

# Decidability in Syntactic Control of Interference

J. Laird

Dept. of Informatics, University of Sussex, UK.  
e-mail:jiml@sussex.ac.uk

**Abstract.** We investigate the decidability of observational equivalence and approximation in “Syntactic Control of Interference” (SCI). By associating denotations of terms in an inequationally fully abstract model of finitary basic SCI with multitape finite state automata, we show that observational approximation is not decidable (even at first order), but that observational equivalence is decidable for all terms. We then consider the same problems for basic SCI extended with non-local control in the form of backwards jumps. We show that both observational approximation and observational equivalence are decidable in this language by describing a fully abstract games model in which strategies are regular languages.

## 1 Introduction

Reynolds’ *Syntactic Control of Interference* [18] is a prototypical functional-imperative language in which covert *interference* between functions and their arguments is prevented by the use of an affine typing discipline. By eliminating phenomena such as aliasing, interference control should make it easier to predict program behaviour. Here, we shall investigate decidability of properties of observational approximation and equivalence for terms over finite datatypes.

The full SCI system contains a notion of passive type (as do related languages with interference control such as SCIR [15]); the programs (and contexts) typable at purely passive types are essentially those of PCF, and equivalence at such types is therefore not decidable [9]. Consequently, we restrict attention to the active types or *basic* SCI, which may be thought of as a common core for interference controlled languages. Basic SCI is essentially a subsystem of Idealized Algol, for which a hierarchy of decidability results has recently been developed, based on “algorithmic game semantics” [3]: associating the denotation of each term as a strategy in a fully abstract game semantics with a formal language. These techniques have been used to show that observational equivalence and approximation in (finitary) Idealized Algol are decidable at third-order types — by showing that the denoting strategies are recognized by deterministic pushdown automata [14] — but not at fourth order [13]. The latter undecidability result depends crucially on the nesting of function calls, which is not possible in SCI.

These results provide motivation and a methodological basis for investigating observational equivalence in basic SCI, but one obstacle is that game semantics appears to be “too sequential” to capture this directly. (McCusker and Wall

have described a games model [12], but the associated full abstraction result depends on a quotient operation.) However, basic SCI has an appealingly simple semantics based on sets and relations, described by Reddy [17], and investigated further by McCusker, who proved that it is fully abstract [10]. We show that the denotations of first-order terms in this semantics correspond to *multitape* deterministic finite state automata, as introduced by Rabin and Scott in 1959 [16]. It is straightforward to show that containment of 2-tape deterministic FSA is undecidable [7]. By contrast, the problem of decidability of equivalence for all multitape deterministic FSA remained open for thirty years, before being resolved (affirmatively) by Harju and Karhumäki [4]. These results have direct implications for basic SCI: observational equivalence (equivalence of denotations) is decidable at first order but observational approximation (inclusion of denotations) is not (even though all first-order terms of Idealized Algol are typable in SCI (up to  $\beta$ -equivalence) — the point being that the possible *observations* of first-order terms are more restricted in SCI). Although denotations of terms at higher-order types do not correspond directly to multitape automata, we show that there is a “definable monomorphism” from every type to a first-order type, and that equivalence is therefore decidable at *all* types.

The undecidability of observational approximation (and associated results, such as the undecidability of equivalence in a non-deterministic variant of the language) raises the question of whether deciding such properties in the presence of interference control is always difficult. The problem in basic SCI is in some respects analogous to the situation in PCF: whilst it is a sequential language, it is not *observably sequential*; different sequentializations of a program may approximate each other (for example,  $\lambda x.\lambda y.x; y$  is equivalent in SCI to  $\lambda x.\lambda y.y; x$ ). In the case of PCF, Cartwright and Felleisen [2] have shown that adding simple non-local control operators results in a language (“observably sequential PCF”) with a finitely presentable fully abstract model, for which equivalence and approximation are therefore decidable. Pursuing the analogy between PCF and SCI, we obtain an observably sequential version of the latter by adding backwards jumps to labelled points. We give a regular game semantics for this language and show that it is inequationally fully abstract (although it contains compact elements which are not definable) and that observational approximation is therefore decidable.

## 2 Basic SCI

The syntax of basic SCI [18] is, in essence, the same as that of Idealized Algol [19] but the typing rules are more restrictive. In our finitary version of the language we assume a base type  $\underline{n}$  (of expressions) for each natural number  $n$ , containing the numerals  $0, \dots, n - 1$ , and generate types using the constructors  $\rightarrow$  (affine function-space) and  $\&$  (additive product). The depth or order of a type (and the corresponding closed terms) is defined:  $o(\underline{n}) = 0$ ,  $o(S\&T) = \max\{o(S), o(T)\}$ ,  $o(S \rightarrow T) = \max\{o(S) + 1, o(T)\}$ .

Typing judgements are based on the affine  $\lambda$ -calculus with pairing:

$$\frac{}{x:T, \Gamma \vdash x:T} \qquad \frac{\Gamma, x:S, \Delta \vdash M:T}{\Gamma, \Delta, x:S \vdash M:T}$$

$$\frac{\Gamma \vdash M:S \quad \Gamma \vdash N:T}{\Gamma \vdash \langle M, N \rangle : S \& T} \quad \frac{\Gamma, x:S \vdash M:T}{\Gamma \vdash \lambda x.M : S \rightarrow T} \quad \frac{\Gamma \vdash M:S \rightarrow T \quad \Delta \vdash N:S}{\Gamma, \Delta \vdash M N:T}$$

For the sake of concision, we work with a lean syntax which can be easily sugared up to more closely resemble that of Idealized Algol. We write  $T^n$  for the  $n$ -fold product of copies of  $T$  (setting  $T^0 = \underline{1}$ ). We represent the type  $\text{var}[n]$  of imperative variables with values in  $\underline{n}$  as the type  $\underline{n} \& \underline{1}^n$  (the product of the types of its “methods”, assignment and dereferencing).<sup>1</sup> We extend the term formation rules with the following constants (we omit parallel composition as it is observationally equivalent to sequential composition):

**Projections**  $\pi_l : A \& B \rightarrow A$ ,  $\pi_r : A \& B \rightarrow B$  yielding  $\pi_i : A^n \rightarrow A$  for  $i < n$ .

**Numerals**  $i : \underline{n}$  for  $i < n$ .

**Case statements**  $\text{case}_{n,m} : \underline{n} \& \underline{m}^n \rightarrow \underline{m}$ . from which we derive  $\text{case}_{n,T} : \underline{n} \& T^n \rightarrow T$  for arbitrary  $T$ .

**Loops**  $\text{while0} : \underline{m+1} \rightarrow \underline{m}$ . We write  $\Omega$  for  $\text{while0 } 0$ .

**New variable declaration**  $\text{new} : (\text{var}[n] \rightarrow \underline{m}) \rightarrow \underline{m}$ .

If  $M : \underline{1}$ , we write  $\text{case} \langle M, N \rangle$  as  $M; N$ , and if  $M : \underline{0}$  we write  $\text{abort} M$  for  $\text{case} \langle M, 0 \rangle$ . Given  $M : \text{var}[n]$ , we may write  $!M$  for  $\pi_l M$  and  $M := N$  for  $\text{case} \langle N, \pi_r M \rangle$ .

The “small-step” operational semantics is based on the evaluation contexts:

$$E[\cdot] ::= [\ ] \mid E[\cdot] M \mid \pi_i E[\cdot] \mid \text{case } E[\cdot] \mid \text{case} \langle E[\cdot], M \rangle \mid \text{while0 } E[\cdot]$$

and the following reductions for pairs  $M, \mathcal{E}$  of a term (of ground type) and a store (a set of pairs  $\langle a, v \rangle$  of location names and natural numbers).

$$\begin{aligned} E[(\lambda x.M) N], \mathcal{E} &\longrightarrow E[M[N/x]], \mathcal{E} \\ E[\pi_i \langle M_1, M_2 \rangle], \mathcal{E} &\longrightarrow M_i, \mathcal{E} \\ E[\text{case} \langle i, M \rangle], \mathcal{E} &\longrightarrow E[\pi_i M] \\ E[\text{while0 } M] &\longrightarrow E[\text{case} \langle M, \langle \text{while0 } M, 0, \dots, 0 \rangle \rangle] \\ E[\text{new } M], \mathcal{E} &\longrightarrow E[M a], \mathcal{E} \cup \{ \langle a, 0 \rangle \} \quad a \notin \pi_1(\mathcal{E}) \\ E[\pi_l a], \mathcal{E} &\longrightarrow E[v], \mathcal{E} \quad \langle a, v \rangle \in \mathcal{E} \\ E[\pi_i (\pi_l a)], \mathcal{E} &\longrightarrow E[0], \mathcal{E}[a \mapsto i] \end{aligned}$$

We write  $M \Downarrow$  if evaluation of  $M, \emptyset$  terminates, and so define standard notions of observational approximation and equivalence:

$M \lesssim N$  if  $C[M] \Downarrow$  implies  $C[N] \Downarrow$ , and  $M \simeq N$  if  $M \lesssim N$  and  $N \lesssim M$ .

<sup>1</sup> So this version of the language implicitly contains “bad variables”. This does not affect the results given here for basic SCI: approximation is shown to be undecidable for the **var**-free types, whilst McCusker has shown that observational equivalence in basic SCI with bad variables is conservative over equivalence in the language without bad variables [11].

## 2.1 Fully Abstract Semantics of Basic SCI

We briefly describe the fully abstract model of basic SCI, based on sets and relations, which we shall use to prove our results. This is essentially a version of Reddy's semantics [17], simplified and proved to be fully abstract by McCusker [10]. We may present this model as a typed **BCK**-algebra [5] with products:

- A collection of objects and a set  $\text{el}(\alpha)$  of elements for each object. In the relational model, objects are sets and  $\text{el}(\alpha) = \mathcal{P}(\alpha)$ .
- Objects  $\alpha \rightarrow \beta$  and  $\alpha \& \beta$  for each pair of objects  $\alpha, \beta$ . In the relational model, we define  $\alpha \rightarrow \beta$  to be the set  $\alpha^* \times \beta$  (where  $\alpha^*$  is the set of finite words over  $\alpha$ ) and  $\alpha \& \beta$  to be the disjoint union  $\alpha + \beta = \{e^l \mid e \in \alpha\} \cup \{e^r \mid e \in \beta\}$ . So in the relational model, elements of  $\alpha \rightarrow \beta$  are relations between  $\alpha^*$  and  $\beta$ , and elements of  $\alpha \& \beta$  correspond to pairs of elements from  $\alpha$  and  $\beta$ .
- An application operation — a function  $\cdot : \text{el}(\alpha \rightarrow \beta) \times \text{el}(\alpha) \rightarrow \text{el}(\beta)$ . In the relational model  $X \cdot Y = \{z \in \beta \mid \exists \mathbf{y} \in Y^*. \langle \mathbf{y}, z \rangle \in X\}$ .
- For each  $\alpha, \beta, \gamma$ , elements  $\mathbf{B} \in \text{el}((\beta \rightarrow \gamma) \rightarrow (\alpha \rightarrow \beta) \rightarrow \alpha \rightarrow \gamma)$ ,  $\mathbf{C} \in \text{el}((\alpha \rightarrow \beta \rightarrow \gamma) \rightarrow \beta \rightarrow \alpha \rightarrow \gamma)$ ,  $\mathbf{T} \in \text{el}(\alpha \rightarrow \beta \rightarrow \alpha \& \beta)$ ,  $\mathbf{P} \in \text{el}((\alpha \rightarrow \beta) \& (\alpha \rightarrow \gamma) \rightarrow \alpha \rightarrow \beta \& \gamma)$ ,  $\pi_l \in \text{el}(\alpha \& \beta \rightarrow \alpha)$  and  $\pi_r \in \text{el}(\alpha \& \beta \rightarrow \beta)$  satisfying the axioms:

$$\mathbf{B} \cdot x \cdot y \cdot z = x \cdot (y \cdot z),$$

$$\mathbf{C} \cdot x \cdot y \cdot z = x \cdot z \cdot y$$

$$\pi_l \cdot (\mathbf{T} \cdot x \cdot y) = x \text{ and } \pi_r \cdot (\mathbf{T} \cdot x \cdot y) = y$$

$$\mathbf{P} \cdot (\mathbf{T} \cdot x \cdot y) \cdot z = \mathbf{T} \cdot (x \cdot z) \cdot (y \cdot z)$$

(note that we may define  $\mathbf{K} \in \text{el}(\alpha \rightarrow \beta \rightarrow \alpha)$  to be  $\mathbf{B} \cdot (\mathbf{B} \cdot \pi_r) \cdot \mathbf{T}$ ). In the relational model we define e.g.  $\mathbf{B} \in \text{el}((\beta \rightarrow \gamma) \rightarrow (\alpha \rightarrow \beta) \rightarrow \alpha \rightarrow \gamma) =$

$$\{\langle \langle \mathbf{b}, c \rangle, \langle \mathbf{e}, \langle \mathbf{a}, c \rangle \rangle \mid \mathbf{a} \in \alpha^*, \mathbf{b} \in \beta^*, c \in \gamma, \mathbf{e} \in (\alpha^* \times \beta)^*. \mathbf{e} | \alpha = \mathbf{a} \wedge \mathbf{e} | \beta = \mathbf{b} \}$$

**BCK**-algebras yield an interpretation of *derivations* in the affine  $\lambda$ -calculus (as standard combinatory algebras do for the  $\lambda$ -calculus), and the extension with products is straightforward. Thus to interpret basic SCI, we define  $\llbracket n \rrbracket = \lceil n \rceil = \{i \mid i < n\}$ , and assign meanings to the remaining constants as follows:

- $\llbracket \text{case}_{m,n} \rrbracket = \{\langle i^l j^{ir}, j \rangle \mid i < m \wedge j < n\}$ ,
- $\llbracket \text{while} \mathbf{0} \rrbracket = \{\langle 0^*(n+1), 0 \rangle \mid s \in \{i < n \mid i \neq 0\}^*\}$ ,
- $\llbracket \text{new} \rrbracket = \{\langle \langle (0^l)^* s, j \rangle, j \rangle \mid s \in \{(0^i)^l (i^r)^* \mid i < n\}^*\}$

Concretely, this interpretation is equivalent to Reddy's and McCusker's (for closed terms), and therefore fully abstract.

**Theorem 1 (McCusker [10]).**  $M \lesssim N$  if and only if  $\llbracket M \rrbracket \subseteq \llbracket N \rrbracket$ .

Note that the semantics does not have the “finite definability” property: there are compact elements which are not the denotation of any term. In particular, we may interpret a non-deterministic choice between any two terms of the same type as the union of their denotations.

### 3 Multitape Automata and SCI Terms

Our definition of deterministic multitape finite state automata is tailored for establishing a correspondence with the relational model. However, it is straightforward to show that it yields the same class of languages as the original [16].

A deterministic  $n$ -tape automaton  $\alpha = (C_0, C_1, \dots, C_n, F, s_0, \delta)$  over an alphabet  $\Sigma = \{0, \dots, k\}$  consists of disjoint sets of states  $C_0, C_1, \dots, C_n, F$ , an initial state  $s_0$  and a (partial) transition function  $\delta : (\Sigma \cup \{\varepsilon\}) \times S \rightarrow S$ , where  $S = C_0 \cup \dots \cup C_n \cup F$ .  $C_0$  consists of states in which (only) a  $\varepsilon$ -transition may be performed,  $C_i$  of states in which a symbol is read from tape  $i$  (for  $0 < i \leq n$ ), and  $F$  of final states. We say that  $\alpha$  accepts a tuple of tapes if it consumes all of them to reach a final state. Formally, for each final state  $f$ , we may say that the set of pairs  $A_f \subseteq (\Sigma^*)^n \times S$  of tuples and states accepted in state  $f$  is the least set such that  $\langle \langle \varepsilon, \dots, \varepsilon \rangle, f \rangle \in A_f$ , and if  $\langle \langle t_1, \dots, t_n \rangle, s \rangle \in A_f$ ,  $s' \in C_i$  and  $\delta(m, s') = s$  then if  $i = 0$  then  $\langle \langle t_1, \dots, t_n \rangle, s' \rangle \in A_f$  and if  $i > 0$  then  $\langle \langle t_1, \dots, mt_i, \dots, t_n \rangle, s' \rangle \in A_f$ .

We write  $\mathcal{L}(\alpha)$  for the set  $\{x \mid \exists f \in F. \langle x, s_0 \rangle \in A_f\}$  of tuples accepted by  $\alpha$ .

There is a natural connection between multitape FSA and SCI terms at first-order types, based on the fact that the denotations of the latter are sets of tuples of words over a finite alphabet.

**Proposition 1.** *For any 2-tape deterministic FSA  $\alpha$  over the alphabet  $\{0, \dots, m-1\}$  there is a (closed) term  $M_\alpha : \underline{m} \rightarrow \underline{m} \rightarrow \underline{1}$  such that  $\langle s, t \rangle \in L(\alpha) \Leftrightarrow \langle s, \langle t, 0 \rangle \rangle \in \llbracket M_\alpha \rrbracket$ .*

*Proof.* Every deterministic  $n$ -tape FSA is equivalent to one with no  $\varepsilon$  transitions, so we assume  $C_0$  is empty. We number the states of  $\alpha$  as  $0, \dots, k-1$ , with  $0$  being initial. We assume terms  $\mathbf{state} : \underline{k} \rightarrow \underline{3}$  such that  $\llbracket \mathbf{state} s \rrbracket = 0$  if  $s \in F$  and  $\llbracket \mathbf{state} s \rrbracket = i$  if  $s \in C_i$ , and  $\mathbf{tr} : \underline{k} \rightarrow \underline{k}^m$  such that  $\llbracket \pi_i (\mathbf{tr} j) \rrbracket = \delta(j, i)$ .  $M_\alpha$  may then be defined as follows:

```
 $\lambda x. \lambda y. \mathbf{new} \lambda s. \mathbf{while0} (\mathbf{case} \langle \mathbf{state} !s, \langle 1, s := \mathbf{case} \langle x, \mathbf{tr} !s \rangle, s := \mathbf{case} \langle y, \mathbf{tr} !s \rangle \rangle \rangle)$ 
```

So we may show undecidability of inclusion of denotations in the fully abstract model, and hence of observational approximation in SCI via the following result, which may be proved via an encoding of Post's correspondence problem.

**Proposition 2.** [7] *Inclusion of deterministic 2-tape FSA is undecidable.*

**Corollary 1.** *Observational approximation in SCI is undecidable at first-order.*

Since  $M \lesssim N$  if and only if  $M$  or  $N \simeq N$ , this entails that observational equivalence (w.r.t. may-testing) in SCI with erratic choice is undecidable at first order. The denotations of first-order terms *without loops* are finite sets, and therefore approximation for such terms is decidable at first-order. However, we can simulate any 2-tape FSA as a term of type  $(\underline{1}^n \rightarrow \underline{1}) \rightarrow (\underline{1}^n \rightarrow \underline{1}) \rightarrow \underline{1}$  (as in Proposition 1) without using the `while0` constant, and so approximation is undecidable at second order for the loop-free language.

The situation with respect to *equivalence* of multitape automata (and hence, as we shall show, observational equivalence of SCI) is different.

**Theorem 2 (Harju and Karhumäki [4]).** *Equivalence of  $n$ -tape deterministic finite state automata is decidable for all  $n$ .*

To use this result to show decidability of SCI equivalence, we prove a converse to Proposition 1: the denotation of every first-order term corresponds to the language accepted by a multitape automaton.

**Proposition 3.** *For each term  $x_1 : T_1, \dots, x_n : T_n \vdash M : \underline{m}$ , where each  $T_i$  is a product of ground types, there is a deterministic  $n$ -tape FSA with final states  $F = f_0, \dots, f_{m-1}$  which accepts  $\langle s_1, \dots, s_n \rangle$  in final state  $f_i$  if and only if  $\langle \langle s_1, \dots, s_n \rangle, i \rangle \in \llbracket M \rrbracket$ .*

*Proof.* We may show using a reducibility argument that every first-order term is reducible to a  $\beta\pi$ -normal form (i.e. no subterms of the form  $(\lambda x.P)Q$  or  $\pi_i \langle M, N \rangle$ ). So we assume that  $M$  is  $\beta\pi$ -normal, and define an  $n$ -tape deterministic automaton  $(C_0^M, \dots, C_n^M, F, s_0^M, \delta^M)$  with the additional property that if  $T_i = \overline{m_1} \& \dots \& \overline{m_k}$  then for any state  $s \in C_i^M$ , if  $\delta^M(v^i, s)$  and  $\delta^M(u^j, s)$  are both defined then  $i = j$ .

- If  $M = i$ , then  $C_j^M = \emptyset$  for all  $j$ ,  $s_0^M = f_i$  and  $\delta^M = \emptyset$ .
- If  $M = \pi_j(x_i)$ , then  $C_i^M = \{s\}$ ,  $C_j^M = \emptyset$  for  $j \neq i$ ,  $s_0^M = s$ ,  
 $\delta(l^k, s_0) = f_l$  if  $k = j$ , and  $\delta(m, s)$  is undefined otherwise.
- If  $M = \mathbf{case} \langle L, N_0, \dots, N_{k-1} \rangle$  then  $C_i^M = C_i^L + (C_i^{N_0} + \dots + C_i^{N_{k-1}})$ ,  
 $s_0^M = (s_0^L)^l$  and  
 $\delta^M(m, s^l) = (s_0^{N_i})^{ir}$  if  $\delta^L(m, s) = f_i$ ,  $\delta^M(m, s^l) = (\delta^L(m, s))^l$  otherwise,  
 $\delta^M(m, s^{ir}) = f_j$  if  $\delta^{N_i}(m, s) = f_j$ ,  $\delta^M(m, s^{ir}) = (\delta^{N_i}(m, s))^{ir}$  otherwise.
- If  $M = \mathbf{while0} N$  then  $C_i^M = C_i^N$  for each  $i$ ,  $s_0^M = s_0^N$ ,  
 $\delta^M(m, s) = s_0^N$  if  $\delta^N(m, s) = f_0$  and  $\delta^M(m, s) = \delta^N(m, s)$ , otherwise.
- If  $M = \mathbf{new} \lambda x_j : \mathbf{var}[n].N$ , then  $C_0^M = (C_0^N \cup C_i^N) \times [n]$  and  $C_k^M = C_k^N \times [n]$   
for  $k \notin \{0, j\}$ ,  $s_0^M = \langle s_0^L, 0 \rangle$  and  
 $\delta^M(\varepsilon, \langle s, i \rangle) = \langle s', i \rangle$  if  $s \in C_j^N$  and  $\delta^N(i^l, s) = s'$ ,  
 $\delta^M(\varepsilon, \langle s, i \rangle) = \langle s', j \rangle$  if  $s \in C_j^N$  and  $\delta(0^{ir}, s) = s'$ ,  
 $\delta^M(m, \langle s, i \rangle) = \langle \delta^N(m, s), i \rangle$ , otherwise.

**Corollary 2.** *Observational equivalence in basic SCI is decidable at first-order.*

At higher types, denotations no longer consist of tuples of words and so we cannot decide observational equivalence between terms at these types simply by constructing multitape automata which recognize their denotations. However, we will show that we can associate a first-order term (and hence a multitape automaton) to each higher-order term in a way which reflects observational equivalence, which is therefore decidable at all types.

**Definition 1.** *A definable monomorphism between types  $S$  and  $T$  is a term  $\mathbf{mono} : S \rightarrow T$  such that  $\llbracket \mathbf{mono} \rrbracket \cdot e = \llbracket \mathbf{mono} \rrbracket \cdot e'$  implies  $e = e'$ .*

By full abstraction, for any  $M, N : S$  we have  $M \simeq N$  iff  $\llbracket M \rrbracket = \llbracket N \rrbracket$  iff  $\llbracket \mathbf{mono} \rrbracket \cdot \llbracket M \rrbracket = \llbracket \mathbf{mono} \rrbracket \cdot \llbracket N \rrbracket$  iff  $\mathbf{mono} N \simeq \mathbf{mono} M$ . Thus we can prove that equivalence is

decidable for terms of type  $S$  by showing that there is a definable monomorphism from  $S$  to some first-order type.

First, we observe that we may restrict attention to *functional* types — i.e. types which do not contain any instances of the product construction. A *definable retraction* between types  $S$  and  $T$  is a pair of terms  $\mathbf{in} : S \rightarrow T$  and  $\mathbf{out} : T \rightarrow S$  such that  $\llbracket \lambda x. \mathbf{in} (\mathbf{out} x) \rrbracket = \llbracket \lambda x. x \rrbracket$ . Clearly,  $\mathbf{in}$  is a monomorphism. We note also that if there is a definable retraction from  $R$  to  $R'$  and from  $S$  to  $S'$  then there are definable retractions from  $S \rightarrow R$  to  $S' \rightarrow R'$  and from  $S \& R$  to  $S' \& R'$ .

**Proposition 4.** *Given functional types  $S, T$  there is a functional type  $\text{prod}(S, T)$  (of order  $\max\{o(S), o(T), 1\}$ ) such that  $S \& T$  is a definable retract of  $\text{prod}(S, T)$ .*

*Proof.* is by induction on  $o(S) + o(T)$ . For the base case, we have  $S = \underline{m}$ ,  $T = \underline{n}$  for some  $m, n$ . We define  $\text{prod}(S, T) = \underline{2} \rightarrow \underline{\max\{m, n\}}$ ,  $\mathbf{in} = \lambda x. \lambda y. \mathbf{case} \langle y, x \rangle$  and  $\mathbf{out} = \lambda x. \langle x \ 0, x \ 1 \rangle$ . For the inductive case, suppose  $T = U \rightarrow V$ . Then  $S \& T \trianglelefteq (U \rightarrow S) \& (U \rightarrow V) \cong U \rightarrow (S \& V) \trianglelefteq U \rightarrow \text{prod}(S, V) =_{df} \text{prod}(S, T)$ .

**Corollary 3.** *For every type  $T$  there exists a functional type  $\bar{T}$  of order at most  $o(T) + 1$  and a definable retraction from  $T$  to  $\bar{T}$ .*

We now define monomorphisms from higher-order to lower-order functional types. They are based on the observation that we may uniquely represent a sequence of tuples of sequences  $\langle \mathbf{s}_{11}, \dots, \mathbf{s}_{1n} \rangle \dots \langle \mathbf{s}_{k1}, \dots, \mathbf{s}_{kn} \rangle$  as a tuple of sequences  $\langle \mathbf{s}_{11} @ \dots @ \mathbf{s}_{k1}, \dots, \mathbf{s}_{1n} @ \dots @ \mathbf{s}_{kn} \rangle$ , where  $@$  is a symbol not occurring in any of the  $\mathbf{s}_{ij}$ .

**Lemma 1.** *For any types  $S, T$ , there is a definable monomorphism from  $(\underline{n} \rightarrow S) \rightarrow T$  to  $\underline{n+1} \rightarrow S \rightarrow T$ .*

*Proof.* Assuming a term  $\mathbf{eq} : \underline{n} \& \underline{n} \rightarrow \underline{2}$  such that  $\llbracket \mathbf{eq} \langle \underline{n}, \underline{n} \rangle \rrbracket = \{0\}$  and  $\llbracket \mathbf{eq} \langle \underline{n}, \underline{m} \rangle \rrbracket = \perp$  if  $n \neq m$ , we define  $\mathbf{mono1} : ((\underline{n} \rightarrow S) \rightarrow T) \rightarrow (\underline{n+1}) \rightarrow S \rightarrow T$ :

$$\lambda f. \lambda x. \lambda y. f (\lambda z. (\mathbf{while0} (\mathbf{case} \langle x, \langle \langle \mathbf{eq} \langle i, z \rangle \mid i < n \rangle, 1 \rangle \rangle)); y)$$

The denotation of  $\mathbf{mono1}$  relates each element  $\langle \langle \mathbf{j}_1, s_1 \rangle \dots \langle \mathbf{j}_m, s_m \rangle, t \rangle$  to the unique element  $\langle \mathbf{j}_1 n \dots n \mathbf{j}_m n, \langle s_1 \dots s_n, t \rangle \rangle$ . Clearly this is a 1-1 relation, and therefore a monomorphism.

**Lemma 2.** *For any type  $((S \rightarrow T) \rightarrow U) \rightarrow V$  there is a definable monomorphism into  $(S \& \underline{1}) \rightarrow (T \rightarrow U) \rightarrow V$ .*

*Proof.* We define  $\mathbf{mono2} : (((S \rightarrow T) \rightarrow U) \rightarrow V) \rightarrow (S \& \underline{1}) \rightarrow (T \rightarrow U) \rightarrow V =$

$$\lambda f. \lambda x. \lambda g. f \lambda y. (g (\pi_2(x); (y \pi_1(x))))$$

The denotation of  $\mathbf{mono2}$  relates each element

$$\langle \langle \mathbf{s}_{11}, t_{11} \rangle \dots \langle \mathbf{s}_{1m_1}, t_{1m_1} \rangle, u_1 \rangle \dots \langle \langle \mathbf{s}_{n1}, t_{n1} \rangle \dots \langle \mathbf{s}_{nm_n}, t_{nm_n} \rangle, u_n \rangle, v \rangle$$

to the unique element

$$\langle 0^r \mathbf{s}_{11}^l \dots 0^r \mathbf{s}_{1m_1}^l \dots 0^r \mathbf{s}_{n1}^l \dots 0^r \mathbf{s}_{nm_n}^l, \langle \langle t_{11} \dots t_{1m_1}, u_1 \rangle \dots \langle t_{n1} \dots t_{nm_n}, u_n \rangle, v \rangle \rangle$$

**Proposition 5.** *For any functional type  $R$  of order  $n + 2$  there is a functional type  $\widehat{R}$  of order  $n + 1$  and a definable monomorphism from  $R$  to  $\widehat{R}$ .*

*Proof.* By induction on the size of  $R$ . For the induction step, if  $R$  is of order 2 then it is isomorphic to a type of the form  $(\underline{n} \rightarrow S) \rightarrow T$ . By Lemma 1 there is a definable isomorphism from  $R$  to  $\widehat{\underline{n+1} \rightarrow S \rightarrow T}$  and hence — using the induction hypothesis — to  $\widehat{\underline{n+1} \rightarrow \widehat{S \rightarrow T}}$ .

If  $o(R) > 2$  then  $R$  is isomorphic to a type of the form  $((S \rightarrow T) \rightarrow U) \rightarrow V$ . By Lemma 2 there is a definable monomorphism from  $R$  to  $\widehat{S \& \underline{1} \rightarrow (T \rightarrow U) \rightarrow V}$  and hence — using the induction hypothesis — to  $\widehat{S \& \underline{1} \rightarrow (T \rightarrow U) \rightarrow V}$ .

**Theorem 3.** *Observational equivalence in finitary basic SCI is decidable.*

*Proof.* We use Proposition 5 to show there is a definable monomorphism from each type of SCI into a first-order type.

## 4 Observably Sequential SCI

We will now show that observational approximation is decidable in an “observably sequential” version of SCI containing a simple form of non-local control in the form of backwards jumps to labelled program points.<sup>2</sup> We extend the syntax of SCI with the constant `label` :  $(\underline{0} \rightarrow \underline{0}) \rightarrow \underline{1}$ , and the operational semantics with the evaluation contexts `label`  $E[\_]$  and `label`  $\lambda k.E[\_]$  and the reduction:

$$E[\text{label } \lambda k.E'_k[k]], \mathcal{E} \longrightarrow E[0], \mathcal{E}$$

(where we write  $E_k[\_]$  for an evaluation context which does not bind  $k$ , and assume that all substitutions are capture-avoiding).

We may use `label` to distinguish SCI-equivalent terms such as  $x; y$  and  $y; x$ . More generally, we can express Cartwright and Felleisen’s `catch` operators [2]. For each  $n$ , `catch` $_n$  :  $(T_0 \rightarrow \dots \rightarrow T_{n-1} \rightarrow \underline{m}) \rightarrow \underline{m+n}$  returns  $i$  if its argument is strict in its  $i$ th argument, or  $n+j$  if it is non-strict with value  $j$ . We define `catch` $_0 = \lambda x.x$ , and `catch` $_{n+1} = \lambda f.\text{new } \lambda x.(\text{label } \lambda k.(x := ((\text{catch}_n(f(\text{abort } k))) + 1); k); !x)$ .

We will now give a fully abstract games model of observably sequential SCI, and show that denotations may be represented as regular languages. It is similar to Abramsky and McCusker’s semantics of Idealized Algol [1], based on Hyland-Ong dialogue games [6]. Since only one thread of computation relating to each argument may be “open” at a time in SCI programs, we may omit explicit justification pointers from our model. As in McCusker and Wall’s semantics of SCI [12] (and the author’s semantics of linearly used continuations [8]) we add an equivalence relation  $\sim$  “interference” to the notion of HO-arena in order to indicate which moves may not occur in the same thread. We allow the interpretation of backwards jumps by abandoning the notion of questions and answers, and hence the “bracketing condition”.

<sup>2</sup> This breaks the equivalence between parallel and sequential composition.

A SCI arena is a directed acyclic graph  $(M_A, \vdash_A)$  — in the form of a set of nodes or moves  $M_A$ , and a set of directed edges (or *enabling relation*)  $\vdash_A \subseteq M_A \times M_A$  — with a labelling function  $\lambda_A : M_A \rightarrow \{O, P\}$ , partitioning the moves between Player and Opponent, and an equivalence relation  $\sim \subseteq M_A \times M_A$  such that:

- Root nodes (the *initial moves*  $M_A^I$ ) are Opponent moves, and there is no edge between two Opponent moves or two Player moves.
- If  $m \vdash n$  and  $m' \vdash n$  then  $m \sim m'$ .

For any sequence  $s$  of moves, we define a subsequence  $\text{open}(s)$  — the “stack of open moves” — containing at most one move from each  $\sim$  equivalence class.

$\text{open}(\varepsilon) = \varepsilon$ ,  
 $\text{open}(sm) = tm$ , if  $tm' \sqsubseteq \text{open}(s)$  and  $m' \sim m$ ,  
 $\text{open}(sm) = \text{open}(s)m$ , otherwise.

By definition, the stack of open moves contains at most one enabler for each move — which we may designate its *justifier* — and so we may unambiguously define the *view*  $\ulcorner \text{open}(s) \urcorner$  as follows:<sup>3</sup>

$\ulcorner \varepsilon \urcorner = \varepsilon$  and  $\ulcorner m \urcorner = m$   
 $\ulcorner smtn \urcorner = \ulcorner s \urcorner mn$  if  $m \vdash n$ .

The set  $L_A$  of *legal* sequences of the arena  $A$  consists of finite alternating sequences of moves of  $A$  containing at most one initial move (well-openedness), in which every non-initial move is preceded by an enabling move, which occurs in the view of the stack of open moves (visibility), and which satisfy the *non-interference condition*. Essentially this requires that two potentially interfering moves cannot be “called” from non-interfering parts of the game.

**Definition 2.** A sequence  $t$  satisfies the non-interference condition if for any  $sb, s'b' \sqsubseteq t$  and moves  $a, a'$  occurring (respectively) in  $\ulcorner \text{open}(s) \urcorner$  and  $\ulcorner \text{open}(s') \urcorner$  and having the same justifier: if  $b \sim b'$  and  $\lambda(a) = \lambda(b)$  then  $a \sim a'$ .

We define a **BCK**-algebra with products in which the objects are arenas, with product and function-space given by the following constructions:

- $A \& B = (M_A + M_B, [\lambda_A, \lambda_B], \vdash_A + \vdash_B, (\sim_A + \sim_B) \cup ((M_A^I)^l \times (M_B^I)^r) \cup ((M_B^I)^l \times (M_A^I)^r))$
- $A \rightarrow B = (M_A + M_B, [\lambda_A, \lambda_B], (\vdash_A + \vdash_B) \cup ((M_A^I)^l \times (M_B^I)^r), \sim_A + \sim_B)$ .

The elements of  $A$  are the deterministic strategies on  $A$  (non-empty and even-branching subsets of  $L_A$ ). Application of  $\sigma : A \rightarrow B$  to  $\tau : A$  is defined  $\sigma \cdot \tau = \{s \upharpoonright B \mid s \in \sigma \wedge s \upharpoonright A \in \tau^*\}$ . The combinators **B**, **C**, **T**, **P** and  $\pi_i$  are interpreted as *copycat strategies*.

**Proposition 6.** The **BCK**-algebra of SCI-arenas and strategies is well-defined.

<sup>3</sup> We do not distinguish Player and Opponent views — the view of any sequence is that of the participant about to move

*Proof.* The key point is to show that application is well-defined — that the set of legal sequences is *compositional*: i.e. if  $s \in L_{A \rightarrow B}$  and  $s \upharpoonright A \in (L_A)^*$  then  $s \upharpoonright B \in L_B$ . The proof that  $s \upharpoonright B \in L_B$  satisfies the non-interference condition is based on techniques used to prove compositionality of innocence in HO-games.

We obtain a model of SCI by setting the denotation of the ground type  $\underline{n}$  to be the arena with a single initial move enabling  $n$  different Player moves. Thus  $\underline{0} \rightarrow \underline{0}$  is isomorphic to  $\underline{1}$ , and we define the denotation of `label` to be this isomorphism. The remaining constants of SCI are interpreted as in the game semantics of Idealized Algol [1]. Proof of soundness and adequacy uses standard techniques and follows those of similar results for games models.

**Proposition 7.** *For any closed term  $M : \underline{1}$ ,  $M \Downarrow$  if and only if  $\llbracket M \rrbracket \neq \perp$ .*

We now observe that denotational equivalence and approximation are decidable because the strategy denoting each term is a *regular language*. The key point is that the set of legal sequences over each SCI type-object is itself regular.

**Lemma 3.** *For any finitary SCI type  $T$ , the set  $L_{\llbracket T \rrbracket}$  is regular.*

*Proof.* We define a FSA in which each state is a pair consisting of a non-repetitive (and hence bounded) legal sequence  $t$  and a function from  $M_A$  to sets of non-repetitive sequences over  $M_A$ . The initial state is the pair of the empty sequence and the constantly  $\emptyset$  function, and the transition function is defined so that having read the legal sequence of moves  $r$ , the FSA is in the state  $\langle \text{open}(r), f \rangle$ , where  $f(m)$  is the set of views of prefixes of  $r$  which contain an occurrence of  $m$  which is in  $\text{open}(r)$ : this information is sufficient to determine the legality of the next move to be read.

Hence the copycat strategies interpreting the **B, C, K, P** and  $\pi$  combinators are regular. Following [3], we show that the application of one regular strategy to another is regular, and that the interpretations of `case`, `while0`, `label` and `new` are regular.

**Proposition 8.** *The denotation of each term of observably sequential SCI is regular.*

#### 4.1 Full Abstraction

We will now prove that our interpretation of observably sequential SCI is (inequationally) fully abstract. This is the case despite the fact that there are finitary strategies in the model which are not the denotations of terms (and may in fact exhibit interfering behaviour). For example, the strategy on the arena  $(\underline{1} \rightarrow \underline{1} \rightarrow \underline{1}) \rightarrow \underline{2}$  — corresponding to the Idealized Algol term  $\lambda f.\text{new } \lambda x.((f(x := 1))x := 0); !x$  — which runs its argument once, and returns one if the last argument tested by  $f$  was the leftmost one and 0 otherwise.

It is sufficient to establish full abstraction for functional types, since every type is a definable retract of such a type (as in Proposition 4). We further simplify the set of types for which we need to prove full abstraction to the *zero* types: functional types generated from the atomic type  $\underline{0}$ .

**Proposition 9.** *Every functional type is definably isomorphic to a zero type.*

*Proof.* Define the type  $\mathbb{Q}_k$  for each  $k \geq 0$  by  $\mathbb{Q}_0 = \mathbb{Q}$  and  $\mathbb{Q}_{k+1} = \mathbb{Q} \rightarrow \mathbb{Q}_k$ . The arenas denoting  $\mathbb{Q}_k$  and  $\underline{k}$  are clearly isomorphic; these isomorphisms are definable as  $\text{catch}_k : \mathbb{Q}_k \rightarrow \underline{k}$  and  $\lambda x. \lambda y. \text{case } \langle x, \langle y_0, \dots, y_{k-1} \rangle \rangle : \underline{k} \rightarrow \mathbb{Q}_k$ .

We now show that sufficient “observations” are definable to prove full abstraction. For any sequence  $s$ , let  $|s|$  be the *multiset* of moves occurring in  $s$ .

**Lemma 4.** *Let  $T$  be a functional type. For any sequence  $s \in L[[T]]$ , there exists a term  $M(s) : T$  such that  $s \in [[M(s)]]$  and if  $t \in [[M(s)]]$  then  $|t| \subseteq |s|$ .*

*Proof.* By Corollary 9 we may assume  $T$  is a zero type. We define  $M(s)$  by induction on the length of  $s$ . For the induction case, suppose  $s = m_1 m_2 r$  for some  $r$ . First we suppose that the move  $m_2$  does not occur in  $r$ , and define  $M(s)$  by induction on the *arity* of  $T$ .

- For the base case, suppose  $T = (\mathbf{S} \rightarrow \mathbb{Q}) \rightarrow \mathbb{Q}$ . For each  $i$ ,  $s \upharpoonright [[S_i]] = t_{i1} t_{i2} \dots t_{in_i}$ , where each  $t_{ij}$  is a well-opened legal sequence on  $[[S_i]]$  which is shorter than  $s$ . Thus we may define  $x_i : \text{var}[n+2] \vdash N_i : S_i = x_i := x_i + 1; \text{case } \langle !x_i, \langle \Omega, M(t_1), \dots, M(t_n), \Omega \rangle \rangle$  for each  $i$ , and

$$M(s) = \lambda f. \text{new } \lambda x_1 \dots \text{new } \lambda x_n. (f N_1 \dots N_n)$$

- If  $T$  has arity greater than one, it is (definably) isomorphic to a type of the form  $\mathbf{R} \rightarrow \mathbf{S} \rightarrow (\mathbf{U} \rightarrow \mathbb{Q}) \rightarrow \mathbb{Q}$ , where  $m_2$  is the initial move in  $[[\mathbf{U} \rightarrow \mathbb{Q}]]$ , so we assume that it has this form. We may show that if  $tb \sqsubseteq s$ , where  $b$  is the initial move in  $S$ , then there exists a unique  $i$  such that the initial move in  $U_i$  occurs in  $\ulcorner \text{open}(tb) \urcorner$ . By the non-interference condition, for any two occurrences of  $b$  in  $s$ , this index  $i$  is the same. Hence we may define a legal sequence  $\hat{s}$  on the arena  $[[\mathbf{R} \rightarrow (\mathbf{V} \rightarrow \mathbb{Q}) \rightarrow \mathbb{Q}]]$  — where  $V_i = S \rightarrow U_i$ , and  $V_j = U_i$  if  $j \neq i$  — by relabelling each move from  $S$  in  $s$  as the corresponding move in  $V_i$ . We may then define

$$M(s) = \lambda \mathbf{x}. \lambda y. \lambda z. (M(\hat{s}) \mathbf{x}) \lambda \mathbf{a}. z a_1 \dots a_{i-1} (a_i y) a_{i+1} \dots a_n$$

If  $m_2$  does occur in  $r$  — i.e.  $s = m_1 m_2 r_1 m_2 r_2$ , where  $m_2$  does not occur in  $r_1$ , then the sequence  $m_1 m_2 r_1 n$  in  $\mathbb{Q} \rightarrow T$  (where  $n$  is the move from  $\mathbb{Q}$ ) is no longer than  $s$ , whilst  $m_1 m_2 r_2$  is shorter than  $s$ . Thus we may define

$$M(s) = \lambda \mathbf{x}. (\text{label } \lambda k. (M(m_1 m_2 r_1 n) k \mathbf{x})); (M(m_1 m_2 r_2) \mathbf{x})$$

**Proposition 10.** *For any closed  $M, N : T$ ,  $[[M]] \subseteq [[N]]$  iff  $M \lesssim N$ .*

*Proof.* From left to right (soundness) this is standard. For the converse, suppose  $[[M]] \not\subseteq [[N]]$  and let  $smn$  be a minimal length sequence in  $[[M]] - [[N]]$ . By prefixing an initial move  $q \in M_1$  and postfixing its “answer”  $a$ , we obtain a legal sequence  $qsmna$  in  $[[T \rightarrow \mathbb{1}]]$ . By Lemma 4, there exists  $L : T \rightarrow \mathbb{1}$  such that  $qsmna \in [[L]]$  (hence  $[[LM]] \neq \perp$ ) and if  $t \in [[L]]$  then  $|t| \subseteq |qsmna|$ .

Suppose  $\llbracket LN \rrbracket \neq \perp$ . Then there exists  $qta \in \llbracket L \rrbracket$  such that  $t \in \llbracket N \rrbracket$ . By minimality of  $smn$ ,  $s \in \llbracket N \rrbracket$ , and hence by determinacy of  $\llbracket L \rrbracket$  and  $\llbracket N \rrbracket$ ,  $qsm \sqsubseteq qt$ , and so  $|qsm| \subseteq |qta| \subseteq |qsmna|$  by definition of  $\llbracket L \rrbracket$ . But then  $t = smn$ , contradicting the assumption that  $smn \notin \llbracket N \rrbracket$ .

Hence  $\llbracket LM \rrbracket \neq \perp$  and  $\llbracket LN \rrbracket = \perp$ , and  $LM \Downarrow, LN \not\Downarrow$  as required

**Theorem 4.** *Observational equivalence and approximation are decidable in finitary observably sequential SCI.*

## References

1. S. Abramsky and G. McCusker. Linearity, Sharing and State: a fully abstract game semantics for Idealized Algol with active expressions. In P.W. O’Hearn and R. Tennent, editors, *Algol-like languages*. Birkhauser, 1997.
2. R. Cartwright and M. Felleisen. Observable sequentiality and full abstraction. In *Proceedings of POPL ’92*, 1992.
3. D. Ghica and G. McCusker. The regular language semantics of second-order Idealised Algol. *Theoretical Computer Science (To appear)*, 2003.
4. T. Harju and J. Karhumäki. The equivalence problem of multitape finite automata. *Theoretical Computer Science*, 78:347–355, 1991.
5. J. R. Hindley. *Basic Simple Type Theory*. Cambridge University Press, 1997.
6. J. M. E. Hyland and C.-H. L. Ong. On full abstraction for PCF: I, II and III. *Information and Computation*, 163:285–408, 2000.
7. E. Kinber. The inclusion problem for some classes of deterministic multitape automata. *Theoretical Computer Science*, 26:62–73, 1983.
8. J. Laird. Game semantics and Linear CPS interpretation. *Theoretical Computer Science*, 333:199–224, 2005.
9. R. Loader. Finitary PCF is undecidable. *Annals of Pure and Applied Logic*, 2000.
10. G. McCusker. A fully abstract relational model of Syntactic Control of Interference. In *Proceedings of Computer Science Logic ’02*, number 2471 in LNCS. Springer, 2002.
11. G. McCusker. On the semantics of the bad-variable constructor in Algol-like languages. In *Proceedings of MFPS XIX, ENTCS*, 2003. To appear.
12. G. McCusker and M. Wall. Categorical and game semantics for SCIR. In the proceedings of Games for Logics and Programming Languages, 2004.
13. A. Murawski. On program equivalence in languages with ground-type references. In *Proceedings of LICS ’03*. IEEE Press, 2003.
14. A. Murawski and I. Walukiewicz. Third-order Idealized Algol with iteration is decidable. In *Proceedings of FoSSACS ’05*, number 3411 in LNCS, pages 202–218. Springer, 2005.
15. P. W. O’Hearn, A.J. Power, M. Takeyama and R.D. Tennent. Syntactic control of interference revisited. *Theoretical Computer Science*, 228(1-2):211–252, 1999.
16. M. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:114–125, 1959.
17. U. S. Reddy. Global state considered unnecessary: Object-based semantics for interference-free imperative programs. *Lisp and Symbolic Computation*, 9(1), 1996.
18. J. Reynolds. Syntactic Control of Interference. In *Conf. Record 5<sup>th</sup> ACM Symposium on Principles of Programming Languages*, pages 39–46, 1978.
19. J. Reynolds. The essence of Algol. In *Algorithmic Languages*, pages 345–372. North Holland, 1981.