1 A language of games

1.1 Terms

In this section, we describe a general framework of the language of games. The underlying language we shall normally start with is the simply typed lambda calculus. To this, we fix a finite set $\mathcal{A}$ of countable sets and for each $A \in \mathcal{A}$ we add a type $\underline{A}$ to the language. The types of the form $\underline{A}$ are called the *ground types*. We then add a set of constants to the language. These take one of two forms:

Value constants A *value constant* takes the form $\underline{a} \colon \underline{A}$, where $A \in \mathcal{A}$ and $a \in A$. For our purposes, we shall assume that our language includes a value constant for every element of every set $A \in \mathcal{A}$.

Function constants Given $A \in \mathcal{A}$ and a type T, a *function constant* takes the form $\underline{\phi}$, where ϕ is some function sending elements of A to terms of type T.

Example 1.1. In the usual presentation of PCF, we have a type $\texttt{nat} = \underline{\mathbb{N}}$. Then numerals $\texttt{n}$ are the value constants $\underline{n}$, while the terms $\texttt{suc} : \texttt{nat} \to \texttt{nat}$ and $\texttt{If0}_{\,T} \colon \texttt{nat} \to T \to T \to T$ are function constants corresponding to the functions

$$\phi_{\texttt{suc}} : n \mapsto \underline{n+1}$$
$$\phi_{\texttt{If0}} : n \mapsto \begin{cases} \lambda x.\lambda y.x & \text{if } n = 0 \\ \lambda x.\lambda y.y & \text{otherwise} \end{cases}$$

Remark 1.2. Function constants may refer to other function constants. To avoid circular definitions, define a directed graph on the set of function constants by drawing an edge from $\underline{\phi}$ to $\underline{\psi}$ if $\phi(n)$ contains an occurrence of $\underline{\psi}$ for some n. We require that this graph be acyclic (in particular, having no loops) and that it be well-founded in the sense that there is no infinite sequence $\underline{\phi_1} \to \underline{\phi_2} \to \cdots$.

This framework is enough to formalize simple deterministic languages such as PCF. In order to turn it into a language that can model games, we shall add additional constants to our language:

Definition 1.3. We fix a finite set $\mathcal{P}$ of *players*.

Move constants A *move constant* takes the form $S^b \colon \underline{A}$, where $S \subseteq A$ is non-empty and $b \in \mathcal{P}$.

We read S^b as 'Player b chooses a value from S'.

Our language will be given by the tuple $(\mathcal{A}, \mathcal{F}, \mathcal{M})$, where $\mathcal{F}$ the set of function constants and $\mathcal{M}$ the set of move constants.

Example 1.4. We can recover the language PCF by setting $\mathcal{P} = \{\text{I}\}$ and defining numerals by $\texttt{n} = \underline{n}$, using the function constants $\texttt{suc}$ and $\texttt{If0}$ as

$$\frac{b \in T \qquad \exists a \in S \,.\, C[\underline{a}] \xrightarrow{T} t}{C[S^b] \xrightarrow{T} t} \qquad \frac{b \notin T \qquad \forall a \in S \,.\, C[\underline{a}] \xrightarrow{T} t}{C[S^b] \xrightarrow{T} t}$$

$$\frac{C[M[N/x]] \xrightarrow{T} t}{C[(\lambda x.M)N] \xrightarrow{T} t} \qquad \frac{C[\phi(a)] \xrightarrow{T} t}{C[\underline{\phi}\ \underline{a}] \xrightarrow{T} t}\ \underline{\phi} \in \mathcal{F}$$

Figure 1: Big-step operational semantics for the language $(\mathcal{A}, \mathcal{F}, \mathcal{M})$

$$\text{MOVE}\ \frac{}{S^b \longrightarrow \underline{a}}\ a \in S \qquad \text{FUNCTION}\ \frac{}{\underline{\phi}\ \underline{a} \longrightarrow \phi(a)}$$

$$\text{BETA-REDUCTION}\ \frac{}{(\lambda x.M)N \longrightarrow M[N/x]} \qquad \text{CONTEXT}\ \frac{M \longrightarrow N}{C[M] \longrightarrow C[N]}$$

Figure 2: Small-step operational semantics for the language $(\mathcal{A}, \mathcal{F}, \mathcal{M})$

defined above. We can add a nondeterministic oracle to the language by setting $? = \mathbb{N}^{\mathrm{I}}$.

1.2 Operational semantics

We define a notion of *evaluation context* in the style of Felleisen:

$$C ::= \_ \mid C\ M \mid \underline{\phi}\ C$$

Here, M ranges over terms in the language and $\underline{\phi}$ ranges over the function constants in $\mathcal{F}$. We write $M = C[M']$ if the term M is obtained from C by replacing the unique occurrence of $\_$ in C with the term M'.

We call a program of ground type $\underline{A}$ a *game*. We call a subset $T \subseteq \mathcal{P}$ a *team* and a subset $t \subseteq A$ a *target*.

We now define a predicate $\xrightarrow{T} t$ on terms as in Figure 1. To understand this semantics better, let's look at the small-step semantics defined in Figure 2. A term $M : T$ now defines a game between team T and the opposing team $\mathcal{P} \setminus T$ as follows: we evaluate the term step by step. At each step, the proof that $M \longrightarrow M'$ must terminate with one of the three rules MOVE, FUNCTION or BETA-REDUCTION. In the last two cases, there is a unique choice of reduction to make. In the first case, there are multiple possibilities, one for each value $a \in S$.

In such a case, if $b \in T$ then team T are allowed to choose which value a to use. If $b \notin T$, then the opposing team $\mathcal{P} \setminus T$ are allowed to choose. Team T wins if the term eventually converges to a value $\underline{a}$, where $a \in t$ and team $\mathcal{P} \setminus T$ wins if the term converges to $\underline{b}$, where $b \notin t$. Then the judgement $M \xrightarrow{T} t$ precisely says that team T has a strategy to ensure that the term M ends up in t.

Example 1.5. Coming back to our nondeterministic PCF example, if $M \colon \mathtt{nat}$ is a term, then the judgement that $M \xrightarrow{\varnothing} \{n\}$ says that M *must converge* to n. The judgement that $M \xrightarrow{\{I\}} \{n\}$ says that M *may converge* to n.

We read $M \xrightarrow{T} t$ as 'Team T can force the term M into the target t'.

Definition 1.6 (Observational equivalence)**.** We define observational equivalence for a specified set $\mathfrak{T} \subseteq \mathfrak{P}(\mathcal{P})$ of 'friendly teams' and a specified set $\mathfrak{t} \subseteq \prod_{A \in \mathcal{A}} \mathfrak{P}(A)$ of specified targets.

If $A \in \mathcal{A}$ and $M, M' \colon \underline{A}$, we say that $M \equiv_e M'$ (M is *observationally equivalent to* M') if for all teams $T \in \mathfrak{T}$ and for all target sets $t \in \mathfrak{t}$ we have $M \xrightarrow{T} t(A)$ if and only if $M' \xrightarrow{T} t(A)$.

For general terms M, M' of the same type, we say that $M \equiv_e M'$ if for all contexts C such that $C[M], C[M']$ have ground type, $C[M] \equiv_e C[M']$.

Definition 1.7 (Observational order)**.** Similarly, we say that a term M is *observationally less than or equal to* another term M', and write $M \leq_e M'$, if for all $T \in \mathfrak{T}$ and for all $t \in \mathfrak{t}$, $M \xrightarrow{T} t(A)$ implies $M' \xrightarrow{T} t(A)$.

For general terms M, M' of the same type, we say that $M \leq_e M'$ if for all contexts C such that $C[M], C[M']$ have ground type, $C[M] \leq_e C[M']$.

Example 1.8. PCF with may equivalence is the case corresponding to the set $\mathfrak{T} = \{\{\mathrm{I}\}\}$ and $\mathfrak{t} = \{\mathbb{N}\}$.

PCF with must equivalence corresponds to $\mathfrak{T} = \{\varnothing\}$ and $\mathfrak{t} = \{\mathbb{N}\}$.

PCF with may-and-must equivalence corresponds to $\mathfrak{T} = \{\varnothing, \{\mathrm{I}\}\}$ and $\mathfrak{t} = \{\mathbb{N}\}$..

Lemma 1.9. *Let $M, M : S$ be such that $M \leq_e M'$, and let $F, F' : S \to T$ be such that $F \leq_e F'$. Then $FM \leq_e F'M'$.*

1.3 Recursion

In order to recover the full language PCF, we need to add recursion combinators $\mathbf{Y}_T$ to our language, corresponding to the following small-step reduction:

$$\textsc{recursion}\ \frac{}{\mathbf{Y}_T M \longrightarrow M(\mathbf{Y}_T M)}$$

giving us the big-step rule

$$\frac{C[M(\mathbf{Y}_T M)] \xrightarrow{T} t}{C[\mathbf{Y}_T M] \xrightarrow{T} t}$$

This rule introduces for the first time the possibility of *divergence*, or non-termination. We will assume that a team T never wants the program to diverge. We do not lose anything by making this assumption; indeed, we can model the predicate $M \xrightarrow{T} t \cup \{\bot\}$

(‘Team T has a strategy to force the term M into the target t or to make it diverge’) as the negation of the predicate $M \xrightarrow{\mathcal{P}\setminus T} \mathcal{A} \setminus t$ (‘The opposition to team T has a strategy to force the term M into the complement of the target t’): the games we are considering are all *open* in the sense that they terminate after finitely many moves, so exactly one of the two teams has a strategy by the Gale-Stewart theorem.

It turns out that we can also model PCF – and other languages we have constructed in this section – as a subset of a larger language that does not involve any special recursion constants, but uses only the function, value and move constants that we have already introduced. The advantage of this is that, since function constants are not self-referential, they are much easier to model in a denotational semantics than recursion, which usually requires some order-theoretic properties of the underlying model. The only innovation we will need to make on the semantics side is to give denotational semantics for the move constants; this will be covered in Section **??**.

As an illustration of our technique, we show how PCF itself can be modelled inside another language. First, we introduce a new function constant, $\mathtt{iter}_T \colon \mathtt{nat} \to T \to (T \to T) \to T$, corresponding to iteration (realization of natural numbers as Church numerals). $\mathtt{iter}_T$ is defined by $\mathtt{iter}_T = \underline{\iota_T}$, where:

$$\iota_T(n) = \lambda z.\lambda s.\,\underbrace{s(s(s\cdots z}_{n\text{ times}})\cdots))$$

In order to model divergence, we add special divergence constants Ω_T for each type T. We include no extra rules for these divergence constants; if we end up with $C[\Omega_T]$ for some context C, then we consider the term to be ‘stuck’, and to have converged to no value.

Now let M be a general PCF term (possibly involving instances of $\mathbf{Y}_T$ for some types T. Since PCF is a deterministic language, if M converges then it does so in a bounded number of steps m. If we now replace each instance of $\mathbf{Y}_T$ in M with the function

$$\lambda N.\mathtt{iter}_T\ \underline{m}\ \Omega_T N$$

then we claim that the term converges to the same value. Indeed, we observe that $\mathtt{iter}_T\ \underline{m}\ \Omega_T N$ is equivalent to the term

$$\underbrace{N(N(N\cdots}_{m\text{ times}}\Omega_T)\cdots))$$

and is therefore itself equivalent to

$$N(\mathtt{iter}_T(\underline{m-1})\Omega_T N)$$

The term $\mathtt{iter}_T\ \underline{m}\ \Omega_T N$ therefore has the same behaviour as the term $\mathbf{Y}_T N$, except that the index m decreases by 1 each time we use the rule. Since the reduction of the term N takes m steps anyway, we will never ‘run out’ of iterations.

However, although this allows us to model *individual programs* in PCF, it does not allow us to model functions that may take an unbounded number of recursion steps depending on their argument. In particular, we cannot model $\mathbf{Y}_T$ itself using iteration alone. In order to do this, we add a nondeterministic oracle ? to the language, and using may-equivalence as our equivalence relation on terms. In our framework, this corresponds to adding a player so that $\mathcal{P} = \{\mathrm{I}\}$ and then setting $? = \mathbb{N}^{\mathrm{I}}$. Then we can model $\mathbf{Y}_T$ as the term

$$\lambda N.\mathtt{iter}_T\ ?\ \Omega_T N$$

Because of the may-equivalence, this now has the same operational semantics (within PCF) as our original $\mathbf{Y}_T$: if a PCF term converges, then it converges after m steps for some m, and so there is the *possibility* that the oracle ? will choose a large enough value such that the term must converge. So the term may converge to the same value as the original term with $\mathbf{Y}_T$. On the other hand, if a PCF term diverges, then there is no number we could replace the ? with that would cause the term to converge. In that case, for every m that the oracle could choose, the term will eventually 'run out' and reach the divergence term Ω_T, at which point it will get stuck.

Theorem 1.10. *There is an embedding of the language*

$$PCF = \lambda + \mathbb{N} + \mathbf{Y}$$

into the language

$$\textit{Nondeterministic iterative PCF} = \lambda + \mathbb{N} + \mathtt{iter} + \Omega + ?$$

under may-equivalence. That is to say, there is an structurally-inductive map r from PCF terms/contexts to terms/contexts of nondeterministic iterative PCF such that two PCF programs M and N are observationally equivalent if and only if the corresponding programs in nondeterministic iterative PCF are may-equivalent.

Proof. The mapping r replaces every instance of $\mathbf{Y}_T$ with $(\lambda N.\mathtt{iter}_T\ ?\ \Omega_T N)$.

Note the following observational equivalence:

$$\mathbf{Y}_T \equiv_e \lambda N.\mathtt{iter}\ \underline{m}\ (\mathbf{Y}_T N)\ N$$

which holds for any n. This is because both terms may converge □

2 Games

In this section, we define the notion of a game, one of our two alternative formulations of our basic intuition of a 'game'.

We fix some set $\mathcal{P}$ of players throughout.

Definition 2.1. Let A be a set. A game Γ *with result set* A is given by a tuple $(\mathcal{M}_\Gamma, P_\Gamma, J_\Gamma)$ where

- $\mathcal{M}_\Gamma$ is a set of moves.
- P_Γ is a prefix-closed set of finite sequences of pairs of the form $p\colon m$, where $p \in \mathcal{P}$ is a player and $m \in \mathcal{M}_\Gamma$ is a move. We define T_Γ, the set of *terminal* plays in P_Γ to be the set of all $s \in P_\Gamma$ such that $sa \notin P_\Gamma$ for any a.
- $J_\Gamma\colon T_\Gamma \to A$ is a judgement function that says what the result is after a complete sequence of moves.

such that

- There is always exactly one player whose turn it is. Explicitly, if $sa, sb \in P_\Gamma$, then the moves a and b are made by the same player.

Remark 2.2. Γ may contain infinitely long chains of sequences of MOVEs. In that case, there is no result value for the PLAY.

Remark 2.3. This construction gives rise to a monad G on the category of (possibly large) sets, where a set A is sent to the set GA of games with result set A. The natural transformation $GGA \to GA$ is the one that takes a game whose results are games with result set A, and forms a game that plays as follows: the players play through the first game; then, if and when they reach a terminal PLAY, they then play through the resulting game to reach a value in A.

We will not make great use of this monad, but it will be implicit in a lot of what we do. As we can model the powerset monad using nondeterministic strategies, so our game semantics will allow us to model this game monad.

Definition 2.4. Let Γ be a game with result set A. Let $T \subseteq \mathcal{P}$ be a team of players. Then a T-strategy S for Γ is a prefix-closed subset of P_Γ subject to the following rules:

- If $s \in S$ and $sa \in P_\Gamma$ is such that the move a is made by a player *not* in T, then $sa \in S$.
- If $sa, sb \in S$, and the moves a, b are made by a player in T, then $a = b$.
- If $s \in S$ and there exists $sa \in P_\Gamma$ such that the move a is made by a player in T, then there exists b such that $sb \in P_\Gamma$.

3 Game Semantics

3.1 Arenas

Our basic game semantics will be the one given in [HO00]. We first present the Hyland-Ong game semantics. Then, we shall enrich the semantics by appending GAMEs on to strategies.

Definition 3.1. An *arena* is a triple $A = (M_A, \lambda_A, \vdash_A)$ where

- M_A is a countable set of *moves.*
- $\lambda_A \colon M_A \to \{O, P\} \times \{Q, A\}$ designates each move as either a *P-move* or an *O-move* and as either a *question* or an *answer.* We will often split λ_A into its two parts

$$\lambda_A^{OP} = \mathrm{pr}_1 \circ \lambda_A \colon M_A \to \{O, P\} \quad (1)$$

$$\lambda_A^{QA} = \mathrm{pr}_2 \circ \lambda_A \colon M_A \to \{Q, A\} \quad (2)$$

- $\vdash_A$ is an *enabling relation* between $M_A + *$ and M_A satisfying:

  Alternation If $a \vdash_A b$ then $\lambda_A^{OP}(a) = \neg(\lambda_A^{OP}(b))$

  Initial moves If $* \vdash_A a$ then $\lambda_A(a) = (O, Q)$ and for all $b \in M_A$, $b \not\vdash_A a$.

  Answers are enabled by questions If $a, b \in M_A$ are such that $a \vdash_A b$ and $\lambda_A^{QA}(b) = A$ then $\lambda_A^{QA}(a) = Q$.

We say that a move a is *initial* if $* \vdash_A a$.

Definition 3.2. A *justified play* s in an arena A is given by a (finite or infinite) sequence of moves of A, together with a *justification pointer* for each non-initial move a occurring in s back to a previous move b such that $b \vdash_A a$. In that case, we say that *b justifies a.*

Furthermore, we require that if $s \neq \epsilon$ then the first move in s is initial.

Definition 3.3. We say that a move *a hereditarily justifies* a move b if there is a sequence of justification pointers in $J(s)$ that leads from b to a.

3.2 Compound arenas

Let A, B be arenas. Then there are two main ways to combine A and B to make a new arena. The first forms the *product* $A \times B$ of A and B:

- $M_{A \times B} = M_A + M_B$.
- $\lambda_{A \times B} = [\lambda_A, \lambda_B]$.
- $x \vdash_{A \times B} m$ if and only if $x \vdash_A m$ or $x \vdash_B m$.

The second forms the *function space* $A \Longrightarrow B$ from A to B:

- $M_{A \Longrightarrow B} = M_A + M_B$.
- $\lambda_{A \Longrightarrow B} = [(\neg \times \mathrm{id}) \circ \lambda_A, \lambda_B]$.
- $x \vdash_{A \Longrightarrow B} m$ if and only if
  either $x \vdash_B m$
  or $x \neq *$ and $x \vdash_A m$
  or $* \vdash_B x$ and $* \vdash_A m$.

3.3 Strategies

In order to capture the possibility of divergence, we follow [HM99] and extend the usual definition of a strategy as a set of traces by adding in an extra set of positions designating which plays may lead to a divergence.

Blah blah blah blah blah.

3.4 Composition of strategies

Same as usual.

4 Strategies as GAMEs

4.1 GAME-plays and GAME-strategies

We now enrich the structure of strategies by equipping them with certain classes of GAMEs. We fix a set $\mathcal{P}$ of players throughout.

Definition 4.1. Let A be an arena and let $\mathcal{M}$ be a set of moves. Then a GAME-play over M_A is a (possibly infinite) sequence s of moves that are either:

- Moves $a \in M_A$, called *moves* or:
- Moves $p\colon m \in \mathcal{P} \times \mathcal{M}$, called MOVE*s*

such that

- The first move in s is an O-move in A.
- If a move of the form $(p\colon m)$ is followed by a move $a \in M_A$, then a is a P-move.
- Every P-move in A is followed immediately (if at all) by an O-move in A.

The *shadow* of a GAME-play is the sequence formed by removing all moves of the form $p\colon m$. A *justified* GAME-*play for* A is a GAME-play s over A, together with the structure of a justified play over the shadow of s.

Definition 4.2. A GAME-*strategy* for an arena A is a set $\mathcal{M}_\sigma$ of moves and a prefix-closed set of GAME-plays over $\mathcal{M}$ such that when we forget about the PLAYS Σ_s^t, the resulting collection of justified sequences is a strategy for A, and such that if $s.(p\colon m), s.(q\colon n) \in S$, then $p = q$.

4.2 Composition

Possibly slightly tricky; the idea is that the process of 'hiding' moves concatenates PLAYs together.

4.3 Ordering strategies

Definition 4.3. Let σ be a GAME-strategy and let $T \subseteq \mathcal{P}$ be a team of players.

A T-STRATEGY for σ[1] is a subset S of σ subject to the following rules:

- If $s \in S$ and $sa \in \sigma$, where a is either an O-move in M_A or a move $(p : m)$, where $m \notin T$, then $sa \in S$.
- If $sa, sb \in S$, where a, b are P-moves in M_A, then $a = b$.
- If $s.(p\colon m), s.(p\colon n) \in S$, and $p \in T$, then $m = n$.
- If $s \in S$ and $sa \in \sigma$, where a is either a P-move or a move of the form $(p\colon m)$ for $m \in T$, then $sb \in S$ for some b.

[1]Yes, that is a STRATEGY for a strategy.

Definition 4.4. Let σ be a GAME-strategy, and let S be a STRATEGY for σ. Then, if s is an O-play in A that is the shadow of some GAME-play in σ, we write R^s_S for the set of moves a such that sa is the shadow of some GAME-play in S.

We write R^s_σ for the set

$$\{R^s_S : \ S \text{ is a STRATEGY for } \sigma \text{ and } s \text{ is the shadow of a play in } S, R^s_S \neq \varnothing\}$$

Definition 4.5. Let σ, τ be GAME-strategies. We say that $\sigma \leq_T \tau$ if the shadow of σ is a subset of the shadow of τ and if for all O-moves s that are the shadow of a play in σ, we have $R^s_\sigma \subseteq R^s_\tau$.

4.4 Example – one-player GAMEs

Suppose that $\mathcal{P} = \{\text{I}\}$. Then there are two possibilities for T: $T = \varnothing$ and $T = \{\text{I}\}$.

We show that these correspond to the may- and must-orderings defined by Harmer and McCusker.

Proposition 4.6. *Suppose $\{P\} = \{\text{I}\}$. Let A be an arena and let σ be a strategy for A. Define*

- *T_σ to be the shadow of σ*
- *D_σ to be the set of all O-positions d in T_σ such that there is either an infinite position in σ having shadow d or such that there is a terminal position in σ having shadow d and ending with a MOVE.*

Now let σ, τ be GAME-strategies for A.

i) $\sigma \leq_{\{\text{I}\}} \tau$ if and only if $T_\sigma \subseteq T_\tau$.

ii) $\sigma \leq_\varnothing \tau$ if and only if ...

Proof. (i): **If.** Suppose that $T_\sigma \subseteq T_\tau$. We want to show that $\sigma \leq_{\{\text{I}\}}$, for which it suffices to show that for all O-positions $s \in T_\sigma$ we have $R^s_\sigma \subseteq R^s_\tau$. Let $s \in T_\sigma$ be an O-position and let S be a STRATEGY for σ such that s is the shadow of some GAME-play in S and $R^s_S \neq \varnothing$. Let $a \in R^s_S$. By the definition of a STRATEGY for σ, we can see that in fact $R^s_S = \{a\}$.

Since $T_\sigma \subseteq T_\tau$, we know that there is some GAME-play in τ whose shadow is sa. Then it is easy to construct a STRATEGY S' for τ such that $a \in R^s_{S'}$. Moreover, by the same argument, we see that $R^s_{S'} = \{a\}$. Therefore, $R^s_\sigma \subseteq R^s_\tau$.

Only if. Suppose that $\sigma \leq_{\{\text{I}\}} \tau$. Let $sa \in T_\sigma$ be a non-empty P-position. Then sa is the shadow of some position in σ. We can construct a STRATEGY S for σ such that sa is in the shadow of S; moreover, we have $R^s_S = \{a\}$. Therefore, $\{a\} = R^s_{S'}$ for some STRATEGY S' for τ. It follows that $sa \in T_\tau$.

□

Example 4.7. Fix two players B (black) and W (white) and a set of moves $\mathcal{M}$ consisting of all legal chess moves. Then we can model games played on a chess board using the arena given by

$$(\{q, ww, dr, bw\}, \{q \mapsto (O, Q),\ ww, dr, bw \mapsto (P, A)\}, \{q \vdash ww, dr, bw\}, \{q\})$$

A justified play over the sequence $q.ww$ is a collection of justification pointers j from ww to q, each annotated with a sequence of chess moves carried out by one or other of the two players. The game of chess itself is then described by three particular justified plays:

- One over the sequence $q.ww$, where the labels of the justification pointers from ww to q are precisely those sequences of moves leading to a win for white.
- One over the sequence $q.dr$, where the labels of the justification pointers from dr to q are precisely those sequences of moves leading to a draw.
- One over the sequence $q.bw$, where the labels of the justification pointers from bw to q are precisely those sequences of moves leading to a win for black.

References

[HM99] R. Harmer and G. McCusker. A fully abstract game semantics for finite nondeterminism. In *Proceedings. 14th Symposium on Logic in Computer Science (Cat. No. PR00158)*, pages 422–430, 1999.

[HO00] J.M.E. Hyland and C.-H.L. Ong. On full abstraction for PCF: I, II, and III. *Information and Computation*, 163(2):285 – 408, 2000.