An approach to deciding the observational equivalence of Algol-like languages

C.-H.L. Ong
Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford OX1 3QD, UK
Available online 3 August 2004

Abstract

We prove that the observational equivalence of third-order finitary (i.e. recursion-free) Idealized Algol (IA) is decidable using Game Semantics. By modelling the state explicitly in our games, we show that the denotation of a term M of this fragment of IA is a compactly innocent strategy-with-state, i.e. the strategy is generated by a finite view function f_M. Given any such f_M, we construct a real-time deterministic pushdown automaton (DPDA) that recognizes the complete plays of the knowing-strategy denotation of M. Since such plays characterize observational equivalence, and there is an algorithm for deciding whether any two PDAs recognize the same language, we obtain a procedure for deciding the observational equivalence of third-order finitary IA. Restricted to second-order terms, the DPDA representation cuts down to a deterministic finite automaton; thus our approach gives a new proof of Ghica and McCusker’s regular-expression characterization for this fragment. Our algorithmic representation of program meanings, which is compositional, provides a foundation for model-checking a wide range of behavioural properties of IA and other cognate programming languages. Another result concerns second-order IA with full recursion: we show that observational equivalence for this fragment is undecidable.

© 2004 Elsevier B.V. All rights reserved.

Keywords: Algorithmic game semantics; Algol-like languages; Automata theory; Software model-checking

1. Introduction

1.1. Algorithmic Game Semantics

Game Semantics has emerged as a powerful paradigm for giving semantics to a variety of programming languages and logical systems. It has been used to construct the
first syntax-independent fully abstract models for a spectrum of programming languages ranging from purely functional languages to languages with non-functional features such as control operators and locally scoped references \([2–4,13,16]\). In the game reading, types (specifications) are interpreted as games, and programs (implementations) as strategies on games. Game Semantics captures the quantitative and algorithmic aspects of a computation typical of Operational Semantics, while admitting compositional methods in the style of Denotational Semantics: it provides a very concrete way of building fully abstract models.

This paper concerns a recent development of Game Semantics in a new, algorithmic direction, with a view to applications in computer-assisted verification and program analysis. The important first steps have been taken by Ghica and McCusker \([9]\); they show that the second-order finitary (i.e. recursion-free) fragment of the fully abstract game semantics of Reynolds’ language, Idealized Algol (IA), can be represented in a remarkably simple form by regular expressions. This yields a procedure for deciding the observational equivalence for this fragment.

The promise of this approach is to transfer the methods of Model Checking (see e.g. \([5]\)) based on automata-theoretic representations, which has been so successful in the analysis of hardware designs and communications and security protocols, to the much more structured setting of programming languages, in which data types and control flow are important.

1.2. Overview

A notable feature of Abramsky and McCusker’s fully abstract knowing (or history-sensitive) strategy game semantics \([4]\) for IA is the implicit nature of its model of state. Knowing strategies that denote finitary IA terms are in general infinite sets of justified sequences (i.e. sequences of moves, each of which is equipped with a justification pointer to an earlier move) of a certain kind. For games of up to second order, such pointers can safely be ignored (because they are uniquely reconstructible), so justified sequences may simply be represented as words over an alphabet of moves. Recently Ghica and McCusker \([9]\) have shown that knowing strategies that denote terms of second-order finitary IA can be represented, compositionally, as regular expressions. Can the approach be extended to higher orders? We show that using a scheme based on view offsets, justified sequences can be encoded as words over an appropriate alphabet. We then show that, in general, complete plays (i.e. plays in which every question is answered) of knowing strategies that denote terms of the third-order fragment of IA are not regular (Lemma 30).

What automata-theoretic formalism then is needed to characterize (the game semantics of) third-order finitary IA? Our approach to the question is via a more intensional version of game semantics that models state explicitly: we attach a state (i.e. a finite function from locations to data values) to each move of the play. Intuitively the locations that are defined at a move correspond to the assignable variables that have been declared (but not yet deallocated) thus far in the history of the computation. As these variables are local (to the block in which they are allocated), their contents can only be changed by the assignment commands that are in scope, i.e. within the same allocating block in the program being modelled (P’s perspective), and not directly by any assignment command of the program context (O’s perspective). This translates to a basic principle governing state: only P can
change state, either by updating the contents at locations introduced by earlier P-moves which are currently P-visible (i.e. appearing in the P-view of the history of play), or by introducing new locations. The key point is that, for games of up to third order, we may take this innocent view of states further by considering as relevant only those locations that have been introduced by earlier P-moves which are currently P-visible. This has the important effect of constraining the growth of states (attached to moves in a play) as the game unfolds.

We show that $[\Gamma \vdash M : A]$, the strategy-with-state that denotes a third-order IA term-in-context, is compactly innocent in the sense (of [13]) that the strategy is generated by a finite view function $f_M$. Moreover $[\Gamma \vdash M : A]$ is closely related to the knowing-strategy denotation, written $[\Gamma \vdash M : A]^\kappa$: we prove in Theorem 25 that

$$\text{erase } [\Gamma \vdash M : A] = [\Gamma \vdash M : A]^\kappa;$$

i.e. the knowing-strategy denotation may be recovered by erasing states from (each play of) the innocent strategy-with-state denotation. We then give a general construction that takes any such finite view function $f_M$ and produces a real-time (i.e. no $\epsilon$-transition) deterministic pushdown automaton (DPDA) $P_{f_M}$. The control states of $P_{f_M}$ are the P-views that define $f_M$; its input alphabet is the set of moves (without state) of the arena, augmented by certain symbols for the purpose of encoding pointers. Writing $L(P_{f_M})$ for the language recognized by $P_{f_M}$, we then prove a major technical result (Theorem 31) of the paper:

$$L(P_{f_M}) = \text{cplays } [\Gamma \vdash M : A]^\kappa.$$

In other words, the set of complete plays of the knowing-strategy semantics of third-order finitary IA is deterministic context-free. This answers the question we posed earlier in the Overview.

Further since compactly innocent strategies-with-state are effectively compositional (Theorem 18), the innocent-strategy denotation $[\Gamma \vdash M : A]$, regarded as a map $M \mapsto f_M$, is effectively computable and, hence, so is the DPDA representation $M \mapsto P_{f_M}$. Finally, thanks to a characterization of observational equivalence (written $\approx$) in terms of complete plays in [4]

$$M \approx N \iff \text{cplays } [\Gamma \vdash M : A]^\kappa = \text{cplays } [\Gamma \vdash N : A]^\kappa$$

we obtain a pleasing application of the DPDA representation:

**Theorem 1.** For any third-order finitary IA terms-in-contexts $\Gamma \vdash M : A$ and $\Gamma \vdash N : A$ that are constructed from finite base types, we have

$$M \approx N \iff L(P_{f_M}) = L(P_{f_N}).$$

Further, because there is an effective procedure for deciding whether any two DPDAs recognize the same language, $M \approx N$ is decidable. □

The DPDA Equivalence Problem

"Is there an effective procedure for deciding whether any two DPDAs recognize the same language?"
was first posed in 1966. Although the real-time case was resolved positively by Oyamaguchi et al. [29] in 1980, the general problem was only solved recently, also positively, by Sénizergues [35], who was awarded the ACM SIGACT/EATCS Gödel Prize for his accomplishment. Stirling subsequently obtained a considerably simpler proof and gave a primitive recursive decision procedure [37,38].

Restricted to finitary second-order terms, the DPDA representation cuts down to a deterministic finite automaton; thus our approach gives a new proof of Ghica and McCusker’s regular-expression characterization for this fragment. A natural question is whether the algorithmic representation can be extended to fourth and higher orders. It turns out that third order is the limit, and the best that we can do.

Another result of the paper concerns second-order IA with recursion (in which fixpoints of functions of second order are definable). We show that observational equivalence for this fragment is undecidable (Theorem 38).

1.3. Related work

Perhaps the most important result that we should mention here is Loader’s undecidability theorem in [18]. He proves that observational equivalence of third-order finitary PCF is undecidable. It is worth pointing out that observational equivalence of (active) IA does not extend that of PCF conservatively. Recently Murawski [22] has shown that observational equivalence of fourth-order IA is undecidable.

On the (Algorithmic) Game Semantics front, Ghica has extended his earlier work with McCusker to a call-by-value language with arrays [8], and to model-checking Hoare-style program correctness assertions [7]. For a tutorial introduction, we recommend [1].

1.4. Outline

The rest of the paper is organized as follows. The syntax and operational semantics of Idealized Algol are introduced in Section 2. We develop the requisite game-semantic machinery in Section 3, and give an innocent strategy-with-state interpretation of third-order IA in Section 4, which also contains a result relating the innocent denotation to the knowing-strategy semantics. In Section 5, using view offsets to encode justifications pointers, we show that the innocent denotation of a third-order finitary IA term gives rise to a DPDA which characterizes the knowing-strategy semantics of the term. As a corollary we obtain the main decidability results of the paper. In Section 6, we prove that observational equivalence of second-order IA with recursion is undecidable. We conclude the paper with a discussion of further directions in Section 7.

Readers who are familiar with the game semantics of Idealized Algol may wish to skip the next two sections and go straight to Section 4, and return to Section 3 at a later stage for the new material on game semantics.

An extended abstract of this paper has appeared in the Proceedings of LICS’02 [28].

---

1 Take the two PCF-terms to be \(\lambda x : \mathbb{B}. \text{if } x \text{ then } x \text{ else } x\) and \(\lambda x : \mathbb{B}. x\), and the separating context to be

\[\text{new } y := 0 \text{ in if } [-](y := !y + 1; \; \#) \text{ then (if } !y = 1 \text{ then } \# \text{ else } \#).\]
2. Idealized Algol

Reynolds’ *Idealized Algol* (IA for short) \[33\] is a compact language that elegantly combines state-based procedural and higher-order functional programming. The basis of IA is a simply typed call-by-name \(\lambda\)-calculus in which the standard constructs of imperative programming and block-allocated assignable variables are definable. In this section, we review the syntax and operational semantics of IA, introduce the notion of observational equivalence, and illustrate the richness of the theory (of observational equivalence) with examples.

2.1. Syntax

We let \(D\) range over *basic data sets* such as \(\mathbb{N}\) (natural numbers), \(\mathbb{B}\) (booleans), \(\mathbb{Z}\) (integers), and \(\mathbb{I}\) (the singleton set). The *base types* of IA (ranged over by \(\beta\)) are then defined by the grammar:

\[A ::= \beta | A \Rightarrow A.\]

The order of a type \(A\), written \(\text{order}(A)\), is defined as follows: for any base type \(\beta\), \(\text{order}(\beta) \equiv 0\) (note that both 0 and 1 are used in the literature); and \(\text{order}(A \Rightarrow B) \equiv \max(\text{order}(A) + 1, \text{order}(B))\).

For simplicity, in the rest of the paper (except for Examples 2 and 26) we shall work with a version of IA that is generated from a single basic data set, namely, \(\mathbb{N}\). To save writing, we shall write \(\text{exp}[\mathbb{N}]\) and \(\text{loc}[\mathbb{N}]\) simply as \(\text{exp}\) and \(\text{loc}\) respectively.

*Term candidates* of IA (ranged over by \(M, N, P\), etc.) are defined as follows:

\[M ::= x | MM | \lambda x : A.M | 0 | n\]

\[| \text{succ} | \text{pred} | \text{ifzero} M \text{ then } M \text{ else } M | Y(M)\]

\[| l | M ::= M | !M | \text{mkloc} M M\]

\[| \text{skip} | M ; M | \text{new} x ::= n \text{ in } M\]

where \(x\) ranges over a countable set of variables, \(l\) ranges over a countable set of *locations*, and \(n\) over natural numbers. The first two lines define the standard PCF constructs \([32, 34]\). We use infix \(\sim\) for variable assignment, prefix \(!\) for dereferencing, infix \(;\) for command sequencing, and write \(\text{skip}\) for the null command. Block-allocated local variables are introduced by the construct \(\text{new} x ::= n \text{ in } M\) (the local variable \(x\) is initialized to \(n\) as control enters the block; note also that \(x\) is bound by the \(\text{new}\)-construct). The *bad-location constructor* (or, more commonly, *bad-variable constructor*) \(\text{mkloc}\) takes a “read method” \(M\) and a “write method” \(N\) and creates a term \(\text{mkloc} M N\) of type \(\text{loc}\). Note that this version of IA (as opposed to Reynolds’ original definition) admits *active* expressions, i.e. expressions with side effects such as \(x : \text{loc} \vdash x ::= !x + 1 ; !x : \text{exp}\) and

\[f : \text{com} \Rightarrow \text{exp} \vdash \text{new} z ::= 0 \text{ in } (f(z ::= !z + 1) + !z) : \text{exp}.\]
Using the fixed-point operator, IA can express iterations (e.g. the while-loop \( \text{while } B \text{ do } C \) is definable as \( Y(\lambda x. \text{if } B \text{ then } C ; x \text{ else } \text{skip}) \) and recursively defined procedures.

**Valid typing judgements** of IA of the form \( \Gamma \vdash M : A \) are defined by induction over a number of rules. We shall give in Fig. 1 only the rules pertaining to the imperative constructs of IA; rules for the other constructs (which are PCF-constructs) are standard. Note that in the figure, \( \beta \) ranges over the base types \( \text{exp} \) and \( \text{com} \). Finally we define the order of a term-in-context \( x_1 : C_1, \ldots, x_n : C_n \vdash M : A \) to be

\[
\max\{\text{order}(C_1) + 1, \ldots, \text{order}(C_n) + 1, \text{order}(A)\}.
\]

2.2. Operational semantics

**Values** of IA, ranged over by \( V, V' \), etc. are values of the base types (namely, numerals, \( \text{skip} \), and locations and \( \text{mkloc} \)-terms) and \( \lambda \)-abstractions. The operational semantics of IA is given in terms of an evaluation relation of the form

\( M, S \Downarrow V, S' \)

where \( M \) and \( V \) range over closed terms and closed values respectively, and \( S \) and \( S' \) range over states which are finite functions from locations to elements of the basic data sets. The evaluation relation \( \Downarrow \) is defined by induction over the set of rules in Fig. 2.

Let \( M \) and \( N \) be terms such that for some \( \Gamma \) and \( A \), we have \( \Gamma \vdash M_i : A \) is provable for \( i = 1 \) and 2. We define \( M \triangleleft N \) (read “\( M \) observationally approximates \( N \)””) to mean: for any context \( C[-] \) such that \( C[M] \) and \( C[N] \) are programs (i.e. closed terms of type \( \text{com} \) in which there is no occurrence of any location),

\[
C[M], \emptyset \Downarrow \text{skip}, \emptyset \implies C[N], \emptyset \Downarrow \text{skip}, \emptyset.
\]

Note that quantification over all program contexts \( C[-] \) ensures that all potential side effects of \( M \) and \( N \) are taken fully into account. We call \( \triangleleft \) observational preorder, and define \( M \approx N \) (“\( M \) and \( N \) are observationally equivalent”) to mean \( M \subseteq N \) and \( N \subseteq M \).

The theory of observational equivalence is rich. We consider a few examples.

**Example 2.** The following examples illustrate features of IA which have been studied in the literature.

(i) No “snap back” [25]: This illustrates the consequences for observational equivalence of the inability of IA to “snap back” the state to some previous point in the thread of
Fig. 2. Rules defining the evaluation relation $\Downarrow$.

\[
\begin{align*}
V, S & \Downarrow V, S \\
M, S & \Downarrow \text{skip}, S' \quad N, S' & \Downarrow V, S'' \quad V = n \text{ or skip} \\
M, S & \Downarrow l, S' \quad !M, S & \Downarrow n, S' \quad l \in \text{dom}(S') \land S'(l) = n \\
M, S & \Downarrow \text{mkloc } P Q, S' \quad P, S' & \Downarrow n, S''
\end{align*}
\]

\[
\begin{align*}
!M, S & \Downarrow n, S'' \\
N, S & \Downarrow n, S' \quad M, S' & \Downarrow l, S'' \\
M := N, S & \Downarrow \text{skip}, S'[l \mapsto n] \\
N, S & \Downarrow n, S' \quad M, S' & \Downarrow \text{mkloc } P Q, S'' \quad Q n, S'' & \Downarrow \text{skip}, S''
\end{align*}
\]

\[
M := N, S \Downarrow \text{skip}, S''
\]

\[
\begin{align*}
M(Y(M)), S & \Downarrow V, S' \\
Y(M), S & \Downarrow V, S'
\end{align*}
\]

\[
\begin{align*}
M[l/x], S[l \mapsto n] & \Downarrow V, S'[l \mapsto m] \\
\text{new } x & := n \text{ in } M, S \Downarrow V, S' \quad l \notin \text{dom}(S) \cup \text{dom}(S').
\end{align*}
\]

\[
\text{computation:}
\begin{align*}
p & : \text{com} \Rightarrow \text{com} \\
\vdash \text{new } x & := 0 \text{ in } p(x := 1) ; \text{if } !x = 1 \text{ then } \Omega \text{ else skip} \approx p(\Omega).
\end{align*}
\]

(ii) **Parametricity** [26]: Note that the local variables $x$ and $y$ in the following have types \text{loc}[$\mathbb{Z}$] and \text{loc}[$\mathbb{B}$] respectively:

\[
\begin{align*}
p & : \text{com} \Rightarrow \text{exp}[$\mathbb{B}$] \Rightarrow \text{com} \\
\vdash \text{new } x & := 1 \text{ in } p(x := -1x) ; (!x > 0) \\
& \approx \text{new } y := \# \text{ in } p(y := \text{not } y) ; (!y).
\end{align*}
\]

(iii) Consider the following third-order terms [31]:

\[
\begin{align*}
\Xi_1 & = \lambda F. \text{new } x := 1 \text{ in } F(\lambda c. x := 0 ; c ; !x) \\
\Xi_2 & = \lambda F. \text{new } x := 1 \text{ in } F(\lambda c. x := !x + 1 ; c ; x := 1x - 2 ; !x) \\
\Xi_3 & = \lambda F. F(\lambda c. c ; 0)
\end{align*}
\]

of type ((com \Rightarrow exp) \Rightarrow exp) \Rightarrow exp. We have $\Xi_1 \not\approx \Xi_2$ and $\Xi_1 \approx \Xi_3$. The reader may wish to verify $\Xi_1 \not\approx \Xi_2$ by applying $\Xi_1$ and $\Xi_2$ respectively to

$\lambda h. h(\text{ifzero}(h\text{skip}) \text{ then skip else } \Omega)$. 

Remark 3. In [20] McCusker has adapted the knowing-strategy semantics of IA to give a characterization of a version of IA without the bad-location constructor \texttt{mkloc}. Using the model he shows that adding \texttt{mkloc} is conservative for observational equivalence but not for observational preorder.

3. Game semantics

This section develops the game semantics that underpins the main decidability results. After a quick review of the basic definitions, we introduce strategies-with-state and a protocol for updating states that correctly models the scoping of locations (or assignable variables) for up to order three. We then consider the strategies of this fragment that are \textit{compactly innocent}, and prove that they are effectively composable.

3.1. Basic definitions

Definition 4. An \textit{arena} is a triple \(A = \langle M_A, \lambda_A, \vdash_A \rangle\) where \(M_A\) is a set of moves;

\[\lambda_A : M_A \rightarrow \{PQ, PA, OQ, OA\}\]

is a labelling function which, for a given move, indicates which of Proponent (P) and Opponent (O) may make the move and whether it is a question (Q) or an answer (A); and \(\vdash_A \subseteq (M_A \cup \{\ast\}) \times M_A\), where \(\ast\) is a dummy move (which is not in \(M_A\)), is called the \textit{justification relation} (we read \(m_1 \vdash_A m_2\) as "\(m_1\) justifies \(m_2\)") satisfying the following axioms: in (3) and (4), we let \(m\) and \(m'\) range over \(M_A\).

1. For each \(m \in M_A\) there is a unique \(x \in M_A \cup \{\ast\}\) such that \(x \vdash_A m\); in case \(\ast \vdash_A m\), we call \(m\) an \textit{initial move}.
2. Every initial move is an O-question.
3. If \(m \vdash_A m'\) then \(m\) and \(m'\) are moves by different players.
4. If \(m \vdash_A m'\) then \(m\) is a question ("Only questions may justify moves.").
5. \(\vdash_A\) on \((M_A \times M_A)\) is well-founded.

It is useful to think of the justification relation \(\vdash_A\) (restricted to \(M_A \times M_A\)) as defining the edge-set of a vertex-labelled directed graph whose vertex-set is \(M_A\). It follows from the definition that the graph so defined, which we shall refer to as the \textit{arena graph} of \(A\), is a forest (of trees). We call an arena \textit{finite} just in the case where it has finitely many moves.

Remark 5. Though our arenas are presented in the style of McCusker [19], they are exactly those introduced in [13]. In the arenas defined in [4], answer-moves may also justify moves. This is a level of generality that we do not need. Our arenas are adequate for the construction of the fully abstract knowing-strategy semantics of Idealized Algol, and in particular for establishing Theorem 23.

The simplest arena is the empty arena \(1 = \langle \emptyset, \emptyset, \emptyset \rangle\). Let \(A\) and \(B\) be arenas. The \textit{product arena} \(A \times B\) is just the disjoint union of the respective arena graphs of \(A\) and \(B\).
Formally we have
\[
M_{A \times B} = M_A + M_B \\
\lambda_{A \times B} = [\lambda_A, \lambda_B] \\
* \vdash_{A \times B} m \iff * \vdash_A m \lor * \vdash_B m \\
\vdash_{A \times B} n \iff \vdash_A n \lor \vdash_B n.
\]

Given an arena \( A \), we write \( \overline{A} \) for the graph that is obtained from the arena graph of \( A \) by inverting the P/O-label at each vertex. As an operation on arena graphs, the function space \( \text{arena} A \Rightarrow B \) is obtained from the arena graph of \( B \) by grafting a copy of \( \overline{A} \) just under each initial move of \( B \). Formally, writing \( M_{\text{Init}} B \) for the set of initial moves of \( B \), we have
\[
M_{A \Rightarrow B} = (M_A \times M_{\text{Init}} B) + M_B \\
\lambda_{A \Rightarrow B} = [\pi_1; \lambda_A, \lambda_B]
\]
(where \( P, Q, O, A \) are defined to be \( OQ, OA, PQ, PA \) respectively) and \( \vdash_{A \Rightarrow B} \subseteq (M_A \times M_{\text{Init}} B + M_B + \{\ast\}) \times (M_A \times M_{\text{Init}} B + M_B) \) is defined by:
\[
* \vdash_{A \Rightarrow B} b \iff * \vdash_B b \\
b \vdash_{A \Rightarrow B} (a, b') \iff b = b' \land * \vdash_A a \\
(a, b) \vdash_{A \Rightarrow B} (a', b') \iff b = b' \land a \vdash_A a' \\
b \vdash_{A \Rightarrow B} b' \iff b \vdash_B b'.
\]

We use square and round parentheses in bold type as meta-variables for moves as follows:

O-question P-answer P-question O-answer
\[
[ ] ( )
\]

We define the type-theoretic order (or simply order) of a question-move \( q \), written \( \text{order}(q) \), to be 0 if \( q \) is initial; otherwise if \( q \vdash_A q' \) then \( \text{order}(q') = 1 + \text{order}(q) \). The order of an answer-move is the order of the (unique) question-move that justifies it. In the following we shall only be concerned with arenas that have finitely many questions. The order of such an arena is defined to be the order of the question that has the highest type-theoretic order.

A justified sequence over an arena \( A \) is a finite sequence of alternating moves such that, except the first move which is initial, every move \( m \) has a justification pointer (or simply pointer) to some earlier move \( m_0 \) such that \( m_0 \vdash_A m \); we say that \( m \) is explicitly justified by \( m_0 \), or \( m_0 \) explicitly justifies \( m \). A question in a justified sequence \( s \) is said to be pending just in the case where no answer in \( s \) is explicitly justified by it. Recall the definition of the P-view \( [13] \) of a justified sequence \( s \), written \( \gamma s \gamma \):
\[
\gamma s m \gamma = \gamma s \gamma m \quad \text{if} \quad m \text{ is a P-move} \\
\gamma m \gamma = m \quad \text{if} \quad m \text{ is initial} \\
\gamma s m_0 u m \gamma = \gamma s \gamma m_0 m \quad \text{if} \quad \text{the O-move} m \text{ is explicitly justified by} \ m_0.
\]
In \( \gamma s m_0 u m \gamma \) the pointer from \( m \) to \( m_0 \) is retained, and similarly for the pointer from \( m \) in \( \gamma s m \gamma \) in the case where \( m \) is a P-move. The definition of O-view \( \langle s, l \rangle \) is obtained from the
above clauses by swapping P and O. We shall consider justified sequences that satisfy the following conditions [13]:

- **Visibility**: Every P-move (respectively non-initial O-move) is explicitly justified by some move that appears in the P-view (respectively O-view) at that point.
- **Well-Bracketing**: Every P-answer (respectively O-answer) is an answer to (i.e. explicitly justified by) the last pending O-question (respectively P-question).

### 3.2. State change conditions

We shall consider justified sequences of moves-with-state, and introduce new conditions of State Change. First some notation. For simplicity we fix \( \mathbb{N} = \{0, 1, 2, \ldots\} \) as our only basic data set. A **state** (ranged over by \( S, S_i, T, \) etc.) is a finite function from locations (ranged over by \( l, l', l_i, \) etc.) to natural numbers. A **move-with-state** of an arena \( A \) is a pair, written \( m^S \) (or just \( m \) if \( S \) is understood), where \( m \) is a move of \( A \) and \( S \) is a state; we refer to a location in \( \text{dom}(S) \) as a location that is **defined** at \( m \).

Given states \( S_0 \) and \( S_1 \), we define the state \( S_0[S_1] \) (read “\( S_0 \) updated by \( S_1 \)”) as: for each \( l \in \text{dom}(S_0) \)

\[
S_0[S_1](l) \overset{\text{def}}{=} \begin{cases} S_1(l) & \text{if } l \in \text{dom}(S_1) \\ S_0(l) & \text{otherwise.} \end{cases}
\]

Note that \( S[S] = S \), for any state \( S \).

Given an arena, we shall consider justified sequences of moves-with-state that satisfy Visibility, Well-Bracketing, and the following **State Change Conditions**:

- **(SC-P)**: The locations defined at a P-move \( m^S \) are those locations defined at the preceding O-move \( m_1^{S_1} \) (say), and possibly some **fresh** locations which are said to be **introduced** by \( m^S \). That is, \( \text{dom}(S_1) \subseteq \text{dom}(S) \), and none in \( \text{dom}(S) \setminus \text{dom}(S_1) \) has appeared earlier in the justified sequence.
- **(SC-O)**: The opening move has a null state. Let \( m^S \) be a non-initial O-move which is explicitly justified by \( m_0^{S_0} \), and let \( m_1^{S_1} \) be the move immediately preceding \( m^S \). Then \( S = S_0[S_1] \).

We shall call justified sequences satisfying these conditions **plays** or **legal positions**. We state some straightforward consequences of conditions (SC-P) and (SC-O):

**Lemma 6.**

(i) Every location that is defined at a move is introduced by some P-move that appears in the P-view at that point.

(ii) The locations defined at a P-move \( m^S \) include every location that is defined at the O-move \( m_0^{S_0} \) which explicitly justifies \( m \). That is, \( \text{dom}(S_0) \subseteq \text{dom}(S) \).

In the following, by a **P-view**, we shall mean a justified sequence that is the P-view of some legal position; similarly for **O-view**. The P-view of a legal position is a justified sequence that satisfies Visibility and Well-Bracketing (see [13]), though not necessarily State Change.

We need to check that the State Change Conditions correctly model the scoping of block-allocated local variables. Take a location \( l \) introduced by some P-move \( m \) in a
play(sm)u. We define the thread of l to be the subsequence of the play(sm)u consisting of moves at which l is defined (which corresponds to that part of the computation history when control is in the scope of l). Specifically we aim to verify that it follows from conditions (SC-P) and (SC-O) that whenever the thread of l is re-entered by O, the contents at l are set to the value last held in the thread. This is indeed the case for arenas of order at most three, as Lemma 7 makes precise (but not so at higher orders; see Remark 8). First we state some structural properties of legal positions of arenas of up to third order.

**Observation 1.** We assume a third-order arena.

(i) For any legal position of the shape $[0 u 1 v 2]$ that ends with a second-order O-question $[2]$ which is explicitly justified by $(1$, we have

$$u[0 u 1 v 2] = [0 (1 a_1 b_1 \cdots a_n b_n) 2$$

where $a_1 b_1 \cdots a_n b_n$ is a subsequence of $v$ such that each $a_i$ is a second-order O-question explicitly justified by $(1$, and each $b_i$ is either a second-order P-answer or a third-order P-question explicitly justified by $a_i$.

(ii) Take any legal position of the shape $u_1 [2 u_2 (3 u_3$ such that the third-order question $(3$ is explicitly justified by $[2$. If no move in $u_3$ is explicitly justified by $(3$ then no move in $u_2$ can appear in either the P-view or O-view at any move in $u_3$, and no move in $u_3$ is explicitly justified by any move in $u_2$. $\square$

As the observations are more or less obvious, we omit the proof.

**Lemma 7.** We assume an arena of order at most three. Let $s m_0 u m_1 m$ be a legal position in which the O-move $m$ is explicitly justified by $m_0$. Suppose the location l is defined at $m_0$. Then either l is defined at $m_1$ or l is not defined at any move in the segment u.

**Proof.** The move $m_0$ must be a P-question, which is either first-order or third-order. First, suppose the former.

(1a) In the case where $m$ is an O-answer which is by assumption explicitly justified by $m_0$, then by Well-Bracketing the segment $u m_1$ must be of the shape $[1 w_1]_1 \cdots [n w_n]_n$ (i.e. a sequence of segments, each beginning with an O-question and ending with a P-answer); and by Visibility each $[1$ is explicitly justified by $m_0$, as follows:

$$\cdots m_0 [1 w_1]_1 \cdots [n w_n]_n m_1 m$$

By condition (SC-O), l is defined at each $[1$, and since $m_1$ is explicitly justified by $[n$, l is defined at $m_1$ according to Lemma 6(ii).

(1b) In the case where $m$ is an O-question, then by Observation 1(i), the segment $u m_1$ must be of the shape $[1 w_1 b_1 [2 w_2 b_2 \cdots [n w_n b_n$ where each $[1$ is an O-question which is explicitly justified by $m_0$, and each $b_i$, which is either a third-order P-question or a
second-order P-answer, is explicitly justified by \([1], \text{ and } b_n = m_1\), as follows:

\[
\cdots (1 \ 2 \ \ldots \ n w_{m_1 m} b_n \ b_1 \ b_2 \ ldots w_n \ w_1) \\quad \text{u}.
\]

Since \(l\) is defined at \(m_0\), it is also defined at \(m_1\), which is within \(u\), and hence also at \(b_n = m_1\), by Lemma 6(ii).

Now suppose \(m_0\) is a third-order P-question. Since the arena is third-order, \(m\) must be an O-answer. By Well-Bracketing, the segment \(u m_1\) has the shape \([w_1] \ldots [w_n]\); further by Visibility each \([\cdot]\) is explicitly justified by some occurrence of a first-order question—call it \(\lambda\)—which explicitly justifies the second-order O-question—call it \(\mu\)—that explicitly justifies \(m_0\), as in the following:

\[
\cdots (1 \ 2 \ \ldots \ n w_{m_0 m} m_1 m m_1 m b_n \ b_1 \ b_2 \ ldots w_n \ w_1) \quad \text{u}.
\]

We consider all possible moves where \(l\) may have been introduced.

(2a) Suppose \(l\) is introduced at \(m_0\). Suppose, for a contradiction, \(l\) is defined at some move in the segment \(u m_1\). Then the first such in the segment at which \(l\) is defined must be an O-move; further it must be explicitly justified by \(m_0\) which forces it to be an O-answer, which contradicts the assumption that \(m\) is explicitly justified by \(m_0\).

(2b) Suppose \(l\) is introduced by some move in the segment \(u\) above. By Observation 1(ii), no move from \(u\) can appear in the P-view at any move in the segment \(u m_1\) and, so \(l\) is not defined in that segment.

(2c) Suppose \(l\) is defined at \(\mu\), then \(l\) must also be defined at \(\lambda\). Thus, by condition (SC-O) and Lemma 6(ii), \(l\) is defined at each \([\cdot]\) within the segment \(u m_1\) and, in particular, at \(m_1\).

Remark 8. Lemma 6 fails for fourth-order games. Consider the following justified sequence

\[
I_0 \ (1 \ 2 \ (3 \ 2) \ (2) \ (3) \ (4) \ (3) \ (2) \ 3) \quad \text{u}.
\]

In (1), the numeric subscript gives the (type-theoretic) order of the move, and the O-question \(\lambda\) is explicitly justified by \(\lambda\). The other pointers are completely determined by Well-Bracketing and Visibility. Assume that \(\lambda\) has a null state. Suppose \(\lambda\) introduces the location \(l\), and \(\lambda\) introduces the location \(l\). Then by condition (SC-O), \(l\) is defined at \(I_0\), and so, by condition (SC-P), \(l\) is also defined at \(I_2\). But \(l\), not \(l\), is defined at \(I_3\) and \(I_2\). Now \(l\) is defined at \(I_3\); it is not defined at \(m_1 = I_2\), but defined at some move in the segment \(u = [2 \ldots 3]\), which contradicts the lemma. The point is that, as the move \(3\) re-enters the thread of \(l\), condition (SC-O) says that \(l\) should be set to the value that was held at \(l\) at move \(3\), failing to take into account possible updates to \(l\) at \(I_4\). For a concrete example, the reader may wish to check that the legal position (1) is an interaction sequence that is played out when (the knowing strategy denoting) \(\lambda H.H(\lambda f.\text{new} z := 0 \ \text{in} f (z := 1 ; 0))\) of type \(((\text{exp} \Rightarrow \text{exp}) \Rightarrow \text{exp}) \Rightarrow \text{exp}) \Rightarrow \text{exp}\) is applied to (that denoting) \(\lambda g. g(\lambda x.g(\lambda y.x))\) of type \(((\text{exp} \Rightarrow \text{exp}) \Rightarrow \text{exp}) \Rightarrow \text{exp}\). □
The State Change conditions, and the consequent Lemma 6, define an innocent (i.e. view-dependent) notion of state, in the sense that at any point in a play, only the states of those moves that appear in the current P-view are relevant. By adopting a “history-sensitive” notion of state, it is possible to avoid the problem identified in Remark 8. Unfortunately the price one would then have to pay is “state explosion”—even restricted to finitary IA terms, the states that must be carried along in an interaction would not be bounded, and the strategies denoting such terms would not be compactly innocent.

A play is said to be complete just in the case where every question in it is answered. A strategy-with-state $\sigma$ of an arena $A$ is a set of prefix-closed legal positions of moves-with-state of $A$ satisfying:

(i) If even-length $s \in \sigma$ and $sm^S$ is a legal position then $sm^S \in \sigma$.
(ii) Determinacy: For any odd-length $s \in \sigma$ if $sm_1^S$ and $sm_2^S$ are both in $\sigma$ then $m_1 = m_2$ and $S_1 = S_2$.

We say that $\sigma$ is of order $n$ just in the case where $A$ is of order $n$. In the following, we shall often consider stateless strategies-with-state (i.e. the state of each move in every legal position is null) which are exactly the strategies in the sense of [13] (they are called knowing strategies in [4]). For any strategy-with-state $\sigma$, we write $cplays \sigma$ for the set of complete plays in $\sigma$.

For any justified (or interaction) sequence $u$ of moves-with-state, we define $erase u$ to be the sequence that is obtained from $u$ by erasing the state from each move; for any strategy-with-state $\sigma$, we define $erase \sigma = \{ erase s : s \in \sigma \}$. We state a useful fact:

**Lemma 9.** Let $\sigma$ be a strategy-with-state.

(i) For any $s, s' \in \sigma$, if $erase s = erase s'$ then $s = s'$.
(ii) $erase \sigma$ is a knowing strategy.

**Proof.** We prove (i) by induction on the length of $s$. The base case is trivial. For the inductive case, suppose $sm^S, sm'^S \in \sigma$. If $s$ is odd-length then $S = S'$ by Determinacy. If $s$ is even-length then, by condition (SC-O), $S$ and $S'$ are completely determined by $s$. Part (ii) is a straightforward consequence of part (i). $\square$

### 3.3. Composition of strategies-with-state

Suppose $\sigma$ and $\tau$ are strategies-with-state of arenas $A \Rightarrow B$ and $B \Rightarrow C$ respectively. Their composite $\sigma ; \tau$, which will be shown to be a strategy-with-state of $A \Rightarrow C$, is defined in the style of “parallel composition with hiding in CSP” (as is standard in Game Semantics) as far as the underlying moves (with pointers) are concerned. Roughly speaking, the states of the composite strategy are obtained as the disjoint unions of the respective states of the component strategies.

Let $S$ and $R$ be states. We define a new state $S(R)$, read “$S$ strongly updated by $R$”, as follows: for any $l \in \text{dom}(S(R))$ $\equiv \text{dom}(S) \cup \text{dom}(R)$, we have

$$S(R)(l) = \begin{cases} R(l) & \text{if } l \in \text{dom}(R) \\ S(l) & \text{otherwise.} \end{cases}$$
Note that the only difference between $S(R)$ and $S[R]$ is the domain of definition: we have $\text{dom}(S(R)) = \text{dom}(S) \cup \text{dom}(R)$, whereas $\text{dom}(S[R]) = \text{dom}(S)$; the angled brackets $\langle \cdot \rangle$ are intended to suggest an “expansion” of the domain of definition.

Let $s$ and $s'$ be justified sequences of an arena such that $\text{erase } s = \text{erase } s'$. We define a relation $s \prec s'$ (read “$s'$ is a state extension of $s$”) by recursion as follows: we have $\epsilon \prec \epsilon$; and $s m^S \prec s' m^{S'}$ holds provided:

(i) $s \prec s'$.
(ii) $S \subseteq S'$.
(iii) If $m$ is a P-move then $S' = S^{-}\langle S \rangle$ where $S^{-}$ is the state of the last move of $s'$.

We shall assume that $s \prec t$ has the force that $\text{erase } s = \text{erase } t$. Later in this section, the relation $\prec$ will be used in the definition of the composition of strategies-with-state; whenever we write $s \prec t$, $s$ will be a legal position, but $t$ will not necessarily be a legal position (because State Change may not necessarily be satisfied).

Let $u$ be a sequence of moves-with-states from $A$, $B$, and $C$ together with justification pointers from all moves except those initial in $C$. Define $u \upharpoonright (B, C)$ to be the subsequence of $u$ consisting of all moves-with-state (with pointers) from $B \Rightarrow C$; similarly define $u \upharpoonright (A, B, b)$ to be the subsequence of $u$ consisting of all moves-with-state (with pointers) from $A \Rightarrow B$ that are hereditarily justified by the occurrence $b$ of an initial $B$-move in $u$. We say that $u$ is an interaction sequence of $(A, B, C)$ if $u \upharpoonright (B, C)$ is a justified sequence of moves-with-state of $B \Rightarrow C$ satisfying Visibility and Well-Bracketing (but not necessarily State Change), and for each occurrence $b$ of an initial $B$-move in $u$, $u \upharpoonright (A, B, b)$ is a justified sequence of moves-with-state of $A \Rightarrow B$ satisfying Visibility and Well-Bracketing (but not necessarily State Change). We shall call $u \upharpoonright (B, C)$ the $(B, C)$-component of $u$, and call $u \upharpoonright (A, B, b)$ the $(A, B, b)$-component of $u$. Note that any move that occurs in the interaction sequence $u$ is either a P-move of $A \Rightarrow C$, or it is a generalized O-move in the sense that it is an O-move in exactly one of the components of $u$.

**Definition 10.** Take $\sigma$ and $\tau$ as before. We define $\text{ISeq}(\sigma, \tau)$ to be the set of interaction sequences $u$ of $(A, B, C)$ satisfying conditions (I1), (I2), and (I3) as follows:

**I1.** There exists some $t \in \tau$ such that $t \prec u \upharpoonright (B, C)$.

**I2.** For each occurrence of an initial $B$-move $b$ in $u$, there exists some $s \in \sigma$ such that $s \prec u \upharpoonright (A, B, b)$.

**I3.** Suppose $m^S_1$ and $m^S_2$ occur consecutively in $u$, and $m_1$ is a P-move in $A \Rightarrow C$. Let $m^S_0$ be the move that explicitly justifies $m_2$ in $u$. Then $S_2 = S_0[S_1]$.

We assume that the following sets are pairwise disjoint:

- $\mathcal{L}_{(A,B,b)}$—the set of locations introduced by P-moves of the component $(A, B, b)$, where $b$ ranges over occurrences of initial $B$-moves in $u$.
- $\mathcal{L}_{(B,C)}$—the set of locations introduced by P-moves of the component $(B, C)$.

Note that it follows from (I1) and (I2) that
I3'. If \( m^1_1 \) and \( m^2_2 \) occur consecutively in \( u \in \text{ISeq}(\sigma, \tau) \), and if \( m_1 \) is an O-move in component \( X \), then \( \text{dom}(S_1) \subseteq \text{dom}(S_2) \), and \( S_2 \upharpoonright \hat{L}_X = S_1 \upharpoonright \hat{L}_X \), where \((-)\) means set-complementation.

For any \( u \in \text{ISeq}(\sigma, \tau) \) we define \( u \upharpoonright (A, C) \) to be the justified sequence of moves-withstate of \( A \Rightarrow C \) that is obtained from \( u \) by first deleting all \( B \)-moves and then resetting pointers from initial \( A \)-moves to the opening \( C \)-move. We can now define the composite strategy

\[
\sigma ; \tau \overset{\text{def}}{=} \{ u \upharpoonright (A, C) : u \in \text{ISeq}(\sigma, \tau) \}.
\]

An important point to note is that conditions (I1), (I2), and (I3) above ensure that the states of an interaction sequence from \( \text{ISeq}(\sigma, \tau) \) are completely determined by the projected components, namely, \( u \upharpoonright (B, C) \) and \( u \upharpoonright (A, B, b) \) for each occurrence \( b \) of an initial \( B \)-move in \( u \), which are required to be state extensions of legal positions in \( \sigma \) and \( \tau \) respectively.

**Lemma 11.** Let \( \sigma \) and \( \tau \) be strategies-with-state over arenas of order at most three as before. For any \( u \in \text{ISeq}(\sigma, \tau) \), there is a unique \( t \in \tau \) such that \( t \ll u \upharpoonright (B, C) \); similarly for each \( b \), there is a unique \( s \in \sigma \) such that \( s \ll u \upharpoonright (A, B, b) \).

**Proof.** We shall just prove the lemma for component \( (B, C) \), as the argument for the other components of the kind \( (A, B, b) \) is the same. We aim to prove: for any \( t_1, t_2 \in \tau \), if \( t_1, t_2 \ll u \upharpoonright (B, C) \) then \( t_1 \equiv t_2 \), by induction on the length of \( t_1 \). The base case is trivial since the opening move of an interaction sequence must have a null state. For the inductive case, suppose \( tm^5s^1, tm^5s^2 \ll u \upharpoonright (B, C) \). If \( m \) is an O-move, then \( S_1 = S_2 \) because \( \tau \) satisfies Determinacy. Suppose \( m \) is a P-move from \( C \) (say), and the move in \( u \) that projects to the last move of \( u \upharpoonright (B, C) \) is \( n^T \). Now \( n^5_1 \) and \( n^5_2 \) must be explicitly justified by the same move in \( tm^5 \). Hence, by condition (SC-O), we have \( \text{dom}(S_1) = \text{dom}(S_2) \). Since \( S_1, S_2 \subseteq T \) by definition of \( \ll \), we have \( S_1 = S_2 \). We omit the case where \( m \) is a P-move from \( B \) as it is similar. \( \square \)

**Remark 12.** It may be helpful to unpack the definition of composition. Let \( b_1, \ldots, b_k \) be all the occurrences of initial \( B \)-moves in \( u m^S \in \text{ISeq}(\sigma, \tau) \). By definition, the state \( S \) is the union of the following “substates” (some or all may be null):

\[
S \upharpoonright \mathcal{L}_{(B, C)}, S \upharpoonright \mathcal{L}_{(A, B, b_1)}, \ldots, S \upharpoonright \mathcal{L}_{(A, B, b_k)}.
\]

We shall refer to \( S \upharpoonright \mathcal{L}_X \) as the \( X \)-component of \( S \). Suppose \( u m^S \in \text{ISeq}(\sigma, \tau) \) and \( m \) is an O-move in component \( (A, B, b) \). By Lemma 11 there is a unique \( s \in \sigma \) such that \( s \ll u m^S \upharpoonright (A, B, b) \). It follows from the definition that

\[
u m^S n^T \in \text{ISeq}(\sigma, \tau) \text{ if and only if for some } R, \text{ we have } s n^R \in \sigma \text{ and } T = S(R); \text{ equivalently } T \text{ is obtained from } S \text{ by replacing the } (A, B, b)\text{-component of } S \text{ by } R.
\]

Note that restricted to every other component, \( S \) and \( T \) are the same. The case where \( m \) is an O-move in component \( (B, C) \) is entirely similar.
Suppose the last move of \( u \in I\text{Seq}(\sigma, \tau) \) is an O-move in component \((B, C)\); thanks to Lemma 11, we can define \( \lceil u \rceil_{(B, C)} \) to be \( \lceil t \rceil \) for the unique \( t \in \tau \) such that \( t < u \upharpoonright (B, C) \); similarly for \( \lceil u \rceil_{(A, B, b)} \).

We can now prove that composition is well-defined.

**Lemma 13.** The set \( \sigma ; \tau \) as defined is a strategy-with-state over \( A \Rightarrow C \).

**Proof.** In view of Lemma 11, we can use arguments (such as those in [13, 19]) that are by now standard to show that \( \sigma ; \tau \) is a prefix-closed set of justified sequences over \( A \Rightarrow C \) that satisfy Visibility and Well-Bracketing. It remains to show that State Change is satisfied. Condition (SC-O) follows from (I3). For (SC-P), take an even-length \( s m^a n T \in \sigma ; \tau \) such that \( a \) is an \( b \)-move. By successive applications of (I3'), we have \( \text{dom}(S) \subseteq \text{dom}(S_1) \subseteq \cdots \subseteq \text{dom}(S_n) \subseteq \text{dom}(T) \) as required. \( \square \)

**Preservation of (third-order) composition by state-erasure**

An important result of this section is the preservation of composition by state-erasure for strategies of up to third order.

**Theorem 14.** For any strategies-with-state \( \sigma : A \Rightarrow B \) and \( \tau : B \Rightarrow C \) of order at most three, we have \( \text{erase}(\sigma ; \tau) = \text{erase} \sigma ; \text{erase} \tau \). \( \square \)

As the proof is rather technical, we relegate it to Appendix A.

The Theorem does not hold if the strategies-with-state in the Theorem are of orders greater than three—see Remark 42 for a discussion, even though composition of strategies-with-state is well-defined at all finite orders. Suitably quotiented, strategies-with-state do form a category, but we do not need this property because we will be using such strategies (and, later, the innocent such), not to build a semantics, but rather to give a programmable representation of a semantics.

### 3.4. Innocent strategies-with-state

The view of a legal position of moves-with-state is defined in exactly the same way as the standard (i.e. stateless) case. However note that the view of a legal position is not in general a legal position, because condition (SC-O) may fail. The notion of innocence in the sense of [13] can be extended to strategies-with-state, but we need to be careful with the freshness requirement in (SC-P). Suppose odd-length \( s \) and \( s' \) in an innocent strategy have the same P-view and suppose \( sm^T \in \sigma \) such that \( l \) is introduced at \( m \) by \( T \). Innocence says that \( s'm^{T'} \in \sigma \) and \( T = T' \), but (SC-P) requires a fresh copy of \( l \) to be introduced at \( T' \). Thus we need to say that a strategy-with-state \( \sigma \) is innocent if whenever the even-length \( sa^bh^T \in \sigma \), if \( ta^S \in \sigma \) and \( \lceil ta^S \rceil \equiv \lceil sa^S \rceil \) then \( ta^S b^{T'} \in \sigma \) such that \( T = T' \), where the two equalities regard different copies of the same location as equal.

**Relaxation of the freshness constraint**

Fortunately for innocent strategies-with-state over arenas of up to order three, the insistence on freshness can safely be relaxed. This has several desirable consequences for compactly innocent strategies. Not only are we justified to use the standard (and neater)
definition of innocence (see the formulation after the proof of Lemma 15), but more importantly, it makes a proof of compositionality possible. Crucially, in Section 5, we shall see that this enables a DPDA characterization of third-order strategies. Safety in this context amounts to the following properties: even though O may be able to force P to open as many threads of a location as he wishes,

1. no two threads of location \( l \) can ever overlap in time,
2. when a previous thread is re-entered, the contents of the location is set to the value last held in that thread.

To establish the properties, we first fix some notation. Take a play \( u \) in an innocent strategy \( \sigma \), and suppose \( t_1 a^S \leq t_2 \) and \( t_2 a^S \leq u \) (assume that these are all the occurrences of \( a \) in \( u \)) such that \( t_1 \gamma = t_2 \gamma = p \). Suppose \( l \) is a location introduced by the P-move \( a^S \). According to the freshness assumption in condition (SC-P), the location \( l \) introduced by the second occurrence of \( a \) (after \( t_2 \)) in \( u \) should be a fresh copy, distinct from the \( l \) introduced by the first occurrence of \( a \), because they define independent lifetimes of the same local variable.

- For (1), suppose \( l \) is defined at some move \( b^T \) in \( u \). Then, by Lemma 6, \( p a^S \) is a prefix of the P-view at that point, where \( a \) is one of the two occurrences of \( a \). Plainly only one such instance of \( a \) can appear in the P-view.
- For (2) we shall show that it is enough to modify the definition of \( S_0[S_1] \) slightly: Suppose \( m^S \) is an O-move which is explicitly justified by \( m_0 \), and suppose the move preceding \( m^S \) is \( m_1^S \) which is explicitly justified by \( m_0^S \); we define \( \text{dom}(S_0[S_1]) \) as follows:

\[
S_0[S_1](l) = \begin{cases} S_1(l) & \text{if } l \in \text{dom}(S_1) \cap \text{dom}(S_10), \\ S_0(l) & \text{otherwise.} \end{cases}
\]

**Lemma 15.** Take the setting of \( t_1 a^S \leq t_2 \) and \( t_2 a^S \leq u \in \sigma \), and \( l \) is introduced at \( a^S \) as before, and write \( p = t_1 \gamma = t_2 \gamma \). Let \( s m_0 u m_1 \) be a prefix of \( u \) in which the O-move \( m \) is explicitly justified by \( m_0 \). Suppose the location \( l \) is defined at \( m_0 \) (so by condition (SC-O), \( l \) is also defined at \( m \) and also at \( m_1 \). Equivalently \( p a^S \) is a prefix of both \( r^S m_0^S \) and \( r^S m_1 \). Then the same occurrence of \( a \) in \( u \) appears in the two P-views (equivalently, the \( l \) at \( m_0 \) and the \( l \) at \( m_1 \) belong to the same thread) iff \( l \) is defined at \( m_0 \).

**Proof.** We consider the two possibilities in turn:

(A) either the same occurrence of \( a^S \) in \( u \) appears in the two P-views, or
(B) they are from different occurrences.

With reference to the case analysis in the proof of Lemma 7, the cases that are consistent with (A) are (1a), (1b), and (2c). In all three, \( p a^S \), where \( a \) is that same occurrence of \( a \), is a prefix of the P-view at \( m_1 \), the move that explicitly justifies \( m_1 \); whence \( l \) is defined at \( m_1 \). The only cases consistent with (B) are (2a) and (2b). The occurrence of \( a^S \) in the P-view at \( m_1 \) is different from the occurrence in the P-view at \( m_0 \); i.e. they are (projected) from different occurrences of \( a^S \) in \( u \). We note that in these cases \( p a^S \) is not a prefix of the P-view at \( m_0 \); whence \( l \) is not defined at \( m_0 \). □
Warning. Henceforth we shall use the new definition of state update $S_0[S_1]$ without further comment.

Thanks to the lemma, we may say that a strategy-with-state $\sigma$ (of order at most three) is innocent if whenever the even-length $sa^Sb^T \in \sigma$, if $ta^S \in \sigma$ and $\tau ta^S \gamma = \tau sa^S \gamma$ then $ta^Sb^T \in \sigma$. As in the stateless case, innocent strategies are completely determined by view functions, which are maps from odd-length P-views $p$ to justified P-moves (i.e. a P-move together with a pointer into $p$). We say that $\sigma$ is generated by a view function $f$, written $\sigma = \text{strat}(f)$, just in the case where for any $s \in \sigma$, if $sa^S$ is a legal position, then we have $sa^Sb^T \in \sigma$ if and only if $\tau sa^S \gamma \in \text{dom}(f)$ and $f(\tau sa^S \gamma) = b^T$. (Note that the definition does not require every P-view in the domain of $f$ to be the P-view of some legal position in $\sigma$.)

**Theorem 16.** Suppose $\sigma : A \Rightarrow B$ and $\tau : B \Rightarrow C$ are innocent strategies-with-state of order at most three; then the composite $\sigma ; \tau$ is an innocent strategy-with-state. □

For the proof, we use essentially the same argument as the compositionality proof in [13]. The key idea is what McCusker has aptly referred to as the core, written $\overline{u}$, of an interaction sequence $u \in \text{ISeq}(\sigma, \tau)$ (see also [13, Proposition 5.4]). We set out the proof in Appendix B.

**Example 17.** We give some examples of innocent strategies-with-state.

(i) The usual innocent strategies (without state) are of course innocent strategies-with-states. Thus the canonical maps such as identities and projections are examples. If $\sigma : C \Rightarrow A$ and $\tau : C \Rightarrow B$ are innocent strategies-with-state, so is the pairing $\langle \sigma, \tau \rangle : C \Rightarrow (A \times B)$. Note also that modulo renaming of moves, innocent strategies-with-state of $C \Rightarrow (A \Rightarrow B)$ are the same as those of $C \times A \Rightarrow B$.

(ii) The strategy generated by the “good-location” view function $\text{loC}_{i,1}$ (see Section 4.2) is an important example.

### 3.5. Effective compositionality of compactly innocent strategies-with-state

The view function $f$ that generates an innocent strategy-with-states can be presented as the least prefix-closed set $\text{tree}(f)$ of P-views such that even-length P-views $pm^S \in \text{tree}(f)$ iff $f(p) = m^S$. We shall refer to $\text{tree}(f)$ as the evaluation tree of $f$ (or of $\text{strat}(f)$). Note that in general $\text{tree}(f) \not\subseteq \text{strat}(f)$ and $\text{strat}(f) \not\subseteq \text{tree}(f)$.

We say that an innocent strategy-with-state is compact if it is generated by a view function whose domain of definition is finite. The last result of this section is the effective compositionality of compact innocent strategies-with-state. We give an algorithm that takes view functions $f$ and $g$ such that $\text{strat}(f) : A \Rightarrow B$ and $\text{strat}(g) : B \Rightarrow C$ are innocent, and returns the evaluation tree of the composite $\text{strat}(f) ; \text{strat}(g)$. This algorithm is not new: it can be extracted from the proof of compositionality of innocent strategies in [13]. The version here, called Algorithm A in Fig. 3, is presented specifically for the stateful case, which of course applies only to innocent strategies-with-states of order at most three. However it is important to stress that the stateless version of the Algorithm (which is obtained from Algorithm A by removing all state information) applies to innocent strategies of all orders in the stateless case.
Algorithm A

Input: Finite view functions \( f \) and \( g \) such that \( \text{strat}(f) : A \Rightarrow B \) and \( \text{strat}(g) : B \Rightarrow C \) are innocent strategies-with-state

Output: Evaluation tree \( \text{tree}(h) \) of a view function \( h \) such that \( \text{strat}(h) = \text{strat}(f) \cup \text{strat}(g) \)

1. letrec Next \( u m^S = \% \) Assume that \( m^S \) is a generalized O-move.
2. % Suppose \( m^S \) is an O-move in component \( X \) of \( u \).
3. if \( h_X (\langle u m^S \rangle X^S) = n^R \)
4. then if \( n^R \) is a P-move in \( A \Rightarrow C \)
5. then return \( u m^S n^T \) where \( T = S(R) \)
6. else Next \( u m^S n^T \) % i.e. \( n^T \) is a generalized O-move
7. else return “fail”;

8. letrec Grow \( u m^S = \% \) Assume that \( m^S \) is a P-move from \( A \Rightarrow C \)
9. for \( n : m \vdash _S n \) do % where \( \vdash_S = A \vee C \) as appropriate
10. for \( T \) such that \( \text{dom}(T) = \text{dom}(S) \) do
11. if Next \( u m^S n^T = v \) (i.e. \( v \neq \text{“fail”} \))
12. then \( H := H \cup \{ v \} \); Grow \( v \);

13. let \( H = \emptyset \) in for \( m : \star \vdash_{A \Rightarrow C} m \) do
14. if Next \( m = u \) (i.e. \( u \neq \text{“fail”} \)) then \( H := H \cup \{ u \} \);
15. for \( u \in H \) do Grow \( u \);
16. return \( H \vdash (A, C) \);

Fig. 3. Algorithm A.

Theorem 18 (Effective Compositionality). Algorithm A takes as input finite view functions \( f \) and \( g \) such that \( \text{strat}(f) : A \Rightarrow B \) and \( \text{strat}(g) : B \Rightarrow C \) are innocent strategies of finite arenas, and returns the evaluation tree \( \text{tree}(h) \) of a view function \( h \) such that \( \text{strat}(h) = \text{strat}(f) \cup \text{strat}(g) \). □

Algorithm A works by generating interaction sequences \( u \) between \( \text{strat}(f) \) and \( \text{strat}(g) \) which are short-sighted in the sense that any move in \( u \) which is an O-move of \( A \Rightarrow C \) is explicitly justified by the preceding move in \( u \) (which must be a P-move in \( A \Rightarrow C \), by the Switching Condition 2 for function space arenas). To the best of our knowledge, no proof of the effective compositionality of compact innocent strategies has ever appeared in the literature. Here we shall just prove termination of the algorithm, which amounts to showing that there are no infinite short-sighted interaction sequences.

The stateless case

We formulate the result, namely Theorem 21, in a general form that applies to stateless innocent strategies of all orders. By virtue of the strong correspondence between compactly

\( ^2 \) For any legal position \( u m m' \) of a function space arena \( A \Rightarrow B \), if \( m \) is a P-move, then either \( m \) and \( m' \) are both from \( A \) or both from \( B \).
Proof. Suppose, for a contradiction, there is such an infinite sequence \( B \) that every \( O \)-move from \( A \) in \( t \) is explicitly justified by the preceding move; then \( \uparrow t \uparrow B = \uparrow t \uparrow A \Rightarrow B \uparrow B \). □

We omit the straightforward induction argument.

**Lemma 20** (Hyland). For any infinite play \( w \) over a finite arena, either the set of \( P \)-views spanned by \( w \) (i.e. \( \{ \downarrow s \vdash s \leq \omega \} \) where \( s \leq \omega \) means “\( s \) is a prefix of \( \omega \)” is infinite, or the set of \( O \)-views spanned by \( w \) is infinite. □

The above lemma first appeared (in a slightly different form) in [12]. It can be proved by induction on the size of the arena.

We can now establish termination of the algorithm:

**Theorem 21** (The Stateless Case). Suppose \( A, B, \) and \( C \) are arenas that have finite depths (i.e. the orders of questions have a finite upper bound), and \( f \) and \( g \) are finite view functions over arenas \( A \Rightarrow B \) and \( B \Rightarrow C \) respectively. There is no infinite justified sequence such that every finite prefix is short-sighted and in \( \text{ISeq}(\text{strat}(f), \text{strat}(g)) \).

**Proof.** Suppose, for a contradiction, there is such an infinite sequence \( \pi \). Then \( \pi \uparrow (B, C) \) must contain infinitely many occurrences of \( B \)-moves; for otherwise, beyond a certain point, either \( \pi \) consists of only \( C \)-moves, or it consists of only \( A \)-moves from \( \pi \uparrow (A, B, b) \), for some occurrence \( b \) of an initial \( B \)-move. Suppose the latter. As \( \pi \) is assumed to be short-sighted, after some finite prefix, \( \pi \uparrow (A, B, b) \) must also be short-sighted (because \( \pi \) consists of only \( A \)-moves). Thus the set of \( P \)-views spanned by \( \pi \uparrow (A, B, b) \) is infinite, which contradicts the finiteness of \( f \). A similar argument applies in the former case.

By assumption, every finite prefix of \( \pi \uparrow (B, C) \) (which we shall write simply as \( \pi_0 \)) is in \( \text{strat}(g) \). Since \( g \) is finite, the set of \( P \)-views spanned by \( \pi_0 \) is finite. We claim that the set of \( O \)-views spanned by \( \pi_0 \) is also finite. As \( \pi_0 \) is infinite, this would contradict Lemma 20.

We first consider \( O \)-views of finite prefixes of \( \pi_0 \) that end in a \( C \)-move. Because of short-sightedness, every \( O \)-move in \( \pi_0 \) that is from \( C \) is explicitly justified by the preceding (\( P \))-move, which is also from \( C \), by the Switching Condition for function space arenas. Thus by a straightforward induction argument, \( \downarrow (\pi_0)_{\leq \omega} \vdash c \), where \( c \) is a \( C \)-move in \( \pi_0 \), is precisely the justification history of \( c \). Since \( B \Rightarrow C \) has finite depth, there are only finitely many \( O \)-views of such a shape.

Next we consider \( O \)-views of finite prefixes of \( \pi_0 \) that end in a \( B \)-move \( b \). We write \( b_0 \) for the (occurrence of the) initial \( B \)-move in \( \pi_0 \) that hereditarily justifies \( b \). By the O-view Projection lemma in [13], we have

\[
\downarrow (\pi_0)_{\leq b} \Rightarrow C = c_0 \uparrow (\pi_0)_{\leq b} \vdash (B, b_0) \Rightarrow B
\]

where \( c_0 \) is the opening move of \( \pi_0 \). Set \( t_b = \pi_{\leq b} \uparrow (A, B, b_0) \). We then have \( (\pi_0)_{\leq b} \vdash (B, b_0) = t_b \uparrow B \). As \( t_b \) satisfies the premises of Lemma 19, we have \( \uparrow t_b \uparrow B = \uparrow t_{b} \uparrow A \Rightarrow B \uparrow B \). Since \( t_b \in \text{strat}(f) \) and \( f \) is finite, there are only finitely many \( P \)-views of the form \( \uparrow t_{b} \uparrow A \Rightarrow B \), and hence there are only finitely many \( P \)-views of the
form $\Gamma b^\top A \Rightarrow B \mid B$. Thus the set of P-views of the form $\Gamma (\pi_0)_{\leq b} \mid (B, b_0)^\top B$, and hence the set of O-views of the form $\angle (\pi_0)_{\leq b} \mid B \Rightarrow C$, as $b$ ranges over occurrences of $B$-moves in $\pi_0$, is finite. □

The third-order stateful case

We specialize Theorem 21 to the stateful case:

Lemma 22 (The Third-order Stateful Case). Suppose $f$ and $g$ are finite view functions over finite arenas $A \Rightarrow B$ and $B \Rightarrow C$ respectively. Assume that the arenas are at most third-order. There is no infinite justified sequence such that every finite prefix is short-sighted and in $ISeq(strat(f), strat(g))$.

Proof. Suppose, for a contradiction, there is such an infinite sequence. Erase all state information from the infinite sequence, and call the resultant sequence $\pi$. With the proof of Theorem 21, noting that all occurrences of $strat(f)$ and $strat(g)$ in the argument should be replaced by $erase(strat(f))$ and $erase(strat(g))$ respectively. Note that even though the erased strategy $erase(strat(f))$ is not in general innocent, the set of P-views spanned by legal positions in $erase(strat(f))$ is finite; similarly for $erase(strat(g))$. Indeed $erase(strat(f))$ is generated, not by a finite view function, but by a finite view relation. □

Lemma 22 can also be proved by a syntactic argument (using Theorem 27). It is essentially equivalent to the normalizability of third-order finitary IA terms to finite canonical forms defined in Section 4.3.

4. Innocent representation of third-order Idealized Algol

We begin this section with a summary of the knowing-strategy semantics of IA. We then give the innocent strategy-with-state denotation of third-order IA, and prove that the strategies denoting the finitary fragment are compactly innocent and effectively computable. The innocent denotation is not a model of the (theory of IA restricted to the) third-order fragment; rather it should be regarded as an algorithmic representation of the knowing-strategy semantics. The relationship between the two denotations is given in Theorem 25. We conclude the section with a definability result.

4.1. Knowing-strategy semantics of IA

For notational simplicity, we shall write $exp$ for the arena that denotes the type $exp$, and similarly for $loc$ and $com$. The arena $exp$ is just the standard natural numbers arena with move-set $\{q\} \cup \mathbb{N}$ where $q$ is the initial question that justifies each answer $n \in \mathbb{N}$. The arena $com$ is a two-move arena whose initial question $run$ justifies the answer-move $done$. Following Reynolds, the type $loc$ of locations is interpreted in an object-oriented style as a product of its “read method” and its “write method”. Thus the arena $loc$ is the product arena $exp \times com^\omega$, whereby the first component is the value that is held at the location, and the second component contains countably many commands to write 0, 1, 2, etc., to the location; we write the question-move and the answer-move in the $i$-th copy of $com$ in $loc$ as $write(i)$ and $ok_i$ (or simply $ok$) respectively.
The purely functional constructs of IA are standardly interpreted by the canonical maps of the cartesian closed category of arenas and knowing strategies. Here we shall just give the semantic definitions of the imperative constructs. Command sequencing is interpreted by the strategy seqcomp over the arena com ⇒ (com ⇒ com) specified by the following (unique) maximal play:

\[
\text{com} \Rightarrow \text{com} \Rightarrow \text{com}
\]

run
done

run
done
done

To interpret dereferencing and assignment to variable, we exploit the fact that \(\text{com}^\omega\) is a retract of \(\text{exp} \Rightarrow \text{com}\). That is to say, writing the maps as

\[
\text{com}^\omega \xrightarrow{\text{sec}} (\text{exp} \Rightarrow \text{com}) \xrightarrow{\text{ret}} \text{com}^\omega
\]

we have \(\text{sec} ; \text{ret} = \text{id}_{\text{com}^\omega}\). For assignment, we use the map

\[
\text{loc} \xrightarrow{\pi_2} \text{com}^\omega \xrightarrow{\text{sec}} (\text{exp} \Rightarrow \text{com})
\]

while dereferencing is just the first projection \(\pi_1 : \text{loc} \rightarrow \text{exp}\). The bad-location construct \(\text{mkloc}\) is interpreted as

\[
\text{exp} \times (\text{exp} \Rightarrow \text{com}) \xrightarrow{\text{id} \times \text{ret}} (\text{exp} \times \text{com}^\omega) = \text{loc}.
\]

The reader may wish to check that all the imperative constructs considered thus far are in fact interpreted by innocent strategies. It now remains to consider the interpretation of the new-block, for which we need to introduce a non-innocent strategy. For each \(i \geq 0\), we define a knowing strategy new\(_i\) over the arena \((\text{loc} \Rightarrow \text{exp}) \Rightarrow \text{exp}\) by specifying its maximal plays to be words that are generated by the linear\(^3\) context-free grammar:

\[
S \rightarrow q_1 \cdot q_2 \cdot S_i
\]

\[
k \geq 0, \quad S_k \rightarrow \text{read} \cdot k \cdot S_k + \left( \sum_{j \geq 0} \text{write}(j) \cdot \text{ok} \cdot S_j \right) + \left( \sum_{m \geq 0} m_2 \cdot m_1 \right)
\]

or equivalently words that match the regular expression:

\[
q_1 \cdot q_2 \cdot \left( \sum_{n \geq 0} \text{write}(n) \cdot \text{ok} \cdot \left( \sum_{m \geq 0} m_2 \cdot m_1 \right) \right)^* \cdot \sum_{m \geq 0} m_2 \cdot m_1.
\]

The alphabet in question is the move-set of \((\text{loc} \Rightarrow \text{exp}) \Rightarrow \text{exp}\). To distinguish the two copies of \text{exp}, we mark moves from the rightmost copy of \text{exp} with subscript 1, and those

\(^3\) A context-free grammar is linear if every rule is either of the form \(P \rightarrow w Q\) or of the form \(P \rightarrow w\), where \(P\) and \(Q\) are non-terminal symbols and \(w\) is a word of the alphabet. A language over a finite alphabet is regular if and only if it is generated by a linear grammar.
from the other copy with subscript 2. For arenas of up to second order, we may safely represent legal positions as sequences of moves, as the justification pointers are uniquely reconstructible. Note that as defined, the language generated by the grammar is not regular because $\exp$ (and hence the alphabet) is infinite. In the following, we shall consider versions of $\text{new}_i$ over finite arenas ($\text{loc} \Rightarrow \exp_N \Rightarrow \exp_N$, where $\exp_N$ is a finite subarena of $\exp$).

The strategy $\text{new}_i$ has precisely the behaviour expected of a “good location” which has been initialized to $i$: the answer in response to a read is always the last value written, or $i$ if no write-move has yet been played. There is a similar version of the “good location strategy” over the arena $(\text{loc} \Rightarrow \text{com}) \Rightarrow \text{com}$, which we shall also write as $\text{new}_i$. We leave its definition as an exercise for the reader. We can now define

$$\llbracket \Gamma \vdash \text{new}_x := n \in M : \beta \rrbracket^\kappa \overset{\text{def}}{=} \llbracket \Gamma \vdash \lambda x : \text{loc} . M : \text{loc} \Rightarrow \beta \rrbracket^\kappa ; \text{new}_i$$

where $\beta = \exp, \text{com}$.

Using knowing-strategy semantics, the observational preorder of IA may be characterized in terms of complete plays as follows:

**Theorem 23** (Characterization). For any $\Gamma$, $A$, $M_1$, and $M_2$ such that $\Gamma \vdash M_i : A$ is provable for $i = 1, 2$, we have

$$M_1 \sqsubseteq M_2 \iff \text{cplays} \llbracket \Gamma \vdash M_1 : A \rrbracket^\kappa \subseteq \text{cplays} \llbracket \Gamma \vdash M_2 : A \rrbracket^\kappa.$$}

For a proof, we recommend the comparatively simple argument in [20]; see also [4]. The Characterization Theorem plays an important role in the main decidability result of the paper.

### 4.2. Innocent strategy-with-state representation

Henceforth, without further mention, all arenas, plays, strategies, and IA terms-in-context are assumed to have order at most three.

Consider the knowing-strategy interpretation $\llbracket \Gamma, x : \text{loc} \vdash M : \exp \rrbracket^\kappa$. Over the arena in question, only P can switch in and out of the loc component. After P has played a write($i$) move, O can only play ok, but after a read move, O is free to play any answer move $n \in \mathbb{N}$. Thus the values that are read from a (free) assignable variable are unrelated to the values that have been written to it. The $\text{new}$ binder, which is interpreted by precomposition with $\text{new}_j$, forces the memory cell to behave as a “good location”; i.e. $\llbracket \Gamma \vdash \text{new}_x := j \in M : \exp \rrbracket^\kappa$ consists of plays that are obtained by first taking those of $\llbracket \Gamma, x : \text{loc} \vdash M : \exp \rrbracket^\kappa$ in which each read is followed by the last written value, and then hiding (i.e. deleting) all moves from the loc component, thus making the state implicit. The idea behind the strategy-with-state denotation is simply to recover the hidden state information $S$ at each move and represent it explicitly with the move $m$, forming a pair $m^S$, which we call moves-with-state. One way to think of the explicit semantics is to consider the uncovering [13] of plays in $\llbracket \Gamma \vdash \text{new}_x := j \in M : \exp \rrbracket^\kappa$ with respect to the hidden loc component: if $m^S$ is the first visible move after a hidden write($i$) move, then $S(l) = i$ recording the state change. Note that such an $m^S$ is necessarily a P-move (because of the Switching Condition), so only P can change the state. A key development of this approach is that up to third order, such strategies-with-state are innocent, i.e. view dependent.
The strategy-with-state denotation of a third-order IA term-in-context, which we shall write as \( \llbracket \Gamma \vdash M : A \rrbracket \), is defined for the most part by the same inductive clauses as define the knowing-strategy semantics. Strictly speaking, the innocent strategy-with-state denotation is not a semantics of IA (because it does not model the theory), but a representation of the knowing-strategy semantics. The precise relationship between the representation and the semantics is given in Theorem 25.

We first observe that the semantic constructions defining the purely functional part of IA are operations that preserve innocent strategies-with-state:

(i) The canonical maps \( \text{eval}_{A, B} : (A \Rightarrow B) \times A \Rightarrow B \) and \( \pi_i : A_1 \times A_2 \rightarrow A_i \) are innocent strategies-with-state (because they are innocent strategies in the standard, state-less, sense).

(ii) Given innocent strategies-with-state \( \sigma : C \Rightarrow A \) and \( \tau : C \Rightarrow B \), the standard pairing construction defines an innocent strategy-with-state \( \langle \sigma, \tau \rangle : C \Rightarrow (A \times B) \). Note also that \( \text{erase} (\sigma, \tau) = (\text{erase} \sigma, \text{erase} \tau) \).

(iii) Modulo renaming of moves, innocent strategies-with-state of the arena \( (C \times A) \Rightarrow B \) are exactly those of the arena \( (A \Rightarrow B) \).

Thus we can use the standard inductive clauses to define the innocent strategy-with-state denotation of the purely functional constructs. For the interpretation of command sequencing, assignment, dereferencing, and the bad-location constructor, we can also use the same inductive clauses as define their respective knowing-strategy semantics:

\[
\begin{align*}
\llbracket \Gamma \vdash M ; N : \beta \rrbracket &= \langle \llbracket \Gamma \vdash M : \text{com} \rrbracket, \llbracket \Gamma \vdash N : \beta \rrbracket \rangle ; \text{seqcomp} \\
\llbracket \Gamma \vdash N : \text{com} \rrbracket &= \langle \llbracket \Gamma \vdash M : \text{loc} \rrbracket, \llbracket \Gamma \vdash N : \text{exp} \rrbracket \rangle ; \text{uncurry}(\pi_2; \text{sec}) \\
\llbracket \Gamma \vdash \text{!}M : \text{exp} \rrbracket &= \langle \llbracket \Gamma \vdash M : \text{loc} \rrbracket, \pi_2 \rangle \\
\end{align*}
\]

and

\[
\begin{align*}
\llbracket \Gamma \vdash \text{mkloc} M N : \text{loc} \rrbracket \\
&= \langle \llbracket \Gamma \vdash \text{mkloc} N : \text{exp} \rrbracket, \llbracket \Gamma \vdash \text{mkloc} M : \text{exp} \Rightarrow \text{com} \rrbracket \rangle ; \text{id} \times \text{ret}.
\end{align*}
\]

Note that \( \text{seqcomp}, \text{uncurry}(\pi_2; \text{sec}), \pi_2, \) and \( \text{id} \times \text{ret} \) are all innocent, and innocent strategies-with-state compose.

Finally, for the new-block, given any \( i \geq 0 \) and any location \( l \), we define the view function \( \text{loc}_{i,l} \) over the arena \( (\text{loc} \Rightarrow \text{exp}) \Rightarrow \text{exp} \):

\[
\text{loc}_{i,l} : \left\{ \begin{array}{c}
q_1 \cdot q_2^{(l,i)} \cdot \text{read}(l,k) \mapsto k^{(l,k)} \\
q_1 \cdot q_2^{(l,i)} \cdot \text{write}(j)^{(l,k)} \mapsto o(k^{(l,i)}) \\
q_1 \cdot q_2^{(l,i)} \cdot m_2^{(l,k)} \mapsto m_1^{(l,k)}
\end{array} \right.
\]

for each \( j, k, m \geq 0 \). As in (2), we use subscripts to distinguish moves from the two copies of \( \text{exp} \); we write \( m^{(l,i)} \) as a shorthand for the move whose state maps location \( l \) to \( i \). The idea behind the view function is simple: the state of a move in \( \text{strat}(\text{loc}_{i,l}) \) is given by the value held at the location \( l \) at that point. Thus the answer in response to a read-move \( \text{read}(l,k) \) is the value stored at \( l \), namely \( k^{(l,k)} \) (with the state left unchanged); the state can only be changed by a write-move: \( P \)'s response to the O-move \( \text{write}(j)^{(l,k)} \)
is to change the state to \((l, j)\). The reader may wish to check that the maximal plays of \(\text{strat}(\text{loc}_l, i)\), as sequences of moves, form a language generated by the following linear grammar:

\[
S \rightarrow q_1 \cdot q_2^{(l,i)} \cdot S_i \\
k \geq 0, \quad S_k \rightarrow \text{read}^{(l,k)} \cdot k^{(l,k)} \cdot S_k \\
+ \sum_{j \geq 0} \text{write}^{(j, l)} \cdot \alpha^{(l,j)} \cdot S_j \\
+ \sum_{m \geq 0} m_2^{(l,k)} \cdot m_1^{(l,k)}.
\]  

(5)

This is just another description of the behaviour of a “good location” initialized to \(i\), which is equivalent to that specified by the knowing strategy \(\text{new}_i\) in the following sense:

**Lemma 24.** For each \(i \geq 0\), we have \(\text{erase}(\text{strat}(\text{loc}_l, i)) = \text{new}_i\).

**Proof.** As the maximal plays of \(\text{strat}(\text{loc}_l, i)\) are generated by the grammar (5), the maximal plays of \(\text{erase}(\text{strat}(\text{loc}_l, i))\) are generated by the rules obtained from (5) by state-erasure. The erased rules are exactly those that generate the maximal plays of \(\text{new}_i\) in (2). \(\Box\)

We leave the definition of the corresponding “good location” view function over the arena \((\text{loc} \Rightarrow \text{com}) \Rightarrow \text{com}\), which we shall also write as \(\text{loc}_{l,i}\), as a exercise. For \(\beta = \exp\) and \(\text{com}\), we are now in a position to define

\[
\llbracket \Gamma \vdash \text{new}_x := n \in M : \beta \rrbracket \cdot \llbracket \Gamma \vdash \lambda x : \text{loc} : \text{loc} \Rightarrow \beta \rrbracket ; \text{strat}(\text{loc}_{l,n})
\]

where \(l\) is a fresh location. Thanks to Lemma 24 and Theorem 14, we can clarify the relationship between the innocent-strategy denotation of a term and its knowing-strategy semantics: the former should be regarded as an algorithmic representation of the latter.

**Theorem 25.** For any valid finitary third-order IA term-in-context \(\Gamma \vdash M : A\):

(i) We have \(\text{erase}(\llbracket \Gamma \vdash M : A \rrbracket) = \llbracket \Gamma \vdash M : A \rrbracket^k\).

(ii) If the types are built up from finite base types, then \(\llbracket \Gamma \vdash M : A \rrbracket\) is a compactly innocent strategy-with-state \(\text{strat}(f_M)\), and the interpretation \(M \mapsto f_M\) is effectively computable.

**Proof.** We prove (i) and (ii) simultaneously by structural induction over \(M\). We shall only consider two constructs for illustration. In the case of application, we have

\[
\text{erase}(\llbracket \Gamma \vdash M \cdot N : B \rrbracket) \\
= \text{erase}(\llbracket \Gamma \vdash M : A \Rightarrow B \rrbracket, \llbracket \Gamma \vdash N : A \rrbracket) ; \text{erase} \text{ eval} \quad \text{by Theorem 14} \\
= (\text{erase}(\llbracket \Gamma \vdash M : A \Rightarrow B \rrbracket, \llbracket \Gamma \vdash N : A \rrbracket) ; \text{eval}) \quad \text{by I.H.} \\
= \llbracket \Gamma \vdash M \cdot N : B \rrbracket^k.
\]

For (ii) we note that \(\llbracket \Gamma \vdash M : A \Rightarrow B \rrbracket^k, \llbracket \Gamma \vdash N : A \rrbracket^k\) is a compactly innocent strategy-with-state, because by the induction hypothesis both the component strategies are compact, and pairing preserves compactness. Since \(\text{eval}\) is compactly innocent,
and composition preserves compactly innocent strategies-with-state (Theorem 18), $\llbracket \Gamma \vdash \text{MN} : B \rrbracket^\kappa$ is a compactly innocent strategy-with-state. For effectivity, we need only observe that pairing and composition are effective operations. The same reasoning applies for the case of command sequencing, assignment, dereferencing and the bad-location constructor.

Next we consider the case of the new-block:

\[
\text{erase} \llbracket \Gamma \vdash \text{new } x := n \text{ in } M : \beta \rrbracket = \text{erase} \llbracket \Gamma \vdash \lambda x : \text{loc}.M : \text{loc} \Rightarrow \beta \rrbracket ; \text{strat}(\text{loc}_{l,n}) \rrbracket \text{ by Theorem 14}
\]

\[
\llbracket \Gamma \vdash \text{new } x := n \text{ in } M : \beta \rrbracket^\kappa = \llbracket \Gamma \vdash \lambda x : \text{loc}.M : \text{loc} \Rightarrow \beta \rrbracket^\kappa ; \text{new}_n
\]

The last step is by the induction hypothesis and by Lemma 24. Part (ii) follows from Theorem 18 and the fact that $\text{loc}_{l,i}$ is a finite view function. □

**Example 26.** For ease of explanation we shall consider a third-order type that is built up from a finite base type $\exp[\mathbb{B}]$ where $\mathbb{B} = \{0, 1\}$. Take the following terms:

\[
\begin{align*}
\Theta_1 &= \lambda F.\text{new } x := 0 \text{ in } F(\lambda y.(\text{ifzero}!x \text{ then } x := y \text{ else } x := \neg y);!x) \\
\Theta_2 &= \lambda F.F(\lambda y.\text{new } x := 0 \text{ in } (\text{ifzero}!x \text{ then } x := y \text{ else } x := \neg y);!x) \\
\Theta_3 &= \lambda F.F(\lambda y.y)
\end{align*}
\]

of type $(\exp[\mathbb{B}] \Rightarrow \exp[\mathbb{B}]) \Rightarrow \exp[\mathbb{B}]$. View functions that generate $\llbracket \Theta_1 \rrbracket$ and $\llbracket \Theta_2 \rrbracket$ are presented as evaluation trees in Figs. 4 and 5 respectively. In the figures, $q_0$, $q_1$, $q_2$, and $q_3$ are the four questions of the arena (the subscript is the type-theoretic order of the question); “$m$, $i$” is a shorthand for the move $m$ whose state maps a fixed location $l$ (say) to $i$, whereas “$m$” means that the state is null. Note that the view function specified by the tree is just the map $p \mapsto m^{(l,i)}$, where $p \cdot "m, i"$ ranges over even-length path of the tree; the pointer of $m$ into $p$ is uniquely determined in each case.
Paths in an evaluation tree (which are P-views) are not necessarily legal positions because the condition (SC-O) is not required to hold. However the reader may wish to verify that every P-view in the evaluation tree (determined by) $\Theta_2$ is the P-view of some legal position in the associated innocent strategy; the same is true of $\Theta_1$. (Note that this is not true in general: just consider, for example, the evaluation tree determined by $\lambda x.\text{new } z := 0 \text{ in } x$.)

A behavioural difference between the two strategies $\Theta_1$ and $\Theta_2$ is worth noting: the else branch of $\Theta_2$ is never executed. In fact $\Theta_2$ determines the same evaluation tree as $\Theta_4 = \lambda F.(\lambda y.\text{new } x := 0 \text{ in } x;!x)$.

That is, $\llbracket \Theta_2 \rrbracket = \llbracket \Theta_4 \rrbracket$.

As an illustration of operational reasoning based on game semantics, we shall prove that $\Theta_2 \approx \Theta_3$, but $\Theta_1 \not\approx \Theta_2$ and $\Theta_2 \not\approx \Theta_1$. Now it is straightforward to see that $\text{erase } \llbracket \Theta_4 \rrbracket = \llbracket \Theta_2 \rrbracket^e$. It then follows from Theorem 25(i) that $\text{cplays } \llbracket \Theta_2 \rrbracket^e = \text{cplays } \llbracket \Theta_3 \rrbracket^e$, and hence, by Theorem 23, we have $\Theta_2 \approx \Theta_3$.

Using the view function in Fig. 4, we construct the following legal positions of moves-with-state:

\[
q_0 (q_1, 0) (q_2, 0) (q_3, 0) (1, 0) (1, 1) (q_2, 1) (q_3, 1) (0, 1) (1, 1) (1, 1)
\]

is in $\llbracket \Theta_1 \rrbracket$. (Note that neither $q_0 (q_1, 0) (q_2, 1)$ nor $q_0 (q_1, 0) (q_2, 0) (q_3, 0) (1, 1)$ (say) are legal positions of $\llbracket \Theta_1 \rrbracket$ because condition (SC-O) is violated.) Similarly we have

\[
q_0 q_1 q_2 (q_3, 0) (1, 0) (1, 1) q_2 (q_3, 0) (0, 0) (0, 0) 00
\]

is in $\llbracket \Theta_2 \rrbracket$. (Again the justification pointers in the legal positions above are uniquely reconstructible.) Thus, by Theorem 25, we obtain the following (complete) plays by erasure:

\[
\begin{align*}
q_0 q_1 q_2 q_3 1 1 & q_2 q_3 0 1 1 1 \in \text{erase } \llbracket \Theta_1 \rrbracket = \llbracket \Theta_1 \rrbracket^e \\
q_0 q_1 q_2 q_3 1 1 & q_2 q_3 0 0 0 0 \in \text{erase } \llbracket \Theta_2 \rrbracket = \llbracket \Theta_2 \rrbracket^e.
\end{align*}
\]
Observe that owing to Determinacy the first legal position above does not belong to $\|\Theta_2\|^\kappa$, and for the same reason, the second does not belong to $\|\Theta_1\|^\kappa$. Hence, thanks to Theorem 23, we have $\Theta_1 \nsubseteq \Theta_2$ and $\Theta_2 \nsubseteq \Theta_1$.

Further, by the definability theorem in [4], given a complete play that belongs to the knowing-strategy denotation of one term but not that of the other, we can construct a program context that separates the two terms. For example, a context that is constructed from the first complete play in (6) is

\[
D[\cdot] = \text{[}\lambda g. \text{ifzero}(g) \text{then } \omega \text{ else } (g(0))\text{]},
\]

where the hole [•] is of type ((exp ⇒ exp[\[\]] ⇒ exp[\[\]]) ⇒ exp[\[\]]) ⇒ exp[\[\]]. The reader may wish to check that $D[\Theta_1] \downarrow 1$ but $D[\Theta_2] \downarrow 0$.

4.3. A definability result

We augment IA by a case construct

\[
\frac{\Gamma \vdash M : \exp \quad \Gamma \vdash M_i : \phi \text{ for each } 0 \leq i < l}{\Gamma \vdash \text{case}(M)[0 \mapsto M_0 \mid \cdots \mid l-1 \mapsto M_{l-1}] : \phi}
\]

where $\phi$ is either exp or com.

Theorem 27 (Definability). For any compactly innocent strategy-with-state $\sigma = \text{strat}(f)$ over an arena $A = (A_1 \Rightarrow \cdots \Rightarrow A_n \Rightarrow \phi)$ where $n \geq 0$, there is a finitary IA-term $M_\sigma$ such that $\|\vdash M_\sigma : A \|= \sigma$.

Proof. We use $(A_1, \ldots, A_n, \phi)$ as a shorthand for $A_1 \Rightarrow A_2 \Rightarrow \cdots \Rightarrow A_n \Rightarrow \phi$. There are three cases, corresponding to $\phi = \text{com}$, or loc, or exp where $\mathbb{N}_d = \{0, 1, \ldots, d-1\}$. We shall just consider the first two cases as the third is similar to the first.

Case 1: $A = (A_1, \ldots, A_n, \text{com})$. If $\text{dom}(f) = \emptyset$ then $M_\sigma = \lambda f_1^{A_1} \cdots f_n^{A_n} \cdot \omega$. If $f : \text{run} \mapsto \text{done}^S$ (for simplicity we shall consider a simple state $S = (l, 2)$, where $(l, k)$ is our shorthand for the function $\{D, (l, k)\}$), then

\[
M_\sigma = \lambda f_1^{A_1} \cdots f_n^{A_n} \cdot \text{new } x := 2 \text{ in skip}.
\]

Suppose $f$ maps run, the initial move of $A$, to an initial question of $A_i = (C_1, \ldots, C_m, \phi')$ where $m \geq 0$ and each $C_j = (D_{j,1}, \ldots, D_{j,r_j}, \phi_j)$ where $r_j \geq 0$. There are three subcases corresponding to $\phi' = \text{exp}$ or com or loc.

(a) Suppose $A_i = (C_1, \ldots, C_m, \text{com})$, so that $f : \text{run} \mapsto \text{run}^{(l, 2)}$. Let $[C_j, t]$ be an initial move of $C_j$.

For each $1 \leq j \leq m$ and for each $0 \leq k < d$, consider the set of P-views $p$ such that $\text{run}^{(l, 2)} \{D_{j,1}^{(l, k)}\} \in \sigma$. This set, which defines a view function, determines an innocent strategy-with-state $\sigma_{jk}$ of the arena $(A_1, \ldots, A_n, D_{j,1}, \ldots, D_{j,r_j}, \text{com})$. As the corresponding view function has a smaller size than $f$, by the induction hypothesis, we have that $M_{\sigma_{jk}}$ is

\[
\lambda f_1^{D_{j,1}} \cdots g_1^{D_{j,r_j}} \cdot \text{new } x, y := a_{jk}, b_{jk} \text{ in } T_{jk}.
\]

(For simplicity we assume that only one new location, namely, $l'$, is introduced.) Similarly, for each $0 \leq k < d$, by considering the set of P-views $p$ such that $\text{run}^{(l, 2)} \text{done}^{(l, k)}$.
Case 2: A = (A₁, . . . , Aₙ, loc). σ decomposes into compactly innocent strategies-with-state σ′ : (A₁, . . . , Aₙ, exp), and for each 0 ≤ i < d, σ′′ : (A₁, . . . , Aₙ, com). By the induction hypothesis, we have Mσ′ = λ fJ.new x := 2 in fJ(θ₁ · · · θₘ) ; case(!x)[k → αₖ] where αₖ is new z := dₖ in x := cₖ ; Uₖ and θ₂ is

λfJ.new x := 2 in fJ(θ₁ · · · θₘ) ; case(!x)[k → αₖ] where αₖ is new z := dₖ in x := cₖ ; Uₖ and θ₃ is

Thus Mσ is

λfJ.new x := 2 in fJ(θ₁ · · · θₘ) ; case(!x)[k → αₖ] where αₖ is new w := eⱼ̅, hⱼ̅ in x := aⱼ̅ ; Tⱼ̅.

(b) A₁ = (C₁, . . . , Cₘ, exp). Let (Aᵢ be the initial move of Aᵢ. For each 0 ≤ j, k < d, by considering the set of P-views p such that run(₁,₂) j₁ k p ∈ σ where “j” is a numeric answer to (Aᵢ, by the induction hypothesis, we obtain λfJ.new x, w := eⱼ̅, hⱼ̅ in x := aⱼ̅ ; Tⱼ̅.

Thus Mσ is

λfJ.new x := 2 in fJ(θ₁ · · · θₘ) ; case(!x)[k → αₖ] where αₖ is new w := eⱼ̅, hⱼ̅ in x := aⱼ̅ ; Tⱼ̅.

where each βⱼ is new w := eⱼ̅, hⱼ̅ in x := aⱼ̅ ; Tⱼ̅ and θⱼ is as before.

(c) A₁ = (C₁, . . . , Cₘ, loc). There are two subcases. If f : run → write(l)(1,2) then Mσ is

λfJ.new x := 2 in fJ(θ₁ · · · θₘ) ; case(!x)[k → αₖ] where each θⱼ and αⱼ are as before. If f : run → read₁ then Mσ is

λfJ.new x := 2 in fJ(θ₁ · · · θₘ) ; case(!x)[k → αₖ] where each θⱼ and αⱼ are as before.

Case 2: A = (A₁, . . . , Aₙ, com). We remark that the Mσ in the theorem can be chosen to be λfJ.c where c has a fixed shape, which is called a finite canonical form. We define FCF[Γ], the set of finite canonical forms whose free variables are in Γ, as follows. Set Γ = {f₁ : A₁, . . . , fₙ : Aₙ}.

• For any value v, we have v ∈ FCF[Γ].
• If P₁,n ∈ FCF[Γ], y₁j₁, . . . , yₘrᵢ], and Pₘ,n ∈ FCF[Γ] then

new x := k in case(f₁ θ₁ · · · θₙ)[m → case(!x)[n → Pₙ,m]] ∈ FCF[Γ]

where A₁ = (D₁, . . . , Dᵢ, β), Dᵢ = (Dᵢ₁, . . . , Dᵢrᵢ, βᵢ), and each

θᵢ = λfJ.case(!x)[n → P₁,n].

(Note that for simplicity, we consider the simple case here where there is just one local variable x.)

5. DPDA characterization and decidability results

In this section we explain how legal positions over finite arenas can be encoded as words of a finite alphabet. Using the encoding, we show that the compactly innocent
strategy-with-state (or equivalently, view function) denoting any third-order finitary IA term determines a deterministic pushdown automaton (DPDA) that recognizes exactly the complete plays of the knowing-strategy semantics of the term. Since such plays characterize observational equivalence, and there is an algorithm for deciding whether any two DPDA's recognize the same language, we obtain a procedure for deciding the observational equivalence for that fragment of IA.

5.1. Encoding pointers by view offset

We use a simple encoding scheme, called view offset, for representing legal positions as words of an alphabet. For any even-length legal position \( sm \), we encode the pointer of the P-move \( m \) to \( q \) (say), which must appear in \( \langle s \rangle \) by Visibility, by a numeric offset \( n \) that is defined to be the number of pending O-questions that occur after \( q \) in the P-view \( \langle s \rangle \). Similarly the pointer of an O-move can be given in terms of an offset relative to the O-view at that point. The move \( m \) with view offset \( n \) may then be encoded by (say) the word \( o^n m \)—the prefix \( o^n = o \cdots o \) is taken to be the unary numeral \( n \). A legal position can thus be represented as a concatenation of such words. Note that the view-offset scheme is quite general: it works for legal positions of all finite orders.

The encoding scheme may be simplified in several ways. First there is no need to represent explicitly the pointer of an answer-move: by decreeing that such a pointer is always to the last pending question, we can use a pushdown stack to verify Well-Bracketing. Secondly, in the third-order case, a further simplification is possible, thanks to the following lemma.

Lemma 28. Fix a third-order arena.

(i) Let \( s \) be an even-length legal position. If there is a pending first-order P-question in \( s \), then there is a unique pending first-order P-question (namely the second element) in \( \downarrow s \uparrow \), and that question must also appear in \( \langle s \rangle \).

(ii) Further if another legal position \( t \) has the same P-view as \( s \), then \( \downarrow s \uparrow \) and \( \downarrow t \uparrow \) contain the same unique pending first-order question.

We may assume that the \( s \) in (i) and the \( t \) in (ii) are legal positions that do not necessarily satisfy State Change. In particular, we may take \( t \) in (ii) to be \( \langle s \rangle \).

Proof. We prove (i) and (ii) simultaneously. Let \( m \) be the last move of \( s \), and so, also of \( t \) because \( \langle s \rangle = \langle t \rangle \). Since \( s \) has a pending first-order question, \( m \) cannot be an answer to the initial move of \( s \) because of Well-Bracketing. Suppose \( m \) is a first-order P-question, we have \( \downarrow s \downarrow = \downarrow t \downarrow = \downarrow \downarrow 0m \) where \( \downarrow 0 \) is the initial move, and \( m \) is the unique pending first-order P-question, and we are done. Suppose \( m \) is either a second-order P-answer or a third-order P-question. Let \( \downarrow 2 \) be the second-order O-question that explicitly justifies \( m \) in \( s \). By Visibility, \( \downarrow 2 \) appears in \( \langle s \rangle \) and so also in \( \langle t \rangle \); for the same reason, the first-order P-question \( \downarrow 1 \) which explicitly justifies \( \downarrow 2 \) in \( s \) also appears in \( \langle t \rangle \). Now \( s_{\downarrow 2} \) and \( t_{\downarrow 2} \) fit the shape of the legal position in Observation 1(i), and so, both \( \downarrow s_{\downarrow 2} \downarrow \) and \( \downarrow t_{\downarrow 2} \downarrow \) have a unique pending first-order P-question, namely, \( \downarrow 1 \). Since \( \downarrow s \downarrow = \downarrow s_{\downarrow 2} \downarrow m \) and \( \downarrow t \downarrow = \downarrow t_{\downarrow 2} \downarrow m \), we are done. □
Remark 29. An essential task in the representation of legal positions (of an innocent strategy) is the verification of O-Visibility. To do that using a state machine, it would seem necessary, in general, to countenance infinitely many control-states, as the legal positions of even a compactly innocent strategy can trace out infinitely many O-views. Lemma 28 tells us that, up to third order, O-views can safely be ignored in the representation, as O-Visibility may be verified using only P-views.

By the lemma (applied to \( t = \lambda \sigma \uparrow s \downarrow \)), the pointer of an O-question (which is necessarily second-order, as the zeroth-order O-question has no pointer by definition) is to the unique first-order pending P-question that appears in the O-view, and that P-question is guaranteed to appear in the P-view at that point. Therefore there is no need to consider O-views at all. Indeed because the justifying P-question is unique, there is no need to represent the pointer of any second-order question. There is also no need to represent the pointer of any first-order question as it must be to the opening question. Thus only the pointers of third-order P-questions (in a legal position of a third-order arena) need to be coded explicitly.

Finally in the case of a third-order compact innocent strategy \( \text{strat}(f) \) generated by some finite view function \( f \), yet another simplification is possible. Instead of coding a third-order P-question with an offset-\( i \) pointer as \( o' \lambda \), we can simply introduce a new symbol as a name for the word \( o' \lambda \) (—this is quite harmless as the view offsets are bounded above by the length of the longest P-view in the domain of \( f \).

**Complete plays of third-order knowing strategies are not regular**

Using the view-offset encoding, we can regard the set of complete plays of a knowing strategy denoting a third-order finitary IA term as a set of words. In general such sets are not regular. Our proof uses a theorem due independently to Myhill [23] and Nerode [24]. (It is also possible to obtain a proof using the Pumping lemma, but the argument is somewhat more complicated.) For any \( L \subseteq \Sigma^* \) where \( \Sigma \) is a given finite alphabet, we say that strings \( x \) and \( y \) are \( L \)-indistinguishable, written \( x \equiv_L y \), just in the case where for every string \( z \), \( xz \in L \) iff \( yz \in L \). It is easy to see that \( \equiv_L \) is an equivalence relation. The Myhill–Nerode Theorem (see e.g. [15] for a careful account) says that \( L \) is regular iff \( \equiv_L \) has finitely many equivalence classes; moreover the number of equivalence classes is the size of the smallest deterministic finite automaton that accepts \( L \).

**Lemma 30.** In general the set of complete plays of the knowing-strategy denotation of a third-order finitary IA term is not regular.

**Proof.** Set \( A = (o \Rightarrow o \Rightarrow o \Rightarrow o) \) where \( o \) is any arena that has only one question and one answer (such as \( \text{com} \)). Take \( \sigma \) to be \( \lambda \text{F.F}(\lambda x.x) \|^e \) which is an innocent strategy over \( A \). We write the four questions of \( A \) as \( [0], [1], [2] \) and \( [3] \), and their corresponding answers as \( [0], [1], [2], [3] \) respectively; the (type-theoretic) order of each move is annotated as a subscript, and we take \( [3] \) to mean “the third-order P-question with view offset 0”. In the following we consider complete plays of \( \sigma \) as words over the alphabet \( \Sigma = \[0, [1], [2], [3], [0], [1], [2], [3]\] \)—let \( L \) be the set of such plays. For each \( i \geq 0 \), we write \( l_i = [0, [1], [2], [3] \cdots [i] [2i] \phantom{} \text{2i questions} \) and \( r_i = [3] [2] \cdots [1] [2i] \phantom{} \text{2i answers} \). Now for any \( i, j \geq 0 \), if \( i \neq j \) then
l_i r_i \in L \text{ but } l_j r_i \notin L. \text{ Hence the relation } \equiv_L \text{ has infinitely many equivalence classes. It follows from the Myhill–Nerode Theorem that } L \text{ is not regular. } \Box

5.2. Deterministic pushdown automata: a review

We give a quick review of the basic definitions. A deterministic pushdown automaton (DPDA) is specified by a 6-tuple \( P = \langle Q, \text{init}, \Sigma, \Gamma, \bot, \delta \rangle \) where \( Q \) is a finite set of (control) states, \( \text{init} \in Q \) is the initial state, \( \Sigma \) is a finite set of input symbols, \( \Gamma \) is a finite set of stack symbols, \( \bot \in \Gamma \) is an auxiliary symbol indicating bottom-of-stack (initially the DPDA stack consists of one instance of \( \bot \) and nothing else), and \( \delta \) is a finite set of transitions, each of the form

\[
p, Z \xrightarrow{a} q, \alpha
\]

where \( p, q \) are states, \( a \in \Sigma \cup \{\epsilon\} \), \( Z \in \Gamma \), and \( \alpha \) is a finite sequence of stack symbols, subject to the following restriction:

1. If \( Z = \bot \) then \( \alpha = \bot \beta \) where \( \beta \in (\Gamma \setminus \{\bot\})^* \); if \( Z \neq \bot \) then \( \alpha \in (\Gamma \setminus \{\bot\})^* \).
2. If \( p, Z \xrightarrow{a} q, \alpha \) and \( p, Z \xrightarrow{a} r, \beta \) and \( a \in \Sigma \cup \{\epsilon\} \) then \( q = r \) and \( \alpha = \beta \).
3. If \( p, Z \xrightarrow{\epsilon} q, \alpha \) and \( p, Z \xrightarrow{a} r, \beta \) then \( a = \epsilon \).

Condition (1) ensures that the bottom-of-stack symbol is always present at the bottom of the stack, but nowhere else. Condition (2) imposes determinism, and Condition (3) enforces disjointness between \( a \)-transitions where \( a \in \Sigma \) and \( \epsilon \)-transitions. A DPDA is said to be real-time just in the case where it has no \( \epsilon \)-transitions.

A configuration of a DPDA is a pair \((q, \alpha)\) where \( q \) is a state, and \( \alpha \in \Gamma^* \), which has the form \( \bot \beta \) where \( \beta \in (\Gamma \setminus \{\bot\})^* \), is the sequence of symbols on the stack. The transitions of a configuration are defined by the following prefix rule:

If \( p, Z \xrightarrow{a} q, \alpha \) is a transition in \( \delta \) then for any \( \gamma \in \Gamma^* \), we have \( p, \gamma Z \xrightarrow{a} q, \gamma \alpha \)

(which we read as: by consuming \( a \) from the input, and replacing \( Z \) on top of the stack by \( \alpha \), we can go from state \( p \) to state \( q \)). The transition relation \( \xrightarrow{a} \) between configurations where \( a \in \Sigma \) is then standardly extended to words \( \xrightarrow{w} \) where \( w \in \Sigma^* \). Acceptance is by empty stack. Formally the language recognized by the DPDA \( P \), written \( L(P) \), is defined to be

\[
L(P) \overset{\text{def}}{=} \{ w \in \Sigma^* : \exists q \in Q, \text{init}, \bot \xrightarrow{w} q, \bot \}.
\]

That is, \( L(P) \) is the set of inputs that \( P \) can consume and at the same time empty its stack.

The DPDA Equivalence Problem was first posed in 1966 by Ginsberg and Greibach [10]:

“Is there an effective procedure for deciding whether any two DPDA recognize the same language?”

Restricted to the real-time case, the problem was solved positively by Oyamaguchi et al. in 1980 [29]. The general decidability problem was only solved recently, also positively, by Sénizergues [35] (solution announced in 1997). A simpler proof [37] of the decidability result, and a primitive recursive complexity bound [38], were subsequently obtained by Stirling.
5.3. Construction of the DPDA $P_f$

Let $f$ be a finite view function that generates an innocent strategy-with-state over a third-order arena $A$. We shall construct a real-time DPDA $P_f$ that recognizes the complete plays of the knowing strategy $\text{erase}(\text{strat}(f))$. Formally the real-time DPDA

$$P_f = \langle Q_f, \text{init}, \Sigma_f, \Gamma_f, \bot, \delta_f \rangle$$

is defined as follows:

- The alphabet $\Sigma_f$ consists of all moves of $A$ less third-order questions, and for each third-order question (with view-offset $i$ that appears in the range of $f$, a symbol representing the justified question.
- The state-set $Q_f \overset{\text{def}}{=} \{\text{init}\} \cup \text{dom}(f) \cup \{p m^S : f(p) = m^S\}$. That is, other than the initial state, every state is either an (odd-length) P-view in the domain of $f$, or an (even-length) P-view that is a section of the graph of $f$.
- The set $\Sigma_f$ of stack symbols consists of the $f$-reachable questions of $A$. An $f$-reachable question of $A$ is either an O-question $\llbracket S \rrbracket$ whereby $p \llbracket S \rrbracket \in \text{dom}(f)$ for some $p$, or it is a P-view of the form $p^S$ such that $f$ maps $p$ to $^S$. Clearly $\Gamma_f$ is a finite set.
- The transitions of $P_f$, which are presented in Fig. 6, are organized into four types:

<table>
<thead>
<tr>
<th>Transition Type</th>
<th>State Before (P-views)</th>
<th>Input Symbol</th>
<th>State After (P-views)</th>
<th>Stack Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>odd-length</td>
<td>P question</td>
<td>even-length</td>
<td>push</td>
</tr>
<tr>
<td>2</td>
<td>odd-length</td>
<td>P answer</td>
<td>even-length</td>
<td>pop</td>
</tr>
<tr>
<td>3</td>
<td>even-length (or init)</td>
<td>O question</td>
<td>odd-length</td>
<td>pop</td>
</tr>
<tr>
<td>4</td>
<td>even-length</td>
<td>O answer</td>
<td>odd-length</td>
<td></td>
</tr>
</tbody>
</table>

For example a transition $p, Z \xrightarrow{a} q, \alpha$ is type-1 just in the case where $p$ is an odd-length P-view and $a$ is a P-question ($q$ is an even-length P-view, and $\alpha = Z Z'$ for some $Z'$, corresponding to a push action on the stack). For ease of reading, we present each transition $p, Z \xrightarrow{a} q, \alpha$ as a block of the form

```
p Z a q α
```

in Fig. 6; where appropriate, we shall write a move of order $k$ as $m_k$ to aid reading.

Inspecting the four types of transitions in turn, it is straightforward to verify that the automaton $P_f$ thus defined is a real-time DPDA.

5.4. Decidability theorems

The main technical result of this section is the following theorem:

**Theorem 31.** For any compactly innocent strategy-with-state $\text{strat}(f)$ of a third-order arena, we have

$$L(P_f) = \text{cplays}(\text{erase}(\text{strat}(f))).$$

Hence, by Theorem 25, the (set of complete plays of the) knowing-strategy semantics of a third-order finitary IA term is deterministic context-free.
Type-1: For each odd-length P-view $p$, and each P-question $q$ such that $f : p \mapsto q$

\[
\begin{array}{|c|c|c|c|c|}
\hline
p & Z & \{ & p^{q} & Z \text{"}p^{q}\text{"} \\
\hline
\end{array}
\]

Type-2: For each odd-length P-view $p$, and each P-answer $q$, such that $f : p \mapsto q^{q}$ which is explicitly justified by $q^{q}$ in $p$

\[
\begin{array}{|c|c|c|c|c|}
\hline
p & q^{q} & \{ & p^{q} & \epsilon \\
\hline
\end{array}
\]

Type-3.1: For each initial question $i_0$

\[
\begin{array}{|c|c|c|c|c|}
\hline
\text{init} & \bot & i_0 & i_0 & \bot \text{ } i_0 \\
\hline
\end{array}
\]

Type-3.2: For each odd-length P-view of the form $p q^{q} w$ such that $f : p q^{q} w \mapsto b^{q}$ where $q^{q}$ is the unique first-order P-question that appears in $\cup_{p} q^{q} w b^{q}$ (see Lemma 28(ii)), and for each second-order O-question $i_2$ such that $(i_1 \vdash_A i_2)$

\[
\begin{array}{|c|c|c|c|c|c|}
\hline
p q^{q} w b^{q} & Z & i_2 & p q^{q} w b^{q} i_2 & Z q^{q} \\
\hline
\end{array}
\]

Type-4.1: For each odd-length P-view $p$ such that $f : p \mapsto (q^q, \text{for each O-answer})$ such that $(\vdash_A)$

\[
\begin{array}{|c|c|c|c|}
\hline
p^{q} & \text{"}p^{q}\text{"} & (q^q) & \epsilon \\
\hline
\end{array}
\]

Type-4.2: For each odd-length P-view of the form $p q^{q} w$ where $q^{q}$ is a first-order P-question such that $f : p q^{q} w \mapsto q^{q}$, which is explicitly justified by $q^{q}$, and for each first-order O-answer $i_1$ such that $(i_1 \vdash_A)$

\[
\begin{array}{|c|c|c|c|}
\hline
p q^{q} w & q^{q} & \text{\"}p^{q}\text{\"} & \epsilon \\
\hline
\end{array}
\]

Type-4.3: For each odd-length P-view of the form $p q^{q} w$ where $q^{q}$ is a first-order P-question such that $f : p q^{q} w \mapsto q^{q}$, and for $f : p q^{q} u \mapsto q^{q}$ and each third-order O-answer $i_3$ such that $(i_3 \vdash_A)$

\[
\begin{array}{|c|c|c|c|}
\hline
p q^{q} w & q^{q} & \text{\"}p^{q}\text{\"} & \epsilon \\
\hline
\end{array}
\]

Fig. 6. Transitions of the DPDA $P_f$.

We first explain the idea behind Theorem 31. The tasks of verifying that an input word is a (complete) play in erase strat $(f)$ have been built into the transitions of the DPDA $P_f$. At any given point during a computation of $P_f$, the state that is reached is precisely the P-view of that prefix, say, of the input word which has been read thus far; and the stack at that point contains the subsequence of (pending) $f$-reachable questions in $s$. The stack actions ensure that the computation admits only moves that observe Well-Bracketing:
• If the prefix $s$ is odd-length, only the P-move $m^S = f(\langle s \rangle)$ may fire—see transitions of Types 1 and 2.
• In the case where $s$ is even-length, all legal O-moves $m$ (i.e. those $m$ such that $s^m S$ satisfies O-visibility, Well-Bracketing, and State Change, and so, $\langle s^m S \rangle \in \text{dom}(f)$) may fire—see transitions of Types 3 and 4.

For a simple example, we consider the DPDA of a state-less view function as follows.

**Example 32.** Take the term $\lambda F F(\lambda x . x)$ of type $((o \Rightarrow o) \Rightarrow o) \Rightarrow o$, first considered in the proof of Lemma 30 (we use the notation therein). The transitions (less the stack actions) of the associated DPDA are shown in Fig. 7. Consider the Type-4.3 transition marked $\dagger$ which we give in full as follows:

$$\begin{array}{c}
\begin{array}{c}
[0 \, (1, 2) \, 3, 2, 3], \quad \text{“} [0 \, (1, 2) \, 3] \xrightarrow{3} [0 \, (1, 2) \, 3], \epsilon.
\end{array}
\end{array}$$

Note that the P-question to which the input symbol $)$ is an answer does not appear in the state (i.e. P-view) before the transition (it occurs in the segment of the play between $1$ and $2$ which does not appear in the P-view); we rely on the information contained in the “symbol” on the top of stack—“$[0 \, (1, 2) \, 3]$” which is the last pending $f$-reachable question—to work out the P-view after the move. This is why the stack symbols are $f$-reachable questions, rather than simply questions. Indeed this alone is why we need
pushdown automata (as opposed to just deterministic finite automata) to characterize third-order strategies.

We first make a few notational remarks.

(i) For any legal position \( s \in \text{strat}(f) \), we define the subsequence \( \text{rq} s \) of (pending) \( f \)-reachable questions of \( s \) as follows. First set \( t \) to be the subsequence of pending questions of \( s \); \( \text{rq} s \) is defined to be the sequence which is obtained from \( t \) by prefixing it by the bottom-of-stack symbol \( \bot \), and by replacing every occurrence \( q \) of a pending \( P \)-question in \( t \) by \( \lfloor s \leq q \rfloor \), where \( s \leq q \) is the prefix of \( s \) that terminates at \( q \).

(ii) By definition of \( P_f \), the input symbol \( m \) of each transition \( p, Z \overset{m}{\rightarrow} p', \alpha \) is a move without state. Nevertheless each such \( m \) can be given a canonical state, namely, the state of the last move-with-state of the \( P \)-view \( p' \). Suppose \( \text{init} \), \( \bot \overset{s}{\rightarrow} p, \alpha \); we write \( \hat{s} \) to be the sequence that is obtained from \( s \) by annotating each move with its canonical state.

**Lemma 33.** Given a finite view function \( f \) so that \( \sigma = \text{strat}(f) \) is an innocent strategy-with-state:

(i) If \( s \) is a legal position in \( \sigma \) then \( \text{init}, \bot \overset{s^-}{\rightarrow} r s \), where \( s^- = \text{erase} s \).

(ii) If \( \text{init}, \bot \overset{s}{\rightarrow} p, \alpha \) for some \( p \) and \( \alpha \) then

1. \( \hat{s} \) is a legal position in \( \sigma \),
2. \( r \hat{s} = p \),
3. \( \alpha = \text{rq} s \).

**Proof.** We prove (i) by induction on the length of \( s \in \sigma \). For an illustration, we shall consider the inductive case of an even-length \( s \in \sigma \), where \( s \in \sigma \) is explicitly justified by \( \gamma_{1,1} \). Let \( b_{S_1} \) be the last move of \( s \). By condition (SC-O), we have \( T = S_0[S_1] \). By the induction hypothesis, we have

\[
\text{init}, \bot \overset{s^-}{\rightarrow} r s, \text{rq} s
\]

where \( s^- = \text{erase} s \). By Lemma 28(i), \( r_{1,1} \), which is the unique pending first-order question in \( r s \), appears in \( r s \). Hence we may write the even-length \( r s \) as \( p r_{1,1} [b_{S_1}] \) where \( f : p r_{1,1} [b_{S_1}] \overset{s^-}{\rightarrow} b_{S_1} \). By Lemma 28(ii), \( r s \) and \( r b_{S_1} \) have the same pending first-order question \( r_{1,1} \), the unique such in both case. Thus, the following instance of Type 3.2 rule:

\[
p r_{1,1} [b_{S_1}], Z \overset{1}{\rightarrow} p r_{1,1} [S_0[S_1]], Z [S_0[S_1] 2]
\]

is applicable and, hence, we have

\[
\text{init}, \bot \overset{s'}{\rightarrow} r s [S_0[S_1] 2], \text{rq} (s [S_0[S_1] 2])
\]

where \( s' = \text{erase} (s [S_0[S_1] 2]) \) and \( r s [S_0[S_1] 2] = p r_{1,1} [S_0[S_1] 2] \) as required.

---

4 We shall show in Lemma 33 that \( \text{rq} s \) is the sequence of symbols on the stack (at an appropriate point in the computation of \( P_f \)), and by definition such sequences always begin with the symbol \( \bot \).
We prove (ii) by induction on the length of the sequence \( s \). The base case of \( |s| = 1 \) is trivial. For the inductive cases, suppose
\[
\text{init}, \epsilon \xrightarrow{s} p, \alpha \xrightarrow{m} p', \alpha'.
\]
The last transition is one of four possible types. We consider the important cases, namely, Types 3.2 and 4.3, for illustration; the rest are comparatively straightforward.

**Type-3.2 transition.** The input symbol \( m \) is a second-order O-question \([2]\). By the induction hypothesis (2), we have \( \hat{\gamma} \hat{s}^1 = p_0 \hat{\ell}_1^0 w b \hat{s}_1 \), for some odd-length P-view \( p_0 \), where \( \ell_1^0 \) is the unique first-order P-question that appears in \( \hat{\ell}_1^0 w b \hat{s}_1 \). Thus \( \hat{s}_1 \), whereby the pointer of \( \hat{\ell}_1^1 \) is to \( \hat{\ell}_1^0 \), satisfies Visibility, and so is a legal position in \( \sigma \), which gives (1), and \( \hat{\gamma} \hat{s}_1 \hat{\ell}_2^1 = p_0 \hat{\ell}_1^0 \hat{\ell}_2^1 \hat{s}_1 \) = \( p' \), which is (2) as required. Finally (3) is an immediate consequence of the induction hypothesis (3).

**Type-4.3 transition.** We have that \( m \) is a third-order O-answer \( )_3 \). By the induction hypotheses (1) and (2), \( \hat{s} \) is an even-length legal position in \( \sigma \), and \( \hat{\gamma} \hat{s}^1 = p \) which has the form \( p_0 \hat{\ell}_1^0 w \hat{\ell}_2^1 \). Thus the last move-with-state of \( \hat{s} \) has state \( S_1 \). Since \( p' = p_0 \hat{\ell}_1^0 u \hat{s}_2^3 \), \( m \) has state \( S_0[S_1] \); thus \( \hat{s}_2^3 \) \( S_0[S_1] \) satisfies State Change. The stack \( \alpha \) has the symbol "\( p_0 \hat{\ell}_1^0 u \hat{s}_3^0 \)" on top. By the induction hypothesis (3), "\( \ell_3^0 \)" is the last pending P-question. Since \( (3 \downarrow 3 \) ), we have that \( \hat{s}_2^3 \) \( S_0[S_1] \) satisfies Well-Bracketing, and hence\(^5\) also Visibility. Therefore we have \( \hat{s}_2^3 = \hat{s}_2^3 \) \( S_0[S_1] \) \( \in \sigma \). The P-view of \( \hat{s}_2^3 \) is \( \hat{\gamma} \hat{s}_2^3 \) \( S_0[S_1] \) \( \gamma \) where \( \gamma \) is the prefix of \( \hat{s} \) that terminates at \( \hat{s}_2^3 \). By the induction hypothesis (3), \( \hat{\gamma} \hat{s}_2^3 \gamma = p_0 \hat{\ell}_1^0 u \hat{s}_3^0 \), the top-of-stack symbol. Thus we have \( \hat{\gamma} \hat{s}_2^3 \gamma = p_0 \hat{\ell}_1^0 u \hat{s}_3^0 \), which is \( p' \) as required. Finally to show (3), we observe that \( \alpha' \) is obtained from \( \alpha \) by popping the top of stack symbol, which is exactly right. □

**Remark 34.** It may have occurred to the reader who has studied the preceding proof that the prefix \( p \) of an \( f \)-reachable P-question, \( p \) \( \ell^3 \), is redundant in the case where the question is first-order. We have included it in the definition so that first- and third-order P-questions are treated uniformly when presenting the transition rules of \( P_f \).

Take a complete play \( s \in \text{strat}(f) \). By (i) of the lemma, we have \( \text{erase} s \in \mathcal{L}(P_f) \) because \( \text{erase} s = \bot \) and acceptance is by empty stack. Since
\[
\text{erase (cplays (strat(f))))} = \text{cplays (erase (strat(f)))}
\]
we have \( \text{cplays (erase (strat(f)))} \subseteq \mathcal{L}(P_f) \). The opposite inclusion follows from (ii) of the lemma. Hence, Theorem 31 is an immediate consequence.

**Corollary 35.** For the same \( f \) as before, the following sets are all real-time deterministic context-free.

1. **Plays of the knowing strategy** \( \text{erase (strat(f))} \).
2. **Plays of the innocent strategy-with-state** \( \text{strat}(f) \).
3. **Complete plays of** \( \text{strat}(f) \).

---

\(^5\) Recall that an O-answer to the last pending question always satisfies the Visibility condition.
Proof. By straightforward modifications of $P_f$. □

As a corollary of Theorem 31, we have the main result of the paper:

**Theorem 36 (Decidability).** (i) Observational equivalence of third-order finitary IA (built up from finite base types) is decidable.

(ii) We refer to the sublanguage of IA without the bad-location construct $\text{mkloc}$ as $\text{IA}^-$. Observational equivalence of third-order finitary $\text{IA}^-$ is decidable.

**Proof.** (i) Take third-order IA-terms $M_1$ and $M_2$ such that $\Gamma \vdash M_i : A$ is provable for some $\Gamma$ and $A$. By Theorem 18, their respective innocent strategy-with-state denotations $\llbracket \Gamma \vdash M_i : A \rrbracket$ (as finite view functions) are computable. By Theorem 31 and since it is decidable whether any two real-time DPDA$s recognize the same language, we have

"$c\text{plays} (\text{erase} \llbracket \Gamma \vdash M_1 : A \rrbracket) = c\text{plays} (\text{erase} \llbracket \Gamma \vdash M_2 : A \rrbracket)$" is decidable. Hence, by Theorem 25, we have that

"$c\text{plays} \llbracket \Gamma \vdash M_1 : A \rrbracket^\kappa = c\text{plays} \llbracket \Gamma \vdash M_2 : A \rrbracket^\kappa$"

is decidable. Finally, by Theorem 23, we have that "$M_1 \approx M_2$" is decidable, as required.

(ii) is an immediate consequence of the result, due to McCusker [20], that adding $\text{mkloc}$ to $\text{IA}^-$ is conservative for observational equivalence (but not for observational preorder). □

Note that since DPDA INCLUSION ("Given DPDA$s M_1$ and $M_2$, is it the case that $L(M_1) \subseteq L(M_2)$?") is undecidable, the preceding argument cannot be used to prove that observational preorder of third-order IA is decidable.

**Complexity of DPDA EQUIVALENCE**

It is not clear what the complexity of deciding on the observational equivalence of third-order finitary IA is. Our decision procedure is dominated by the computation of the view-function denotation of the two IA terms being compared, and by the procedure that decides equivalence of the two derived real-time DPDA$s$. The former is in essence a normalization procedure (which is essentially linear head reduction in the sense of [6]) for a sublanguage consisting of appropriate canonical forms of third-order finitary IA; the complexity of the latter is a topic of current research. Stirling has suggested that his algorithm in [39] can be modified to give an elementary upper bound for deciding on the equivalence of real-time, strict (i.e. acceptance by empty stack) DPDA$s$.

**Observational equivalence of fourth-order IA is undecidable**

Can the algorithmic representation be extended to fourth and higher orders? It turns out that the third order is the limit, and the best that we can do. By demonstrating that there are fourth-order terms whose complete plays correspond to runs of Turing-complete machine models, Murawski [22] has shown that observational equivalence is undecidable for the fourth-order fragment.
Deciding observational equivalence of second-order IA

Using the view function approach, we obtain a new proof of a result due to Ghica and McCusker in [9]:

**Theorem 37.** The set of complete plays of the knowing-strategy denotation of a second-order finitary IA term is regular.

**Proof.** Let \( f \) be the view function determined by a second-order finitary IA term. We consider the various types of transitions of the DPDA \( P_f \) as set out in Fig. 6. Type-4.3 transitions are not applicable as they are for the case where the input symbol is a third-order answer. Consequently the pushdown stack is redundant, and the resultant transition function cuts down to that of a deterministic finite automaton. (See Example 32 for an explanation of the rôle played by the pushdown stack in the DPDA determined by a third-order term.) \( \square \)

Ghica and McCusker’s result in [9] is for a version of second-order IA that has an iteration construct in the form of while-loops. We believe that the view function approach can be used to prove this stronger form of the result. The idea is that the states of the finite automaton should be regular sets of P-views. IA terms-in-contexts will be compiled to (finite) evaluation graphs, which will be shown to be the generators of the required sets of complete plays. Details of this approach will be presented elsewhere.

6. Second-order IA with recursion is undecidable

Ghica and McCusker have shown in [9] that the regular-expression representation of the knowing-strategy semantics of IA, and hence the decidability of observational equivalence, applies to second-order finitary IA augmented by iteration in the form of while-loops. Is observational equivalence still decidable for the same fragment augmented by the fixpoint operator (so that one can compute fixpoints of terms of order 2)? The answer is no.

**Theorem 38 (Undecidability).** Assuming that base types are finite, observational preorder of second-order IA with full recursion (thus admitting recursively defined first-order functions) is undecidable.

The proof is by an idea due to Jones and Muchnick in [14]. Fix a finite alphabet \( \Sigma \), and consider a programming system called Queue, which has a single memory cell \( z \) that can store a symbol from \( \Sigma \), a queue (unbounded, first-in–first-out) data structure, and four instructions as follows: let \( a \in \Sigma \) and \( L \geq 0 \) be a label:

1. **enqueue** \( a \): write the symbol \( a \in \Sigma \) onto the right end of the queue.
2. **dequeue** \( z \): if the queue is empty, halt; otherwise remove the leftmost symbol from the queue and write it to \( z \).
3. **if** \( z = a \) **goto** \( L \).
4. **halt**.

A Queue program is a finite sequence of the form

\[
1 : I_1, 2 : I_2, \ldots, m : I_m
\]
where each $I_i$ is an instruction. By simulating Post’s Tag Systems [21] in Queue, the problem

**QUEUE-HALTING:** Given a Queue program, will it halt eventually?

can be shown to be undecidable.

**Lemma 39.** Given a Queue program $P$, there is an IA program $\vdash M_P : \text{com}$ (closed term of type $\text{com}$, which belongs to the fragment of IA in question) that simulates $P$.

For simplicity, we represent $\Sigma$ by $\{1, \ldots, N\}$, for some appropriate $N$. Given a Queue program $P = 1 : I_1, 2 : I_2, \cdots, m : I_m$, we define an IA program of the fragment in question $M_P$ in Fig. 8.

For ease of reading we use $\text{letrec}$ instead of $\text{Y}(-)$, and avail ourselves of case construct and while-loop. We use three “global variables”: $\text{halt}$, $pc$ (program control), and $z$ (cell variable of the Queue system) in the simulation. For each $I_i$ in $P$, we define a corresponding $J_i$ as a branch of the case-construct, as set out in Fig. 9.

The idea is to simulate the queue data structure by a stack, namely, the call stack of the recursively defined $F$, together with a way of marking the position of the head of the queue. Each “enqueue $a$” instruction is simulated by a call of $F$, which declares three variables:

- $cur$, an assignable variable, is initialized to $a$, the symbol being added to the queue.
- $pre$ of type $\text{loc}$ is bound to the assignable variable $cur$ declared by the calling $F$.
- $w$ of type $\text{loc}$ is bound to the term $(\text{ifzero} ! pre \text{then} cur \text{else} w)$, which evaluates to the assignable variable $cur$ declared by the $F$ that was called at a recursive depth corresponding to the current head of the queue.

<table>
<thead>
<tr>
<th>$I_i$</th>
<th>$J_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{enqueue } n$</td>
<td>$pc := i + 1$; $F n \text{cur} (\text{ifzero} ! \text{pre} \text{then} cur \text{else} w)$</td>
</tr>
<tr>
<td>$\text{dequeue } z$</td>
<td>$\text{ifzero} ! \text{cur} \text{then} \text{halt} := 1 \text{else}$</td>
</tr>
<tr>
<td></td>
<td>$z := ! (\text{ifzero} ! \text{pre} \text{then} cur \text{else} w)$;</td>
</tr>
<tr>
<td></td>
<td>$(\text{ifzero} ! \text{pre} \text{then} cur \text{else} w) := 0$;</td>
</tr>
<tr>
<td></td>
<td>$pc := i + 1$</td>
</tr>
<tr>
<td>$\text{halt}$</td>
<td>$\text{halt} := 1$</td>
</tr>
<tr>
<td>$\text{if } z = n \text{ goto } m$</td>
<td>$\text{if } z = n \text{ then } pc := m \text{ else } pc := i + 1$</td>
</tr>
</tbody>
</table>
During the simulation, the position of the head of the queue is indicated by the highest stack frame whose local variable \textit{cur} contains 0, a special symbol not in \( \Sigma \). The \texttt{dequeue} instruction is simulated by assigning the contents of \textit{cur}, that is local to that call of \( F \) at a call-depth corresponding to the current head of the queue, to \( z \); followed by assigning 0 to that \textit{cur}, thus moving the head of the queue one frame up the stack. The call stack continues to grow until \texttt{halt} is set to 1, which has the effect of collapsing the call stack, as one recursive call of \( F \) after another exits.

Plainly the IA program \( M_F \) eventually halts if and only if it is observationally equivalent to the program \texttt{skip}. Undecidability of observational equivalence for this fragment of IA then follows from the undecidability of \texttt{QUEUE-HALTING}.

7. Conclusions and further directions

In this paper we have answered the following questions:

1. Does Ghica and McCusker’s regular-expression representation of second-order IA extend to the third-order fragment? 
   No: Lemma 30.
3. Is observational equivalence of third-order IA without recursion decidable? 
   Yes: Theorem 36.
4. Is observational preorder of second-order IA with full recursion decidable? 
   No: Theorem 38.

7.1. Some open problems

We intend to investigate applications of this work in Software Model Checking (see e.g. [27]). (Work in collaboration with Abramsky, Ghica and Murawski at Oxford, and with Lazić at Warwick, is already under way.) In addition to model-checking observational equivalence which is of basic importance, the same algorithmic representations of program meanings as are derived in this work can be put to use in verifying a wide range of program properties of IA and cognate programming languages, and in practice this is where the interesting applications are most likely to lie. Here we just mention a few specific open problems and directions of a more foundational nature.

1. A problem of practical importance is understanding the complexity of deciding the third-order finitary IA, and extracting a feasible algorithm tailored to the task at hand. We conjecture that there is an algorithm in elementary time.
2. We believe that our approach can be used to prove that third-order finitary IA augmented by \textit{iteration} (given by e.g. while-loops) is decidable (the idea is that a state of the corresponding DPDA is a regular set of P-views).

In a related direction, restricted to the second-order fragment, we intend to compare our approach (which is based on the algorithmic representation of view functions) with McCusker and Ghica’s (which compiles terms directly to regular expressions or finite automata by recursion on syntax).
(3) Our decidability results are for IA terms that are generated from finite base types. In the presence of infinite data types, the automata representations become infinite, thus losing their algorithmic properties. Similarly large finite data types are likely to make the automata infeasible. In the literature on process algebras, problems of this kind are addressed using symbolic transition graphs ([11,17]). We expect similar approaches to be fruitful for Algorithmic Game Semantics.

(4) In view of Theorem 38, a natural question to ask is whether observational equivalence of second-order call-by-value IA augmented by recursion is decidable: a positive answer would be good news from a model-checking perspective because of the direct relevance to procedural languages such as C.

(5) Finally it would be very interesting to find out whether the observational equivalence of (low orders of) Reduced ML is decidable. Reduced ML is a call-by-value PCF augmented by a statically scoped, dynamically allocated, mutable state, with an equality test for references—as studied in [30] (see also [36]). Our preferred approach is by Algorithmic Game Semantics, but to our knowledge, the prior (for this approach) problem of the existence of a fully abstract game semantics for Reduced ML is still open.

Acknowledgements

The author is grateful to Samson Abramsky and Dan Ghica for helpful discussions, and to Andrzej Murawski and Ranko Lazic for their careful comments on a draft of the paper. This work was partially supported by a UK EPSRC grant GR 888861/01 Algorithmic Game Semantics and its Applications.

Appendix A. Proof of Theorem 14

We first prove two useful lemmas. Let \( \sigma \) and \( \tau \) be strategies-with-state over arenas \( A \Rightarrow B \) and \( B \Rightarrow C \) respectively. We assume that both arenas are of order at most three.

Lemma 40. (i) For any \( u \in \text{ISeq}(\sigma, \tau) \), we have
\[
\text{erase } u \in \text{ISeq(erase } \sigma, \text{ erase } \tau).
\]

(ii) For any \( u, v \in \text{ISeq}(\sigma, \tau) \), if \( \text{erase } u = \text{erase } v \) then \( u = v \).

Proof. (i) Take any \( u \in \text{ISeq}(\sigma, \tau) \). By Lemma 11, there is a unique \( t \in \tau \) such that \( t \triangleleft u \vdash (B, C) \). It follows that \( \text{erase } t = \text{erase } (u \vdash (B, C)) = (\text{erase } u) \vdash (B, C) \). That is, \( (\text{erase } u) \vdash (B, C) \in \text{erase } \tau \). Similarly we have
\[
(\text{erase } u) \vdash (A, B, b) \in \text{erase } \sigma
\]
for each occurrence \( b \) of an initial \( B \)-move in \( u \).

(ii) We prove \( u = v \) by induction on the length of \( u \). The base case is trivial. For the inductive step, suppose \( u \triangleleft m^S n^{S_1}, u \triangleleft n^{S_2} \in \text{ISeq}(\sigma, \tau) \). We aim to prove \( S_1 = S_2 \) by a case analysis of \( m \). Suppose \( m \) is a P-move in \( A \Rightarrow C \). Then \( n^{S_1} \) and \( n^{S_2} \) are explicitly justified by the same move in \( u \), say \( m_{S_0} \). By axiom (13) of composition of
strategies-with-state, we have $S_1 = S_0[S] = S_2$. Suppose $m$ is an O-move in component $(\beta, C)$. We have $S_1 \nexists L(B, C) = S \nexists L(B, C) = S_2 \nexists V(B, C)$. So it remains to prove that $S_1 \nexists L(B, C) = S_2 \nexists V(B, C)$. By axiom (I1) and Lemma 11, for some unique $t \in \tau$, and some unique states $R$, $R_1$, and $R_2$, we have $tnR_1 \nless \langle u m^S, 1 \rangle \nexists L(B, C)$ and $tnR_2 \nless \langle u m^S, 1 \rangle \nexists L(B, C)$. Since $\tau$ satisfies Determinacy, we have $R_1 = R_2 = R'$ (say). Now $tnR' \nless \langle u m^S, 1 \rangle \nexists L(B, C)$ for $i = 1, 2$. By definition of $\nless$, we have $S_i = S(R')$. We omit the other case of $m$ being an O-move in component $(A, B, b)$ as it is similar. □

**Lemma 41.** For any $v \in \text{IsEq}(\text{erase } \sigma, \text{erase } \tau)$, there exists $u \in \text{IsEq}(\sigma, \tau)$ such that $\text{erase } u = v$ (thanks to (ii), $u$ is unique).

**Proof.** The proof is by induction on the length of $v$. The base case is trivial. For the inductive case take $v m n \in \text{IsEq}(\text{erase } \sigma, \text{erase } \tau)$. By the induction hypothesis, we have $\text{erase } (u m^S) = v m$ for some $u m^S \in \text{IsEq}(\sigma, \tau)$.

Suppose $m^S$ is a P-move in $A \Rightarrow C$. Let $m_0$ be the move in $v$ that explicitly justifies $n$, and let $m_0^S$ be the move in $u$ that erases to $m_0$. It suffices to show that $u m^S n_0(S) \in \text{IsEq}(\sigma, \tau)$ as its erasure is obviously $v m n$. W.l.o.g. assume that $m$ and hence $n$ are C-moves. Since $u m^S \in \text{IsEq}(\sigma, \tau)$, (I1) says that there is some $t \in \tau$ such that $t \langle u m^S \rangle \nexists (B, C)$. Let $m_0^T$ be the move in $t$ that corresponds to $m_0^S$ in $u$ under $\nless$, and let $m^T$ be the last move of $t$. We then have $t n^T \nexists (B, C)$. So it remains to prove that $S_0[T] \subseteq S$.

Take a location $l$ that is defined at $a_0^T$. Suppose $T_0[T](l) = l$. It is easy enough to show that $l$ is also defined at $S$. The tricky part is to show that $S(l) = l$.

Let $m^S_0$ be the move in $u$ that explicitly justifies $m^S$. By Visibility, $a_0$ must appear in the P-view of $v m \nexists (B, C)$, and so, $u m^S$ has the following shape:

$$\cdots b \cdots a_0^T \cdots a_1 \cdots b_1 \cdots b_2 \cdots b_k \cdots \cdot_{l} \cdot_{l_{0}} \cdots \cdot_{l_k} \cdot_{l_{0}} \cdots \cdot_{l_k} m^S_0 \cdots m^S$$

where each $\cdot_{l_i}$, which is an O-move in $B \Rightarrow C$ (for if some $\cdot_{l_j}$ were an A-move, then it would follow that $m^S$ is an A-move, contradicting our assumption), is explicitly justified by $\cdot_{l_i}$; for notational convenience, we set $\cdot_{l_k+1}$ and $\cdot_{l_{k+1}}$ to be $m_0^S$ and $m^S$ respectively. Now $a_0$ must be a second-order O-move of $B \Rightarrow C$. By Visibility, we note that each segment $\cdot_{l_i} \cdots \cdot_{l_k}$ has the following shape:

$$\cdot_{l_i} a_1 u_1 b_1 \cdots a_r u_r b_r \cdots \cdot_{l_k}$$
where each \( b_j \) is explicitly justified by \( a_j \). We aim to prove the following by induction on \( i \).

**Claim**

1. For each \( 1 \leq i \leq k + 1 \), \( l \) is defined at \( o_i \) and \( \bullet_i \).
2. Suppose there is an O-question \( [ \cdot ] \), which is a move of \( A \Rightarrow C \) (say) in \( o_i \cdots \bullet_i \), that is explicitly justified by some (necessarily first-order) P-question \( q \) (say) at which \( l \) is not defined. Then it must be one of the O-questions in a segment of the form

\[
\alpha = (\ldots \ldots 1)_{a_0}
\]

where

(a) \( [ \cdot ] \) is a third-order question of \( C \), at which \( l \) is defined, and which is explicitly justified by \([ \cdot ]\) (say),
(b) each \([ \cdot ] \) in \( \alpha \) is explicitly justified by \( q \), and
(c) \( \alpha \) is either the whole of \( o_i \cdots \bullet_i \), or it is a segment of some \( u_j \) in \( o_i \cdots \bullet_i \).

By (I3), we observe that \( l \) is not defined at \([ \cdot ]\), which must also be explicitly justified by \( q \).

We claim that \( l \) is not defined at any move in the subsegment \( a_0 \) of \( \alpha \). Hence by condition (SC-O) the respective states at \( [ \cdot ] \) map \( l \) to the same value. (The point is that if there were moves deep inside \( a_0 \) at which \( l \) is defined, then any updates by them might not be reflected at \( [ \cdot ] \) by State Change. Essentially the same point is made in Remark 8.)

Suppose \( l \) is defined at each of \( o_1 \cdot \bullet_1 \cdots \bullet_i \cdot \bullet_j \). It follows from (I3') that \( l \) is also defined at \( o_{i+1} \). Now suppose a segment of the form \( \alpha \) from (2) occurs in \( o_{i+1} \cdots \bullet_{i+1} \).

By induction on the number of times such \( \alpha \)-segments occur after \( a_0 \), it is straightforward to see that \([ \cdot ]\) (as defined in (2) above), and hence \( q \), must occur before \( b \), the move that explicitly justifies \( a_0 \). Suppose now, for a contradiction, we have

\[
\cdots q \cdots [\ldots \cdots b \cdots \alpha\overset{-1} \rightarrow \cdots a_0 \cdots \bullet_1 \cdots o_{i+1} \cdots (\ldots \ldots \ldots e \ldots \ldots \ldots)]_{u_2}
\]

where \( e \), which must be a P-move of \( B \Rightarrow C \) in component \( (A, B, b) \), is the first move, after \( [ \cdot ] \), in the segment \( \alpha \), at which \( l \) is defined. Now this means that \( e \) is explicitly justified by some move \( q' \) (say) which must occur in the segment \( u_2 \). As \( [ \cdot ] \) is explicitly justified by \([ \cdot ]\), and \( e \) occurs before \([ \cdot ]\) is answered (as the three named moves are in \( B \Rightarrow C \), we can take the projection of the above interaction sequence onto \( B \Rightarrow C \) and obtained a legal position), we obtain a contradiction of Observation 1(ii). Note that \( l \) is defined at \( [ \cdot ] \), the last move of \( \alpha \), since \( l \) is defined at \([ \cdot ]\). Thus we see that \( l \) is defined at \( o_{i+1} \).

On the other hand, if no segment of the form \( \alpha \) occurs within \( o_{i+1} \cdots \bullet_{i+1} \), then it follows from (I3) and (I3') that \( l \) is defined in every move in the segment \( o_{i+1} \cdots \bullet_{i+1} \), and so, in particular, it is defined at \( o_{i+1} \).

The upshot of (1) and (2) is that after \( a_0 \) in the interaction sequence \( um^z \), the only moves at which \( l \) is not defined are moves from \( a_0 \) within segments of the form \( \alpha \). Thus \( l \) is not just defined at \( S \) but is mapped to the same value as \( T_0[T](l) \). □
Remark 42. Lemma 41 does not hold if the restriction to order three is removed (as Observation 1 would fail). For example, take $\sigma$ and $\tau$ as given by

\begin{align*}
\lambda f, \lambda z &. f(z := 0) \in (\exp \Rightarrow \com) \Rightarrow \exp \\
F &. (\exp \Rightarrow \com) \Rightarrow \exp \vdash \lambda H. H F : ( ((\exp \Rightarrow \com) \Rightarrow \exp) \Rightarrow \exp) \Rightarrow \exp.
\end{align*}

respectively: the former is a strategy-with-state of order 2, the latter, being an innocent strategy of order 4, is of course also a strategy-with-state.

We can now prove the theorem. We define $\text{erase } \text{ISeq}(\sigma, \tau) = \{ \text{erase } u : u \in \text{ISeq}(\sigma, \tau) \}$. It suffices to prove

$$
\text{erase } (\text{ISeq}(\sigma, \tau)) = \text{ISeq}(\text{erase } \sigma, \text{erase } \tau).
$$

The $\subseteq$-inclusion follows from Lemma 40(i), and the other inclusion follows from Lemma 41.

Appendix B. Proof of Theorem 16

Let $u \in \text{ISeq}(\sigma, \tau)$. We define the core of $u$, written $\overline{u}$, to be the subsequence of $u$ that is obtained by deleting all segments of the form $\bullet \cdots \circ$ from $u$, where $\bullet$ is an O-move in $A$ or $C$, $\circ$ is a P-move in $A$ or $C$, all the intervening moves are in $B$, and neither $\bullet$ nor $\circ$ appear in $\overline{u} \downarrow (A, C)$. We shall omit the proofs of the three lemmas that follow, as they are stateful versions of those that are used to prove the compositionality of innocent strategies in [19].

Lemma 43. Let $u \in \text{ISeq}(\sigma, \tau)$ and suppose the last move of $u$ is an O-move in component $X$. Then $\overline{u} \downarrow X = \overline{\overline{u}} \downarrow X$. □

Remark 44. Recall that the definition of $\overline{u} \downarrow X$ depends on Lemma 11, which assumes that $u$ is an interaction sequence. Now $\overline{u}$ is not in general an interaction sequence, but we observe that the same argument in the proof of the lemma can be used to infer that there is a unique justified $t$, in which the O-moves do not necessarily satisfy State Change, such that $t \lessdot \overline{u} \downarrow X$, where $t$ is in the evaluation tree of $\sigma_X$, the strategy-with-state corresponding to $X$. Thus we can define $\overline{\overline{u} \downarrow X}$ to be $\overline{\overline{u}} \downarrow X$.

Since all of the moves of $\overline{u}$ which occur in $A$ or $C$ come from $\overline{u} \downarrow (A, C)$, it is clear that we can calculate $\overline{\overline{u}}$ from $\overline{u} \downarrow (A, C)$ alone.

Lemma 45. If $s = u \downarrow (A, C)$ and $t = v \downarrow (A, C)$ such that $u, v \in \text{ISeq}(\sigma, \tau)$ and $s, t \in \sigma \downarrow \tau$ such that $\overline{\overline{s}} \downarrow \overline{\overline{t}} = \overline{\overline{\overline{s}}} \downarrow \overline{\overline{\overline{t}}}$, then $ua^\overline{s} = va^\overline{t}$. □

Lemma 46. If $s = u \downarrow (A, C)$ for some $u \in \text{ISeq}(\sigma, \tau)$, and $sa^\overline{s}$ is an odd-length legal position of $A \Rightarrow C$, then $sa^\overline{s}b^T \in \sigma \downarrow \tau$ if and only if there exist $B$-moves $m_1^{S_1}, \ldots, m_k^{S_k}$ such that, writing $m_0^{S_0} = a^S$ and $m_k^{S_{k+1}} = b^T$, for each $i = 0, \ldots, k$,

1. $h_{X_{m_i}}(\overline{u}m_0^{S_0}m_1^{S_1} \cdots m_i^{S_i} \downarrow X_{m_i}^{-1}) = R_{i+1}$,
2. $S_{i+1} = S_i(R_{i+1})$,

in which case, we have $sa^\overline{s}b^T = (ua^S m_1^{S_1} \cdots m_k^{S_k}b^T) \downarrow (A, C)$. □
We can now prove Theorem 16. Suppose even-length $sa^Sb^T, t \in \sigma \triangleright \tau$, and $ta^S$ is a legal position of $A \Rightarrow C$ such that $\tau sa^S = \tau ta^S$. We aim to show that $ta^Sb^T \in \sigma ; \tau$. By definition of $\sigma ; \tau, s$ is witnessed by some $u \in \text{ISeq}(\sigma, \tau)$ (we take the shortest such) such that there is a unique $s^- \in \tau$ such that $s^- \triangleleft u \upharpoonright (B, C)$, and for each occurrence $b$ of an initial $B$-move in $u$, there is a unique $s_b \in \sigma$ such that $s_b \triangleleft u \upharpoonright (A, B, b)$; similarly $t$ is witnessed by $v \in \text{ISeq}(\sigma, \tau)$. Since $sa^Sb^T \in \sigma ; \tau$, by Lemma 46, there exist $B$-moves $m_1^S, \ldots, m_k^S$ satisfying conditions (1) and (2) therein such that $sa^Sb^T = (ua^Sm_1^S \cdots m_k^S) \upharpoonright (A, C)$. Since $\tau sa^S = \tau ta^S$, Lemma 45 tells us that $ua^S = va^S$. By Lemmas 43 and 46, and setting $m_0^S = a^S$ and $m_{k+1}^S = b^T$ for notational brevity, we have for each $i = 0, \ldots, k$,

$$h_{X_{m_i}} (\text{um}_0^Sm_1^S \cdots m_i^S \upharpoonright X_{m_i}) = R_{i+1}$$

(B.1)

using the fact that $ua^S m_1^S \cdots m_k^S = ua^S m_1^S \cdots m_k^S$. But since $ua^S = va^S$, we can substitute $v$ for $u$ in Eq. (B.1), and use Lemmas 43 and 46 again. We need to check that condition (2) of the lemma remains valid after the substitution. Thus we deduce that $ta^Sb^T = va^S m_1^S \cdots m_k^Sb^T \upharpoonright (A, C)$. \hfill \Box

References


