Game Semantics for Bounded Polymorphism

J. Laird

Department of Computer Science, University of Bath, UK
jiml@cs.bath.ac.uk

Abstract. We describe a denotational, intensional semantics for programs with polymorphic types with bounded quantification, in which phenomena such as inheritance between stateful objects may be represented and studied. Our model is developed from a game semantics for unbounded polymorphism, by establishing dinaturality properties of generic strategies, and using them to give a new construction for interpreting subtyping constraints and bounded quantification. We use this construction to give a denotational semantics for a programming language with general references and an expressive polymorphic typing system. We show that full abstraction fails in general in this model, but that it holds for all terms at a rich collection of bounded types.

1 Introduction

By combining subtype and parametric polymorphism, type systems with bounded quantification increase the expressive power of both: they may be used to write programs which are generic, but range over a constrained set of types (a program of type $\forall X \leq S.T$ may be instantiated only with a subtype of S). They have been used to develop formal theories of key aspects of object oriented languages such as *inheritance* [6]. Our aim is to develop an *intensional* denotational semantics for subtyping and bounded polymorphism — i.e. a formal semantic account of the constraints on behaviour which can be expressed in such a system. This allows for models which combine bounded polymorphism with computational effects (in particular, state) and, potentially, for semantics-based subtyping theories which capture aspects of program behaviour.

Previous denotational models of bounded quantification have been based on an *extensional* interpretation of polymorphism — for instance by interpreting subtyping as inclusion of *partial equivalence relations* [5] or as a relation between games [8], so that bounded quantification corresponds to a product or intersection over all instances satisfying the bound. This has generated valuable insights into the theory of subtyping and polymorphism, but it is not clear how it may be extended to include computational effects (for example). We take an alternative, more intrinsically *intensional* approach by constructing an interpretation of bounded quantification without first specifying a subtyping relation, beyond the requirement that it gives rise to coercion morphisms between objects: we will subsequently show how to interpret subtyping judgements as such coercions.

Technically, our model is based on the game semantic framework for generic call-by-name polymorphism described in [15] and reviewed and updated in Section 2, which develops earlier notions of variable game [12,1] with a new interpretation of quantification as a relation between question and answer moves. In Section 3, we extend this framework by showing that context games correspond to mixed variance functors on games, and generic strategies to (composable) *dinatural* transformations between them. In Section 4, we use these properties to construct an interpretation of subtyping environments and bounded quantification. The basic idea is to represent a bounded quantification $\forall X \leq S.T(X^-, X^+)$ as an unbounded quantification $\forall X.T(S \times X, X)$, so that negative occurrences of type variables may be coerced into their bounding types by right projection: defining the composition of strategies with free, constrained variables requires precisely the dinaturality properties established in the previous section.

In Section 5 we give a semantics for a stateful programming language with a second-order typing system based on Cardelli and Wegner's *Bounded Fun* [6], and its extensively studied fragment, System $F_{\leq}$("F-sub") [9,7] (an extension of the second-order λ-calculus, System F [11,17] with bounded quantification). However, these are pure type theories: the language we study is (in essence) a conservative extension of the metalanguage with general references, and its games model, originally defined (and proposed as a semantic basis for object oriented programs) by Abramsky, Honda and McCusker [3]. Finally, we show that full abstraction does not hold in our model (arguably, due to an expressive deficit in the language and typing system) but holds for a significant fragment determined by a simple constraint on types.

2 Second Order Game Semantics

Our games model is based on a semantics of unbounded "System F - style" polymorphism which is essentially equivalent to that defined in [13,15], with some minor differences in the representation of games (up to these, our model of bounded quantification is a conservative extension). Here we recall and modify the key definitions, referring to [15] for details and proofs.

A "*n-context game*" as defined in [13] is a Hyland-Ong arena — a set of moves partitioned between two players, with a binary *enabling relation* defining a directed, bipartite acyclic graph — with a further partitioning of moves into sets of *questions*, *answers*, and *i-holes* for $i \in \{1, \ldots, n\}$ into which other arenas may be instantiated, and a *scoped question answer/relation* — a ternary relation on moves — $q \rhd_m a$ means a can answer q within the scope of m — such that Player questions are scoped by Player moves and answered by Opponent moves, and Opponent moves are scoped by Opponent moves and answered by Player moves.

In the following, we will work within a *universal* set of moves in which enabling, labelling of moves and the question/answer relation are determined by a fixed representation — so a context arena may be specified simply by giving a

set of moves within this universe (although this is not essential for our model of bounded quantification, it facilitates some constructions).

Definition 1. *Let $\mathcal{A}$ be the set $\mathbb{N} \cup \{l, r\} \cup \{\forall_i \mid i \in \mathbb{N}\}$. Define the universal set of moves $\mathcal{U}$ to be $\{w \bullet_n \mid w \in \mathcal{A}^* \wedge n \in \mathbb{N} \backslash \{0\}\}$ — i.e. a move is a word over $\mathcal{A}$ followed by a terminal symbol $\bullet_n$ for some $n \geq 1$.*

Say that a sequence $w \in \mathcal{A}^*$ is *positive* if it contains an even number of occurrences of the symbol l, and *negative* otherwise. If w is positive, then $w\bullet_i$ is an *Opponent* move, otherwise it is a *Player* move. (Given $X \subseteq \mathcal{U}$, X^+ is the set of Opponent moves and X^- the set of Player moves in X.)

Definition 2. *We define the* enabling relation $\vdash \subseteq \mathcal{U} \times \mathcal{U}$*: $m \vdash n$ if there exist sequences u, v, w with v, w not containing l, such that $m = u \cdot v$ and $n = u \cdot lw$.*

This is evidently a directed, bipartite acyclic graph on $\mathcal{U}$. Source nodes (Opponent moves containing no occurrence of l) are called initial moves.

Definition 3. *Moves are partitioned into* questions*,* answers*, and* i-holes *as follows:*

- *If $m = v\bullet_i$ where v contains no occurrences of $\forall_i$, then m is an i-hole move.*
- *Otherwise, $m = u\forall_i \cdot v\bullet_i$, for some unique v which contains no occurrences of $\forall_i$. Then m is a* question *if v is negative, and an* answer *if v is positive.*

The (ternary) scoped question/answer relation on moves is defined as follows:

- *$q \rhd_m a$ if there exists t, u, v, w such that u contains no occurrences of l and u, v, w contain no occurrences of $\forall_i$, and $m = t\forall_i \cdot u$, $q = t\forall_i \cdot v\bullet_i$ and $a = \forall_i \cdot w\bullet_i$.*

Definition 4. *A* game *is a subset $A \subseteq \mathcal{U}$ such that if $u\bullet_i \in A$ is a hole move then u contains no occurrences of $\forall_j$ for $j \leq i$, and if $u \sqsubseteq m \in A$ then $m = u\bullet_i$.*

A is a *n-context game* if $H_A(i)$ (the set of i-hole moves in A) is empty for $i > n$.

2.1 Examples

Given a n-context game A and element $a \in \mathcal{A}$, we write $a.A$ for $\{am \mid m \in A\}$.

- We denote the empty game (the empty set of moves) by 1.
- For each $i \leq n$, $\{\bullet_i\}$ (or just $\bullet_i$) is the n-context game containing the single (Opponent hole) move $\bullet_i$, corresponding to a free type variable.
- For any indexing set $I \subseteq \mathbb{N}$, the disjoint union $\bigcup_{i \in I} i.A$ corresponds to an indexed product or record type.
- Let $A \Rightarrow B$ denote the game $l.A \cup r.B$ (the disjoint union of A and B, with the initial moves of B becoming enablers for the initial moves of A, and the moves of A switched between Player and Opponent.

– If A is a n-context game, $\forall_n.A$ is a $n-1$-context game; the Opponent i-hole moves of A become questions, scoped by the O-initial moves of A, and the Player i-holes of A become their answers.

In this way, we can interpret an (unbounded) second-order type (in which variables and their binders are represented as occurrences of $\bullet$ and $\forall$ with de Bruijn-style indices) directly as a game in which the moves correspond to the paths through the syntax tree of the type from the root to a leaf node (bound or free variable).

For example, suppose A and B are n-context games. The game $\forall_{n+1}((A \Rightarrow \bullet_{n+1}) \Rightarrow (B \Rightarrow \bullet_{n+1}) \Rightarrow \bullet_{n+1})$ (corresponding to the System F type $\forall X_{n+1}.(A \to X_{n+1}) \Rightarrow (B \to X_{n+1}) \to X_{n+1}$) consists of the initial Opponent question $\forall_{n+1} rr\bullet_{n+1}$, which enables and scopes its two (Player) answers $\forall_{n+1} rlr\bullet_{n+1}$ and $\forall_{n+1} lr\bullet_{n+1}$, and the moves $\forall_{n+1} ll.A$ and $\forall_{n+1} rll.B$, of which the moves of the form $\forall_{n+1} llm$ and $\forall_{n+1} rllm$ where m is initial in A or B respectively, are enabled by $\forall_{n+1} rlr\bullet_{n+1}$ and $\forall_{n+1} lr\bullet_{n+1}$, respectively. In other words, this is the *lifted sum* game [16], used to define a computational monad, and thus a call-by-value semantics in [2], which we will extend with bounded polymorphism.

2.2 Legal Sequences and Strategies

Interaction in the game may be seen as alternately choosing leaf nodes of this syntax tree, according to rules we now describe. Define the *pending question prefix* of a sequence of moves t over a 0-context arena A (if any):

– $\mathsf{pending}(sq) = sq$, if q is a question,
– $\mathsf{pending}(sa) = \mathsf{pending}(s')$, if a is an answer, and $\mathsf{pending}(s) = s'q$.

Definition 5. *A* legal sequence *t on A is a finite sequence of moves in A, alternating between Opponent and Player moves, equipped with a unique* justification pointer *from each non-initial move to a preceding move which enables it, such that:*

– *If $sqs'a \sqsubseteq t$, where q is the pending question in sqs' then there exists a move m in s which hereditarily justifies both q and a, such that $q \rhd_m a$ (the* bracketing condition*).*

t is single threaded *if it contains at most one initial move, in which case any pointers from moves enabled by (and therefore justified by) the initial move are considered* implicit*.*

A *strategy* for a 0-context game A is a non-empty, even-prefix-closed, even-branching set of single-threaded legal sequences on A. For each n, we define a category $\mathcal{G}(n)$ in which objects are n-context games and morphisms from A to B are strategies on $\forall_1 \ldots \forall_n (A \Rightarrow B)$ (we will also write $\mathcal{G}(0)$ as just $\mathcal{G}$). To define composition in $\mathcal{G}(n)$, we use the following generalised notion of restriction on justified sequences: given a partial function $f : A \to B$ we may derive a function on sequences over B — $f^* : \mathcal{U}^* \to \mathcal{U}^*$ applies f pointwise to the moves on which

it is defined, and omits moves on which it is not defined. This extends to justified sequences: $f(n)$ points to $f(m)$ in $f^*(s)$ iff $f(m)$ and $f(n)$ are both defined and n points to m in s. Where f is evident from the context, we shall write $s{\restriction}B$ for $f^*(s)$.

Let $\sigma^\dagger$ be the least set of legal sequences containing σ and closed under interleaving. The composition of $\sigma : A \to B$ with $\tau : B \to C$ is defined: $\sigma;\tau = \{s \in L_{\forall(A \Rightarrow C)} \mid \exists t \in A + B + C.t{\restriction}\forall(A \Rightarrow B) \in \sigma^\dagger \wedge t{\restriction}\forall(B \Rightarrow C) \in \tau \wedge s = t \restriction \forall(A \Rightarrow C)\}$. The identity on A is the set of copycat sequences $\{t \in L_{\forall(A \Rightarrow A)} \mid \forall s \sqsubseteq^E t.s{\restriction}A^+ = s{\restriction}A^-\}$.

For any n-context games A and B, $A \times B = 0.A \cup 1.B$ is a Cartesian product of A and B in $\mathcal{G}(n)$, and $A \Rightarrow B$ is an internal hom (so $\mathcal{G}(n)$ is a Cartesian closed category). For each n, if A is a n-context game, and B is a $n+1$ context game, there is an evident isomorphism between the arenas $\forall_{n+1}(A \Rightarrow B)$ and $A \Rightarrow \forall_{n+1}B$, yielding an evident natural correspondence between $\mathcal{G}(n+1)(J_{n+1}(A), B)$ and $\mathcal{G}(n)(A, \forall_{n+1}.B)$ — i.e. the inclusion $J_{n+1} : \mathcal{G}(n) \to \mathcal{G}(n+1)$ has $\forall_{n+1}$ as its *right adjoint* [15].

2.3 Instantiation

Let A be a n-context game, and B and C be i-context games, with $i \leq n$. The *instantiation* of B and C into the (negative and positive) i-holes of A (respectively) is defined $A(B,C)_i =:$

$$A \backslash H_A(i) \cup \{w \cdot m \mid w\bullet_i \in H_A(i)^- \wedge m \in B\} \cup \{w \cdot m \mid w\bullet_i \in H_A(i)^+ \wedge m \in C\}.$$

This operation behaves as one would expect for a substitution[1]: e.g. if $A = \bullet_1 \Rightarrow \bullet_1 \Rightarrow \bullet_1$ then $A(B,C)_1 = B \Rightarrow B \Rightarrow C$. Moreover, each morphism in $\mathcal{G}(n)(A,B)$ corresponds to a *generic* family of morphisms from $A(C,C)$ to $B(C,C)$ for each n-context game C. These are defined (as in [15]) using the notion of instantiation of a game into strategy by playing *copycat* between the arenas plugged into the holes. Suppose t is a legal sequence on $\forall(A(C,C)_i \Rightarrow B(C,C)_i)$, such that $t{\restriction}\forall(A \Rightarrow B)$ is a legal sequence, and $sa \sqsubseteq t{\restriction}\forall(A \Rightarrow B)$, where $a = v\bullet_i$ is a Player answer in $\forall(A \Rightarrow B)$ corresponding to a i-hole move in $A \Rightarrow B$. As t is *well bracketed*, $\mathsf{pending}(s) = s'q$ where $q = u\bullet_i$ is an Opponent question in $\forall(A \Rightarrow B)$ corresponding to an i-hole move in $A \Rightarrow B$. So we may define $t{\restriction}(\mathsf{pending}(s), sa)$ to be the projection of t onto $\forall(C \Rightarrow C)$ of moves which are suffixes of u and v (and hereditarily justified by q and a, respectively).

Definition 6. *Given a strategy $\sigma : A \to B$, let $\sigma[C]_i : A(C,C)_i \to B(C,C)_i$ be the set of sequences $t \in L_{\forall(A(C,C)_i \Rightarrow B(C,C)_i)}$ such that $t{\restriction}\forall(A \Rightarrow B) \in \sigma$ and if $sa \sqsubseteq^E t$, where a is a Player i-hole move in $A \Rightarrow B$, then $t{\restriction}(\mathsf{pending}(s), sa)) \in \mathsf{id}_C$.*

As shown in [15], the $\mathcal{G}(n)$ may be organised as an indexed cartesian closed category of context games, in which the J_n and $\forall_i$ form an indexed adjunction. Here we note the following result from *loc. cit.*.

[1] The condition that i-hole moves contain no occurrence of $\forall_j$ for $j \leq i$ prevents "capture" of the holes.

Lemma 1. *$\_[C]_i$ is an endofunctor on $\mathcal{G}(n)$, such that $\_[C]_i \cdot \_[D]_i = \_[B(C,C)_i]_i$.*

3 Generic Strategies as Dinatural Transformations

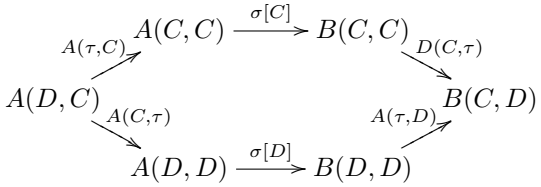
In addition to the structure defined in [15], instantiation also acts on strategies as a (mixed variance) functor (which we will require in order to define our semantics of bounded quantification) in the following sense:

Definition 7. *Given an Opponent hole $w\bullet_i \in H_A^+(i)$, let $p_{w\bullet_i}$ be the partial projection on moves sending $\forall lw \cdot n$ to $\forall ln$ and $\forall rw \cdot n$ to $\forall rn$, so that it projects a move in $\forall(A(B,C)_i \Rightarrow A(B',C')_i)$ into $\forall(C \Rightarrow C')$. If $m \in H_A^-(i)$ is a Player hole move, then let p_m be the corresponding projection from $\forall(A(B,C)_i \Rightarrow A(B',C')_i)$ into $\forall(B' \Rightarrow B)$.*

Now, given morphisms $\sigma : B' \to B, \tau : C \to C'$ in $\mathcal{G}(n)$, we may define $A(\sigma,\tau)_i : A(B,C) \to A(B',C')$ to be the set of even-length sequences $t \in L_{\forall(A(B,C)\Rightarrow A(B',C'))}$ such that for all even prefixes $s \sqsubseteq t$, $s \restriction \forall(A \Rightarrow A) \in \mathsf{id}_A$ and for any hole move $m \in H_i^+(A)$, $p_m^*(s) \in \sigma^\dagger$ and for any $m \in H_n^-(A)$, $p_m^*(s) \in \tau^\dagger$.

Proposition 1. *For any n-context game A, $A(\_,\_)_i$ is a functor from $\mathcal{G}(n)^{op} \times \mathcal{G}(n)$ to $\mathcal{G}(n)$.*

One might hope that for any generic n-context strategy $\sigma : A \to B$, the family $\{\sigma[C] : A(C,C)_i \to B(C,C)_i\}$ is a *dinatural transformation* [4] from $A(\_,\_)_i$ to $B(\_,\_)_i$ — i.e. that for any morphism $\tau : C \to D$, the following diagram commutes:



However, this is not the case:

Proposition 2. *$\sigma[\_]_i$ is not a dinatural transformation from $A(\_,\_)_i$ to $B(\_,\_)_i$ in general.*

Proof. Take, for example, $A = \forall_1(\bullet_1 \Rightarrow \bullet_1)$, $B = \bullet_1 \Rightarrow \bullet_1$ and $\sigma : \forall_1(\bullet_1 \Rightarrow \bullet_1) \to (\bullet_1 \to \bullet_1)$ to be the counit of the adjunction $J_1 \dashv \forall_1$ (i.e. the copycat strategy between A and B). At any game, $\bot$ is the strategy containing only the empty sequence. Then $A(\bot,1);\sigma[1];B(1,\bot) = \bot$, but $A(\bullet_1,\bot);\sigma[\bullet_1];B(\bot,\bullet_i) \neq \bot$ (it contains a response to the initial move).

The generic strategies (in Hughes' model of System F) which do correspond to dinatural transaformations are characterized in [10], to which we refer for further discussion. Here we take an alternative approach by showing that $\sigma[\_]_i$ *is* dinatural with respect to a subcategory of *linear morphisms*.

Definition 8. *A morphism $\sigma : A \to B$ is* linear *if for every initial move m in B there is an initial move m' in A such that $msm' \in \sigma$ if and only if $s = \varepsilon$ (so every non-empty sequence $s \in \sigma$ contains exactly one initial move from A).*

It is easy to see that the composition of linear morphisms is linear, and so for each n we may form a subcategory $\mathcal{G}^L(n)$ of $\mathcal{G}(n)$ consisting of n-context games and linear morphisms, with an inclusion $J_L : \mathcal{G}^L(n) \to \mathcal{G}(n)$. Note that $\times$ is a cartesian product in $\mathcal{G}^L(n)$.

Proposition 3. *$\sigma[\_]_i$ is a dinatural transformation from $A \cdot J_L$ to $B \cdot J_L$.*

In other words, for any *linear* morphism $\tau : C \to D$ the dinaturality hexagon does commute. We prove this by defining the instantiation of τ directly into σ, i.e. $\sigma[\tau]_i : A(B,B)_i \to D(C,C)_i$ by replacing the identity on B with τ in Definition 6 and showing that $A(C,\tau)_i; \sigma[C]_i; D(\tau,C)_i$ and $A(\tau,B)_i; \sigma[B]_i; D(B,\tau)_i$ are both equal to $\sigma[\tau]_i$. Although the composition of dinatural transformations is not, in general, dinatural [4], by Proposition 1, those which arise by instantiation of generic strategies do compose.

4 Semantics of Bounded Quantification

We may now use the functors and dinatural transformations derived from instantiation to represent bounded quantification. A subtyping constraint of the form $X_i \leq T$, where X_i is a type variable, corresponds to the ability to subsume any term of type X_i into the type T. Thus, in an environment where such a constraint holds, *negative occurences* of X_i may be represented as the game $[\![T]\!] \times \bullet_i$, from which there is a canonical coercion (left projection) into $[\![T]\!]$. On the other hand, *positive occurrences* of X_i are simply represented as $\bullet_i$ (we can't, for example, decompose X_i into the bound T and an "unknown" extension). The difficulty is to define a notion of composition between morphisms by "unifying" this different treatment of positive and negative occurrences[2]. The key to doing so is the following operation:

Definition 9. *Let A be an $n-1$-context game. Given $\sigma : B(\bullet_n, A \times \bullet_n)_n \to C(A \times \bullet_n, \bullet_n)_n$, define $\widehat{\sigma} : B(A \times \bullet_n, A \times \bullet_n)_n \to C(A \times \bullet_n, A \times \bullet_n)_n$ as follows: ($\delta : A \to A \times A = \langle A, A \rangle$).*

$$B(A \times \bullet_n, A \times \bullet_n)_n \xrightarrow{B(A \times \bullet_n, \delta \times \bullet_n)_n} B(A \times \bullet_n, A \times A \times \bullet_n)_n \xrightarrow{\sigma[A \times \bullet_n]_n} C(A \times A \times \bullet_n, A \times \bullet_n)_n \xrightarrow{C(\delta \times \bullet_n, A \times \bullet_n)_n} C(A \times \bullet_n, A \times \bullet_n)_n$$

We illustrate by describing a setting corresponding to an environment containing a single constrained type variable. Given a $n-1$-context game A, we may define a category $\mathcal{G}_A$ in which objects are n-context games and morphisms from B to C are the morphisms from $B(\bullet_n, A \times \bullet_n)$ to $C(A \times \bullet_n, \bullet_n)$ in $\mathcal{G}$. The composition of

[2] Representing negative *and* positive occurrences of X_i as $A \times \bullet_i$ leads to a simple interpretation of composition, but to the failure of bounded universal quantification to be antitone in type bounds — i.e. we get a model of "*kernel* $F_{\leq}$" [6].

$\sigma \in \mathcal{G}_A(B,C)$ with $\tau \in \mathcal{G}_A(C,D)$ is the composition of $B(\pi_r, A\times\bullet); \widehat{\sigma} : B(\bullet_n, A\times \bullet_n) \to C(A\times\bullet_n, A\times\bullet_n)$ with $\widehat{\tau}; D(A\times\bullet_n, \pi_r) : C(A\times\bullet_n, A\times\bullet_n) \to D(A\times\bullet_n, \bullet_n)$ in $\mathcal{G}$. The identity on B in $\mathcal{G}_A$ is $B(\pi_r, \pi_r)$.

Lemma 2. *The operation $\widehat{(\_)}$ satisfies the following properties:*

1. $\widehat{B(\pi_r, \pi_r)} = B(A \times \bullet, A \times \bullet)$.
2. *For* $\sigma : B(\bullet, A \times \bullet) \to C(A \times \bullet, \bullet)$, $\sigma = B(\pi_r, A \times \bullet); \widehat{\sigma}; C(A \times \bullet, \pi_r)$.
3. *For* $\tau : C(\bullet, A \times \bullet) \to D(A \times \bullet, \bullet)$, $\widehat{\sigma}; \widehat{\tau} = B(\pi_r, A \times \bullet); \widehat{\sigma; \widehat{\tau}}; D(A \times \bullet, \pi_r)$.

Proposition 4. *$\mathcal{G}_A$ is a well-defined category.*

Proof. That $B(\pi_r, \pi_r)$ is an identity follows from Lemma 2 (1). For associativity of composition, suppose $\rho \in \mathcal{G}_A(B,C)$, $\sigma \in \mathcal{G}_A(C,D)$ and $\tau \in \mathcal{G}_A(D,E)$. Then $(\rho;\sigma);\tau = B(\pi_r, A \times \bullet); B(\pi_r, A \times \bullet); \widehat{\widehat{\rho}; \widehat{\sigma}}; C(A \times \bullet, \pi_r); \widehat{\tau}; D(\pi_r, A \times \bullet, \pi_r)$. By Lemma 2 (3), this is equal to $B(\pi_r, A \times \bullet); \widehat{\rho}; \widehat{\sigma}; \widehat{\tau}; E(\pi_r, A \times \bullet, \pi_r)$, which is similarly equal to $\rho; (\sigma; \tau)$.

We also have operations on $\mathcal{G}_A$ corresponding to bounded quantification and instantiation: let $\forall^A C = \forall.C(A \times \bullet, \bullet)$. Then for any 0-context game B, there is an evident isomorphism of arenas between $\forall(B \Rightarrow C(A \times \bullet, \bullet))$ and $D \Rightarrow \forall^A C$ yielding:

Proposition 5. *The inclusion $J_A : \mathcal{G} \to \mathcal{G}_A$ is left adjoint to $\forall^A : \mathcal{G}_A \to \mathcal{G}$.*

Definition 10. *Given a 0-context arena D, and a linear morphism $c : D \to A$, representing a coercion of D into A, we define the* bounded instantiation *functor sending $\sigma : B \to C$ to $\sigma\{c, D\} : B(D,D) \to C(D,D)$:*

$$B(D,D) \overset{B(D,\langle c,D\rangle)}{\longrightarrow} B(D, A \times D) \overset{\sigma[D]}{\longrightarrow} C(D \times A, D) \overset{C(\langle c,D\rangle, D)}{\longrightarrow} C(D,D)$$

Proposition 6. *$\_\{c, D\}$ is a (Cartesian closed) functor from $\mathcal{G}_A$ to $\mathcal{G}$.*

We iterate these constructions to define categories of context games for representing types with multiple bounded free variables.

Definition 11. *A* bounding sequence *of context games $A_1, \ldots, A_n$ is one in which each A_i is a $i-1$-context game. Given such a sequence, we define categories $\mathcal{G}(n)_{A_1,\ldots,A_i}$ (and its subcategory $\mathcal{G}(n)^L_{A_1,\ldots,A_i}$) for each $i \leq n$, in which objects are n-context games and:*

- $\mathcal{G}(n)_\varepsilon = \mathcal{G}(n)$.
- *In $\mathcal{G}(n)_{A_1,\ldots,A_{i+1}}$, morphisms from B to C are the morphisms from $B(\bullet_i, A_{i+1}\times \bullet_{i+1})_{i+1}$ to $C(A_{i+1} \times \bullet_{i+1}, \bullet_{i+1})$ in $\mathcal{G}(n)_{A_1,\ldots,A_i}$, with the composition of $\sigma : B \to C$ with $\tau : C \to D$ defined to be the composition of $B(\pi_r, A_{i+1} \times \bullet_{i+1})_{i+1}; \widehat{\sigma}_i$ with $\widehat{\tau}_i; C(A_{i+1} \times \bullet_i, \pi_r)$ in $\mathcal{G}(n)_{A_1,\ldots,A_i}$. The identity on B is $B(\pi_r, \pi_r)_1, \ldots, (\pi_r, \pi_r)_{i+1}$.*

Proof (by induction) that each $\mathcal{G}_{A_1,\ldots,A_i}$ is a (Cartesian closed) category follows the proof of Proposition 4 above. For any bounding sequence $A_1,\ldots,A_n,B$ we have a right adjoint $\forall^B_{n+1}$: to the inclusion functor $J^B_{n+1} : \mathcal{G}(n)_{A_1,\ldots,A_n} \to \mathcal{G}(n+1)_{A_1,\ldots,A_n,B}$.

Definition 12. *For any $i < n$, i-context game C and $c \in \mathcal{G}(n)^L_{A_1,\ldots,A_{i-1}}(C, A_i)$, the instantiation functor $\_\{c, C\}_i$ from $\mathcal{G}(n)_{A_1,\ldots,A_i}$ to $\mathcal{G}(n)_{A_1,\ldots,A_{i-1}}$ sends D to $D(C,C)_{n+1}$ and $\sigma : D \to E$ to $D(C, \langle c, C\rangle)_i; \sigma[C]_i; E(\langle c, C\rangle, C)_i$. This lifts to a functor from $\mathcal{G}(n+1)_{A_1,\ldots,A_n}$ to $\mathcal{G}(n)_{A_1,\ldots,A_{i-1},A_{i+1}(A_i,A_i),\ldots,A_j(A_i,A_i)}$ for each $n \geq j > i$, since e.g. $D(A_j \times \bullet_j, \bullet_j)(C,C)_i = D(C,C)_i(A_j(C,C)_i \times \bullet_j, \bullet_j)$*

Instantiation preserves cartesian closed structure, and also commutes with the second-order structure, in the following sense:

Proposition 7. $\_\{c, C\}_i \cdot \forall^A_j = \forall^{A(C,C)}_j \cdot \_\{c, C\}_i$.

5 A Stateful Language with Bounded Quantification

So far, we have described subtyping constraints on variables, and bounded quantification, without explicitly defining or discussing the semantics of the subtyping relation on games, beyond the assumption that it yields a coercing linear morphism from a subtype to its supertype. In this sense, our semantics is somewhat independent of any particular notion of subtyping, which can be supplied either *intensionally*, by a formal system of derivations for subtyping judgments with their interpretation as coercing morphisms, or *extensionally*, arising from a relation on objects in our category with a corresponding notion of coercion (for example, Chroboczek's notion of subtyping for games [8]). Taking the former approach, our interpretation of subtyping constraints, bounded quantification and instantiation is sufficient to model typing systems such as Cardelli and Wegner's *Fun* [6] or System $F_{\leq}$[7]. However, since we also wish to describe the behaviour of (possibly side effecting) programs in this context, we define a stateful, call-by-value metalanguage, $\mathcal{L}_{\leq}$, with such a typing system. (The main difference with respect to $F_{\leq}$as a typing system is the inclusion of explicit object (i.e. record) types: although record types can be encoded in $F_{\leq}$, the semantics of this encoding in our model is not faithful to the direct interpretation of such types as products).

In essence, $\mathcal{L}_{\leq}$ and its model are an extension with subtyping and bounded quantification of the metalanguage with general references, $\mathcal{L}$, defined with its games model by Abramsky, Honda and McCusker [3] (from which there is a simple translation into $\mathcal{L}_{\leq}$). Raw types of $\mathcal{L}_{\leq}$ are generated by the grammar:

$$S ::= \texttt{unit} \mid \texttt{nat} \mid S \to O \mid \forall X \leq O.O \qquad\qquad O ::= X \mid [l_1 : S_1, \ldots, l_n : S_n]$$

(where the l_i are drawn from an unbounded collection of labels). Type variables are restricted to range over the (concrete and variable) *object types* O, which are assigned to records containing (possibly side-effecting) methods typed according

to the values they may return. We can wrap any method type S in an object type $[S]$ with a single field, for which we omit the label (corresponding to the lifting operation of the computational λ-calculus), and "thunk" an object type O as the method type $\underline{O} = \mathtt{unit} \to O$. Judgments $\Sigma \vdash T$ (T is a well-formed type with free variables in Σ) are derived via the rules:

$$\frac{}{\Sigma,X,\Sigma' \vdash X} \qquad \frac{}{\Sigma : B} \qquad \frac{\Sigma \vdash S_1, \ldots, \Sigma \vdash S_n}{\Sigma \vdash [l_1:S_1, \ldots, l_n:S_n]} \qquad \frac{\Sigma \vdash S \qquad \Sigma \vdash O}{\Sigma \vdash S \to O} \qquad \frac{\Sigma \vdash O \qquad \Sigma, X \vdash P}{\Sigma \vdash \forall X \leq O.P}$$

$$\frac{|\Theta| \vdash T}{\Theta \vdash T \leq T} \qquad \frac{\Theta \vdash T \leq T' \quad \Theta \vdash T' \leq T''}{\Theta \vdash T \leq T''} \qquad \frac{\Theta \vdash S_1 \leq S_1' \ldots \Theta \vdash S_m \leq S_m' \qquad m \leq n}{\Theta \vdash [l_1:S_1, \ldots, l_n:S_n] \leq [l_1:S_1', \ldots, l_m:S_m']}$$

$$\frac{}{\Theta, X \leq O, \Theta' \vdash X \leq O} \qquad \frac{\Theta \vdash S' \leq S \qquad \Theta \vdash O \leq O'}{\Theta \vdash S \to O \leq S' \to O'} \qquad \frac{\Theta \vdash O' \leq O \qquad \Theta, X \leq O \vdash P \leq P'}{\Theta \vdash \forall X \leq O.P \leq \forall X \leq O'.P'}$$

Table 1: Subtyping Judgments for $\mathcal{L}_{\leq}$

A *subtyping context* is a sequence of subtyping assumptions $X_1 \leq O_1, \ldots, X_n \leq O_n$ such that $X_1, \ldots, X_{i-1} \vdash O_i$ for each $1 \leq i \leq n$. $|X_1 \leq O_1, \ldots, X_n \leq O_n|$ is the sequence $X_1, \ldots, X_n$. *Subtyping judgments* $\Theta \vdash T \leq T'$ — where $\Theta = X_1 \leq O_1, \ldots, X_n \leq O_n$ is a subtyping context and $|\Theta| \vdash T$ and $|\Theta| \vdash T'$ — are derived according to the rules in Table 1. Note that there is a $\leq$-greatest object type —

$$\frac{}{\Theta;\Gamma,x:S,\Gamma' \vdash x:S} \qquad \frac{\Theta;\Gamma \vdash M:O \qquad \Theta \vdash O \leq O'}{\Theta;\Gamma \vdash M:O'}$$

$$\frac{\Theta;\Gamma,x:S \vdash M:O}{\Theta;\Gamma \vdash \lambda x^S.M:S \to O} \qquad \frac{\Theta;\Gamma \vdash M:S \to O \qquad \Theta;\Gamma \vdash N:S}{\Theta;\Gamma \vdash M\,N:O}$$

$$\frac{\Theta;\Gamma \vdash M_1:S_1, \ldots, \Theta;\Gamma \vdash M_1:S_n}{\Theta;\Gamma \vdash \{l_1 = M_1, \ldots, l_n = M_n\}:[l_1:S_1, \ldots, l_n:S_n]} \qquad \frac{\Theta;\Gamma \vdash M:[l_1:S_1, \ldots, l_n:S_n]}{\Theta;\Gamma \vdash M.l_i:S_i}$$

$$\frac{\Theta, X \leq O;\Gamma \vdash M:P}{\Theta;\Gamma \vdash \Lambda X \leq O.M:\forall X \leq O.P}\, X \notin FV(\Gamma) \qquad \frac{\Theta;\Gamma \vdash M:\forall X \leq O.P \qquad \Theta \vdash O' \leq O}{\Theta;\Gamma \vdash M\{O'\}:P[O'/X]}$$

Table 2: Typing Judgments for $\mathcal{L}_{\leq}$

the record type with no labels, for which we will write $\top$. So (as in $F_{\leq}$) we can represent unbounded quantification $\forall X.O$ as $\forall X \leq \top.O$. There is no greatest method type, however. Nor do we capture inclusion of values in the subtyping relation, but leave it as a possible extension. The point is that the distinction

between subtyping as inclusion of values, versus extension of records, which is blurred in the pure type theory $F_{\leq}$, becomes necessarily more significant in a setting with side effects. We define the type $\mathtt{var}[S]$ of variables storing values of type S to be the object type with two methods, $[\mathtt{get} : S, \mathtt{set} : S \to [\mathtt{unit}]]$.

Typing judgments $\Theta; \Gamma \vdash M : T$ — where Θ is a subtyping context, Γ is a sequence of typing assumptions $x_1 : S_1, \ldots, x_n : S_n$ and T is a type such that $|\Theta| \vdash S_1, \ldots, S_n, T$ — are derived according to the rules in Table 2 extended with constants for arithmetic, conditional branching ($\mathtt{If0}_S : \mathtt{nat} \to [S \to [S \to [S]]]$) and reference declaration ($\mathtt{new}_S : \mathtt{var}[S]$).

The operational semantics of $\mathcal{L}_{\leq}$ is based on that given for $\mathcal{L}$ in [3]. We extend the language with an unbounded set of constants $a, b, \ldots$ representing location names, and define the *values* of this extended language by the grammar:
$V ::= \mathtt{n} \mid a \mid a.\mathtt{set} \mid \{l_1 = M_1, \ldots, l_n = M_n\} \mid \lambda x.M \mid \Lambda X.M$
An environment $\mathcal{E}$ is a pair of a set of location names, and a partial function from this to the set of values. The evaluation relation $\Downarrow$ between pairs $(M, \mathcal{S})$ of a closed term and environment and $(V, \mathcal{S}')$ of a value and environment is defined in Table 3.

$$\frac{}{V,\mathcal{S}\Downarrow V,\mathcal{S}} \qquad \frac{M,\mathcal{S}\Downarrow \mathtt{suc}(\mathtt{n}),\mathcal{S}'}{\mathtt{pred}(M),\mathcal{S}\Downarrow \mathtt{n},\mathcal{S}'}$$

$$\frac{M,\mathcal{S}\Downarrow 0,\mathcal{S}'}{\mathtt{If0}_S\, M,\mathcal{S}\Downarrow\{\lambda x.\{\lambda y.\{x\}\}\},\mathcal{S}'} \qquad \frac{M,\mathcal{S}\Downarrow \mathtt{suc}(n),\mathcal{S}'}{\mathtt{If0}_S\, M,\mathcal{S}\Downarrow\{\lambda x^S.\{\lambda y^S.\{y\}\}\},\mathcal{S}'}$$

$$\frac{M,\mathcal{S}\Downarrow \lambda x^S.M',\mathcal{S} \qquad N,\mathcal{S}'\Downarrow U,\mathcal{S}'' \qquad M'[U/x],\mathcal{S}''\Downarrow V,\mathcal{S}'''}{M\,N,\mathcal{S}\Downarrow V,\mathcal{S}'''}$$

$$\frac{M,\mathcal{S}\Downarrow\{l_1{=}N_1,\ldots,l_k{=}N_k\},\mathcal{S}' \qquad N_i,\mathcal{S}'\Downarrow U,\mathcal{S}''}{M.l_i,\mathcal{S}\Downarrow U,\mathcal{S}''}$$

$$\frac{M,\mathcal{S}\Downarrow \Lambda X{\leq}O.M',\mathcal{S}' \qquad M'[P/X],\mathcal{S}'\Downarrow V,\mathcal{S}''}{M\{P\},\mathcal{S}\Downarrow V,\mathcal{S}''}$$

$$\frac{}{\mathtt{new},\mathcal{S}\Downarrow a,\mathcal{S}\cup\{a\}}\, a \notin \mathcal{S} \qquad \frac{M,\mathcal{S}\Downarrow a,\mathcal{S}'}{M.\mathtt{get},\mathcal{S}\Downarrow V,\mathcal{S}'}\, \mathcal{S}'(a) = V$$

$$\frac{M,\mathcal{S}\Downarrow a,\mathcal{S}'}{M.\mathtt{set},\mathcal{S}\Downarrow a.\mathtt{set},\mathcal{S}'} \qquad \frac{M,\mathcal{S}\Downarrow a.\mathtt{set},\mathcal{S}' \qquad N,\mathcal{S}'\Downarrow V,\mathcal{S}''}{M\,N,\mathcal{S}\Downarrow\{()\},\mathcal{S}''[a{\mapsto}V]}$$

Table 3: Operational Semantics for $\mathcal{L}_{\leq}$

5.1 Denotational Semantics

The denotational semantics of of $\mathcal{L}_{\leq}$ extends the games interpretation of references [3] with our model of bounded quantification. We define the denotations of types and the subtyping relation first.

Following [2,3], we will interpret method types as *indexed families* of games. That is, we interpret types with n free variables in the category $\mathsf{Fam}(\mathcal{G}(n))$ (the coproduct completion of $\mathcal{G}(n)$) in which objects are set-indexed families of objects of $\mathcal{G}(n)$, and morphisms from $\{A_i \mid i \in I\}$ to $\{B_j \mid j \in J\}$ are pairs $(f : I \to J, \{\sigma_i : A_i \to A_{f(i)} \mid i \in I\}$ of a reindexing function and an indexed family of morphisms from $\mathcal{G}(n))$. $\mathsf{Fam}(\mathcal{G}(n))$ is Cartesian closed and has (small) coproducts [2]. Defining $\Sigma\{A_i \mid i \in I\} = \forall_{n+1}(\Pi_{i\in I} J_{n+1}(A_i) \Rightarrow \bullet_{n+1}) \Rightarrow \bullet_{n+1}$ (as already observed in Section 2, this is concretely the same as the "lifted sum" construction used to interpret $\mathcal{L}$ in [3]) we have [15]:

Proposition 8. *Inclusion of $\mathcal{G}^L(n)$ in $\mathsf{Fam}(\mathcal{G}(n))$ is right adjoint to $\Sigma\_$.*

This yields a strong monad on $\mathsf{Fam}(\mathcal{G}(n)\}$. Object types are interpreted as objects of $\mathcal{G}(n)$ and method types as objects of $\mathsf{Fam}(\mathcal{G}(n))$, as follows:

$$\begin{array}{ll}
[\![\mathtt{nat}]\!]_n = \{1 \mid i \in \mathbb{N}\} & [\![\mathtt{unit}]\!] = \{1\} \\
[\![[l_1 : S_1, \ldots, l_m : S_m]]\!]_n = \Pi_{i\leq m}\Sigma[\![S_i]\!]_n & [\![S \to O]\!]_n = [\![S]\!]_n \Rightarrow \{[\![O]\!]_n\} \\
[\![X_i]\!]_n = \bullet_i & [\![\forall X_{n+1} \leq O.P]\!]_n = \forall_n^{[\![O]\!]}[\![P]\!]_{n+1}
\end{array}$$

Subtyping contexts $X_1 \leq O_1, \ldots, X_n \leq O_n$ are interpreted as the corresponding bounding sequences $[\![O_1]\!]_0, \ldots, [\![O_n]\!]_{n-1}$, and *subtyping judgments* $\Theta \vdash T \leq T'$ are interpreted as linear coercing morphisms from T to T' in $\mathsf{Fam}(\mathcal{G}^L_{[\![\Theta]\!]})$. The key point here is that while $\Theta \vdash T \leq T'$ may have multiple derivations, they all correspond to the same morphism.

$$\begin{array}{l}
[\![\Theta \vdash T \leq T]\!] = \mathsf{id}_{[\![T]\!]} \\
[\![\Theta \vdash T \leq T'']\!] = [\![\Theta \vdash T \leq T']\!]; [\![\Theta \vdash T' \leq T'']\!] \\
[\![\Theta, X \leq O, \Theta' \vdash X \leq O]\!] = J_{\Theta'}(\pi_l) \\
[\![\Theta \vdash [l_1 : S_1, \ldots, l_n : S_n] \leq [l_1 : T_1, \ldots, l_m : T_m]]\!] = \langle \pi_1; [\![\Theta \vdash S_1 \leq T_1]\!], \ldots, \pi_m; [\![\Theta \vdash S_m \leq T_m]\!]\rangle \\
[\![\Theta \vdash S \to O \leq S' \to O']\!] = [\![\Theta \vdash S' \leq S]\!] \to [\![\Theta \vdash O \leq O']\!]_\Theta \\
[\![\Theta \vdash \forall X_{n+1} \leq O.P \leq \forall X \leq O'.P']\!] = [\![P]\!][[\![\Theta \vdash O' \leq O]\!] \times \bullet_{n+1}]\{[\![\Theta, X_{n+1} \leq O \vdash P \leq P']\!]\}
\end{array}$$

Table 4: Denotational Semantics of Subyping Judgments

Proposition 9. *For any derivable subtyping judgment $\Theta \vdash T \leq T'$, there is a unique morphism $[\![\Theta \vdash T \leq T']\!]$ satisfying the rules in Table 4.*

Proof. We define a notion of *canonical derivation* for subtyping judgments by replacing the transitivity and reflexivity rules with:

$$\frac{}{\Theta \vdash X \leq X} \qquad\qquad \frac{\Theta, X \leq O, \Theta' \vdash O \leq O'}{\Theta, X \leq O \vdash X \leq O'}$$

(This is essentially Curien and Ghelli's deterministic system for deriving subtyping judgments for $F_{\leq}$[9].) We then prove that every derivation is denotationally equivalent to a canonical derivation.

5.2 Semantics of Terms

Terms in context are interpreted as morphisms in $\mathsf{Fam}(\mathcal{G}_{[\![\Theta]\!]})$ — terms of method type $\Theta; \Gamma \vdash M : S$ denote morphisms from $[\![\Gamma]\!]$ to $\Sigma[\![S]\!]$, and terms $\Theta; \Gamma \vdash M : O$ of object type denote morphisms from $[\![\Gamma]\!]$ to $[\![O]\!]$. The operations of $\mathcal{L}$ are interpreted as in the computational λ-calculus, and the constant `new` as the reference cell strategy defined in [3], while subsumption, universal quantification and instantiation are interpreted as follows:
$[\![\Theta; \Gamma \vdash M : O']\!] = [\![\Theta; \Gamma \vdash M : O]\!]; [\![\Theta \vdash O \leq O']\!]$
$[\![\Theta; \Gamma \vdash M : \forall X \leq O.P]\!] = \forall_{[\![O]\!]_\Theta} [\![\Theta, X \leq O; \Gamma \vdash M : P]\!]$
$[\![\Theta; \Gamma \vdash M\{O'\} : P[O'/X]]\!] = [\![\Theta; \Gamma \vdash M : \forall X \leq O.P]\!]\{[\![\Theta \vdash O' \leq O]\!], [\![O']\!]_{|\Theta|}\}$

As in the semantics of subtyping judgments, these rules show how to interpret each *derivation* of a typing judgment as a morphism. We need to show that any derivation for a given term in context yields the same denotation.

Proposition 10. *For any derivable typing judgment $\Theta; \Gamma \vdash M : T$, there is a unique morphism $[\![\Theta; \Gamma \vdash M : T]\!]$.*

Proof. This follows the proof in [5]. Use of the subsumption rule generates multiple derivations of the same typing judgment, which are shown to be equivalent by extending the language with a constant $\mathsf{convert} : \forall X.[\forall Y \leq X.\underline{Y} \to X]$ (denoting the corresponding coercion), and replacing all uses of the subsumption rule with explicit coercions using $\mathsf{convert}$. Using the (di)naturality properties of $\mathsf{convert}$, we show that any two terms which correspond to the same term of $\mathcal{L}_{\leq}$ obtained by erasing all occurrences of $\mathsf{convert}$ have the same denotation.

We prove *soundness* of the operational rules using the properties of the computational λ-calculus, the equations relating the cell strategy to declaration, assignment and derefering in $\mathcal{L}$ [3] together with the properties of our semantics of bounded quantification (i.e. β-reduction for type-instantiation, which follows from Proposition 7).

Proposition 11. *If $M, \_ \Downarrow V; \mathcal{S}$ then $[\![M]\!] \neq \bot$.*

We prove *computational adequacy* using the approach described in [15]: defining an approximating semantics in which each cell can be accessed a bounded number of times, for which adequacy follows from the soundness of the operational rules by induction; this implies adequacy for the unbounded semantics by continuity.

Proposition 12 (Computational Adequacy). *$[\![M]\!] \neq \bot$ implies $M \Downarrow$.*

Let $\lesssim_T$ be the *observational preorder* at on closed terms of (closed) type T induced by the operational semantics — i.e. $M \lesssim_T N$ if and only if for all

contexts $C[\_ : T]$, $C[M] \Downarrow$ implies $C[N] \Downarrow$. (Note that if $S \leq T$, terms may be observationally equivalent at type T but not at type S.) By a standard argument from adequacy:

Corollary 1. *If* $[\![M : T]\!] \subseteq [\![N : T]\!]$ *then* $M \lesssim_T N$.

6 Full Abstraction

We now consider how closely our model reflects the observational preorder and equivalence. First, we give a full abstraction result for a type-restricted fragment of the language. Define the *concretely bounded types* by the following grammar:

$$S :: B \mid S \to O \mid \forall X \leq [l_1 : S_1, \ldots, l_n : S_n].O \qquad O ::= X \mid [l_1 : S, \ldots, l_n : S]$$

That is, we require that quantification bounds are not type variables. This is a significant constraint — it prevents the inheritance between variable types from being represented directly as a subtyping assumption — but leaves an expressive typing system.

Proof that every finite strategy over a concretely bounded type is definable as a term closely follows the decomposition argument given in [15] for unbounded polymorphism (System F) with general references. The reason for the restriction on types is that we can eliminate negatively occuring concretely bounded quantifiers by instantiating them with a type which extends their bound with a single method of type $\mathtt{unit}$ — i.e. given a strategy $\sigma : [\![\forall X \leq [l_1 : T_1, \ldots, l_n : T_n].O]\!] \to [\![T]\!]$, we can find a strategy $\sigma' : [\![[O[l_1 : T_1, \ldots, l_n : T_n, l' : \mathtt{unit}]/X]]\!] \to [\![T]\!]$ such that if σ' is the denotation of a term $x : [O[[l_1 : T_1, \ldots, l_n : T_n, l' : \mathtt{unit}]/X]] \vdash M : T$, then σ is the denotation of $y : \forall X \leq [l_1 : T_1, \ldots, l_n : T_n] \vdash M[y\{[l_1 : T_1, \ldots, l_n : T_n, l' : \mathtt{unit}]/x]$. Using this property, in conjunction with the decomposition argument given in [15], we may show that:

Proposition 13. *For any concretely bounded type* T*, context* Γ *and subtyping context* Θ*, each finite strategy* $\sigma : [\![\Gamma]\!]_\Theta \to [\![T]\!]_\Theta$ *is the denotation of a term* $\Theta; \Gamma \vdash M : T$.

This is sufficient to establish full abstraction at concretely bounded types, following the argument from finite definability in [3]. For any strategy σ, let $\mathsf{comp}(\sigma)$ be the *complete plays* (sequences with no unanswered questions) of σ.

Theorem 1. *For any terms* $M, N : T$*, where* T *is a concretely bounded type,* $M \lesssim_T N$ *if and only if* $\mathsf{comp}([\![M]\!]) \subseteq \mathsf{comp}([\![N]\!])$.

Finally, we give a counterexample showing that full abstraction does not hold at all types. Consider the strategy consisting of prefixes of the sequence:

$$\begin{array}{ccccccc}
\forall_1^\top(\forall_2^{\bullet_1}.\Sigma & (\bullet_2 & \Rightarrow & \bullet_2)) & \Rightarrow & \Sigma 1) \\
 & & & & & O_Q \\
P_Q & & & & & \\
O_A & & & & & \\
 & & & P_Q & & \\
 & O_A & & & & \\
 & & & & & P_A
\end{array}$$

This strategy is not the denotation of any term $X \leq \top; x : S \vdash M : [\mathtt{unit}]$, where $S(X) = \forall Y \leq X.[\underline{Y} \to Y]$. Informally, if M calls x then it must instantiate it with a subtype of X — in the absence of further subtyping assumptions this must be X itself, giving an object of type $[\underline{X} \to X]$. But M cannot use such a method, as X is unbounded and does not occur anywhere else among its types. Formally, we observe that:

Lemma 3. *For any term $x : S \vdash M : O$, where X does not occur in Γ, O*
$[\![\Gamma, x : S \vdash M : O]\!] = [\![\Gamma, x : S \vdash M[\Lambda Y.[\bot]/x] : O]\!]$

From this we may derive a counterexample to full abstraction. Let $T = \forall X \leq \top.[(S \to [\mathtt{unit}]) \to [\mathtt{unit}]]$ and consider the terms $M = \lambda f^T.f\{\top\}\Lambda Y.[\bot]$ and $N = \lambda f^T.f\{\top\}\Lambda Y.[\lambda y^{\underline{Y}}.(y\,())]$ of type T. Evidently, $M \lesssim_T N$, moreover the inclusion of $[\![M]\!]$ in $[\![N]\!]$ in the games model at this type is strict — they are separated by playing against the strategy defined above.

Proposition 14. *M_1 and M_2 are observationally equivalent at $T \to [\mathtt{unit}]$.*

Proof. It is sufficient to show that for any term $V = \Lambda X.\lambda x.M : T$, $V\{\top\}\Lambda Y.\lambda y^Y.y$ is equivalent to $V\{\top\}\Lambda Y.\lambda y^Y.\bot$. This follows from Lemma 3, since M is (denotationally and thus observationally) equivalent to $M[\Lambda Y.[\bot]/x]$.

Arguably, this example shows up an expressive deficit in $\mathcal{L}_{\leq}$ (and similar type theories for bounded quantification such as $\mathrm{F}_{\leq}$): it does not allow the *extension* of a variable type X with further methods to create a new subtype of X — in the absence of further assumptions, the only subtype of X is X itself, even though X represents an object which could be extended with new fields.

7 Further Directions

Avenues for further research include:

- Our model for bounded quantification has some quite general aspects, being based on a model of unbounded quantification in which morphisms correspond to composable dinatural transformations. A systematic account of this construction, and its coherence properties remains to be given.
- $\mathcal{L}_{\leq}$ could be seen as a more general system with polymorphism at value (method) types, extending the unbounded case [14]. Subtyping for such types is based on inclusion of their values, sucggesting that quantifiecation is bounded *below* (e.g. by the empty type).
- Extension of the full abstraction result to all types, by allowing the extension of a variable type with further methods. This would appear to require the capacity to declare new method labels, suggesting a role for *nominal games*.
- An open problem is to give a directly defined subtyping relation on games which captures all subtyping judgments of $\mathcal{L}_{\leq}$ (or $\mathrm{F}_{\leq}$itself). This appears feasible, notwithstanding the undecidability of subtyping in $\mathrm{F}_{\leq}$(which extends readily to $\mathcal{L}_{\leq}$), since such a relation need not precisely coincide with subtyping in $\mathrm{F}_{\leq}$— for example, the types $\top$ and $S \Rightarrow \top$ are syntactically distinct, but semantically equivalent.

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

References

1. S. Abramsky and R. Jagadeesan. A game semantics for generic polymorphism. *Annals of Pure and Applied Logic*, 133(1):3–37, 2004.
2. S. Abramsky and G. McCusker. Call-by-value games. In M. Neilsen and W. Thomas, editors, *Proceedings of CSL '97*, pages 1–17. Springer-Verlag, 1998.
3. S. Abramsky, K. Honda and G. McCusker. A fully abstract games semantics for general references. In *Proceedings of the 13th Annual Symposium on Logic In Computer Science, LICS '98*. IEEE Press, 1998.
4. E. S. Bainbridge, P. J. Freyd, A. Scedrov, and P.J. Scott. Functorial polymorphism. *Theoretical Computer Science*, 70(1):35–64, 1990.
5. K. Bruce and G. Longo. A modest model of records, inheritance and bounded quantification. *Information and Computation*, 87(1/2):196–240, 1990.
6. L. Cardelli and P. Wegner. On understanding types, data abstraction and polymorphism. *Computing Surveys*, 17(4):471 – 522, 1985.
7. Luca Cardelli, John C. Mitchell, Simone Martini, and Andre Scedrov. An extension of System F with subtyping. *Information and Computation*, 109(1–2):4–56, 1994.
8. J. Chroboczek. Game semantics and subtyping. In *Proceedings of the fifteenth annual symposium on Logic in Computer Science*, pages 192–203. IEEE press, 2000.
9. P.-L. Curien and G. Ghelli. Coherence of subsumption, minimum typing and typechecking in f. *Mathematical Structures in Computer Science*, 2(1):5591, 1992.
10. J. de Lataillade. Dinatural terms in System F. In *Proceedings of the 24th annual symposium on Logic in Computer Science, LICS '09*. IEEE Press, 2009.
11. J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50, 1987.
12. 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.
13. J. Laird. Game semantics for a polymorphic programming language. In *Proceedings of LICS '10*. IEEE Press, 2010.
14. J. Laird. Game semantics of call-by-value polymorphism. In *Proceedings of ICALP '10*, number 6198 in LNCS. Springer-Verlag, 2010.
15. J. Laird. Game semantics for a polymorphic programming language. *Journal of the ACM*, 60(4), 2013.
16. G. McCusker. *Games and full abstraction for a functional metalanguage with recursive types*. PhD thesis, Imperial College London, 1996. Published by Cambridge University Press.
17. J. C. Reynolds. Towards a theory of type structure. In *Proceedings of the Programming Symposium, Paris 1974*, number 19 in LNCS. Springer, 1974.