be the downclosure of x in E. By construction all maximal coincidences of y intersect V. Conversely given $y \in \mathscr{C}(E)$, it is easy to see that $y \cap V \in \mathscr{C}(E \downarrow V)$. Moreover, the mapping $x \mapsto [x] \mapsto [x] \cap V$ is the identity on configurations of $E \downarrow V$.

Conversely if $y \in \mathscr{C}(E)$ is such that its maximal coincidences all intersect V, then let $y' := [y \cap V]$ to be the smallest configuration containing $y \cap V$. By Lemma B.11, $y = y'$. □

Lemma B.13. *Let E be a coincident event structure and $V \subseteq E$.*

- *If $x \overset{X}{\mathrel{-\!\!\subset}} x'$ in E, with $X \cap V \neq \emptyset$ then $x \cap V \overset{X \cap V}{\mathrel{-\!\!\subset}} x' \cap V$ in $E \downarrow V$.*
- *If $x \cap V \overset{X}{\mathrel{-\!\!\subset}} y$ in $E \downarrow V$ and $x \cup y \in \mathrm{Con}_E$, then there exists $x \subseteq x'$ such that:*

(1) $x' \cap V = y$,

(2) $x \subseteq x_0 \overset{X \cup Z}{\mathrel{-\!\!\subset}} x'$

Proof. The first point is a simple observation. Any configuration between $x \cap V$ and $x' \cap V$ would give, by downclosure a configuration of E between x and x'.

For the second point, let $x' = [x \cup y]$. It is clear that $x' \cap V = y$. Moreover, a covering chain from x to x' must go through an atomic step of the form $X \cup Z$ since X is a set of coincident event of x'. Moreover, since the maximal coincidence of x' not in x intersects V, this step must be the last one. □

We now investigate properties of hiding with respect to maps.

Lemma B.14. *Let $f : E \rightharpoonup F$ be a map of coincident event structures and $dom(f) \subseteq V \subseteq E$. The restriction of f defines a map $f \downarrow V : E \downarrow V \rightharpoonup F$.*

Proof. Since $dom(f) \subseteq V$, $f(x) = f(x \cap V)$. This means that $f \downarrow V$ does preserve configurations. Local injectivity follows from that of f.

If $y \overset{X}{\text{—}\!\subset} y'$ in $E \downarrow V$, consider $x' := [y']$ and $x := [y]$. By Lemma B.13, we have that

$$x \subseteq x_0 \overset{X \cup Z}{\text{—}\!\subset} x_1 \subseteq x'.$$

It is easy to see that $f(x) = f(x_0)$ and $f(x') = f(x_1)$, and $f(X \cup Z) = f(X)$. Hence applying the fact that f is a map of event structure we get that $f(X)$ splits in $Y_1, \ldots, Y_n$ concurrent extensions of $f(x)$. This concludes the proof. □

Definition B.15 (Hiding maps). Let $f : A \rightharpoonup B$ be a (partial) map of coincident event structures. The following are equivalent:

(1) Writing V for the domain of f, there is an isomorphism $\varphi : A \downarrow V \cong B$ such that $A \rightharpoonup A \downarrow V \xrightarrow{\varphi} B = f$
(2) There exists a **hiding witness** for f which is a monotonic map $wit_f : \mathscr{C}(B) \to \mathscr{C}(A)$ with $f \circ wit_f(x) = x$ for $x \in \mathscr{C}(F)$ and $wit_f \circ f(x) \subseteq x$ for $x \in \mathscr{C}(A)$.

Proof. (1) $\Rightarrow$ (2). The hiding witness of f is $y \in \mathscr{C}(B) \mapsto [\varphi^{-1}(y)] \in \mathscr{C}(A)$.

(2) $\Rightarrow$ (1). We have a monotonic map $\alpha : \mathscr{C}(A \downarrow V) \to \mathscr{C}(B)$ defined as $x \mapsto f([x])$. Since $V = dom(f)$, x and $f([x])$ has the same cardinal by injectivity. Conversely, define $\beta : y \in \mathscr{C}(B) \mapsto wit_f(y) \cap V \in \mathscr{C}(A \downarrow V)$ which is also monotonic.

First, notice that $wit_f(f\,x) = x$ if all the top coincidences of x intersect V. This is due to $wit_f(f\,x)) \subseteq x$ and $f(wit_f(f\,x)) = f\,x$ hence by local injectivity $wit_f(f\,x) \cap V = x \cap V$. By Lemma B.11, we conclude $wit_f(f\,x) = x$.

We have:

$$\alpha(\beta(y)) = f\,[wit_f(y) \cap V] = f(wit_f(y)) = y$$
$$\text{because } wit_f(y) \cap V = [wit_f(y) \cap V] \cap V \text{ as } V = dom f$$
$$\beta(\alpha(x)) = wit_f(f[x]) \cap V = [x] \cap V = x$$
$$\text{because the top coincidences of } [x] \text{ intersect } V \text{ by construction}$$

Hence this order-isomorphism induces the desired isomorphism $A \downarrow V \cong B$ by Lemma B.1. □

B.3 Definition of the composition

Lemma 5.10. *For any coincident pre-strategy S on A, we have $S \approx \mathscr{E}(S)$.*

Proof. Write E_S for the set of essential events of S. We construct $\mathcal{R} = \{(x, y) \in \mathscr{C}(S) \times \mathscr{C}(\mathscr{E}(S)) \mid x \cap E_S = y\}$. We show it is a weak bisimulation.

Consider $x \mathcal{R} y$.

- If $x \overset{X}{\text{—}\subset} x'$: if $X \cap E_S = \emptyset$, then $x' \mathcal{R} y$ and we are done. Otherwise, by Lemma B.13, $y \overset{X \cap E_S}{\text{—}\subset} x' \cap E_S$. Since E_S contains all visible events, we can conclude.
- If $y \overset{Y}{\text{—}\subset} y'$: write $x_0 = [y']$. If x_0 and x are not consistent, then there must exist a minimal conflict $s \sim\!\!\sim s'$ with $s \in x_0$ and $s' \in x$. Since $x \cap E_S \subseteq y'$ which is consistent, both at least one of the coincidences of s and s' do not contain a visible event. Assume it is that of s. Without loss of generality, we can assume s to be the lowest in its coincidence for the map $S \to \mathbb{N}$. Then, $s \in E_S$ by definition hence $s \in y'$ - contradicting the consistency of y'.

□

Lemma B.16. *Let X be a coincidence of $\mathscr{C}(S' \circledast S)$. Then $(X, \Pi_S, \Pi_{S'})$ is connected and acyclic. Moreover, if S and S' are strategies, nodes of $(X, \Pi_S, \Pi_{S'})$ have degree at most two, and if $e \in X$ has degree two, then $S(\Pi_S e) = S'(\Pi_{S'} e) \in B$.*

Proof. Since there are no coincidences in A or C, (X, Π_1, Π_2) and $(X, \Pi_S, \Pi_{S'})$ are the same graph, that we know to be acyclic and connected by Lemma B.8. □

Lemma B.17. *Let $x \in \mathscr{C}(S *_A S')$ with two incompatible extensions X and X'. Then $\Pi_S X$ and $\Pi_S X'$ are incompatible extensions of $\Pi_S x$ or $\Pi_{S'} X$, $\Pi_{S'} X'$ of $\Pi_{S'} x$.*

Proof. Direct consequence of B.9: if a conflict occurs in the C component of Π_1 or the A component Π_2, then they occur in the game and must be reflected in the other strategy. □

Lemma B.18. *Let $S : A^\perp \parr B$ and $S' : B^\perp \parr C$ be coincident pre-strategies.*

(a) *If S and S' are secret, so are $S' \circledast S$ and $S' \circledcirc S$.*
(b) *If S and S' are coincident strategies, so is $S' \circledcirc S$.*

Proof. (a) Assume that S and S' are secret.

(1) First, we show that $S' \circledast S$ is secret. Consider two coincidences X and X' involved in a minimal conflict at $x \in \mathscr{C}(S' \circledast S)$. This conflict must originate from S or S' by Lemma B.17; assume it is S. This means that there exists $Y \subseteq \Pi_S(X)$ and $Y' \subseteq \Pi_S(X')$ which are incompatible extensions of $\Pi_S x$. Applying the fact that S is secret, we get two cases. Either Y and Y' are singleton reduced to an internal event; then so must be X and X'. Or there exists negative $s \in Y$ and $s' \in Y$. If they are projected to A, then the corresponding events in X and X' are also negative as desired. Otherwise, they are mapped to B, which means that there exists $t \in \Pi_{S'}(X)$ and $t' \in \Pi_{S'}(X')$ with $lbl_S(s) = lbl_{S'}(t)$ and $lbl_S\, s' = lbl_{S'}\, t'$. Those must be positive, hence since they are in immediate conflict, they must be coincident to t_1, t_1' negative. If they are mapped to C, then we are done. Otherwise we continue. We know we do not reach a cycle because $(X, \Pi_S, \Pi_{S'})$ is acyclic, hence this must terminate on negative events mapped to $A^\perp \parr C$ as desired.
(2) We conclude that $S' \circledcirc S$ is secret. If we have two coincidences X and X' involved in a minimal conflict in $S' \circledcirc S$, then this means that we have a minimal conflict $X \cup I$ and $X' \cup I'$ in $S' \circledast S$. The previous point that shows that either $X \cup I$ and $X' \cup I'$ are singletons reduced to internal events of S or S', which means that $I = I' = \emptyset$; or there exist $s \in X \cup I$, $s' \in X' \cup I'$ negative in conflict in the game. Since visible events are not hidden, it follows that $s \in X$ and $s' \in X'$ as desired.

(b) Assume that S and S' are strategies. We already know that $S' \circledcirc S$ is secret. We show property (1) and (2) of strategies:
(1) Let X be a coincidence of $S' \circledcirc S$. Then we have in $S' \circledast S$:

$$[X) \overset{X \cup Z}{\text{—}\subset} [X].$$

where $X \cup Z$ is a coincidence of $S' \circledast S$. By Lemma B.8, we know that $(X \cup Z, \Pi_S, \Pi_{S'})$ is a connected acyclic graph with all edges of degree below two. As a result it must be a line. If X contains a neutral event e for S or for S', then $\{e\}$ would be connected, hence $X = \{e\}$. Otherwise, if X contains only visible events, then they must be the endpoints of the line. As a result X must be of size below two. Moreover, it is an easy induction on the length of the line that they must have distinct polarity.

(2) Same reason: only the internal events involved in minimal conflicts are kept.

□

Lemma 5.11. *$T \circledcirc S$ is a coincident strategy on $A^\perp \parallel C$.*

Proof. Consequence of the previous lemma. □

B.4 Categorical structure

Lifting maps. We now explain how to lift a map of coincident event structures to a coincident strategy. Consider an injective map $f : A \to B$ which preserves minimal conflicts. (If X is downclosed in A, and $f\,X \notin \mathscr{C}(B)$ then $X \notin \mathscr{C}(A)$.)

Lemma B.19. *The coincident event structure $\mathbb{CC}_A$ with labelling function $\overline{f} := A \parallel f : \mathbb{CC}_A \to A^\perp \parallel B$ is a coincident strategy $\overline{f} : A^\perp \parallel B$.*

Proof. *Coincident strategy.* Since $\overline{f}$ does not have internal events, conditions (1), (2) are vacuous. We check condition (3). Consider two coincidences $X = \{a, f\,a\}$ and $X' = \{b, f\,b\}$ in minimal conflict in S_f at $x = x_A \parallel x_B$. This means that $x \cup X \cup X'$ is not a configuration of $A \parallel B$. Since f preserves minimal conflicts, and also reflects them by virtue of being a map, both $x_A \cup \{a, b\}$ and $x_B \cup \{f\,a, f\,b\}$ are not consistent in A and B. Moreover, by race-freeness of A, a and b have the same polarity. By definition of the polarities on $A^\perp \parallel B$, then either a and b must be negative, or $f\,a$ and $f\,b$ must be, as f preserves polarities and we can conclude. □

We now show that this operation is well-behaved with respect to composition.

Lemma B.20. *Let $S : A \parallel B$ be a coincident strategy and $f : B \to C$ an injective and conflict-preserving map. For $x \in \mathscr{C}(S)$ with image x_B in B, there exists a configuration $y \in \overline{f} \circledast S$ such that $\Pi_1 y = x \parallel x_B$. Moreover, the top coincidences of y all contain one essential event of $\overline{f} \circledast S$.*

Proof. Write $lbl_S\, x = x_A \parallel x_B$. We show that there exists a configuration of $\mathscr{C}(S \parallel C) \times \mathscr{C}(A \parallel S_f)$ whose first projection is $x \parallel x_B$, which allows us to conclude by Lemma B.6. There is a synchronised configuration $\varphi_x : x \parallel x_B \simeq x_* \parallel x_A \parallel x_B \parallel x_B$ obtained from local injectivity of S. This synchronised configuration is easily shown to be reachable.

Finally, consider a top coincidence X of φ_x not intersecting an essential events. It projects to top coincidences $X_1, \ldots, X_n$ of x in S. By assumption, these top coincidences are not internal, since as they are involved in a minimal conflict, and singletons, they would also occur in the interaction. So they must all contain only events in B which are hidden by $\mathscr{E}(\cdot)$. In that case, X must also contain their image by f by definition of the interaction, which are visible in $\overline{f} \circledcirc S$, hence a contradiction. □

Lemma B.21. *Let $S : A^\perp \parallel B$ be an essential coincident strategy. For any injective $f : B \to C$, then $\overline{f} \circledcirc S \cong (S, (A \parallel f) \circ lbl_S)$.*

Proof. We show that their underlying configuration domains are isomorphic, in a way that preserves projection to the game, hence by Lemma B.2, we conclude.

Consider a configuration x of $(A \parallel f) \circ S$ with image $x_A \parallel f(x_B)$ on the game. By Lemma B.20, there exists $z \in \overline{f} \circledast S$ such that $\Pi_1 z = x \parallel f(x_B)$. Mapping x to $z \cap E_{\overline{f} \circledast S}$ defines a monotonic mapping $\mathscr{C}(S) \to \mathscr{C}(\overline{f} \circledcirc S)$.

Injectivity. Assume that we have two configurations x, x' of $(A \parallel f) \circ S$ giving birth to two configurations z, z' with top coincidences containing essential events, such that $z \cap E_{\overline{f} \circledast S} = z' \cap E_{\overline{f} \circledast S}$. By Lemma B.10, we conclude that $z = z'$ and then $x = x'$ by projecting.

Surjectivity. Given $w \in \mathscr{C}(\overline{f} \circledcirc S)$, we first form $z = [w] \in \mathscr{C}(\overline{f} \circledast S)$ and then we project this configuration to its component on S. This yields the desired inverse map: $\mathscr{C}(\overline{f} \circledcirc S) \to \mathscr{C}(S)$. □

Proposition 5.12. *For a coincident strategy S on A, $\mathbb{CC}_A \circledcirc S \cong \mathscr{E}(S)$.*

Proof. Consequence of Lemma B.21. □

B.4.1 Associativity. We first start with a generalisation of the *Zipping lemma*:

Lemma B.22. *Let $h : S \rightarrowtail S'$ be a hiding map making the following diagram commute:*

$$\begin{array}{ccc} S & \xrightarrow{\quad h \quad} & S' \\ {\scriptstyle lbl_S}\downarrow & & \downarrow{\scriptstyle lbl_{S'}} \\ A \parallel B \parallel C & \longrightarrow & A \parallel C \end{array}$$

where S, S' are agents.

Then for any $\rho : U \to C \parallel D$, the morphism $U \circledast h : U \circledast S \rightarrowtail U \circledast S'$ defined using the universal property of $U \circledast S'$ is a hiding map.

Proof. Configurations of $U \circledast S'$ correspond to pairs $x_{S'} \parallel x_D$ and $x_A \parallel x_U$ such that:

$$lbl_S(x'_S) = x_A \parallel x_C \qquad \rho x_U = x_C = x_D$$

for some $x_C \in \mathscr{C}(C)$. From such a pair, we can construct the pair $wit_h(x_{S'}) \parallel x_D$ and $x_A \parallel x_B \parallel x_U$ where x_B is obtained by $lbl_S(wit_h(x_{S'})) = x_A \parallel x_B \parallel x_C$. The induced bijection is reachable, as the original pair was. This defines a map

$$wit_{U \circledast h} : \mathscr{C}(U \circledast S') \to \mathscr{C}(U \circledast S)$$

which satisfies the required assumptions. □

Lemma B.23. *The composition $\circledcirc$ is associative*

Proof. Consider $S : A^\perp \parallel B, U : B^\perp \parallel C$ and $V : C^\perp \parallel D$. First, it is easy to show that interaction is associative (using the universal property), hence we have an isomorphism:

$$a : (V \circledast U) \circledast S \cong V \circledast (U \circledast S).$$

By using the Zipping Lemma, we have the following two hiding maps:

$$V \circledast (U \circledast S) \xrightarrow{V \circledast h_{S,U}} V \circledast (U \circledcirc S) \xrightarrow{h_{U \circledcirc \rho, S}} V \circledcirc (U \circledcirc S)$$

where $h_{S,U} : U \circledast S \rightarrowtail U \circledcirc S$ is the hiding map of the composition.

Because hiding maps are stable under composition, and isomorphisms are special cases of hiding maps; by pre-composing the hiding map by the isomorphism on the interaction, we get the hiding map:

$$(V \circledast U) \circledast S \cong V \circledast (U \circledast S) \rightarrowtail V \circledcirc (U \circledcirc S).$$

Since its domain is clearly $(U \circledcirc T) \circledcirc S$, we get the desired result:

$$(V \circledcirc U) \circledcirc S \cong V \circledcirc (U \circledcirc S).$$

□

Theorem 5.13. *We obtain a compact-closed category* $\mathbf{CG}_S$ *whose objects are games, and morphisms are essential coincident strategies up to isomorphism.*

Proof. That it is a category follows from Proposition 5.12, and Lemma B.23. Compact-closed structure is obtained by noticing that the monoidal operation $\|$ lifts to a monoidal operation on $\mathbf{CG}_S$. □

Lemma 5.16. *For* $S : A$, *we have* $L_e \circledcirc S \cong e \rightarrowtriangle S$.

Proof. We establish that their configuration domains are isomorphic. We have on one hand:

$$\mathscr{C}(e \rightarrowtriangle S) \cong \{x_S \parallel x_e \in \mathscr{C}(S \parallel e) \mid (x_S)_v \neq \emptyset \Rightarrow x_e \neq \emptyset\}.$$

On the other hand, if $x \in \mathscr{C}(L_e \circledcirc S)$, it is easy to see that if $\Pi_{L_e}[x]$ is empty, then $\Pi_S[x]$ can only contain internal. From there it is easy to see that $\mathscr{C}(L_e \circledcirc S)$ is isomorphic of the right-hand side of the right equation via $x \mapsto \Pi_S[x] \parallel \Pi_{L_e}[x] \cap \{e\}$. □

Lemma 5.17. $\mathbf{CG}_S(A)$ *along with inclusion forms an* ω*-CPO.*

Proof. First, it easy to see that coincident event structures are closed under least upper bounds of ω-chain by simply constructing the union of a ascending chain of coincident event structures. Then, all conditions to be a coincident strategy are *local*, ie. involve only a finite number of events, which must occur at a finite stage, hence the least upper bound of an ω-chain of coincident strategies must be a coincident strategies. □

Lemma 5.18. *For a coincident strategy* $S : A^\perp \parallel B$, *the operation* $R : B^\perp \parallel C \mapsto R \circledcirc_B S$ *is a continuous map* $\mathbf{CG}_S(B, C) \to \mathbf{CG}_S(A, C)$.

Proof. Straightforward. If $S \subseteq S'$, then it is easy to see that $R \circledcirc S \subseteq R \circledcirc S'$. That it preserves least upper bounds of ω-chain is also a simple observation. □

B.5 Behavioural equivalences

Lemma 5.20. *If* $S \approx S' : A^\perp \parallel B$, *then for all* $R : B^\perp \parallel C$, *then* $R \circledcirc S \approx R \circledcirc S'$.

Proof. Consider $S, S' : A^\perp \parallel B$ which are weakly bisimilar and $U : B^\perp \parallel C$. We use characterisation of weak bisimulation of Lemma B.3.

Consider $\mathcal{R}$ a relation between configurations S and S' satisfying the assumptions of the lemma. Define $U \circledast R = \{(x, y) \in \mathscr{C}(U \circledast S) \times \mathscr{C}(U \circledast S') \mid \Pi_U x = \Pi_U y \wedge (\Pi_S x) \mathcal{R} (\Pi_{S'} y)\}$.

We show it satisfies the conditions of the lemma. Consider $x \in \mathscr{C}(U \circledast S)$ and $y \in \mathscr{C}(U \circledast S')$ such that $x (U \circledcirc \mathcal{R}) y$ and $x \overset{X}{\text{—}\!\subset} x'$. This means that $\Pi_S x$ has n concurrent extensions $Y_1, \ldots, Y_n$ leading to $\Pi_S x'$ and $\Pi_U x$ has m concurrent extensions $Z_1, \ldots, Z_m$ leading to $\Pi_U x'$. If one of the Y_i or the Z_i contains an internal event, then X is a singleton and we can conclude easily. The interesting case is when all the Y_i and Z_j are visible.

Since $\Pi_S x \subseteq \Pi_S x'$, we get that $\Pi_{S'} y \subseteq y'$ with $\Pi_{S'} x' \mathcal{R} y'$. By Lemma B.4, we know that y' can be chosen so that the extensions corresponding to $Y_1, \ldots, Y_n$ are also concurrent. Hence there exists a configuration $z \in \mathscr{C}(U \circledast S)$ such that $\Pi_{S'} z = y'$ and $\Pi_U z = \Pi_U x'$.

This proves that $U \circledast S \approx U \circledast S'$. We then conclude by Lemma 5.10. □

Lemma 5.21. *For any submodel* $\mathscr{M}$ *of* $\mathbf{CG}_S$ *and* $S, R \in \mathscr{M}(A)$, *if* $S \approx T$ *then* $S \simeq_{\mathscr{M}} S'$.

Proof. We show that weak bisimilarity is a reduction-closed congruence. By definition of weak bisimilarity and Lemma 5.20, we only have to show that if $S \approx S'$, then they have the same barb, which is again a trivial consequence of the definition. □

Lemma 5.24. *If $\mathscr{M}$ is a testing-closed submodel, then for $S, R \in \mathscr{M}(A)$, we have: $S \approx R$ iff $S \simeq_{\mathscr{M}} T$.*

Proof. The left-to-right implication follows from Lemma 5.21. We show the right-to-left implication.

First, when we have an isomorphism $A \cong B/x$ on two games for $x \in \mathscr{C}(B)$, then we can lift strategies from B to A by iterating the $e \rightarrowtriangle S$ construction. Given $S : A$, we write $x \rightarrowtriangle S$ for the strategy on B obtained by adding all the events in x.

Define a relation

$$\mathcal{R} = \{(x \rightarrowtriangle S, x \rightarrowtriangle R) \mid S, R \in \mathscr{C}(A) \wedge A \cong B/x \wedge S \simeq_{\mathscr{C}} R\}.$$

We show that $\mathcal{R}$ is a weak bisimulation. Consider $x \rightarrowtriangle S \mathcal{R} x \rightarrowtriangle R$.

- If $x \rightarrowtriangle S \xrightarrow{\tau} S'$, then we know that $S' \cong x \rightarrowtriangle S_0$ with $S \xrightarrow{\tau} S_0$ and we conclude because $S \simeq_{\mathscr{C}} R$.
- If $x \rightarrowtriangle S \xrightarrow{e} S'$ with $e \in x$, then $S' \cong (x \setminus \{e\}) \rightarrowtriangle S$, and we conclude since $x \rightarrowtriangle R \xrightarrow{e} (x \setminus \{e\}) \rightarrowtriangle R$.
- If $x = \emptyset$ and $S \xrightarrow{e} S'$, consider a strategy $U : A^{\perp} \parallel \mathrm{test}(A, \alpha)$ in $\mathscr{M}(A, B)$ that defines α. Then we know both that $(U \circledcirc S)\mathcal{R}(U \circledcirc R)$ by definition of reduction-closed congruences, and that $U \circledcirc S \overset{\tau}{\Rightarrow} \mathsf{succ}^+ \cdot S'$.
  As a result, we know that $U \circledcirc R$ evolves to R_0, related by $\mathcal{R}$ to $\mathsf{succ}^+ \cdot S'$. Since $\mathit{barb}(R_0) = \mathit{barb}(U \circledcirc R) = \{\mathsf{succ}^+\}$, it follows that $R_0 \overset{\tau}{\Rightarrow} (\mathsf{succ}^+ \rightarrowtriangle R')$ with $R \overset{\alpha}{\Rightarrow} R'$. Moreover, $\mathcal{R}$ also relates $(\mathsf{succ}^+ \rightarrowtriangle R')$ and $(\mathsf{succ}^+ \rightarrowtriangle S')$ since the latter is deterministic and cannot do any internal transitions. We hence deduce $e \rightarrowtriangle S' \mathcal{R} e \rightarrowtriangle R'$ from the definition of $\mathscr{M}$-congruences.

□

Theorem 5.25 (Semantic full abstraction – $\mathbf{CG}_S$). *The coincident strategy U_α defines the action α for $\mathbf{CG}_S$. As a result, $\mathbf{CG}_S$ is testing-closed, and $S \simeq R$ iff $S \approx R$ for $S, R \in \mathbf{CG}_S(A)$.*

Proof. We prove that the strategies proposed actually define the action. Consider a game A and an action α. There are two cases.

(1) If $\alpha = e$:
  (a) Assume that $S \xrightarrow{e} S'$.
  $$U_e \circledcirc S \cong \mathsf{succ}^+ \rightarrowtriangle S'.$$
  We have
  $$(U_e \circledcirc S) \overset{\tau}{\Rightarrow} (\mathsf{succ}^+ \cdot \mathbb{CC}_{A/e}) \circledcirc S' \cong \mathsf{succ}^+ \rightarrowtriangle S'.$$
  (b) Assume that $U \circledcirc S$ has succ as only barb. this means that $U \circledcirc S \cong \mathsf{succ}^+ \rightarrowtriangle S'$ with $S' : A/e$. It is then straightforward to see that $S/e \cong S'$ as desired.

(2) If $\alpha = \{a^-, b^+\}$:
  (a) Consider $S \xrightarrow{\alpha} S'$. Then we have
  $$(U_e \circledcirc S) \xrightarrow{\tau} ((a \parallel b) \cdot \mathbb{CC}_{A/\alpha}) \circledcirc S$$
  Since $a \parallel b$ can synchronize with a coincidence on $a\,b$, we have that
  $$((a \parallel b) \cdot \mathsf{succ}^+ \cdot \mathbb{CC}_{A/\alpha}) \circledcirc S \overset{\tau}{\Rightarrow} (\mathsf{succ}^+ \cdot \mathbb{CC}_{A/\alpha}) \circledcirc S' \cong \mathsf{succ}^+ \rightarrowtriangle S'.$$
  as desired.

(b) Suppose that $U_\alpha \circledcirc S$ evolves to S_0 and both have only a barb on succ^+. This shows that S must contain a configuration x where a and b occurs. Moreover if a and b were concurrent, or causally comparable, then they would trigger one of the fail branch. This means that there exists a coincidence X whose label is $\{a, b\}$ and $[X)$ contains only neutral event in S hence $S \overset{\alpha}{\Rightarrow} S'$ where S' is such that $S_0 \overset{\tau}{\Rightarrow} \mathsf{succ}^+ \multimap S'$.

□

We now move on to the asynchronous setting. Given a strategy $S : A$ where S is an event structure, we write $\overline{S} : A$ for the asynchronous strategy $\mathbb{CC}_A \circledcirc S : A$. If $S : A$ is asynchronous, we have

$$\overline{L_e} \circledcirc S \cong \mathbb{CC}_{e \cdot A} \circledcirc L_e \circledcirc \mathbb{CC}_A \circledcirc S \cong e \multimap S : e \cdot A.$$

Moreover, we have $L'_e \circledcirc S \cong e \multimap S$ for **asynchronous** $S : A$.

Theorem 5.26 (Semantic full abstraction – $\mathbf{CG}_A$). *The submodel $\mathbf{CG}_A$ is testing-closed, hence for any asynchronous strategies $S, R : A$, we have:*

$$S \simeq_{\mathbf{CG}_S} R \quad \textit{iff} \quad S \approx R \quad \textit{iff} \quad S \simeq_{\mathbf{CG}_A} R.$$

Proof. We already know that the left and middle statements are equivalent. Moreover we also know that $S \approx S'$ implies $S \simeq_{\mathbf{CG}_A} S'$.

First, define $\mathcal{R}_A$ to be the set of pairs of asynchronous strategies (S, S') such that $S \simeq_{\mathbf{CG}_A} S'$. It is easy to see that $\mathcal{R}_A$ is a $\mathbf{CG}_A$-congruence (for the last property by using the test L'_e).

Note that for asynchronous strategies, the only actions of a game A are single events. For an action $e \in \mathit{min}(A)$, consider $U_e = \overline{(e \cdot \mathsf{succ}^+) \cdot \mathbb{CC}_A} : (e \cdot A)^\perp \parallel \mathsf{test}(A, e)$. Using receptivity and courtesy it is easy to show that U_e defines e for $\mathbf{CG}_A$ (but not for $\mathbf{CG}_S$).

Since U_e is asynchronous, this means that $\mathbf{CG}_A$ is $\mathbf{CG}_A$-closed. Since the domain of $\mathcal{R}_A$ is included in $\mathbf{CG}_A(A)$, it follows that $S \simeq_{\mathbf{CG}_A} S' \Rightarrow S \approx S'$ as desired. □

C PROOFS OF SECTION 6

Lemma 6.1. *Consider $\Gamma \vdash P, Q \triangleright \Delta$. If $P \equiv Q$ then $[\![P]\!](\gamma) \cong [\![Q]\!](\gamma)$ for $\gamma \in [\![\Gamma]\!]$.*

Proof. By simple induction on the derivation that $P \equiv Q$. □

Given a process $\vdash P \triangleright \Delta$, we write P_k where fixpoints have been unfolded k times, using a deadlocking process $\perp$ for $k = 0$. As a result, P_k is a finite process (*i.e.* without recursion). Moreover, since $[\![\perp]\!]$ can be embedded in any strategy, we have an embedding $[\![P_k]\!] \hookrightarrow [\![P_{k+1}]\!]$.

Lemma C.1. *For any well-typed process P, $[\![P]\!] = \lim_i([\![P_i]\!])$.*

Proof. It is clear that each of the $[\![P_i]\!]$ embeds into $[\![P]\!]$. As a result, there is an embedding $\lim([\![P_i]\!]) \hookrightarrow [\![P]\!]$. By induction on $[\![P]\!]$, it is easy to see that this embedding must be surjective since any event must occur after a finite number of unfolding. □

Lemma 6.3 (Soundness and adequacy). *Let $\vdash P \triangleright \Delta$ be a well-typed process. Then we have: (1) if $P \to^* Q$, then $[\![P]\!] \overset{\tau}{\Rightarrow} [\![Q]\!]$; and (2) if $[\![P]\!] \overset{\tau}{\Rightarrow} S$ then $S = [\![Q]\!]$ with $P \to^* Q$.*

Proof. (1) Assume that $P \to Q$. We proceed by induction on the derivation of $P \to Q$:

- If $P = Q + R \to Q$, then $[\![P]\!] = (* \cdot [\![Q]\!] + * \cdot [\![R]\!]) \overset{\tau}{\Rightarrow} [\![Q]\!]$.
- If $P = P_0 \parallel R \to Q_0 \parallel R = Q$. From $[\![P_0]\!] \overset{\tau}{\Rightarrow} [\![Q_0]\!]$ we deduce easily $[\![P]\!] \overset{\tau}{\Rightarrow} [\![Q]\!]$.
- Similarly if $P \equiv P' \to Q' \equiv Q$ we conclude by Lemma 6.1.
- If $P = (\nu a)P_0 \to (\nu a)Q_0 = Q$, then from $[\![P_0]\!] \overset{\tau}{\Rightarrow} [\![Q_0]\!]$ we deduce that $\mathbb{CC}_A \circledcirc [\![P_0]\!] \overset{\tau}{\Rightarrow} \mathbb{CC}_A \circledcirc [\![Q_0]\!]$ since initial minimal events will be preserved during composition.

- If $P = (\nu a)(\overline{a}!l_k\langle\widetilde{u}\rangle.\, P' \mid \&_{i\in I} a?l_i(\widetilde{x_i}).\, P_i \to P' \parallel P_k[\widetilde{u}/\widetilde{x_k}] = P$. Write $R_0 = \overline{a}!l_k\langle\widetilde{u}\rangle.\, P'$ and $R_1 = \&_{i\in I} a?l_i(\widetilde{x_i})P_i$. Writing $A = \sum_{i\in I}(l_i, -)\cdot(\llbracket\widetilde{S_i}\rrbracket \parallel \llbracket S\rrbracket)$, we have

$$\llbracket R_0\rrbracket : A^\perp \parallel \llbracket\Delta\rrbracket \qquad \llbracket R_1\rrbracket : A \parallel \llbracket\Delta'\rrbracket$$

  By the law of compact-closed category we have

$$\llbracket P\rrbracket = \mathfrak{CC}_A \circledcirc (\llbracket R_0 \parallel R_1\rrbracket) \cong R_0 \circledcirc_A R_1 : \llbracket\Delta\rrbracket \parallel \llbracket\Delta'\rrbracket.$$

  R_0 has a single minimal event, labelled $(a, +, l_k)$ which will match with the corresponding negative event in R_1. This event is not on $\llbracket\Delta\rrbracket, \llbracket\Delta'\rrbracket$ and will be hidden during composition. Hence:

$$\llbracket P\rrbracket \cong \llbracket P'\rrbracket \parallel \llbracket P_k\rrbracket$$

  We can conclude by noticing that $\llbracket P'_k\rrbracket \cong \llbracket P_k[\widetilde{u}/\widetilde{x}]\rrbracket$ since renaming is invisible in the semantics (it is simply relabelling).

Notice in the proof a crucial thing: if $P \to Q$ due to communication, then $\llbracket P\rrbracket \cong \llbracket Q\rrbracket$.

(2) Assume that $\llbracket P\rrbracket \xrightarrow{\tau} \llbracket P\rrbracket/s$ for a minimal internal event $s \in \llbracket P\rrbracket$. Write P_k for the first finite approximation of P that contains s. Since P_k is recursion-free, we can reduce P_k into P'_k by performing all the possible communications. Since communication are deterministic, we have $\llbracket P_k\rrbracket \cong \llbracket P'_k\rrbracket$. Up to structural congruence, we can assume that P'_k is of the form $(\nu\widetilde{a})(Q_1 \mid \ldots \mid Q_n)$ where Q_i does not start with a restriction or parallel. The event corresponding to s in $\llbracket P'_k\rrbracket$ must correspond to an initial event of Q_i. As a result, Q_i must be of the form $Q + Q'$, and say that s corresponds to the Q branch. Then we have that

$$P_k \to^* P'_k \to (\nu\widetilde{a})(Q_1 \mid \ldots \mid Q \mid \ldots \mid Q_n)$$

from which we deduce a similar derivation for P.

□

Lemma 6.2. *Consider a context $\Delta = a_1 : A_1, \ldots, a_n : A_n$, and $\vdash P \triangleright \Delta, \Delta_P$ and $\vdash Q \triangleright \overline{\Delta}, \Delta_Q$. We have*

$$\llbracket Q\rrbracket \circledcirc_{\llbracket\Delta\rrbracket} \llbracket P\rrbracket \cong \llbracket Q \circledcirc_\Delta P\rrbracket \ (= \llbracket(\nu\widetilde{a})(P \mid Q)\rrbracket).$$

Moreover, for a type T without recursion, we have $\llbracket[u = v]_T\rrbracket = \mathfrak{C}_{\llbracket T\rrbracket}$.

Proof. Consequence of the law of compact closed categories. □

C.1 Full abstraction

Lemma 6.4. *Let $\vdash P \triangleright \Delta$. Then $P \Downarrow_a$ if and only if $(\!|P|\!) \overset{a!l}{\Longrightarrow}$ for some $l \in \mathscr{L}$.*

Proof. Assume that $P \to^* P'$ and $P' \downarrow_a$. Then by definition of the interpretation, $\llbracket P'\rrbracket \xrightarrow{e}$ where $chan(e) = a$. As a result, by soundness, $\llbracket P\rrbracket \overset{\tau}{\Rightarrow} \llbracket P'\rrbracket$, which altogether by soundness implies that $(\!|P|\!) \overset{e}{\Rightarrow}$ since $\overline{a} \notin \Delta$.

Assume that $(\!|P|\!) \overset{e}{\Rightarrow} S$ with $chan(e) = a$. Then there exists a neutral configuration x of $\llbracket P\rrbracket$ such that $x \overset{s}{\text{—}\!\subset}$ with $lbls = e$. By adequacy, this means that $P \to^* Q$ and $\llbracket Q\rrbracket \cong \llbracket P\rrbracket/x$. As a result, $\llbracket Q\rrbracket$ has a minimal positive event on channel a. As for the adequacy proof, it is easy to see that Q can be reduced until it is of the form $(\nu\widetilde{b})(a!l\langle\widetilde{u}\rangle \mid Q')$ which entails the result since $\overline{a} \notin \Delta$. □

Lemma C.2. *The submodel $\mathscr{C}_\pi$ is closed.*

Proof. Suppose that Δ is of the form $u_1 : T_1, \ldots, u_n : T_n$. Consider $e \in min(\llbracket\Delta\rrbracket)$. Then $chan(e)$ is one of the u_i. Assume that e is positive. This implies that T_i is of the form $\oplus_{i\in I} !l_i\langle\widetilde{S_i}\rangle.\, T_i'$. Write l_k for the message of e. We then let:

$$T_e = (\nu\widetilde{b})(u_i!l_k\langle\widetilde{b}\rangle.c!\mathsf{succ}\langle u_1, \ldots, u_i, \widetilde{\widetilde{b}}, u_i, u_{i+1}, \ldots, u_n\rangle)$$

where $\widetilde{b} : \widetilde{T_k}$. It is easy to see that T_e is well-typed on the context:

$$\Delta' = \Delta, c : !\mathsf{succ}(T_1, \ldots, T_{i-1}, \widetilde{\widetilde{S}_k}, T_k', T_{i+1}, \ldots, T_n)$$

A simple observation yields that $\llbracket\Delta'\rrbracket \cong \llbracket\Delta\rrbracket \parallel \mathrm{test}(\llbracket\Delta\rrbracket, e)$ and $(\!|T_e|\!) = U_e$. The case when e is negative is similar, but we replace the the selection on u_i by a branching, and use a deadlocking process for the branches not corresponding to l_k.

Consider now two events e^- and e'^+ in $min(\llbracket\Delta\rrbracket)$. As before, assume that e corresponds to channel u_i and e' to channel u_j with labels l and l' respectively. Then we let

$$\begin{aligned} T_{\{e,e'\}} =& (\nu\widetilde{w}d)(u_j!l\langle\widetilde{w}\rangle.\overline{d}! \parallel u_i?l(\widetilde{v}).d?c_{\mathsf{succ}}!\mathsf{succ}\langle\widetilde{u}, \widetilde{v}, \widetilde{\widetilde{w}}\rangle \\ &+(u_i?l(\widetilde{v}).\, u_j!(\widetilde{w}).\, c_{\mathtt{fail}}!\mathtt{fail}) \\ &+(u_j!(\widetilde{w}).\, u_i?l(\widetilde{v}).\, c_{\mathtt{fail}}!\mathtt{fail}) \end{aligned}$$

Similarly, we see that $T_{\{e,e'\}}$ lives on a context whose interpretation is $\llbracket\Delta\rrbracket \parallel \mathrm{test}(\llbracket\Delta\rrbracket, \{e, e'\})$, and that $\llbracket T_{\{e,e'\}}\rrbracket \cong U_{\{e,e'\}}$.

This shows that C_π is closed. □

Theorem 6.5. *$\mathscr{C}_\pi$ is testing-closed. As a result, for $\vdash P, Q \triangleright \Delta$, $P \simeq Q$ if and only if $(\!|P|\!) \simeq (\!|Q|\!)$.*

Proof. *Right-to-left.* From Lemma 6.3 and 6.4, the relation $(\!|P|\!) \simeq (\!|Q|\!)$ is a reduction-closed congruence which entails the result.

Left-to-right. Because $\mathscr{C}_\pi$ is testing-closed, we get:

$$(\!|P|\!) \simeq (\!|Q|\!) \qquad \text{iff} \qquad (\!|P|\!) \simeq_{\mathscr{C}_\pi} (\!|Q|\!).$$

We then observe that the relation $\mathcal{R} = \{((\!|P|\!), (\!|P'|\!)) \mid P \simeq P'\}$ is a $\mathscr{C}_\pi$-congruence by Lemma 6.3 and 6.4. As a result: $P \simeq P' \Rightarrow (\!|P|\!) \simeq_{\mathscr{C}_\pi} (\!|P'|\!) \Rightarrow (\!|P|\!) \simeq (\!|P'|\!)$. □

C.2 Finite definability of internal strategies

Lemma C.3. *Let $S : A^\perp \parallel B$ and $T : B^\perp \parallel C$ be confusion-free. Then $T \circledcirc S$ is confusion-free.*

Proof. We first show the interaction $T \circledast S$ is confusion-free.

(1) If $e \frown_{T\circledast S} e' \frown_{T\circledast S} e''$, then we have *e.g.* $\Pi_S e \frown_S \Pi_S e'$. This is either because $\Pi_S e$ and $\Pi_S e'$ are internal, in which case $\Pi_S e''$ must also be internal and $\Pi_S e' \frown_S \Pi_S e''$, or because they are negative, and in which case we have $\Pi_S e' \frown_S \Pi_S e''$ as well. In both cases, we can conclude by confusion-freeness of S.

(2) If $e \frown_{T\circledast S} e'$, with say $\Pi_S e \frown_S \Pi_S e'$, then we have $[\Pi_S e) = [\Pi_S e')$. If e and e' are internal, then we conclude that $[e) = [e')$ since $[e) \cup [e')$ is a configuration; and if they are negative, then they must be sent to A which also entails $[e) = [e')$ as desired.

From this we deduce that $T \circledcirc S$ is also confusion-free since $e \frown_{T\circledcirc S} e'$ implies $e \frown_{T\circledast S} e'$ as events in minimal conflicts are never hidden. □

Lemma 6.7. *Let $\vdash P \triangleright \Delta$ be an internal session process. Then $\llbracket P\rrbracket$ is an internal strategy.*

Proof. By the previous lemma, it is easy to see that $\llbracket P\rrbracket$ is confusion-free and exhaustive-matching. We focus on the fact that $\llbracket P\rrbracket$ has no-nontrivial coincidences. There are two cases where the coincident copycat appears:

- If $P = (\nu a)Q$. We know that $[\![P]\!] = \mathfrak{C}_{[\![\Delta]\!]} \circledcirc [\![Q]\!]$. Consider a non-trivial coincidence $\{s^-, s'^+\}$ coincidence of $[\![P]\!]$. In $\mathfrak{C}_A \circledast [\![Q]\!]$, this must correspond to a coincidence $X = \{s, s', e_1, \ldots, e_n\}$ and the graph $(X, \Pi_{[\![Q]\!]}, \Pi_{\mathfrak{C}_{[\![\Delta]\!]}})$ must be connected. Since $lbl(s)$ is not in $[\![\Delta]\!]$, and s must be connected to another element s_0 of X, it must be that $\Pi_{[\![Q]\!]} s$ and $\Pi_{[\![Q]\!]} s_0$ are coincident in $[\![Q]\!]$, absurd by induction hypothesis.
- When $P = (\nu(\widetilde{b} : \widetilde{B}))a!l\langle\widetilde{b}\rangle.Q(\widetilde{\widetilde{b}})$. In this case, we have

$$\begin{aligned}[\![P]\!] &= \mathfrak{C}_{[\![\widetilde{B}]\!]} \circledcirc ((a, l, +) \cdot (\mathfrak{C}_{[\![\widetilde{B}]\!]} \parallel [\![Q]\!]) \\ &= (a, l, +) \circledcirc (\mathfrak{C}_{[\![\widetilde{B}]\!]} \circledcirc (\mathfrak{C}_{[\![\widetilde{B}]\!]} \parallel [\![Q]\!]) \\ &= (a, l, +) \circledcirc ([\![Q]\!] \circledcirc \mathfrak{C}_{[\![\widetilde{B}]\!]}) \\ &= (a, l, +) \circledcirc [\![Q]\!]\end{aligned}$$

  which is coincident-free.

□

Lemma 6.8. *Let $S : [\![\Delta]\!]$ be a* finite *internal strategy such that S is a forest. Then there exists an internal process P such that $[\![P]\!] = S$.*

Proof. We proceed simply by induction on S. If S is empty but $[\![\Delta]\!]$ is not, we define S by

$$P = (\nu ab)(\overline{a}!*.\overline{b}!* \mid a?* . b?* . P_\Delta)$$

where P_Δ is a process exploring the type Δ, easily defined by induction on Δ. Because of the deadlock $[\![P]\!]$ is indeed empty.

Otherwise, since S is a forest, $S = S_1 \parallel \ldots \parallel S_n$ where S_i has a unique minimal cell $\mathfrak{c}_i$. Then Δ must split into $\Delta_1, \ldots, \Delta_n$ so that S_i is an internal strategy on $[\![\Delta_i]\!]$ which is a forest. Then by induction, $S_i = [\![P_i]\!]$ for some $\vdash P_i \triangleright \Delta_i$ and

$$[\![P_1 \parallel \ldots \parallel P_n]\!] \cong [\![S]\!].$$

If S has a unique minimal cell $\mathfrak{c} = \{s_1, \ldots, s_n\}$. Write $S_j = S/[s_j]$. If $\mathfrak{c}$ is an internal cell, then S_j is an internal strategy on $[\![\Delta]\!]$ which is a forest, hence defined by some P_j. Then S is defined by $P_1 + \ldots + P_n$.

If $\mathfrak{c}_i$ is a positive cell, then Δ must be of the form $\Delta_0, a : \oplus!l_i\langle\widetilde{T_i}\rangle S_i$, with $\mathfrak{c}$ a singleton with channel a, and message l_k. Write Δ' for $\Delta_0, a : S_k, \widetilde{b} : \widetilde{T_k}$ where the names $\widetilde{b}$ do not occur in Δ_0. Then S_1 is an internal strategy on $[\![\Delta']\!]$ hence is defined by P'. Therefore, S is defined by

$$P = a!l_k(\widetilde{b}).P'$$

If $\mathfrak{c}_i$ is a negative cell, then Δ must be of the form $\Delta_0, a : \&_{1 \le i \le n} ?l_i(\widetilde{T_i}).S_i$. Moreover, because S is exhaustive matching, $\mathfrak{c}$ must be of the form $\{s_1, \ldots, s_n\}$. Then as for the positive case, each $S/[s_i]$ can be defined by a process P_i with extra free variables $\widetilde{x_i}$ and S is defined by $\&_{1 \le i \le n} a?l_i(\widetilde{x_i}).P_i$. □

Lemma 6.9. *Let $S : [\![\Delta]\!]$ be an internal strategy. If S has causal complexity > 0, then there exists $S' : [\![a : \bot, b : 1]\!]^\bot \parallel A$ such that $\mathrm{cc}(S') < \mathrm{cc}(S)$ and $S \cong S' \circledcirc_{[\![a:\bot, b:1]\!]} \mathfrak{C}_{[\![1]\!]}$.*

Proof. Since S has nonzero causal complexity, there exists $s \in S$ with nonzero causal complexity. As a result, it has at least two predecessors. Because the game is a forest, there must exist at least one $s' \rightarrowtriangle s$ such that $lbl(s')$ and $lbl(s)$ are concurrent.

We define S' on the game $[\![a : \bot, b : 1]\!]^\bot \parallel A$ as follows. Its events are $S \cup \{s_1, s_2\}$. Conflict is that of S and labelling is that of S plus $lbl(s_1) = a!()$ and $lbl(s_2) = b?()$. Causality is defined as

$$(\le_S \setminus \{(s', s)\} \cup \{(s_0, s_2) \mid s_0 < s \wedge s_0 \neq s'\} \cup \{(s', s_1)\}.$$

It is easy to see that S' is indeed a coincident strategy. The fact that lbl_S is a map of event structure relies on $lbl_S s'$ and $lbl_S s$ being concurrent. Now let us compute the join complexity of S':

- The join arity of s_1 is 1, so its join complexity is zero.
- The join complexity of s_1 is $|[t_1]|+\ldots|[t_k]|+|[r_1]|+\ldots+|[r_m]|$ where the t_i are the predecessors of s apart from s', and the r_i are the predecessors of s'. Since $|[s']| > |[r_1]| + \ldots |[r_n]|$, we conclude that $\mathrm{cc}(s_1) < \mathrm{cc}(s)$.
- In S', s has a unique predecessor which is s_2. Hence its join complexity is zero.
- Other events of S' have the same join complexity as in S.

Hence $\mathrm{cc}(S') < \mathrm{cc}(S)$ as desired.

Then, in the interaction $S' \circledast \mathbb{C}_1$, as an event structure, is the same as S' with the extra causal link $s_1 \leq s_2$ added by copycat. When we hide, it is easy to see that we recover the missing link $s' \leq s$ through the chain $s' \rightarrowtriangle s_2 \rightarrowtriangle s_1 \rightarrowtriangle s$. Hence $S' \circledcirc \mathbb{C}_1 \cong S$.

□

Theorem 6.10. *If $S : [\![\Delta]\!]$ is finite internal, there exists an internal process P with $[\![P]\!] \cong S$.*

Proof. We iterate Lemma 6.9, using the fact that composition is definable (Lemma 6.2). □

Lemma C.4. *Let $S : A$ be a coincident strategy with satisfies the uniqueness condition of receptivity. If $\mathbb{C}_A \circledcirc S \approx S$ then S is an asynchronous strategy.*

Proof. First, $\mathbb{C}_A \circledcirc S$ is a coincidence-free event structure since $\mathbb{C}_A$ has no non-trivial coincidences. As a result S is also a coincidence-free event structure. We show it is a strategy:

- *Receptivity.* We already know the uniqueness. Let $x \in \mathscr{C}(S)$ such that $lbl(x)$ extends by a negative $e \in A$. There exists $y \in \mathscr{C}(\mathbb{C}_A \circledcirc S)$ such that $S/x \approx (\mathbb{C}_A \circledcirc S)/y$. Because $\mathbb{C}_A \circledcirc S$ is receptive, y can extend by an event mapped to e, forcing x to be able to extend by an event mapped to e.
- *Courtesy.* Let $e \rightarrowtriangle e'$ such that $lbl_S(e)$ and $lbl_S(e')$ are concurrent. Consider a configuration $y \in \mathscr{C}(\mathbb{C}_A \circledcirc S)$ weakly bisimilar to $[e)$. Assume that e is positive or e' is negative. Then playing the bisimulation game, y can extend by e_0, and then by e_0' matching e and e' respectively. Since $[e)$ cannot do $lbl_S(e')$ right away, it must be that $e_0 \rightarrowtriangle e_0'$. By courtesy of $\mathbb{C}_A \circledcirc S$, it follows that e_0 must be negative and e_0' positive, as desired.

□

Lemma 6.6. *Consider a context $\Delta = a_1 : A_1, \ldots, a_n : A_n$ which does not contain a name and its coname. For $\vdash P \triangleright \Delta$, $[\![P]\!]$ is an asynchronous strategy if and only if $(\nu \widetilde{a})(P \mid [\widetilde{a} = \widetilde{b}]) \simeq P\{\widetilde{b}/\widetilde{a}\}$.*

Proof. Consequence of Lemma C.4, since strategies in the interpretation obviously satisfy the uniqueness of receptivity. □

D PROOFS OF SECTION 7

D.1 Encoding of strategies

Lemma D.1. *$\overline{S}$ is an event structure.*

Proof. Clearly $\leq_{\uparrow S}$ is a preorder and $\uparrow S$ is a coincident event structure. We show that $\leq_{\uparrow S}$ is antisymmetric. Consider a cycle

$$(s_1, \alpha_1) \prec \ldots \prec (s_n, \alpha_n) \prec (s_1, \alpha_1)$$

in $\uparrow S$. The function $\pi_1 : \overline{S} \to S$ can easily be seen to be monotonic. As a result, we have:

$$s_1 \leq_S \ldots \leq_S s_n \leq_S s_1$$

ie. the s_i are coincident. This implies that $n = 2$ and s_1 and s_2 are visible and have opposite polarities. Say that s_1 is negative and s_2 positive. From $(s_1, \alpha_1) \prec (s_2, \alpha_2)$, since s_1 and s_2 are coincident, we get that $\alpha_1 = \alpha_2 = \mathsf{r}$, and from $(s_2, \alpha_2) \prec (s_1, \alpha_1)$ we get $\alpha_1 = \alpha_2 = \mathsf{a}$: absurd. □

Lemma 7.2. *$\overline{S}$ is a coincidence-free strategy on $\uparrow A$.*

Proof. Since $\overline{S}$ is an event structure, we need only to show that it is secret.

Consider $x \in \mathscr{C}(\overline{S})$ with two incompatible extensions (s_1, α_1) and (s_2, α_2). By definition of consistency in $\overline{S}$ we must have $\alpha_1, \alpha_2 \in \{\mathsf{r}, *\}$. Write $y = [\pi_1 x] \in \mathscr{C}(S)$. Then since (s_1, α_1) and (s_2, α_2) are incompatible extensions of x, y cannot contain s_1 or s_2: this would mean, eg. s_1 is coincident to some $s_1' \in \pi_1 x$ and this s_1' would also be in conflict with s_2, absurd. Write X_1 and X_2 for the coincidences of s_1 and s_2. We have $y \overset{X_1}{\frown} \subset$ and $y \overset{X_2}{\frown} \subset$. By assumption they must be incompatible hence we have by secrecy of S:

- Either $X_1 = \{s_1\}$ and $X_2 = \{s_2\}$ with s_1, s_2 neutral and in a minimal conflict at y. Then we can conclude.
- Either there exists $s_1' \in X_1$ and $s_2' \in X_2$ negative, whose image in the game is in conflict. Then we must have $s_1 = s_1'$ and $s_2 = s_2'$ since in a coincidence, negative requests are performed first. Hence we can conclude.

□

D.2 Characterisation of the configurations of $\uparrow S$

For the proofs of the next subsection, we will need to have a tight grasp of the configurations of $\uparrow S$. First, notice that any configuration y of $\uparrow A$ splits into $\mathfrak{r}(y) = \{a \mid (a, \mathsf{r}) \in y\}$ and $\mathfrak{a}(y) = \{a \mid (a, \mathsf{a}) \in y\}$. A configuration of $y \in \uparrow A$ is **complete** when $\mathfrak{r}(y) = \mathfrak{a}(y)$. There is an obvious order-isomorphism:

$$\varphi : \mathscr{C}(A) \cong \mathscr{C}_c(\uparrow A)$$

where $\mathscr{C}_c(\uparrow A)$ are the complete configurations of $\uparrow A$.

To characterise the structure of configurations of $\uparrow S$, we need to first characterise configurations of $\mathbb{C}_A \circledcirc S$ when S is a coincidence-free strategy on A. We write $\sqsubseteq_A$ for the Scott order on a game A: $x \sqsubseteq_A y$ when $x \supseteq^- \subseteq^+ y$. Say that $x \in \mathscr{C}(S)$ is ***fully propagated*** for $y \in \mathscr{C}(A)$ if $y \sqsubseteq lbl_S(x)$, and for any $s \in max(x)$ which is visible but not involved in a minimal conflict, then $lbl_S(s) \in y^+$.

Lemma D.2. *Let $S : A$ be a coincidence-free strategy. There is an order isomorphism:*

$$\mathscr{C}(\mathbb{C}_A \circledcirc S) \cong \{(x, y) \mid x \text{ is fully propagated for } y\}.$$

Proof. First, because interaction with copycat is deadlock free, we have an order-isomorphism:

$$\mathscr{C}(\mathbb{C}_A \circledast S) = \{(x, y) \mid x \in \mathscr{C}(S) \;\&\; y \sqsubseteq lbl_S(x)\}.$$

From $\mathscr{C}(\mathbb{C}_A \circledcirc S)$. Consider $w \in \mathscr{C}(\mathbb{C}_A \circledcirc S)$. Its downclosure $[w]$ yields a configuration of $\mathbb{C}_A \circledast S$, corresponding to a pair $(x, y) \in \mathscr{C}(S) \times \mathscr{C}(A)$ with $y \sqsubseteq lbl_S(x)$. Consider an element $s \in max(x)$ visible, but not involved in a minimal conflict. The configuration $[w]$ corresponds to a secured bijection $\varphi : x \parallel lbl_S(y) \simeq x_* \parallel lbl_S(x) \parallel y$, and events of $[w]$ are in bijection with elements in the graph of φ. So s corresponds to an element $((0, s), (1, lbl_S(s))) \in \varphi$. This element cannot be maximal in $[w]$ since $[w]$ only contains maximal element $((0, s), (1, lbl_S(s)))$ where s is involved in a minimal conflict; or $((1, a), (2, a))$ with $a \in y$ which is not the case by assumption. Hence this element is not maximal in $[w]$, and since s is maximal in x, this implies that s must be positive and $lbl_S(s)$ must belong to y.

This map is also clearly monotonic and injective as simply being a restriction of the aforementionned isomorphism on configurations of the interaction.

To $\mathscr{C}(\mathbb{C}_A \circledast S)$. Given a pair (x, y) we can form the configuration of the interaction $z \in \mathscr{C}(\mathbb{C}_A \circledast S)$ corresponding to it. Consider a maximal event $\alpha \in z$. There are three possibilities:

- If $\alpha = ((1, a), (2, a))$ then α is indeed not hidden during composition
- If $\alpha = ((0, s), (0, s))$ then s is an internal event of S so it must be involved in a minimal conflict, and so must α hence it is also not hidden.
- Finally, if $\alpha = ((0, s), (1, \mathit{lbl}_S(s)))$ then we know that s is visible, must also be maximal (since α) is. If it is involved in a minimal conflict, it is not hidden during composition; and if it is not involved in a minimal conflict, then $\mathit{lbl}_S(s) \in y^+$ which contradicts minimality of α since

$$\alpha \rightarrowtriangle ((1, \mathit{lbl}_S(s)), (2, \mathit{lbl}_S(s))).$$

Hence $z \cap \mathbb{C}_A \circledcirc S$ is the desired configuration.

It is easy to show that those two maps are inverse of each other. □

In the case of $\overline{S}$, since it is secret, $x \in \mathscr{C}(\overline{S})$ is fully-propagated for $y \in \mathscr{C}(\uparrow A)$ when $y \sqsubseteq \mathit{lbl}_S(x)$ and $\mathit{lbl}_S(\mathit{max}(x)) \subseteq y^+$.

To understand the structure of $\uparrow S$, we remark this:

Lemma D.3. *Let z be a configuration of $\uparrow S$ corresponding to (x, y) via Lemma D.2. Then for any visible $s \in \pi x$, $(s, \mathsf{a}) \in x$ if and only one of the conditions is satisfied:*

- *s is not maximal in $[\pi x]$*
- *$(\mathit{lbl}_S(s), \mathsf{a}) \in y$ and if s is positive, then $s \equiv s'$ with $(\mathit{lbl}_S(s'), \mathsf{a}) \in y$.*

Proof. Consider $(s, \mathsf{a}) \in x$ such that $s \in \pi x$ is maximal. Note that (s, a) is not involved in a minimal conflict as it is an acknowledgement.

- If (s, a) is maximal in s, because x is fully propagated in y, $(\mathit{lbl}_S(s), \mathsf{a})$ should be a positive event of y, implying that s is negative.
- If (s, a) is not maximal, this means $(s, \mathsf{a}) < (s', \mathsf{a}) \in x$ for $s^+ \equiv s'^-$ and $(\mathit{lbl}_S(s'), \mathsf{a}) \in y$ because (s', a) is maximal in x.

□

This means that the mapping

$$\begin{array}{rcccc} \chi : \mathscr{C}(\uparrow S) \to & \mathscr{C}(S) \times & \mathscr{C}(A) \times & \mathscr{C}(A) \\ (x, y) \mapsto & (\pi\, x, & \mathfrak{r}(y), & \mathfrak{a}(y)) \end{array}$$

is injective. We now spell out a list of axioms on these triples to characterise the image of χ. A triple $(x, y_\mathsf{r}, y_\mathsf{a}) \in \mathscr{C}(S) \times \mathscr{C}(A) \times \mathscr{C}(A)$ is a ***lifted configuration*** of S when:

(1) The set $\mathit{lbl}_S(x) := y_\mathsf{a} \times \{\mathsf{a}\} \cup y_\mathsf{r} \times \{\mathsf{r}\}$ is a configuration of $\uparrow A$.
(2) $y_\mathsf{a} \subseteq \mathit{lbl}_S(x) \subseteq^- y_\mathsf{r}$
(3) If $s^- \in \mathit{max}(x)$, and $\mathit{lbl}_S(s) \notin y_\mathsf{a}$ then s is not separated.
(4) If $s^+ \in [x)$, then $\mathit{lbl}_S(s) \in y_\mathsf{a}$.
(5) If $\{s^+, s'^-\}$ is a coincidence of x, then $\mathit{lbl}_S(s') \in y_\mathsf{a}$ implies $\mathit{lbl}_S(s) \in y_\mathsf{a}$.

Lemma D.4. *For (x_0, y_0) corresponding to a configuration of $\uparrow S$, the triple $\chi(x_0, y_0)$ is a lifted configuration.*

Proof. Write $\chi(x_0, y_0) = (x, y_\mathsf{r}, y_\mathsf{a})$. We verify the axioms:

(1) Trivial by definition of y_r and y_a
(2) We prove the two inclusions and the polarity condition:
   - Let $a \in y_\mathsf{a}$. Among (a, r) or (a, a), one of them is positive. Since both belong to $y_0 \sqsubseteq \mathit{lbl}_{\overline{S}}(x_0)$, then we have at least $(a, \mathsf{r}) \in \mathit{lbl}_{\overline{S}}(x_0)$. This implies that $a \in \mathit{lbl}_S(\pi x_0)$ as desired.

- Let $lbl_S\ s \in lbl_S\ x$. We have $(s, \mathsf{r}) \in x_0$. If s is negative, then from $y_0 \sqsubseteq lbl_S(x_0)$, we conclude that $(lbl_S\ s, \mathsf{r}) \in y_0$ hence $lbl_S(s) \in y_\mathsf{r}$. If s is positive, then either $(s, \mathsf{a}) \in x_0$, which since it is negative implies $lbl_S(s) \in y_\mathsf{a} \subseteq y_\mathsf{r}$ as desired; or $(s, \mathsf{a}) \in x_0$ meaning that (s, r) is maximal in x_0 (because s is positive). Since x_0 is fully propagated with y_0, it follows that $(lbl_S(s), \mathsf{r}) \in y_0$ which implies the desired conclusion.
- Let $a \in y_\mathsf{r} \setminus lbl_S(x)$. Then $(a, \mathsf{r}) \in y_\mathsf{r} \setminus lbl_{\overline{S}}\ x_0$ hence it must be negative.

(3) Consider $s^- \in max(x)$ such that $lbl_S(s) \notin y_\mathsf{a}$. Since (s, r) is negative, it cannot be maximal. If $(s, \mathsf{a}) \in x_0$, then it will be maximal but its image will not be in y_0: absurd. Then $(s, \mathsf{a}) \notin x_0$ but (s, r) is not maximal. This means that s must be coincident to s' and $(s, \mathsf{r}) < (s', \mathsf{r}) \in x_0$. In particular s' is not separated.

(4) Consider $s^+ \in x$ not maximal. This means that $s < s'$ so that $(s, \mathsf{a}) < (s', \mathsf{r})$ in $\overline{S}$. We get that $(s, \mathsf{a}) \in x_0$ is negative, hence must belong to y_0 which implies $s \in y_\mathsf{a}$.

(5) Consider a coincidence $\{s^+, s'^-\}$ such that $lbl_S(s') \in y_\mathsf{a}$. As a result $(s, \mathsf{a}) < (s', \mathsf{a}) \in x$. Since (s, a) is negative, we get that $(lbl_S(s), \mathsf{a}) \in y_\mathsf{a}$.

□

Given $\mathfrak{x}$ a lifted configuration, we define the ***internal acknowledgements***, and its ***internal state*** as follows:

$$\mathsf{a}(\mathfrak{x}) = x_v \setminus max(x) \cup \{s \in max(x) \cap S^{-1}(y_\mathsf{a}) \mid s \text{ positive} \Rightarrow s \equiv s' \wedge lbl_S(s') \in y_\mathsf{a}\}$$
$$\mathsf{st}(\mathfrak{x}) = x_* \times \{*\} \cup x_v \times \{\mathsf{r}\} \cup \mathsf{a}(\mathfrak{x}) \times \{\mathsf{a}\}.$$

We have:

Lemma D.5. *$\mathsf{st}(\mathfrak{x})$ is a configuration of $\overline{S}$ and is fully propagated for $lbl_S(\mathfrak{x})$. Moreover, $\chi(\mathsf{st}(\mathfrak{x}), lbl_S(\mathfrak{x})) = \mathfrak{x}$.*

Proof. *It is a configuration of $\overline{S}$.* We have $\pi(\mathsf{st}(\mathfrak{x})) = x \in \mathrm{Con}_S$ hence by definition $\mathsf{st}(\mathfrak{x})$ is consistent in $\overline{S}$. We show downclosure, consider the possible cases for $e \prec e' \in x$:

- $e = (s, \mathsf{r}) \prec e' = (s, \mathsf{a})$: trivial since $\mathsf{a}(\mathfrak{x}) \subseteq x$
- $e = (s, \mathsf{a}) \prec e' = (s', \mathsf{r})$: we know that $s < s'$ and $\{s, s'\}$ is not a coincidence. As a result $s \in x_v \setminus max(x) \subseteq \mathsf{a}(\mathfrak{x})$.
- $e = (s, \mathsf{r}) \prec e' = (s', \mathsf{r})$ when $\{s^-, s'^+\}$ is a coincidence: trivial since $s \in x$.
- $e = (s', \mathsf{a}) \prec e' = (s, \mathsf{a})$ when $\{s^-, s'^+\}$ is a coincidence: we know that $s \in \mathsf{a}(\mathfrak{x})$. If s is not maximal in x, neither is s' and we can conclude. Otherwise, we know that $lbl_S(s) \in y_\mathsf{a}$ hence via axiom (5), $lbl_S(s') \in y_\mathsf{a}$ as well, which shows that $a \in \mathsf{a}(\mathfrak{x})$ as desired.
- $e = (s, \mathsf{a}) \prec e' = (s', *)$: since $s < s'$, we have that s is not maximal in x.
- $e = (s, *) \prec e' = (s', \mathsf{r})$: trivial since $s < s'$ hence $s \in x_*$.
- $e = (s, *) \prec e' = (s', *)$ trivial since $s < s'$ in x_*.

Full-propagation. Now we show that $lbl_S(\mathfrak{x}) \sqsubseteq lbl_S(\mathsf{st}(\mathfrak{x}))$:

- If $(a^+, \mathsf{r}) \in lbl_S(\mathfrak{x})$: then $a \in y_\mathsf{r} \supseteq^- lbl_S(x)$ by (2), hence $a \in lbl_S(x)$ which implies $(a, \mathsf{r}) \in S(\mathsf{st}(\mathfrak{x}))$.
- If $(a^-, \mathsf{a}) \in lbl_S(\mathfrak{x})$: then since $y_\mathsf{a} \subseteq lbl_S(x)$ by (2), we have that $a = lbl_S(s)$ with negative $s \in x$. It is easy to see that $s \in \mathsf{a}(\mathfrak{x})$, so $(s, \mathsf{a}) \in \mathsf{st}(\mathfrak{x})$.
- If $(S\, s^-, \mathsf{r}) \in lbl_S(\mathsf{st}(\mathfrak{x}))$, then $s \in x$ hence $lbl_S(s) \in y_\mathsf{r}$ by (2) and we can conclude.
- If $(S\, s^+, \mathsf{a}) \in lbl_S(\mathsf{st}(\mathfrak{x}))$, then $s \in \mathsf{a}(\mathfrak{x})$. If s is maximal in x, we are done by definition of $\mathsf{a}(\mathfrak{x})$. If it is not maximal, we conclude by axiom (4).

Consider $e \in max(\mathsf{st}(\mathfrak{x}))$ which is visible. There are two cases:

- $e = (s, \mathsf{r})$: then s is maximal and visible. First, if s is negative, we know that $s \notin \mathsf{a}(\mathfrak{x})$, hence $s \notin S^{-1}(y_{\mathsf{a}})$. This contradicts axiom (3). As a result, s must be positive. Since $lbl_S(s) \in y_{\mathsf{r}}$, we get that $\overline{S}\, e \in y_0$.
- $e = (s, \mathsf{a})$: by definition $s \in lbl_S^{-1}(y_{\mathsf{a}})$. We need to show that s is negative (so that e is positive). We know that s is not a separated positive event. If it were a positive event coincident with negative $s' \in x$, then we know that $lbl_S(s') \in y_{\mathsf{a}}$ by definition of $\mathsf{a}(\mathfrak{x})$.

$\chi(\mathsf{st}(\mathfrak{x}), S(\mathfrak{x})) = \mathfrak{x}$. It is easy to see that they agree on components y_{a} and y_{r}. Write $x^\dagger$ for the x component of $\chi(\mathsf{st}(\mathfrak{x}), S(\mathfrak{x}))$. We have, by definition

$$\begin{aligned} x^\dagger &= \{s \mid (s, \mathsf{r}) \in \mathsf{st}(\mathfrak{x}) \vee (s, *) \in \mathsf{st}(\mathfrak{x})\} \\ &= x_{\upsilon} \cup x_* = x \end{aligned}$$

□

We can now conclude:

Proposition D.6. *Lifted configurations of S are order-isomorphic to configurations of $\uparrow S$.*

Proof. Via Lemma D.2, χ defines an injective monotonic map from $\mathscr{C}(\uparrow S)$ to lifted configurations of S. Lemma D.5 shows it is surjective. We conclude since χ trivially preserves and reflects inclusion. □

D.3 Injectivity and functoriality

Injectivity of the encoding. We start by showing that the configurations of S can be recovered inside $\uparrow S$ by looking at complete configurations.

Lemma D.7. *Let $\mathfrak{x} = (x, y_{\mathsf{r}}, y_{\mathsf{a}})$ be a lifted configuration of S. The configuration $lbl_S(\mathfrak{x})$ is complete if and only if $y_{\mathsf{a}} = y_{\mathsf{r}} = lbl_S(x)$.*

Proof. Assume that $lbl_S(\mathfrak{x})$ is complete. Clearly $y_{\mathsf{a}} = y_{\mathsf{r}}$. Since $y_{\mathsf{a}} \subseteq lbl_S\, x \subseteq y_{\mathsf{r}}$, it follows that $y_{\mathsf{a}} = x$ as desired.
The converse implication is clear. □

As a result the map $\varphi_S : \mathscr{C}(S) \to \mathscr{C}_c(\uparrow S)$ mapping a configuration $x \in \mathscr{C}(S)$ to the configuration of $\uparrow S$ corresponding to the lifted configuration $(x, lbl_S(x), lbl_S(x))$ is an order-isomorphism. From this, the injectivity of $S \mapsto \uparrow S$ follows:

Lemma D.8. *For S, R two coincident strategies on A, if $\uparrow S \cong \uparrow R$, then $S \cong R$.*

Proof. Clearly if $S \cong T$ then $\uparrow S \cong \uparrow T$. Conversely, from $\uparrow S \cong \uparrow T$, we deduce easily that $\mathscr{C}_c(\uparrow S) \cong \mathscr{C}_c(\uparrow T)$, hence:

$$\mathscr{C}(S) \cong \mathscr{C}_c(\uparrow S) \cong \mathscr{C}_c(\uparrow T) \cong \mathscr{C}(T).$$

□

Preservation of copycat. We now establish that the encoding sends the synchronous copycat to the asynchronous copycat. Given $y_{\mathsf{r}}, y_{\mathsf{a}} \in \mathscr{C}(A)$ we write $y_{\mathsf{r}} \uplus y_{\mathsf{a}}$ for $y_{\mathsf{r}} \times \{\mathsf{r}\} \cup y_{\mathsf{a}} \times \{\mathsf{a}\} \subseteq \uparrow A$.

Lemma D.9. *If $(x \parallel x, y_{\mathsf{r}} \parallel y'_{\mathsf{r}}, y_{\mathsf{a}} \parallel y'_{\mathsf{a}})$ is a lifted configuration of $\mathfrak{CC}_A$. We have:*

$$y_{\mathsf{r}} \sqsubseteq_A y'_{\mathsf{r}} \qquad y'_{\mathsf{a}} \sqsubseteq y_{\mathsf{a}}.$$

Moreover $x = y_{\mathsf{r}} \cap y'_{\mathsf{r}}$.

Proof. We start with the Scott inclusions.

- ($y_{\mathsf{r}} \sqsubseteq_A y'_{\mathsf{r}}$) Consider $a^+ \in y_{\mathsf{r}}$. Then, we know $a \in x \subseteq y'_{\mathsf{r}}$ by (2). The other inclusion is symmetric.
- ($y'_{\mathsf{a}} \sqsubseteq y_{\mathsf{a}}$) If $a^+ \in y'_{\mathsf{a}}$, then $a \in x$. If $a \notin max(x)$, then $a \in y'_{\mathsf{a}}$ by (4). Otherwise, if $a \in max(x)$, then we know that $(0, a) \equiv (1, a)$ in $\mathbb{CC}_A$, and since a is positive, $a \in y'_{\mathsf{a}} \Rightarrow a \in y_{\mathsf{a}}$ which allows us to conclude by assumption.

By applying (2), we get that $x \parallel x \subseteq^- y_{\mathsf{r}} \parallel y'_{\mathsf{r}}$. This implies that $x \subseteq y_{\mathsf{r}} \cap y'_{\mathsf{r}}$. Moreover, if $a \in y_{\mathsf{r}} \cap y'_{\mathsf{r}}$, then one of $(0, a)$ or $(1, a)$ is positive in $A^\perp \parallel A$, hence $a \in x$. □

Lemma D.10. *Let A be game. We have $\uparrow\mathbb{CC}_A \cong \mathbb{C}_{\uparrow A}$.*

Proof. The previous lemma tells us that the mapping:

$$\begin{aligned} \psi : \mathscr{C}(\uparrow\mathbb{CC}_A) &\longrightarrow & \mathscr{C}(\mathbb{C}_{\downarrow A}) \\ \chi(\_, y_{\mathsf{r}} \parallel y'_{\mathsf{r}}, y_{\mathsf{a}} \parallel y'_{\mathsf{a}}) &\mapsto & (y_{\mathsf{r}} \uplus y_{\mathsf{a}}) \parallel (y'_{\mathsf{r}} \uplus y'_{\mathsf{a}}) \end{aligned}$$

is an order-isomorphism, which allows us to conclude by Lemma B.2. □

D.4 Preservation of composition.

We now show that our encoding preserves composition. Consider $S : A^\perp \parallel B$ and $T : B^\perp \parallel C$. We show that $\uparrow(T \circledcirc S) \cong \uparrow T \circledcirc \uparrow S$.

D.4.1 From $\uparrow T \circledcirc \uparrow S$ to $\uparrow(T \circledcirc S)$. Consider $w \in \mathscr{C}(\uparrow T \circledast \uparrow S)$. Consider the lifted configurations $(w_S, y^A_{\mathsf{r}} \parallel y^B_{\mathsf{r}}, y^A_{\mathsf{a}} \parallel y^B_{\mathsf{a}})$ and $(w_T, y^B_{\mathsf{r}} \parallel y^C_{\mathsf{r}}, y^B_{\mathsf{a}} \parallel y^C_{\mathsf{a}})$ associated to $\Pi_S\, w$ and $\Pi_T\, w$.

Lemma D.11. *We have $lbl_S(w_S) \cap B = T\, w_T \cap B$.*

Proof. Let $lbl_S(s) \in lbl_S(w_S) \cap B$. If s is negative, then since $T\, x_T \subseteq^- y^B_{\mathsf{r}} \parallel y^C_{\mathsf{r}}$ and $lbl_S(s)$ is positive and in y^B_{r}, then it is $lbl_T(x_T)$ as desired.

Otherwise, assume it is positive. If it is not maximal, then $lbl_S(x_S) \in y^B_{\mathsf{a}} \subseteq T x_T \cap B$ as desired. If it is maximal, the element $e \in [w]$ such that $\Pi_S e = (s, \mathsf{r})$. Note that $\Pi_T(e) = (t, \mathsf{r})$ with negative $t \in y^B_{\mathsf{r}}$. Then this event cannot be maximal in $[w]$ as it is not visible, hence there must exist $e < e'$ with $\Pi_T(e') = (t'^+, \mathsf{r})$. Since t' is positive and in y^B_{r}, then $t' \in w_T$ and so is t as desired. □

As a result, we get a synchronized configuration

$$\varphi_w : w_S \parallel (w_T)_* \parallel T(w_T) \cap C \simeq S(w_S) \cap A \parallel (w_S)_* \parallel w_T.$$

Lemma D.12. *Assume that $w \mathrel{-\!\!\subset} w'$. Then $\varphi_w \mathrel{-\!\!\subset} \ldots \mathrel{-\!\!\subset} \varphi_{w'}$ and each step is in $C_{\varphi_{w'}}$.*

Proof. Assume that $w \overset{e}{\mathrel{-\!\!\subset}} w'$. We proceed by case distinction:

(1) If e is neutral for, say, $\uparrow S$, that is $\Pi_{\uparrow S}(e) = (s, *)$. Then $\varphi_w \overset{(0,s),(1,s)}{\mathrel{-\!\!\subset}}$, and the step is indeed valid. Similar for an event in T.
(2) If e is a negative visible event, then $w_S = w'_S$ and $w'_T = w_T$ hence $\varphi_w = \varphi_{w'}$.
(3) If e is an acknowledgement, then $\varphi_w = \varphi_{w'}$.
(4) If e is a request sent on B then $\varphi_w = \varphi_{w'}$
(5) If e is a positive visible request, say on A, then we can look in $\uparrow T \circledast \uparrow S$ which requests are below e but whose acknowledgements are above that of e. This gives a set of events X such that $\varphi_w \overset{X}{\mathrel{-\!\!\subset}} \varphi_{w'}$ where $X \in C_{\varphi_{w'}}$. Acyclicity comes from the total order in the pullback between the coincident requests. □

As a result $\varphi_w \in \mathscr{C}(T \circledast S)$.

Lemma D.13. *If the maximal elements of w are essential, then so are those of φ_w.*

Proof. Consider a maximal element of φ_w. We show it cannot be mapped to B: assume it corresponds to a pair (s,t). Then s is maximal in x_S and $t \in x_T$. Then $(lbl_S(s), \mathsf{r})$ and $(lbl_S(s), \mathsf{a})$ cannot be maximal in w by assumption, so there must must be an event in w above them, which is not negative visible by courtesy. This event would appear in φ_w as well, which contradicts the assumption. □

Lemma D.14. *If the maximal elements of w are essential, then $(\varphi_w, y_\mathsf{r}^A \parallel y_\mathsf{r}^C, y_\mathsf{a}^A \parallel y_\mathsf{a}^C)$ is a lifted configuration of $T \circledcirc S$.*

Proof. Mostly routine verification using the fact that $(w_S, y_\mathsf{r}^A \parallel y_\mathsf{r}^B, y_\mathsf{a}^A \parallel y_\mathsf{a}^B)$ and $(w_T, y_\mathsf{r}^B \parallel y_\mathsf{r}^C, y_\mathsf{a}^B \parallel y_\mathsf{a}^C)$ are lifted configurations. □

Hence we have a monotonic map $\uparrow T \circledcirc \uparrow S$ to $\uparrow(T \circledcirc S)$.

D.4.2 From $\uparrow(T \circledcirc S)$ to $\uparrow T \circledcirc \uparrow S$. Consider $z \in \mathscr{C}(\uparrow(T \circledcirc S))$, and write

$$(T \circledcirc S)z = (y_\mathsf{r}^A \uplus y_\mathsf{a}^A) \parallel (y_\mathsf{r}^C \uplus y_\mathsf{a}^C).$$

Moreover, remember that $\pi\, z \in \mathscr{C}(T \circledcirc S)$. Define $z_S = \Pi_S([\pi\, z]) \in \mathscr{C}(S)$ and $z_T = \Pi_T([\pi\, z]) \in \mathscr{C}(T)$. We also define

$$y_\mathsf{r}^B := lbl_S(z_S) \cap B = T\, z_T \cap B.$$

Write $z_B = \{e \in [z] \mid lbl_S(\Pi_S\, e) \in B\}$ and $\mu : z_B \to B$ to be the composite $S \circ \Pi_S$. We then reconstruct the acknowledgements on B:

$$y_\mathsf{a}^B := \{\mu\, e \mid e \in z_B \wedge e \text{ is separated}\} \cup \{\mu\, e \mid e \in z_B \wedge e \in [z)\}$$
$$\{\mu\, e \mid e \in z_B \wedge \exists e'^- \in z_v, e' \equiv e \wedge (T \circledast S)e \in y_\mathsf{a}^A \parallel y_\mathsf{a}^C\}$$

Lemma D.15. *The tuples $(z_S, y_\mathsf{r}^A \parallel y_\mathsf{r}^B, y_\mathsf{a}^A \parallel y_\mathsf{a}^B)$ and $(z_T, y_\mathsf{r}^B \parallel y_\mathsf{r}^C, y_\mathsf{a}^B \parallel y_\mathsf{a}^C)$ are lifted configurations of S and T respectively.*

Proof. The two statements are symmetric; we only show the case for S.

(1) Because μ is a map of event structures, both y_r^B and y_a^B are configurations of B. Moreover, $y_\mathsf{a}^B \subseteq y_\mathsf{r}^B$ hence $y_\mathsf{r} \uplus y_\mathsf{a} \in \mathscr{C}(\uparrow B)$.
(2) It is easy to see that $y_\mathsf{a}^A \parallel y_\mathsf{a}^B \subseteq lbl_S(z_S)$. Consider $b^+ \in y_\mathsf{r}^A \parallel y_\mathsf{r}^B$. If $b \in y_\mathsf{r}^A$, then we have $b \in (T \circledcirc S)z \cap A = Sz_S \cap A$ as desired. If $b^+ \in y_\mathsf{r}^A$, then $b \in Sz_S$ by definition.
(3) Let $s^- \in max(z_S)$ and $lbl_S(s) \notin y_\mathsf{a}^A \parallel y_\mathsf{r}^B$. If $lbl_S(s) \in A$, then we just apply the fact that $(z, y_\mathsf{r}^A \parallel y_\mathsf{r}^C, y_\mathsf{a}^A \parallel y_\mathsf{a}^C)$ is a lifted configuration. If $lbl_S(s) \in B$, we know that if s was separated, the $lbl_S(s) \in y_\mathsf{r}^B$, absurd.
(4) If $s^+ \in [z_S)$, then $s = \Pi_S\, e$ with $e \in [z]$. Moreover, since s is not maximal in z_S, then e is not maximal in $[z]$, hence we can conclude by case distinction on whether $lbl_S(s)$ is in A or B.
(5) Consider $s^+ \equiv_{z_S} s'^-$ and consider $lbl_S(s') \in y_\mathsf{a}^A \parallel y_\mathsf{a}^B$. If both belong to A, we can conclude because $(z, y_\mathsf{r}^A \parallel y_\mathsf{r}^C, y_\mathsf{a}^A \parallel y_\mathsf{a}^C)$ is a lifted configuration. If s' is in A and s in B, we can also conclude by the definition of y_a^B. If both are in B, then we know that $s = \Pi_S\, e$ and $s' = \Pi_S\, e'$ with $e, e' \in [z]$. Since both are events mapped to B they cannot be visible, hence cannot be maximal in $[z]$, and $lbl_S(s) = \mu\, e$ and $lbl_S(s') = \mu\, e'$ are both in y_r^B as required.

□

Write $x_S \in \mathscr{C}(\uparrow S)$ and $x_T \in \mathscr{C}(\uparrow T)$ for the corresponding configurations. We know that $Sx_S \cap B = y_\mathsf{r}^B \uplus y_\mathsf{a}^B = Tx_T \cap B$. Hence, there exists a synchronized configuration $\varphi_z \in \mathscr{C}(\uparrow T \circ \uparrow S)$. We need to show it is reachable.

Lemma D.16. *The synchronized configuration φ_z is reachable and its maximal events are visible.*

Proof. *Reachability.* Assume that there is a cycle in φ_z, wlog we can assume it is in B, eg.

$$(e_1, \alpha_1) < \ldots < (e_n, \alpha_n) < (e_1, \alpha_1).$$

where $\alpha_i \in \{\mathsf{r}, \mathsf{a}\}$ and $e_i \in \pi\, z$. Since we can project these events to $T \circledast S$ in a monotonic way, we get that $\{e_1, \ldots, e_n\}$ is a coincidence X of $\pi\, z$. Since $(e_i, \alpha_i) < (e_{i+1}, \alpha_{i+1})$, it must be that e_i and e_{i+1} are related in the graph (X, Π_S, Π_T). As a result, the cycle in φ_z induces a cycle in (X, Π_S, Π_T) which is absurd since $\pi_1\, z \in \mathscr{C}(T \circledcirc S)$.

Maximal events. The maximal events of φ_z cannot be projected to an element of B, as otherwise there would be a maximal element of z projected to an element of $\uparrow B$. □

Hence $\varphi_z \in \mathscr{C}(\uparrow T \circledcirc \uparrow S)$. This defines a monotonic map $\uparrow(T \circledcirc S)$ to $\uparrow T \circledcirc \uparrow S$ which is the inverse to the previous map.

We can then conclude:

Theorem 7.5. *The operation $\uparrow$ defines a faithful functor $\mathbf{CG}_S \to \mathbf{CG}_A$,* i.e.,

$$(1)\ \uparrow \mathbb{CC}_A \cong \mathbb{C}\mathbb{C}_{\uparrow A} \qquad (2)\ \uparrow(R \circledcirc S) \cong \uparrow R \circledcirc \uparrow S \qquad (3)\ \uparrow R \cong \uparrow S \Rightarrow S \cong R$$

D.5 Well-acknowledging strategies

Lemma 7.8. *For any $\sigma : S \rightharpoonup A$, the strategy $\uparrow\sigma$ is well-acknowledging.*

Proof. (1) First, since copycat is deterministic, $X \in \mathrm{Con}_{\uparrow S}$ if and only if $\Pi_{\overline{S}} X \in \mathscr{C}(\overline{S})$. We then conclude since consistency in $\overline{S}$ does not depend on the presence of acknowledgements.
(2-3) Consequence of Lemma 7.6.
(4) If s is a positive request, it must be acknowledged by receptivity. Otherwise, because copycat is a tree and s is not maximal, there exists $s \rightarrowtriangle s'$ in the interaction $\mathbb{C}\mathbb{C}_{\uparrow S} \circledast \overline{S}$ with $\Pi_{\overline{S}} s'$ defined. Hence it is acknowledged in $\overline{S}$, and copycat will eventually forward this acknowledgement. □

Construction of the inverse.

Lemma D.17. *Let $S : S \rightharpoonup \uparrow A$ be well-acknowledging. For $x \in \mathscr{C}(S)$ and $s \in x$, if s has two distinct acknowledgements s_1 and s_2 compatible with x, then s is maximal in x and negative.*

Proof. Clearly, s cannot be positive: positive requests have a unique acknowledgement by receptivity. Assume that negative s has a successor $s' \in x$. If $lbl_S(s) \rightarrowtriangle lbl_S(s')$, then s' is an acknowledgement of s, and since it is in x it must be the only one by local receptivity: absurd. Hence s' must be positive. Write $s_1' \leq s_1$ and $s_2' \leq s_2$ for two events compatible with x in minimal conflict. Because acknowledgements are not involved in minimal conflict, and non maximal requests must be acknowledged, we must have $s_1', s_2' \in \downarrow S$. Moreover, $s_1' \prec s$ and $s \prec s'$. Hence $s_1' \prec s'$ by transitivity. Since s' is positive, this implies that $s_1' \leq s'$ and $s_1' \in x$. Similarly $s_2' \in x$ which is absurd since s_1' and s_2' are not compatible. □

Lemma D.18. *Let $S : S \rightharpoonup \uparrow A$ be well-acknowledging. If $s \in S$ is not negative maximal, then it is acknowledged.*

Proof. If s is positive, then receptivity forces it to be acknowledged. If s is negative not maximal, consider $s \rightarrowtriangle s'$ with $s' \in S$. By courtesy, either $lbl_S(s) \rightarrowtriangle lbl_S(s')$ and s' is an acknowledgement of s; or s' is positive. Then, since s' is positive, in the interaction $\mathbb{C}\mathbb{C}_{\uparrow A} \circledast \overline{S}$, we must have $s \rightarrowtriangle e \rightarrowtriangle e' \rightarrowtriangle s'$ with $\Pi_{\overline{S}}(e)$ defined and projects to the same event as s. In $\overline{S}$, all events are acknowledged, so e has an acknowledgement, which will eventually be forwarded by copycat. □

Lemma D.19. *Let $S : S \rightharpoonup \uparrow A$ be well-acknowledging. For $x \in \mathscr{C}(\downarrow S)$ without maximal negative requests, there exists an extension $x \subseteq x'$ by only acknowledgements such that all requests of x are acknowledged in x'.*

Proof. This amounts to showing that if $s \in x$, then there exists an acknowledgement $s' \in S$ such that $x \cup [s']$ is a configuration. Once again, the difficulty is for negative requests. □

Proposition 7.9. *If S is well-acknowledging, then $\downarrow S$ is a coincident strategy. Moreover $\uparrow\downarrow S \cong S$.*

Proof. That $\downarrow S$ is a coincident strategy is a simple observation.

To show that $\uparrow T \cong S$, we show that lifted configurations are order-isomorphic to configurations of S.

From S. Consider $z \in \mathscr{C}(S)$. First, define y_{r}^z and y_{a}^z the unique configurations of A such that $lbl_S(z) = y_{\mathsf{r}}^z \uplus y_{\mathsf{a}}^z$. Then, define $x_z := [z^+] \cap \downarrow S$. Note that x_z contains all the requests of z by Lemma D.18. By Lemma 7.6, it is downclosed.

We show that $(x_z, y_{\mathsf{r}}^z, y_{\mathsf{a}}^z)$ is a lifted configuration of $\downarrow S$.

(1) By definition
(2) Clearly $y_{\mathsf{a}}^z \subseteq \downarrow S(x_z) \subseteq y_{\mathsf{r}}^z$. Moreover, if $a^+ \in y_{\mathsf{r}}^z$, then $a = lbl_S(s)$ with $s \in z$. Since s is a positive request, it is in T hence $s \in x_z$ as desired.
(3) Consider $s^- \in max(x_z)$ such that $lbl_S(s) \notin y_{\mathsf{a}}^z$. Since $lbl_S(s) \notin y_{\mathsf{a}}^z$, s is not acknowledged in z, hence s is maximal in $[z^+]$. As a result, s must be coincident to a positive event.
(4) Consider $s^+ \in [x_z)$. We have $s \preceq s'$ in x_z. By receptivity, s has a unique acknowledgement s_0. If s' is neutral, then we must have $s_0 \preceq s'$ and $s_0 \in z$ which entails $lbl_S(s) \in y_{\mathsf{a}}^z$. Otherwise, writing s_1' for an acknowledgement of s', we have $s \preceq s_1'$ which must imply $s_0' \preceq s_1'$. Since the maximal events of x_z are positive, we can assume that s' is positive. Then courtesy implies that $s_0' \preceq s'$ as desired.
(5) Consider $s^+ \equiv s'^-$ a coincidence of x_z such that $lbl_S(s') \in y_{\mathsf{a}}^z$. By assumption $s \preceq s'$, hence s must be below the acknowledgement of s' in z. By courtesy, this implies that the unique acknowledgement of s must also be below it, which by downclosure of z implies that it is present in z as desired.

To S. Consider now a lifted configuration $\mathfrak{x} = (x, y_{\mathsf{r}}, y_{\mathsf{a}})$ of $\downarrow S$. First, we extend $[x]_S$ by receptivity such that $\downarrow lbl_S(x') = y_{\mathsf{r}}$, and then by receptivity to include all the negative events in y_{a}. This gives a configuration $z \in \mathscr{C}(S)$, in which only negative requests are not acknowledged. Then, for any $s \in max(x)$, there exists a unique acknowledgement s' of s compatible with x. Moreover, assume that $s_0 \rightarrowtriangle s'$. If s_0 is neutral or a request, then $s_0 \prec s$, hence since x is downclosed, $s_0 \in x \subseteq z$. If s_0 is an acknowledgement, it must be negative, and the corresponding request is coincident to s. As a result z extends, only by acknowledgements to z' where all the requests in y_{a} must be acknowledged. Moreover, all acknowledged requests are easily shown to be in y_{a} as result, we have

$$lbl_S(z') = y_{\mathsf{r}} \uplus y_{\mathsf{a}}.$$

as desired.

By receptivity, these constructions are inverse to each other, hence the desired isomorphism. □

D.6 Encoding and behavioural equivalences

Given a coincident strategy $S : A$, remember the order-isomorphism

$$\varphi_S : \mathscr{C}(S) \cong \mathscr{C}_c(S)$$

Lemma D.20. *Let $S : A$ be a coincident strategy and $x \in \mathscr{C}(S)$. If $\varphi_S(x)$ extends by neutral events to z then $z = \varphi_S(x')$ for some $x \subseteq x'$.*

Proof. Since $\varphi_\sigma(x)$ is complete, z is also complete. Hence $x' = \varphi_\sigma^{-1}(z)$ is the desired configuration. □

Lemma 7.11. *For $S, R : A$, if $\uparrow S \approx \uparrow R$ then $S \approx R$.*

Proof. Consider a weak bisimulation $\mathcal{R} : \uparrow S \approx \uparrow R$. Define

$$\downarrow \mathcal{R} = \{(x, y) \in \mathscr{C}(S) \times \mathscr{C}(R) \mid \varphi(x)\mathcal{R}\varphi(y)\}.$$

We show it is a weak bisimulation.

Assume $x \downarrow \mathcal{R}y$ and consider $x \overset{X}{\text{—}\!\subset} x'$. There are several cases:

- If $X = \{s\}$ with s neutral: then $\varphi(x) \text{–}\!\subset \varphi(x')$ (neutral extension). Applying the assumption, we get that: $\varphi(y) \subseteq z$ where the extension only contains neutral events, and we have $\varphi(x \cup \{s\})\mathcal{R}z$. Since z extends $\varphi(y)$ by neutral events, y extends to y' with $\varphi(y') = z$ by Lemma D.20. By construction we have $x' \downarrow \mathcal{R}y'$.
- If $X = \{s\}$ with s visible: then $\varphi(x) \overset{e}{\text{—}\!\subset} \overset{e'}{\text{—}\!\subset} \varphi(x')$ where $lbl_{\uparrow S}(e) = (lbl_S(s), \mathsf{r})$, $lbl_{\uparrow S}(e') = (lbl_S(s), \mathsf{a})$. Applying the assumption, we get that $\varphi(y)$ extends to z such that the visible events of $z \setminus \varphi(y)$ are the request and acknowledgement of $lbl_S(s)$. Then z by construction is complete, hence is $\varphi(y')$ and we can conclude.
- If $X = \{s_1^-, s_2^+\}$: write $a = lbl_S(s_1)$ and $b = lbl_S(s_2)$. In $\uparrow S$, we have by construction:

$$\varphi(x) \overset{e_1}{\text{—}\!\subset} \subseteq \overset{e_2}{\text{—}\!\subset} \overset{e_1'}{\text{—}\!\subset} \subseteq \overset{e_2'}{\text{—}\!\subset} z$$

  where

$$lbl_{\uparrow S}(e_1) = (a, \mathsf{r}) \quad lbl_{\uparrow S}\, e_2 = (b, \mathsf{r}) \quad lbl_{\uparrow S}(e_1') = (b, \mathsf{a}) \quad lbl_{\uparrow S}(e_2')$$

  and the inclusions only contain neutral events. As before, we apply the assumption to get that $\varphi(y)$ extensd to z. Since z is complete (we added two requests and their acknowledgements), $z = \varphi(y')$. Moreover, by construction, $\tau(y' \setminus y) = \{a, b\}$. Hence we can conclude.

□

E COUNTEREXAMPLES

E.1 Non adequacy of the encoding of session π into internal session π

Example E.1. Remember the two processes

$$P = a!\mathsf{go}\langle\overline{b}\rangle \quad \textit{and} \quad Q = (\nu c)(a!\mathsf{go}\langle\overline{c}\rangle.\, c?\mathsf{ack}.\, \overline{b}!\mathsf{ack}) \qquad \text{typeable on } \Delta = \overline{b} : !\mathsf{ack}, a : !\mathsf{go}\langle!\mathsf{ack}\rangle$$

Consider now the context

$$C[] = (\nu ab)([] \mid a?\mathsf{go}(x).\, (x!\mathsf{ack}.\, b?\mathsf{ack}.\, c_f! + (b?\mathsf{ack} \mid x!\mathsf{ack}).\, c_s!)$$

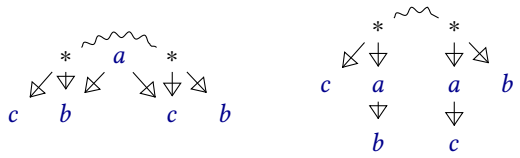
where we use the notation $(\alpha \mid \beta).\, P$ for $(\nu d)(\alpha.\, d?().\, P \mid \beta.\, \overline{d}!())$. We have that $C[P]$ and $C[Q]$ are well-typed processes on the context $c_s : !\mathsf{succ}, c_f : !\mathtt{fail}$.

The process $C[Q]$ has a weak barb on c_f but not $C[P]$, hence they cannot be barbed congruent:

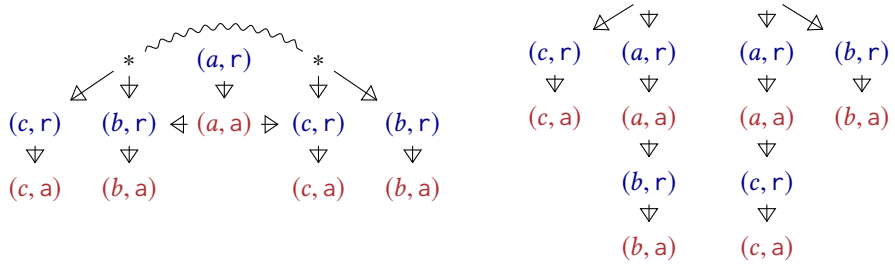
$$\begin{aligned} C[Q] &\to (\nu abc)(c?\mathsf{ack}.\, \overline{b}!\mathsf{ack} \mid (\overline{c}!\mathsf{ack}.b?\mathsf{ack}.\ c_f! + (b?\mathsf{ack} \mid \overline{c}!\mathsf{ack}).\, c_s!) \\ &\to c_f! \end{aligned}$$

E.2 Non full abstraction of the encoding

Example E.2. Consider the game A comprising three concurrent moves $a\ b\ c$, along with the two strategies S and R on A:



This two strategies are weakly bisimilar. They are, however not history-preserving weakly bisimilar: there is no weak bisimulation $\mathcal{R}$ satisfying $x\mathcal{R}y$ implies $x_v \cong y_v$. Indeed, the left strategy can do an action a without comitting on the causal history, while the right one cannot. If we compute their encoding, we get:



However, these two strategies are not bisimilar. The left one can do (a, r) and still keep the possibility of performing (b, r) and (c, r) while the righthand strategy must commit before issuing (a, r).