A Fully Abstract Game Semantics of Local Exceptions

J. Laird
COGS, University of Sussex
jiml@cogs.susx.ac.uk

Abstract

A fully abstract game semantics for an extension of Idealized Algol with locally declared exceptions is presented. It is based on “Hyland-Ong games”, but as well as relaxing the constraints which impose functional behaviour (as in games models of other computational effects such as continuations and references), new structure is added to plays in the form of additional pointers which track the flow of control. The semantics is proved to be fully abstract by a factorization of strategies into a “new-exception generator” and a strategy with local control flow. It is shown, using examples, that there is no model of exceptions which is a conservative extension of the semantics of Idealized Algol without the new pointers.

1 Introduction

All practical programming languages provide some means of manipulating the flow of control, primarily to recover from errors and deal with other exceptional eventualities. Dynamically bound, locally declared exceptions are a simple, elegant and effective way to do this, making them a key part of ML and Java, for example. Despite their ease of use for programmers, however, these exceptions are not “easy” from a semantic point of view; no denotational model of a language containing them has hitherto been described. Statically bound exceptions can be implemented using call-with-current-continuation, but fail to account for one of the most important features of exceptions — that the same error may be handled in different ways if it occurs in different contexts. On the other hand, dynamically bound global exceptions have been modelled abstractly via the exceptions monad [13], but this approach has not been applied to locally exceptions. It may be argued that local exceptions have proved resistant to the efforts of semanticists in part because they are a kind of hybrid effect. Their main purpose is to give access to the flow of control, but dynamic binding distinguishes them from statically bound control constructs such as call/cc, whilst locality gives rise to some of the identity-related issues which appear with reference variables. But since continuations and store have traditionally been modelled by (very different) constructions, simply piling them on top of a functional basis is likely to lead to a complicated semantics which is not fully abstract.

The basis for a possible solution to these problems can be found in the “intensional hierarchy” [4] of games models of various effects such as state [1, 3], first-class continuations [11] and higher type references [5]. These all extend the basic model of PCF described by Hyland and Ong [9], and Nickau [14], by relaxing, one-by-one, the constraints on games and strategies which oblige them to behave in a purely functional way. This “direct” approach to modelling side-effects means that they can often be combined simply (and fully abstractly) by relaxing the relevant combination of constraints.

However, even in the context of game semantics, the dynamic nature of exceptions has significant ramifications. Rather than simply weakening the appropriate constraints on the model of PCF, it proves necessary to to add significant new structure — in the form of additional ‘contingency pointers” — to the traces which represent the states of a game, in order to describe the dynamic binding of exceptions. These pointers give an explicit representation of control flow in the model, allowing a move to be played as if it immediately follows an earlier move which is not actually its immediate predecessor.

The main contribution of this paper is therefore to define a new category of games by adding contingency pointers to HO-style games, to show that this category contains a model of locally declared, dynamically bound exceptions, and — by a full abstraction result — to show precisely how these combine locality with manipulation of control-flow. Using this analysis, it will show that the contingency structure really is necessary to interpret exceptions in HO games. Because the
games models of references and continuations do not have this structure, this suggests that exceptions cannot be expressed using continuations and references.

Acknowledgements

The work reported here was supported by several UK EPSRC grants. I am grateful to Guy McCusker in particular for his comments.

2 Idealized Exceptions

The language which will be modelled — Idealized Algol [16] with (idealized) exceptions, or IAx for short — is a typed call-by-name λ-calculus with locally declared ground-type references and a pared down call-by-name version of the simple exceptions (based on ML exceptions) described by Gunther Rémy and Rösch [7]. IAx types are generated from the base types \( 0 \) (empty), \( \text{com} \) (commands), \( \text{n} \) (natural numbers), \( \text{var} \) (natural number references) and \( \text{exn} \) (exceptions).

\[ T ::= 0 | \text{com} | \text{n} | \text{var} | \text{exn} | T \rightarrow T. \]

Terms are formed according to the grammar:

\[
M ::= x | \text{skip} | 0 | \text{suc} M | \text{pred} M | \text{IF} M | \text{IF} F M | M M | \text{Y} M | M M | M M | M M | M M \]

\[ \text{new}_{\text{exn}} M | \text{mkexn} M M | \text{raise} M | \text{handle} M M | \text{new} M | \text{mkvar} M M | M := M | !M. \]

Typing judgments extend those for IA as follows

\[ (B = 0) \text{com} \text{nat}: \]

\[
\begin{array}{c}
\Gamma \vdash M : \text{exn} \rightarrow B \\
\Gamma \vdash M : 0 \rightarrow \text{com} \\
\Gamma \vdash N : 0 \\
\Gamma \vdash M : \text{exn} \\
\Gamma \vdash M : B \\
\Gamma \vdash \text{new}_{\text{exn}} M : B \\
\Gamma \vdash \text{mkexn} M N : \text{exn} \\
\Gamma \vdash \text{raise} M : B \\
\Gamma \vdash M : B \\
\Gamma \vdash \text{handle} M N : \text{com} \\
\end{array}
\]

The “big step” operational semantics for the imperative fragment of IAx is given in Table 1. Evaluation takes place in an environment consisting of a set of exception names \( E \), a set of variable names or locations \( L \), and a store \( S \) — a partial mapping from \( L \) to natural numbers. By convention, mention of the environment is omitted where possible. The \( \text{new}_{\text{exn}} \) and new constants evaluate in the same way; each generates a new name which is added to the environment, and supplied to its argument. Similarly, the \( \text{mkvar} \) construct for generating “bad variables” [15, 1] has a precise analogue in the \( \text{mkexn} \) operation for constructing “bad exceptions”; terms of exception type which may not have the correct raising and handling behaviour.

Programs are evaluated to a final form \( D \), which is either a value \( V \) or an exception \( E \Rightarrow \text{raise} h \) for some name \( h \); the latter are propagated through the program until they are caught. The handler is simply an operation for capturing a named exception. Because there are no values of type \( 0, N : 0 \) can only evaluate to an exception \( \text{raise} k \), so \( \text{handle} h N \) compares the names \( h \) and \( k \) and evaluates to \( \text{skip} \) if they are equal and propagates the exception \( \text{raise} k \) if they are not. Unlike ML exceptions, in which the use of a universal type of exceptions results in recursive behaviour, the much more restrictive typing of IAx prevents this.

**Proposition 2.1** For any program \( M \) of IAx — \( \{ Y \} \), there is some \( D \) such that \( M \downarrow D \).

A standard notion of observational equivalence can be defined.

**Definition 2.2** Terms \( M, N : T \) are observationally equivalent (written \( M \simeq N \)) if for any closing context \( \text{C}[\cdot] : \text{com}, \text{C}[M] \downarrow \text{skip} \) if and only if \( \text{C}[N] \downarrow \text{skip}. \)

Idealized exceptions fit well with the block structure of Idealized Algol and, despite their apparent simplicity, are quite expressive. For instance, although exceptions in Java (and to a lesser extent ML) are more sophisticated in that one handler can be used to trap different exceptions using subtyping, the basic behaviour of Java’s try and catch operations can be captured by defining (for \( M, N : \text{com}, H : \text{exn} \)):

\[ \text{try} M \text{catch} H N \equiv \text{newexn} k, \]

\[ \text{handle} k ((\text{handle} H (M; \text{raise} k)); N; \text{raise} k). \]

This executes the command \( M \); if this is completed the catch block is discarded, but if the exception \( H \) is raised whilst running \( M \), then it is caught and the command \( N \) is executed.

Exceptions in ML can carry values; this “storage” aspect of exceptions has not been included in IAx because it seems peripheral to the more significant features of exceptions (control-flow manipulation and local declaration) and can be simulated very easily using explicit store; for example, in IAx exceptions carrying natural numbers as values can be represented using (\( \text{var} \Rightarrow (\text{exn} \Rightarrow \text{com}) \Rightarrow \text{com} \) as the type of natural-number-carrying exceptions as follows:

\[ \text{new}_{\text{exn}} M =_{df} \text{new}_{\text{exn}} \lambda x. \text{new}_{\text{exn}} (M \lambda y (g y) y) x \]

\[ \text{raise} M N : B =_{df} (M (\lambda x g. y. N; \text{raise} x)); \Omega \]

\[ \text{handle} M N =_{df} \text{new}_{\text{exn}} \lambda z. (M \lambda y (\text{handle} y N); z := !x); !z. \]

3 Control Games

The games constructions which will be used to model IAx are based on those given by Hyland and Ong [9] and Nickau [14], in which states of the game are represented as justified sequences of moves. Several developments of this basic framework will be used
\[ 
\begin{align*}
\forall \downarrow \forall \\
M \downarrow h_\downarrow \forall \downarrow (x) \downarrow D \quad & M \downarrow (h_\downarrow \forall \downarrow D) \quad & h \notin \mathcal{E} \\
L \downarrow n \quad & M \downarrow \text{mkvar} N_1 N_2 N_1 \downarrow n \downarrow D \quad & M \downarrow \text{mkvar} N_1 N_2 N_2 \downarrow D \\
N \downarrow D \quad & M \downarrow \text{new\_exn} M \downarrow \text{L} \downarrow D \\
M \downarrow E \quad & M \downarrow \text{mkexn} N_1 N_2 N_3 L \downarrow D \\
N \downarrow E \quad & M \downarrow E \\
M \downarrow h \quad & M \downarrow \text{raise} e \quad & e \neq h \\
N \downarrow \text{E} \quad & M \downarrow \text{raise} \quad & M \downarrow \text{skip} \\
N \downarrow \text{L} \downarrow D \quad & M \downarrow \text{handle} M \downarrow \text{L} \downarrow D \\
N \downarrow \text{N} \downarrow E \quad & M \downarrow \text{handle} M \downarrow \text{N} \downarrow E \\
N \downarrow \text{N} \downarrow E \quad & M \downarrow \text{N} \downarrow E \\
N \downarrow \text{N} \downarrow \text{E} \\
M \downarrow \text{N} \downarrow \text{raise} e \quad & M \downarrow \text{N} \downarrow \text{raise} h \\
\end{align*} 
\]

Table 1. Operational semantics of exceptions and store

Here — in particular the relaxation of constraints to define a model of Idealized Algol [1, 5]. However, it has also been necessary to enrich more significantly the structure on which the games are based — justified sequences — by adding a new notion of "contingency pointer" to track the flow of control. Fortunately, this fits in relatively smoothly with the original constructions and developments aforementioned.

The structure of a game (the moves, their labels, how they are related) is specified by its arena, defined essentially as in [9]. An arena \( A \) is a triple: \( \langle M_A, \vdash_A \subseteq (M_A)_s \times M_A, \lambda_A : M_A \to \{Q, A\} \rangle \), where \( M_A \) is a set of tokens called moves, \( \vdash_A \subseteq (M_A)_s \times M_A \) is a relation called enabling, which allows a unique polarity for moves to be inferred by the following rule — \( m \) is an O-move if it is initial (i.e., \( \vdash m \)), or enabled by a P-move, \( m \) is a P-move if it is enabled by an O-move, \( \lambda_A : M_A \to \{Q, A\} \) is a function which labels moves as answers (A) or questions (Q), such that every answer has a unique enabling move which is a question.

A justified sequence over an arena \( A \) is a sequence of elements of \( M_A \) in which each occurrence of a non-initial move comes with a justification pointer to a preceding occurrence of an enabling move. The transitive closure of justification is referred to as hereditary justification. A sequence is alternating if Opponent moves are always followed by Player moves, and vice versa.

In order to capture the control behaviour of exceptions in a compositional way, additional pointers of a very similar kind will be added to justified sequences.

(The key difference is that there is no structure constraining these pointers analogous to the enabling relation.)

**Definition 3.1** A contingency pointer for a move in a justified sequence is a pointer (distinct from its justification pointer) to a preceding question. A move is contingent if it has such a pointer. A control sequence is a justified sequence in which contingency pointers satisfy the same conditions as justification pointers: i.e.

- every Player move is contingent on some Opponent move,
- every contingent Opponent move is contingent on a Player move,
- every answer move is contingent on its enabling question.

The set of alternating control sequences over the arena \( A \) will be written \( C_A \). If \( a \) can be reached by following contingency pointers back from \( c \), then \( c \) is said to be hereditarily contingent on \( a \). To avoid ambiguity caused by multiple occurrences of the same move, we shall sometimes say that in the sequence \( tb \), \( b \) is contingent on the prefix \( sa \subseteq tb \) instead of saying that \( b \) is contingent on \( a \).

### 3.1 A Category

A category of arenas and strategies can now be defined using the standard constructions [9, 12].
Product: For any set-indexed family of arenas \( \{A_i \mid i \in I \} \), form the product \( A = \prod_{i \in I} A_i \) as follows:

- \( M_{i \in I} A_i = \prod_{i \in I} M_{A_i} \)
- \( \langle m, i \rangle \vdash_{A \times B} \langle n, j \rangle \) if \( i = j \) and \( m \vdash A_i n \), and \( \ast \vdash_{A \times B} \langle n, j \rangle \) if \( \ast \vdash A_j n \)
- \( \lambda^A_{i \in I} A_i(m, i) = \lambda A_i(m) \)

For finite \( k \), the product of \( k \) copies of the arena \( A \) will be written \( A^k \).

Function Space: For arenas \( A_1, A_2 \)

- \( M_{A_1 \rightarrow A_2} = M_{A_1} + M_{A_2} \)
- \( \langle m, i \rangle \vdash_{A \rightarrow B} \langle n, j \rangle \) if \( i = j \) and \( m \vdash n \) or \( m \in M_{B_1}, n \in M_{A_1} \) and \( \ast \vdash B_1 n \), \( \ast \vdash (m, j) \) if \( m \in M_B \) and \( \ast \vdash B \),
- \( \lambda_{A \rightarrow B}^{A_1} (m, i) = \lambda_{A_1}(m) \).

The arena with a single question move is written \( q \).

Definition 3.2 A (deterministic) strategy over an arena \( A \) is a non-empty even-prefix-closed set of even length alternating justified sequences which is evenly branching: \( sa, sb \in \sigma \Rightarrow b = c \).

A control-strategy on \( A \) is a strategy consisting of control-sequences (i.e. a subset of \( C_A \)).

The control-strategies will be referred to simply as strategies where the context is clear.

Composition of control-strategies is a straightforward extension of 'parallel composition with hiding' [6] to control sequences.

If \( s \in C_{A_1 \rightarrow (A_2 \rightarrow A_3)} \) then \( s(A_1, A_2) \) is a sequence with contingency pointers (not necessarily a true control sequence) defined as follows:

- \( e(A_1, A_2) = e \),
- \( sa(A_1, A_2) = sa(A_1, A_2) \) if \( s \notin A_1, A_2 \),
- \( sa(A_1, A_2) = (sa(A_1, A_2)) / a \) if \( a \notin A_1, A_2 \),

where \( a \) is justified by the most recently played move from \( A_1 \) or \( A_2 \) which hereditarily justifies \( a \) in \( s \) (if any) and \( a \) is contingent on the most recent move from \( A_1 \) or \( A_2 \) on which it is hereditarily contingent.

Definition 3.3 For \( \sigma : A_1 \rightarrow A_2 \rightarrow A_3 \)

- \( \sigma, \tau = \{ s \in C_{A_1 \rightarrow A_2} \mid \exists t \in C_{(A_1 \rightarrow A_2) \rightarrow A_3}, t(A_1, A_2) \in \sigma \land t(A_2, A_3) \in \tau \land t, A_1, A_3 \in s \} \).

As usual, canonical morphisms are copycat strategies which just copy Opponent moves between different parts of a game. However, contingency pointers (unlike justification pointers) are not copied; to define copycat control-strategies requires the notion of pending question.

Definition 3.4 Define the “pending question prefix” of a justified sequence as follows:

- \( \text{pending}(\varepsilon) = \varepsilon \)
- \( \text{pending}(sa) = sa, \) if \( a \) is a question,
- \( \text{pending}(satb) = \text{pending}(s), \) if \( b \) is an answer to \( a \).

Definition 3.5 For any arena \( A \), define the identity control-strategy \( \text{id}_A : A \Rightarrow A \) to be the least subset of \( C_{A \Rightarrow A} \) containing \( \varepsilon \) and closed under the condition:

- If \( s \in \text{id}_A \) and \( sa, A^+ = s, A^- \) and \( b \) is contingent on the last move in \( s \) then \( sb \in \text{id}_A \).

So, for example, in the play of \( \text{id}_{c \Rightarrow o} \) represented in Figure 1, the last move is contingent on its immediate predecessor, but justified by the initial move. As

\[
\text{(o \Rightarrow o) \Rightarrow (o \Rightarrow o)}
\]

for general strategies [12], arenas and control-strategies form a SMCC which can be refined to a CCC of well-opened strategies.

Definition 3.6 The thread of the last move in a non-empty control sequence is defined as follows:

- \( \text{thread}(sa) = a, \) (a initial)
- \( \text{thread}(satb) = \text{thread}(sa) / b, \) (a is the last move in sat justified by the same initial move as b),
- \( b \) is contingent on the most recent move in \( \text{thread}(sa) \) on which it is hereditarily contingent in \( sb \).

A strategy \( \sigma \) is well-opened if every control sequence in \( \sigma \) contains at most one initial move.

If \( \tau : A \rightarrow A \) is a well-opened strategy, then \( \tau^\dagger : A \rightarrow A \) is the least subset of \( C_A \) containing \( \varepsilon \) and closed under the condition:

- If \( s \in \tau^\dagger \), and \( \text{thread}(sb) \in \tau \), then \( sb \in \tau^\dagger \).

The well-opened identity is the restriction of \( \text{id}_A \) to well-opened sequences.

Thus we have two cartesian closed categories of games, both of which have arenas as objects and well-opened strategies over the function space \( A \Rightarrow B \) as morphisms from \( A \) to \( B \), with composition defined \( \sigma \cdot \tau = \tau^\dagger ; \sigma \).
$G, 1, x, \Rightarrow$ — the category of games, which has (general) strategies as morphisms — and $CG, 1, x, \Rightarrow$ — the category of control games which has control-strategies as morphisms.

Apart from exception-declaration and handling, the semantics of IAx is given by an embedding which takes the semantics of IA in $G$ [1] to $CG$. To define this embedding requires the notion of well-bracketing.

**Definition 3.7** A strategy $\sigma$ in $G$ is well-bracketed if every answer played by $\sigma$ is justified by the pending question. A control-strategy $\sigma$ in $CG$ is well-bracketed if every move made by $\sigma$ is contingent on the pending question: i.e. if $sa \in \sigma$, then $b$ is contingent on $\text{pending}(sa)$.

The well-bracketed strategies form cartesian closed subcategories of $G$ and $CG$, which will be written $GW$ and $CGW$. All of the strategies required to interpret IA [1] are well-bracketed.

**Definition 3.8** For any control sequence $s$, let $|s|$ be the underlying justified sequence obtained by forgetting the contingency pointers. For a control-strategy $\sigma$, let $|\sigma| = \{ |s| : s \in \sigma \}$. Say that $\sigma$ is control-blind if $|\sigma|$ is a deterministic strategy.

**Proposition 3.9** There is an embedding of $GW$ into $CG$, which has as its image the well-bracketed and control-blind strategies.

**Proof:** For any $\sigma : A \in GW$ define $\tilde{\sigma}$ to be the least subset of $CA$ containing $\varepsilon$ and closed under the following condition:

If $s \in \tilde{\sigma}$ and $|s\text{ab}| \in \sigma$ and $b$ is contingent on $\text{pending}(sa)$ then $s\text{ab} \in \sigma$.

Then $\tilde{\sigma}$ is a well-bracketed strategy (well-bracketedness of $\sigma$ implies that every answer is contingent on its justifying question and $\tilde{\sigma}$ is compositional and preserves cartesian closed structure). For any $\tau \in GW$, $|\tau| = \tau$, and for any well-bracketed and control-blind $\sigma \in CG$, $|\sigma| = \sigma$. □

4 Semantics of Exceptions

The interpretation of locally bound exceptions given here is based on viewing elements of exception type $h : \text{exn}$ as ‘objects’ defined by their ‘methods’ — in this case $\text{raise } h : \text{comm}$ and $\text{handle } h : \text{comm} \Rightarrow \text{comm}$. This was suggested as an interpretation for reference types by Reynolds [15] and followed in a game semantics setting in [1, 5].

The type $\text{exn}$ is interpreted as the arena $\text{exn} = ([0] \Rightarrow [[\text{comm}]] \times [0])$ (where $[[\text{comm}]]$ is the arena with one question and one answer, and $[0]$ is the arena $o$ with just a question). The initial questions in the two components $[0] \Rightarrow [\text{comm}]$ and $[0]$ will be referred to as $\text{handle}$ and $\text{raise}$ respectively. The answer to $\text{handle}$ will be referred to as $\text{caught}$, and the question enabled by $\text{handle}$ as $\text{ok}$. The handle and raise methods are the first and second projections from $\text{exn}; \text{mkeX}$ is pairing.

\[
\begin{align*}
\Gamma \vdash \text{handle } M \ N & = \langle \langle \Gamma \vdash M : \pi_1, \Gamma \vdash N : \text{App} \rangle \\
\Gamma \vdash \text{raise } M : B & = \langle \langle \Gamma \vdash M : \pi_2, \text{Wk}[B] \rangle \\
\Gamma \vdash \text{mkeX } M \ N & = \langle \langle \Gamma \vdash M, \Gamma \vdash N \rangle 
\end{align*}
\]

(Where $\text{Wk}[B] : o \Rightarrow A$ is the strategy which responds to the initial question in $A$ with the unique question in $\omega$.) Thus the only part of IAx which is not represented by a control-blind and well-bracketed strategy is new-exception declaration. This is defined using composition with a strategy $\text{xcell}$ (similar to the strategy cell which gives the denotation of new [1]) that uses contingency pointers in an essential way to match up raises and handles appropriately, via the notion of an open question.

**Definition 4.1** The set of prefixes of a control sequence which terminate in an open question is defined by induction on length, as follows:

open($\varepsilon$) = {};

open($sa$) = {sa}, if $a$ is not contingent,

if $b$ is contingent on $a$, then:

open($s\text{ab}$) = open($s$) if $A^\sim b = A$,

open($s\text{ab}$) = open($sa$) $\cup \{ sa \cdot tb \}$, otherwise.

![Figure 2. A typical play of xcell](image)

A “typical play” of $\text{xcell}$ is depicted in Figure 2 (arrows are contingency pointers). Its behaviour can be described informally as follows:

- If Opponent plays a handle move then $\text{xcell}$ responds with an ‘ok’ move, justified by (and contingent on) it.
4.1 Soundness

Soundness of the interpretation with respect to the operational semantics can now be established; the only novel feature of the proof is that it requires meanings to be assigned to programs which raise exceptions. Given $M : \text{comm}$ or $M : \text{nat}$, $E = e_1, \ldots, e_n$, $L = x_1, \ldots, x_m$, and $k \leq m$ such that $S(x_i) \Downarrow$ if and only if $i \leq k$, let $\text{new } L \downarrow S$ in $M \Downarrow$ if and only if $\text{new } L \downarrow S$ in $M \downarrow$. Then $[M, E, L, S] \Downarrow \text{new } L \downarrow S$ in $M$ if and only if $S(x_k) \Downarrow$ such that $S(x_k) \in \text{xcell}$.

Soundness is proved (by induction on derivation, using standard facts about the model together with analysis of xcell) with respect to the following binary approximation relation ($\rightsquigarrow$): $[M, E, L, S] \rightsquigarrow [M', E', L', S']$ if the last move in $[M, E, L, S]$ is the same as the last move in $[M', E', L', S']$.

Proposition 4.3 If $M, E, L, S \Downarrow D, E', L', S'$ then $[M, E, L, S] \rightsquigarrow [D, E', L', S']$.

The interpretation is also adequate. This follows directly from soundness and termination of all evaluations of Y-free terms (Proposition 2.1).

Proposition 4.4 For any IAx program $M : \text{comm}$, $[M] \neq \perp$ if and only if $M \Downarrow \text{skip}$.

Proof: The proof of completeness is by induction on the number of occurrences of Y in $M$. Suppose $[M] \neq \perp$. By proposition 2.1 $M \Downarrow D$ for some $D$ and $D \Downarrow \text{skip}$ by soundness. If $M = C[YN]$ for some Y-free $N$, then $C[YN] = \bigcup_{i \in \text{xcell}} [C[N^i]] \neq \perp$ (where $N^0 = \Omega$, and $N^{k+1} = N \cdot N_k$). Hence $C[N^i] \neq \perp$ for some $k \in \omega$ and by induction $C[N^i] \Downarrow$ and an induction on derivations shows that $C[YN] \Downarrow$. \qed

5 A Fully Abstract Model

An adequate model of IAx with exceptions has been described which is not fully abstract because it lacks the following ‘definability property’.

Definition 5.1 A model $\mathcal{M}$ of IAx has the definability property if for every context $\Gamma$ and type $T$, every (compact) $f : [\Gamma] \rightarrow [\Gamma]$ in $\mathcal{M}$ is definable; i.e., there exists an IAx term $M_f$ such that $f = [\Gamma \vdash M_f : T]$.

In this section, the category of control games will be cut down so that all compact strategies are definable in IAx by giving a series of semantic definability criteria, and hence a full abstraction result will be achieved. The criteria are based on constraining three aspects of behaviour on control games; which moves Player’s contingency pointers can point to (a variant of the bracketing condition), which moves Player’s justification pointers can point to (a variant of the visibility condition [9]), and a new condition governing which of Opponent’s contingency pointers can be observed by Player.

Definition 5.2 (Weak Bracketing) A strategy $\sigma$ is weakly bracketed if every Player move in $\sigma$ is contingent on an open question — i.e., if $sb \in \sigma$ where $b$ is contingent on $ta \subseteq sb$ then $ta \in \text{open}(s)$.

The notion of view, defined for justified sequences in [9], extends to control sequences in line with the intuition that when Player makes a move contingent on an earlier move it may be regarded as if they occurred in direct succession.

Definition 5.3 (View) The Player-view of a control sequence is defined as follows:

\[
\langle s \rangle a = a, \text{ if } a \text{ is initial.}
\]

\[
\langle s \rangle ab = \langle s \rangle b \text{ if } b \text{ is an O-move justified by } a,
\]

\[
\langle s \rangle b = \langle s \rangle b \text{ if } b \text{ is a P-question contingent on } a,
\]

\[
\langle s \rangle b = \langle s \rangle b \text{ if } b \text{ is a P-answer to } a.
\]

This accords with the original notion of views given in [9] in that for any well-bracketed strategy $\sigma \in \mathcal{G}$, $s \in \sigma$ implies that $\langle s \rangle = \langle [s] \rangle$. It “dualizes” to a notion of O-view ($\langle-\rangle$) as in [9].

Definition 5.4 (Visibility) A strategy ($\in \mathcal{G}$ or $\mathcal{G}$) satisfies the visibility condition if for every $s \in \sigma$, $\langle s \rangle$ is a well-defined justified sequence. The cartesian closed subcategory $\mathcal{G}$ of well-bracketed strategies satisfying visibility will be written $\mathcal{G}^V$.

(A well-bracketed) strategy satisfies visibility if and only if $[\sigma]$ satisfies visibility. Hence the embedding of $\mathcal{G}^V$ into $\mathcal{G}$ restricts to $\mathcal{G}^V$.

The third definability criterion limits the power of Player to observe contingency pointers. (It corresponds
to the fact that in IAx the only way to observe exception handling is by raising and handling a competing exception.)

Let \( satb \) be a control sequence in which \( b \) is a \( P \)-move contingent on \( a \), and let \( rc \in satb \) where \( c \) is a \( O \)-move. Then \( c \) is prematurely closed by \( b \) if \( rc \in \text{open}(sat) \) and \( rc \notin \text{open}(sa) \). Player's perspective on a control sequence is obtained by deleting the contingency pointers which are not attached to \( O \)-questions prematurely closed by some \( P \) move. It can be defined concisely (for extensions to control-sequences) as follows:

\[ [\varepsilon] = \varepsilon. \]

If \( b \) is contingent on \( a \), then \([satb] = [s]a[t]b\), where the pointer from \( b \) to \( a \) is included if and only if \( a \) is a \( P \)-question, and all of the pointers from \( a[t] \) into \( s \) are omitted.

**Definition 5.5** A strategy is control-innocent if whenever \( satb, t \in \sigma \) and \([satb] = [tab]\), then \( tab \in \sigma \).

**Proposition 5.6** If \( \sigma \) is well-bracketed and control-innocent then \( \sigma \) is control-blind.

**Proof:** If \( \sigma \) is a well-bracketed strategy, then \([s] = [s] \) as \( \sigma \) closes only pending \( O \)-questions. \( \square \)

Hence the image of the embedding of \( G_{AB} \) into \( G \) consists of the well-bracketed strategies satisfying visibility and control-innocence. The following proposition is just a straightforward extension of the definability theorem for IA [1] to include the base type \( \text{exm} \).

**Proposition 5.7** All finite strategies in \( G_{AB} \) over \( IAx - \{\text{new,exm}\} \) type-objects are definable in \( IAx - \{\text{new,exm}\} \).

**Corollary 5.8** The (compact) definable strategies of \( IAx - \{\text{new,exm}\} \) are the well-bracketed and control-innocent finite strategies which satisfy visibility.

### 5.1 Factorization and Definability

The finite, well-bracketed, visibility-satisfying and control-innocent strategies can now be identified as the compact \( IAx \)-definable morphisms by showing that they are obtained by composing \( xcell \) with the well-bracketed and control blind strategies.

**Definition 5.9** Define \( CG_{/xcell} \) to be the cartesian closed subcategory of control games in which morphisms are finite strategies \( f : A \to B \) such that there exists \( k \in \omega \) and a well-bracketed strategy \( g : A \times \text{exm}^k \to B \) such that \( \text{id} \times xcell^k \cdot g = f \).

**Proposition 5.10** The compact elements of \( CG \) which are definable in \( IAx \) are precisely the morphisms of \( CG_{/xcell} \).

**Proof:** It is straightforward to establish by structural induction on \( M \) that every \( \Gamma \vdash M : T \) is the least upper bound of a chain of approximants in \( CG_{/xcell} \).

Conversely, if \( \sigma : \Gamma \vdash M : T \) is a morphism in \( CG_{/xcell} \) then there is a well-bracketed — and hence \( IAx- \{\text{new,exm}\} \) definable — strategy \( \hat{\sigma} : [\Gamma] \times \text{exm}^k \to (A \Rightarrow B) \) such that \( \sigma = \text{id}_{[\Gamma]} \times xcell^k \cdot \hat{\sigma} \) and hence \( \sigma = [\Gamma] \times \text{new,exm}_A \times \ldots \text{new,exm}_A \text{,M}_A \). \( \square \)

**Proposition 5.11** A strategy is in \( CG_{/xcell} \) if and only if it is finite, weakly-bracketed, control-innocent and satisfies visibility.

**Proof** of this proposition comes in two parts: first it is shown that if \( \sigma : \text{exm} \Rightarrow A \) satisfies weak-bracketing, control-innocence and visibility then so does \( xcell \cdot \sigma \), which is a consequence of the following two lemmas.

**Lemma 5.12** Suppose \( \sigma : \text{exm} \Rightarrow A \) and \( sa \in \sigma \) is such that \( a \) is a move in \( A \), and \( s | \text{exm}, t | \text{exm} \in xcell \) and \( b \) is a move in \( A \) such that \([satb]A = [tab]A \). Then \( tab \in \sigma \).

The second part of the proof of Proposition 5.11 is to show that all weakly-bracketed strategies can be obtained from well-bracketed strategies by composition with \( xcell \) (so in fact the stronger result that every compact strategy can be defined using a single exception variable is established). This is achieved by methods similar to the factorizations described in [1, 10, 5, 8], in this case using the jump in control between the \text{raise} and \text{caught} moves of \( xcell \) to generate all of the control jumps in a weakly-bracketed strategy. The complicating factor is that the properties of control-innocence and visibility must be maintained. In particular, forcing Opponent to close questions instead of Player hides their contingency pointers — as has already been observed, well-bracketed strategies cannot observe any pointers at all. So it is necessary to make all of the information carried by the perspectives of \( \sigma \) manifest as explicit exception handling.

The factorization of a strategy \( \sigma : B \to \hat{\sigma} : \text{exm} \Rightarrow B \) by adding \text{handle, ok} and \text{raise, caught} moves in \text{exm} (see Figure 3), can be informally described as follows.

- Immediately before playing a question in \( A \), \( \hat{\sigma} \) plays a \text{handle} move (contingent on the pending question), to which Opponent responds with \text{ok}.
- If \( \sigma \) responds to \( sa \) by playing a move \( b \) which prematurely closes \( n \) Opponent moves then \( \hat{\sigma} \) responds to \( \hat{sa} \) by playing \( n \) \text{raise} moves — each of
which is caught by a handler corresponding to one of the $O$-moves which are closed by $b$—until all of these $O$-moves have been closed. If $b$ is an answer then $\hat{\sigma}$ plays $b$ contingent on the pending question; if $b$ is a question, then $\hat{\sigma}$ plays a handle (as above) and then plays $b$ pointing to the pending question.

Note that as all of the contingency pointers from the control-view of $\sigma$ are used to match up the raises and handles, they are now observable as play in the premiss $\text{exn}$.

**Proposition 5.14 (Control Factorization)** If $\sigma : A$ is a finite, weakly-bracketed and control-innocent strategy (satisfying visibility) then there is some finite, well-bracketed strategy $\hat{\sigma} : \text{exn} \to A$ (satisfying visibility) such that $\chi_{\text{cell}} \hat{\sigma} = \sigma$.

**Proof:** is by defining $\hat{\sigma} = \{ \bar{t} \in \text{even} \mid s \in \sigma \}$, where $\chi : C_A \to C_{\text{even} \Rightarrow A}$ is a translation on even-length control sequences such that:

- $\hat{\sigma}[A = s$ and $\hat{\sigma}[\text{exn}] \in \chi_{\text{cell}}$,
- every Player move in $\hat{\sigma}$ is contingent on the pending question,
- if $\Gamma \hat{\sigma}$ is well-defined then so is $\Gamma \hat{\sigma}$,
- $[s] = [\bar{t}]$ if and only if $\hat{\sigma} = \bar{t}$.

For an even-length sequence $s$, let $\psi(s)$ be the number of $O$-questions prematurely closed by the last move in $s$. Now define $\hat{\sigma}$ by induction on sequence length:

$$\hat{\sigma} = \varepsilon,$$

$$\hat{\sigma}\bar{p} = \hat{\sigma}(\text{raise} \text{caught}) \psi(s)p \text{handle} \text{ok}, q,$$

$$\hat{\sigma}\bar{a} = \hat{\sigma}(\text{raise} \text{caught}) \psi(s)a, q,$$

where $\chi_{\text{CA}}(q) = Q$, and $\chi_{\text{CA}}(a) = A$.

\[\square\]

### 5.2 Full Abstraction

In a now-standard fashion, definability for the compact elements of the model of IAx yields full abstraction for its "intrinsic preorder collapse". Moreover, this fully abstract model can be described directly, showing that it is effectively presentable.

**Definition 5.15** Given strategies $\sigma, \tau : A$, $\sigma \preceq_A \tau$ if for every well-bracketed strategy $\rho : A \to [\text{comm}], \sigma \rho \neq [\varepsilon]$ implies $\tau \rho \neq [\varepsilon]$.

$\sigma \equiv \tau$ if $\sigma \preceq_A \tau$ and $\tau \preceq_A \sigma$.

**Theorem 5.16 (Full Abstraction)** For any closed terms $M, N : T$, $[M]_{\text{CG}} \equiv [N]_{\text{CG}}$ if and only if $M \simeq N$.

In fact, the equationally fully abstract model can be directly presented simply by including visibility and bracketing in the definition of a legal play.

**Definition 5.17** An alternating control sequence $s$ over an arena $A$ is legal if both Player and Opponent satisfy the weak-bracketing and visibility conditions: i.e. $\sigma a \in L_A$ if and only if $s \in L_A$ and if $a$ is contingent on $b$ then $b \in \text{open}(s)$, and $\sigma a$ and $\sigma b \in L_A$ are both well-formed justified sequences.

For $\sigma : A$, write $L(\sigma)$ for $\sigma \cap L_A$.

**Lemma 5.18** For any $\sigma, \tau : A$, $L(\sigma) \subseteq L(\tau)$ if and only if $\sigma \preceq_A \tau$.

**Proof:** To prove the implication right to left (showing that control-innocence does not affect the intrinsic preorder) suppose $L(\sigma) \subseteq L(\tau)$. Let $s \bar{b} \in \sigma$ be a minimal-length control sequence such that $s \bar{b} \notin \tau$. Let $q$ be the initial question in $\text{comm}$ and $a$ its answer, and define:

$$\rho : A \to [\text{comm}] = \{ t \in C_A \text{ even} \mid [t] \simeq \text{even} \text{ [}qs[ba] \}$$

Then $\rho$ is by definition a control-innocent strategy such that $\rho \sigma \neq [\varepsilon]$. Moreover, $\rho \tau = [\varepsilon]$—if $s \bar{c} \in \rho$ then
6 Contingency and Expressiveness

The complex structure of control games provokes the question: Are contingency pointers really necessary to model exceptions? The category $G$ contains models of a wide range of sequential features, including both references $[1, 5]$ and call/cc $[10, 11]$ at all types. Might there not be a semantics of exceptions in $G$? Part of the interest in this question arises because it is closely related to a problem which is both independent of semantics, and an area of current research interest: When can one combination of programming language features be macro-expressed in terms of another $[17, 18]$? For example, it is “folklore” $[18]$ that exceptions may be expressed in terms of continuations and references. As both of the latter can be modelled in $G$, if the folklore were true then a semantics of exceptions in $G$ could be given by factoring through this interpretation. On the other hand, given a model of exceptions, continuations and references in $G$ it should be possible to use the a combination of the definability results for references and continuations to extract an encoding of exceptions.

In fact, it is not possible to give a semantics of exceptions which is a conservative extension of the model of IA in $G$, and hence it is not possible to macro-express exceptions using continuations and references. A paper is in preparation which contains formal proofs of the latter claim, using syntactic counterexamples extracted from the game semantics of exceptions, continuations and references. This section will sketch a proof of the former claim, showing how differences in contingency structure can cause differences in observable behaviour. A starting point is the observation there are strategies which contain the same underlying justified sequences, but are observationally distinct because they have different contingency pointers. A simple arena in which this may be observed is $(o \Rightarrow o) \Rightarrow (o \Rightarrow o)$ which will be called $A_1$ for short; it is the denotation of the type $T_1 = (0 \Rightarrow 0) \Rightarrow (0 \Rightarrow 0)$. Recall that in the identity strategy on $o \Rightarrow o$ (Figure 1) Player moves are always contingent on the preceding O-move.

**Proposition 6.1** There exists a weakly bracketed, visibility-satisfying and control-innocent strategy $\text{not}$.id such that $[\text{not}$.id$] = [\text{id}_{\llcorner o\lrcorner}]$ but id $\neq$ not.id.

**Proof:** Let $\text{not}$.id be the strategy consisting of the even prefixes of the play depicted in Figure 4. Then $[\text{not}$.id$]_{\llcorner o\lrcorner} = [\text{id}_{\llcorner o\lrcorner}]$ but $\text{not}$.id $\neq$ id by Lemma 5.18. Moreover (as the definability and full abstraction results imply) not.id and id are the denotations of terms which are not observationally equivalent: $\text{id} = [\lambda f$.f$], \text{not}$.id $= [\text{NOT}.ID]$ where $\text{NOT}.ID = \lambda x$.new.$\lambda x$.handle $f$ (f raise $h$); $x$. NOT.ID $\neq$ $\lambda f$.f; let ID.TEST : $T_1 \Rightarrow \text{comm} = \lambda z$.new.$\lambda x$.handle $h (((g \lambda h.$raise $h z) \raise h) \Omega)$. ID.TEST NOT.ID $\downarrow$ skip and ID.TEST $\lambda f$.f $\downarrow$ skip.

The distinction between not.id and id can be used to show that there is no model of IA in $G$.

**Definition 6.2** Define the $G$-strategy $\text{id}$.trunc : $A_1 \rightarrow A_1 = \{ t \in L_{A_1} | t | A_1^{+} = t | A_1^{+} \in \text{id}_{\llcorner o\lrcorner} \}$

As the definability result of [1] entails, id.trunc is definable as a term of Idealized Algol:

$\text{TRUNC} = \lambda g$.new.$\lambda g$.f.$\lambda f$.z := 0 ((g M_1) M_2), where $M_1 = \lambda y$.IF0 !z then $z := 1$; (f y)) else $\Omega$ and $M_2 = \Omega$ then $\Omega$ else $x$.

**Lemma 6.3** For all $\sigma : A_1 \in G$, $\sigma$.id.trunc $\leq A_1$, $\text{id}_{\llcorner o\lrcorner}$

**Proof:** This is direct by definition of id.trunc.

In $G$, not.id.id.trunc $\neq$ not.id $\llcorner A_1, \text{id}_{\llcorner o\lrcorner}$ and this fact can be exploited to prove the following.

**Proposition 6.4** There is no adequate model of IA in $G$ which conservatively extends the semantics of IA.

**Proof:** Suppose there is such an interpretation. Then ID.TEST (TRUNC NOT.ID) $\downarrow$ skip implies that $[[\text{ID}.TEST (\text{TRUNC NOT.ID})]]_{G} \neq \bot$, and hence $[[\text{NOT}.ID]]_{G} \neq \bot$. By Lemma 6.3, $[[\lambda f$.f$]$; ID.TEST]$_{G} \neq \bot$, and so $[[\text{ID}.TEST \lambda f$.f$]]_{G} \neq \bot$. But this contradicts adequacy, as ID.TEST $\lambda f$.f $\downarrow$ skip.

6.1 Further Directions

By demonstrating that the games semantics of exceptions requires new structure, unlike the models of

\[
(o \Rightarrow o) \Rightarrow (o \Rightarrow o)
\]

![Figure 4. A typical play of not.id](image)
references and continuations, we have shown the limitations of the "semantic cube" [4] of models of programming language features based simply upon relaxing constraints on the original model of PCF. But the basic analysis implicit in the cube is strengthened by the new structure — in effect, we have added an extra dimension to it. The extra degree of freedom available in the category of control games can be exploited to give a thorough analysis of the interactions between exceptions, continuations and references. The latter can be modelled by dropping the "visibility condition" in the style of [1] to reach a fully abstract semantics of "core ML". (It is straightforward to move to a call-by-value perspective by using, for instance, the Fam(C) construction [2].)

To allow call/cc to be interpreted, the weak bracketing condition is relaxed. In this model, throwing a continuation and and handling an exception both correspond to playing a move which is not contingent on the pending question; the distinguishing feature of exceptions is that they allow contingency pointers to be observed. The most interesting feature of the model is that (unlike the model of continuations in Σ [11]) it is not an example of continuation-passing-style construction; it contains observably distinct strategies which represent terms which are equivalent in all call stack models. These terms constitute a further counterexample to the claim that exceptions can be expressed using continuations which can be presented without recourse to the game semantics.

References


