Games, mobile processes, Dfunctions
Guilhem Jaber, Davide Sangiorgi

To cite this version:
Guilhem Jaber, Davide Sangiorgi. Games, mobile processes, Dfunctions. CSL 2022 - 30th EACSL Annual Conference on Computer Science Logic, Feb 2022, Göttingen, Germany. pp.1-35. hal-03407123

HAL Id: hal-03407123
https://hal.archives-ouvertes.fr/hal-03407123
Submitted on 28 Oct 2021

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

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

Guilhem Jaber
Université de Nantes, France

Davide Sangiorgi
Università di Bologna, Italy

Abstract

We establish a tight connection between two models of the λ-calculus, namely Milner’s encoding into the π-calculus (precisely, the Internal π-calculus), and operational game semantics (OGS). We first investigate the operational correspondence between the behaviours of the encoding provided by π and OGS. We do so for various LTSs: the standard LTS for π and a new ‘concurrent’ LTS for OGS; an ‘output-prioritised’ LTS for π and the standard alternating LTS for OGS. We then show that the equivalences induced on λ-terms by all these LTSs (for π and OGS) coincide. These connections allow us to transfer results and techniques between π and OGS. In particular we import up-to techniques from π onto OGS and we derive congruence and compositionality results for OGS from those of π. The study is illustrated for call-by-value; similar results hold for call-by-name.

1 Introduction

The topic of the paper is the comparison between Operational Game semantics (OGS) and the π-calculus, as generic models or frameworks for the semantics of higher-order languages. Game semantics [4,20] provides intentional models of higher-order languages, where the denotation of a program brings up its possible interactions with the surrounding context. Distinct points of game semantics are the rich categorical structure and the emphasis on compositionality. Game semantics provides a modular characterization of higher-order languages with computational effects like control operators [24], mutable store [3,5] or concurrency [15,26]. This gives rise to the “Semantic Cube” [2], a characterization of the absence of such computational effects in terms of appropriate restrictions on the interactions, with conditions like alternation, well-bracketing, visibility or innocence. For instance, well-bracketing corresponds to the absence of control operators like call/cc.

Game semantics has spurred Operational Game Semantics (OGS) [16,22,23,27,29], as a way to describe the interactions of a program with its environment by embedding programs into appropriate configurations and then defining rules that turn such configurations into an LTS. Besides minor differences on the representation of causality between actions, the main distinction with “standard” game semantics is in the way in which the denotation of programs is obtained: via an LTS, rather than, compositionally, by induction on the structure of the programs (or their types). It is nonetheless possible to establish a formal correspondence between these two representations [29].

OGS is particularly effective on higher-order programs. To avoid being too intensional, functional values exchanged between the program and its environment are represented as atoms, seen as free variables. Therefore OGS configurations include open terms. The basic actions in the LTS produced by OGS represent the calls and returns of functions between a program and its environment. The OGS semantics has been shown fully-abstract, that is, to characterize observational equivalence, for a wide class of programming languages, including effectful subsets of ML [21,27], fragments of Java [23], aspect-oriented programs [22]. The conditions in the above-mentioned Semantic Cube (alternation, well-bracketing, etc.) equally
In this paper, we consider forms of OGS for the pure untyped call-by-value \( \lambda \)-calculus, which enforce some of such conditions. Specifically we consider: an *Alternating* OGS, where only one term can be run at a time, and the control on the interactions alternates between the term and the environment; and a *Concurrent* OGS, where multiple terms can be run in parallel.

The \( \pi \)-calculus is the paradigmatical name-passing calculus, that is, a calculus where names (a synonym for ‘channels’) may be passed around. In the literature about the \( \pi \)-calculus, and more generally in Programming Language theory, Milner’s work on functions as processes [32], which shows how the evaluation strategies of call-by-name \( \lambda \)-calculus and call-by-value \( \lambda \)-calculus [1,35] can be faithfully mimicked, is generally considered a landmark. The work promotes the \( \pi \)-calculus to be a model for higher-order programs, and provides the means to study \( \lambda \)-terms in contexts other than the purely sequential ones and with the instruments available to reason about processes. In the paper, \( \pi \)-calculus is actually meant to be the *Internal \( \pi \)-calculus* (I\( \pi \)), a subset of the original \( \pi \)-calculus in which only fresh names may be exchanged among processes [41]. The use of I\( \pi \) avoids a few shortcomings of Milner’s encodings, notably for call-by-value; e.g., the failure of the \( \beta_v \) rule (i.e., the encodings of \((\lambda x. M) V\) and \(M\{V/\!x\}\) may be behaviourally distinguishable in \( \pi \)).

Further investigations into Milner’s encodings [11,42] have revealed what is the equivalence induced on \( \lambda \)-terms by the encodings, whereby two \( \lambda \)-terms are equal if their encodings are behaviourally equivalent (i.e., bisimilar) I\( \pi \) terms. In call-by-value, this equivalence is *eager normal-form bisimilarity* [28], a tree structure proposed by Lassen (and indeed sometimes referred to as ‘Lassen’s trees’) as the call-by-value counterpart of Böhm Trees (or Lévy-Longo Trees).

In a nutshell, when used to give semantics to a language, major strengths of the \( \pi \)-calculus are its algebraic structure and the related algebraic properties and proof techniques; major strengths of OGS are its proximity to the source language — the configurations of OGS are built directly from the terms of the source language, as opposed to an encoding as in the \( \pi \)-calculus — and its flexibility — the semantics can be tuned to account for specific features of the source language like control operators or references.

The general goal of this paper is to show that there is a tight and precise correspondence between OGS and \( \pi \)-calculus as models of programming languages, and that such a correspondence may be profitably used to take advantage of the strengths of the two models. We carry out the above program in the specific case of (untyped) call-by-value \( \Lambda_V \), which is richer and (as partly suggested above) with some more subtle aspects than call-by-name. However similar results also hold for call-by-name; see Appendix I for the technical details for more comments on it. Analogies and similarities between game semantics and \( \pi \)-calculus have been pointed out in various papers in the literature (e.g., [6,19]; see Section 9), and used to, e.g., explain game semantics using \( \pi \)-like processes, and enhance type systems for \( \pi \)-terms. In this paper, in contrast, we carry out a direct comparison between the two models, on their interpretation of functions.

We take the (arguably) canonical representations of \( \Lambda_V \) into I\( \pi \) and OGS. The latter representation is Milner’s encoding, rewritten in I\( \pi \). We consider two variant behaviours for the I\( \pi \) terms, respectively produced by the ordinary LTS of I\( \pi \), and by an ‘output-prioritised’ LTS, opLTS, in which input actions may be observed only in absence of outputs and internal actions. Intuitively, the opLTS is intended to respect sequentiality constraints in the I\( \pi \) terms: an output action stands for an ongoing computation (for instance, returning the result of a previous request) whereas an input action starts a new computation (for instance,
a request of a certain service); therefore, in a sequential system, an output action should have priority over input actions. For OGS, the \( \lambda_V \) representation is the straightforward adaptation of the OGS representations of typed \( \lambda \)-calculi in the literature, e.g., [27].

We then develop a thorough comparison between the behaviours of the OGS and \( I\pi \) representations. For this we define a mapping from OGS configurations to \( I\pi \) processes. We also exploit the fact that, syntactically, the actions in the OGS and \( I\pi \) LTSs are the same. We derive a tight correspondence between the two models, which allows us to transfer techniques and to switch freely between the two models in the analysis of the OGS and \( I\pi \) representations of \( \lambda_V \), so to establish new results or obtain new proofs. On these aspects, our main results are the following:

1. We show that the representation of \( \lambda_V \) in the Alternating OGS is behaviourally the same as the representation in \( I\pi \) assuming the opLTS. Thus the semantics on \( \lambda \)-terms induced by the OGS and \( I\pi \) representations coincide. The same results are obtained between the Concurrent OGS and \( I\pi \) under its ordinary LTS.

2. We transfer ‘bisimulation up-to techniques’ for \( I\pi \), notably a form of ‘up-to context’, onto (Concurrent) OGS. The result is a powerful technique, called ‘up-to composition’, that allows us to split an OGS configuration into more elementary configurations during the bisimulation game.

3. We show that the semantics induced on \( \lambda_V \) by the Alternating and by the Concurrent OGS are the same, both when the equality in OGS is based on traces and when it is based on bisimulation. In other words, all the OGS views of \( \lambda_V \) (Alternating or Concurrent, traced-based or bisimulation-based) coincide. Moreover, we show that such induced semantics is the equality of Lassen’s trees. We derive the result in two ways: one in which we directly import it from \( I\pi \); the other in which we lift eager normal-form bisimulations into OGS bisimulations via the up-to-composition technique.

4. We derive congruence and compositionality properties for the OGS semantics, as well as a notion of tensor product over configurations that computes interleavings of traces. The results about OGS in (2-4) are obtained exploiting the mapping into \( I\pi \) and its algebraic properties and proof techniques, as well as the up-to-composition technique for OGS imported from \( I\pi \).

Structure of the paper. Sections 2 to 5 contain background material: general notations, \( I\pi \), \( \lambda_V \), the representations of \( \lambda_V \) in the Alternating OGS (A-OGS) and in \( I\pi \). The following sections contain the new material. In Section 6 we study the relationship between the two \( \lambda_V \) representations, in \( I\pi \) using the output-prioritised LTS. In Section 7 we establish a similar relationship between a new Concurrent OGS (C-OGS) and \( I\pi \) using its ordinary LTS. We also transport up-to techniques onto OGS, and prove that all the semantics of \( \lambda_V \) examined (OGS, \( I\pi \), traces, bisimulations) coincide. We import compositionality results for OGS from \( I\pi \) in Section 8.

2 Notations

In the paper we use various LTSs and behavioural relations for them, both for OGS and for the \( \pi \)-calculus. In this section we introduce or summarise common notations.

We use a tilde, like in \( \tilde{a} \), for (possibly empty) tuples of objects (usually names). Let \( K \xrightarrow{\mu \tau} K' \) be a generic LTS (for OGS or \( I\pi \); the grammar for actions in the LTSs for OGS and \( I\pi \) will be the same). Actions, ranged over by \( \mu \), can be of the form \( \alpha(b), \pi(b), \tau \), and \( \tilde{a} \), where \( \tau \), called silent or (invisible) action, represents an internal step in \( K \), that is, an action that does not require interaction with the outside, and \( \tilde{a} \) is a special action performed by
abstractions in $\pi$ and initial configurations in OGS. If $\mu \neq \tau$ then $\mu$ is a visible action; we
use $\ell$ to range over them. We sometimes abbreviate $\xrightarrow{\mu}_g$ as $\xrightarrow{\mu}$.
We write $\xrightarrow{\mu}_g$ for the reflexive and transitive closure of $\xrightarrow{\mu}_g$. We also write $K \xrightarrow{\mu}_g K'$ if $K \xrightarrow{\mu}_g \xrightarrow{\mu}_g K'$
(the composition of the three relations). Then $\xrightarrow{\mu}_g$ is $\xrightarrow{\mu}_g$ if $\mu \neq \tau$, and $\xrightarrow{\mu}_g$ if $\mu = \tau$.

Traces, ranged over by $s$, are finite (and possibly empty) sequences of visible actions.
If $s = \ell_1, \ldots, \ell_n$ ($n \geq 0$), then $K \xrightarrow{s}_g K'$ holds if there are $K_0, \ldots, K_n$ with $K_0 = K$, $K_n = K'$, and $K_i \xrightarrow{\ell_i+1}_g K_{i+1}$ for $0 \leq i < n$; and $K \xrightarrow{s}_g$ if there is $K'$ with $K \xrightarrow{s}_g K'$.

Two states $K_1, K_2$ of the LTS are trace equivalent, written $K_1 \simeq_g K_2$, if $(K_1 \xrightarrow{s}_g K_2)$, for all $s$.

Similarly, bisimilarity, written $\approx_g$, is the largest symmetric relation on the state of the
LTS such that whenever $K_1 \approx_g K_2$ then $K_1 \xrightarrow{s}_g K_1'$ implies there is $K_2'$ with $K_2 \xrightarrow{s}_g K_2'$
and $K_1' \approx_g K_2'$. For instance, in the $\pi$ LTS $\xrightarrow{\mu}_g$ of Section 3.1, $P \equiv_\pi Q$ means that the $\pi$
processes $P$ and $Q$ are trace equivalent, and $P \equiv_\pi Q$ means that they are bisimilar.

\begin{remark}[bound names] In an action $a(b)$ or $\nu b(\pi b)$ or $\tilde{b}$, name $a$ is free whereas $\tilde{b}$ are
bound; the free and bound names of a trace are defined accordingly. Throughout the paper,
in any statement (concerning OGS or $\pi$), the bound names of an action or of a trace that
appears in the statement are supposed to be all fresh; i.e., all distinct from each other and
from the free names of the objects in the statement. \end{remark}

3 Background

3.1 The Internal $\pi$-calculus

The Internal $\pi$-calculus, $\pi$, is, intuitively, a subset of the $\pi$-calculus in which all outputs
are bound. This is syntactically enforced by having outputs written as $\pi b(\tilde{b})$ (which in the
$\pi$-calculus would be an abbreviation for $\nu b(\pi b)$). All tuples of names in $\pi$ are made of
pairwise distinct components. Abstractions are used to write name-parametrised processes,
for instance, when writing recursive process definitions. The instantiation of the parameters
of an abstraction $B$ is done via the application construct $B(\tilde{a})$. Processes and abstractions
form the set of agents, ranged over by $T$. Lowercase letters $a, b, \ldots, x, y, \ldots$ range over the
infinite set of names. The grammar of $\pi$ is thus:

$$
P \equiv 0 \mid a(b).P \mid \pi b.P \mid \nu a.P \mid P_1 \mid P_2 \mid !a(b).P \mid B(\tilde{a}) \quad \text{(processes)}
$$

$$
B \equiv \begin{array}{c}
\tilde{a}P \\
K
\end{array} \quad \text{(abstractions)}
$$

The operators have the usual meaning; we omit the standard definition of free names, bound
names, and names of an agent, respectively indicated with $\text{fn}(-)$, $\text{bn}(-)$, and $\text{n}(-)$. In the
grammar, $K$ is a constant, used to write recursive definitions. Each constant $K$ has a defining
equation of the form $K \equiv (\tilde{x})P$, where $(\tilde{x})P$ is name-closed (that is, without free names);
$\tilde{x}$ are the formal parameters of the constant $\tilde{}$. Replication could be avoided in the syntax
since it can be encoded with recursion. However its semantics is simple, and it is useful in
encodings.

An application redex $((\tilde{x})P)(\tilde{a})$ can be normalised as $P[\tilde{a}/\tilde{x}]$. An agent is normalised if
all such application redexes have been contracted. In the remainder of the paper we identify
an agent with its normalised expression.

Since the calculus is polyadic, we assume a sorting system \cite{31} to avoid disagreements in
the arities of the tuples of names. Being not essential, it will not be presented here.
Operational semantics and behavioural relations

In the LTS for $\pi$, recalled in Appendix A, transitions are of the form $T \xrightarrow{\mu} T'$, where the bound names of $\mu$ are fresh, i.e., they do not appear free in $T$.

Trace equivalence ($\equiv$) and bisimilarity ($\approx$) have been defined in Section 2. We refer to Appendix A for the standard definition of expansion, written $\preceq$. (The expansion relation $\preceq$ is an asymmetric variant of $\approx$ in which, intuitively, $P \preceq Q$ holds if $P \approx Q$ but also $Q$ has at least as many $\tau$-moves as $P$.) All behavioural relations are extended to abstractions by requiring ground instantiation of the parameters; this is expressed by means of a transition; e.g., the action $\bar{x}$ $P \xrightarrow{\bar{x}} P$ (see rule $\text{abs}$ in Appendix A).

The “up-to” techniques

The “up-to” techniques allow us to reduce the size of a relation $R$ to exhibit for proving bisimilarities. Our main up-to technique will be $\text{up-to context and expansion}$ [40], which admits the use of contexts and of behavioural equivalences such as expansion to achieve the closure of a relation in the bisimulation game. So the bisimulation clause becomes:

If $P \mathrel{R} Q$ and $P \xrightarrow{\mu} P''$ then there are a static context $C_{\text{ctx}}$ and processes $P'$ and $Q'$ s.t.

$$P'' \equiv C_{\text{ctx}}[P'], \quad Q \xrightarrow{\mu} Q' \equiv C_{\text{ctx}}[Q'] \text{ and } P'' \mathrel{R} Q'$$

where a static context is a context of the form $\nu\bar{c}(R\mid\bar{v})$.

We will also employ: $\text{bisimulation up-to } \approx_{\pi}$ [30], whereby bisimilarity itself is employed to achieve the closure of the candidate relation during the bisimulation game; a variant of bisimulation up-to context and expansion, called $\text{bisimulation up-to context and up-to } (\equiv_{\pi}, \approx_{\pi})$, in which, in ($\ast$), when $\mu$ is a visible action, expansion is replaced by the coarser bisimilarity, at the price of imposing that the static context $C_{\text{ctx}}$ cannot interact with the processes $P$ or $Q$. (This technique, as far as we know, does not appear in the literature.) Details on these techniques may be found in Appendix A.

3.2 The Call-By-Value $\lambda$-calculus

The grammar of the untyped call-by-value $\lambda$-calculus, $\Lambda_V$, has values $V$, terms $M$, evaluation contexts $E$, and general contexts $C$:

$$\begin{align*}
\text{Vals} & \quad V \triangleq x \mid \lambda x. M \\
\text{Terms} & \quad M, N \triangleq V \mid MN \\
\text{ECtxs} & \quad E \triangleq [\cdot] \mid VE \mid EM \\
\text{Ctxs} & \quad C \triangleq [\cdot] \mid \lambda x. C \mid MC \mid CM
\end{align*}$$

where $[\cdot]$ stands for the hole of a context. The call-by-value reduction $\rightarrow_{\pi}$ has two rules:

$$\begin{align*}
(\lambda x. M)V & \rightarrow_{\pi} M/V[x] \\
E[M] & \rightarrow_{\pi} E[N]
\end{align*}$$

In the following, we write $M \Downarrow M'$ to indicate that $M \Rightarrow_{\pi} M'$ with $M'$ an eager normal form, that is, either a value or a stucked call $E[xV]$.

4 Operational Game Semantics

We introduce the representation of $\Lambda_V$ in OGS. The LTS produced by the representation of a term intends to capture the possible interactions between the term and its environment. Values exchanged between the term and the environment are represented by names, akin to free variables, called variable names and ranged over by $x, y, z$. Continuations (i.e., evaluation
(Pr) \( (M, p, \gamma, \phi) \xrightarrow{\tau} (N, p, \gamma, \phi) \) when \( M \xrightarrow{\tau} N \)

(PA) \( (V, p, \gamma, \phi) \xrightarrow{\bar{p}(x)} (\gamma \cdot [x \mapsto V], \phi \uplus \{x\}) \)

(PQ) \( (E[xV], p, \gamma, \phi) \xrightarrow{\bar{x}(y,q)} (\gamma \cdot [y \mapsto V \cdot [y \mapsto (E, p)]], \phi \uplus \{y, q\}) \)

(OA) \( (\gamma \cdot [y \mapsto (E, p)], \phi) \xrightarrow{\bar{y}(x)} (E[x], p, \gamma, \phi \uplus \{x\}) \)

(OQ) \( (\gamma, \phi) \xrightarrow{x(y,p)} (V[y, p, \gamma, \phi \uplus \{y, p\}) \) when \( \gamma(x) = V \)

(IOQ) \( ([? \mapsto M], \phi) \xrightarrow{(p)} (M, p, \varepsilon, \phi \uplus \{p\}) \)

**Figure 1** The LTS for the Alternating OGS (A-OGS)

Actions \( \mu \) have been introduced in Section 2. In OGS, we have five kinds of (visible) actions:

- **Player Answers** (PA), \( \bar{p}(x) \), and **Opponent Answers** (OA), \( p(x) \), that exchange a variable \( x \) through a continuation name \( p \);
- **Player Questions** (PQ), \( \bar{x}(y, p) \), and **Opponent Questions** (OQ), \( x(y, p) \), that exchange a variable \( y \) and a continuation name \( p \) through a variable \( x \);
- **Initial Opponent Questions** (IOQ), \( (p) \), that introduce the initial continuation name \( p \).

**Remark 2.** The denotation of terms is usually represented in game semantics using the notion of pointer structure rather than traces. A pointer structure is defined as a sequence of moves, together with a pointer from each move (but the initial one) to a previous move that 'justifies' it. Taking a trace \( s \), one can reconstruct this pointer structure in the following way:

- an action \( \mu \) is justified by an action \( \mu' \) if the free name of \( \mu \) is bound by \( \mu' \) in \( s \) (here we are taking advantage of the 'freshness' convention on the bound names of traces, Remark 1).

**Environments**, ranged over by \( \gamma \), maintain the association from names to values and evaluations contexts, and are partial maps. A single mapping is either of the form \([x \mapsto V]\) (the variable \( x \) is mapped onto the value \( V \)), or \([p \mapsto (E, q)]\) (the continuation name \( p \) is mapped onto the pair of the evaluation context \( E \) and the continuation \( q \)).

There are two main kinds of configurations \( F \): active configurations \( (M, p, \gamma, \phi) \) and passive configurations \( (\gamma, \phi) \), where \( M \) is a term, \( p \) a continuation name, \( \gamma \) an environment and \( \phi \) a set of names called its name-support. Names in \( \text{dom}(\gamma) \) are called \( P \)-names, and those in \( \phi \backslash \text{dom}(\gamma) \) are called \( O \)-names. So we obtain a polarity function \( \text{pol}_F \) associated to \( F \), defined as the partial maps from \( \phi \) to \( \{O, P\} \) mapping names to their polarity. In the following, we only consider valid configurations, for which:

- \( \text{dom}(\gamma) \subseteq \phi \)
- \( \forall v(M), p \) are \( O \)-names;
- for all \( a \in \text{dom}(\gamma) \), the names appearing in \( \gamma(a) \) are \( O \)-names.

The LTS is introduced in Figure 1. It is called Alternating, since, forgetting the \( \text{Pr} \) transition, it is bipartite between active configurations, that perform Player actions, and passive configurations, that perform Opponent actions. Accordingly, we call Alternating the resulting OGS, abbreviated A-OGS. In the OA rule, \( E \) is "garbage-collected" from \( \gamma \), a behavior corresponding to linear continuations. More details on the rules may be found in Appendix C.

To build the denotation of a term \( M \), we introduce an initial configuration associated to it, written \( (\langle ? \mapsto M \rangle, \phi) \), with \( \phi \) the set of free variables we start with. When this set is taken to be the free variables of \( M \), we simply write it as \( \langle M \rangle \). In the initial configuration
\[ V[V] \triangleq (p) \, p(y). \, V^* [V](y) \quad V^* [\lambda x. \, M] \triangleq (y) \, !y(x, q). \, V[M](q) \quad V^* [x] \triangleq (y) \, y \triangleright x \]
\[ V[MN] \triangleq (p) \, \nu q \, (V[M](q) \mid q(y). \, \nu r \, (V[N](r) \mid r(w). \, \nu (q | w \triangleright | w \triangleright w \triangleright p))) \]

\textbf{Figure 2} The encoding of call-by-value \( \lambda \)-calculus into \( I\pi \)

the choice of the continuation name \( p \) is made, by performing an Initial Opponent question (IOQ). (Formally, initial configurations should be considered as passive configurations.)

\section{5 The encoding of call-by-value \( \lambda \)-calculus into the \( \pi \)-calculus}

We recall here Milner’s encoding of call-by-value \( \lambda \)-calculus, transplanted into \( I\pi \). The core of any encoding of the \( \lambda \)-calculus into a process calculus is the translation of function application. This becomes a particular form of parallel combination of two processes, the function and its argument; \( \beta \)-reduction is then modelled as a process interaction. As in OGS, so in \( I\pi \) the encoding uses \textit{continuation names} \( p, q, r, \ldots \), and \textit{variable names} \( x, y, v, w \ldots \). Figure 2 presents the encoding. Process \( a \triangleright b \) represents a \textit{link} (sometimes called forwarder; for readability we have adopted the infix notation \( a \triangleright b \) for the constant \( \triangleright \)). It transforms all outputs at \( a \) into outputs at \( b \); thus the body of \( a \triangleright b \) is replicated, unless \( a \) and \( b \) are continuation names:

\[ \triangleright \triangleq \begin{cases} (p, q). \, p(x). \, q(y). \, y \triangleright x & \text{if } p, q \text{ are continuation names} \\ (x, y). \, !x(z, p). \, \nu (w, q). \, (q \triangleright p \mid w \triangleright z) & \text{if } x, y \text{ are variable names} \end{cases} \]

The equivalence induced on call-by-value \( \lambda \)-terms by their encoding into \( I\pi \) coincides with Lassen’s \textit{eager normal-form (enf) bisimilarity} \([28]\). That is, \( V[M] \approx_{\pi} V[N] \) iff \( M \) and \( N \) are enf-bisimilar \([11]\). In proofs about the behaviour of the \( I\pi \) representation of \( \lambda \)-terms we sometime follow \([11]\) and use an optimisation of Milner’s encoding, reported in Appendix D together with its correctness (Lemma 47).

\section{6 Relationship between \( I\pi \) and A-OGS}

To compare the A-OGS and \( I\pi \) representations of the (call-by-value) \( \lambda \)-calculus, we set a mapping from A-OGS configurations and environments to \( I\pi \) processes. The mapping is reported in Figure 3. It is an extension of Milner’s encoding of the \( \lambda \)-calculus and is therefore indicated with the same symbol \( V \). The mapping uses a representation of environments \( \gamma \) as associative lists.

\begin{itemize}
  \item \textbf{Remark 3.} The encoding of a configuration \( F \) with name-support \( \phi \) does not depend on \( \phi \). This name-support \( \phi \) is used in OGS both to enforce freshness of names, and to deduce the polarity of names, as represented by the function \( \text{pol} \). And indeed, the process \( V[F] \) has its set of free names included in \( \phi \), and uses \( P \)-names in outputs and \( O \)-names in inputs. The polarity property could be stated in \( \pi \)-calculus using \( i/o \)-sorting \([34]\). Indeed, a correspondence between arenas of game semantics (used to enforce polarities of moves) and sorting has been explored \([18, 19]\).
\end{itemize}

\subsection{6.1 Operational correspondence}

The following theorems establish the operational correspondence between the A-OGS and \( I\pi \) representations. In Theorem 4, as well as in following theorems such as Theorems 5, 7,
Encoding of environments:
\[
V[[y \mapsto V] \cdot \gamma] \triangleq V[V](y) \mid V[\gamma]
\]
\[
V[[q \mapsto (E, p)] \cdot \gamma] \triangleq q(x).V[E](p) \mid V[\gamma]
\]
\[
V[e] \triangleq 0
\]

Encoding of configurations:
\[
V[[M, p, \gamma, \phi]] \triangleq V[M](p) \mid V[\gamma]
\]
\[
V[[\gamma, \phi]] \triangleq V[\gamma]
\]
\[
V[[?] \mapsto M], \phi] \triangleq V[M]
\]

\textbf{Figure 3} From OGS environments and configurations to Iπ

and 13, the appearance of the expansion relation \(\mathcal{E} \supseteq\) (in place of the coarser \(\approx\)), in the statement about silent actions, is essential, both to derive the statement in the theorems about visible actions, and to use the theorems in up-to techniques for Iπ (more generally, in applications of the theorems in which one reasons about the number of steps performed).

\textbf{Theorem 4.}
1. If \(F \xrightarrow{\mu} F'\), then \(V[F] \xrightarrow{\pi} \mathcal{E} V[F']\);
2. If \(F \xrightarrow{\ell} F'\), then \(V[F] \xrightarrow{\ell} \approx V[F']\).

\textbf{Theorem 5.}
1. If \(V[F] \xrightarrow{\pi} P\) then there is \(F'\) such that \(F \xrightarrow{\mu} F'\) and \(P \approx V[F']\);
2. If \(V[F] \xrightarrow{\pi} \ell P\) and \(\ell\) is an output, then there is \(F'\) such that \(F \xrightarrow{\ell} F'\) and \(P \approx V[F']\);
3. If \(F\) is passive and \(V[F] \xrightarrow{\ell} P\), then there is \(F'\) such that \(F \xrightarrow{\ell} F'\) and \(P \approx V[F']\).

In Theorem 5, a clause is missing for input actions from \(V[F]\) when \(F\) active. Indeed such actions are possible in Iπ, stemming from the (encoding of the) environment of \(F\), whereas they are not possible in A-OGS. This is rectified in Section 6.2, introducing a constrained LTS for Iπ, and in Section 7, considering a concurrent OGS.

\textbf{Corollary 6.} If \(F \xrightarrow{s} A\) then also \(V[F] \xrightarrow{s} \pi\).

\subsection{An output-prioritised Transition System}

We define an LTS for Iπ in which input actions are visible only if no output can be consumed, either as a visible action or through an internal action (i.e., syntactically the process has no unguarded output). The new LTS, called output-prioritised and indicated as opLTS, is defined on the top of the ordinary one by means of the two rules below. A process \(P\) is input reactive if whenever \(P \xrightarrow{\mu} P'\), for some \(\mu, P'\) then \(\mu\) is an input action.

\[
\frac{P \xrightarrow{\mu} P' \text{ P input reactive}}{P \xrightarrow{\mu} \gamma_\pi P'}
\]
\[
\frac{P \xrightarrow{\mu} \gamma_\pi P' \mu \text{ is an output or } \tau \text{ action}}{P \xrightarrow{\mu} \gamma_\pi P'}
\]

The opLTS captures an aspect of \textit{sequentiality} in \(\pi\)-calculi: a free input prefix is to be thought of as a service offered to the external environment; in a sequential system such a service is available only if there is no ongoing computations due to previous interrogations of the server. An ongoing computation is represented by a \(\gamma\)-action, indicating a step of computation internal to the server, or an output, indicating either an answer to a client or a request to an external server. The constraint imposed by the new LTS could also be formalised compositionally, see Appendix A.

Under the opLTS, the analogous of Theorem 4 continue to hold: in A-OGS configurations, input transitions only occur in passive configurations, and the encodings of passive configurations are input-reactive processes. However, now we have the full converse of Theorem 5 and, as a consequence, we can also establish the converse direction of Corollary 6.
Theorem 7.
1. If \( \mathcal{V}[F] \xrightarrow{\sigma \pi} P \) then there is \( F' \) such that \( F \rightarrow_\pi a F' \) and \( P \underset{\sigma}{\simeq} \mathcal{V}[F'] \).
2. If \( \mathcal{V}[F] \xrightarrow{\sigma \pi} P \) then there is \( F' \) such that \( F \xrightarrow{\pi} a F' \) and \( P \approx_\pi \mathcal{V}[F'] \).

Corollary 8. For any configuration \( F \) and trace \( s \), we have \( F \xrightarrow{\sigma \pi} a \iff \mathcal{V}[F] \xrightarrow{s} \sigma \pi \).

Remark 9. We recall that, following Remark 1 on the usage of bound names, in Corollary 8 the bound names in \( s \) are fresh; thus they do not appear in \( F \). (Similarly, in Theorems 5 and 7 for the bound names in \( \ell \).

For both results we first establish a correspondence result on strong transitions. See Appendix D for details.

Remark 10. Corollary 8 relies on Theorems 4 and 7. The corollary talks about the \( \text{opLTS} \) of \( \Pi; \) however the theorems make use of the ordinary expansion relation \( \simeq_\pi \), that is defined on the ordinary LTS. Such uses of expansion can however be replaced by expansion on the \( \text{opLTS} \) (defined as ordinary expansion, but on the \( \text{opLTS} \)). For more details on this, see Appendix B.

As a consequence of Corollary 8, trace equivalence is the same, on A-OGS configurations and on the encoding \( \Pi; \) terms. Moreover, from Theorem 7 the same result holds under a bisimulation semantics. Further, since the LTS produced by A-OGS is deterministic, its trace semantics coincides with its bisimulation semantics. We can thus conclude as in Corollary 11.

We recall that \( \approx_\sigma \pi \) and \( \approx_\sigma \pi \) are, respectively, trace equivalence and bisimilarity between \( \Pi; \) processes in the \( \text{opLTS} \); similarly for \( \approx_\pi \) and \( \approx_\pi \) between A-OGS terms.

Corollary 11. For any \( F,F' \) we have: \( F \approx_\pi a \iff \mathcal{V}[F] \approx_\sigma \pi \mathcal{V}[F'] \) iff \( F \approx_\pi a \iff \mathcal{V}[F] \approx_\sigma \pi \mathcal{V}[F'] \).

Corollary 12. For any \( \lambda \)-terms \( M,N \), we have:

\[ (M) \approx_\pi (N) \iff (M) \approx_\pi (N) \iff \mathcal{V}[M] \approx_\sigma \pi \mathcal{V}[N] \iff \mathcal{V}[M] \approx_\sigma \pi \mathcal{V}[N] \].

From Theorem 5 and Corollary 8, it also follows that \( F \) and \( \mathcal{V}[F] \) are weakly bisimilar, on the union of the respective LTSs.

7 Concurrent Operational Game Semantics

In this section we explore another way to derive an exact correspondence between OGS and \( \Pi; \), by relaxing the Alternating LTS for OGS so to allow multiple terms in configurations to run concurrently. We refer to the resulting OGS as the Concurrent OGS, briefly C-OGS (we recall that A-OGS refers to the Alternating OGS of Section 4).

We introduce running terms, ranged over by \( A,B \), as finite mappings from continuation names to \( \lambda \)-terms. A concurrent configuration is a triple \( \langle A, \gamma, \phi \rangle \) of a running term \( A \), an environment \( \gamma \), and a set of names \( \phi \). Moreover, the domains of \( A \) and \( \gamma \) must be disjoint.

We extend the definition of the polarity function, considering names in the domain of both \( A \) and \( \gamma \) as Player names.

Passive and active configurations can be seen as special case of C-OGS configurations with zero and one running term, respectively. For this reason we still use \( F,G \) to range
The encoding of configurations is then defined as:

\[ \text{Corollary 14.} \]

\[ \text{Lemma 15.} \]

Figure 4 The LTS for the Concurrent OGS

The results about operational correspondence between C-OGS and Iπ are as those between A-OGS and Iπ under the opLTS.

**Theorem 13.**

1. If \( F \Rightarrow \epsilon F' \) then \( \mathcal{V}[F] \Rightarrow \pi \mathcal{V}[F'] \);
2. If \( F \overset{\ell}{\Rightarrow} \epsilon F' \) then \( \mathcal{V}[F] \overset{\ell}{\Rightarrow} \pi \mathcal{V}[F'] \);
3. The converse of (1), i.e. if \( \mathcal{V}[F] \Rightarrow \pi P \) then there is \( F' \) such that \( F \Rightarrow \epsilon F' \) and \( \mathcal{V}[F'] \Rightarrow \pi P \).
4. The converse of (2), i.e. if \( \mathcal{V}[F] \overset{\ell}{\Rightarrow} \pi P \) then there is \( F' \) such that \( F \overset{\ell}{\Rightarrow} \epsilon F' \) and \( P \approx \pi \mathcal{V}[F'] \).

**Corollary 14.** For any C-OGS configuration \( F \) and trace \( s \), we have \( F \overset{s}{\Rightarrow} \epsilon \) iff \( \mathcal{V}[F] \overset{s}{\Rightarrow} \pi \).

From Corollary 14 and Theorem 13, we derive:

**Lemma 15.** For any \( F, F' \) we have:

over C-OGS configurations. Moreover we freely take A-OGS configurations to be C-OGS configurations, and conversely for C-OGS configurations with zero and one running term, omitting the obvious syntactic coercions. Both the running term and the environment may be empty.

We present the rules of C-OGS in Figure 4. Since there is no more distinction between passive and active configurations, a given configuration can perform both Player and Opponent actions. Notice that only Opponent can add a new term to the running term \( A \). A singleton is a configuration \( F \) whose \( \pi \)-support has only one element (that is, in C-OGS, \( F \) is either of the form \( \langle p \Rightarrow M, \epsilon, \phi \rangle \), or \( \langle \ell, [x \Rightarrow V], \phi \rangle \), or \( \langle \ell, [p \Rightarrow (E, q)], \phi \rangle \).

In this and in the following section \( F, G \) ranges over C-OGS configurations, as reminded by the index ‘c’ in the symbols for LTS and behavioural equivalence with which \( F, G \) appear (e.g., \( \Rightarrow \)).

### 7.1 Comparison between C-OGS and Iπ

The encoding of C-OGS into Iπ is a simple adaptation of that for A-OGS. We only have to consider the new or modified syntactic elements of C-OGS, namely running terms and configurations; the encoding remains otherwise the same. The encoding of running term is:

\[ \mathcal{V}[\langle p \Rightarrow M \cdot A \rangle] \overset{\text{def}}{=} \mathcal{V}[M] \langle p \rangle \mid \mathcal{V}[A] \quad \mathcal{V}[\epsilon] \overset{\text{def}}{=} 0 \]

The encoding of configurations is then defined as:

\[ \mathcal{V}[\langle A, \gamma, \phi \rangle] \overset{\text{def}}{=} \mathcal{V}[A] \mid \mathcal{V}[\gamma] . \]

For any C-OGS configuration \( F \) and trace \( s \), we have \( F \overset{s}{\Rightarrow} \epsilon \) iff \( \mathcal{V}[F] \overset{s}{\Rightarrow} \pi . \)

From Corollary 14 and Theorem 13, we derive:

**Lemma 15.** For any \( F, F' \) we have:
To derive the full analogous of Corollary 12, we now show that, on the \( \Pi \) representation of \( \lambda \)-terms, trace equivalence is the same as bisimilarity. This result needs a little care: it is known that on deterministic LTSs bisimilarity coincides with trace equivalence. However, the behaviour of the \( \Pi \) representation of a C-OGS configuration need not be deterministic, because there could be multiple silent transitions as well as multiple output transitions (for instance, in C-OGS rule OQ may be applicable to different terms).

**Lemma 16.** For any \( M, N \) we have: \( \mathcal{V}[M] \equiv \mathcal{V}[N] \) iff \( \mathcal{V}[M] \approx \mathcal{V}[N] \).

The proof uses the ‘bisimulation up-to context and up-to \( (\varepsilon \approx \Pi) \)’ technique. We can finally combine Lemmas 16 and 15 to derive that the C-OGS and I\( \pi \) semantics of \( \lambda \)-calculus coincide, both for traces and for bisimilarity.

**Corollary 17.** For all \( M, N \) we have: \( \langle M \rangle \equiv \langle N \rangle \) iff \( \mathcal{V}[M] \approx \mathcal{V}[N] \).

More details on proofs may be found in Appendix E.

### 7.2 Tensor Product

We now introduce a way of combining configurations, which corresponds to the notion of tensor product of arenas and strategies in (denotational) game semantics.

**Definition 18.** Two concurrent configurations \( F, G \) are said to be compatible if their polarity functions \( \text{pol}_F, \text{pol}_G \) are compatible — that is, for all \( a \in \text{dom}(\text{pol}_F) \cap \text{dom}(\text{pol}_G) \), we have \( \text{pol}_F(a) = \text{pol}_G(a) \).

**Definition 19.** For compatible configurations \( F = \langle A, \gamma, \phi \rangle \) and \( G = \langle B, \delta, \phi' \rangle \), the tensor product \( F \otimes G \) is defined as \( F \otimes G \triangleq \langle A \cdot B, \gamma \cdot \delta, \phi \cup \phi' \rangle \)

The polarity function of \( F \otimes G \) is then equal to \( \text{pol}_F \cup \text{pol}_G \), and \( \mathcal{V}[F_1 \otimes F_2] \equiv \mathcal{V}[F_1] \mid \mathcal{V}[F_2] \), where \( \equiv \) is the standard structural congruence of \( \pi \)-calculi. In the following, we write \( \text{inter}(s_1, s_2) \) for the set of traces obtained from an interleaving of the elements in the sequences \( s_1 \) and \( s_2 \).

**Lemma 20.** Suppose \( F_1, F_2 \) are compatible concurrent configurations. The set of traces generated by \( F_1 \otimes F_2 \) is the union of the sets of interleaving \( \text{inter}(s_1, s_2) \), for \( F_1 \xrightarrow{s_1} \varepsilon \) and \( F_2 \xrightarrow{s_2} \varepsilon \).

The tensor product of A-OGS configurations is defined similarly, with the additional hypothesis that at most one of the two configurations is active, in order for their tensor product to be a valid A-OGS configuration. The details can be found in Appendix H.2.

### 7.3 Up-to techniques for games

We introduce up-to techniques for C-OGS, which allow, in bisimulation proofs, to split two C-OGS configurations into separate components and then to reason separately on these. These up-to techniques are directly imported from I\( \pi \). Abstract settings for up-to techniques have been developed, see \([36,37]\); we cannot however derive the OGS techniques from them because these setting are specific to first-order LTS (i.e., CCS-like, without binders within actions).
The new techniques are then used to prove that C-OGS and A-OGS yield the same semantics on λ-terms; a further application is in Section 7.5, discussing eager normal-form bisimilarity.

A relation \( \mathcal{R} \) on configuration is well-formed if it relates configurations with the same polarity function. Below, all relations on configurations are meant to be well formed. Given a well-formed relation \( \mathcal{R} \) we write:

- \( \mathcal{R}^1 \) for the relation \( \{(F_1, F_2) : \exists G \text{ s.t. } F_i = F'_i \otimes G (i = 1, 2) \text{ and } F'_1 \mathcal{R} \mathcal{R}^1 \} \).
- \( \mathcal{R}^{1\star} \) for the reflexive and transitive closure of \( \mathcal{R}^1 \). Thus from \( F_1 \mathcal{R} G_1 \) and \( F_2 \mathcal{R} G_2 \) we obtain \( (F_1 \otimes F_2) \mathcal{R}^{1\star} (G_1 \otimes F_2) \mathcal{R}^{1\star} G_1 \otimes G_2 \).
- \( \Rightarrow_{\varepsilon} \mathcal{R}^{1\star} \varepsilon \Leftarrow \) for the closure of \( \mathcal{R}^{1\star} \) under reductions. That is, \( F_1 \Rightarrow_{\varepsilon} \mathcal{R}^{1\star} \varepsilon \Leftarrow F_2 \) holds if there are \( F'_i, i = 1, 2 \) with \( F_i \Rightarrow_{\varepsilon} F'_i \) and \( F'_1 \mathcal{R}^{1\star} F'_2 \). (As \( \Rightarrow_{\varepsilon} \) is reflexive, we may have \( F_i = F'_i \).)

**Definition 21.** A relation \( \mathcal{R} \) on configurations is a bisimulation up-to reduction and composition if whenever \( F_1 \mathcal{R} F_2 \):

1. if \( F_1 \Rightarrow_{\varepsilon} F'_1 \) then there is \( F'_2 \) such that \( F_2 \Rightarrow_{\varepsilon} F'_2 \) and \( F'_1 \Rightarrow_{\varepsilon} \mathcal{R}^{1\star} \varepsilon \Leftarrow F'_2 \);
2. the converse, on the transitions from \( F_2 \).

A variant of the technique in Definition 21, where the bisimulation game is played only on visible actions at the price of being defined on singleton configurations, is presented in Appendix F and used in Section 7.5 to lift any eager normal form bisimulation to an OGS 'bisimulation up-to'.

**Theorem 22.** If \( \mathcal{R} \) is bisimulation up-to reduction and composition then \( \mathcal{R} \subseteq \approx_{\varepsilon} \).

The theorem is proved by showing that the \( \mathcal{I}\pi \) image of \( \mathcal{R} \) is a bisimulation up-to context and up-to \( (\varepsilon \geq_{\varepsilon} \approx_{\varepsilon}) \), and appealing to Lemma 15(2). See Appendix F for details.

**Remark 23.** Results such as Corollary 17 and Theorem 22 might suggest that the equality between two configurations implies the equality of all their singleton components. That is, if \( F \approx_{\varepsilon} G \), with \([p \mapsto M]\) part of \( F \) and \([p \mapsto N]\) part of \( G \), then also \( \{[p \mapsto M]\} \approx_{\varepsilon} \{[p \mapsto N]\} \).

A counterexample is given by the configurations

\[
F_1 \overset{\text{def}}{=} \langle [p_1 \mapsto M] \cdot [p_2 \mapsto \Omega] \rangle \quad \text{where } M \overset{\text{def}}{=} (\lambda x. \Omega)(x \lambda y. \Omega)
\]

\[
F_2 \overset{\text{def}}{=} \langle [p_1 \mapsto \Omega] \cdot [p_2 \mapsto M] \rangle
\]

Intuitively the reason why \( F_1 \approx_{\varepsilon} F_2 \) is that \( M \) can produce an output (along the variable \( x \)), but an observer will never obtain access to the name at which \( M \) is located \((p_1 \text{ or } p_2)\). That is, the term \( M \) can interrogate \( x \), but it will never answer, neither at \( p_1 \) nor at \( p_2 \).

### 7.4 Relationship between Concurrent and Alternating OGS

In Section 6 we have proved that the trace-based and bisimulation-based semantics produced by A-OGS and by \( \mathcal{I}\pi \) under the opLTS coincide. In Section 7.1 we have obtained the same result for C-OGS and \( \mathcal{I}\pi \) under the ordinary LTS. In this section we develop these results so to conclude that all such equivalences for \( \lambda \)-terms actually coincide. In other words, the equivalence induced on \( \lambda \)-terms by their representations in OGS and \( \mathcal{I}\pi \) is the same, regardless of whether we adopt the alternating or concurrent flavour for OGS, the opLTS or the ordinary LTS in \( \mathcal{I}\pi \), a trace or a bisimulation semantics. For this, in one direction we show that the trace semantics induced by C-OGS implies that induced by A-OGS (Lemma 63). In the opposite direction, we lift a bisimulation over the alternating LTS on singleton configurations into a bisimulation up-to composition over the concurrent LTS.
Lemma 24. If $F_1, F_2$ are $\Lambda$-OGS singleton configurations and $F_1 \approx_a F_2$, then also $F_1 \approx_c F_2$.

Details may be found in Appendix G. From Corollaries 12, 17, and Lemmas 63, 24 we can thus conclude.

Corollary 25. For any $\lambda$-terms $M, N$, the following statements are the same: $\langle M \rangle \approx_a \langle N \rangle$; $\langle M \rangle \approx_c \langle N \rangle$; $\forall V[M] \approx_{\sigma_\pi} \forall V[N]$; $\forall V[M] \approx_{\sigma_\pi} \forall V[N]$; $\forall V[N]$; $\forall V[M] \approx_a \forall V[N]$.

7.5 Eager Normal Form Bisimulations

We recall Lassen’s eager normal-form (enf) bisimilarity [28].

Definition 26. An enf-bisimulation is a triple of relation on terms $R_M$, values $R_V$, and evaluation contexts $R_\pi$ that satisfies:

- $M_1 R_M M_2$ if either:
  - both $M_1, M_2$ diverge;
  - $M_1 \Downarrow E_1[xV_1]$ and $M_2 \Downarrow E_2[xV_2]$ for some $x$, values $V_1, V_2$, and evaluation contexts $E_1, E_2$ with $V_1 R_V V_2$ and $K_1 R_\pi K_2$;
  - $M_1 \Downarrow V_1$ and $M_2 \Downarrow V_2$ for some values $V_1, V_2$ with $V_1 R_V V_2$.

- $V_1 R_V V_2$ if $V_1 \pi M_1 \Downarrow V_2 x$ for some fresh $x$;
- $K_1 R_\pi K_2$ if $K_1[x] R_M K_2[x]$ for some fresh $x$.

The largest enf-bisimulation is called enf-bisimilarity.

From Corollary 25 and existing results in the $\pi$-calculus [11] we can immediately conclude that the semantics on $\lambda$-terms induced by OGS (Alternating or Concurrent) coincides with enf-bisimilarity (i.e., Lassen’s trees).

In this section we show a direct proof of the result, for C-OGS bisimilarity, as an example of application of the up-to composition technique for C-OGS.

Terms, Values, and Evaluations contexts can be directly lifted to singleton concurrent configurations, meaning that we can transform a relation on terms, values, and contexts $R$ into a relation $\widehat{R}$ on singleton concurrent configurations in the following way:

- If $(M_1, M_2) \in R$ then $((p \mapsto M_1), \varepsilon, \phi, (p \mapsto M_2), \varepsilon, \phi)) \in \widehat{R}$, with $\phi = fV(M_1, M_2) \cup \{p\}$
- If $(V_1, V_2) \in R$ then $((x \mapsto V_1), \phi, (x \mapsto V_2), \phi)) \in \widehat{R}$, with $\phi = fV(V_1, V_2) \cup \{x\}$
- If $(E_1, E_2) \in R$ then $((p \mapsto (E_1, q), \phi, (p \mapsto (E_2, q), \phi)) \in \widehat{R}$, with $\phi = fV(E_1, E_2) \cup \{p, q\}$.

Theorem 27. Taking $(R_M, R_V, R_\pi)$ an enf-bisimulation, then $(\widehat{R_M} \cup \widehat{R_V} \cup \widehat{R_\pi})$ is a singleton bisimulation up-to composition in C-OGS (as defined in Appendix F).

Proof. Taking $F_1, F_2$ two singleton configurations s.t. $(F_1, F_2) \in \widehat{R_M}$, then we can write $F_i$ as $(p \mapsto M_i, \phi)$ for $i = 1, 2$, such that $(M_i, M_2) \in R$. suppose that $F_1 \Rightarrow_{\xi^{\ell}} F_1'$, with $\ell$ a Player action. We only present the Player Question case, which is where ‘up-to composition’ is useful. So writing $\ell$ as $\bar{x}(y, q), F_1'$ can be written as $(y \mapsto V_1) \cdot (q \mapsto (E_1, p))$, so that $M_1 \Rightarrow_{\xi^{\ell}} E_1[xV_1]$.

As $(M_1, M_2) \in R$, there are $E_2, V_2$ s.t. $M_2 \Rightarrow_{\xi^{\ell}} E_2[xV_2]$. $(V_1, V_2) \in R$ and $(E_1, E_2) \in R$.

Hence $F_2 \Rightarrow_{\xi^{\ell}} F_2'$ with $F_2' = (y \mapsto V_2) \cdot (q \mapsto (E_2, p))$.

Finally, $((y \mapsto V_1), (y \mapsto V_2)) \in \widehat{R}$ and $((q \mapsto (E_1, p)), (q \mapsto (E_2, p))) \in \widehat{R}$, so that $(F_1', F_2') \in \widehat{R}^\ast$.

The case of passive singleton configurations is proved in a similar way.
8 Compositionality of OGS via $I\pi$

We now present some compositionality results about C-OGS and A-OGS, that can be proved via the correspondence between OGS and $I\pi$.

Compositionality of OGS amounts to compute the set of traces generated by $\langle M\{V/x\}\rangle$ from the set of traces generated by $\langle M \rangle$ and $\langle [x \mapsto V]\rangle$. This is the cornerstone of (denotational) game semantics, where the combination of $\langle M \rangle$ and $\langle [x \mapsto V]\rangle$ is represented via the so-called ‘parallel composition plus hiding’. This notion of parallel composition of two processes $P, Q$ plus hiding over a name $x$ is directly expressible in $I\pi$ as the process $\nu x(P \mid Q)$. Precisely, suppose $F, G$ are two configurations that agree on their polarity functions, but on a name $x$; then we write

$$\nu x(F \mid G)$$

for the $I\pi$ process $\nu x(V[F] \mid V[G])$, the parallel composition plus hiding over $x$. To define this operation directly at the level of OGS, we would have to generalize its LTS, allowing internal interactions over a name $x$ used both in input and in output.

As the translation $V$ from OGS configurations into $I\pi$ validates the $\beta_o$ rule, we can prove that the behaviour of $\langle M\{V/x\}\rangle$ (e.g., its set of traces) is the same as that of the parallel composition plus hiding over $x$ of $\langle M \rangle$ and $\langle [x \mapsto V]\rangle$. We use the notation $(\ast)$ above to express the following two results.

- **Theorem 28.** For $\forall a \in \{\approx_{\pi}, \approx_{\sigma}, \approx_{\nu}, \approx_{\tau}\}$, we have $V[\{p \mapsto M\{V/x\}\}, \gamma, \phi \cup \phi'] \blacktriangleright_{\pi} \nu x(\{p \mapsto M[\cdot], \varepsilon, \phi \cup \{x\}\} \mid \{\gamma : [x \mapsto V], \phi' \cup \{x\}\})$

- **Corollary 29.** For any trace $s$:
  1. $\langle M\{V/x\}, p, \gamma, \phi \cup \phi'\rangle \xrightarrow{\approx_{\pi}} \iff \nu x(\{p \mapsto M[\cdot], \varepsilon, \phi \cup \{x\}\} \mid \{\gamma : [x \mapsto V], \phi' \cup \{x\}\})
  2. $\langle [p \mapsto M\{V/x\}], \gamma, \phi \cup \phi'\rangle \xrightarrow{\approx_{\pi}} \iff \nu x(\{p \mapsto M[\cdot], \varepsilon, \phi \cup \{x\}\} \mid \{\gamma : [x \mapsto V], \phi' \cup \{x\}\})

Other important properties that we can import in OGS from $I\pi$ are the congruence properties for the A-OGS and C-OGS semantics. We report the result for $\approx_{\sigma}$; the same result holds for $\approx_{\nu}, \approx_{\tau}, \approx_{\tau}$.

- **Theorem 30.** If $\langle M \rangle \approx_{\sigma} \langle N \rangle$ then for any $A_v$ context $C$, $\langle C[M] \rangle \approx_{\sigma} \langle C[N] \rangle$.

The result is obtained from the congruence properties of $I\pi$, Corollary 25, and the compositionality of the encoding $V$.

9 Related and Future Work

Analogies between game semantics and $\pi$-calculus, as semantic frameworks in which names are central, have been pointed out from the very beginning of game semantics. In the pioneering work [19], the authors obtain a translation of PCF terms into the $\pi$-calculus from a game model of PCF by representing strategies (the denotation of PCF terms in the game model) as processes of the $\pi$-calculus. The encoding bears similarities with Milner’s, though they are syntactically rather different (“it is clear that the two are conceptually quite unrelated”, [19]). The connection has been developed in various papers, e.g., [8, 14, 17, 18, 48]. Milner’s encodings into the $\pi$-calculus have sometimes been a source of inspiration in the definition of the game semantics models (e.g., transporting the work [19], in call-by-name, onto call-by-value [18]). In [48], a typed variant of $\pi$-calculus, influenced by differential linear logic [13], is introduced as a metalanguage to represent game models. In [9], games are defined using algebraic operations on sets of traces, and used to prove type soundness of a
simply-typed call-by-value $\lambda$-calculus with effects. Although the calculus of traces employed
is not a $\pi$-calculus (e.g., being defined from operators and relations over trace sets rather
than from syntactic process constructs), there are similarities, which would be interesting to
investigate.

Usually in the above papers the source language is a form of $\lambda$-calculus, that is interpreted
into game semantics, and the $\pi$-calculus (or dialects of it) is used to represent the resulting
strategies and games. Another goal has been to shed light into typing disciplines for $\pi$-calculus
processes, by transplanting conditions on strategies such as well-bracketing and innocence
into appropriate typings for the $\pi$-calculus (see, e.g., [6, 47]).

In contrast with the above works, where analogies between game semantics and $\pi$-calculus
are used to better understand one of the two models (i.e., explaining game semantics in
terms process interactions, or enhancing type systems for processes following structures in
game semantics), in the present paper we have carried out a direct comparison between the
two models (precisely OGS and $I\pi$). For this we have started from the (arguably natural)
representations of the $\lambda$-calculus into OGS and $I\pi$ (the latter being Milner’s encodings). Our
goal was understanding the relation between the behaviours of the terms in the two models,
and transferring techniques and results between them.

Technically, our work builds on [10, 11], where a detailed analysis of the behaviour of
Milner’s call-by-value encoding is carried out using proof techniques for $\pi$-calculus based
on unique-solution of equations. Various results in [10, 11] are essential to our own (the
observation that Milner’s encodings should be interpreted in $I\pi$ rather than the full $\pi$-calculus
is also from [10, 11]).

Bisimulations over OGS terms, and tensor products of configurations, were introduced
in [29], in order to provide a framework to study compositionality properties of OGS. In
our case the compositionality result of OGS is derived from the correspondence with the
$\pi$-calculus. In [39], a correspondence between an i/o typed asynchronous $\pi$-calculus and
a computational $\lambda$-calculus with channel communication is established, using a common
categorical model (a compact closed Freyd category). It would be interesting to see if our
current operational game model could be equipped with this categorical semantics.

Normal form (or open) bisimulations [28, 44], as game semantics, manipulate open terms,
and sometimes make use of environments or stacks of evaluation contexts (see e.g., the
recent work [7], where a fully abstract normal-form bisimulation for a $\lambda$-calculus with store
is obtained).

There are also works that build game models directly for the $\pi$-calculus, i.e., [12, 25, 26, 38]
A correspondence between a synchronous $\pi$-calculus with session types and concurrent game
semantics [46] is given in [8], relating games (represented as arenas) to session types, and
strategies (defined as coincident event structures) to processes.

We have exploited the full abstraction results between OGS and $I\pi$ to transport a few
up-to techniques for bisimulation from $I\pi$ onto OGS. However, in $I\pi$ there are various other
such techniques, even a theory of bisimulation enhancements. We would like to see which
other techniques could be useful in OGS, possibly transporting the theory of enhancements
itself.
References


A. Additional material on \( \Pi \)

A.1 LTSs

In Figures 5 and 6 we report, respectively:
- the standard LTS of \( \Pi \);
- a compositional definition of the output-prioritised LTS.

The symbol \( \equiv_\alpha \) is used to indicate \( \alpha \)-equivalence between \( \Pi \) agents. If \( \mu = a(\overline{b}) \) then \( \overline{\mu} \) is \( \overline{\Pi(\overline{b})} \), and conversely. Rule \textbf{abs} is used to formalise, by means of an action, the ground instantiation of the parameters of an abstraction. In both tables we omit the symmetric of \textbf{parL}, called \textbf{parR}.

**Figure 5** The standard LTS for \( \Pi \)

```
alpha: \( P \equiv_\alpha P' P' \xrightarrow{\mu \cdot \rho_x} P'' \)
pre: \( \mu. P \xrightarrow{\mu \cdot \rho_x} P \)
parL: \( P \mid Q \xrightarrow{\mu \cdot \rho_x} P' \mid Q \)
```

```
res: \( P \xrightarrow{\nu \cdot \rho_x} P' \quad x \notin \mathbb{N}(\mu) \)
com: \( P \xrightarrow{\nu \cdot \rho_x} Q \xrightarrow{\mu \cdot \rho_x} Q' \mu \neq \tau, \overline{x} = \text{bn}(\mu) \)
```

```
rep: \( !\mu. P \xrightarrow{\mu \cdot \rho_x} P' \mid !\mu. P \)
```

```
abs: \( (B = (\overline{y}).Q \text{ or } (B = K \text{ and } K \xrightarrow{\text{def}} (\overline{y}).Q)) \) \( (\overline{y}).Q \equiv_\alpha (\overline{x}).P \)
```

**Figure 6** A compositional presentation of the output-prioritised LTS for \( \Pi \)

```
alpha: \( P \equiv_\alpha P' P' \xrightarrow{\mu \cdot \rho_x} P'' \)
pre: \( \text{bn}(\mu) = \overline{x} \quad \overline{x} \cap \phi = \emptyset \)
parL: \( P \mid Q \xrightarrow{\mu \cdot \rho_x} P' \mid Q \)
```

```
res: \( P \xrightarrow{\nu \cdot \rho_x} P' \quad x \notin \mathbb{N}(\mu) \)
com: \( P \xrightarrow{\nu \cdot \rho_x} Q \xrightarrow{\mu \cdot \rho_x} Q' \mu \neq \tau, \overline{x} = \text{bn}(\mu) \)
```

```
rep: \( !\mu. P \xrightarrow{\mu \cdot \rho_x} P' \mid !\mu. P \)
```

```
abs: \( K \xrightarrow{\text{def}} (\overline{y}).Q \xrightarrow{\mu \cdot \rho_x} (\overline{y}).Q \equiv_\alpha (\overline{x}).P \)
```

\( K(\overline{y}) \xrightarrow{\mu \cdot \rho_x} P' \)
A.2 The expansion relation and “up-to” techniques

For the proof of the results in this paper, we appeal to the expansion relation and to a few “up-to” techniques, that are discussed in Section 3.1 and in the following two subsections.

The expansion relation The expansion relation, $\preceq_{\pi}$ [43], is an asymmetric variant of $\approx_{\pi}$ which allows us to count the number of $\tau$-actions performed by the processes. Thus, $P \preceq_{\pi} Q$ holds if $P \approx_{\pi} Q$ but also $Q$ has at least as many $\tau$-moves as $P$. We recall that $P \xrightarrow{\mu} P'$ holds if $P \xrightarrow{\mu} P'$ or ($\mu = \tau$ and $P = P'$).

Definition 31 (expansion). A relation $\mathcal{R} \subseteq P \times P$ is an expansion if $P \mathcal{R} Q$ implies:

1. Whenever $P \xrightarrow{\mu} P'$, there exists $Q'$ s.t. $Q \xrightarrow{\mu\pi} Q'$ and $P' \mathcal{R} Q'$;
2. whenever $Q \xrightarrow{\mu} Q'$, there exists $P'$ s.t. $P \xrightarrow{\mu} P'$ and $P' \mathcal{R} Q'$.

We say that $Q$ expands $P$, written $P \preceq_{\pi} Q$, if $P \mathcal{R} Q$, for some expansion $\mathcal{R}$.

Relation $\preceq_{\pi}$ is a precongruence, and is strictly included in $\approx_{\pi}$.

Bisimulation up-to $\approx_{\pi}$

Definition 32 (bisimulation up-to $\approx_{\pi}$). A symmetric relation $\mathcal{R} \subseteq P \times P$ is a bisimulation up-to $\approx_{\pi}$ if $P \mathcal{R} Q$ and $P \xrightarrow{\mu\pi} P'$ imply that there exists $Q'$ s.t. $Q \xrightarrow{\mu\pi} Q'$ and $P' \approx_{\pi} R \approx_{\pi} Q'$.

Theorem 33. Suppose $\mathcal{R}$ is a bisimulation up-to $\approx_{\pi}$. Then $\mathcal{R} \subseteq \approx_{\pi}$.

Bisimulations up-to contexts

Definition 34 (bisimulation up-to context and up-to $\pi \preceq_{\pi}$). A symmetric relation $\mathcal{R} \subseteq P \times P$ is a bisimulation up-to context and up-to $\pi \preceq_{\pi}$ if $P \mathcal{R} Q$ and $P \xrightarrow{\mu\pi} P'$ imply that there are a static context $C_{ctx}$ and processes $P'$ and $Q'$ s.t. $P'' \pi \preceq_{\pi} C_{ctx}[P']$, $Q \xrightarrow{\mu\pi} \approx_{\pi} C_{ctx}[Q']$ and $P' \mathcal{R} Q'$.

The soundness of the above technique relies on the following lemma.

Lemma 35. Suppose $\mathcal{R}$ is a bisimulation up-to context and up-to $\pi \preceq_{\pi}$, $(P, Q) \in \mathcal{R}$ holds and $C_{ctx}$ is a static context. If $C_{ctx}[P] \xrightarrow{\mu_1 \pi} \cdots \xrightarrow{\mu_n \pi} P_1$, $n \geq 0$, then there are a static context $C'$ and processes $P'$ and $Q'$ s.t. $P_1 \pi \preceq_{\pi} C''[P']$, $C_{ctx}[Q] \xrightarrow{\mu_1} \cdots \xrightarrow{\mu_n} \pi \preceq_{\pi} C''[Q']$ and $P' \mathcal{R} Q'$.

Theorem 36. If $\mathcal{R}$ is a bisimulation up-to context and up-to $\pi \preceq_{\pi}$, then $\mathcal{R} \subseteq \approx_{\pi}$.

Proof. By showing that the relation $S$ containing all pairs $(P, Q)$ such that

for some static context $C_{ctx}$ and processes $P', Q'$, it holds that $P \pi \preceq_{\pi} C_{ctx}[P'], Q \pi \preceq_{\pi} C_{ctx}[Q']$ and $P' \mathcal{R} Q'$, is a bisimulation. See [40] for details (adapting the proof to $1\pi$ is straightforward).

We now consider a refinement of the above proof technique, namely bisimulation up-to context and up-to $\pi \preceq_{\pi}$.

Definition 37. A static context $C_{ctx}$ cannot interact with a process $P$ if there is no name that appears free both in the context and in the process and with opposite polarities (that is, either in input position in the context and output position in the process, or the converse).
If $C_{\text{ctx}}$ cannot interact with $P$, whenever $C_{\text{ctx}}[P] \rightarrow Q$ then the interaction is either internal to $C_{\text{ctx}}$ or internal to $P$. Moreover, the property is invariant for reduction; i.e., if $Q = C'[P']$ where $C'$ is the derivative of $C_{\text{ctx}}$ and $P'$ the derivative of $P$, then also $C'$ cannot interact with $P'$.

\textbf{Definition 38.} [bisimulation up-to context and up-to $(\approx_\pi, \approx_\pi)$] A symmetric relation $R \subseteq P \times P$ is a bisimulation up-to context and up-to $(\approx_\pi, \approx_\pi)$ if $P \mathrel{R} Q$ implies:

1. If $P \xrightarrow{\mu} P''$ then that there are a static context $C_{\text{ctx}}$ and processes $P'$ and $Q'$ s.t. $P'' \approx_\pi C_{\text{ctx}}[P'], Q \xrightarrow{\mu} \approx_\pi C_{\text{ctx}}[Q']$ and $P' \mathrel{R} Q'$, and, moreover, $C_{\text{ctx}}$ cannot interact with $P$ or $Q$;

2. If $P \longrightarrow P''$ then there are a static context $C_{\text{ctx}}$ and processes $P'$ and $Q'$ s.t. $P'' \approx_\pi C_{\text{ctx}}[P'], Q \longrightarrow \approx_\pi C_{\text{ctx}}[Q']$ and $P' \mathrel{R} Q'$.

\textbf{Lemma 39.} Suppose $R$ is a bisimulation up-to context and up-to $(\approx_\pi, \approx_\pi)$ and $C_{\text{ctx}}$ is a static context that does not interact with processes $P$ and $Q$.

1. If $C_{\text{ctx}}[P] \Rightarrow P_1$, then there are a static context $C'$ and processes $P'$ and $Q'$ s.t. $P_1 \approx_\pi C'[P'], C_{\text{ctx}}[Q] \Rightarrow \approx_\pi C'[Q']$ and $P' \mathrel{R} Q'$.

2. If $C_{\text{ctx}}[P] \xrightarrow{\tau} P_1$, then there are a static context $C'$ and processes $P'$ and $Q'$ s.t. $P_1 \approx_\pi C'[P'], C_{\text{ctx}}[Q] \xrightarrow{\tau} \approx_\pi C'[Q']$ and $P' \mathrel{R} Q'$.

\textbf{Theorem 40.} If $R$ is a bisimulation up-to context and up-to $(\approx_\pi, \approx_\pi)$, then $R \subseteq \approx_\pi$.

\textbf{Proof.} By showing that the relation $R$ containing all pairs $(P, Q)$ such that

\[
\text{for some static context } C_{\text{ctx}} \text{ and processes } P', Q' \text{ where } C_{\text{ctx}} \text{ does not interact with } P', Q' \text{ it holds that } P \approx_\pi C_{\text{ctx}}[P'], Q \approx_\pi C_{\text{ctx}}[Q'] \text{ and } P' \mathrel{R} Q'
\]

is a bisimulation. In a challenge transition $P \xrightarrow{\mu} P_1$ one distinguishes the case when $\mu$ is a silent or visible action. In both cases, one exploits Lemma 39. \hfill \blacksquare

\section*{B \hspace{1cm} Behavioural relations in the ordinary LTS and in the opLTS for $\pi$}

A behavioural relation on the ordinary LTS ‘almost always’ implies the corresponding relation on the opLTS. The only exceptions are produced by processes that are related although they exhibit different divergence behaviours (by divergence we mean the possibility of performing an infinite number of $\tau$ actions) and at the same time may perform input actions. For instance, using $\tau^\omega$ to indicate a purely divergent process (i.e., a process whose only transition is $\tau^\omega \rightarrow \tau^\omega$), the processes $a.0 \mid \tau^\omega$ and $a.0$ are bisimilar (and trace) equivalent in the ordinary LTS, but they are not so in the opLTS, since the divergence in the former process prevents observation of the input at $a$.

Indeed, a ‘bisimilarity respecting divergence’ would imply bisimilarity on the opLTS. Such ‘bisimilarity respecting divergence’, written $\approx_\pi^\omega$, is defined as $\approx_\pi$ with the additional requirement that $P \approx_\pi Q$ implies

\[
P \uparrow \text{ iff } Q \uparrow
\]

where the predicate $\uparrow$ holds for an agent $R$ if it has a divergence (an infinite path consisting of only $\tau$ actions).

\textbf{Lemma 41.} Relation $\approx_\pi^\omega$ implies $\approx_{\omega\pi}$. 
The same result applies to other relations; e.g., the ‘expansion respecting divergence’ (defined by adding the clause \(*\) above to those of the expansion relation $\mathcal{L}_\pi$) on the ordinary LTS implies expansion on the opLTS.

The results of operational correspondence between $\mathcal{I}_\pi$ and A-OGS rely on Lemmas 47-50. In the proofs of all these lemmas [10], the uses of expansion ($\mathcal{L}_\pi$) come from application of some simple algebraic laws that respect divergence (i.e., the laws do not add or remove any divergence). Indeed the laws either are valid for strong bisimilarity, or they are laws such as
\[ \nu a (a(x), P | \bar{a}(x), Q) \pi \succeq \nu a (P | Q) \]

As a consequence, all occurrences of $\mathcal{L}_\pi$ in Lemmas 47-50 can be replaced by the expansion relation defined on the opLTS ($\mathcal{L}_{\pi\lor}$). This in turn implies that the same could be done for all occurrences of $\mathcal{L}_\pi$ in Theorems 4-7 (that rely on the lemmas).

A final remark concerns the encoding of $\lambda$-terms. First we recall that the equivalence induced on call-by-value $\lambda$-terms by their encoding into $\mathcal{I}_\pi$ coincides with Lassen’s eager normal-form (enf) bisimilarity [28,45] (also recalled in Definition 26).

Theorem 42 ([11]). $\mathcal{V}[M] \approx_\pi \mathcal{V}[N]$ iff $M$ and $N$ are enf-bisimilar.

On the encoding of $\lambda$-terms, the ordinary bisimilarity $\approx_\pi$ implies the ‘bisimilarity respecting divergence’ $\approx_\pi^d$. This because, intuitively, a term $\mathcal{V}[M](p)$ is divergent iff $M$ is so in the call-by-value $\lambda$-calculus. Formally, the result is proved by rephrasing the result about the correspondence between $\approx_\pi$ and Lassen’s trees (Theorem 42) in terms of $\approx_\pi^d$ in place of $\approx_\pi$.

Again, this property boils down to the fact that the uses of $\mathcal{L}_\pi$ in Lemmas 47-50 (on which also the characterisation in terms of Lassen’s trees is built) respect divergence. Thus, using also Lemma 41, we derive the following result.

Theorem 43. For $M, N \in \Lambda$, we have: $\mathcal{V}[M](p) \approx_\pi \mathcal{V}[N](p)$ iff $\mathcal{V}[M](p) \approx_{\pi\lor} \mathcal{V}[N](p)$.

Below we report an alternative proof of the theorem above, however always relying on the variant of Lemmas 47-50 with $\mathcal{L}_{\pi\lor}$ in place of $\mathcal{L}_\pi$.

Lemma 44. If $F$ is a singleton configuration, then $\mathcal{V}[F]$ is of three possible forms:
1. $\mathcal{V}[M](p)$, for some $M, p$;
2. $q(x). \mathcal{V}[E[x]](p)$, for some $E, x, q, p$;
3. $\mathcal{V}^\ast[V](y)$, for some $V, y$ (that is, either $!y(x, q). \mathcal{V}[M](q)$ if $V = \lambda x. M$, or $y \triangleright x$ if $V = x$).

Lemma 45.

Suppose $P = P_1 | !a(b). P_2$, and $Q = Q_1 | !a(b). Q_2$, and $P \approx_\pi Q$. Then also $P_1 \approx_\pi Q_1$.

Similarly, suppose $P = P_1 | a(b). P_2$, and $Q = Q_1 | a(b). Q_2$, and $P \approx_\pi Q$. Then also $P_1 \approx_\pi Q_1$.

Theorem 46. If $F$ and $G$ are singleton configurations, then $\mathcal{V}[F] \approx_\pi \mathcal{V}[G]$ implies $\mathcal{V}[F] \approx_{\pi\lor} \mathcal{V}[G]$.

Proof. Let $R$ be the relations with all pairs $(P, Q)$ where $P$ and $Q$ are of the form
\[
\begin{align*}
P &= P_1 | \ldots | P_n \\
Q &= Q_1 | \ldots | Q_n
\end{align*}
\]
for some $n$, where all $P_i, Q_i$ are encodings of singleton configurations and for all $i$, we have $P_i \approx_\pi Q_i$. Moreover:

there is at most one $i$ for which $P_i, Q_i$ are encodings of active configurations;
in any \( P_j \) (resp. \( Q_j \)) that is the encoding of a passive configuration, the name of the initial input does not appear free in any other component \( P_{j'} \) (resp. \( Q_{j'} \)) for \( j' \neq j \).

A consequence of the two conditions above is that two distinct components \( P_i \) and \( P_j \) cannot interact. That is, if \( P \xrightarrow{\mu} P' \), then there is \( i \) such that \( P_i \xrightarrow{\mu} P_i' \) and \( P' = P_1'' \ldots P_i''' P_{i+1}''' \ldots P_j''' \ldots P_n''' \) where \( P_{i'} = P_j \) if \( j \neq i \) and \( P_{i'} = P_j' \). And similarly if \( P \xrightarrow{\mu \otimes o} P'' \).

We show that \( \mathcal{R} \) is a \( \preceq_{\text{bisim}} \) (the bisimulation up-to expansion is sound for bisimilarity in any LTS). In the proof below, Lemmas 49 and 50 actually refer to the versions of the lemmas mentioned above, with \( \preceq_{\text{bisim}} \) in place of \( \preceq_{\gamma} \).

Suppose \( P \xrightarrow{\mu \otimes o} P'' \). The action originates from some \( P_i \) alone, say \( P_i \xrightarrow{\mu \otimes o} P_i'' \). If \( \mu \) is a \( \tau \) action, then also \( P_i \xrightarrow{\mu} P_i' \); and since \( P_i \approx_{\gamma} Q_i \), we have \( Q_i \Rightarrow Q_i' \) with \( P_i'' \approx_{\gamma} Q_i' \).

Moreover, also \( Q_i \Rightarrow_{\otimes o} Q_i' \) — and the corresponding transition from \( Q \).

Using Lemma 49 we infer that, up-to expansion, the derivative processes \( P_i' \) and \( Q_i' \) can be rewritten into processes that fit the definition of \( \mathcal{R} \).

For the case when \( \mu \) is an output, one reasons similarly, possibly also using Lemma 45 when \( \mu \) is of the form \( \pi(z, q) \).

Finally, if \( P \xrightarrow{\mu \otimes o} P'' \) and \( \mu \) is an input, then \( P \) is input reactive. From the conditions in the definition of \( \mathcal{R} \) we infer that \( Q \) is input reactive too. Then we can conclude, reasoning as in the previous cases, but this time using Lemma 50 with \( \gamma \) a singleton. □

Theorem 43 is then a corollary of Theorem 46.

### C Explanation of the A-OGS LTS

This appendix contains some additional explanations on the rules of the A-OGS LTS. To help the reader, we first recall the rules (also presented in the main text as Figure 1):

\[
\begin{align*}
\text{(PT)} & \quad \langle M, p, \gamma, \phi \rangle \xrightarrow{\gamma} a \quad \langle N, p, \gamma, \phi \rangle \quad \text{when } M \xrightarrow{\gamma} N \\
\text{(PA)} & \quad \langle V, p, \gamma, \phi \rangle \xrightarrow{\phi(x)_a} \langle \gamma \cdot [x \mapsto V], \phi \uplus \{x\} \rangle \\
\text{(PQ)} & \quad \langle E[xV], p, \gamma, \phi \rangle \xrightarrow{\phi(x)_a} \langle \gamma \cdot [y \mapsto V] \cdot [q \mapsto (E,p)], \phi \uplus \{y,q\} \rangle \\
\text{(OA)} & \quad \langle \gamma \cdot [q \mapsto (E,p)], \phi \rangle \xrightarrow{\phi(x)_a} \langle E[x], p, \gamma, \phi \uplus \{x\} \rangle \\
\text{(OQ)} & \quad \langle \gamma, \phi \rangle \xrightarrow{\phi(x)_a} \langle V y, p, \gamma, \phi \uplus \{y,p\} \rangle \quad \text{when } \gamma(x) = V \\
\text{(IOQ)} & \quad \langle ? \mapsto M, \phi \rangle \xrightarrow{\phi(p)_a} \langle M, p, \epsilon, \phi \uplus \{p\} \rangle \\
\end{align*}
\]

The term of an active configuration determines the next transition to perform. First the term needs to be reduced, using the rule (PT). When the term is a value \( V \), a Player Answer (PA) is performed, providing a fresh variable \( x \) to Opponent, while \( V \) is stored in \( \gamma \) at position \( x \). Freshness is enforced using the disjoint union \( \uplus \). When the term is a callback \( E[xV] \), with \( p \) the current continuation name, a Player Question (PQ) at \( x \) is performed, providing two fresh names \( y, q \) to Opponent, while storing \( V \) at \( y \) and \( (E,p) \) at \( q \) in \( \gamma \).

On passive configurations, Opponent has the choice to perform different actions. It can perform an Opponent Answer (OA), by interrogating an evaluation context \( E \) stored in \( \gamma \). For this, Opponent provides a fresh variable \( x \) that is plugged into the hole of \( E \), while the continuation name \( q \) associated to \( E \) in \( \gamma \) is restored. Opponent may also perform an Opponent Question (OQ), by interrogating a value \( V \) stored in \( \gamma \). For this, Opponent provides a fresh variable \( y \) as an argument to \( V \).
D Auxiliary results for Section 6

We present results that are needed to establish the operational correspondence between OGS
and π, studied in Section 6.

We begin discussing an optimisation O of the encoding of call-by-value λ-calculus.
Following Durier et al.'s [11], we sometimes use O to simplify proofs. We report the
full definition of O; for convenience, we also list the definition of O on OGS environments
and configurations, thought is the same as that for the initial encoding V — just replacing V
with O in Figure 3).

The encoding O is obtained from the initial one V by inlining the encoding and performing
a few (deterministic) reductions, at the price of a more complex definition. Precisely, in the
encoding of application some of the initial communications are removed, including those with
which a term signals to have become a value. To achieve this, the encoding of an application
goes by a case analysis on the occurrences of values in the subterms.

We begin with a few results from [10, 11] that are needed to reason about the optimised
encoding O. The first is about the correctness of the optimisation, established by algebraic
reasoning [10, 11]; its extension to encodings of configurations is straightforward.

Lemma 47 (correctness of the optimisation). For all M, it holds that \( V[M] \equiv O[M] \).
Similarly, for all Λ-OGS configurations F, it holds that \( V[F] \equiv O[F] \).

Lemma 48. We have:

\[
O[E[xV]](p) \equiv O'(z, q). \langle O^*[V] \rangle | q(y). \langle O[E[y]] \rangle (p). 
\]

The proof [10, 11] goes by induction on the evaluation context E.

The following lemma [10, 11] uses Lemma 48 to establish the shape of the possible
transitions that a term \( O[M] \) can perform.

Lemma 49. For any \( M \in \Lambda \) and \( p \), process \( O[M] \) has exactly one immediate transition,
and exactly one of the following clauses holds:

1. \( O[M](p) \xrightarrow{\pi(y)} P \) and \( M \) is a value, with \( P = O^*[M](y) \);
2. \( O[M](p) \xrightarrow{\pi(z, q)} P \) and \( M \) is of the form \( E[xV] \), for some \( E, x, V \), with
   \[
   P \equiv O^*[V](z) | q(y). \langle O[E[y]] \rangle (p),
   \]
   and moreover \( z \) is not free in \( q(y). \langle O[E[y]] \rangle (p) \) whereas \( q \) is not free in \( O^*[V](z) \);
3. \( O[M](p) \xrightarrow{T} P \) and there is \( N \) with \( M \rightarrow \gamma N \) and \( P \equiv O[N](p) \).

We now move to reasoning about the behaviour of the encoding of configurations. Two
key lemmas are the following ones; they are derived from Lemmas 48 and 49.

Lemma 50. If \( O[\gamma] \xrightarrow{\mu} P \) and \( \mu \) is an input action, then we have three possibilities:

1. \( \mu \) is an input \( g(x, q) \) and \( \gamma = [y \rightarrow \lambda x. M] \cdot \gamma' \), and \( P \equiv O[M](q) | O[\gamma] \);
2. \( \mu \) is an input \( g(x, q) \) and \( \gamma = [y \rightarrow z] \cdot \gamma' \), and \( P \equiv O[z](q) | O[\gamma] \);
3. \( \mu \) is an input \( q(x) \) and \( \gamma = [g \rightarrow (E, p)] \cdot \gamma' \), and \( P \equiv O[E[x]](p) | O[\gamma] \).

Lemma 51. \( O^*[V](y) \xrightarrow{y(z, q)} O[V, x](q) | O^*[V](y) \)

In Lemma 51, the occurrence of \( \approx \) cannot be replaced by \( \equiv \); for instance, if \( V = \lambda x. M \) then, using Lemma 49, we can infer that the derivative process is in the relation \( \equiv \) with
Encoding of $\lambda$-terms:

\[
\begin{align*}
O[xV] & \overset{\text{def}}{=} (p) \, \exists (z, q). (O^*[V](z) \mid q \triangleright p) \\
O[(\lambda x. M)V] & \overset{\text{def}}{=} (p) \, \nu y, w (O^{*[\lambda x. M]}(y) \mid O^*[V](w) \mid \overline{\nu}(w', r'). (w' \triangleright w \mid r' \triangleright p)) \\
O[VM] & \overset{\text{def}}{=} (p) \, \nu y (O^*[V](y) \mid \nu r (O^*[M](r) \mid r(w). \overline{\nu}(w', r'). (w' \triangleright w \mid r' \triangleright p))) \quad (\ast) \\
O[MV] & \overset{\text{def}}{=} (p) \, \nu q (O^*[M](q) \mid q(y). \nu w (O^*[V](w) \mid \overline{\nu}(w', r'). (w' \triangleright w \mid r' \triangleright p))) \quad (\ast) \\
O[MN] & \overset{\text{def}}{=} (p) \, \nu q (O^*[M](q) \mid q(y). \nu r (O^*[N](r) \mid r(w). \overline{\nu}(w', r'). (w' \triangleright w \mid r' \triangleright p))) \quad (\ast\ast) \\
O[V] & \overset{\text{def}}{=} (p) \, \overline{\nu}(y). O^*[V](y)
\end{align*}
\]

where in the rules marked (\ast), $M$ is not a value, and in the rule marked (\ast\ast) $M$ and $N$ are not values; and where $O^*[V]$ is thus defined:

\[
\begin{align*}
O^{*[\lambda x. M]} & \overset{\text{def}}{=} (y) \, !y(x, q). O[M](q) \\
O^{*[x]} & \overset{\text{def}}{=} (y) \, y \triangleright x
\end{align*}
\]

Encoding of environments:

\[
\begin{align*}
O[(y \mapsto V) \cdot \gamma'] & \overset{\text{def}}{=} O^{*[V]}(y) \mid O[\gamma'] \\
O[(q \mapsto (E, p)) \cdot \gamma'] & \overset{\text{def}}{=} q(x). O[E[x]](p) \mid O[\gamma'] \\
O[e] & \overset{\text{def}}{=} 0
\end{align*}
\]

Encoding of configurations:

\[
\begin{align*}
O[(M, p, \gamma)] & \overset{\text{def}}{=} (O[M](p) \mid O[\gamma], \phi) \\
O[(\gamma)] & \overset{\text{def}}{=} (O[\gamma], \phi) \\
O[(M)] & \overset{\text{def}}{=} (O[M], \phi)
\end{align*}
\]

\section*{Figure 7} The optimised encoding
Theorem 56. If \( \mathcal{O}[F] \xrightarrow{\pi} P \) then there is \( F' \) such that \( F \rightarrow_a F' \) and \( P \sim_\pi \mathcal{O}[F'] \).

2. If \( \mathcal{O}[F] \xrightarrow{\pi} P \) then there is \( F' \) such that \( F \rightarrow_a F' \) and \( P \approx_\pi \mathcal{O}[F'] \).

We report now more details on the proof of Corollary 8. First, we recall the assertion:

Suppose \( F \) has an irredundant name-support.

For any trace \( s \), we have \( F \rightarrow_a s \) if \( \mathcal{V}[F] \rightarrow_a \pi \).
Proof. The result is first established for \( O \), and then extended to \( V \), exploiting the correctness of the optimisations in \( O \) (Lemma 47; again, in this lemma, the occurrence of \( \leq_{\pi} \) can be lifted to \( \leq_{\pi'} \)).

For \( O \), both directions are proved proceeding by induction on the length of a trace \( \ell_1, \ldots, \ell_n \). In the direction from left to right, we use Theorem 55(2). For the converse direction, we proceed similarly, this time relying on Theorem 7(2) (again, the version with \( \approx_{O,\pi} \) in place of \( \approx_{\pi} \)).

\[ \]

E Auxiliary results for Section 7.1

We report here auxiliary results needed to establish the relationship between Concurrent OGS (C-OGS) and \( I \pi \).

The two lemmas below are needed in the proofs of Theorems 13 and 22. The reasoning in their proofs is similar to that for Lemmas 52 and 53.

\[ \]

Lemma 57 (from C-OGS to \( I \pi \), strong transitions).

1. If \( F \xrightarrow{\pi} F' \), then \( O[F] \xrightarrow{\tau_{\pi}} O[F'] \).
2. If \( F \xrightarrow{\ell_{\pi}} F' \), then \( O[F] \xrightarrow{\ell_{\pi}} O[F'] \).

Lemma 58 (from \( I \pi \) to C-OGS, strong transitions).

1. If \( O[F] \xrightarrow{\pi} P \) then there is \( F' \) such that \( F \xrightarrow{\pi} F' \) and \( P \approx_{\pi} O[F'] \); 
2. If \( O[F] \xrightarrow{\ell_{\pi}} P \) then there is \( F' \) such that \( F \xrightarrow{\ell_{\pi}} F' \) and \( P \approx_{\pi} O[F'] \).

The results below are used in the proof of Lemma 16.

A process \( P \) has deterministic immediate transitions if, for any \( \mu \), whenever \( P \xrightarrow{\mu} P_1 \) and \( P \xrightarrow{\mu} P_2 \), then \( P_1 = P_2 \).

Lemma 59. Suppose \( P, Q \) have deterministic immediate transitions, \( P \approx_{\pi} Q \), and \( P \xrightarrow{\mu_{\pi}} P_1 \). Then \( Q \xrightarrow{\mu_{\pi}} Q_1 \) implies \( P_1 \approx_{\pi} Q_1 \).

Lemma below is the analogous to Lemma 45 for traces.

Lemma 60. Suppose \( P = P_1 \mid a(b).P_2 \), and \( Q = Q_1 \mid a(b).Q_2 \), and \( \langle P, \phi \rangle \approx_{\pi} \langle Q, \phi \rangle \). Then also \( \langle P_1, \phi \rangle \approx_{\pi} \langle Q_1, \phi \rangle \).

Similarly, suppose \( P = P_1 \mid a(b).P_2 \), and \( Q = Q_1 \mid a(b).Q_2 \), and \( \langle P, \phi \rangle \approx_{\pi} \langle Q, \phi \rangle \). Then also \( \langle P_1, \phi \rangle \approx_{\pi} \langle Q_1, \phi \rangle \).

We can now conclude the proof of Lemma 16. We recall the assertion:

For any \( M, N \) we have:

\[ V[M] \approx_{\pi} V[N] \text{ iff } V[M] \approx_{\pi} V[N]. \]

Proof. We have to prove that trace equivalence implies bisimilarity. We work with the optimised encoding \( O \). The relation

\[ (O[F], O[G]) : F, G \text{ are singleton with } F \approx_{\pi} G \]

is a bisimulation up-to context and up-to \( (\approx_{\pi}, \approx_{\pi}) \).

\( F \) and \( G \), as singleton, are of the forms described in Lemma 44. Moreover, to be in the relation \( \approx_{\pi} \) they must be of the same form, and, using also Lemmas 49 and 50 we deduce that they have deterministic immediate transitions.

Suppose \( F \) and \( G \) are of the form \( V[M](p) \) and \( V[N](q) \), respectively.
Consider the case $V[M](p) \xrightarrow{\tau} P$; then by Lemma 49, $P \pi \succeq V[M'](p)$, for some $M'$. Moreover, we have $V[M](p) \simeq_c V[M](p) \simeq_c V[N](q)$, and we are done.

The case when $V[M](p)$ performs an output action $\overline{\tau}(x)$ is simple, as usual using Lemma 49; we also deduce that $p \approx q$.

Suppose now $V[M](p)$ performs an output action $\overline{\tau}(z, p')$. Using Lemma 49, we have

$$V[M](p) \xrightarrow{\overline{\tau}(z, p')} \pi \succeq O^*[V](z) \mid p'(y), O[E[y]](p)$$

for some $V, y, E$. Since $V[M](p) \simeq_c V[N](q)$, and again using Lemma 49, we deduce

$$V[N](q) \Rightarrow \overline{\tau}(z, p') \pi \succeq O^*[W](z) \mid q'(y), O[E'[y]](q)$$

By Lemma 59,

$$O^*[V](z) \mid p'(y), O[E[y]](p) \simeq_c$$

$$O^*[W](z) \mid q'(y), O[E'[y]](q)$$

Applying Lemma 60 twice, we deduce

$$O^*[V](z) \simeq_c O^*[W](z)$$

and

$$q'(y), O[E'[y]](q) \simeq_c q'(y), O[E'[y]](q)$$

and we are done, up to expansion and context.

The cases when $F$ and $G$ are of a different form are similar, this time using Lemma 50. ▶

**F Auxiliary results for Section 7.3**

In Definition 21 we have introduced the technique of bisimulation up-to reduction and composition. Sometimes we do not need the ‘closure under reduction’; that is, referring to the clauses of Definition 21 we may take $F_i = F_i$, $i = 1, 2$). In this case we say that $R$ is a ‘bisimulation up-to composition’.

We now present an additional up-to technique, which is used in when relating OGS bisimilarity with eager normal form bisimilarity. When $R$ is a relation on singleton configuration, a straightforward simplification of ‘bisimulation up-to reduction and composition’ allows us to play the bisimulation game only on visible actions; the bisimulation clause becomes:

$$\text{if } F_1 \Rightarrow \overline{\tau}_c F'_1 \text{ then there is } F'_2 \text{ such that } F_2 \Rightarrow \overline{\tau}_c F'_2 \text{ and } F_1 \mathrel{R^*} F'_2.$$ 

We call this a singleton bisimulation up-to composition; its soundness is a corollary of that for bisimulation up-to reduction and composition, derived from the up-to techniques for $\pi$ and the mapping from C-OGS onto $\pi$.

▶ **Proposition 61.** If $R$ is a singleton bisimulation up-to composition, then $R \subseteq \simeq_c$.

**Proof.** Take

$$S \overset{\text{def}}{=} R \cup \{(F_1, F_2) : \text{ there are } F'_1, F'_2 \text{ with } F'_i \mathrel{R} F'_i$$

$$\text{and } F'_i \Rightarrow \overline{\tau}_c F_i \text{ for } i = 1, 2\}$$

Intuitively, $S$ is the closure of $R$ under deterministic reductions. Then $S$ is a bisimulation up-to composition. (Note: we exploit the fact that transitions of singleton configurations are deterministic.) ▶
We now report more details on the proof of Theorem 22. We recall its assertion:

(Theorem 22) If $\mathcal{R}$ is bisimulation up-to reduction and composition then $\mathcal{R} \subseteq \approx_{\pi}$.

We first need the following Lemma 62, which is a consequence of the compositionality of the mapping from C-OGS to $I\pi$.

Lemma 62. For a relation $\mathcal{R}$ on C-OGS configurations, if $(F_1, F_2) \in \mathcal{R}^\ast$, then we have $(O[F_1], O[F_2]) \in (O[\mathcal{R}])^{\text{ctx}}$.

We are now ready to show the proof of Theorem 22.

Proof. We prove the result via $I\pi$, exploiting Lemma 15(2). Here again, it is convenient to use the optimised encoding $O$.

If $\mathcal{R}$ is a relation on C-OGS configurations, then $O[\mathcal{R}]$ is the relation on $I\pi$ processes obtained by mapping each pair in $\mathcal{R}$ in the expected manner:

$$O[\mathcal{R}] \overset{\text{def}}{=} \{(O[F_1], O[F_2]) : F_1 R F_2\}$$

And, if $\mathcal{S}$ is a relation on $I\pi$ processes, then $\mathcal{S}^{\text{ctx}}$ is the closure of $\mathcal{S}$ under polyadic contexts, i.e., the set of pairs of the form $(C^{\text{ctx}}[P_1, \ldots, P_n], C^{\text{ctx}}[Q_1, \ldots, Q_n])$ where $C^{\text{ctx}}$ is a polyadic context and for each $i, P_i, Q_i$.

As a consequence of the compositionality of the mapping from C-OGS to $I\pi$, for a relation $\mathcal{R}$ on C-OGS configurations, if $(F_1, F_2) \in \mathcal{R}^\ast$, then $(O[F_1], O[F_2]) \in (O[\mathcal{R}])^{\text{ctx}}$.

Now, let $\mathcal{R}$ be the relation in the hypothesis of the theorem. We prove the theorem by showing that $O[\mathcal{R}]$ is a bisimulation up-to context and up-to $(\pi, \approx_{\pi})$ in $I\pi$, and then appealing to Theorem 40.

Suppose $O[F_1] \xrightarrow{\ell} \pi P$. Then there is $F'_1$ with $F_1 \xrightarrow{\ell} \pi F'_1$ and $P \approx_{\pi} O[F'_1]$ (this is derived from Lemma 57). Since $\mathcal{R}$ is bisimulation up-to reduction and composition and $F_1 R F_2$, there are $F'_1$, and $F'_2$, with $F'_1 \Rightarrow_{\pi} F'_2$, $F_2 \xrightarrow{\ell} \pi F'_2$, and $F'_1 \mathcal{R}^\ast F'_2$. Using Theorem 13(1) from $F'_1 \Rightarrow_{\pi} F'_2$ and the inclusion $\pi \subseteq \approx_{\pi}$, we derive $O[F'_1] \approx_{\pi} O[F'_2]$; and by Theorem 13(1), and Lemmas 47 and 57, from $F_2 \xrightarrow{\ell} \pi \Rightarrow_{\pi} F'_2$, we derive $O[F_2] \xrightarrow{\ell} \approx_{\pi} O[F'_2]$.

Finally, from $(F'_1, F'_2) \in \mathcal{R}^\ast$, we derive $(O[F'_1], O[F'_2]) \in (O[\mathcal{R}])^{\text{ctx}}$. This closes the proof, up-to polyadic contexts and bisimilarity.

The case when $O[F_1]$ makes a silent move is simpler, using expansion in place of bisimilarity.

G Auxiliary results for Section 7.4

Lemma 63. For any term A-OGS configuration, its traces in A-OGS are a subset of the traces in C-OGS, precisely, the traces with an alternation between Player and Opponent actions.

We now prove Lemma 24.

Proof. In this proof, we write $F \subseteq G$ when, as partial maps, $G$ is an extension of $F$.

Let $\mathcal{R}$ be the relation on configurations with $F \mathcal{R} G$ if:

1. $F, G$ are singleton, and
2. either
   a. there are passive $F', G'$ with $F \subseteq F'$ and $G \subseteq G'$ and $F' \approx_{\pi} G'$,
b. or there are \( p, M, N, \gamma, \delta \) with \( F = \langle [p \mapsto M], \phi \rangle, G = \langle [p \mapsto N], \phi \rangle, \) and \( \langle [p \mapsto M], \gamma, \delta \rangle \simeq_a \langle [p \mapsto N], \delta, \phi \rangle \).

We show that \( \mathcal{R} \) is a bisimulation up-to composition. We distinguish the cases in which (2.a) or (2.b) holds. First suppose \( F, R, G \) because (2.a) holds, and there are \( F', G' \) as in (2.a). Suppose

\[
F \xrightarrow{\mu} \varepsilon F_1
\]

in C-OGS. The rule can be OA or OQ. We assume it is OQ; the case for OA is similar. We thus have, for some \( p, V, x, y \)

\[
\begin{align*}
F &= \langle \gamma_1, \phi \rangle \text{ with } \gamma_1 = [x \mapsto V] \\
\mu &= x(y, p) \\
F_1 &= \langle A, \gamma_1, \phi \uplus \{y\} \rangle \text{ with } A = [p \mapsto V y]
\end{align*}
\]

with \( F \xrightarrow{\mu} \varepsilon A F_1 \). Moreover, we have \( G = \langle \delta_1, \phi \rangle \) with \( \delta_1 = [x \mapsto V'] \). Since \( F \simeq_a G \), there is \( G_1 \) such that

\[
G \xrightarrow{\mu} \varepsilon G_1 \simeq_a F_1
\]

and \( G_1 = \langle B, \delta_1, \phi \uplus \{y\} \rangle \) for \( B = [p \mapsto V y] \). We also have

\[
G \xrightarrow{\varepsilon} \varepsilon G_1
\]

This is sufficient to match (1), as \( F, R\star G_1 \); indeed we have both \( \langle \gamma_1, \phi \rangle \mathcal{R} \langle \delta_1, \phi \rangle \) and (using clause (2.b)) \( \langle A, \phi \uplus \{y\} \rangle \mathcal{R} \langle B, \phi \uplus \{y\} \rangle \).

Now the case (2.b); let \( p, \gamma, \delta, \phi \) as in (2.b). There are 3 possibilities of transitions, corresponding to rules \( P\tau, PA, PQ \). The case of \( P\tau \) is straightforward, since bisimilarity is preserved by internal actions. We only look at \( PQ \), as \( PA \) is simpler. Thus we have

\[
F = \langle [p \mapsto E[xV]], \phi \rangle, \text{ for some } E, x, V \text{ and then}
\]

\[
F \xrightarrow{\bar{x}(y, q)} \varepsilon \langle [y \mapsto V] \cdot [q \mapsto (E, p)], \phi \uplus \{y, q\} \rangle \overset{\text{def}}{=} F_1
\]

We also have

\[
F \xrightarrow{\bar{x}(y, q)} \varepsilon F_1
\]

As \( F \simeq_a G \), there are \( E', W \) such that

\[
G \implies \langle [p \mapsto E'[xW]], \phi \rangle \xrightarrow{\bar{x}(y, q)} \varepsilon G_1
\]

with \( G_1 \overset{\text{def}}{=} \langle [y \mapsto W]; [q \mapsto (E', p)], \phi \uplus \{y, q\} \rangle \) and \( F_1 \simeq_a G_1 \). We also have \( G \xrightarrow{\bar{x}(y, q)} \varepsilon G_1 \). From \( F_1 \simeq_a G_1 \), appealing to (2.a) we deduce that both \( \langle [y \mapsto V], \phi \uplus \{y\} \rangle \mathcal{R} \langle [y \mapsto W], \phi \uplus \{y\} \rangle \)

and \( \langle [y \mapsto (E, p)], \phi \uplus \{q\} \rangle \mathcal{R} \langle [q \mapsto (E', p)], \phi \uplus \{q\} \rangle \) hold. Hence \( F_1 \mathcal{R} G_1 \).

In summary, as an answer to the challenge (2), we have found \( G_1 \) such that \( G \xrightarrow{\bar{x}(y, q)} \varepsilon G_1 \)

and \( F_1 \mathcal{R} G_1 \); this closes the proof.

### H Tensor Product

In this section, we prove results about the tensor product of compatible configurations for the Concurrent, the Alternating and the Well-Bracketed LTS presented in Section 7.2. We relate the traces generated by the tensor product of two configurations to the set of interleavings of traces generated by the component configurations themselves. The result is proved by going through the \( \pi \)-calculus.
H.1 C-OGS

Two π-calculus processes $P_1, P_2$ cannot interact if there is no name that appears free in both processes and with opposite polarities (that is, in input position in one, and output position in the other). In $1\pi$, this property is invariant for transitions (the invariance is false in the full π-calculus).

▶ Lemma 64. If $P_1, P_2$ cannot interact, and $P_1 \mid P_2 \xrightarrow{\alpha} P'_1 \mid P'_2$, then also $P'_1, P'_2$ cannot interact.

▶ Lemma 65. Suppose $P_1, P_2$ cannot interact.

1. If $P_1 \mid P_2 \xrightarrow{\pi} P'_1 \mid P'_2$, then, for $i = 1, 2$, there is $s_i$ such that $P_i \xrightarrow{\pi} P'_i$, and $s \in \text{inter}(s_1, s_2)$;

2. Conversely, if for $i = 1, 2$, we have $P_i \xrightarrow{\pi} P'_i$, and $s \in \text{inter}(s_1, s_2)$, then $P_1 \mid P_2 \xrightarrow{\pi} P'_1 \mid P'_2$.

Proof. Straightforward. The interesting case is (1). Consider the sequence of one-step actions (including silent actions) that are performed to obtain the trace $P_1 \mid P_2 \xrightarrow{\pi} P'_1 \mid P'_2$.

Then tag each of these actions with $L$ or $R$ depending on whether in the derivation proof of that action, the last rule applied was $\text{parL}$ or $\text{parR}$. Then $s_1$ is obtained by collecting the subsequence of $s$ with the $\text{parL}$ tag, and similarly for $s_2$ with $\text{parR}$.

▶ Lemma 66. Taking two C-OGS configurations $F_1, F_2$, if their polarity function are compatible then $V[F_1]$ and $V[F_2]$ cannot interact.

The proof of Lemma 20 then follows from these two lemmas, Corollary 14, and the fact that $V[F_1 \otimes F_2] \equiv V[F_1] \mid V[F_2]$.

H.2 A-OGS

Definition 67. Taking $F_1, F_2$ two A-OGS configurations, with $\gamma_1, \phi_1$ their respective environment and set of names, $F_1 \otimes F_2$ is defined as:

$(M,p,\gamma_1 \cdot \gamma_2, \phi_1 \cup \phi_2)$ when one of $F_1, F_2$ is active, with $M,p$ its toplevel term and continuation name;

$(\gamma_1 \cdot \gamma_2, \phi_1 \cup \phi_2)$ when $F_1, F_2$ are passive;

We can then state the adaptation of Lemma 20 to A-OGS. We write $\text{inter}_a(s_1, s_2)$ for the subset of $\text{inter}(s_1, s_2)$ formed by alternated traces.

Lemma 68. Suppose $F_1, F_2$ are compatible A-OGS configurations, and at most one of the two is active. Then the set of traces generated by $F_1 \otimes F_2$ is equal to the set of interleavings $\text{inter}_a(s_1, s_2)$, with $F_1 \xrightarrow{\alpha} a$ and $F_2 \xrightarrow{\alpha} a$.

To prove Lemma 68, we use the correspondence between A-OGS and $\Rightarrow_{\alpha}$ the output-prioritised LTS introduced at the end of Section 6.1, via the following Lemma.

Lemma 69. Suppose $F_1, F_2$ are compatible configurations, and at most one of the two is active.

1. If $O[F_1] \mid O[F_2] \xrightarrow{\alpha} F_1 \mid P_2$, then, for $i = 1, 2$, there are traces $s_i$ such that $O[F_i] \xrightarrow{\alpha} F_i$, and $s \in \text{inter}_a(s_1, s_2)$.

2. Conversely, if for $i = 1, 2$, we have $O[F_i] \xrightarrow{\alpha} F_i$, $s \in \text{inter}_a(s_1, s_2)$, then also $O[F_1] \mid O[F_2] \xrightarrow{\alpha} F_1 \mid P_2$. 

Games, mobile processes, and functions

\[
E := EM \mid [v] \quad \text{evaluation contexts}
\]

\[
V := \lambda x. M \mid v \quad \text{values}
\]

\[
\gamma := [v \mapsto V] \cdot \gamma' \mid [q \mapsto (E,p)] \cdot \gamma' \mid [x \mapsto M] \cdot \gamma' \mid \varepsilon \quad \text{environments}
\]

\[
F := \langle L, p, \gamma, \phi \rangle \mid \langle \gamma, \phi \rangle \mid \langle \bar{x} \mapsto \phi \rangle \cdot M \quad \text{configurations}
\]

\[
L := M \mid E[vM] \mid v \quad \text{extended } \lambda\text{-terms}
\]

**Figure 8** Grammars for call-by-name A-OGS configurations

**Proof.** Proof of (1). We know

\[O[F_1] \mid O[F_2] \xRightarrow{\sigma} P_1 \mid P_2\]

Hence, by the definition of the LTS $\xRightarrow{}$, we also have

\[O[F_1] \mid O[F_2] \xRightarrow{\pi} P_1 \mid P_2\]

We now apply Lemma 65 to infer $O[F_i] \xRightarrow{\sigma} P_i$. It also holds $O[F_i] \xRightarrow{\pi} P_i$ (the observations in $s_i$ were possible in (3), with an extra parallel component, which may only reduce the possibility of transitions in the LTS $\xRightarrow{}$).

For (2): $s \in \text{inter}(s_1, s_2)$ implies that each $s_i$ is alternating, at most one of them is $P$-starting, $s$ is alternating, and active iff one of the composing traces is $P$-starting. This means that we can assume that when an input transition of $s$ is performed the source process is input reactive.

Then Lemma 68 follows from the previous lemma and the operational correspondence for the output-prioritised LTS $\xRightarrow{\sigma, \pi}$ (Corollary 8).

**The call-by-name $\lambda$-calculus**

The relationship between OGS and $\pi$-calculus representations of the $\lambda$-terms examined in Sections 6 and 7 is not specific to call-by-value. A similar relationship can be established for call-by-name. We summarise here the main aspects and the main results. We maintain terminologies and notations in the call-by-value sections. Call-by-name langue is given in Figure 8. Its reduction $\xRightarrow{}$ is defined by the following two rules:

\[
(\lambda x. M)N \xRightarrow{} M[N/x] \quad M \xRightarrow{} N
\]

The OGS for call-by-name uses three kinds of names: continuation names (ranged over by $p, q$), variable names (ranged over by $x, y$), and value-variables (ranged over by $v, w$). Continuation names are used as in call-by-value; variable names acts as pointers to $\lambda$-terms and correspond to the variables of the $\lambda$-calculus; value names are pointers to values.

The OGS uses the syntactic categories in Figure 8, where $L$ range over extended $\lambda$-terms, which are needed to accommodate value names. We write $\Lambda^+$ for the set of extended $\lambda$-terms.

In a call-by-name setting, Player Questions are of two possible shapes:

- **Player Value-Question (PVQ)** $\bar{v}(x, p)$, receiving a variable $x$ and a continuation name $p$ through a value name $v$;
(τ) \[\langle M, p, \gamma, \phi \rangle \xrightarrow{\tau} \langle N, p, \gamma, \phi \rangle\] when \(M \nrightarrow_n N\)

(PA) \[\langle V, p, \gamma, \phi \rangle \xrightarrow{p(v)} \langle \gamma \cdot [v \mapsto V], \phi \uplus \{v\} \rangle\]

(PVQ) \[\langle E[vM], p, \gamma, \phi \rangle \xrightarrow{v(g)} \langle \gamma \cdot [y \mapsto M] \cdot [y \mapsto (E, p)], \phi \uplus \{y, q\} \rangle\]

(PTQ) \[\langle E[x], p, \gamma, \phi \rangle \xrightarrow{p(x)} \langle \gamma \cdot [q \mapsto (E, p)], \phi \uplus \{q\} \rangle\]

(OA) \[\langle \gamma \cdot [p \mapsto (E, q)], \phi \rangle \xrightarrow{p(v)} \langle E[v], q, \gamma, \phi \uplus \{v\} \rangle\]

(OVQ) \[\langle \gamma \cdot [v \mapsto V], \phi \rangle \xrightarrow{v(g)} \langle V[y], p, \gamma, \phi \uplus \{p, y\} \rangle\] when \(\gamma(v) = V\)

(OTQ) \[\langle \gamma, \phi \rangle \xrightarrow{\hat{a}} \langle M, p, \gamma, \phi \uplus \{p\} \rangle\] when \(\gamma(x) = M\)

(IOQ) \[\langle ? \mapsto M, \phi \rangle \xrightarrow{\hat{a}} \langle M, p, \epsilon, \phi \uplus \{p\} \rangle\]

**Figure 9** CBN OGS LTS

---

**Corollary 70.** For any \(F, F'\) we have: \(F \equiv_a F'\) iff \(N[F] \equiv_{\pi} N[F']\) iff \(F \equiv_a F'\) iff \(N[F] \approx_{\pi} N[F']\).

The modification to obtain the concurrent version of OGS (C-OGS) for call-by-name are similar to those for call-by-value in Section 7.

We report the results of operational correspondence and full abstractions.

**Lemma 71** (CBN-OGS and \(\pi\), weak transitions). Suppose \(F \in C\)-OGS.

1. If \(F \xrightarrow{\mu} F'\) then \(N[F] \xrightarrow{\mu} N[F']\).

2. Conversely, if \(N[F] \xrightarrow{\mu} P\) then there is \(F'\) such that \(F \xrightarrow{\mu} F'\) and \(P \approx_{\pi} N[F']\).

As for call-by-value, so for call-by-name the 'up-to composition' technique for bisimulation (Definition 21) can be imported from the \(\pi\)-calculus, and then used to establish that the equivalence induced on \(\lambda\)-terms by the sequential and concurrent versions of OGS coincide.

Specifically, the technique is needed to prove the call-by-name analogous of Lemma 24

**Corollary 72.** For any \(\lambda\)-terms \(M, N\), the following statement are the same: \(\langle M \rangle \equiv_a \langle N \rangle\);

\(\langle M \rangle \equiv_{\pi} \langle N \rangle\);

\(N[M] \equiv_{\pi} N[N]\); \(N[M] \approx_{\pi} N[N]\); \(N[M] \approx_{\pi} N[N]\).

The equivalence induced by the \(\pi\) (or \(\pi\)-calculus) encoding of the call-by-name \(\lambda\)-calculus is known to coincide with that of the Lévy-Longo Trees [42]. In the light of Corollary 72, the result can thus be transported to OGS, both in its Alternating and its Concurrent variants, and both for traces and for bisimilarity.
Encoding of call-by-name $\Lambda$ into $I\pi$ processes:

$$\mathcal{N}[\lambda x. M] \overset{\text{def}}{=} (p) \!\! \overrightarrow{\varphi(v), v(x, q), \mathcal{N}[M](q)}$$

$$\mathcal{N}[x] \overset{\text{def}}{=} (p) \, \overrightarrow{\varphi(r), r \triangleright p}$$

$$\mathcal{N}[MN] \overset{\text{def}}{=} (p) \, \nu q \left( \mathcal{N}[M](q) \mid !q(v). \overrightarrow{\varphi(x, p'), (p' \triangleright p \mid !x(r), \mathcal{N}[N](r))} \right)$$

Extension to $\Lambda^+$:

$$\mathcal{N}[E[vM]] \overset{\text{def}}{=} (p) \, \overrightarrow{\varphi(x, r), (\mathcal{N}[[r \mapsto (E, p)]] \mid !x(q), \mathcal{N}[M]q)}$$

$$\mathcal{N}[v] \overset{\text{def}}{=} (p) \, \overrightarrow{\varphi(w), w \triangleright v}$$

Encoding of OGS environments to $I\pi$ processes:

$$\mathcal{N}[[v \mapsto V] \cdot \gamma] \overset{\text{def}}{=} \begin{cases} v(x, q). \mathcal{N}[M](q) \mid \mathcal{N}[[\gamma]] & \text{if } V = \lambda x. M \\ v \triangleright w \mid \mathcal{N}[[\gamma]] & \text{if } V = w \end{cases}$$

$$\mathcal{N}[[x \mapsto M] \cdot \gamma] \overset{\text{def}}{=} !x(q). \mathcal{N}[M](q) \mid \mathcal{N}[[\gamma]]$$

$$\mathcal{N}[[q \mapsto (E, p)] \cdot \gamma] \overset{\text{def}}{=} \begin{cases} q(v). \overrightarrow{\varphi(x, r), (\mathcal{N}[[r \mapsto E', p]] \mid !x(q), \mathcal{N}[M](q)) \mid \mathcal{N}[[\gamma]]} & \text{if } E = E'M \\ q \triangleright p \mid \mathcal{N}[[\gamma]] & \text{if } E = [\cdot] \end{cases}$$

$$\mathcal{N}[\emptyset] \overset{\text{def}}{=} 0$$

Encoding of OGS configurations to $I\pi$ configurations:

$$\mathcal{N}[[M, p, \gamma, \phi]] \overset{\text{def}}{=} \langle \mathcal{N}[M](p) \mid \mathcal{N}[[\gamma]], \phi \rangle$$

$$\mathcal{N}[[v, p, \gamma, \phi]] \overset{\text{def}}{=} \langle \mathcal{N}[v](p) \mid \mathcal{N}[[\gamma]], \phi \rangle$$

$$\mathcal{N}[[E[vM], p, \gamma, \phi]] \overset{\text{def}}{=} \langle \mathcal{N}[E[vM]](p) \mid \mathcal{N}[[\gamma]], \phi \rangle$$

$$\mathcal{N}[[\gamma, \phi]] \overset{\text{def}}{=} \langle \mathcal{N}[[\gamma]], \phi \rangle$$

The encoding of call-by-name $\lambda$ into $I\pi$
Connections between game semantics and tree representation of $\lambda$-terms have been previously explored, e.g., in [33], where a game model of untyped call-by-name $\lambda$-calculus is shown to correspond to Levy-Longo trees. The connections that we have derived above, however, go beyond Lévy-Longo Trees: in particular, the bisimilaritise that have been established between the observables (i.e., the dynamics) in the two models set a tight relationship between them while allowing one to transfer results and techniques along the lines of what we have shown for call-by-value.