Polymorphic Game Semantics for Dynamic Binding

James Laird*

Department of Computer Science, University of Bath, U.K.

Abstract

We present a game semantics for an expressive typing system for block-structured programs with late binding of variables and System F style polymorphism. As well as generic programs and abstract datatypes, this combination may be used to represent behaviour such as dynamic dispatch and method overriding.

We give a denotational models for a hierarchy of programming languages based on our typing system, including variants of PCF and Idealized Algol. These are obtained by extending polymorphic game semantics to block-structured programs. We show that the categorical structure of our models can be used to give a new interpretation of dynamic binding, and establish definability properties by imposing constraints which are identical or similar to those used to characterize definability in PCF (innocence, well-bracketing, determinacy). Moreover, relaxing these can similarly allow the interpretation of side-effects (state, control, non-determinism) - we show that in particular we may obtain a fully abstract semantics of polymorphic Idealized Algol with dynamic binding by following exactly the methodology employed in the simply-typed case.

1998 ACM Subject Classification F.3.2 Semantics of Programming Languages

Keywords and phrases Game semantics, denotational models, PCF, Idealized Algol

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.27

1 Introduction

We describe a semantics for higher-order programs with combinations of three features: System F-style polymorphism, block structured *late binding*, and side-effects (control and state). Generic, higher-rank polymorphism is a powerful principle supporting information hiding and encapsulation, through features such as generics and interfaces, modules or abstract data types. It is frequently encountered in combination with various side-effects, and in contexts in which binding of variables to values (and, implicitly or explicitly to types) is late, or dynamic, – i.e. dependent on where they are called. Examples include features such as dynamic dispatch, virtual methods and method overriding.

Previous denotational models of parametric polymorphism have focussed on λ-calculus-based (and effect-free) type theories with static binding such as System F [7, 23]. Late binding has received less theoretical attention, and there is a lack of formal principles on which to base construction of a model. The aims of this work are to develop such principles for an *intensional semantics* of generic polymorphism, to show that they can be used to capture program behaviour in the presence or absence of side-effects such as local state and non-local control, and to characterise program equivalence precisely and concretely, as a step towards semantics-based verification.

* Research supported by EPSRC grant EP/K037633/1.

Table 1 Typing Judgements.

$$\frac{\Theta \vdash \Gamma, T}{\Gamma, x{:}T, \Gamma' \vdash_\Theta x{:}T} \qquad \frac{\Gamma, x{:}S, \Gamma' \vdash_\Theta N{:}T \qquad \Gamma, x{:}S, \Gamma' \vdash_\Theta M{:}S}{\Gamma, x{:}S, \Gamma' \vdash_\Theta \mathtt{let}\ x{=}M\ \mathtt{in}\ N{:}T} \qquad \frac{\Gamma \vdash_{\Theta, X} M{:}T \qquad \Theta \vdash \Gamma}{\Gamma \vdash_\Theta \Lambda X.M{:}\forall X.T}$$

$$\frac{\Gamma, x{:}S \vdash_\Theta M{:}T}{\Gamma \vdash_\Theta \lambda x^S.M{:}S{\to}T} \qquad \frac{\Gamma \vdash_\Theta M{:}S{\to}T \qquad \Gamma \vdash_\Theta N{:}S}{\Gamma \vdash_\Theta M\ N{:}T} \qquad \frac{\Gamma \vdash_\Theta M{:}\forall X.T \qquad \Theta \vdash S}{\Gamma \vdash_\Theta M\{S\}{:}T[S/X]}$$

Our model extends the game semantics for generic call-by-name polymorphism described in [17], which developed earlier notions of variable game [10, 2] with an interpretation of quantification as a relation between question and answer moves. It was used in [17] to construct a fully abstract semantics of a programming language with a typing system based on System F, and locally declared *general references*. The capacity to escape from static binding using references is fundamental to the full abstraction result, posing the question of whether there are more constrained models of polymorphism based on similar principles. We address this question by giving a model of a language based on PCF and its game semantics [12, 4], with dynamic binding (as in dialects of Lisp, for example). This has a block structure – i.e. definitions are scoped by the block in which they occur, which is captured by our games model in a natural way, whereas instantiation breaks *static* scope, suggesting a semantic relationship between dynamic scope and polymorphism. We develop a formal semantics of late binding using the categorical properties of our model – in particular, it is an instance of *sequoid structure*, which was introduced in [15].

We also consider the extension of our language with side-effects. In a close analogy with the games models of PCF with static binding, we identify three conditions on strategies (determinacy, first-order well-bracketing and dynamic innocence) required for our definability result, which may be relaxed individually, or in combination, to interpret side-effects (non-determinism, non-local control flow, integer state). We focus on the last of these, giving a fully abstract semantics of Idealized Algol [24] – a block-structured language with integer state, now extended with System F-style polymorphism and dynamic binding, a combination which can be used to represent objects and classes [22]. Our model gives a very concrete representation of polymorphic types, suggesting that it could lend itself to algorithmic models for determining program equivalence and other properties, as in e.g. [6].

2 A Polymorphic Type Theory with Dynamic Binding

We define a second-order type theory with late binding by extending System F [23, 7] with a set of ground types B and a (*non-binding*) `let` operation. The formation rules for types are:

$$\frac{}{\Theta, X, \Theta' \vdash X} \qquad \frac{}{\Theta \vdash B} \qquad \frac{\Theta \vdash S \qquad \Theta \vdash T}{\Theta \vdash S \to T} \qquad \frac{\Theta, X \vdash T}{\Theta \vdash \forall X.T}$$

Typing rules for terms are given in Table 1.

Universally quantified types may be used to type generic programs [23], but also to construct new datatypes [7] including abstract data types, modules and interfaces, represented using existential types as described in [20] – e.g. the type $\exists X.(T_1 \& \ldots \& T_n) =_{df} \forall Y.(\forall X.T_1 \to \ldots \to T_n \to Y) \to Y$ may be assigned to objects which implement methods of type $T_1, \ldots, T_n$ which interface with a shared, abstract type X.

We define a programming language, PCF^2_d (second-order PCF with dynamic binding) based on this type theory by adding appropriate constants for arithmetic: $0 : \mathtt{nat}$, $\mathtt{succ}, \mathtt{pred} : \mathtt{nat} \to \mathtt{nat}$, divergence $(\Omega_\mathtt{T} : T)$ and conditionals $\mathtt{If0} : \mathtt{nat} \to T \to T \to T$. This contains

Table 2 Operational Semantics of PCF^2_d.

$$\frac{}{C;\mathcal{E}\Downarrow C;\mathcal{E}} \qquad \frac{M;\mathcal{E}\Downarrow \mathtt{0};\mathcal{E}'}{\mathtt{If0}\ M;\mathcal{E}\Downarrow\lambda xy.x;\mathcal{E}'} \qquad \frac{M;\mathcal{E}\Downarrow\lambda x.M';\mathcal{E}'_a \quad M'[a/x];\mathcal{E}'_a,(a,N)\Downarrow C;\mathcal{E}''}{M\ N;\mathcal{E}\Downarrow C,\mathcal{E}''} \qquad \frac{M;\mathcal{E},(a,M),\mathcal{E}_a\Downarrow C;\mathcal{E}''}{a;\mathcal{E},(a,M),\mathcal{E}_a\Downarrow C;\mathcal{E}''}$$

$$\frac{M;\mathcal{E}\Downarrow \mathtt{n+1};\mathcal{S}';\mathcal{E}'}{\mathtt{pred}\ M;\mathcal{E}\Downarrow \mathtt{n};\mathcal{E}'} \qquad \frac{M;\mathcal{E}\Downarrow \mathtt{n+1};\mathcal{E}'}{\mathtt{If0}\ M;\mathcal{E}\Downarrow\lambda xy.y;\mathcal{E}'} \qquad \frac{M;\mathcal{E}\Downarrow\Lambda X.M';\mathcal{E}' \quad M'[T/X];\mathcal{E}'\Downarrow C;\mathcal{E}''}{M\{T\};\mathcal{E}\Downarrow C,\mathcal{E}''} \qquad \frac{N;\mathcal{E},(a,M)\Downarrow C;\mathcal{E}'}{\mathtt{let}\ a{=}M\ \mathtt{in}\ N;\mathcal{E}\Downarrow C;\mathcal{E}'}$$

PCF itself: note that `let` definitions are implicitly recursive – thus we may define fixed points (e.g. $\mathbf{Y} : (T \to T) \to T =_{df} \lambda f^{T\to T}.(\lambda x^T.\mathtt{let}\ x = (f\ x)\ \mathtt{in}\ x)\ \Omega$).

The operational semantics for PCF^2_d (Table 2) is given by a convergence relation between configurations $M;\mathcal{E}$ and $C;\mathcal{E}'$ where:

- M and C are terms of PCF^2_d extended with a set of *procedure names*.
- C is in *canonical form*, given by the grammar: $C ::= \mathtt{n} \mid \lambda x.M \mid \Lambda X.M$
- $\mathcal{E}, \mathcal{E}'$ are finite *sequences* $(a_1, M_1), (a_2, M_2), \ldots, (a_n, M_n)$ of bindings of procedure names to terms (which may contain more than one binding for each name). $\mathcal{E}_a$ denotes a stack which does not contain a binding for a. We write $M \Downarrow \mathtt{n}$ if $M;\_ \Downarrow \mathtt{n};\mathcal{E}$ for some $\mathcal{E}$.

Here are some examples clarifying the expressive power of our language:

- Late binding can be used to vary the meaning of a procedure, depending on the (dynamic) scope in which it is called – for example, in $\mathtt{let}\ f = (\mathtt{let}\ x = 1\ \mathtt{in}\ M)\ \mathtt{in}\ \mathtt{let}\ x = 0\ \mathtt{in}\ N$, the variable x takes the value 1 if called inside the scope of f, and 0 otherwise. Note that this is an instance of *overriding* of one definition of x by another.
- Like general references, late binding can allow the arguments of a function to escape from its static scope – for example $g : (R \to S \to S) \to R \to T) \vdash \lambda x^R.(g\ (\lambda y^T.\lambda z^S.\mathtt{let}\ x = y\ \mathtt{in}\ z))\ x : R \to T$ – in which the second argument to g may be captured from inside the scope of the first.
- Unlike references, late binding does not allow procedures to pass information out of the block in which they are defined. So, for example the functions $\lambda x^{\mathtt{nat}}.(\mathtt{If0}\ x)\ \mathtt{then}\ x\ \mathtt{else}\ x$ and $\lambda x^{\mathtt{nat}}.x$ are observationally equivalent in PCF^2_d (as in PCF itself), as no information can be passed between sequential invocations of x. As in e.g. Idealized Algol adding integer state (see Section 7) allows these terms to be separated.
- Late binding of terms extends implicitly to types – e.g. a variable of type $\exists X.T$ may become bound to procedures in which the explicit witnessing type varies depending on where the binding takes place.

3 Simple Games

Our model is based on a games interpretation of second-order (intuitionistic) linear type theory described in [17], adapted to modelling block structure and dynamic scope. It may also be constructed in a setting with explicit justification pointers, closer to the original model of PCF [12] and models based on relaxing the constraints on its strategies [4, 14, 9] – comparison with the latter would be one reason for preferring such a construction. Our interpretation of second-order types adapts readily to such a setting [16], where we may give a combinatorial definition of the interpretation of dynamic binding along the lines of the semantics of general references in [5]. However, the presentation here via a model of intuitionistic linear type theory allows a categorical decomposition of our model using structures introduced in [15], which are useful in proving its soundness.

We begin by describing our semantics of dynamic binding at first-order types, which may be interpreted as *simple games*, given by a set of moves partitioned between two players, and

between questions and answers, together with event-structure-style *enabling* and *conflict* relations on moves. We work with a *universal* set of moves, for which these relations and predicates are fixed, principally because this will ultimately simpify the definition of type substitution.

Fix a set of *tags* $\mathcal{A}$ which includes left and right tags l, r and special "enabling" and "conflict" tags e, c), and a set of *values* $\mathcal{V}$ which includes the natural numbers. The set $\mathcal{U}$ of *moves* is the set of sequences $\mathcal{A}^* \cup \{w!_v \mid w \in \mathcal{A}^*, v \in \mathcal{V}\}$. For any set of moves X, we write X^+ and X^- f for the subsets of X consisting of Opponent and Proponent moves (even and odd-length sequences, respectively) and $X^?$ and $X^!$ for the sets $X \cap \mathcal{A}^*$ of (*first-order questions* or 1-questions) and $X \backslash \mathcal{A}^*$ (*first-order answers* or 1-answers). These play the same roles as questions and answers in the game semantics of PCF [3, 12] and Idealized Algol [4] – opening and closing blocks of moves, corresponding to procedure calls. We will subsequently introduce a further Q/A-labelling to capture second order structure, effectively giving two levels of bracketing on games.

▶ **Definition 1.** A *simple game* is a set of moves $A \subseteq \mathcal{U}$ such that if $u \cdot w \in A$ then $u \in A$ if and only if $w = \varepsilon$ or $w = !_v$ for some $v \in \mathcal{V}$.

In other words, for every 1-answer $m!_v \in A^!$, there is a corresponding 1-question $m \in A^?$ (belonging to the other player), but moves are otherwise incomparable with respect to the prefix order. For example, given a set of values $X \subseteq \mathcal{V}$, $\underline{X} = \{\varepsilon\} \cup \{!_v \mid v \in X\}$ is the "flat" game with a single, Opponent 1-question ε, and a corresponding Proponent 1-answer $!_v$ for each $v \in X$ (corresponding to an atomic datatype with values in X). We define the following relations on 1-questions (cf. *event structures*, although conflict is reflexive):

Enabling: $m \vdash n$ if there exist $u \in \mathcal{A}^*$, $w, w' \in (\mathcal{A}\backslash\{e\})^*$ such that $m = u \cdot w$ and $n = u \cdot ew'$.

Conflict: $m \# n$ if $m = n$ or there exist $u, w, w' \in \mathcal{A}^*$ such that $m = u \cdot cw$ and $n = u \cdot cw'$.

A move $m \in (\mathcal{A}\backslash\{e\})^*$ is *initial*. A game A is *negative* if every initial move in M is an Opponent move. We construct simple games using the following operations:

- For a game A, and sequence $w \in \mathcal{A}^*$, $w.A = \{w \cdot m \mid m \in A\}$ is a game.
- For games A, B such that $m \not\sqsubseteq m'$ and $m \not\sqsubseteq m'$ for all $(m, m') \in A \times B$, $A \cup B$ is a game.
- $A \& B = cl.A \cup cr.B$ – a disjoint union of A and B with initial moves in conflict.
- $A \otimes B = ll.A \cup rr.B$ – a disjoint union of A and B with initial moves not in conflict.
- $A \to B = l.A \cup rr.B$ a disjoint union of A and B with initial moves not in conflict and Proponent/Opponent roles switched in A.

Positions of a game are represented as finite sequences of moves subject to certain conditions on the *stack of open* 1*-questions* – the subsequence obtained by erasing all moves between any answer and its corresponding question (inclusive) – i.e. $\mathsf{stack}(sq) = \mathsf{stack}(s)q$ if $q \in \mathcal{U}^?$, and $\mathsf{stack}(s(q!_v)) = s'$ if $\mathsf{stack}(s) = s'qt$ and q doesn't occur in t. ($\mathsf{stack}(s)$ is undefined, otherwise). In other words playing a 1-question move pushes it onto the stack, corresponding to opening a block, and playing an answer pops the corresponding question from the stack, closing the block. Valid positions of the game (*legal sequences*) are defined by requiring that in the stack every move is preceded by an enabling move (i.e. every call to an argument is preceded by an open call to its originating function) and no open moves are in conflict.

▶ **Definition 2.** The set L_A of legal sequences on A consists of $t \in A^*$ such that if $sn \sqsubseteq t$:

Alternation: n is an Opponent move if and only if s is even length.

Conflict-freeness: If $s'm \sqsubseteq \mathsf{stack}(s)$ then $m \not\# n$

Enabling: If m is a non-initial 1-question then $m \vdash n$ for some $s'm \sqsubseteq \mathsf{stack}(s)$.

A *strategy* on A is a non-empty, even-prefix-closed set of even-length sequences in L_A.

The strategies in our model satisfy a further constraint – they depend only on the moves in the current (open) block.

▶ **Definition 3.** A sequence $s \in L_A$ is *complete* if $\mathsf{stack}(s) = \varepsilon$. A strategy σ is $*$*-closed* if whenever $s \in \sigma$ is complete then for any $t \in L_A$, $s \cdot t \in \sigma$ if and only $t \in \sigma$.[1]

We now define a series of additional constraints on strategies which will combine to yield definability in PCF^2_d. These are analogous (or identical) to the constraints used to characterize definability in PCF itself [12]. First, we define a restricted history of play accessible to strategies definable with dynamic binding – a version of the *view function* (which may be seen as a representation of *static* scope) which considers only the (implicit) justification pointers between 1-answers and their corresponding questions.

▶ **Definition 4.** The (Proponent) *dynamic view* $\lceil s \rceil$ of $s \in L_A$ is defined as follows:
$\lceil sq \rceil = \lceil s \rceil q$, if $q \in A^?$ (q is a question),
$\lceil s(q!_v) \rceil = s'q(q!_v)$, if $(q!_v) \in A^+$ ($q!_v$ is an Opponent 1-answer) and $\lceil s \rceil = s'qt$,
$\lceil s(q!_v) \rceil = s'$, if $(q!_v) \in A^-$ ($q!_v$ is a Proponent 1-answer) and $\lceil s \rceil = s'qt$.

▶ **Definition 5.** A $*$-closed strategy σ on a simple game A is:

- *deterministic* if $sa, sb \in \sigma$ implies $a = b$.
- *dynamically innocent* if it is deterministic and $sab, tab \in L_A$, $sab, t \in \sigma$ and $\lceil sa \rceil = \lceil ta \rceil$ implies $tab \in \sigma$. In other words, play by σ may only depend on the dynamic view.[2]
- 1*-well-bracketed* if $s(m!_v) \in \sigma$ implies $\mathsf{stack}(s) = s'm$ – in other words, Proponent may only answer the most recently asked, unanswered question (close the most recently opened enclosing block).

We abbreviate combinations of these constraints as combinations of the letters $\{B, D, I\}$.

So, for example, the only deterministic $*$-closed strategies on the game $\underline{\mathbb{N}}$ are the dynamically innocent and well-bracketed strategies $\bot = \{\varepsilon\}$ and for each $i \in \mathbb{N}$ the strategy $(\varepsilon \cdot !_i)^*$ which responds to Opponent's initial 1-question ε with the answer $!_i$ each time it is asked.

3.1 Categories of Simple Games

For each combination C of constraints (B, D, I) we construct a category $\mathcal{G}_C$, of (negative) simple games, in which morphisms from A to B are $*$-closed C-strategies on $A \to B$.

▶ **Definition 6.** Given a sequence s over A, and an evident partial projection function from A to B, we write $s{\restriction}B$ for the sequence obtained by applying the projection to the moves on which it is defined, and omitting moves on which it is not defined. Given sets of moves $X, Y \subseteq \mathcal{U}$, a sequence s is a *Proponent* X, Y-copycat if for any *even-length* prefix $t \sqsubseteq s$, $t{\restriction}X = t{\restriction}Y$ and an *Opponent* X, Y-copycat if for any *odd-length* prefix $t \sqsubseteq s$, $t{\restriction}X = t{\restriction}Y$.

The identity $\mathsf{id}_A : A \to A$ consists of legal sequences which are Proponent (A^+, A^-) copycats.

▶ **Definition 7.** An *interaction sequence* on games A_1, A_2, A_3 is a sequence s on $A_1 \uplus A_2 \uplus A_3$ such that $s \restriction A_i \to A_j$ is legal for each $i < j$. The *composition* of $\sigma : A_1 \to A_2$ with $\tau : A_2 \to A_3$ is the set of legal sequences t on $A_1 \to A_3$ such that there exists an interaction sequence s on A_1, A_2, A_3 with $s{\restriction}A_1 \to A_2 \in \sigma$ and $s{\restriction}A_2 \to A_3 \in \tau$, and $t = s{\restriction}A_1 \to A_3$.

[1] Cf. the notion of single-threadedness [9].
[2] Note that dynamic innocence implies determinacy by definition: naive attempts to form a category of nondeterministic dynamic-innocent strategies fail for precisely the reasons described in [8] for nondeterministic innocent strategies.

This is a simplified but equivalent definition of composition to that given in [17] (we use the latter, in combination with the following property of the stack, to prove associativity):

▶ **Lemma 8.** *If s is an interaction sequence then* $\mathsf{stack}(s\restriction A_i \to A_j) = \mathsf{stack}(s)\restriction A_i \to A_j$.

Using the proofs of *compositionality* of bracketing, determinacy, and innocence (of which dynamic innocence is a variant) [18, 14, 8], we show:

▶ **Proposition 9.** *For each constraint C, the composition of C-strategies is a C-strategy.*

Since the identity is a BDI-strategy, we have a category $\mathcal{G}_C$ for each constraint. We define symmetric monoidal structure on $\mathcal{G}_C$: $A \otimes B$ is the disjoint union $ll.A \cup rr.B$ with unit $I = \varnothing$ (which is a terminal object). Given $\sigma : A \to C$ and $\tau : B \to D$, we define $\sigma \otimes \tau : A \otimes B \to C \otimes D = \{s \in L_{A\otimes B \to C \otimes D} \mid s\restriction A \to C \in \sigma \wedge s\restriction B \to D \in \tau\}$.

$\mathcal{G}_C$ is not symmetric monoidal closed but does have sufficient exponentials to allow us to construct a Cartesian closed category based on its cofree commutative comonoids. Say that a negative game is *well-opened* if any two of its initial moves are in conflict (so any legal sequence contains at most one unanswered initial question).

▶ **Proposition 10.** *The well-opened games form an* exponential ideal *in $\mathcal{G}_C$.*

Proof. Given a well-opened game B, the (well-opened) exponential of a game A by B is $A \multimap B = e.A \cup rr.B$ – a disjoint union of A and B in which Opponent and Proponent swap roles in A, and initial moves of A are enabled by initial moves of B. ◀

▶ **Proposition 11.** *Any family of well-opened games has a well-opened cartesian product in $\mathcal{G}_C$.*

Proof. E.g. if A_1, A_2 are well-opened then $A_1 \& A_2 = cl.A_1 \cup cr.A_2$ is a (well-opened) cartesian product of A_1 and A_2: by conflict-freeness, any legal sequence on $B \to A_1 \& A_2$ has the form $t \cdot s$, where t is a complete sequence and s is a sequence in $B \to A_1$ or $B \to A_2$. So by the $*$-closure condition, $\sigma : B \to A_1 \& A_2$ is uniquely determined by its left and right projections. ◀

The rules defining legal plays allow the repeated *sequential* calling of procedures, since a 1-question may be replayed once it has been closed, but not – for example – sharing between function and argument. To capture the latter, we show that well-opened games possess a *cofree commutative comonoid* in $\mathcal{G}_C$ based on the following construction:

▶ **Definition 12.** For a well-opened game A let $!A$ be the well-opened game $(er)^*.ll.A$.

In other words $!A = ll.A \cup erll.A \cup ererll.A \cup \ldots$ consists of countably many distinctly tagged copies of A: initial moves in each copy $(er)^n.ll.A$ must be played (by Opponent) in order of the size of n – i.e. it is an instance of a construction introduced in [11].

We may define copycat morphisms $\delta_A : !A \to !A \otimes !A$ and $\epsilon_A : !A \to I$ making $(!A, \delta_A, \epsilon_A)$ a *commutative comonoid*: informally, δ_A plays copycat between $!A \otimes !A$ and $!A$, opening a fresh copy of A in $!A$ for each thread opened on either side of $!A \otimes !A$ (ϵ_A is the strategy $\{\varepsilon\}$). In section 5 we will use the categorical properties of $\mathcal{G}_C$ to define these morphisms formally, and moreover to show that $(!A, \delta_A, \epsilon_A)$ is the *cofree commutative comonoid* on A in $\mathcal{G}_C$ [13] (Proposition 22). In other words, if $U : \mathsf{Comon}(\mathcal{G}_C) \to \mathcal{G}_C$ is the forgetful functor into $\mathcal{G}_C$ from its category of commutative comonoids, then for any object $(B, \delta_B, \epsilon_B)$ of $\mathsf{Comon}(\mathcal{G}_C)$ there is an equivalence (natural in B) between $\mathcal{G}_C(U(B), A)$ and $\mathsf{Comon}(\mathcal{G}_C)(B, !A)$ given by a morphism $\mathsf{der}_A : U(!A) \to A$ and an operation sending each morphism $f : U(B) \to A$

in $\mathcal{G}_C$ to $f^\dagger : B \to !A$ such that $f^\dagger; \mathsf{der}_A = f$ and $\mathsf{der}_A^\dagger = \mathsf{id}_{!A}$. Thus we have a comonad on the full subcategory $\mathcal{G}_C^W$ of $\mathcal{G}_C$ consisting of well-opened games, with co-Kelisli triple $(!\_, \mathsf{der}, (\_)^\dagger)$. By Lemmas 10 and 11 its co-Kleisli category is Cartesian closed.

4 Second-Order Games

Second-order games are based on the model in [17], which represents unbound type variables as $\mathbb{N}$-indexed "holes" into which games may be plugged, and quantification as a labelling of these holes as questions or answers. However, we need to extend this interpretation to take account of block structure (type variables are represented as generic blocks with an opening and closing 1-question). This "second-order" Q/A-labelling is imposed on top of the existing, intrinsic, question-answer structure of simple games (and the constraints of dynamic innocence and 1-well-bracketing):

▶ **Definition 13.** A *second-order game* A is a tuple $(|A|, \lambda_A, \rhd_A)$, where:

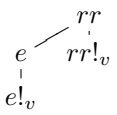
- $|A| \subseteq \mathcal{U}$ is a simple game.
- $\lambda_A : |A|^? \rightharpoonup (\{\mathsf{Q}, \mathsf{A}\} \cup \mathbb{N}\backslash\{0\}$ is a *partial* labelling of 1-questions as 2-*questions*, 2-*answers* or i-*holes* for $i \in \mathbb{N}\backslash\{0\}$
- $\rhd_A \subseteq |A|^? \times |A|^?$ is a *Q/A-relation* on 1-questions such that $\rhd_A \subseteq (\lambda_A^{-1}(\mathsf{Q})^- \times \lambda_A^{-1}(\mathsf{A})^+) \cup (\lambda_A^{-1}(\mathsf{Q})^+ \times \lambda_A^{-1}(\mathsf{A})^-)$ (i.e. Proponent answers Opponent questions and vice-versa).

2-questions and 2-answers represent bound variables which may be instantiated by the questioner. We say that A is a n-*context* game if the set $\lambda_A^{-1}(i)$ of i-hole moves (representing the ith free type variable) is empty for $i > n$ (it is *closed* if $n = 0$).

Note that some 1-questions may be unlabelled – these correspond to concrete ground types. In particular, every simple game A corresponds to a n-context game $(A, \varnothing, \varnothing)$ and the constructions on simple games extend to context games. In addition:

- For $1 \leq i \leq n$, let $\bullet_i$ be the n-context game $(\underline{\mathcal{V}}, \{(\varepsilon, i)\}, \varnothing)$ (corresponding to the type variable X_i), with a single (i-hole) 1-question, ε, with 1-answers $!_v$ for each $v \in \mathcal{V}$.
- $\forall_n A = (|A|, \lambda_A[\lambda_A^{-1}(n)^- \mapsto \mathsf{Q}, \lambda_A^{-1}(n)^+ \mapsto \mathsf{A}], \rhd_A \cup (\lambda_A^{-1}(n)^- \times \lambda_A^{-1}(n)^+)$ – the Proponent n-hole moves of A become answers to the Opponent n-hole moves, which become questions.

For example, the game $\Sigma = \forall_1(\bullet_1 \multimap \bullet_1)$ is played over the simple game $\{rr, e\} \cup \{rr!_v, e!_v \mid v \in \mathcal{V}\}$; rr is a 2-question, with 1-answers $rr!_v$ and 2-answer e, which has 1-answer $e!_v$ for $v \in \mathcal{V}$:



Contrast with the corresponding game in [17] (which has a single question and answer).

▶ **Definition 14.** The set L_A of legal sequences on A consists of $t \in L_{|A|}$ such that:

Well-bracketing (2): If $\lambda_A(m) = A$ then $\mathsf{pending}(\mathsf{stack}(s)) = s'q$, where $q \rhd_A n$ and $\mathsf{pending}(s)$ is the last-asked, unanswered 2-question in s.[3]

Q/A-copycat: If $sm(q!_v) \sqsubseteq t$ (where $\lambda_A(q) = \mathsf{Q}$) or $s(q!_v)m \sqsubseteq t$ (where $\lambda_A(q) = \mathsf{A}$) then $m = q'!_v$ for some q'.

A second-order strategy is a non-empty even-prefix-closed set of even-length legal sequences which satisfies the *Q/A-copycat condition for strategies*: if $s \in \sigma$ and $s(q!_v) \in L_A$, where $\lambda_A(q) = \mathsf{A}$ then $s(q!_v)(q'!_v) \in \sigma$ for some q'.

[3] i.e. $\mathsf{pending}(sm) = \mathsf{pending}(s')$ if $\lambda_A(m) = \mathsf{A}$ and $\mathsf{pending}(s) = s'q$; $\mathsf{pending}(sm) = sm$ otherwise.

In other words, the stack obeys a well-bracketing condition for 2-questions and answers as defined in [17] – calls to procedures of generic type are matched as opening and closing parentheses – and the new Q/A-copycat condition. This requires that the 1-answer move which closes a 2-question must copy the preceding 1-answer (so, for example, Proponent cannot immediately close Opponent's opening question in Σ with a 1-answer) and that a 1-answer which closes a 2-answer must be copied by the following move. In combination, they force the propagation of return values from 2-answers up to the corresponding 2-question. For an example of how the well-bracketing and Q/A-copycat conditions bite, observe that the only strategies on $\Sigma = \forall_1(\bullet_1 \multimap \bullet_1)$ are $\bot = \{\varepsilon\}$ and the "generic identity"copycat strategy $\top : I \to \Sigma$ consisting of even prefixes of sequences in $(\Sigma_{v\in\mathcal{V}}(rr)(e)(e!_v)(rr!_v))^*$.

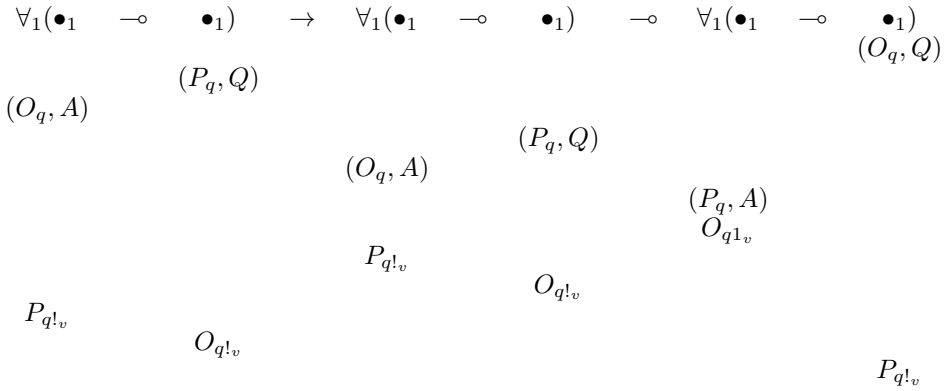
Instantiation of m-context games $B_1, \ldots, B_n$ in a n-context game A (corresponding to instantion of types for type variables) is by replacing each i-hole move (*and its response*) in A with a copy of B_i:

▶ **Definition 15.** For a n-context game A and well-opened m-context games $B_1, \ldots, B_n$, the m-context game $A[B_1, \ldots, B_n] = (A \backslash \bigcup\{\{u, u!_v \mid v \in \mathcal{V}\} \mid \lambda_A(u) = i, 1 \leq i \leq n\}) \cup \bigcup\{u.B_i \mid \lambda_A(u) = i, 1 \leq i \leq n\}$.

Instantiation into strategies is defined by extending them to the instantiated games by playing *copycat* between the game substituted for a question and that substituted for its answer. (Essentially, the application of the operation defined in [17] to the stack.)

▶ **Definition 16.** Given a legal stack t on a game D, let $\mathsf{Cl}(D, t)$ be the set of 2-question-answer pairs $\{(q, a) \in \lambda_D^{-1}(\mathsf{Q}) \times \lambda_D^{-1}(\mathsf{A}) \mid \exists sa \sqsubseteq t.\mathsf{pending}(s) = q\}$. For a morphism $\sigma : A \to C$ in $\mathcal{G}_C(n)$ define $\sigma[B_1, \ldots, B_n] : A[B_1, \ldots, B_n] \to C[B_1, \ldots, B_n]$ to be the set of legal sequences t on $A[B_1, \ldots, B_n] \to C[B_1, \ldots, B_n]$ such that $t\restriction\overrightarrow{\forall}(A \to C) \in \sigma$ and:
If $s \sqsubseteq t$, $\lambda_{A\to C}(q) = i$ and $(q, a) \in \mathsf{Cl}(\overrightarrow{\forall}(A \to C), \mathsf{stack}(s))$ then $\mathsf{stack}(s)\restriction q = \mathsf{stack}(s)\restriction a$, where $t\restriction m$ is the restriction of t to suffixes of m. We use the Q/A-copycat condition to show that this is a deterministic strategy.

As an example of how instantiation combines "schematic" copycat behaviour derived from second-order bracketing with direct copying of 1-answers, consider the strategy on $\forall_1(\bullet_1 \multimap \bullet_1) \to \forall_1(\bullet_1 \multimap \bullet_1) \multimap \forall_1(\bullet_1 \multimap \bullet_1)$ obtained by instantiating $\forall_1(\bullet_1 \multimap \bullet_1)$ into the identity on $\forall_1(\bullet_1 \multimap \bullet_1)$. This contains the following play, in which the first 1-answer determines all of the remaining moves by the Q/A-copycat condition.



We now define indexed categories of context games for each combination of constraints on strategies, yielding a model of System F typing [21]. For each $n \in \mathbb{N}$, let $\mathcal{G}_C(n)$ be the

category in which objects are negative n-context games and morphisms from A to B are C-strategies on the closed game $\forall_1 \ldots \forall_n (A \to B)$. Composition and identity in each $\mathcal{G}_C(n)$ are defined as for $\mathcal{G}_C$: proof that this yields a category follows [17].

Let $\mathcal{I}$ be the category in which objects are positive natural numbers and morphisms from m to n are m-tuples of well-opened n-context games. Composition of $(B_0, \ldots, B_m)$ with $(A_1, \ldots, A_l)$ is $(A_1[B_1, \ldots, B_m], \ldots A_l[B_1, \ldots, B_m])$ and the identity on n is $(\bullet_1, \ldots, \bullet_n)$. This is a category with finite products (arithmetic sums) and is finitely generated by them from the object 1. We define an $\mathcal{I}$-indexed symmetric monoidal category with (specified) finite products: the functor $\breve{\mathcal{G}}_C$ from $\mathcal{I}^{op}$ into the category of symmetric monoidal categories which sends n to the category $\mathcal{G}_C(n)$ and $(B_0, \ldots, B_m) : n \to m$ to the instantiation functor $\_[B_0, \ldots, B_m] : \mathcal{G}_C(m) \to \mathcal{G}_C(n)$.

Let $\breve{\mathcal{G}}_C^{+1}$ be the $\mathcal{I}$-indexed category which sends n to $\mathcal{G}_C(n+1)$ and $(B_1, \ldots, B_m)$ to $\_[B_1, \ldots, B_m, \bullet_{n+1}]$, and $J : \breve{\mathcal{G}}_C \to \breve{\mathcal{G}}_C^{+1}$ the $\mathcal{I}$-indexed inclusion functor (i.e. $J_n : \mathcal{G}_C(n) \to \mathcal{G}_C^{+1}(n)$ is the inclusion of $\mathcal{G}_C(n)$ into $\mathcal{G}_C(n+1)$).

▶ **Proposition 17.** *J has an indexed right adjoint $\forall : \breve{\mathcal{G}}_C^{+1} \to \breve{\mathcal{G}}_C$.*

Proof. For any n-context game A, and $n+1$-context game B, $\forall_n(J_n(A) \to B) = A \to \forall_n B$. Hence there is a natural isomorphism between $\mathcal{G}_C(n+1)(J_n(A), B)$ and $\mathcal{G}_C(n)(A, \forall_n(B))$. These satisfy the *Beck-Chevalley* condition and so form an indexed adjunction (see [17]). ◀

The indexed category $\breve{\mathcal{G}}_C$ possesses the structure identified for simple games – i.e. well-opened (indexed) Cartesian products, exponentials and cofree commutative comonoids.

5 Semantics of Dynamic Binding

Dynamically bound variables may be considered as objects with two methods – *definition* (using `let`), and *invocation*. In similar fashion to the semantics of ground-type and higher-order references [4, 5], our semantics is based on representing such an object as a strategy on the game $\S A = !(!A \multimap \Sigma) \otimes !A \cong !((!A \multimap \Sigma) \& A)$ so that invocation corresponds to right-projection, and $\texttt{let}\ x = M\ \texttt{in}\ N : T$ corresponds to applying the left projection from this product to M, instantiating with $[\![T]\!]$ and applying to N.

So we interpret a term $x_1 : S_1, \ldots, x_n : S_n \vdash M : T$ as a morphisms from $\S[\![S_1]\!] \otimes \ldots \otimes \S[\![S_n]\!]$ to $[\![T]\!]$ in $\mathcal{G}_C$. To interpret λ-abstraction, we define a declaration strategy $\mathsf{dec}_A : !A \to \S A$ which dynamically connects definitions to invocations, so that abstraction corresponds to composition with the strategy $\mathsf{dec}_A : !A \to \S A$, followed by currying: $[\![\Gamma \vdash \lambda x^S.M : S \to T]\!] : [\![\Gamma]\!] \to [\![S \to T]\!] = \Lambda((\![\Gamma]\!] \otimes \mathsf{dec}_{[\![S]\!]}); [\![\Gamma, x : S \vdash M : T]\!])$

The dynamic binding strategy is very similar to the reference cell strategy defined in [5]. The main difference that it is dynamically innocent – connecting only "reads" and "writes" *in the current block*. Informally, Proponent plays as the generic identity $\top$ on Σ and if Opponent requests an invocation by opening a thread of $!A$ on the right, then Proponent responds by playing copycat with the last-opened definition thread *on the stack*, if there is one, and

playing copycat with the source copy of $!A$, otherwise. An example play:

$$
\begin{array}{ccccccc}
!A & \to & !(!A & \multimap & \Sigma) & \otimes & !A \\
 & & & & O_q & & \\
 & & & & P_q & & \\
 & & & & & & O_q \\
 & & P_q & & & & \\
 & & O_{q!_v} & & & & \\
 & & & & & & P_{q!_v} \\
 & & & & O_{q!_v} & & \\
 & & & & P_{q!_v} & & \\
 & & & & & & O_q \\
P_q & & & & & &
\end{array}
$$

5.1 A Categorical Model of Dynamic Binding

In order to define the commutative monoid $(!A, \delta_A, \epsilon_A)$ and the dynamic binding strategy formally, and establish their key properties (that the former is the cofree commutative comonoid on A in $\breve{\mathcal{G}}_C$, and that the latter connects invocation and definition correctly), we give a categorical decomposition of the tensor into an *action* $\_ \oslash \_$: of each monoidal category $\breve{\mathcal{G}}_C$ on its (wide) subcategory of *block-linear* morphisms.

▶ **Definition 18.** $\sigma : A \to B$ is *block-linear* if $\forall s \in \sigma$ if $s{\restriction}B$ is complete then $s{\restriction}A$ is complete.

In other words, σ responds to the initial 1-question move by either closing it (with a 1-answer) or playing a 1-question move in B; when this is closed, σ either closes the opening move or opens a new block in B with a 1-question, and so on. This property is possessed by the identity, and preserved by composition, so for each n we may define $\breve{\mathcal{L}}_C$ to be the subcategory of $\breve{\mathcal{G}}_C$ in which objects are well-opened games and morphisms are block-linear strategies. This has cartesian products (given by &). For any game B, the monoidal exponential restricts to a functor $B \multimap \_ : \breve{\mathcal{L}}_C \to \breve{\mathcal{L}}_C$ with the following property.

▶ **Lemma 19.** *For any B, the functor $B \multimap \_ : \breve{\mathcal{L}}_C \to \breve{\mathcal{L}}_C$ has right and left adjoints, which commute with $B \multimap \_$ (and therefore coincide) – i.e. $B \multimap \_$ has a dual in the monoidal category of endofunctors on $\breve{\mathcal{L}}_C$.*

Proof. For any well-opened A, let $A \oslash B = ll.A + er.B$ (i.e. equivalent to $B \multimap A$ except that Proponent/Opponent roles are not swapped in B). Then there are evident adjunctions:

$$\frac{\breve{\mathcal{L}}_C(A \oslash B, C)}{\breve{\mathcal{L}}_C(A, B \multimap C)} \qquad \frac{\breve{\mathcal{L}}_C(B \multimap A, C)}{\breve{\mathcal{L}}_C(A, C \oslash B)}$$

and a natural isomorphism $\gamma : B \multimap (A \oslash B') \cong (B \multimap A) \oslash B'$. ◀

In the terminology of [15], this commuting adjunction defines a *sequoid*, i.e.:

- $\_ \oslash \_ : \breve{\mathcal{L}}_C \times \breve{\mathcal{G}}_C \to \breve{\mathcal{L}}_C$ is a monoidal *action* of $(\breve{\mathcal{G}}_C, \otimes, I)$ on $\breve{\mathcal{L}}_C$ (a strong monoidal functor $\_ \oslash \_$ from $\breve{\mathcal{G}}_C$ into the category of endofunctors on $\breve{\mathcal{L}}_C$).
- There is a monoidal natural transformation $\omega : J\_ \otimes \_ \to J(\_ \oslash \_)$ (where $J : \breve{\mathcal{L}}_C \to \breve{\mathcal{G}}_C$ is the inclusion functor) which commutes with the adjunctions $\otimes \dashv \multimap$ and $\oslash \dashv \multimap$.

The sequoid has the further property of "sequential decomposability": by conflict-freeness, any legal sequence on $A \otimes B$ is a concatenation of sequences from $A \oslash B$ and $B \oslash A$, and so:

▶ **Lemma 20.** *For any well-opened games A, B, $\langle \omega_{A,B}, \theta; \omega_{B,A} \rangle : A \otimes B \to (A \oslash B) \& (B \oslash A)$ is an isomorphism (where θ is the symmetry isomorphism of $\otimes$).*

Observe that $!A$ is a *minimal invariant* for the endofunctor $J(A \oslash \_) : \breve{\mathcal{G}}_C \to \breve{\mathcal{G}}_C$: $!A = (er)^*.ll.A = ll.A \cup er.(er)^*.ll.A = A \oslash !A$ and the identity on $!A$ is the least fixed point for the operation taking $f : !A \to !A$ to $A \oslash f$. Following [15], we may use this property to define the cofree commutative comonoid structure of $!A$:

▶ **Definition 21.** For a well-opened game A, the commutative comonoid $(!A, \delta_A, \epsilon_A)$ in $\breve{\mathcal{G}}_C$ is *defined* as follows:

- $\delta_A : !A \to !A \otimes !A$ is the least fixed point of the endofunction on $\breve{\mathcal{G}}_C(!A, !A \otimes !A)$ sending f to $!A \cong A \oslash !A \xrightarrow{\mathsf{id}_A \oslash f} A \oslash (!A \otimes !A) \cong (A \oslash !A) \oslash !A \cong !A \oslash !A \xrightarrow{\Delta_{!A}} !A \otimes !A$.
- $\epsilon_A : !A \to I$ is the terminal map into I.

We show that this is the cofree commutative comonoid on A by defining:

- der_A is the morphism $!A \cong A \oslash !A \xrightarrow{\mathsf{id}_A \oslash \epsilon_A} A \oslash I \cong A$
- given $f : U(B) \to A$, $f^\dagger : B \to !A$ is the least fixed point of the operation sending $g : B \to !A$ to $B \xrightarrow{\delta_B} B \otimes B \xrightarrow{f \otimes g} A \otimes !A \xrightarrow{\omega_{A,!A}} A \oslash !A \cong !A$.

▶ **Proposition 22.** $!A$ *is the* cofree commutative comonoid *on* A *in* $\breve{\mathcal{G}}_C$.

Like the reference cell strategy [15], we can define $\mathsf{dec}_A : !A \multimap !(A \multimap \Sigma) \otimes !A$ formally, using the sequoidal decomposition of the cofree commutative comonoid, based on the counit $\eta_B : B \to (A \multimap B) \oslash A$ of the adjunction $\_ \oslash A \dashv A \multimap \_$.

▶ **Definition 23.** Recall that for any object B, $!(!A \multimap B) \otimes !A$ is the Cartesian product of $!(!A \multimap B) \oslash !A$ and $!A \oslash !(!A \multimap B)$ in $\breve{\mathcal{G}}_C$. and Let $\mathsf{dec}_A : !A \to !(!A \multimap \Sigma) \otimes !A$ be the least fixed point of the map $\Phi : \breve{\mathcal{G}}_C(!A, !(!A \multimap \Sigma) \otimes !A) \to \breve{\mathcal{G}}_C(!A, !(!A \multimap \Sigma) \otimes !A)$ sending f to the pairing of:
$!A \cong A \oslash !A \xrightarrow{A \oslash f} A \oslash (!(A \multimap \Sigma) \otimes !A) \cong A \oslash (!A \otimes !(!A \multimap \Sigma)) \cong (A \oslash !A) \oslash !(!A \multimap \Sigma) \cong !A \oslash !(!A \multimap \Sigma)$ and
$!A \xrightarrow{\epsilon_A} I \xrightarrow{\top} \Sigma \xrightarrow{\eta_{\Sigma,!A}} (!A \multimap \Sigma) \oslash !A \xrightarrow{(!A \multimap \Sigma) \oslash f} (!A \multimap \Sigma) \oslash (!(!A \multimap \Sigma) \otimes !A) \cong !(!A \multimap \Sigma) \oslash !A$

From this definition, we derive the following key equations relating $\mathsf{invoke}_A : \S A \to A \oslash \S A = (\delta_{\S A}; \omega_{\S A, \S A}); (\pi_r; \epsilon_A \oslash \S A)$ and $\mathsf{define} : \S A \to (!A \multimap \Sigma) \oslash \S A = (\delta_{\S A}; \omega_{\S A, \S A}); (\pi_l; \epsilon_A \oslash \S A)$.

Invocation: $\mathsf{dec}_A; \mathsf{invoke}_A : !A \to A \oslash \S A = (\delta_{!A}; \omega_{!A,!A}); (\mathsf{der}_A \oslash \mathsf{dec}_A)$ (i.e. reading a dynamically assigned variable returns its value and leaves the stack unchanged).

Definition: $\mathsf{dec}_A; \mathsf{define}_A : !A \to (!A \multimap \Sigma) \oslash \S A = t_{!A}; \Lambda((\top \otimes \mathsf{dec}_A); \omega_{\Sigma, \S A}); \gamma_{!A, \Sigma, \S A}$ – i.e. composing with definition returns a method which updates the variable and returns $\top$.

6 Denotational Semantics of $\mathbf{PCF}^{\mathbf{2}}_{d}$

We interpret our type theory using the categorical structure identified in the previous section. Each type $X_1, \ldots, X_n \vdash T$ is interpreted as a well-opened n-context game $[\![T]\!]_{X_1,\ldots,X_n}$:

$[\![S \to T]\!]_\Theta = !\,[\![S]\!]_\Theta \multimap [\![T]\!]_\Theta \qquad [\![X_i]\!]_{X_1,\ldots,X_n} = \bullet_i \qquad [\![\forall X_n.T]\!]_\Theta = \forall_n [\![T]\!]_{\Theta, X_n}$

Interpretation of the types and constants of PCF as games and BDI-strategies follows the games models in [3, 12] – i.e. `nat` denotes the flat game $\underline{\mathbb{N}}$. Terms $x_1 : S_1, \ldots, x_n : S_n \vdash_\Theta M : T$ are interpreted as morphisms from $\S[\![S_1]\!]_\Theta \otimes \ldots \otimes \S[\![S_n]\!]_\Theta$ to $[\![\mathbf{T}]\!]_\Theta$ in $\breve{\mathcal{G}}_C$, defined using the categorical structure we have identified:

Invocation and definition correspond to right and (application of) left projection from §:

$[\![x_1, : T_1 \ldots, x_n : T_n \vdash x_i : T_i]\!]_\Theta : \S[\![T_1]\!]_\Theta \otimes \ldots \otimes \S[\![T_n]\!]_\Theta \to [\![T_i]\!]_\Theta = \pi_i; \pi_r; \mathsf{der}$

$[\![\Gamma \vdash \mathtt{let}\ x_i = M\ \mathtt{in}\ N : T]\!]_\Theta : [\![\Gamma]\!]_\Theta \to [\![T]\!]_\Theta$

$= \delta_{[\![\Gamma]\!]}; ((\delta_{[\![\Gamma]\!]}; ((\pi_i; \pi_l; \mathsf{der}) \otimes [\Gamma \vdash M : S]\!]_\Theta)); \mathsf{eval})[[\![T]\!]_\Theta] \otimes [\![\Gamma \vdash N : T]\!]_\Theta); \mathsf{eval}$

Abstraction and application composition with the strategy dec, with currying/application:
$[\![\Gamma \vdash \lambda x^S.M : S \to T]\!]_\Theta : [\![\Gamma]\!]_\Theta \to [\![S \to T]\!]_\Theta = \Lambda((\![\Gamma]\!]_\Theta \otimes \mathsf{dec}_{[\![S]\!]_\Theta}); [\![\Gamma, x : S \vdash M : T]\!]_\Theta)$
$[\![\Gamma \vdash M\ N : T]\!]_\Theta : [\![\Gamma]\!]_\Theta \to [\![T]\!]_\Theta = \delta_{[\![\Gamma]\!]}; ([\![\Gamma \vdash M : S \to T]\!]_\Theta \otimes [\![\Gamma \vdash N : S]\!]_\Theta); \mathsf{eval}$
Type abstraction and instantiation are interpreted using second-order structure:
$[\![\Gamma \vdash \Lambda X.M : \forall X.T]\!]_\Theta : [\![\Gamma]\!]_\Theta \to [\![\vdash \forall X.T]\!]_\Theta = \forall_n [\![\Gamma \vdash M]\!]_{\Theta, X_n}$
$[\![\Gamma \vdash M\{S\} : T[S/X]]\!]_\Theta : [\![\Gamma]\!]_\Theta [[\![\vdash S]\!]_\Theta] \to [\![T]\!]_\Theta [[\![S]\!]_\Theta] = [\![\Gamma \vdash M]\!]_\Theta [[\![S]\!]_\Theta]$
Given a well-typed term $x_1 : S_1, \ldots, x_k : S_k \vdash N : T$, and $\mathcal{E} = (x_1, M_1), \ldots, (x_k, M_k)$, we define the interpretation of a configuration of the operational semantics:
$[\![\mathcal{E}; N]\!] = ((\bot; \mathsf{dec}_{S_1}) \otimes \ldots \otimes (\bot; \mathsf{dec}_{S_n})); [\![\mathtt{let}\ x_1 = M_1\ \mathtt{in}\ \ldots \mathtt{let}\ x_k = M_k\ \mathtt{in}\ N]\!]$.
Using the equalities established for the the dynamic binding operation, and the soundness of β-equivalence in the semantics of second order types, we show:

▶ **Proposition 24** (Soundness). *If* $(M; \mathcal{E}) \Downarrow C; \mathcal{E}'$ *then* $[\![M; \mathcal{E}]\!] = [\![C; \mathcal{E}']\!]$.

As in [17], we establish *computational adequacy* via a sequence of approximating semantics: For each n, we define an operational semantics of PCF^2_d in which each definition may be called at most n times in a block – i.e. application and `let` push n copies of (a, M) onto the stack and invocation removes one copy. Reduction with respect to this operational semantics is size-reducing with respect to a simple measure and thus terminating.

We then show that this operational semantics is sound, and therefore adequate, with respect to a denotational semantics in which the dec strategy is replaced by its nth approximant. Adequacy of the semantics now follows: for any term $M : \mathtt{nat}$, $[\![M]\!] = \bigcup_{n \in \omega} [\![M]\!]_n$. Hence if $[\![M]\!] \neq \bot$, there exists n with $[\![M]\!]_n \neq \bot$ and so $M \Downarrow$ as required.

▶ **Proposition 25.** $[\![\mathcal{E}; M]\!] = [\![\mathtt{n}]\!]$ *implies* $M; \mathcal{E} \Downarrow \mathtt{n}; \mathcal{E}'$ *for some* $\mathcal{E}'$.

We prove a definability property for first-order types analogous to the corresponding results for the games models of PCF: finite definability holds at type T if every *compact* BDI-strategy σ on $[\![T]\!]$ is the denotation of a term of type T (a dynamically innocent strategy σ is compact with the respect to the inclusion order if and only if the set $\{\lceil s \rceil \mid s \in \sigma\}$ is finite). The proof uses a decomposition argument based on similar principles to the original definability proof for PCF [3, 12]; we use the categorical decomposition of the tensor and cofree commutative comonoid into the sequoid to axiomatize this in a similar style to [1].

▶ **Proposition 26.** *Finite definability holds at all first-order types.*

To extend our definability result from first-order to second-order types, the key step is to simplify the latter using the observation that the type $\mathsf{I} = \forall X.X \to X$ is *strongly generic*:

▶ **Lemma 27.** *For any type* $T(X)$*, there is a* definable retraction $\forall X.T \trianglelefteq T[\mathsf{I}/X]$ *(i.e. there are* PCF^2_d *terms which denote a retraction/section between the corresponding type-objects).*

Proof. The retraction of $[\![\forall X.T]\!]$ into $[\![T[\mathsf{I}/X]]\!]$ is the instantiation morphism. The corresponding section plays copycat between $[\![T[\mathsf{I}/X]]\!]$ and $[\![\forall X.T]\!]$ until Opponent asks a question q in $[\![T[\mathsf{I}/X]]\!]$ corresponding to a negative occurrence of X in T: Proponent gives the sole answer to this; if Opponent copies this move as an answer to a question corresponding to a positive occurrence of X in T, then Proponent plays the move corresponding to q (which is an answer) in $[\![\forall X.[\![T]\!]]\!]$.

We may define this in PCF^2_d by giving terms $x : X \vdash \mathsf{in}_T : T[X] \to T[\mathsf{I}/X]$ and $x : X \vdash \mathsf{proj}_T : T[\mathsf{I}/X] \to T[X]$ such that $\lambda y^{T[\mathsf{I}/X]}.\Lambda X.(\lambda x.\mathsf{proj}_T)\,\Omega_X$ denotes the required strategy – i.e. $[\![\lambda z^{\forall X.T}.\Lambda X.(\lambda x.\mathsf{proj}_T\, z\{\mathsf{I}\})\,\Omega_X]\!] = [\![\lambda z.z]\!]$. The free variable x of type X is used to dynamically associate positive occurrences and negative occurrences of X:

- $\mathsf{in}_X : X \to \mathsf{I} = \lambda y^X.\mathtt{let}\ x = y\ \mathtt{in}\ \Lambda Z.\lambda z^Z.z$, $\mathsf{proj}_X : \mathsf{I} \to X = \lambda y^{\mathsf{I}}.y\{X\}\ x$.
- $\mathsf{in}_T = \mathsf{proj}_T = \lambda x^T.x$ for T a ground type, or variable type other than X.
- $\mathsf{in}_{S\to T} = \lambda f^{S\to T}.\lambda y^{S[\mathsf{I}/X]}.\mathsf{in}_T\ (\mathsf{proj}_S\ y)$, $\mathsf{proj}_{S\to T} = \lambda f^{S[\mathsf{I}/X]\to T[\mathsf{I}/X]}.\lambda y^S.\mathsf{proj}_T\ (\mathsf{in}_S\ y)$
- $\mathsf{in}_{\forall Y.T} = \lambda y^{\forall Y.T}.\Lambda Y.(\mathsf{in}_T\ y\{Y\})$, $\mathsf{proj}_T = \Lambda y^{\forall Y.T[\mathsf{I}/X]}.\Lambda Y.(\mathsf{proj}_T\ y\{Y\})$ ◀

This result may be used to eliminate (almost) all quantifiers: define the *almost quantifier free* types by the grammar: $U ::= \mathtt{nat} \mid \mathsf{I} \mid U \to U$.

▶ **Proposition 28.** *Finite definability holds at almost quantifier free types.*

▶ **Lemma 29.** *Finite definability holds at all types.*

Proof. Using Lemma 27, we may show that every closed type T is a retract of an almost quantifier free type T'. Thus if the image of $\sigma : [\![T]\!]$ under this retraction is definable as a term $M : T'$, then σ is definable as $\mathsf{proj}_T\ M$. ◀

7 Computational Effects with Dynamic Binding: Idealized Algol

As we have noted, we are in a situation analogous to PCF in that our definability result is obtained by applying three constraints to strategies (dynamic innocence, first-order well-bracketing and determinacy), which may be relaxed singly and in combination to obtain models of PCF^2_d with side-effects (local state [4], non-local control [14] and non-determinism [9]), with definability results obtained via the *factorization* results described in *loc. cit.* We illustrate the case of integer state, describing a model of IA^2_d – second-order Idealized Algol with dynamic binding. This is an expressive and useful combination of types and effects. In particular, existential types may be used to encapsulate the state of an object, and so define classes in Idealized Algol [22].

We define IA^2_d to be the proper extension of PCF^2_d obtained from our type theory by taking the set B of base types to consist of the types $\mathtt{nat}$, $\mathtt{com}$ (commands) and $\mathtt{var}$ (integer references) and adding additional constants for sequential composition of commands ($\mathtt{skip} : \mathtt{com}$ and $\mathsf{seq}_T : \mathtt{com} \to T \to T$) and declaration, assignment and dereferencing of integer variables ($\mathtt{new}_i : (\mathtt{var} \to \mathtt{com}) \to \mathtt{com}$, $\mathtt{assign} : \mathtt{var} \to \mathtt{nat} \to \mathtt{var}$ and $\mathtt{deref} : \mathtt{var} \to \mathtt{nat}$). We use common syntactic sugar for sequential composition, declaration, assignment, etc.

The operational semantics of IA^2_d extends that of PCF^2_d with a set of location names, Loc (which may occur as canonical forms), and adding to configurations a *store* (heap) $\mathcal{S}$ – a set of bindings of location names to integers defining a partial function from loc to $\mathbb{N}$ – i.e. $\Downarrow$ becomes a relation between triples $M;\mathcal{E};\mathcal{S}$ and $C,\mathcal{E}',\mathcal{S}'$. We decorate each PCF^2_d-rule with a store and add the following rules:

$$\frac{M\ a;\mathcal{E};\mathcal{S}\cup\{(a,i)\} \Downarrow C;\mathcal{E}';\mathcal{S}'\cup\{(a,v)\}}{\mathtt{new}_i\ M;\mathcal{E};\mathcal{S}\Downarrow C;\mathcal{E}';\mathcal{S}'}\ a\notin\mathsf{dom}(\mathcal{S}) \qquad \frac{M;\mathcal{E};\mathcal{S}\Downarrow \mathtt{skip};\mathcal{E}';\mathcal{S}'}{\mathsf{seq}_T\ M;\mathcal{E};\mathcal{S}\Downarrow \lambda x^T.x,\mathcal{E}'',\mathcal{S}''}$$

$$\frac{M;\mathcal{E};\mathcal{S}\Downarrow a;\mathcal{E}';\mathcal{S}' \quad N;\mathcal{E};\mathcal{S}'\Downarrow \mathtt{n},\mathcal{E}'',\mathcal{S}''}{M:=N;\mathcal{E};\mathcal{S}\Downarrow \mathtt{skip};\mathcal{E}'';\mathcal{S}''[a\mapsto n]} \qquad \frac{M;\mathcal{E};\mathcal{S}\Downarrow a;\mathcal{E}';\mathcal{S}'}{!M;\mathcal{E};\mathcal{S}\Downarrow \mathtt{n};\mathcal{E}';\mathcal{S}'}\ \mathcal{S}'(a)=n$$

$\mathtt{com}$ denotes the flat game $\underline{\{*\}}$ (with a single 1-question and 1-answer), and $\mathtt{var}$ the game $\mathsf{var} = [\![\mathtt{nat}]\!]\&[\![\mathtt{com}]\!]^\omega$ (the cartesian product of the types of its methods – read from the cell, assign it with 0, assign with 1,...). The constants for sequential composition and assignment and dereferencing (right and left projection from var) all denote dynamically innocent strategies. The only constant which does not is $\mathtt{new}_i : (\mathtt{var} \to \mathtt{com}) \to \mathtt{com}$, which denotes the composition of $x : \mathtt{var} \vdash \lambda f.f\ x$ with the "reference cell" strategy $\mathsf{cell}_i : I \multimap !\mathsf{var}$ which returns the last value written to it (or its initial value, i) [4]. This is not dynamically

innocent (in fact, cell_i is not $*$-closed): each Proponent 1-answer hides the preceding Opponent 1-question, so the state of the cell is never visible to Proponent.

To establish the soundness of this interpretation, we define the denotation of a configuration relative to a state transformation. Let cell_i^j be the restriction of the cell strategy which starts in a state in which it is assigned with i and terminates in a state in which it is assigned with j. Given a well-typed term $a_1 : \mathtt{var}, \ldots, a_m : \mathtt{var}, x_1 : S_1, \ldots, x_n : S_n \vdash N : T$, and $\mathcal{E} = (x_{i1}, M_1), \ldots, (x_{ik}, M_k)$, we define $[\![\mathcal{E}; N]\!]_{\mathcal{S}\mapsto\mathcal{S}'} = (\mathsf{cell}_{\mathcal{S}(a_1)}^{\mathcal{S}'(a_1)} \otimes \ldots \otimes \mathsf{cell}_{\mathcal{S}(a_m)}^{\mathcal{S}'(a_m)}); [\![E; N]\!]$.

▶ **Proposition 30.** *$[\![\mathcal{E}; M]\!]_{\mathcal{S}\mapsto\mathcal{S}'} = [\![v]\!]$ if and only if $M; \mathcal{E}; \mathcal{S} \Downarrow v; \mathcal{E}'; \mathcal{S}'$ for some $\mathcal{E}'$.*

Terms $M, N : T$ of IA_d^2 are *observationally equivalent* ($M \simeq_T N$) if for all contexts $C[\_ : T] : \mathtt{com}$, $C[M] \Downarrow$ if and only if $C[N] \Downarrow$. We now show that our model of IA_d^2 captures this equivalence (full abstraction), following precisely the arguments for full abstraction of the game semantics of Idealized Algol itself in [5], extended to the language without *bad variables* in [19]. First, we add to IA_d^2 and its games model a *bad-variable constructor* $\mathtt{mkvar} : (\mathtt{nat} \to \mathtt{com}) \to \mathtt{nat} \to \mathtt{var}$, which denotes the currying of the pairing $\langle\langle(\pi_l \otimes \underline{i}); \mathsf{eval} \mid i \in \omega, \pi_r\rangle : [\![\mathtt{nat} \to \mathtt{com}]\!] \& [\![\mathtt{nat}]\!] \to ([\![\mathtt{com}]\!]^\omega \& [\![\mathtt{nat}]\!])$. Then for any IA_d^2 type T:

▶ **Proposition 31.** *Every compact BD-strategy on $[\![T]\!]$ is denoted by a term of $\mathrm{IA}_d^2 + \mathtt{mkvar}$.*

Proof. σ may be *factorized* into the composition of a dynamically innocent strategy $\widehat{\sigma} : ![\![\mathtt{var}]\!] \to [\![T]\!]$ with the strategy $\mathsf{cell}_0 : 1 \to ![\![\mathtt{var}]\!]$, by using the reference cell to record the history of play (see [4]). As in loc. cit. we may easily extend the proof of definability for PCF_d^2 to establish that $\widehat{\sigma}$ is the denotation of a term $x : \mathtt{var} \vdash M : T$ of $\mathrm{IA}_d^2 + \{\mathtt{mkvar}\} - \{\mathtt{new}\}$, and thus σ is definable as $\mathtt{new}_0\,(\lambda x.M)$ ◀

A sequence t is 1-well-bracketed if both Proponent and Opponent obey the 1-well-bracketing condition – i.e. if $s(m!_v) \sqsubseteq t$ then $\mathsf{stack}(s) = s'm$ for some s'. Let $\mathsf{comp}(\sigma)$ be the set of complete, well-bracketed sequences in σ.

▶ **Theorem 32** (Full abstraction)**.** *$M \simeq N$ if and only if $\mathsf{comp}([\![M]\!]) = \mathsf{comp}([\![N]\!])$.*

Proof. The implication from left to right is a consequence of computational adequacy. For the converse, if $\mathsf{comp}([\![M : T]\!]) \neq \mathsf{comp}([\![N : T]\!])$ then without loss of generality there exists a complete sequence $s \in [\![M]\!]$ such that $s \notin [\![N]\!]$. Thus the strategy $\sigma : [\![T]\!] \to [\![\mathtt{com}]\!]$ which consists of even prefixes of $(qs*)^*$ distinguishes them – i.e. $[\![M]\!]; \sigma \neq \bot$ and $[\![N]\!]; \sigma = \bot$. By Proposition 28, σ is definable as a term of $\mathrm{IA}_d^2 + \mathtt{mkvar}$. We can extend this result to IA_d^2 without $\mathtt{mkvar}$ by following McCusker's analysis of good variables in Idealized Algol [19]. This uses the observation that any IA_d^2-definable object of $\mathtt{var}$ type which responds to a read request with a given value, must behave in the same way when presented with a write request for the same value, to show that any (denotations of) terms which may be distinguished by a term of $\mathrm{IA}_d^2 + \mathtt{mkvar}$ may be distinguished by a term of IA_d^2. This argument extends *mutatis mutandis* to IA_d^2: in other words, there is a $\mathtt{mkvar}$-free term $L : T \to \mathtt{com}$ such that $[\![\mathcal{L}\, M]\!] \neq \bot$ and $[\![L\, N]\!] = \bot$ and so $L\, M \Downarrow$ and $L\, N \not\Downarrow$ by adequacy. ◀

8 Conclusions and Further Directions

We have described games models for polymorphic languages with late binding of variables, with and without mutable state (IA_d^2 and PCF_d^2): the former fully abstract for closed terms. By relaxing further constraints (bracketing, determinacy) we may interpret further effects such as non-local control.

The definability result for PCF^2_d implies that a fully abstract model can be obtained by an intrinsic preorder collapse construction (as in the original semantics of PCF [12, 3]): the problem of characterising this preorder remains open. Our definability and full abstraction results are restricted to terms of closed type: terms with free variables introduce a form of the *bad variable problem* which remains to be solved.

Although they do not contain general references, both languages are still very expressive. However, they open the possibility of defining bounded fragments (e.g. by bounding the nesting of function calls as in [6]) in which programs may be interpreted as as regular or visibly pushdown automata and so used to verify properties of polymorphic programs such as observational equivalence.

References

1. S. Abramsky. Axioms for full abstraction and full completeness. In *Essays in Honour of Robin Milner*. MIT Press, 1997.
2. S. Abramsky and R. Jagadeesan. A game semantics for generic polymorphism. *Annals of Pure and Applied Logic*, 133(1):3–37, 2004.
3. S. Abramsky, R. Jagadeesan, and P. Malacaria. Full abstraction for PCF. *Information and Computation*, 163:409–470, 2000.
4. S. Abramsky and G. McCusker. Linearity, Sharing and State: a fully abstract game semantics for Idealized Algol with active expressions. In P.W. O'Hearn and R. Tennent, editors, *Algol-like languages*. Birkhauser, 1997.
5. S. Abramsky, K. Honda and G. McCusker. A fully abstract games semantics for general references. In *Proceedings of LICS'98*. IEEE Press, 1998. doi:10.1109/LICS.1998.705669.
6. D. Ghica, A. S. Murawaki, and C.-H. L. Ong. Syntactic control of concurrency. In *Proceedings of ICALP'04*, number 3142 in LNCS. Springer, 2004.
7. J.-Y. Girard. *Interprétation functionelle et elimination des coupures de l'arithmètique d'ordre supérieur*. PhD thesis, Université Paris VII, 1972.
8. R. Harmer. *Games and Full Abstraction for Nondeterministic Languages*. PhD thesis, Imperial College London, 1999.
9. R. Harmer and G. McCusker. A fully abstract games semantics for finite non-determinism. In *Proceedings of the Fourteenth Annual Symposium on Logic in Computer Science, LICS'99*. IEEE Computer Society Press, 1998.
10. D. Hughes. Games and definability for System F. In *Proceedings of the Twelfth International syposium on Logic in Computer Science, LICS'97*. IEEE Computer Society Press, 1997.
11. J. M. E. Hyland. Game semantics. In *Semantics and Logics of Computation (Publications of the Newton Institute)*. Cambridge University Press, 1995.
12. J. M. E. Hyland and C.-H. L. Ong. On full abstraction for PCF: I, II and III. *Information and Computation*, 163:285–408, 2000.
13. Y. Lafont. *Logiques, catégories et machines*. PhD thesis, Université Paris 7, 1988.
14. J. Laird. Full abstraction for functional languages with control. In *Proceedings of LICS'97*. IEEE Computer Society Press, 1997. doi:10.1109/LICS.1997.614931.
15. J. Laird. A categorical semantics of higher-order store. In *Proceedings of CTCS'02*, number 69 in ENTCS. Elsevier, 2002.
16. J. Laird. Game semantics for a polymorphic programming language. In *Proceedings of LICS'10*. IEEE Press, 2010.
17. J. Laird. Game semantics for a polymorphic programming language. *Journal of the ACM*, 60(4), 2013.
18. G. McCusker. *Games and full abstraction for a functional metalanguage with recursive types*. PhD thesis, Imperial College London, 1996. Cambridge University Press.

19 G. McCusker. On the semantics of the bad-variable constructor in Algol-like languages. In *Proceedings of MFPS XIX*, ENTCS, 2003. To appear.

20 J. Mitchell and G. Plotkin. Abstract types have existential type. *ACM transactions on Programming Languages and Systems*, 10(3):470–502, 1988.

21 A. Pitts. Polymorphism is set-theoretic constructively. In D. Pitt, editor, *Proceedings of CTCS'88*, number 283 in LNCS. Springer, 1988.

22 U. Reddy. Objects and classes in Algol-like languages. *Information and Computation*, 172(1):63–97, 2002.

23 J. C. Reynolds. Towards a theory of type structure. In *Proceedings of the Programming Symposium, Paris 1974*, number 19 in LNCS. Springer, 1974.

24 J. C. Reynolds. The essence of Algol. In *Algorithmic Languages*, pages 345–372. North Holland, 1981.