Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 218 (2008) 241–259
www.elsevier.com/locate/entcs

Global State Considered Helpful

Paul Blain Levy [1]

University of Birmingham, Birmingham B15 2TT, U.K.

Abstract

Reynolds' view of a storage cell as an expression-acceptor pair has been widely used by researchers. We present a different way of organizing semantics of state, and in particular game semantics, by adding to typing contexts a zone for global state. This has the following advantages.
Firstly, it causes the "good variable" equations for references to be validated, and also the noninterference equations between distinct references, as enumerated by Plotkin and Power.
Secondly, it gives a cleaner categorical structure based on the configurations (state + program) used to describe operational semantics.
Thirdly, it leads to a simpler proof that the game semantics is sound and adequate with respect to the operational semantics.

Keywords: game semantics, categorical semantics, good variables

1 Introduction

Languages with state often have special types for storage cells, such as `ref` types in ML, and `var` types in Algol-like languages. Some denotational models interpret these as a type of distinguishable names [9,14,22], but in other models that is unsuitable. Reynolds [20] suggested that a cell could be regarded as an "expression-acceptor pair", i.e. a function that returns the current value, together with a procedure that updates it. This suggestion was successfully adopted in both possible world [15] and game semantics [1,2,19].

It was noted, however, that a cell is not an arbitrary expression-acceptor pair, but enjoys some special properties. For example, writing to a cell and then reading it gives the value just written. These so-called "good variable" properties [2] were enumerated in [18] in order to axiomatize the computational effect of global state. Recent papers in game semantics have addressed the "good variable" issue by restricting strategies in various ways [12,13].

[1] Email: pbl@cs.bham.ac.uk

[2] The call-by-name nature of Idealized Algol obscures this issue, because a `var` type is a type not of cells but of state-dependent expressions that return a cell. Such expressions are not necessarily "good variables".

doi:10.1016/j.entcs.2008.10.015

In this paper, we propose a different approach that entirely avoids this problem. We do not have cell types at all. Instead, we have two-zone contexts $\Delta;\ \Gamma$, where Γ is an ordinary context, and Δ (the "storage context") is a list of cells. For example, a context $\mathtt{n} : \mathtt{bool}, \mathtt{n}' : \mathtt{bool}, \mathtt{n}'' : \mathtt{bool};\ \mathtt{f} : \mathtt{nat} \to \mathtt{nat}$ means that $\mathtt{n}, \mathtt{n}', \mathtt{n}''$ are bound to *distinct* cells storing booleans, and $\mathtt{f}$ is bound to a function.

A two-zone context of this kind is hardly novel, as it has been used for the formulation of operational semantics in [4,5] and Chapter 5 of [21]. But in our formulation the `new` construct binds cells from the storage context Δ, which is less familiar, though it does appear in [17].

We shall look at the impact of this arrangement on game semantics of state. For illustration, let us consider the following call-by-value term $M : \mathtt{nat}$ in the above two-zone context:

$$
\begin{array}{l}
\mathtt{n} := \mathtt{true}. \\
\mathtt{n} := \mathtt{false}. \\
\mathtt{n}'' := \mathtt{false}. \\
\mathtt{read\ n\ as}\ \{ \\
\quad \mathtt{true}.\ \mathtt{f}(3) + 4 \\
\quad \mathtt{false}.\ \mathtt{f}(7) + 5 \\
\}
\end{array}
$$

In the traditional game semantics in [2], M denotes a set of dialogues between Proponent (P) and Opponent (O) such as the following.

P: Set $\mathtt{n}$ to be true.
O: OK.
P: Set $\mathtt{n}$ to be false.
O: OK.
P: Set $\mathtt{n}''$ to be false.
O: OK.
P: What is $\mathtt{n}$ currently?
O: True.
P: I call $\mathtt{f}$ with argument 3.
O: Your call returns 2.
P: I return 6.

Note that this play involves unnecessary information (the first call to $\mathtt{n}$) and an impossible response from Opponent (that $\mathtt{n}$ is true). The assignment and reading commands are seen as no different from function calls.

In the game semantics we shall present, here is an example dialogue of M proceeding from the initial state $\mathtt{n} \mapsto \mathtt{true}, \mathtt{n}' \mapsto \mathtt{false}, \mathtt{n}'' \mapsto \mathtt{true}$.

P: I call $\mathtt{f}$ with argument 7, in state $\mathtt{n} \mapsto \mathtt{false}, \mathtt{n}' \mapsto \mathtt{false}, \mathtt{n}'' \mapsto \mathtt{false}$.
O: Your call returns 14, in state $\mathtt{n} \mapsto \mathtt{false}, \mathtt{n}' \mapsto \mathtt{true}, \mathtt{n}'' \mapsto \mathtt{false}$.
P: I return 19, in state $\mathtt{n} \mapsto \mathtt{false}, \mathtt{n}' \mapsto \mathtt{true}, \mathtt{n}'' \mapsto \mathtt{false}$.

Now the entire global state must be described in each move, but no moves are required to read or assign to a cell. Note that Opponent assigns to $\mathtt{n}'$—it is a global cell, so both players have access to it.

It is clear how to calculate the denotation of $M' \stackrel{\text{def}}{=} \mathtt{new}\ \mathtt{n}'' := \mathtt{true}.\ M$ from the denotation of M. The effect of `new` is to make $\mathtt{n}''$ into a private cell that Opponent does not have access to. So we look at those dialogues in which $\mathtt{n}''$ is initially true, and Opponent never changes $\mathtt{n}''$—such as the dialogue we saw—and then erase all mention of $\mathtt{n}''$

It is also clear how to weaken M by calculating its denotation in the bigger context $\mathtt{n} : \mathtt{bool}, \mathtt{n}' : \mathtt{bool}, \mathtt{n}'' : \mathtt{bool}, \mathtt{n}''' : \mathtt{nat};\ \mathtt{f} : \mathtt{nat} \to \mathtt{nat}$. This time, we consider those plays where Proponent never changes the contents of $\mathtt{n}'''$ and erasing $\mathtt{n}'''$ yields a play on M.

These two operations, hiding and weakening, in combination with the traditional strategy operations of composition and copycat, provide a simple categorical structure from which the semantics of the individual syntactic constructs is easily obtained. Indeed the game semantics in this paper is not new—it is the same as [1]—it is only the organization which is different [3]. Moreover, the soundness of the model wrt operational semantics is immediate, and this had previously proved challenging, especially in the setting of higher-order store. And the method of [11] can easily be applied to give computational adequacy.

Structure of Paper First we look at a calculus without store, its categorical semantics and then game semantics. Then we do the same with store. Along the way, we shall need in Sect. 3.1 to develop the theory of *expansions* in order to formulate an injective renaming lemma.

2 Basic Language

2.1 Syntax

To make the game semantics as simple as possible, we work with a calculus JWA where functions are called (by value) but do not return. The types are given by

$$A ::= \quad \neg A \mid \textstyle\sum_{i \in I} A_i \mid 1 \mid A \times A \mid \mathtt{X} \mid \mathtt{rec}\ \mathtt{X}.\ A$$

where I ranges over finite sets (or countable, for an infinitary variant). The type $\neg A$ corresponds to $A \to 0$ in call-by-value. There are two kinds of terms, *values* and *nonreturning commands*, indicated by the judgements $\Gamma \vdash^{\mathtt{v}} V : A$ and $\Gamma \vdash^{\mathtt{n}} M$ respectively. The types in Γ and A are all closed. The syntax is shown in Fig. 1.

A *renaming* $\Gamma \ ^{\theta}\!\succ \Gamma'$ maps each identifier in Γ to one of the same type in Γ', whereas a *substitution* $\Gamma \ ^{k}\!\succ \Gamma'$ maps each identifier in Γ to a value. These induce operations $\theta^\dagger$ and k^* on terms in the usual way. They are used in the operational semantics (Fig. 2) and the equational theory (Fig. 3). We write ${}^{\mathtt{x}}M$ to mean M weakened by $\mathtt{x}$.

[3] Another recent categorical semantics for higher-order store is that of [8], but this has been applied to

$$\frac{}{\Gamma \vdash^{\mathsf{v}} \mathtt{x} : A}\ (\mathtt{x} : A) \in \Gamma \qquad \frac{\Gamma \vdash^{\mathsf{v}} V : A \quad \Gamma, \mathtt{x} : A \vdash^{\mathsf{n}} M}{\Gamma \vdash^{\mathsf{n}} \mathtt{let}\ V\ \mathtt{be}\ \mathtt{x}.\ M}$$

$$\frac{\Gamma \vdash^{\mathsf{v}} V : A_{\hat{\imath}}}{\Gamma \vdash^{\mathsf{v}} \langle \hat{\imath}, V \rangle : \sum_{i \in I} A_i}\ \hat{\imath} \in I \qquad \frac{\Gamma \vdash^{\mathsf{v}} V : \sum_{i \in I} A_i \quad \Gamma, \mathtt{x}_i : A_i \vdash^{\mathsf{n}} M_i\ \ (\forall i \in I)}{\Gamma \vdash^{\mathsf{n}} \mathtt{pm}\ V\ \mathtt{as}\ \{\langle i, \mathtt{x}_i \rangle . M_i\}_{i \in I}}$$

$$\frac{\Gamma \vdash^{\mathsf{v}} V : A \quad \Gamma \vdash^{\mathsf{v}} V' : A'}{\Gamma \vdash^{\mathsf{v}} \langle V, V' \rangle : A \times A'} \qquad \frac{\Gamma \vdash^{\mathsf{v}} V : A \times A' \quad \Gamma, \mathtt{x} : A, \mathtt{y} : A' \vdash^{\mathsf{n}} M}{\Gamma \vdash^{\mathsf{n}} \mathtt{pm}\ V\ \mathtt{as}\ \langle \mathtt{x}, \mathtt{y} \rangle .\ M}$$

$$\frac{\Gamma, \mathtt{x} : A \vdash^{\mathsf{n}} M}{\Gamma \vdash^{\mathsf{v}} \lambda \mathtt{x}.M : \neg A} \qquad \frac{\Gamma \vdash^{\mathsf{v}} V : \neg A \quad \Gamma \vdash^{\mathsf{v}} W : A}{\Gamma \vdash^{\mathsf{n}} VW}$$

$$\frac{\Gamma \vdash^{\mathsf{v}} V : A[\mathtt{rec}\ \mathtt{X}.\ A/\mathtt{X}]}{\Gamma \vdash^{\mathsf{v}} \mathtt{fold}\ V : \mathtt{rec}\ \mathtt{X}.\ A} \qquad \frac{\Gamma \vdash^{\mathsf{v}} V : \mathtt{rec}\ \mathtt{X}.\ A \quad \Gamma, \mathtt{x} : A[\mathtt{rec}\ \mathtt{X}.\ A/\mathtt{X}] \vdash^{\mathsf{n}} M}{\Gamma \vdash^{\mathsf{n}} \mathtt{pm}\ V\ \mathtt{as}\ \mathtt{fold}\ \mathtt{x}.\ M}$$

Fig. 1. Syntax of JWA with type recursion (the 1 type is omitted)

Transitions		**Terminal configurations**
$\mathtt{let}\ V\ \mathtt{be}\ \mathtt{x}.\ M$	$\rightsquigarrow M[V/\mathtt{x}]$	
$\mathtt{pm}\ \langle \hat{\imath}, V \rangle\ \mathtt{as}\ \{\langle i, \mathtt{x} \rangle . M_i\}_{i \in I}$	$\rightsquigarrow M_{\hat{\imath}}[V/\mathtt{x}]$	$\mathtt{pm}\ \mathtt{z}\ \mathtt{as}\ \{\langle i, \mathtt{x} \rangle . M_i\}_{i \in I}$
$\mathtt{pm}\ \langle V, V' \rangle\ \mathtt{as}\ \langle \mathtt{x}, \mathtt{y} \rangle .\ M$	$\rightsquigarrow M[V/\mathtt{x}, V'/\mathtt{y}]$	$\mathtt{pm}\ \mathtt{z}\ \mathtt{as}\ \langle \mathtt{x}, \mathtt{y} \rangle .\ M$
$\mathtt{pm}\ \mathtt{fold}\ V\ \mathtt{as}\ \mathtt{fold}\ \mathtt{x}.\ M$	$\rightsquigarrow M[V/\mathtt{x}]$	$\mathtt{pm}\ \mathtt{z}\ \mathtt{as}\ \mathtt{fold}\ \mathtt{x}.\ M$
$(\lambda \mathtt{x}.M)V$	$\rightsquigarrow M[V/\mathtt{x}]$	$\mathtt{z}V$

Fig. 2. Operational semantics on commands in fixed context Γ

Remark 2.1 For recursive types, we have included only the most rudimentary equations—merely asserting an isomorphism $\mathtt{rec}\ \mathtt{X}.\ A \cong A[\mathtt{rec}\ \mathtt{X}.\ A/\mathtt{X}]$.

2.2 Semantics of Types, Contexts, Renamings

We recall the "families" construction from [3]: if $\mathcal{C}$ is a category, then an object of $\mathsf{fam}(\mathcal{C})$ is a family of $\mathcal{C}$-objects. The homset from $\{A_i\}_{i \in I}$ to $\{B_j\}_{j \in J}$ is given by $\prod_{i \in I} \sum_{j \in J} \mathcal{G}(A_i, B_j)$. This inherits finite products from $\mathcal{C}$.

An *arena* is a countable forest; we write $\vdash$ to mean "is a parent of", and say $* \vdash r$ when r is a root. We write $\mathsf{rt}\, R$ for the roots of R, and $R \upharpoonright r$ for the arena of

different kinds of games.

$$\begin{aligned}
\texttt{let } V \texttt{ be x. } M &= M[V/\texttt{x}] \\
\texttt{pm } \langle \hat{\imath}, V\rangle \texttt{ as } \{\langle i, \texttt{x}\rangle.M_i\}_{i\in I} &= M_{\hat{\imath}}[V/\texttt{x}] \\
\texttt{pm } \langle V, V'\rangle \texttt{ as } \langle \texttt{x}, \texttt{y}\rangle.\ M &= M[V/\texttt{x}, V'/\texttt{y}] \\
\texttt{pm fold } V \texttt{ as fold x. } M &= M[V/\texttt{x}] \\
(\lambda \texttt{x}.M)V &= M[V/\texttt{x}]
\end{aligned}$$

$$\begin{aligned}
M[V/\texttt{z}] &= \texttt{pm } V \texttt{ as } \{\langle i, \texttt{x}\rangle.\ ^{\texttt{x}}M[\langle i, \texttt{x}\rangle/\texttt{z}]\}_{i\in I} \\
M[V/\texttt{z}] &= \texttt{pm } V \texttt{ as } \langle \texttt{x}, \texttt{y}\rangle.\ ^{\texttt{x,y}}M[\langle \texttt{x}, \texttt{y}\rangle/\texttt{z}] \\
M[V/\texttt{z}] &= \texttt{pm } V \texttt{ as fold x. } ^{\texttt{x}}M[\texttt{fold x}/\texttt{z}] \\
V &= \lambda \texttt{x}.\ ^{\texttt{x}}V\texttt{x}
\end{aligned}$$

Fig. 3. Equational laws for JWA

proper descendants of r. We write $\uplus$ for disjoint union, and $\mathsf{pt}_{i\in I}R_i$ for the arena with I roots and a copy of R_i placed below the ith root.

A closed type denotes an arena family [4], in the following manner:

$$\begin{aligned}
1 &\stackrel{\text{def}}{=} \{\emptyset\}_{\langle\rangle\in 1} \\
\{R_i\}_{i\in I} \times \{S_j\}_{j\in J} &\stackrel{\text{def}}{=} \{R_i \uplus S_j\}_{\langle i,j\rangle\in I\times J} \\
\textstyle\sum_{i\in I}\{R_{ij}\}_{j\in J_i} &\stackrel{\text{def}}{=} \{R_{ij}\}_{\langle i,j\rangle\in I\times J} \\
\neg\{R_i\}_{i\in I} &\stackrel{\text{def}}{=} \{\mathsf{pt}_{i\in I}R_i\}_{\langle\rangle\in 1}
\end{aligned}$$

The semantics of (open types and) recursive types follows [3], giving an arena isomorphism $[\![\texttt{rec X. } A]\!] \cong [\![A[\texttt{rec X. } A/\texttt{X}]]\!]$. A context Γ denotes an arena family using the 1 and $\times$ operations.

An *arena renaming morphism* is a function $R \overset{f}{\succ} S$ that maps each root $b \in \mathsf{rt}\, R$ to a root $fb \in \mathsf{rt}\, S$ and restricts to an arena isomorphism $R\restriction_b \overset{f_b}{\succ} S\restriction_{fb}$. These form a cocartesian category **TokCh**. Renamings between contexts are interpreted in $\mathcal{B}^{\mathsf{op}}$ where $\mathcal{B} \stackrel{\text{def}}{=} \mathsf{fam}(\mathbf{TokCh}^{\mathsf{op}})$. The category $\mathcal{B}$ is a countably distributive and equipped with an endofunctor $\neg$ on its isomorphism groupoid $\mathsf{Isos}\ \mathcal{B}$. Such a category is called a *JWA base*.

2.3 Categorical Structure

For a category $\mathcal{C}$, a *left $\mathcal{C}$-module* is a functor $\mathcal{C}^{\mathsf{op}} \overset{\mathcal{N}}{\succ} \mathbf{Set}$. We think of $\mathcal{N}(R)$ as a homset—its elements are "morphisms from R" and written $R \overset{f}{\succ}$. We use them to interpret nonreturning commands.

Definition 2.2 (i) A *first-order JWA model* on a base $\mathcal{B}$ consists of

- a category $\mathcal{C}$, with the same objects as $\mathcal{B}$
- a left $\mathcal{C}$-module $\mathcal{N}$
- an identity-on-objects functor $\mathcal{B} \overset{K}{\succ} \mathcal{C}$

[4] Throughout this paper, "family" means *countable* family.

such that all the following functions are isomorphisms

$$\begin{array}{rcl} \mathcal{C}(A, B \times B') & \longrightarrow & \mathcal{C}(A,B) \times \mathcal{C}(A,B') \\ f & \longmapsto & \langle (f; K\pi), (f; K\pi') \rangle \end{array} \qquad \begin{array}{rcl} \mathcal{C}(A,1) & \longrightarrow & 1 \\ f & \longmapsto & \langle\rangle \end{array}$$

$$\begin{array}{rcl} \mathcal{C}(A \times \sum_{i\in I} B_i, C) & \longrightarrow & \prod_{i\in I} \mathcal{C}(A \times B_i, C) \\ \mathcal{N}(A \times \sum_{i\in I} B_i) & \longrightarrow & \prod_{i\in I} \mathcal{N}(A \times B_i) \\ f & \longmapsto & \lambda i.((A \times \mathsf{in}_i); f)) \end{array}$$

A *JWA model* on $\mathcal{B}$ is a first-order JWA model together with an isomorphism

$$\mathcal{N}(A \times B) \cong \mathcal{C}(A, \neg B) \text{ natural in } A \in \mathcal{C}^{\mathsf{op}}, B \in \mathsf{Isos}\ \mathcal{B} \tag{1}$$

We write $\mathsf{FOJWA}(\mathcal{B})$ for the (large) category of first-order JWA models on base $\mathcal{B}$, and $\mathsf{JWA}(\mathcal{B})$ for the (large) category of JWA models on base $\mathcal{B}$. Morphisms are identity on objects.

We emphasize that, in the semantics of JWA, a renaming is interpreted in $\mathcal{B}^{\mathsf{op}}$, whereas a substitution is interpreted in $\mathcal{C}^{\mathsf{op}}$.

2.4 Strategies

If $\mathcal{S}$ is a left $\mathcal{G}$-module, then $\mathsf{fam}(\mathcal{S}) : \{R_i\}_{i\in I} \mapsto \prod_{i\in I} \mathcal{S}(R_i)$ is a left $\mathsf{fam}(\mathcal{C})$-module. Using this construction, we will build a JWA model $(\mathcal{C}, \mathcal{N}) = (\mathsf{fam}(\mathcal{G}), \mathsf{fam}(\mathcal{S}))$ on the base $\mathcal{B} \stackrel{\text{def}}{=} \mathsf{fam}(\mathbf{TokCh}^{\mathsf{op}})$.

Definition 2.3 (NB Proponent begins) Let R be an arena. We define $* \stackrel{\text{def}}{=} -1$.

(i) A (finite) *justified sequence* on an arena R is a sequence $m_0, \ldots, m_{n-1}$ where each $m_i = (p_i, r_i)$ consists of a *pointer* $* \leqslant p_i < i$ and an element $r_i \in R$ such that $r_{p_i} \vdash r_i$, where $r_* \stackrel{\text{def}}{=} *$.

(ii) A justified sequence is a *play* when $i - p_i$ is odd for every $i < n$. In a play, a move $i < n$ is a *Proponent move* or an *Opponent move* according as i is even or odd. A play is *prior* or *posterior* according as its length is even or odd.

(iii) A *strategy* is a prefix-closed set σ of posterior plays that is deterministic: if $sm, sm' \in \sigma$ then $m = m'$. We write $\mathcal{S}(R)$ for the set of strategies on R.

(iv) We write $\mathcal{S}_{\mathsf{HO}}(R)$ for $\prod_{b\in \mathsf{rt}\, R} \mathcal{S}(R_b)$. This is (isomorphic to) the set of strategies where (as in [7]) Opponent begins, and may not point to $*$ after the initial move.

Definition 2.4 Let R and S be arenas.

(i) We define the Hyland-Ong exponential $R \to_{\mathsf{HO}} S \stackrel{\text{def}}{=} \mathsf{pt}_{b\in \mathsf{rt}\, S}(R \uplus S_b)$

(ii) We define the homset

$$\mathcal{G}(R,S) \stackrel{\text{def}}{=} \mathcal{S}_{\mathsf{HO}}(R \to_{\mathsf{HO}} S) \cong \prod_{b\in \mathsf{rt}\, S} \mathcal{S}(R \uplus S_b)$$

(iii) For an arena renaming morphism $S \xrightarrow{f} R$, we define the *copycat* $R \xrightarrow{Kf} S$ in $\mathcal{G}$. At $b \in \mathsf{rt}\, S$, it is the set of posterior plays on $R \uplus S_b$ in which Proponent begins with $(*, f(b))$ and responds to (j, r) by pointing to $j - 1$ and playing $f_b(r)$ or $f_b^{-1}(r)$ according as $r \in S_b$ or $r \in R_{f(b)}$.

Definition 2.5 Let R, S, T be arenas.

(i) An *interaction sequence* on R, S, T is a justified sequence s on $(R \to_{\mathsf{HO}} S) \uplus T$ such that
- the *right inner thread* $s \upharpoonright S \uplus T$ is a play
- for each move m playing $b \in \mathsf{rt}\, S$, the *left inner thread* $s \upharpoonright m$ consisting of moves strictly descended from m is a play on $R \uplus S_b$
- the *outer thread* $s \upharpoonright R \uplus T$ (with the pointer from each R root move changed to $*$) is a play.

It is *outer-posterior* when the outer thread is posterior.

(ii) Let $\sigma \in \mathcal{G}(R, S)$ and let $\tau \in \mathcal{S}(S \uplus T)$. We define $\sigma \circledast \tau \in \mathcal{S}(R \uplus T)$ to be the outer thread of each outer-posterior interaction sequence s on R, S, T whose inner threads $(s \upharpoonright m) \in \sigma_{r(m)}$ and $(s \upharpoonright S \uplus T) \in \tau$.

The composite of $R \overset{f}{\succ} S \overset{g}{\succ} T$ is defined at $c \in \mathsf{rt}\, T$ by $\circledast$, while the composite of $R \overset{f}{\succ} S \overset{g}{\succ}$ is given by $\circledast_{R,S,\emptyset}$. The identity on R is given by $K\mathsf{id}_R$. This gives all the required structure, and we recover $f \circledast g$ as $(f \times T); g$. Moreover, pre- and post-composition with Kf is given by renaming of elements. Applying the families construction to $(\mathcal{G}, \mathcal{S})$ gives a JWA model $(\mathcal{C}, \mathcal{N})$ on base $\mathcal{B}$ as required.

2.5 Computational Adequacy

To model divergence in JWA, we require the following structure.

Definition 2.6 A JWA model $(\mathcal{C}, \mathcal{N})$ on base $\mathcal{B}$ is *pointed* when it is equipped with a distinguished element $\bot_A \in \mathcal{N}(A)$ for each object A, such that $f; \bot_B = \bot_A$ for each $A \overset{f}{\succ} B$ in $\mathcal{C}$.

Clearly our game model is pointed: the $\bot$ morphism from an arena family $\{R_i\}_{i \in I}$ is given at $i \in I$ by the empty strategy.

We shall say that a pointed JWA model (equipped with a $\mathcal{B}$-isomorphism to interpret each recursive type) is *adequate* when $M \rightsquigarrow^{\omega}$ implies $[\![M]\!] = \bot$. Our aim is to show that our game model is adequate. We proceed as follows.

Definition 2.7 Let f be an endofunction on a set A.

(i) A sequence $(a_n)_{n \in \mathbb{N}}$ in A is a *fixed sequence* of f when $f(a_{n+1}) = a_n$ for all $n \in \mathbb{N}$.

(ii) A fixpoint a of f is *sequentially unique* when every fixed sequence of f is the constant sequence at a. (Clearly this implies uniqueness.)

Definition 2.8 (i) A JWA model $(\mathcal{C}, \mathcal{N})$ on base $\mathcal{B}$ is *ticking* when it is equipped with an endofunction $\checkmark_A$ on $\mathcal{N}(A)$ for each object A, such that

- $\checkmark_A$ has a sequentially unique fixpoint $\checkmark^\omega_A$, for each object A
- $\checkmark(f;g) = f;\checkmark(g)$ for each $A \xrightarrow{f} B \xrightarrow{g}$ in $\mathcal{C}$ and $\mathcal{N}$.

(Clearly this implies $f;\checkmark^\omega = \checkmark^\omega$ for each $A \xrightarrow{f} B$ in $\mathcal{C}$.)

(ii) A *tick-hiding* from a ticking JWA model $\mathcal{M}^\checkmark = (\mathcal{C}^\checkmark, \mathcal{N}^\checkmark)$ to a pointed JWA model $\mathcal{M} = (\mathcal{C}, \mathcal{N})$ on the same base $\mathcal{B}$ is a morphism $\mathcal{M}^\checkmark \xrightarrow{\alpha} \mathcal{M}$ in $\mathsf{JWA}(\mathcal{B})$ such that

- $\alpha(\checkmark(f)) = \alpha f$ for each $A \xrightarrow{f}$ in $\mathcal{N}^\checkmark$
- $\alpha(\checkmark^\omega_A) = \bot_A$ for each object A.

Proposition 2.9 *Let $\mathcal{M} = (\mathcal{C}, \mathcal{N})$ be a pointed JWA model on base $\mathcal{B}$. If there exists a ticking JWA model $\mathcal{M}^\checkmark = (\mathcal{C}^\checkmark, \mathcal{N}^\checkmark)$ on base $\mathcal{B}$ and a tick-hiding $\mathcal{M}^\checkmark \xrightarrow{\alpha} \mathcal{M}$, then $\mathcal{M}$ is adequate.*

To apply Prop. 2.9 to our game semantics, we define

Definition 2.10 (i) A *prior ticking play* is a prior play where each Proponent-move has a number attached (representing the number of ticks output by Proponent).

(ii) A *posterior ticking play* is either a posterior play where each Proponent-move has a number attached, or a prior ticking play followed by ω (representing infinitely many ticks).

(iii) A *ticking strategy* on an arena R is a prefix-closed set σ of posterior ticking plays that is deterministic: i.e. if $sm, sm' \in \sigma$ then $m = m'$. We write $\mathcal{S}^\checkmark(R)$ for the set of ticking strategies on R.

(iv) The *tick-hiding* of a ticking strategy σ on an arena R is the strategy obtained by discarding all the numbers of ticks in each play, and discarding all the plays that end in ω (they become divergences).

We then define the rest of the ticking model just as in Sect. 2.4. For composition, some plays ending in ω arise as the outer thread of an infinite interaction sequence, as in [11]. We omit details.

3 Adding Storage

To add storage to JWA, we use contexts of the form $\Delta;\ \Gamma$, where Δ is a list of distinct locations with associated type and Γ as before is a list of distinct identifiers with associated type. The syntax is given by Fig. 1 with Γ replaced by $\Delta;\ \Gamma$, and also by Fig. 4.

Again a renaming $\Gamma \overset{\theta}{\succ} \Gamma'$ maps identifiers to identifiers, and a substitution $\Gamma \overset{k}{\underset{\Delta}{\succ}} \Gamma'$ maps each $(\mathtt{x} : A) \in \Gamma$ to a value $\Delta;\ \Gamma' \vdash^{\mathsf{v}} k(\mathtt{x}) : A$. An *injection* $\Delta \overset{\phi}{\succ} \Delta'$ maps each location in Δ injectively to one of the same type in Δ'. These induce operations $\theta^\dagger$, k^* and $\phi^\dagger$ on terms.

We write $\Delta;\ \Gamma \vdash^{\mathsf{sn}} E$ to mean that E is a configuration that can arise during the execution of a command $\Delta;\ \Gamma \vdash^{\mathsf{n}} M$. It will consist of a list of local cells, a

$$\frac{\Delta;\ \Gamma \vdash^{\mathsf{v}} V : A \quad \Delta;\ \Gamma \vdash^{\mathsf{n}} M}{\Delta;\ \Gamma \vdash^{\mathsf{n}} \mathtt{n} := V.\ M}\ (\mathtt{n} : A) \in \Delta \qquad \frac{\Delta;\ \Gamma, \mathtt{x} : A \vdash^{\mathsf{n}} M}{\Delta;\ \Gamma \vdash^{\mathsf{n}} \mathtt{read\ n\ as\ x}.\ M}\ (\mathtt{n} : A) \in \Delta$$

$$\frac{\Delta, \overrightarrow{\mathtt{n} : A};\ \Gamma \vdash^{\mathsf{v}} \overrightarrow{V : A} \qquad \Delta, \overrightarrow{\mathtt{n} : A};\ \Gamma \vdash^{\mathsf{n}} M}{\Delta;\ \Gamma \vdash^{\mathsf{n}} \mathtt{new}\ \overrightarrow{\mathtt{n} := V}.\ M}$$

Fig. 4. Syntax For State

$$\begin{array}{llll} \overrightarrow{A};\ s;\ \mathtt{n} := V.\ M & \rightsquigarrow\ \overrightarrow{A};\ & s[\mathtt{n} \mapsto V];\ & M \\ \overrightarrow{A};\ s;\ \mathtt{read\ n\ as\ x}.\ M & \rightsquigarrow\ \overrightarrow{A};\ & s;\ & M[s(\mathtt{n})/\mathtt{x}] \\ \overrightarrow{A};\ s;\ \mathtt{new}\ \overrightarrow{\mathtt{n} := V}.\ M & \rightsquigarrow\ \overrightarrow{A}, \overrightarrow{A'};\ & \overrightarrow{\mathtt{n}} s, \overrightarrow{\mathtt{n} \mapsto V};\ & M \end{array}$$

Fig. 5. Transitions For Storage

global state, a local state and a command, as follows.

$$\frac{\overrightarrow{\mathtt{n} : A}, \overrightarrow{\mathtt{n}' : A'};\ \Gamma \vdash^{\mathsf{v}} \overrightarrow{V : A}, \overrightarrow{V' : A'} \qquad \overrightarrow{\mathtt{n} : A}, \overrightarrow{\mathtt{n}' : A'};\ \Gamma \vdash^{\mathsf{n}} M}{\overrightarrow{\mathtt{n} : A};\ \Gamma \vdash^{\mathsf{sn}} \overrightarrow{\mathtt{n}' : A'};\ \overrightarrow{\mathtt{n} \mapsto V},\ \overrightarrow{\mathtt{n}' \mapsto V'};\ M}$$

We define operational semantics for commands in a fixed context $\Delta;\ \Gamma$. The transitions are those in Fig. 2 (leaving the store unchanged) and those in Fig. 5. The terminal configurations are as in Fig. 2, with any store. To execute a command $\Delta;\ \Gamma \vdash^{\mathsf{n}} M$ in a given *global state*, mapping each $(\mathtt{n} : A) \in \Delta$ to a value $\Delta;\ \Gamma \vdash^{\mathsf{v}} V : A$, we begin with $\varepsilon; s; M$ and follow the transitions.

For understanding the operational semantics, it is convenient to assume that, given a storage context Δ, each new cell is named in a canonical way. But in fact, the choice does not matter, because within the configuration $\overrightarrow{\mathtt{n}' : A'};\ \overrightarrow{\mathtt{n} \mapsto V},\ \overrightarrow{\mathtt{n}' \mapsto V'};\ M$ the identifiers $\overrightarrow{\mathtt{n}'}$ are bound.

The equational theory of JWA with store is given by Fig. 3 together with Fig. 6. We extend it to configurations by taking Fig. 5 and the "exchange" equation

$$\begin{array}{l} \overrightarrow{\mathtt{n}}, \mathtt{p}, \mathtt{p}', \overrightarrow{\mathtt{n}'};\ s,\ \overrightarrow{\mathtt{n} \mapsto V}, \mathtt{p} \mapsto W, \mathtt{p}' \mapsto W', \overrightarrow{\mathtt{n}' \mapsto V'};\ M\ = \\ \overrightarrow{\mathtt{n}}, \mathtt{p}', \mathtt{p}, \overrightarrow{\mathtt{n}'};\ s,\ \overrightarrow{\mathtt{n} \mapsto V}, \mathtt{p}' \mapsto W', \mathtt{p} \mapsto W, \overrightarrow{\mathtt{n}' \mapsto V'};\ M \end{array}$$

It is important to note the limitations on structural rules for terms and configurations in context $\Delta;\ \Gamma$.

- We do not have semantically meaningful *contraction* in Δ. For example, the equation

$$\begin{array}{r} \mathtt{n} : \mathtt{bool}, \mathtt{n}' : \mathtt{bool};\ \mathtt{k} : \neg 1 \vdash^{\mathsf{n}} \mathtt{n} := \mathtt{true}.\ \mathtt{n}' := \mathtt{false}.\ \mathtt{k}\langle\rangle \\ = \mathtt{n}' := \mathtt{false}.\ \mathtt{n} := \mathtt{true}.\ \mathtt{k}\langle\rangle \end{array}$$

$$\begin{aligned}
\mathtt{read\ n\ as\ x.\ n} &:= \mathtt{x}.\ {}^{\mathtt{x}}M = M \\
\mathtt{read\ n\ as\ x.\ read\ n\ as\ y.}\ M &= \mathtt{read\ n\ as\ z.}\ M[\mathtt{z/x,z,y}] \\
\mathtt{n} := V.\ \mathtt{n} := W.\ M &= \mathtt{n} := W.\ M \\
\mathtt{n} := V.\ \mathtt{read\ n\ as\ x.}\ M &= \mathtt{n} := V.\ M[V/\mathtt{x}] \\
\mathtt{read\ n\ as\ x.\ read\ n'\ as\ y.}\ M &= \mathtt{read\ n'\ as\ y.\ read\ n\ as\ x.}\ M \quad (\mathtt{n} \neq \mathtt{n}') \\
\mathtt{n} := V.\ \mathtt{n}' := W.\ M &= \mathtt{n}' := W.\ \mathtt{n} := V.\ M \quad (\mathtt{n} \neq \mathtt{n}') \\
\mathtt{n} := V.\ \mathtt{read\ n'\ as\ x.} M &= \mathtt{read\ n'\ as\ x.\ n} := {}^{\mathtt{x}}V.\ M \quad (\mathtt{n} \neq \mathtt{n}') \\
\mathtt{new}\ \overrightarrow{\mathtt{n} := V}, \mathtt{p} := W, \mathtt{p}' = W', \overrightarrow{\mathtt{n}' := V}.\ M &= \mathtt{new}\ \overrightarrow{\mathtt{n} := V}, \mathtt{p}' := W', \mathtt{p} = W, \overrightarrow{\mathtt{n}' := V}.\ M \\
\mathtt{new}\ \overrightarrow{\mathtt{n} := V}, \mathtt{m} := V'.\ \mathtt{m} := W.\ M &= \mathtt{new}\ \overrightarrow{\mathtt{n} := V}, \mathtt{m} := W.\ M \\
\mathtt{new}\ \overrightarrow{\mathtt{n} := V}, \mathtt{m} := V'.\ \mathtt{read\ m\ as\ x.}\ M &= \mathtt{new}\ \overrightarrow{\mathtt{n} := V}, \mathtt{m} := V'.\ M[V'/\mathtt{x}] \\
\mathtt{new}\ \overrightarrow{\mathtt{n} := V}.\ \mathtt{m} := {}^{\overrightarrow{\mathtt{n}}}W.\ M &= \mathtt{m} := W.\ \mathtt{new}\ \overrightarrow{\mathtt{n} := V}.M \\
\mathtt{new}\ \overrightarrow{\mathtt{n} := V}.\ \mathtt{read\ m\ as\ x.}\ M &= \mathtt{read\ m\ as\ x.\ new}\ \overrightarrow{\mathtt{n} := {}^{\mathtt{x}}V}.\ M \\
\mathtt{new}\ \overrightarrow{\mathtt{n} := V}.\ {}^{\overrightarrow{\mathtt{n}}}M &= M \\
\mathtt{new}\ \overrightarrow{\mathtt{n} := V}.\ \mathtt{new}\ \overrightarrow{\mathtt{m} := W}.\ M &= \mathtt{new}\ \overrightarrow{\mathtt{n} := {}^{\overrightarrow{\mathtt{m}}}V}, \overrightarrow{\mathtt{m} := W}.\ M
\end{aligned}$$

Fig. 6. Equations for storage (cf. [18])

is provable, but renaming both $\mathtt{n}$ and $\mathtt{n}'$ as $\mathtt{n}'' : \mathtt{bool}$ makes the theory inconsistent.

- For configurations, we also lack *weakening* in Δ. For example, there is a configuration

$$;\mathtt{k} : \neg 1 \vdash^{\mathsf{sn}} \varepsilon; \varepsilon; \mathtt{k}\langle\rangle$$

but no configuration in context $\mathtt{n} : 0; \mathtt{k} : \neg 1$.

3.1 Modelling Injections

We want to model each syntactic category: types, contexts, storage contexts, values, commands, configurations, renamings, substitutions and injections. Storage contexts are interpreted using the finite products in the base $\mathcal{B}$, just like ordinary contexts. Renamings are interpreted in $\mathcal{B}^{\mathrm{op}}$, but what about injections? An elegant solution was provided by Oles and Reynolds [15].

Definition 3.1 Let $\mathcal{A}$ be a cartesian category.

(i) An *expansion* $A \overset{(r,*)}{\rightarrowtail} B$ consists of a "read" morphism $B \overset{r}{\rightarrowtail} A$ and an "update" morphism $B \times A \overset{*}{\rightarrowtail} B$ satisfying[5]

$$\begin{aligned}
\forall b \in B, a \in A.\ r(b * a) &= a \\
\forall b \in B.\ b * r(b) &= b \\
\forall b \in B, a, a' \in A.\ (b * a) * a' &= b * a'
\end{aligned}$$

(ii) The *composite* of expansions $A \overset{(r,*)}{\rightarrowtail} B \overset{(r',*')}{\rightarrowtail} C$ is $A \overset{(r'',*'')}{\rightarrowtail} C$ where

[5] These equations represent commutative diagrams in the standard way. We write the binary operation $*$ in infix style.

$$r''(c) \stackrel{\text{def}}{=} r(r'(c))$$
$$c *'' a \stackrel{\text{def}}{=} c *' (r'(c) * a)$$

and the *identity* expansion on A is $(\mathsf{id}_A, \pi'_{A,A})$.

(iii) We write $\mathsf{expan}(\mathcal{A})$ for the category of objects of $\mathcal{A}$ and expansions. It is a *coaffine category* i.e. a symmetric monoidal category (under $\times$) whose unit is an initial object. Hence, by coaffine coherence [16], we can interpret injections in it.

Any isomorphism $B \xrightarrow{\alpha} A \times Q$ gives an expansion $A \xrightarrow{e} B$

read $\quad B \overset{\alpha}{\succ} A \times Q \overset{\pi}{\succ} A$

update $\quad B \times A \overset{\alpha \times A}{\succ} (A \times Q) \times A \overset{\langle \pi', (\pi;\pi') \rangle}{\succ} A \times Q \overset{\alpha^{-1}}{\succ} B$

We say that (Q, α) is a *quotient* of e. A *morphism* between quotients is $Q \overset{f}{\succ} R$ such that $\alpha;\ (A \times f) = \beta$. (This guarantees that (Q, α) and (R, β) give the same expansion.) So any expansion has a category of quotients.

Definition 3.2 Let $\mathcal{A}$ be a cartesian category $\mathcal{A}$ with a strict initial object. We say $\mathcal{A}$ has *nonsingular quotients* when every expansion from a non-initial object has a quotient that is unique up to unique morphism (and hence, in the usual manner, unique up to isomorphism).

Proposition 3.3 *Both* CSet *(the category of countable sets) [15] and* $\mathsf{fam}(\mathbf{TokCh}^{\mathsf{op}})$ *have nonsingular quotients.*

A storage context denote an object of CSet in the case of ground store, and $\mathsf{fam}(\mathbf{TokCh}^{\mathsf{op}})$ in the case of general store, and we shall see that Prop. 3.3 enables us to define the requisite structure in terms of products, rather than in terms of expansions.

In general, for an object D and object sequence $\overrightarrow{A} = A_0, \ldots, A_{n-1}$, we write $e_{D:\overrightarrow{A}}$ for the expansion from D to the left-associated product $D \times A_0 \cdots \times A_{n-1}$ given by induction on n in the evident way. This is useful, because a storage context $\Delta, \overrightarrow{\mathtt{n} : A}$ denotes the left-associated product $[\![\Delta]\!] \times [\![A_0]\!] \times \cdots \times [\![A_{n-1}]\!]$.

3.2 Configurations and their categorical structure

By way of motivation for our categorical semantics, we note some pertinent facts.

Proposition 3.4 *Let* $\Delta = \overrightarrow{\mathtt{n} : A}$. *The map from commands* $\Delta; \Gamma \vdash^{\mathsf{n}} M$ *to configurations* $\Delta; \Gamma, \overrightarrow{\mathtt{x} : A} \vdash^{\mathsf{sn}} E$ *that maps* M *to* $\varepsilon;\ \overrightarrow{\mathtt{n} \mapsto \mathtt{x}};\ \overrightarrow{\mathtt{x}} M$ *is a bijection up to provable equality.*

This suggests that configurations can be regarded as the primitive entity, and commands as a derived one. That is quite reasonable: whereas the behaviour of a

command is dependent on an initial state, a configuration has just one behaviour. We next consider some operations on configurations.

- Any configuration $\Delta;\ \Gamma \vdash^{\mathsf{sn}} E$ can be converted into a configuration $\Delta, \overrightarrow{\mathtt{n} : A};\ \Gamma, \overrightarrow{\mathtt{x} : A} \vdash^{\mathsf{sn}} E'$ by injective renaming. The additional cells are initialized by $\overrightarrow{\mathtt{n} \mapsto \mathtt{x}}$.
- Any configuration $\Delta, \overrightarrow{\mathtt{n} : A};\ \Gamma \vdash^{\mathsf{sn}} E$ can be converted into a configuration $\Delta;\ \Gamma \vdash^{\mathsf{sn}} E'$ by *hiding* the global cells $\overrightarrow{\mathtt{n}}$ i.e. making them local.
- More generally, for any injection $\Delta \xrightarrow{\phi} \Delta'$, a configuration $\Delta';\ \Gamma \vdash^{\mathsf{sn}} E$ can be converted into a configuration $\Delta;\ \Gamma \vdash^{\mathsf{sn}} E'$. The order of hiding is immaterial, up to provable equality.

In the following definition, $\mathcal{C}$ homsets should be thought of as values (or substitutions), and $\mathcal{E}$ homsets should be thought of as configurations.

Definition 3.5 Let $\mathcal{B}$ be a base category with nonsingular quotients.

A *JWA model with global state* on $\mathcal{B}$ consists of

- a first-order JWA model $(\mathcal{C}^D, \mathcal{E}^D)$ functorial in $D \in \mathsf{Isos}\ \mathcal{B}$—more precisely: a functor
$$\mathsf{Isos}\ \mathcal{B} \xrightarrow{(\mathcal{C}^-, \mathcal{E}^-)} \mathsf{FOJWA}(\mathcal{B})$$
- an isomorphism
$$\mathcal{E}^D(D \times (A \times B)) \cong \mathcal{C}^D(A, \neg B) \quad \text{natural in } D, B \in \mathsf{Isos}\ \mathcal{B},\ A \in (\mathcal{C}^D)^{\mathsf{op}}$$
- functions
$$\mathcal{C}^D(A,B) \xrightarrow{\Upsilon_{\mathsf{v}}^{D,P}(A,B)} \mathcal{C}^{D\times P}(A,B) \qquad \text{natural in } D, P \in \mathsf{Isos}\ \mathcal{B},\ A \in \mathcal{B}^{\mathsf{op}},\ B \in \mathcal{B}$$
$$\mathcal{E}^D(A) \xrightarrow{\Upsilon^{D,P}(A)} \mathcal{E}^{D\times P}(A \times P) \qquad \text{natural in } D, P \in \mathsf{Isos}\ \mathcal{B},\ A \in \mathcal{B}^{\mathsf{op}}$$

such that

- $\mathcal{C}^D \xrightarrow{\Upsilon_{\mathsf{v}}^{D,P}} \mathcal{C}^{D\times P}$ is a functor
- Υ preserves composition in the sense that

$$\begin{array}{ccc} \mathcal{C}^D(A,B) \times \mathcal{E}^D(B) & ; & \mathcal{E}^D(A) \\ \downarrow{\scriptstyle \Upsilon_{\mathsf{v}} \times \Upsilon} & & \downarrow{\scriptstyle \Upsilon} \\ \mathcal{C}^{D\times P}(A,B) \times \mathcal{E}^{D\times P}(B \times P) & & \\ \downarrow{\scriptstyle (-\times P)\times \mathsf{id}} & & \\ \mathcal{C}^{D\times P}(A \times P, B \times P) \times \mathcal{E}^{D\times P}(B \times P) & ; & \mathcal{E}^{D\times P}(A \times P) \end{array}$$

- Υ acts monoidally in the sense that

$$\begin{array}{ccc} \mathcal{E}^D(A) & & \\ {\scriptstyle \Upsilon^{D,1}(A)}\downarrow & \downarrow\cong & \\ \mathcal{E}^{D\times 1}(A\times 1) & & \end{array} \qquad \begin{array}{ccc} \mathcal{E}^D(A) & & \\ \downarrow{\scriptstyle \Upsilon^{D,P\times Q}(A)} & & \\ \mathcal{E}^{D\times(P\times Q)}(A\times(P\times Q)) & \xrightarrow{\cong} & \mathcal{E}^{(D\times P)\times Q}((A\times P)\times Q) \end{array} \qquad \begin{array}{ccc} & \overset{\Upsilon^{D,P}(A)}{\succ} & \mathcal{E}^{D\times P}(A\times P) \\ & & \downarrow{\scriptstyle \Upsilon^{D\times P,Q}(A\times P)} \\ & & \end{array}$$

  and likewise for Υ_{v}

- Υ respects singularity in the sense that $\mathcal{E}^0(A) \overset{\Upsilon^{0,P}(A)}{\underset{\mathcal{E}^{\pi^{-1}}(\pi)}{\rightrightarrows}} \mathcal{E}^{0\times P}(A\times P)$ and likewise for Υ_{v}.

A *JWA model with global ground state* on $\mathcal{B}$ is defined similarly except that D, P, Q range over $\mathsf{Isos}\ \mathsf{CSet}$ instead of $\mathsf{Isos}\ \mathcal{B}$.

Given a JWA model with global state $\mathcal{M} = (\mathcal{C}, \mathcal{E}, \Upsilon_{\mathsf{v}}, \Upsilon)$ on $\mathcal{B}$, we can now recover the "commands" from "configurations". We define $\mathcal{N}^D(A) \stackrel{\text{def}}{=} \mathcal{E}^D(D\times A)$, and define $\mathcal{M}^D$ to be the JWA model $(\mathcal{C}^D, \mathcal{N}^D)$. Just as for values, we can define weakening maps $\mathcal{N}^D(A) \overset{\Upsilon_{\mathsf{n}}^{D,P(A)}}{\succ} \mathcal{N}^{D\times P}(A)$ to be the composite

$$\mathcal{E}^D(D\times A) \overset{\Upsilon}{\succ} \mathcal{E}^{D\times P}((D\times A)\times P) \overset{\cong}{\succ} \mathcal{E}^{D\times P}((D\times P)\times A)$$

For an expansion $D \overset{e}{\succ} D'$, we define a $\mathsf{JWA}(\mathcal{B})$ morphism $\mathcal{M}^D \overset{e^\dagger}{\succ} \mathcal{M}^{D'}$ by the composites

$$\mathcal{C}^D(A,B) \overset{\Upsilon_{\mathsf{v}}^{D,Q}(A,B)}{\succ} \mathcal{C}^{D\times Q}(A,B) \overset{\mathcal{C}^{\alpha^{-1}}(A,B)}{\succ} \mathcal{C}^{D'}(A,B)$$

$$\mathcal{N}^D(A) \overset{\Upsilon_{\mathsf{n}}^{D,Q}(A)}{\succ} \mathcal{N}^{D\times Q}(A) \overset{\mathcal{C}^{\alpha^{-1}}(A)}{\succ} \mathcal{N}^{D'}(A,B)$$

where (Q,α) is any quotient of e. Note that this is independent of the particular choice of quotient, by naturality and (in the case A is initial) the singularity respecting property. And so $\mathcal{M}^D$ is functorial in $D \in \mathsf{expan}(\mathcal{B})$.

Now we can proceed to interpret terms. For a given storage context $\Delta = \overrightarrow{\mathtt{n} : A}$, we interpret JWA with global state in Δ within the JWA model $\mathcal{M}^{[\![\Delta]\!]}$.

- For $\Delta;\ \Gamma, \mathtt{x} : A_i \vdash^{\mathsf{n}} M$, the command $\mathtt{read}\ \mathtt{n}_i\ \mathtt{as}\ \mathtt{x}.\ M$ denotes the composite in $\mathcal{E}^{[\![\Delta]\!]}$

$$[\![\Delta]\!] \times [\![\Gamma]\!] \overset{\langle \pi, \langle \pi', (\pi;\pi_i)\rangle\rangle}{\succ} [\![\Delta]\!] \times ([\![\Gamma]\!] \times [\![A_i]\!]) \overset{[\![M]\!]}{\succ}$$

- For $\Delta;\ \Gamma \vdash^{\mathsf{v}} V : A_i$ and $\Delta;\ \Gamma \vdash^{\mathsf{n}} M$, the command $\mathtt{n}_i := V.\ M$ denotes the composite in $\mathcal{E}^{[\![\Delta]\!]}$

$$[\![\Delta]\!] \times [\![\Gamma]\!] \overset{\langle q, \pi'\rangle}{\succ} [\![\Delta]\!] \times [\![\Gamma]\!] \overset{[\![M]\!]}{\succ}$$

where $[\![\Delta]\!] \times [\![\Gamma]\!] \xrightarrow{q} [\![\Delta]\!]$ has ith component of p is $\pi'; [\![V]\!]$ and jth component $\pi; \pi_j$ for each $j \neq i$

We then interpret configurations $\Delta;\ \Gamma \vdash^{\mathsf{sn}} E$—without local cells—in $\mathcal{E}^{[\![\Delta]\!]}([\![\Gamma]\!])$.

Definition 3.6 Let $\mathcal{B}$ be a base category with nonsingular quotients.

A *JWA model with state* on $\mathcal{B}$ consists of

- a JWA model with global state $(\mathcal{C}, \mathcal{E}, \Upsilon_{\mathsf{v}}, \Upsilon)$
- functions

$$\mathcal{E}^{D\times P}(A) \quad \curlywedge^{D,P}(A) \succ \mathcal{E}^{D}(A) \qquad\qquad \text{natural in } D, P \in \mathsf{Isos}\ \mathcal{B}, A \in \mathcal{B}^{\mathsf{op}}$$

such that

- $\curlywedge$ agrees with composition in the sense that

$$\begin{array}{ccc} \mathcal{C}^{D}(A,B) \times \mathcal{E}^{D\times P}(B) & \xrightarrow{\mathsf{id}\times\curlywedge} & \mathcal{C}^{D}(A,B) \times \mathcal{E}^{D}(B) \\ \downarrow{\scriptstyle \Upsilon_{\mathsf{v}}\times\mathsf{id}} & & \\ \mathcal{C}^{D\times P}(A,B) \times \mathcal{E}^{D\times P}(B) & & \downarrow{\scriptstyle ;} \\ \downarrow{\scriptstyle ;} & & \\ \mathcal{E}^{D\times P}(A) & \curlywedge & \mathcal{E}^{D}(A) \end{array}$$

- $\curlywedge$ acts monoidally and respects singularity in the same sense as Υ
- initializing some cells, then hiding them, has no effect:

$$\begin{array}{ccc} \mathcal{E}^{D}(A) & \xrightarrow{\Upsilon^{D,P}(A)} & \mathcal{E}^{D\times P}(A\times P) \\ & {\scriptstyle \mathcal{E}^{D}(\pi)}\searrow & \downarrow{\scriptstyle \curlywedge^{D,P}(A\times P)} \\ & & \mathcal{E}^{D}(A\times P) \end{array}$$

- initialization and hiding commute on distinct cells:

$$\begin{array}{ccccc} \mathcal{E}^{D\times P}(A) & \xrightarrow{\Upsilon^{D\times P,Q}(A)} & \mathcal{E}^{(D\times P)\times Q}(A\times Q) & \cong & \mathcal{E}^{(D\times Q)\times P}(A\times Q) \\ \downarrow{\scriptstyle \curlywedge^{D,P}(A)} & & & & \downarrow{\scriptstyle \curlywedge^{D\times Q,P}(A\times Q)} \\ \mathcal{E}^{D}(A) & & \xrightarrow[\Upsilon^{D,Q}(A)]{} & & \mathcal{E}^{D\times Q}(A\times Q) \end{array}$$

A *JWA model with ground state* is defined similarly except that D, P, Q range over $\mathsf{Isos}\ \mathsf{CSet}$ instead of $\mathsf{Isos}\ \mathcal{B}$.

Just as with global state, there is additional structure that can be derived from a JWA model with state $\mathcal{M} = (\mathcal{C}, \mathcal{E}, \Upsilon_{\mathsf{v}}, \Upsilon, \curlywedge)$ on $\mathcal{B}$. For any expansion $D \quad {}^{e}\!\succ D'$

object A, we define $\mathcal{E}^{D'}(A) \xrightarrow{e^{\ddagger}(A)} \mathcal{E}^{D}(A)$ to be the composite

$$\mathcal{E}^{D'}(A) \xrightarrow{\mathcal{E}^{\alpha}(A)} \mathcal{E}^{D\times Q}(A) \xrightarrow{\curlywedge^{D,Q}(A)} \mathcal{E}^{D}(A)$$

where (Q, α) is any quotient of e—again, the choice of quotient does not affect the definition. And so $\mathcal{E}^{D}(A)$ is functorial in $D \in \mathsf{expan}(\mathcal{B})^{\mathsf{op}}$ and $A \in \mathcal{B}^{\mathsf{op}}$.

We can now complete our semantics of terms. Given terms $\Delta, \overrightarrow{\mathtt{n} : A};\ \Gamma \vdash^{\mathsf{v}} \overrightarrow{V : A}$ and $\Delta, \overrightarrow{\mathtt{n} : A};\ \Gamma \vdash^{\mathsf{n}} M$, the denotation $[\![\mathtt{new}\ \mathtt{n} := \overrightarrow{V}.\ M]\!]$ is obtained by forming the composite

$$[\![\Delta]\!] \times [\![\Gamma]\!] \quad {}^{\langle\langle \pi : \pi'; \overrightarrow{[\![V]\!]}\rangle, \pi'\rangle} \succ [\![\Delta, \overrightarrow{\mathtt{n} : A}]\!] \times [\![\Gamma]\!] \quad {}^{[\![M]\!]} \succ$$

over $[\![\Delta, \overrightarrow{\mathtt{n} : A}]\!]$, then applying $e^{\ddagger}_{[\![\Delta]\!]:\overrightarrow{[\![A]\!]}}$ to obtain a morphism over $[\![\Delta]\!]$.

That completes our semantics of terms. A substitution $\Gamma \ {}^{k}_{\Delta}\!\succ \Gamma'$ denotes a $\mathcal{C}$-morphism $[\![\Gamma']\!] \ {}^{[\![k]\!]}_{[\![\Delta]\!]}\!\succ [\![\Gamma]\!]$ in the usual way.

Proposition 3.7 *Let P be a term (command or value) in context $\Delta;\ \Gamma$.*

(i) *(Substitution) For any substitution $\Gamma \ {}^{k}_{\Delta}\!\succ \Gamma'$, we have $[\![k^* P]\!] = [\![k]\!]; [\![P]\!]$ in $\mathcal{M}^{[\![\Delta]\!]}$.*

(ii) *(Injective renaming) For any injection $\Delta \ {}^{\phi}\!\succ \Delta'$ we have $[\![\phi^\dagger P]\!] = [\![\phi]\!]^\dagger [\![P]\!]$.*

We proceed to semantics of configurations. If $\Delta = \overrightarrow{\mathtt{n} : A}$ and we have values $\Delta, \overrightarrow{\mathtt{n}' : A'};\ \Gamma \vdash^{\mathsf{v}} \overrightarrow{V : A}, \overrightarrow{V' : A'}$ and a command $\Delta, \overrightarrow{\mathtt{n}' : A'};\ \Gamma \vdash^{\mathsf{n}} M$ then the denotation $[\![\overrightarrow{\mathtt{n}' : A'};\ \overrightarrow{\mathtt{n} \mapsto V},\ \overrightarrow{\mathtt{n}' \mapsto V'};\ M]\!]$ is obtained by first forming the composite

$$[\![\Gamma]\!] \quad {}^{\langle\langle \overrightarrow{[\![V]\!]}, \overrightarrow{[\![V']\!]}\rangle, \mathsf{id}\rangle} \succ [\![\Delta, \overrightarrow{A'}]\!] \times [\![\Gamma]\!] \quad {}^{[\![M]\!]} \succ$$

over $[\![\Delta, \overrightarrow{\mathtt{n} : A}]\!]$, then applying $e^{\ddagger}_{[\![\Delta]\!]:\overrightarrow{[\![A]\!]}}$ to obtain a morphism over $[\![\Delta]\!]$.

Proposition 3.8 *In any JWA model with state, the interpretation of values, commands and configurations validates the equational theory.*

3.3 Game Semantics Of State

In this section, we define a JWA model with state on base $\mathsf{fam}(\mathbf{TokCh}^{\mathsf{op}})$. First we fix an arena family $D = \{U_l\}_{l\in L}$ in order to define $(\mathcal{C}^D, \mathcal{E}^D)$. We modify Def. 2.3:

Definition 3.9 Let R be an arena.

(i) A (finite) *justified sequence* on an arena R in store context D is a sequence $m_0, \ldots, m_{n-1}$ where each $m_i = (p_i, l_i, r_i)$ consists of a *pointer* $* \leqslant p_i < i$, a *state element* $l_i \in L$ and an element $r_i \in (\biguplus_{l\in L} U_l) \uplus R$ such that either

- $r_{p_i} \vdash r_i$, where $r_* \stackrel{\text{def}}{=} *$, or
- $p_i \neq *$ and $r_i \in \mathsf{rt}\, U_{l_{p_i}}$.

(ii) *Plays* and *strategies* are unchanged, and $\mathcal{S}^D(R)$ is the set of strategies on R in store context D. We write $\mathcal{S}^D_{\mathsf{HO}}(R)$ for $\prod_{l \in L, b \in \mathsf{rt}\, R} \mathcal{S}(U_l \uplus R_b)$.

For arenas R and S, we define the homset

$$\mathcal{G}^D(R,S) \stackrel{\text{def}}{=} \mathcal{S}^D_{\mathsf{HO}}(R \to_{\mathsf{HO}} S) \cong \prod_{l \in L, b \in \mathsf{rt}\, S} \mathcal{S}^D(U_l \uplus R \uplus S_b)$$

Copycat is defined as above, except that Proponent must also copy the state element, and must copy any moves that explore D. Composition is as before, except that each thread includes root moves in D pointing to its moves, and all their descendants. We now define

$$\mathcal{C}^D \stackrel{\text{def}}{=} \mathsf{fam}(\mathcal{G}^D)$$
$$\mathcal{E}^D \stackrel{\text{def}}{=} \mathsf{fam}(\mathcal{S}^D)$$

Our next task is to define the weakening operation.

Definition 3.10 Let $D = \{T_k\}_{k \in K}$ and $P = \{U_l\}_{l \in L}$ be arena families.

(i) Let R be an arena, let $\hat{l} \in L$ be given, and let s be a play on $R \uplus U_{\hat{l}}$ in storage context $D \times P$. We say s is *weakened* (wrt $D, P, \hat{l}$) when for each Proponent move m with state element $\langle k, l \rangle$
- $l = \hat{l}$ if $m = 0$, otherwise l is the L component of the state element of $m - 1$
- if either $m = 0$ or move $m-1$ plays an element of $R \cup D$ (i.e. not an element of P), then so does move m
- any Opponent-move n pointing to m and playing $b \in \mathsf{rt}\, U_l$ and state element $\langle k', l' \rangle$ is followed by a Proponent-move $n+1$ pointing to $m-1$ (or $*$ if $m = 0$) and playing b with state element $\langle k', l' \rangle$; and any Opponent-move p pointing to a descendant $q + 1$ of n or $n + 1$ and playing c (necessarily in D or P) with state element $\langle k'', l'' \rangle$ is followed by a Proponent-move $p+1$ pointing to q and playing c with state element $\langle k'', l'' \rangle$.

If s is weakened, its *outer thread* is the play on R in storage context D given by all the moves of s that are in R and D, with only the K-component of each state element. (This is a posterior play if s is.)

(ii) Let R be an arena and let $\hat{l} \in L$ be given. We define $\mathcal{S}^D R \overset{\Upsilon^{D,P}}{\rightarrowtail} \mathcal{S}^{D \times P}(R \uplus U_{\hat{l}})$ to map σ to the strategy consisting of every weakened (wrt $D, P, \hat{l}$) posterior play whose outer thread is in σ.

(iii) From $\curlyvee$ we derive maps

$$\mathcal{S}^D_{\mathsf{HO}} R \qquad \succ \mathcal{S}^{D\times P}_{\mathsf{HO}} R$$

$$\mathcal{G}^D(R,S) \qquad \succ \mathcal{G}^{D\times P}(R,S)$$

$$\mathcal{C}^D(A,B) \qquad \curlyvee_{\mathsf{v}}^{D,P}(A,B) \succ \mathcal{C}^{D\times P}(A,B) \qquad \text{where } \mathcal{C}^D = \mathsf{fam}(\mathcal{G}^D)$$

$$\mathcal{E}^D(A) \qquad \curlyvee^{D,P}(A) \succ \mathcal{E}^{D\times P}(A\times P) \qquad \text{where } \mathcal{E}^D = \mathsf{fam}(\mathcal{S}^D)$$

in the evident way.

Finally we have to define hiding.

Definition 3.11 Let $D = \{T_k\}_{k\in K}$ and $P = \{U_l\}_{l\in L}$ be arena families.

(i) Let s be a play on an arena R in storage context $D \times P$. We say that s is *hideable* when for every Opponent move $m+1$ with state element $\langle k, l\rangle$
- l is the L-component of the state element of move m
- if m plays an element of $R \cup D$, then so does $m+1$
- any Proponent move n pointing to $m+1$ and playing $b \in \mathsf{rt}\, U_l$ with state element $\langle k', l'\rangle$ is followed by an Opponent move $n+1$ pointing to m and playing b with state element $\langle k', l'\rangle$; and any Proponent move pointing to a descendant $q+1$ of n or $n+1$ and playing c (necessarily in D or P) with state element $\langle k'', l''\rangle$ is followed by an Opponent move pointing to q and playing c with state element $\langle k'', l''\rangle$.

If s is hideable, its *outer thread* is the play on R in storage context D given by all moves of s that are in R and D, with only the K-component of each state element. (This is a posterior play iff s is.)

(ii) For any arena R we define a map $\mathcal{S}^{D\times P}(R) \quad \curlywedge^{D,P}(R) \succ \mathcal{S}^D(R)$ mapping σ to the set consisting of the outer thread of every hideable posterior play in σ.

(iii) We define $\mathcal{E}^{D\times P}(A) \quad \curlywedge^{D,P}(A) \succ \mathcal{E}^D(A)$ from $\curlywedge$ using the families construction.

Proposition 3.12 *The game model* $(\mathcal{C}, \mathcal{E}, \curlyvee_{\mathsf{v}}, \curlyvee, \curlywedge)$ *is a JWA model with state, on base* $\mathsf{fam}(\mathbf{TokCh}^{\mathsf{op}})$.

3.4 Computational Adequacy

Let $\mathcal{B}$ be a base category with nonsingular quotients.

Definition 3.13 A JWA model with state $(\mathcal{C}, \mathcal{E}, \curlyvee_{\mathsf{v}}, \curlyvee, \curlywedge)$ on base $\mathcal{B}$ is *pointed* when it is equipped with a distinguished element $\bot_A \in \mathcal{N}(A)$ for each object A,

Clearly our game model is pointed: the $\bot$ morphism from an arena family $\{R_i\}_{i\in I}$ is given at $i \in I$ by the empty strategy.

We shall say that a pointed JWA model (equipped with a $\mathcal{B}$-isomorphism to interpret each recursive type) is *adequate* when $E \rightsquigarrow^{\omega}$ implies $[\![E]\!] = \bot$. Our aim is to show that our game model is adequate. We proceed as follows.

Definition 3.14 (i) We write $\mathsf{JWAS}(\mathcal{B})$ for the category of JWA models with state on base $\mathcal{B}$. Morphisms are identity-on-objects and preserve all structure on the nose.

(ii) A JWA model with state $(\mathcal{C}, \mathcal{E}, \curlyvee_{\mathsf{v}}, \curlyvee, \curlywedge)$ on base $\mathcal{B}$ is *ticking* when it is equipped with an endofunction $\checkmark_A$ on $\mathcal{N}(A)$ for each object A, such that
- $\checkmark_A$ has a sequentially unique fixpoint $\checkmark_A^{\omega}$, for each object A
- $\checkmark_A$ is preserved by precomposition with a $\mathcal{C}$-morphism, by $\curlyvee$ and by $\curlywedge$.

(iii) A *tick-hiding* from a ticking JWA model with state $\mathcal{M}^{\checkmark} = (\mathcal{C}^{\checkmark}, \mathcal{E}^{\checkmark}, \curlyvee_{\mathsf{v}}^{\checkmark}, \curlyvee^{\checkmark}, \curlywedge^{\checkmark})$ to a pointed JWA model with state $\mathcal{M} = (\mathcal{C}, \mathcal{E}, \curlyvee_{\mathsf{v}}, \curlyvee, \curlywedge)$ on the same base $\mathcal{B}$ is a morphism $\mathcal{M}^{\checkmark} \overset{\alpha}{\succ} \mathcal{M}$ in $\mathsf{JWAS}(\mathcal{B})$ such that
- $\alpha(\checkmark(f)) = \alpha f$ for each $A \overset{f}{\succ}$ in $\mathcal{N}^{\checkmark}$
- $\alpha(tick_A^{\omega}) = \bot_A$ for each object A.

Proposition 3.15 *Let $\mathcal{M} = (\mathcal{C}, \mathcal{E}, \curlyvee_{\mathsf{v}}, \curlyvee, \curlywedge)$ be a pointed JWA model on base $\mathcal{B}$. If there exists a ticking JWA model $\mathcal{M}^{\checkmark} = (\mathcal{C}^{\checkmark}, \mathcal{E}^{\checkmark}, \curlyvee_{\mathsf{v}}^{\checkmark}, \curlyvee^{\checkmark}, \curlywedge^{\checkmark})$ on base $\mathcal{B}$ and a tick-hiding $\mathcal{M}^{\checkmark} \overset{\alpha}{\succ} \mathcal{M}$, then $\mathcal{M}$ is adequate.*

We apply Prop. 3.15 to prove the adequacy of the game model just as in Sect. 2.5.

4 Further Work

We have now constructed a model of state and proved adequacy. Some remaining tasks are as follows.

(i) To construct a model for a direct style calculus; this simply follows the construction in [10]. The treatment of storage is just as in this paper.

(ii) To show that for every storage context Δ and context Γ using finite sums, every computable element of $\mathcal{N}^{[\![\Delta]\!]}([\![\Gamma]\!])$ is definable by a command $\Delta; \Gamma \vdash^{\mathsf{n}} M$. Even in [], where Δ was empty, this was an open problem.

(iii) To show that the observational preorder corresponds to inclusion of complete traces.

(iv) To extend the model to include `ref` types, so as to reorganize the model of [9].

References

[1] Abramsky, S., K. Honda and G. McCusker, *A fully abstract game semantics for general references*, in: *Proceedings, 13th Annual IEEE Symposium on Logic in Computer Science*, 1998, pp. 334–344.

[2] Abramsky, S. and G. McCusker, *Linearity, sharing and state: a fully abstract game semantics for Idealized Algol with active expressions*, in: P. W. O'Hearn and R. D. Tennent, editors, *Algol-like languages* (1997).

[3] Abramsky, S. and G. McCusker, *Call-by-value games*, in: M. Nielsen and W. Thomas, editors, *Computer Science Logic: 11th International Workshop Proceedings*, LNCS (1998), pp. 1–17.

[4] Benton, N. and B. Leperchey, *Relational reasoning in a nominal semantics for storage.*, in: *Typed Lambda Calculi and Applications, 7th International Conference, TLCA 2005, Nara, Japan, April 21-23, 2005, Proceedings*, Lecture Notes in Computer Science **3461** (2005), pp. 86–101.

[5] Bohr, N. and L. Birkedal, *Relational reasoning for recursive types and references*, in: N. Kobayashi, editor, *APLAS*, Lecture Notes in Computer Science **4279** (2006), pp. 79–96.
URL http://dx.doi.org/10.1007/11924661_5

[6] de Lataillade, J., "Quantification du second ordre en smantique des jeux - Application aux isomorphismes de types," Ph.D. thesis, Université Paris Diderot (Paris 7) (2007).

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

[8] Laird, J., *A categorical semantics of higher-order store*, in: *Proc., 9th Conference on Category Theory and Computer Science, Ottawa, 2002*, Electronic Notes in Theoretical Computer Science **69**, 2003.

[9] Laird, J., *A fully abstract trace semantics for general references*, in: L. Arge, C. Cachin, T. Jurdzinski and A. Tarlecki, editors, *ICALP*, Lecture Notes in Computer Science **4596** (2007), pp. 667–679.
URL http://dx.doi.org/10.1007/978-3-540-73420-8_58

[10] Levy, P. B., *Adjunction models for call-by-push-value with stacks*, Theory and Applications of Categories **14** (2005), pp. 75–110.

[11] Levy, P. B., *Infinite trace equivalence*, Annals of Pure & Applied Logic **151** (2008).

[12] McCusker, G., *On the semantics of the bad-variable constructor in Algol-like languages*, in: *Proceedings, MFPS 2003*, ENTCS **83**, 2003.

[13] Murawski, A. S., *Bad variables under control*, in: J. Duparc and T. A. Henzinger, editors, *CSL*, Lecture Notes in Computer Science **4646** (2007), pp. 558–572.
URL http://dx.doi.org/10.1007/978-3-540-74915-8_41

[14] O'Hearn, P. W. and R. D. Tennent, *Parametricity and local variables*, Journal of the ACM **42** (1995), pp. 658–709.

[15] Oles, F. J., *Functor categories and store shapes*, in: P. W. O'Hearn and R. D. Tennent, editors, *Algol-like languages*, Birkhaüser, 1997 .

[16] Petric, Z., *Coherence in substructural categories*, Studia Logica **70** (2002), pp. 271–296.

[17] Pitts, A. M. and I. D. B. Stark, *Observable properties of higher order functions that dynamically create local names, or: What's new?*, in: *Mathematical Foundations of Computer Science, Proc. 18th Intl. Symp., Gdańsk, 1993*, LNCS **711** (1993), pp. 122–141.

[18] Plotkin, G. and J. Power, *Notions of computation determine monads*, in: *Proceedings, Foundations of Software Science and Computation Structures, 2002*, LNCS **2303** (2002), pp. 342–356.

[19] Reddy, U. S., *Global state considered unnecessary: An introduction to object-based semantics*, Lisp and Symbolic Computation **9** (1996), pp. 7–76.

[20] Reynolds, J. C., *The essence of Algol*, in: J. W. de Bakker and J. C. van Vliet, editors, *Algorithmic Languages* (1981), pp. 345–372.

[21] Stark, I. D. B., "Names and Higher-Order Functions," Ph.D. thesis, University of Cambridge (1994).

[22] Tzevelekos, N., *Full abstraction for nominal general references*, in: *LICS* (2007), pp. 399–410.