A Typed Semantics for Languages with Higher-Order Store and Subtyping

JAN SCHWINGHAMMER

INFORMATICS, UNI ERSITY OF S SSEX, BRIGHTON, UK

j.schwinghammer@sussex.ac.uk

ABSTRACT. **We consider a call-by-value language, with higher-order functions, records, references to values of arbitrary type, and subtyping. We adapt an intrinsically typed denotational model for a similar language based on a possible-world semantics, recently given by Levy [29], and relate it to an untyped model by a logical relation. Following the methodology of Reynolds [45], this relation is used to establish coherence of the typed semantics, with a coercion interpretation of subtyping. Moreover, we demonstrate that this technique scales to ML-like polymorphic type schemes. We obtain a typed denotational semantics of (imperative) object-oriented languages, both class-based and object-based ones.**

Contents

mixed-variant recursive equation. So far, only few models of (typed) languages with general references appeared in the literature [5, 6, 29], and most of the work done on semantics of storage does not readily apply to languages with higher-order store [48].

In a recent paper, Paul Levynot

to the mixed-variant recursion forced by the higher-order store we can no longer use induction over the type structure to establish properties of

TABLE 1. Typing

$$\frac{\Gamma \mathbin{.} e : A \quad A \prec: B}{\Gamma \mathbin{.} e : B} \qquad \frac{x{:}A \in \Gamma}{\Gamma \mathbin{.} x : A}$$

$$\frac{\Gamma \mathbin{.} e\ \ : B \quad \Gamma; x{:}B \mathbin{.} e_2 : A}{\Gamma \mathbin{.} \text{let } x{=}e\ \ \text{in } e_2 : A} \qquad \frac{}{\Gamma \mathbin{.} \text{true} : \text{bool}}$$

$$\frac{\Gamma \mathbin{.} x : \text{bool} \quad \Gamma \mathbin{.} e\ \ : A \quad \Gamma \mathbin{.} e_2 : A}{\Gamma \mathbin{.} \text{if } x \text{ then } e\ \ \text{else } e_2 : A} \qquad \frac{}{\Gamma \mathbin{.} \text{false} : \text{bool}}$$

$$\frac{\Gamma \mathbin{.} x_i : A_i \ \ \forall i \in I}{\Gamma \mathbin{.} \{\text{m}_i = x_i\}_{i\ \ I} : \{\text{m}_i : A_i\}_{i\ \ I}} \qquad \frac{\Gamma \mathbin{.} x : \{\text{m}_i : A_i\}_{i\ \ I}}{\Gamma \mathbin{.} x{:}\text{m}_j : A_j}\ (j \in I)$$

$$\frac{\Gamma; x{:}A \mathbin{.} e : B}{\Gamma \mathbin{.}\ \ x{:}e : A \Rightarrow B} \qquad \frac{\Gamma \mathbin{.} x : A \Rightarrow B \quad \Gamma \mathbin{.} y : A}{\Gamma \mathbin{.} x(y) : B}$$

$$\frac{\Gamma \mathbin{.} x : A}{\Gamma \mathbin{.} \text{new}_A\ x : \text{ref } A} \qquad \frac{\Gamma \mathbin{.} x : \text{ref } A}{\Gamma \mathbin{.} \text{deref } x : A}$$

$$\frac{\Gamma \mathbin{.} x : \text{ref } A \quad \Gamma \mathbin{.} y : A}{\Gamma \mathbin{.} x{:=}y : 1}$$

STRUCTURE OF THE REPORT. In the next section, language and type system are introduced. Then, in Sects. 3 and 4, typed and untyped models are presented. The logical relation is defined next, and retractions between types of the intrinsic semantics and the untyped value space are used to prove coherence in Section 6. In Section 7 both a derived per semantics and the relation to our earlier work on an interpretation of objects are discussed. Section 8 presents the applications of the theory, providing a semantics of classes and objects, as well as an example specification and verification of a non-trivial program. In Section 9 the type system is enriched with (predicative) polymorphism and proved useful in obtaining a semantics of generic collection classes. Finally, Section 10 discusses related work.

2 Language

We consider a single base type of booleans, bool, records $\mathsf{f}\text{m}_\mathsf{i} : A_\mathsf{i}\mathsf{g}_{\mathsf{i}\in\mathsf{I}}$ with labels m 2 L, and function types A) B. We set $1 \overset{de}{=}$ fg for the (singleton) type of empty records. Finally, we have a type ref A of mutable references to values of type A. Term forms include constructs for creating, dereferencing and updating of storage locations. The syntax of types and terms is given by the grammar:

$$\begin{aligned}
\mathbf{A}; \mathbf{B} \in \ y \quad &::= \mathsf{bool} \mid \{\mathsf{m}_i : \mathbf{A}_i\}_{i \quad I} \mid \mathbf{A} \Rightarrow \mathbf{B} \mid \mathsf{ref}\ \mathbf{A} \\
\mathbf{v} \in \quad , &::= \mathbf{x} \mid \mathsf{true} \mid \mathsf{false} \mid \{\mathsf{m}_i = \mathbf{x}_i\}_{i \quad I} \mid \quad \mathbf{x}\mathbf{:}\mathbf{e} \\
\mathbf{e} \in \quad &::= \mathbf{v} \mid \mathsf{let}\ \mathbf{x}{=}\mathbf{e} \ \ \mathsf{in}\ \mathbf{e}_2 \mid \mathsf{if}\ \mathbf{x}\ \mathsf{then}\ \mathbf{e} \ \ \mathsf{else}\ \mathbf{e}_2 \mid \mathbf{x}\mathbf{:}\mathsf{m} \mid \mathbf{x}(\mathbf{y}) \\
&\quad \mid \mathsf{new}_A\ \mathbf{x} \mid \mathsf{deref}\ \mathbf{x} \mid \mathbf{x}{:=}\mathbf{y}
\end{aligned}$$

Subterms in most of these term forms are restricted to variables in order to simplify the statement of the semantics in the next section: There, we can exploit the fact that subterms that exhibit side-effects only appear in the let-construct. However, in subsequent examples we will use a more generous syntax. The reduction of such syntax sugar to the expressions above should always be immediate.

The subtyping relation $A \quad : B$ is the least reflexive and transitive relation closed under the rules

$$\frac{\mathbf{A}_i \prec: \mathbf{A}_i \ \ \forall \mathbf{i} \in \mathbf{I} \quad \mathbf{I} \ \subseteq \mathbf{I}}{\{\mathsf{m}_i : \mathbf{A}_i\}_{i \quad I} \prec: \{\mathsf{m}_i : \mathbf{A}_i\}_{i \quad I'}} \qquad \frac{\mathbf{A} \ \prec: \mathbf{A} \quad \mathbf{B} \prec: \mathbf{B}}{\mathbf{A} {\Rightarrow} \mathbf{B} \prec: \mathbf{A} \ {\Rightarrow} \mathbf{B}}$$

Note that there is no rule for reference types as these need to be invariant, i.e., ref $A \quad :$ ref B only if $A \quad B$. A type inference system is given in Table 1, where contexts Γ are finite sets of variable-type pairs, with each variable occurring at most once. As usual, in writing $\Gamma \quad :A$ we assume does not occur in Γ. A subsumption rule is used to for subtyping of terms.

3 Intrinsic Semantics

In this section we recall the possible worlds model of [29]. Its extension with records is straightforward, and we interpret the subsumption

Following [29] the semantics of types can now be obtained as minimal invariant of the locally continuous functor $: \mathsf{C}^{op}$ C ! C (derived from the domain equations for types by separating positive and negative occurrences of the store) given in Table 2. Here, C is the bilimit-compact category

$$\mathcal{C} \overset{def}{=} \mathbf{Q}_{w \ \mathcal{W}} \mathbf{pCpo} \times \mathbf{Q}_{A \ Type} [\mathcal{W}; \mathbf{Cpo}] \bullet\!\!\to [\mathcal{W}; \mathbf{pCpo}] \qquad \textbf{(2)}$$

where $[$W Cpo$]$! $[$W pCpo$]$ denotes the category with objects the functors $A \ B :$ W ! Cpo and morphisms the partial natural transformations $: A$! B, i.e., for $A \ B :$ W ! Cpo the diagram

$$\begin{tikzcd} \mathbf{A}_w \arrow[r, "\mu_w"] \arrow[d, "A_w^{w'}"'] & \mathbf{B}_w \arrow[d, "B_w^{w'}"] \\ \mathbf{A}_{w'} \arrow[r, "\mu_{w'}"'] & \mathbf{B}_{w'} \end{tikzcd} \qquad \textbf{(3)}$$

commutes. The rst component of the product in (2) is used to obtain ${}_{\mathsf{w}} \overset{de}{=} D_{\mathsf{Sw}}$ from the minimal invariant $D =$ hfD_{Sw}g$_{\mathsf{w}}$ fD_{A}g$_{\mathsf{A}}$i, and the second component yields $[\![A]\!] \overset{de}{=} D_{\mathsf{A}}$.

In fact, for every type A 2 Type the minimal invariant D provides isomorphisms $(D \ D)_{\mathsf{A}} = D_{\mathsf{A}}$ in the category $[$W Cpo$]$ of functors W ! Cpo and total natural transformations.

SEMANTICS. Each subtyping derivation $A \quad : B$ determines a coercion, which is in fact a (total) natural transformation from $[\![A]\!]$ to $[\![B]\!]$, de ned in Table 3: We follow the notation of [45] and write P$(\quad)$ to distinguish a derivation of judgement from the judgement itself.

In the following we write $[\![\Gamma]\!]_{\mathsf{w}}$ for the set of environments, i.e., maps from variables to $\bigcup_{\mathsf{A}} [\![A]\!]_{\mathsf{w}}$ s.t. $A \quad \_ \quad \_ \quad B \quad \_ \quad f\_$

TABLE 2. Defining $\mathbf{F} : \mathcal{C}^{op} \times \mathcal{C} \longrightarrow \mathcal{C}$

On $\mathcal{C}$-objects $\mathbf{D}; \mathbf{E}$

$$\begin{aligned}
\mathbf{F}(\mathbf{D};\mathbf{E})_{Sw} &= \prod_{l_A\ \ w} \mathbf{E}_{Aw} \\
\mathbf{F}(\mathbf{D};\mathbf{E})_{\ \ w} &= \mathsf{BVal} = \{true; false\} \\
\mathbf{F}(\mathbf{D};\mathbf{E})_{\ \ (w\ \ w')} &= \mathsf{id}_{\mathsf{B}} \\
\mathbf{F}(\mathbf{D};\mathbf{E})_{\{\ \ i:A_i\}w} &= \{\!|\mathsf{m}_i : \mathbf{E}_{A_i w}|\!\} \\
\mathbf{F}(\mathbf{D};\mathbf{E})_{\{\ \ i:A_i\}(w\ \ w')} &= \ \mathbf{r}{:}\{\!|\mathsf{m}_i = \mathbf{E}_{A_i(w\ \ w')}(\mathbf{r}{:}\mathsf{m}_i)|\!\} \\
\mathbf{F}(\mathbf{D};\mathbf{E})_{A\ \ Bw} &= \prod_{w'\ \ w}(\mathbf{D}_{Sw'} \times \mathbf{D}_{Aw'}\ \ *\ \ \sum_{w''\ \ w'}(\mathbf{E}_{Sw''} \times \mathbf{E}_{Bw''})) \\
\mathbf{F}(\mathbf{D};\mathbf{E})_{A\ \ B(w\ \ w')} &= \ \mathbf{f}\ \ \mathbf{w}\ \ \geq \mathbf{w} : \mathbf{f}_{w''} \\
\mathbf{F}(\mathbf{D};\mathbf{E})_{\ \ Aw} &= \{\mathbf{l}_A \mid \mathbf{l}_A \in \mathbf{w}\} \\
\mathbf{F}(\mathbf{D};\mathbf{E})_{\ \ A(w\ \ w')} &= \ \mathbf{l}{:}\mathbf{l}
\end{aligned}$$

On $\mathcal{C}$-morphisms $\mathbf{h} : \mathbf{D}\ \ \longrightarrow \mathbf{D}$ and $\mathbf{k} : \mathbf{E} \longrightarrow \mathbf{E}$ by

$$\begin{aligned}
\mathbf{F}(\mathbf{h};\mathbf{k})_{Sw} &= \ \mathbf{s}{:} \begin{cases} \mathbf{l}_A \mapsto \mathbf{k}_{Sw}(\mathbf{s})_{l_A} & \text{if } \mathbf{k}_{Sw}(\mathbf{s})_{l_A}\downarrow \text{ for all } \mathbf{l}_A \in \mathbf{w} \\ \text{undefined} & \text{otherwise} \end{cases} \\
\mathbf{F}(\mathbf{h};\mathbf{k})_{\ \ w} &= \mathsf{id}_{\mathsf{B}} \\
\mathbf{F}(\mathbf{h};\mathbf{k})_{\{\ \ i:A_i\}w} &= \ \mathbf{r}{:} \begin{cases} \{\!|\mathsf{m}_i = \mathbf{k}_{A_i w}(\mathbf{r}{:}\mathsf{m}_i)|\!\} & \text{if } \mathbf{k}_{A_i w}(\mathbf{r}{:}\mathsf{m}_i)\downarrow \text{ for all } i \\ \text{undefined} & \text{otherwise} \end{cases} \\
\mathbf{F}(\mathbf{h};\mathbf{k})_{A\ \ Bw} &= \ \mathbf{f}\ \mathbf{w}\ \geq \mathbf{w}\ \ \langle \mathbf{s};\mathbf{a}\rangle{:} \\
&\quad \begin{cases} \langle \mathbf{w}\ \ ; \langle \mathbf{k}_{Sw''}(\mathbf{s}\ \ ); \mathbf{k}_{Bw''}(\mathbf{b})\rangle\rangle \\ \qquad \text{if } \mathbf{h}_{Sw'}(\mathbf{s})\downarrow \text{ and } \mathbf{h}_{Aw'}(\mathbf{a})\downarrow \text{ and} \\ \qquad \mathbf{f}_{w'}(\mathbf{h}_{Sw'}(\mathbf{s}); \mathbf{h}_{Aw'}(\mathbf{a})) = \langle \mathbf{w}\ \ ; \langle \mathbf{s}\ \ ; \mathbf{b}\rangle\rangle\downarrow \\ \qquad \text{and } \mathbf{k}_{Sw''}(\mathbf{s}\ \ )\downarrow \text{ and } \mathbf{k}_{Bw''}(\mathbf{b})\downarrow \\ \text{undef. otherwise} \end{cases} \\
\mathbf{F}(\mathbf{h};\mathbf{k})_{\ \ Aw} &= \ \mathbf{l}{:}\mathbf{l}
\end{aligned}$$

of the subsumption rule,

$$\left[\!\!\left[\frac{\mathcal{P}(\Gamma\ .\ \mathbf{e} : \mathbf{A})\ \ \mathcal{P}(\mathbf{A} \prec: \mathbf{B})}{\Gamma\ .\ \mathbf{e} : \mathbf{B}}\right]\!\!\right]_w \ \ \mathbf{s}$$

$$= \begin{cases} \langle \mathbf{w}\ ; \langle \mathbf{s}\ ; [\![\mathcal{P}(\mathbf{A} \prec: \mathbf{B})]\!]_{w'}\ \mathbf{a}\rangle\rangle & \text{if } [\![\mathcal{P}(\Gamma\ .\ \mathbf{e} : \mathbf{A})]\!]_w\ \ \mathbf{s} = \langle \mathbf{w}\ ; \langle \mathbf{s}\ ; \mathbf{a}\rangle\rangle\downarrow \\ \text{undefined} & \text{otherwise} \end{cases}$$

As explained above, the semantics of functions is parameterised over extensions of the current world ,

$$\left[\!\!\left[\frac{\mathcal{P}(\Gamma; \mathbf{x} : \mathbf{A}\ .\ \mathbf{e} : \mathbf{B})}{\Gamma\ .\ \ \mathbf{x}{:}\mathbf{e} : \mathbf{A} \Rightarrow \mathbf{B}}\right]\!\!\right]_w \ \ \mathbf{s}$$

$$= \langle \mathbf{w}; \langle \mathbf{s};\ \ \mathbf{w}\ \ \geq \mathbf{w}\ \ \langle \mathbf{s}\ ; \mathbf{a}\rangle{:} [\![\mathcal{P}(\Gamma; \mathbf{x}{:}\mathbf{A}\ .\ \mathbf{e} : \mathbf{B})]\!]_{w'}\ ([\![\Gamma]\!]_w^{w'}\ \ )[\mathbf{x} := \mathbf{a}]\ \mathbf{s}\ \rangle\rangle$$

TABLE **3. Coercion maps**

$$\left[\!\!\left[\frac{}{\mathbf{A} \prec: \mathbf{A}} \right]\!\!\right]_w = \mathsf{id}_{[\![A]\!]_w}$$

$$\left[\!\!\left[\frac{\mathcal{P}(\mathbf{A} \prec: \mathbf{A}\ )\quad \mathcal{P}(\mathbf{A}\ \ \prec: \mathbf{B})}{\mathbf{A} \prec: \mathbf{B}} \right]\!\!\right]_w = [\![\mathcal{P}(\mathbf{A}\ \ \prec: \mathbf{B})]\!]_w \circ [\![\mathcal{P}(\mathbf{A} \prec: \mathbf{A}\ )]\!]_w$$

$$\left[\!\!\left[\frac{\mathbf{I}\ \ \subseteq \mathbf{I}\quad \mathcal{P}(\mathbf{A}_i \prec: \mathbf{A}_i)\ \forall \mathbf{i} \in \mathbf{I}}{\{\mathsf{m}_i : \mathbf{A}_i\}_{i\ \ I} \prec: \{\mathsf{m}_i : \mathbf{A}_i\}_{i\ \ I'}} \right]\!\!\right]_w = \quad \mathbf{r:}\{|\mathsf{m}_i = [\![\mathcal{P}(\mathbf{A}$$

TABLE 4. Semantics of typing judgements

$$\left\llbracket \frac{\mathcal{P}(\Gamma \;.\; \mathbf{e} : \mathbf{A}) \;\; \mathcal{P}(\mathbf{A} \prec: \mathbf{B})}{\Gamma \;.\; \mathbf{e} : \mathbf{B}} \right\rrbracket_w \;\; \mathbf{s}$$

$$= \begin{array}{ll} \langle \mathbf{w}\,; \langle \mathbf{s}\,; \llbracket \mathcal{P}(\mathbf{A} \prec: \mathbf{B}) \rrbracket_{w'}\, \mathbf{a} \rangle\rangle & \textbf{if } \llbracket \mathcal{P}(\Gamma \;.\; \mathbf{e} : \mathbf{A}) \rrbracket_w \;\; \mathbf{s} = \langle \mathbf{w}\,; \langle \mathbf{s}\,; \mathbf{a} \rangle\rangle \downarrow \\ \textbf{undefined} & \textbf{otherwise} \end{array}$$

$$\left\llbracket \frac{}{\Gamma \;.\; \mathbf{x} : \mathbf{A}} \right\rrbracket_w \;\; \mathbf{s} = \langle \mathbf{w}; \langle \mathbf{s}; \;\; (\mathbf{x}) \rangle\rangle$$

$$\left\llbracket \frac{\mathcal{P}(\Gamma \;.\; \mathbf{e} \;\; : \mathbf{B}) \quad \mathcal{P}(\Gamma; \mathbf{x}{:}\mathbf{B} \;.\; \mathbf{e}_2 : \mathbf{A})}{\Gamma \;.\; \text{let } \mathbf{x} = \mathbf{e} \;\; \text{in } \mathbf{e}_2 : \mathbf{A}} \right\rrbracket_w \;\; \mathbf{s}$$

$$= \begin{cases} \mathcal{P}(\llbracket \Gamma; \mathbf{x}{:}\mathbf{B} \;.\; \mathbf{e}_2 : \mathbf{A} \rrbracket)_{w'} (\llbracket \Gamma \rrbracket_w^{w'} \quad)[\mathbf{x} := \mathbf{b}]\, \mathbf{s} & \\ \qquad\qquad\qquad\qquad \textbf{if } \llbracket \mathcal{P}(\Gamma \;.\; \mathbf{e} \;\; : \mathbf{B}) \rrbracket_w \;\; \mathbf{s} = \langle \mathbf{w}\,; \langle \mathbf{s}\,; \mathbf{b} \rangle\rangle \downarrow \\ \textbf{undefined} \quad \textbf{otherwise} \end{cases}$$

$$\left\llbracket \frac{}{\Gamma \;.\; \text{true} : \text{bool}} \right\rrbracket_w \;\; \mathbf{s} = \langle \mathbf{w}; \langle \mathbf{s}; true \rangle\rangle$$

$$\left\llbracket \frac{\mathcal{P}(\Gamma \;.\; \mathbf{x} : \text{bool}) \quad \mathcal{P}(\Gamma \;.\; \mathbf{e}_i : \mathbf{A}) \quad \mathbf{i} = 1; 2}{\Gamma \;.\; \text{if } \mathbf{x} \text{ then } \mathbf{e} \;\; \text{else } \mathbf{e}_2 : \mathbf{A}} \right\rrbracket_w \;\; \mathbf{s}$$

$$= \;\; \llbracket \mathcal{P}(\Gamma \;.\; \mathbf{e} \;\; : \mathbf{A}) \rrbracket_w \;\; \mathbf{s} \quad \textbf{if } \llbracket \;\; \mathbf{i} \quad .$$

Table 5. Semantics of typing judgements (continued)

$$\left[\!\!\left[\frac{\mathcal{P}(\Gamma \,.\, \mathbf{x} : \mathbf{A})}{\Gamma \,.\, \mathsf{new}_A\ \mathbf{x} : \mathsf{ref}\ \mathbf{A}}\right]\!\!\right]_w \ \mathbf{s} = \langle \mathbf{w}\ ; \langle \mathbf{s}\ ; \mathbf{l}_A\rangle\rangle$$

$$\text{where } \langle \mathbf{w}; \langle \mathbf{s}; \mathbf{a}\rangle\rangle = [\![\mathcal{P}(\Gamma \,.\, \mathbf{x} : \mathbf{A})]\!]_w \ \ \mathbf{s},$$

$$\mathbf{w}\ = \mathbf{w} \cup \{\mathbf{l}_A\} \text{ for } \mathbf{l}_A \in \mathsf{Loc}_A \setminus \mathsf{dom}(\mathbf{w}) \text{ and for all } \mathbf{l}\ \in \mathbf{w}\ :$$

$$\mathbf{s}\ {:}\mathbf{l}\ = \begin{cases} [\![\mathbf{A}\ ]\!]_w^{w'}\ (\mathbf{s}{:}\mathbf{l}\ ) & \text{for } \mathbf{l}\ \in \mathbf{w} \cap \mathsf{Loc}_{A'} \\ [\![\mathbf{A}]\!]_w^{w'}\ (\mathbf{a}) & \text{for } \mathbf{l}\ = \mathbf{l}_A \end{cases}$$

$$\left[\!\!\left[\frac{\mathcal{P}(\Gamma \,.\, \mathbf{x} : \mathsf{ref}\ \mathbf{A})}{\Gamma \,.\, \mathsf{deref}\ \mathbf{x} : \mathbf{A}}\right]\!\!\right]_w \ \mathbf{s} = \langle \mathbf{w}; \langle \mathbf{s}; \mathbf{s}{:}\mathbf{l}\rangle\rangle$$

$$\text{where } \langle \mathbf{w}; \langle \mathbf{s}; \mathbf{l}\rangle\rangle = [\![\mathcal{P}(\Gamma \,.\, \mathbf{x} : \mathsf{ref}\ \mathbf{A})]\!]_w \ \ \mathbf{s}$$

$$\left[\!\!\left[\frac{\mathcal{P}(\Gamma \,.\, \mathbf{x} : \mathsf{ref}\ \mathbf{A}) \quad \mathcal{P}(\Gamma \,.\, \mathbf{y} : \mathbf{A})}{\Gamma \,.\, \mathbf{x}{:=}\mathbf{y} : 1}\right]\!\!\right]_w \ \mathbf{s} = \langle \mathbf{w}; \langle \hat{\mathbf{s}}; \{\!|\,|\!\}\rangle\rangle$$

$$\text{where } \langle \mathbf{w}; \langle \mathbf{s}; \mathbf{l}\rangle\rangle = [\![\mathcal{P}(\Gamma \,.\, \mathbf{x} : \mathsf{ref}\ \mathbf{A})]\!]_w \ \ \mathbf{s};$$

$$\langle \mathbf{w}; \langle \mathbf{s}; \mathbf{a}\rangle\rangle = [\![\mathcal{P}(\Gamma \,.\, \mathbf{y} : \mathbf{A})]\!]_w \ \ \mathbf{s} \text{ and for } \mathbf{l}\ \in \mathbf{w} :$$

$$\hat{\mathbf{s}}{:}\mathbf{l}\ = \begin{matrix} \mathbf{a} & \text{if } \mathbf{l}\ = \mathbf{l} \\ \mathbf{s}{:}\mathbf{l} & \text{if } \mathbf{l}\ \neq \mathbf{l} \end{matrix}$$

type information in

TABLE 6. **Interpretation of untyped terms**

$[\![\mathbf{x}]\!]$	$=$	$\langle\ ;\ (\mathbf{x})\rangle$ **if** $(\mathbf{x})\downarrow$ **undefined otherwise**
$[\![\text{let } \mathbf{x}{=}\mathbf{e}\ \text{ in } \mathbf{e}_2]\!]$	$=$	$[\![\mathbf{e}_2]\!]\ [\mathbf{x} := \mathbf{v}]$ **if** $[\![\mathbf{e}\ ]\!]\ = \langle\ ;\mathbf{v}\rangle\downarrow$ **undefined otherwise**
$[\![\text{true}]\!]$	$= \langle\ ;\mathit{true}\rangle$	
$[\![\text{if } \mathbf{x} \text{ then } \mathbf{e}\ \text{ else } \mathbf{e}_2]\!]$	$=$	**8** **<** $[\![\mathbf{e}\ ]\!]$ **if** $(\mathbf{x}) = \mathit{true}\downarrow$ $[\![\mathbf{e}_2]\!]$ **if** $(\mathbf{x}) = \mathit{false}\downarrow$ **:** **undefined otherwise**
$[\![\{\mathsf{m}_i = \mathbf{x}_i\}]\!]$	$=$	$\langle\ ;\{\!\vert \mathsf{m}_i =\ (\mathbf{x}_i)\vert\!\}\rangle$ **if** $(\mathbf{x}_i)\downarrow$ **for all i** **undefined otherwise**
$[\![\mathbf{x}\mathbf{:}\mathsf{m}]\!]$	$=$	$\langle\ ;\ (\mathbf{x})\mathbf{:}\mathsf{m}\rangle$ **if** $(\mathbf{x}) \in \mathsf{Rec}_{\mathcal{M}}(\mathsf{Val})$ **and** $(\mathbf{x})\mathbf{:}\mathsf{m}\downarrow$ **undefined otherwise**
$[\![\ \ \mathbf{x}\mathbf{:}\mathbf{a}]\!]$	$= \langle\ ;\ \langle\ ;\mathbf{v}\rangle\mathbf{:}\ [\![\mathbf{a}]\!]\ [\mathbf{x} := \mathbf{v}]\ \ \rangle$	
$[\![\mathbf{x}(\mathbf{y})]\!]$	$=$	**8** **<** $(\mathbf{x})\langle\ ;\ (\mathbf{y})\rangle$ **if** $(\mathbf{x}) \in [\mathsf{St} \times \mathsf{Val}$ ***** $\mathsf{St} \times \mathsf{Val}]$ **and** $(\mathbf{y})\downarrow$ **:** **undefined otherwise**
$[\![\mathsf{new}_A\ \mathbf{x}]\!]$	$= \langle\ \ + \{\!\vert \mathbf{l}_A =\ (\mathbf{x})\vert\!\}; \mathbf{l}_A\rangle;$ **where** $\mathbf{l}_A \in \mathsf{Loc}_A \setminus \mathsf{dom}(\ \ )$	
$[\![\mathsf{deref}\ \mathbf{x}]\!]$	$=$	$\langle\ ;\ \mathbf{:}\ (\mathbf{x})\rangle$ **if** $(\mathbf{x}) \in \mathsf{Loc}$ **and** $\mathbf{:}\ (\mathbf{x})\downarrow$ **undefined otherwise**
$[\![\mathbf{x}{:=}\mathbf{y}]\!]$	$=$	$\langle\ \ ;\{\!\vert\ \vert\!\}\rangle$ **if** $(\mathbf{x}) \in \mathsf{Loc};\ \mathbf{:}\ (\mathbf{x})\downarrow$ **and** $(\mathbf{y})\downarrow$ **undefined otherwise**

where $\ \mathbf{:l} =$ $(\mathbf{y})$ **if l** $=\ (\mathbf{x})$ / **:l otherwise**

straightforward: There aye

TABLE 7. **Kripke logical relation**

$$\langle \mathbf{x}; \mathbf{y}\rangle \in \mathbf{R}_w \quad \overset{def}{\Longleftrightarrow} \quad \mathbf{y} \in \mathsf{BVal} \ \wedge \ \mathbf{x} = \mathbf{y}$$

$$\langle \mathbf{r}; \mathbf{s}\rangle \in \mathbf{R}_w^{\{ \quad i:A_i\}} \quad \overset{def}{\Longleftrightarrow} \quad \mathbf{s} \in \mathsf{Rec}_{\mathbb{L}}(\mathsf{Val}) \ \wedge \ \forall \mathbf{i}\textbf{:}\ (\mathbf{s}\textbf{:}\mathsf{m}_i\downarrow \ \wedge \langle \mathbf{r}\textbf{:}\mathsf{m}_i; \mathbf{s}\textbf{:}\mathsf{m}_i\rangle \in \mathbf{R}_w^{A_i})$$

$$\begin{aligned}\langle \mathbf{f}; \mathbf{g}\rangle \in \mathbf{R}_w^{A \quad B} \quad \overset{def}{\Longleftrightarrow} \quad & \mathbf{g} \in [\mathsf{St} \times \mathsf{Val} \ \textbf{*} \ \mathsf{St} \times \mathsf{Val}] \ \wedge \\ & \forall \mathbf{w} \ \geq \mathbf{w}\ \forall \langle \mathbf{s}; \quad \rangle \in \mathbf{R}_{w'}^{St}, \forall \langle \mathbf{x}; \mathbf{y}\rangle \in \mathbf{R}_{w'}^{A}, \\ & \quad (\mathbf{f}_{w'}(\mathbf{s}; \mathbf{x})\uparrow \ \wedge \mathbf{g}(\ ; \mathbf{y})\uparrow) \\ & \vee \ \exists \mathbf{w} \ \ \geq \mathbf{w} \ \exists \mathbf{s} \ \in \mathbf{S}_{w'} \ \exists \mathbf{x} \ \in [\![\mathbf{B}]\!]_{w'} \ \exists \quad \in \mathsf{St} \ \exists \mathbf{y} \ \in \mathsf{Val}\textbf{:} \\ & \qquad (\mathbf{f}_{w'}(\mathbf{s}; \mathbf{x}) = \langle \mathbf{w} \ ; \langle \mathbf{s} \ ; \mathbf{x} \ \rangle\rangle \ \wedge \ \mathbf{g}(\ ; \mathbf{y}) = \langle \quad ; \mathbf{y} \ \rangle \\ & \qquad \wedge \langle \mathbf{s} \ ; \quad \rangle \in \mathbf{R}_{w''}^{St} \ \wedge \ \langle \mathbf{x} \ ; \mathbf{y} \ \rangle \in \mathbf{R}_{w''}^{B})\end{aligned}$$

$$\langle \mathbf{x}; \mathbf{y}\rangle \in \mathbf{R}_w^{\ A} \quad \overset{def}{\Longleftrightarrow} \quad \mathbf{y} \in \mathbf{w} \cap \mathsf{Loc}_A \ \wedge \ \mathbf{x} = \mathbf{y}$$

with the auxiliary relation $\mathbf{R}_w^{St} \subseteq \mathbf{S}_w \times \mathsf{St}$**,**

$$\langle \mathbf{s}; \quad \rangle \in \mathbf{R}_w^{St} \quad \overset{def}{\Longleftrightarrow} \quad \mathsf{dom}(\mathbf{s}) = \mathbf{w} = \mathsf{dom}(\quad) \ \wedge \ \forall \mathbf{l}_A \in \mathbf{w}\textbf{:} \ \langle \mathbf{s}\textbf{:}\mathbf{l}_A; \ \textbf{:}\mathbf{l}_A\rangle \in \mathbf{R}_w^{A}$$

5.1 *Existence of* $^{\mathsf{A}}_{\mathsf{w}}$

To establish the existence of such a relation one uses Pitts' technique for the bilimit-compact product category C   pCpo. Let   : pCpo^{op} **pCpo   ! pCpo be the locally continuous functor for which (4) is the minimal invariant,**

$$\mathbf{G}(\mathbf{D}; \mathbf{E}) \ = \ \mathsf{BVal} + \mathsf{Loc} + \mathsf{Rec}_{\mathbb{L}}(\mathbf{E}) + (\mathsf{Rec}_{\mathsf{L} \ \mathsf{c}}(\mathbf{D}) \times \mathbf{D} \ \textbf{*} \ \mathsf{Rec}_{\mathsf{L} \ \mathsf{c}}(\mathbf{E}) \times \mathbf{E})$$

and let   be the functor defined in Table 2 on

Table 8. The functional Φ

At A, w the map Φ is defined according to

$$\langle \mathbf{x};\mathbf{y}\rangle \in \Phi(\mathbf{R};\mathbf{S})_w \quad \overset{def}{\iff} \quad \mathbf{y} \in \mathsf{BVal} \text{ and } \mathbf{x} = \mathbf{y}$$

$$\langle \mathbf{r};\mathbf{s}\rangle \in \Phi(\mathbf{R};\mathbf{S})_w^{\{\ i:A_i\}} \quad \overset{def}{\iff} \quad \mathbf{s} \in \mathsf{Rec}_{\mathcal{M}}(\mathbf{Y}\ ) \text{ and } \forall \mathbf{i}\ \mathbf{s}{:}\mathsf{m}_i\downarrow \wedge \langle \mathbf{r}{:}\mathsf{m}_i;\mathbf{s}{:}\mathsf{m}_i\rangle \in \mathbf{S}_w^{A_i}$$

$$\langle \mathbf{f};\mathbf{g}\rangle \in \Phi(\mathbf{R};\mathbf{S})_w^{A\quad B} \quad \overset{def}{\iff} \quad \mathbf{g} \in [\mathsf{Rec}_{\mathsf{L\ c}}(\mathbf{Y}\ ) \times \mathbf{Y} \ \ * \ \mathsf{Rec}_{\mathsf{L\ c}}(\mathbf{Y}\ ) \times \mathbf{Y}\ ] \text{ and}$$
$$\forall \mathbf{w}\ \geq \mathbf{w}\ \forall \langle \mathbf{s};\ \ \rangle \in \mathbf{R}_{w'}^{St}, \forall \langle \mathbf{x};\mathbf{y}\rangle \in \mathbf{R}_{w'}^{A},$$
$$(\mathbf{f}_{w'}(\mathbf{s};\mathbf{x})\uparrow \wedge \mathbf{g}(\ \ ;\mathbf{y})\uparrow) \text{ or}$$
$$(\mathbf{f}_{w'}(\mathbf{s};\mathbf{x}) = \langle \mathbf{w}\ \ ;\langle \mathbf{s}\ ;\mathbf{x}\ \rangle\rangle\downarrow \wedge \mathbf{g}(\ \ ;\mathbf{y}) = \langle\ \ \ ;\mathbf{y}\ \rangle\downarrow$$
$$\wedge \langle \mathbf{s}\ ;\ \ \ \rangle \in \mathbf{S}_{w''}^{St} \wedge \langle \mathbf{x}\ ;\mathbf{y}\ \rangle \in \mathbf{S}_{w''}^{B})$$

$$\langle \mathbf{x};\mathbf{y}\rangle \in \Phi(\mathbf{R};\mathbf{S})_w^{\ \ A} \quad \overset{def}{\iff} \quad \mathbf{y} \in \mathbf{w} \cap \mathsf{Loc}_A \text{ and } \mathbf{x} = \mathbf{y}$$

and at $\mathbf{S}_w$ it is given by

$$\langle \mathbf{s};\ \ \rangle \in \Phi(\mathbf{R};\mathbf{S})_w^{St} \quad \overset{def}{\iff} \quad \mathsf{dom}(\mathbf{s}) = \mathbf{w} = \mathsf{dom}(\ \ ) \text{ and } \forall \mathbf{l}_A \in \mathbf{w}{:}\ \langle \mathbf{s}{:}\mathbf{l};\ \ {:}\mathbf{l}\rangle \in \mathbf{S}_w^{A}$$

According to [40], Lemma 5.2 guarantees that Φ has a unique fixed point x(Φ) in R$(D\ \ \mathsf{Val})$, and we obtain the Kripke logical relation $\overset{de}{=}$ x(Φ) satisfying $= \Phi(\qquad)$ as required.

Theorem 5.3 (Existence, [40]). The functional Φ has a unique fixed point.

Proof of Lemma 5.2. Let 2 W and A 2 Type. We consider cases for A.

A is bool: By definition of the functors and , $($ $)(e\ \ f) =$ h $(e_1$

ool (=1dom 2x) 1S 0 ( ) ey 1 1 1 2 02 0 ( ) 1 10 0 2 2 0 and)(e

A **is** B **)** B'**. Suppose h i 2** $\Phi(\quad)_{\mathbf{w}}^{\mathbf{B}\Rightarrow\mathbf{B}'}$**, we have to show that**

$$\langle \mathbf{F}(\mathbf{e}\ ;\mathbf{f}\ )_{B\quad B'\,w}(\mathbf{h});\mathbf{G}(\mathbf{e}_2;\mathbf{f}_2)(\mathbf{k})\rangle \in \Phi(\mathbf{R}\ ;\mathbf{S}\ )_{w}^{B\quad B'} \qquad \textbf{(6)}$$

So let $'$ **, h i 2** $'_{\mathbf{w}'}$ **and h** y**i 2** $'^{\mathbf{B}}_{\mathbf{w}'}$**. By assumption,**

$\mathbf{e}\ _{Sw'}(\mathbf{s})\downarrow.\quad \mathsf{Rec}_{\mathsf{L}\ \mathsf{c}}(\mathbf{e}_2)(\ )\downarrow$ **and then** $\langle\mathbf{e}\ _{Sw'}(\mathbf{s});\mathsf{Rec}_{\mathsf{L}\ \mathsf{c}}(\mathbf{e}_2)(\ )\rangle \in \mathbf{R}^{St}_{w'}$

$\mathbf{e}\ _{Bw'}$

A is f$\mathsf{m_i}$: A_ig$_{i\in I}$. By definition of ${}^{A}_{w}$ we know y 2 $\mathsf{Rec}_{\mathcal{M}}(\mathsf{Val})$ and h $\mathsf{m_i}$ y $\mathsf{m_i}$i 2 ${}^{A_i}_{w}$ for all 2 . So by induction hypothesis, h$[\![A_i]\!]^{w'}_{w}$ ($\mathsf{m_i}$) y $\mathsf{m_i}$i 2 ${}^{A_i}_{w}$ for all , and h$[\![A]\!]^{w'}_{w}$ ( ) yi 2 ${}^{A}_{w'}$ follows since

$$[\![\mathbf{A}]\!]^{w'}_{w}(\mathbf{x}):\mathsf{m}_i = [\![\mathbf{A}_i]\!]^{w'}_{w}(\mathbf{x}:\mathsf{m}_i)$$

A is B) B'. By definition, $[\![B$) $B']\!]^{w'}_{w}$ ( ) = ${}_{w''\geq w'}$ ${}_{w''}$, so the result follows directly from the definition of ${}^{B\Rightarrow B'}_{w'}$ and the assumption h yi 2 ${}^{B\Rightarrow B'}_{w}$.

A is ref B. Immediately from $[\![\mathsf{ref}\ B]\!]^{w'}_{w}$ ( ) = .

□

Lemma 5.5 (Subtype Monotonicity). Let 2 W, A : B and h i 2 ${}^{A}_{w}$. Then h$[\![A$: $B]\!]_{w}$ ( ) i 2 ${}^{B}_{w}$.

Proof. By a straightforward induction on the derivation of A : B: Suppose h yi 2 ${}^{A}_{w}$. If the last step in A : B is

(Reflexivity). In this case, A B and $[\![A$: $B]\!]_{w}$ ( ) = , so that h$[\![A$: $B]\!]_{w}$ ( ) yi 2 ${}^{B}_{w}$ is immediate.

(Transitivity). Assume A : B was derived from A : A' and A' : B. Applying the induction hypothesis, h$[\![A$: $A']\!]_{w}$ ( ) yi 2 ${}^{A'}_{w}$ and again by induction hypothesis,

$$\text{h}[\![A' \ : B]\!]_{w}([\![A \ : A']\!]_{w}(\ ))\ y\text{i}\ 2\ {}^{B}_{w}$$

as required.

(Arrow). Write $'$:= $[\![A$) B : A') $B']\!]_{w}$ ( ), we

By assumption y 2 $\mathsf{Rec}_{\mathcal{M}}(\mathsf{Val})$ and h m$_{\mathsf{i}}$ y m$_{\mathsf{i}}$i 2 $\overset{\mathsf{A}}{\mathsf{w}}{}^{i}$, for all 2 . By induction hypothesis, h$[\![A_{\mathsf{i}} : A'_{\mathsf{i}}]\!]_{\mathsf{w}}($ m$_{\mathsf{i}})$ y m$_{\mathsf{i}}$i 2 $\overset{\mathsf{A}}{\mathsf{w}}{}^{i}$ for all 2 $'$.

the inductive hypothesis to Γ $: A$ $e_2 : B$ we obtain that either both $[\![e_2]\!]$ $[$ $:=$ $]$ $'$" and $[\![\Gamma$ $:A$ $e_2 : B]\!]_{\mathrm{w}'}$ $([\![\Gamma]\!]^{\mathrm{w}'}_{\mathrm{w}}$ $($ $)[$ $:=$ $])$ $'$", or

- $[\![\Gamma$ $:A$ $e_2 : B]\!]_{\mathrm{w}'}$ $([\![\Gamma]\!]^{\mathrm{w}'}_{\mathrm{w}}$ $($ $)[$ $:=$ $])$ $' =$ h $''$ h $''$ $'$ii# and
- $[\![e_2]\!]$ $[$ $:=$ $]$ $' =$ h $''$ $'$i

where h $''$ $''$i 2 $_{\mathrm{w}''}$ and h $'$ $'$i 2 $^{\mathrm{B}}_{\mathrm{w}''}$. Using the definition of $[\![\Gamma$ let $=e_1$ in $e_2 : B]\!]$ and $[\![$let $=e_1$ in $e_2]\!]$, this is all that needed to be shown.

(Const) Suppose we have derived Γ true : bool by the rule for constant true. The result follows directly from $[\![\Gamma$ true : bool$]\!]_{\mathrm{w}}$ $=$ h $true$i and $[\![$true$]\!]$ $=$ h $true$i, the assumption h i 2 $_{\mathrm{w}}$ and the definition of $_{\mathrm{w}}^{\mathrm{ool}}$. The case where Γ false : bool is analogous.

(If) By induction hypothesis on the premiss Γ : bool, the assumption h i 2 $^{\Gamma}_{\mathrm{w}}$ and the definition of the semantics, $[\![\Gamma$: bool$]\!]_{\mathrm{w}}$ $=$ h h ii and $[\![$ $]\!]$ $=$ h i s.t. h i 2 $_{\mathrm{w}}^{\mathrm{ool}}$, for all h i 2 $_{\mathrm{w}}$. By definition this means 2 BVal and $=$.

We consider the case where $= true = v$, the case where both equal $false$ is analogous. By induction hypothesis on Γ $e_1 : A$, either both $[\![\Gamma$ $e_1 : A]\!]_{\mathrm{w}}$ " and $[\![e_1]\!]$ ", or $[\![\Gamma$ $e_1 : A]\!]_{\mathrm{w}}$ $=$ h $'$ h $'$ $'$ii# and $[\![e_1]\!]$ $=$ h $'$ $'$i where h $'$ $'$i 2 $_{\mathrm{w}'}$ and h $'$ $'$i 2 $^{\mathrm{A}}_{\mathrm{w}'}$. The result follows now by observing that $[\![\Gamma$ if then e_1 else $e_2 : A]\!]_{\mathrm{w}}$ $=$ h $'$ h $'$ $'$ii and $[\![$if then e_1 else $e_2]\!]$ $=$ h $'$ $'$i.

(Record) For all 2 , by induction hypothesis and from the fact that $[\![$ i$]\!]$ $=$ h $($ i$)$i one obtains $[\![\Gamma$ i $: A_{\mathrm{i}}]\!]_{\mathrm{w}}$ $=$ h h iii s.t. h i $($ i$)$i 2 $^{\mathrm{A}_i}_{\mathrm{w}}$. By definition, $[\![\Gamma$ fm$_{\mathrm{i}}$ $=$ ig : fm$_{\mathrm{i}}$ $: A_{\mathrm{i}}$g$]\!]_{\mathrm{w}}$ $=$ h h fjm$_{\mathrm{i}}$ $=$ ijgii and $[\![$fm$_{\mathrm{i}}$ $=$ ig$]\!]$ $=$ h fjm$_{\mathrm{i}}$ $=$ $($ i$)$jgi, both $[\![\Gamma \quad :A \quad e : B]\!]_{\mathbf{w}'}$ $([\![\Gamma]\!]_{\mathbf{w}}^{\mathbf{w}'}$ $(\ )[\ := \ ])$ $'$" and $[\![e]\!]$ $[\ := \ ]$ $'$", or

$$[\![\Gamma; \mathbf{x}:\mathbf{A} \ . \ \mathbf{e} : \mathbf{B}]\!]_{w'} \, ([\![\Gamma]\!]_{w}^{w'} \, (\ )[\mathbf{x} := \mathbf{u}])\mathbf{s} \ = \langle \mathbf{w} \ ; \langle \mathbf{s} \ ; \mathbf{u} \ \rangle\rangle\downarrow$$

and $[\![e]\!]$ $[\ := \ ]$ $' = $ h $''$ $'$i where h $''$ $''$i 2 $_{\mathbf{w}''}$ and h $'$ $'$i 2 $^{\mathbf{B}}_{\mathbf{w}''}$

Theorem 5.7 (Bracketing). For all 2 W and A 2 Type,

1. for all 2 $[\![A]\!]_{\mathsf{w}}$ h $^{\mathsf{A}}_{\mathsf{w}}(\ )$i 2 $^{\mathsf{A}}_{\mathsf{w}}$,
2. for all 2 $_{\mathsf{w}}$ h $_{\mathsf{w}}(\ )$i 2 $_{\mathsf{w}}$
3. for all h yi 2 $^{\mathsf{A}}_{\mathsf{w}}$ $=$ $^{\mathsf{A}}_{\mathsf{w}}(y)$,
4. for all h i 2 $_{\mathsf{w}}$ $=$ $_{\mathsf{w}}(\ )$

Compared to Reynolds work, the proof of Theorem 5.7 is more involved, again due to the (mixed-variant) type recursion caused by the use of higher-order store. Therefore we first show a preliminary lemma, which uses the projection maps that come with the minimal invariant solution D of the endofunctor on C: For $(e) = \ (e\ e)$ we set $^{\mathsf{Aw}}_{\mathsf{n}} \overset{de}{=} {}^{\mathsf{n}}(\mathbf{?})_{\mathsf{Aw}}$, and similarly $^{\mathsf{Sw}}_{\mathsf{n}} \overset{de}{=} {}^{\mathsf{n}}(\mathbf{?})_{\mathsf{Sw}}$. Note that by definition of the minimal invariant solution,

$$\mathbf{F}_{n}\ {}^{Aw}_{n} = (\mathbf{F}_{n}\ {}^{n}(\perp))_{Aw} = (\mathsf{lfp}(\ ))_{Aw} = \mathsf{id}_{Aw}$$

follows. Similarly, $\bigsqcup_{\mathsf{n}}\ {}^{\mathsf{Sw}}_{\mathsf{n}} = \mathsf{id}_{\mathsf{Sw}}$ holds.

Lemma 5.8. For all n 2 N, 2 W, A 2 Type,

1. 8 2 $[\![A]\!]_{\mathsf{w}}$ $^{\mathsf{Aw}}_{\mathsf{n}}(\ )\# =$) h $^{\mathsf{Aw}}_{\mathsf{n}}(\ )$ $^{\mathsf{A}}_{\mathsf{w}}(\ ^{\mathsf{Aw}}_{\mathsf{n}}(\ ))$i 2 $^{\mathsf{A}}_{\mathsf{w}}$
2. 8 2 $_{\mathsf{w}}$ $^{\mathsf{Sw}}_{\mathsf{n}}(\ )\# =$) h $^{\mathsf{Sw}}_{\mathsf{n}}(\ )$ $_{\mathsf{w}}(\ ^{\mathsf{Sw}}_{\mathsf{n}}(\ ))$i 2 $_{\mathsf{w}}$
3. 8h yi 2 $^{\mathsf{A}}_{\mathsf{w}}$ $^{\mathsf{Aw}}_{\mathsf{n}}(\ )\# =$) $^{\mathsf{Aw}}_{\mathsf{n}}(\ ) =$ $^{\mathsf{Aw}}_{\mathsf{n}}(\ ^{\mathsf{A}}_{\mathsf{w}}(y))$
4. 8h i 2 $_{\mathsf{w}}$ $^{\mathsf{Sw}}_{\mathsf{n}}(\ )\# =$) $^{\mathsf{Sw}}_{\mathsf{n}}(\ ) =$ $^{\mathsf{Sw}}_{\mathsf{n}}(\ _{\mathsf{w}}(\ ))$

Proof. By a simultaneous induction on n, considering cases for A in parts 1 and 3. Clearly the result holds for $n = 0$ since then $^{\mathsf{Aw}}_{0}$ and $^{\mathsf{Sw}}_{0}$ are undefined everywhere. For the case n 0:

1. We consider cases for A:

   A is bool: By definition, $^{\mathsf{oolw}}_{\mathsf{n}}(\ ) =$ 2 BVal, and therefore $^{\mathsf{ool}}_{\mathsf{w}}(\ ^{\mathsf{oolw}}_{\mathsf{n}}(\ )) =$ $^{\mathsf{oolw}}_{\mathsf{n}}(\ ) =$ 2 BVal. Hence,

   $$\langle\ _{n}\ ^{w}(\mathsf{x});\ _{w}\ (\ _{n}\ ^{w}(\mathsf{x}))\rangle = \langle \mathsf{x}; \mathsf{x}\rangle \in \mathsf{R}_{w}$$

   by the definition of $^{\mathsf{ool}}_{\mathsf{w}}$.

   A is fjm$_{\mathsf{i}}$: A_{i}jg: We know $^{\{\!|\ i:\mathsf{A}_i |\!\}}_{\mathsf{n}}(\ ) =$ fjm$_{\mathsf{i}}$ $=$ $^{\mathsf{A}_i\mathsf{w}}_{\mathsf{n}-1}(\ $ m$_{\mathsf{i}}$)jg. By induction hypothesis,

   $$\langle\ ^{A_i w}_{n-}(\mathsf{x}{:}\mathsf{m}_i);\ ^{A_i}_{w}(\ ^{A_i w}_{n-}(\mathsf{x}{:}\mathsf{m}_i))\rangle \in \mathsf{R}^{A_i}_{w}$$

for all and, by the definition of $\{\!|_{\mathbf{n}} \; _i : \mathbf{A}_i |\!\}\mathbf{w}$ and $\{\!| \; _i : \mathbf{A}_i |\!\}$

$^{\mathsf{ref}}_{\mathbf{w}}{}^{\mathbf{B}}(\ ^{\mathsf{ref}}_{\mathbf{n}}{}^{\mathbf{Bw}}(\ )) = \ ^{\mathsf{ref}}_{\mathbf{w}}{}^{\mathbf{B}}(\ ) = \quad \mathbf{2}$ Loc**, which entails**

$$\langle\ {}_n^{\varsigma Bw}(\mathbf{x});\ \ {}_w^{\varsigma B}(\ \ {}_n^{\varsigma Bw}(\mathbf{x}))\rangle = \langle \mathbf{x};\mathbf{x}\rangle \in \mathbf{R}_w^{\varsigma B}$$

This concludes this part of the proof.

2. **Suppose** $^{\mathbf{Sw}}_{\mathbf{n}}(\ )$**# and let** $_{\mathbf{n}} = \ ^{\mathbf{Sw}}_{\mathbf{n}}(\ ) = $ **fj** $_{\mathbf{A}} = \ ^{\mathbf{Aw}}_{\mathbf{n}-1}(\quad_{\mathbf{A}})$**jg**$_{\mathbf{l}_A \in \mathbf{w}}$**, and so**

$$\begin{aligned} {}^{St}_{w}(\mathbf{s}_n) &= \{\!|\mathbf{l}_A = \ {}^{A}_{w}(\mathbf{s}_n{:}\mathbf{l}_A)|\!\}_{l_A\ \ w} \\ &= \{\!|\mathbf{l}_A = \ {}^{A}_{w}(\ {}^{Aw}_{n-}\ (\mathbf{s}{:}\mathbf{l}_A))|\!\}_{l_A\ \ w} \end{aligned}$$

Then dom$(\ _{\mathbf{n}}) = \quad = $ dom$(\ _{\mathbf{w}}(\ _{\mathbf{n}}))$**. Moreover, the rst part of the induction hypothesis yields** h $_{\mathbf{n}}\ _{\mathbf{A}}\quad _{\mathbf{w}}(\ _{\mathbf{n}})\ _{\mathbf{A}}$**i 2** $^{\mathbf{A}}_{\mathbf{w}}$**, for all** $_{\mathbf{A}}$ **2 ,** **i.e.,** h $_{\mathbf{n}}\quad _{\mathbf{w}}(\ _{\mathbf{n}})$**i 2** $_{\mathbf{w}}$ **as required.**

3. **Again, we consider cases for** A**:**

A **is** bool**: By the definition of** $^{\mathsf{ool}}_{\mathbf{w}}$**,** y **2** BVal **and** $= y$**. The result follows immediately from** $^{\mathsf{oolw}}_{\mathbf{n}}(\ ) = \quad = y = $ co er

and

$$ {}^{B}_{n}\quad B'\,w(\quad {}^{B}_{w}\quad B'(\mathbf{y}))_{w'}(\mathbf{s};\mathbf{u}) = \begin{cases} \langle \mathbf{w}\ \ ;\langle\ \ {}^{Sw''}_{n-}(\ \ {}^{St}_{w''}(\ \ ));\ \ {}^{B'w''}_{n-}(\ \ {}^{B'}_{w''}(\mathbf{v}))\rangle\rangle \\ \quad \textbf{if } \mathbf{y}(\ \ {}^{St}_{w'}(\ \ {}^{Sw'}_{n-}(\mathbf{s}));\ \ {}^{B}_{w'}(\ \ {}^{Bw'}_{n-}(\mathbf{u}))) \\ \quad = \langle\ \ ;\mathbf{v}\rangle\downarrow\ \textbf{and}\qquad (\ )\qquad 5\ \ 2\qquad\qquad 5\qquad (=) \\ \end{cases} $$

as required.

□

Proof of Theorem 5.7. For the first part, let 2 $[\![A]\!]_{\mathsf{w}}$. As observed above we have $= \bigsqcup_{\mathsf{n}} {}^{\mathsf{Aw}}_{\mathsf{n}}(\ )$, and in particular ${}^{\mathsf{Aw}}_{\mathsf{n}}(\ )$# for sufficiently large n 2 N. By Lemma 5.8,

$$\langle\ {}^{Aw}_{n}(\mathbf{x});\ {}^{A}_{w}(\ {}^{Aw}_{n}(\mathbf{x}))\rangle \in \mathbf{R}^{A}_{w}$$

for all sufficiently large n. Since this forms an increasing chain in the cpo $[\![A]\!]_{\mathsf{w}}$ Val, completeness of ${}^{\mathsf{A}}_{\mathsf{w}}$ and continuity of ${}^{\mathsf{A}}_{\mathsf{w}}$ shows

$$\begin{aligned}\langle \mathbf{x};\ {}^{A}_{w}(\mathbf{x})\rangle &= \langle \textstyle\bigsqcup_{n}\ {}^{Aw}_{n}(\mathbf{x});\ {}^{A}_{w}(\bigsqcup_{n}\ {}^{Aw}_{n}(\mathbf{x}))\rangle\\ &= \textstyle\bigsqcup_{n}\langle\ {}^{Aw}_{n}(\mathbf{x});\ {}^{A}_{w}(\ {}^{Aw}_{n}(\mathbf{x}))\rangle\ \in \mathbf{R}^{A}_{w}\end{aligned}$$

as required. The other parts are similar. □

6 Coherence of the Intrinsic Semantics

We have now all the parts assembled in order to prove coherence (which proceeds exactly as in [45]): Suppose $\mathbf{P}_1(\Gamma\ \ e : A)$ and $\mathbf{P}_2(\Gamma\ \ e : A)$ are derivations of the judgement $\Gamma\ \ e : A$. We show that their semantics agree. Let 2 W, 2 $[\![\Gamma]\!]_{\mathsf{w}}$ and 2 $_{\mathsf{w}}$. By Theorem 5.7 parts (1) and (2), h ${}^{\Gamma}_{\mathsf{w}}(\ )$i 2 ${}^{\Gamma}_{\mathsf{w}}$ and h $_{\mathsf{w}}(\ )$i 2 $_{\mathsf{w}}$. Hence, by two applications of the Basic Lemma of logical relations, either

$$[\![\mathcal{P}\ (\Gamma\ .\ \mathbf{e} : \mathbf{A})]\!]_{w}\ \ \mathbf{s}\uparrow\ \wedge\ [\![\mathbf{e}]\!](\ {}^{\Gamma}_{w}(\ ))(\ {}^{St}_{w}(\mathbf{s}))\uparrow\ \wedge\ [\![\mathcal{P}_2(\Gamma\ .\ \mathbf{e} : \mathbf{A})]\!]_{w}\ \ \mathbf{s}\uparrow$$

or else there exist $_{\mathsf{i}}\ \ _{\mathsf{i}}\ \ _{\mathsf{i}}$ and such that

$$\begin{aligned}&[\![\mathcal{P}\ (\Gamma\ .\ \mathbf{e} : \mathbf{A})]\!]_{w}\ \ \mathbf{s} = \langle \mathbf{w}\ ; \langle \mathbf{s}\ ; \mathbf{v}\ \rangle\rangle\\ \wedge\ &[\![\mathbf{e}]\!](\ {}^{\Gamma}_{w}(\ ))(\ {}^{St}_{w}(\mathbf{s})) = \langle\ ; \mathbf{v}\rangle\\ \wedge\ &[\![\mathcal{P}_2(\Gamma\ .\ \mathbf{e} : \mathbf{A})]\!]_{w}\ \ \mathbf{s} = \langle \mathbf{w}_2; \langle \mathbf{s}_2; \mathbf{v}_2\rangle\rangle\end{aligned}$$

where h $_{\mathsf{i}}$ i 2 $_{\mathsf{w}_i}$ and h $_{\mathsf{i}}$ i 2 ${}^{\mathsf{A}}_{\mathsf{w}_i}$, for $= 1\ 2$. The definition of the relation $_{\mathsf{w}_i}$ entails $_1 = \mathrm{dom}(\ ) =\ _2$, and by Theorem 5.7 parts (3) and (4), $_1 =\ _{\mathsf{w}_1}(\ ) =\ _{\mathsf{w}_2}(\ ) =\ _2$ and $_1 =\ {}^{\mathsf{A}}_{\mathsf{w}_1}(\ ) =\ {}^{\mathsf{A}}_{\mathsf{w}_2}(\ ) =\ _2$. We have therefore shown

Theorem 6.1 (Coherence). All derivations of a judgement $\Gamma\ \ e : A$ have the same meaning in the intrinsic semantics.

Note that this result does not hold if the type annotation A in $\mathsf{new}_{\mathsf{A}}$ was removed. In particular, there would then be two different derivations of the judgement

$$\mathbf{x}{:}\{\mathsf{m} : \mathsf{bool}\}\ .\ \mathsf{new}\ \mathbf{x}; \mathsf{true} : \mathsf{bool} \qquad \textbf{(7)}$$

one without use of subsumption, and one where is coerced to type 1 before allocation. The denotations of these two derivations are different (clearly not even the resulting extended worlds are equal). It could be argued that, at least in this particular case, this is a defect of the underlying model: The use of a global store does not reflect the fact that the cell allocated in (7) above remains local and cannot be accessed by any enclosing program. However, in the general case we do not know if the lack of locality is the only reason preventing coherence for terms without type annotations.

7 A PER Model of Higher-Order Storage and Subtyping

We consider two consequences of the preceding technical development in more detail. Firstly, the results can be used to obtain an (extrinsic) semantics over the untyped model, based on partial equivalence relations. Secondly, we discuss how this relates to a model of Abadi and Leino's logic for objects that was considered in [43].

7.1 Extrinsic PER Semantics

Apart from proving coherence, Reynolds used (his analogue of) Theorem 5.7 to develop an extrinsic semantics of types for the (purely applicative) language

However, locality is a fundamental assumption underlying many reasoning principles about programs, such as object and class invariants in object-oriented programming. The work of Reddy and Yang [41], and Benton and Leperchey [7], shows how more useful equivalences can be built in into typed models of languages with storable references. We plan to investigate in how far these ideas carry over to full higher-order store.

We remark that, unusually, the per semantics sketched above does not seem to work over a “completely untyped” partial combinatory algebra: The construction relies on the partition of the location set $\mathsf{Loc} = \bigcup_{\mathbf{A}} \mathsf{Loc}_{\mathbf{A}}$. In particular, the definition of the pers jj$A$jj$_{\mathbf{w}}$ depends on this rather arbitrary partition. The amount of type information retained by using typed locations allows to express the invariance required for references in the presence of subtyping. We have been unable to find a more “semantic” condition. In view of this, the “untyped” model could be viewed simply as a means to an end, facilitating the definition of the logical relation and bracketing maps in order to prove coherence.

Nevertheless, as pointed out to us by Bernhard Reus, the per model may be useful for providing a semantics of languages with down-casts, for example in the form of a construct

$$\frac{\Gamma \,\mathbf{.}\, \mathbf{x} : \mathbf{A} \quad \Gamma \,\mathbf{.}\, \mathbf{e} \;: \mathbf{B}{\Rightarrow}\mathbf{C} \quad \Gamma \,\mathbf{.}\, \mathbf{e}_2 : \mathbf{A}{\Rightarrow}\mathbf{C}}{\Gamma \,\mathbf{.}\, \mathsf{try}\ (\mathbf{B})\mathbf{x}\ \mathsf{in}\ \mathbf{e}\ \ \mathsf{else}\ \mathbf{e}_2 : \mathbf{C}} \quad (B \prec: A)$$

The intrinsic semantics of Section 3 is not suitable for this purpose: For instance, due to the use of coercions, it is impossible to recover “forgotten” fields of a record.

7.2 On Abadi and

-stores ,

$$\mathbf{m}(\ \ ) = \langle\ \ ;\mathbf{v}\rangle \implies \exists \mathbf{w}\ \geq \mathbf{w}{:}\mathbf{v} \in [\![\mathbf{A}]\!]_{w'} \text{ and } \quad \text{is a w-store} \qquad (10)$$

where $[\![A]\!]_{\mathbf{w}'}$ is the appropriate denotation of type A. But the use of an existential quantification is

for all n 2 N. Thus $[\![A\,)\,B]\!]^{\mathbf{w}'}_{\mathbf{w}}$ lfp($_{\mathbf{w}}$) $=$ lfp($_{\mathbf{w}'[\![\Gamma]\!]^{w'}_{w}(\;)}$) and 7! lfp($_{\mathbf{w}}$) is a natural transformation $[\![\Gamma]\!]$! $[\![A\;)\;B]\!]$. Given the notation as above, we now set

$$\left[\!\!\left[\frac{\Gamma;\mathbf{f}{:}\mathbf{A}{\Rightarrow}\mathbf{B};\mathbf{x}{:}\mathbf{A}\;.\;\mathbf{e}:\mathbf{B}}{\Gamma\;.\quad\mathbf{f}(\mathbf{x}){:}\mathbf{e}:\mathbf{A}\Rightarrow\mathbf{B}}\right]\!\!\right]_{w}\quad \mathbf{s}=\langle\mathbf{w};\langle\mathbf{s};\mathrm{lfp}(\mathbf{G}_{w\rho})\rangle\rangle\in \mathbf{P}_{w'\quad w}\mathbf{S}_{w'}\times[\![\mathbf{A}{\Rightarrow}\mathbf{B}]\!]_{w'}$$

to obtain a semantics for recursive functions in the typed model. In the untyped model, we simply set

$$[\![\quad\mathbf{f}(\mathbf{x}){:}\mathbf{e}]\!]\qquad=\langle\;\;;\mathrm{lfp}(\;\;\mathbf{h}{:}[\![\;\;\mathbf{x}{:}\mathbf{e}]\!]\;\;[\mathbf{f}\;:=\mathbf{h}])\rangle$$

Finally, we turn to the proof of the Basic Lemma, which extends to the case of recursive functions, too.

Proof of Lemma 5.6, continued. Let h i 2 $_{\mathbf{w}}$ and h i 2 $^{\Gamma}_{\mathbf{w}}$. We know that by definition,

$$\left[\!\!\left[\frac{\Gamma;\mathbf{f}{:}\mathbf{A}{\Rightarrow}\mathbf{B};\mathbf{x}{:}\mathbf{A}\;.\;\mathbf{e}:\mathbf{B}}{\Gamma\;.\quad\mathbf{f}(\mathbf{x}){:}\mathbf{e}:\mathbf{A}\Rightarrow\mathbf{B}}\right]\!\!\right]_{w}\quad \mathbf{s}=\langle\mathbf{w};\langle\mathbf{s};\mathrm{lfp}(\mathbf{G}_{w\rho})\rangle\rangle$$

and

$$[\![\quad\mathbf{f}(\mathbf{x}){:}\mathbf{e}]\!]\qquad=\langle\;\;;\mathrm{lfp}(\;\;\mathbf{h}{:}[\![\;\;\mathbf{x}{:}\mathbf{e}]\!]\;\;[\mathbf{f}\;:=\mathbf{h}])\rangle$$

By assumption, h i 2 $_{\mathbf{w}}$, and it remains to show that the two fixed points are related by $^{\mathbf{A}\Rightarrow\mathbf{B}}_{\mathbf{w}}$.

To see this, first observe that h $[f\;:=\;\;]$ $[f\;:=\;\;']$i 2 $^{\Gamma;\mathbf{f}:\mathbf{A}\Rightarrow\mathbf{B}}_{\mathbf{w}}$ for all h $'$i 2 $^{\mathbf{A}\Rightarrow\mathbf{B}}_{\mathbf{w}}$. Therefore as in the case (Lambda) of non-recursive functions, from the induction hypothesis $\Gamma\;\;f{:}A$) $B\quad{:}A\quad e:B$ it follows that

$$\langle\mathbf{G}_{w\rho}(\mathbf{h});[\![\;\;\mathbf{x}{:}\mathbf{e}]\!]\;\;[\mathbf{f}\;:=\mathbf{h}]\rangle\in\mathbf{R}^{A\quad B}_{w} \tag{12}$$

for all h $'$i 2 $A\quad y\quad A\quad f\quad f\quad f\quad f\quad B$

type $B_{\mathbf{j}}$, is written $[\mathsf{f}_{\mathbf{i}}{:}A_{\mathbf{i}}\ \mathsf{m}_{\mathbf{j}}{:}B_{\mathbf{j}}\ \mathbf{)}\ C_{\mathbf{j}}]_{\mathbf{i;j}}$. The introduction rule is

$$\frac{\mathbf{A} \equiv [\mathsf{f}_i{:}\mathbf{A}_i;\mathsf{m}_j{:}\mathbf{B}_j{\Rightarrow}\mathbf{C}_j]_{i,j} \qquad \Gamma\ .\ \mathbf{x}_i : \mathbf{A}_i\ \forall\mathbf{i} \quad \Gamma;\mathbf{y}_j{:}\mathbf{A};\mathbf{z}_j{:}\mathbf{B}_j\ .\ \mathbf{b}_j : \mathbf{C}_j\ \forall\mathbf{j}}{\Gamma\ .\ [\mathsf{f}_i = \mathbf{x}_i;\mathsf{m}_j = \&(\mathbf{y}_j)\ \ \mathbf{z}_j{:}\mathbf{b}_j]_{i,j} : \mathbf{A}} \tag{13}$$

Subtyping on objects is by width, and for methods also by depth:

$$\frac{\mathbf{B}_j{\Rightarrow}\mathbf{C}_j \prec: \mathbf{B}_j{\Rightarrow}\mathbf{C}_j\ \forall\mathbf{j} \in \mathbf{J} \quad \mathbf{I}\ \subseteq \mathbf{I} \quad \mathbf{J}\ \subseteq \mathbf{J}}{[\mathsf{f}_i : \mathbf{A}_i;\mathsf{m}_j : \mathbf{B}_j \Rightarrow \mathbf{C}_j]_{i\ \ I,j\ \ J} \prec: [\mathsf{f}_i : \mathbf{A}_i;\mathsf{m}_j : \mathbf{B}_j \Rightarrow \mathbf{C}_j]_{i\ \ I',j\ \ J'}} \tag{14}$$

The following is essentially a (syntactic) presentation of the fixed-point (or closure) model of objects [26], albeit in a typed setting: Objects of type $A\qquad[\mathsf{f}_{\mathbf{i}}{:}A_{\mathbf{i}}\ \mathsf{m}_{\mathbf{j}}{:}B_{\mathbf{j}}\ \mathbf{)}\ C_{\mathbf{j}}]_{\mathbf{i;j}}$ are simply interpreted as records of the corresponding record type $A^*\qquad \mathbf{f}\mathsf{f}_{\mathbf{i}}{:}\mathsf{ref}\ A^*_{\mathbf{i}}\ \mathsf{m}_{\mathbf{j}}{:}B^*_{\mathbf{j}}\ \mathbf{)}\ C^*_{\mathbf{j}}\mathbf{g}_{\mathbf{i;j}}$. Note that the self parameter does not play any part in this type (in contrast to functional interpretations of objects, see [14] for instance), and soundness of the subtyping rule (14) follows directly from the rules of Section 2.

A new object $[\mathsf{f}_{\mathbf{i}}{=}\ \ _{\mathbf{i}}\ \mathsf{m}_{\mathbf{j}}{=}\ \ (y_{\mathbf{j}})\ \ z_{\mathbf{j}}\ \ \ _{\mathbf{j}}]_{\mathbf{i;j}}$ of type A is created by allocating a state record   and defining the methods by mutual recursion (using obvious syntax sugar),

$$\mathsf{let}\ \mathbf{s} = \{\mathsf{f}_i = \mathsf{new}_{A_i}(\mathbf{x}_i)\}_{i\ \ I}\ \mathsf{in}\ M\ \ \ _A(\mathbf{s})(\{\mathsf{m}_j =\ \ \mathbf{y}_j\ \ \mathbf{z}_j{:}\mathbf{b}_j\}_{j\ \ J})$$

where Meth$_{\mathbf{A}}$: $\mathbf{f}\mathsf{f}_{\mathbf{i}}{:}\mathsf{ref}\ A_{\mathbf{i}}\mathbf{g}_{\mathbf{i}\in\mathbf{I}}\ \ \mathbf{)}\ \ \mathbf{f}\mathsf{m}_{\mathbf{j}}{:}A^*\ \mathbf{)}\ B_{\mathbf{j}}\ \mathbf{)}\ C_{\mathbf{j}}\mathbf{g}_{\mathbf{j}\in\mathbf{J}}\ \ \mathbf{)}\ \ A^*$ is given by

$$M\ \ \ _A \equiv\ \ \mathbf{f}(\mathbf{s}){:}\ \ \mathbf{m}{:}\ \{\mathsf{f}_i = \mathbf{s}{:}\mathsf{f}_i;\mathsf{m}_j =\ \ \mathbf{z}_j{:}(\mathbf{m}{:}\mathsf{m}_j(\mathbf{f}(\mathbf{s})(\mathbf{m})))(\mathbf{z}_j)\}_{i\ \ I,j\ \ J}$$

Soundness of the introduction rule (13) follows immediately from this interpretation of objects and object types.

The semantics of field selection and field update are simply dereferencing and update, resp., of the corresponding field of the record. The reduction $(\ \ )^*$ of objects to the procedural language of Section 2 is summarized in Table 10.

8.3 Reasoning about Higher-order Store and Objects

One of the main motivations for devising a denotational semantics is to provide proof principles. It should enable us to specify, and reason about, concrete programs.

We look at two small case studies in this section: Firstly, recursion through the store, exemplified by an object-based implementation of the factorial function, where the recursion is resolved by calling the method through an object stored in a member field. This calls for recursively defined predicates whose well-definedness has to be established first (similar to the existence proof for the Kripke logical relation of Section 5). Secondly, we consider a simple call-back mechanism [21]: the method cb we

TABLE 10. Translation of object calculus

Types	$[\mathsf{f}_i{:}\mathbf{A}_i;\mathsf{m}_j{:}\mathbf{B}_j{\Rightarrow}\mathbf{C}_j]_{i\ \ I,j\ \ J} \equiv \{\mathsf{f}_i{:}\mathsf{ref}\ \mathbf{A}_i;\mathsf{m}_j{:}\mathbf{B}_j{\Rightarrow}\mathbf{C}_j\}_{i,j}$
Terms	$(\mathbf{a}{:}\mathsf{m}(\mathbf{b})) \equiv \mathbf{a}\ \ {:}\mathsf{m}(\mathbf{b}\ \ )$
	$(\mathbf{a}{:}\mathsf{f}) \equiv \mathsf{deref}(\mathbf{a}\ \ {:}\mathsf{f})$
	$(\mathbf{a}{:}\mathsf{f} := \mathbf{b}) \equiv (\mathbf{a}\ \ {:}\mathsf{f}){:=}\mathbf{b}$
	$[\mathsf{f}_i{=}\mathbf{x}_i;\mathsf{m}_j{=}\&(\mathbf{y}_j)\ \ \mathbf{z}_j{:}\mathbf{b}_j]_{i\ \ I,j\ \ J}$ $\equiv \mathsf{let}\ \mathbf{s} = \{\mathsf{f}_i = \mathsf{new}_{A_i}(\mathbf{x}_i)\}_{i\ \ I}\ \mathsf{in}\ M\ \ _A(\mathbf{s})(\{\mathsf{m}_j = \ \ \mathbf{y}_j\ \ \mathbf{z}_j{:}\mathbf{b}_j\}_{j\ \ J})$
where	$\mathbf{A} \equiv [\mathsf{f}_i{:}\mathbf{A}_i;\mathsf{m}_j{:}\mathbf{B}_j{\Rightarrow}\mathbf{C}_j]_{i\ \ I,j\ \ J}$
	$M\ \ _A \equiv\ \ \mathbf{f}(\mathbf{s}){:}\ \ \mathbf{m}{:}\ \{\mathsf{f}_i = \mathbf{s}{:}\mathsf{f}_i;\mathsf{m}_j = \ \ \mathbf{z}_j{:}(\mathbf{m}{:}\mathsf{m}_j(\mathbf{f}(\mathbf{s})(\mathbf{m})))(\mathbf{z}_j)\}_{i\ \ I,j\ \ J}$

cessible via one of its elds f. As such, this method may be changed at run-time. To re ect this, a sensible speci cation of the call-back would be of the form if method m satis es a speci cation , then holds of cb too, where ranges over a suitable class of speci cations.

RECURSION THROUGH THE STORE: THE FACTORIAL. In the following program let A [fac : int) int], and B [f : A fac : int) int] (so B : A). The program computes the factorial, making the recursive calls through the store. Suppose is declared as integer variable, and consider the program

$$
\begin{aligned}
&\mathsf{let}\ \mathbf{a} : \mathbf{A} = [\mathsf{fac} = \&(\mathbf{x})\ \ \mathbf{n}{:}\,\mathbf{n}]\\
&\mathsf{let}\ \mathbf{b} : \mathbf{B} = [\mathsf{f} = \mathbf{a}; \mathsf{fac} = \&(\mathbf{x})\ \ \mathbf{n}.\ \mathsf{if}\ \mathbf{n} < 1\ \mathsf{then}\ 1\ \mathsf{else}\ \mathbf{n} \times (\mathbf{x}{:}\mathsf{f}{:}\mathsf{fac}(\quad \mathbf{n}\\
&\quad \mathsf{in}\ \mathbf{b}{:}\mathsf{f} := \mathbf{b};\ \mathbf{b}{:}\mathsf{fac}(\mathbf{x})
\end{aligned}
$$

While we certainly do not claim that this is a particularly realistic example, idoes show hown614.0 Td (higher-order)Tj 82.8174 0 Td (store)Tj 35.0013 0 Td (com

neral ideas of [44]: To prove that the

the factorial of , consider the family

anges over worlds f :Ag and $P_{\mathbf{w}}$

tion using a termination order).

Due to the (negative) occurrence of $P_{\mathbf{w}'}$ in the definition of $P_{\mathbf{w}}$ existence of such a family P has to be established. This can be done along the lines of Theorem 5.3: A relational structure R on the category C is given by defining $\mathbf{R}(X)$ to be the type- and world-indexed admissible relations on X, and defining

$$\mathbf{f} : \mathbf{R} \subset \mathbf{T} \quad \text{iff} \quad \begin{array}{l} \forall \mathbf{w} \in \mathcal{W}\, \forall \mathbf{A} \in \ y \quad \forall \mathbf{x} \in \mathbf{R}^{A}_{w}\text{:}\ \mathbf{f}_{Aw}(\mathbf{x})\downarrow \Longrightarrow \mathbf{f}_{Aw}(\mathbf{x}) \in \mathbf{T}^{A}_{w} \\ \forall \mathbf{w} \in \mathcal{W}\, \forall \mathbf{s} \in \mathbf{R}^{St}_{w}\text{:}\ \mathbf{f}_{Sw}(\mathbf{s})\downarrow \Longrightarrow \mathbf{f}_{Sw}(\mathbf{s}) \in \mathbf{T}^{St}_{w} \end{array}$$

for all 2 $\mathbf{R}(X)$, 2 $\mathbf{R}(Y)$ and C-morphisms $f : X$! Y. A functional Φ is defined corresponding to the predicate P above,

$$\mathbf{f} \in \Phi(\mathbf{R})^{\mathsf{n}}_{w} \quad {}^{\mathsf{n}} \iff \forall \mathbf{w} \ \geq \mathbf{w} \ \geq \mathbf{w}\, \forall \mathbf{n} \geq 0\, \forall \mathbf{s} \in \mathbf{S}_{w'}\, \forall \mathbf{m} \in [\![\mathsf{int}]\!]_{w''}\, \forall \mathbf{s} \ \in \mathbf{S}_{w''}\text{:}$$

$$(\mathbf{s}\text{:}\mathbf{l} \in \mathbf{R}^{\mathsf{n}}_{w'} \quad {}^{\mathsf{n}} \ \wedge \ \mathbf{f}_{w'}(\mathbf{s};\mathbf{n}) = \langle \mathbf{w} \ ; \langle \mathbf{s} \ ; \mathbf{m}\rangle\rangle \implies \mathbf{m} = \mathbf{n}!)$$

at worlds f :int) intg (the value of Φ at other types, as well as on worlds not extending f :int) intg, does not really matter and could be chosen as the empty relation, for instance). This definition forms an admissible action of the functor : C ! C used to construct the model:

$$\mathbf{e}^{-} : \mathbf{R} \ \subset \mathbf{R} \wedge \mathbf{e}^{+} : \mathbf{T} \subset \mathbf{T} \quad \Longrightarrow \quad \mathbf{F}(\mathbf{e}^{-};\mathbf{e}^{+}) : \Phi(\mathbf{R}) \subset \Phi(\mathbf{R} \ ) \qquad \textbf{(15)}$$

As in Section 5, property (15) suffices to establish well-definedness of the predicates P (see [40]).

Assuming that is the location allocated for field f, a simple fixed-point induction shows

$$[\![\mathbf{x}\text{:int};\mathbf{a}\text{:}\mathbf{A}\ .\ [\mathsf{f} = \mathbf{a}; \mathsf{fac} = \&(\mathbf{x}) \ \ \mathbf{n}\text{:}\,\mathsf{if} \ \ldots]\, : \mathbf{B}]\!]_{w} \ \ \mathbf{s} = \langle \mathbf{w} \ ; \langle \mathbf{s} \ ; \mathbf{o}\rangle\rangle$$

such that $'$ is [f :Ag, and o fac **2** $P_{\mathbf{w}'}$.

Now let $\hat{} = \ '[\ := [\![B \ \ : A]\!]_{\mathbf{w}'}\,(o)]$. Thus, $\hat{}$ fac $= o$ fac **2** $P_{\mathbf{w}'}$; and if ( ) 0 we conclude

$$\begin{aligned} &[\![\mathbf{x}\text{:int};\mathbf{a}\text{:}\mathbf{A};\mathbf{b}\text{:}[\mathsf{f}\text{:}\mathbf{A};\ \mathsf{fac}\text{:int}\Rightarrow\mathsf{int}]\ .\ \mathbf{b}\text{:}\mathsf{f} := \mathbf{b}; \mathbf{b}\text{:}\mathsf{fac}(\mathbf{x}) : \mathsf{int}]\!]_{w'} \ \ [\mathbf{b} := \mathbf{o}]\hat{\mathbf{s}} \\ &= \hat{\mathbf{s}}\text{:}\mathbf{l}\text{:}\mathsf{fac}_{w'}(\hat{\mathbf{s}}; \ \ (\mathbf{x})) \\ &= \langle \mathbf{w} \ \ ; \langle \mathbf{s} \ \ ; \ \ (\mathbf{x})!\rangle\rangle \end{aligned}$$

for some $''$ and $''$.

CALL-BACKS. As a second example, we treat the call-back example considered in [44]. Call-backs are used in object-oriented programming to decouple the dependency between caller and callee objects. A typical example is that of generic buttons in user interface libraries, described in [21] by the command pattern: As the implementor of the button class cannot have any knowledge about the functionality associated with a particular window button instance, it is assumed that there will be an object supplied (at run-time) that encapsulates the desired behaviour for the button

pressed event, by providing a method execute**. Apart from implementing this interface, there are no further requirements on the supplied object. In particular, no assumptions about its** execute **method are made. The** buttonPressed **method of the button class will then react to events by forwarding to the** execute **method. In terms of specifications,** buttonPressed **would thus satisfy any specification that** execute **satisfies.**

The techniques developed in [44

over ${}_{w'}$ for xed 2 ${}_{w}$. Thus, the set

$$\mathrm{n}\; h \in [\![1 \Rightarrow 1]\!]_w \;\Big|\; \forall s; s : h_w(s; \{\}) = \langle w \;; \langle s \;; \{\}\rangle\rangle \implies \langle s; s \;\rangle \in T^{l}_{w,w'} \;\mathrm{o} \tag{16}$$

is admissible in $[\![1$) $1]\!]_{w}$. Now

Table 11. Typing of classes

$$\mathbf{B} \equiv [\overline{\mathsf{ff}}\ :\overline{\mathbf{AA}}\ ;\ \mathsf{m}_k{:}\mathbf{B}_k{\Rightarrow}\mathbf{B}_k;\ \mathsf{m}_j{:}\mathbf{C}_j{\Rightarrow}\mathbf{C}_j]_{k \quad K-J,j \quad J}$$

$$\frac{\begin{array}{ll} .\ \mathbf{c} : \mathsf{class}(\bar{\mathsf{f}}{:}\overline{\mathbf{A}};\ \mathsf{m}_k{:}\mathbf{B}_k{\Rightarrow}\mathbf{B}_k)_{k \quad K} & \mathbf{B}_j \prec: \mathbf{C}_j \quad \forall \mathbf{j} \in \mathbf{J} \cap \mathbf{K} \\ \mathsf{this}{:}\mathbf{B}\ \ ;\mathbf{y}_j{:}\mathbf{C}_j\ .\ \mathbf{e}_j : \mathbf{C}_j \quad \forall \mathbf{j} \in \mathbf{J} & \mathbf{C}_j \prec: \mathbf{B}_j \quad \forall \mathbf{j} \in \mathbf{J} \cap \mathbf{K} \end{array}}{\begin{array}{r} .\ \mathsf{class}\ (\overline{\mathbf{x}}\,\overline{\mathbf{y}})\{\overline{\mathbf{A}}\ \ \bar{\mathsf{f}}\ = \overline{\mathbf{y}};\mathbf{C}_j\ \mathsf{m}_j = \ \ (\mathbf{y}_j{:}\mathbf{C}_j){:}\mathbf{e}_j\}_{j \quad J}\ \mathsf{extends}\ \mathbf{c}(\overline{\mathbf{x}}) : \\ \mathsf{class}(\overline{\mathsf{ff}}\ :\overline{\mathbf{AA}}\ ;\ \mathsf{m}_k{:}\mathbf{B}_k{\Rightarrow}\mathbf{B}_k;\ \mathsf{m}_j{:}\mathbf{C}_j{\Rightarrow}\mathbf{C}_j)_{k \quad K-J,j \quad J} \end{array}}$$

We introduce class types in order to express the well-formedness of class tables constructed from the these class expressions,

$$\mathsf{class}(\mathsf{f}_i{:}\mathbf{A}_i;\mathsf{m}_j{:}\mathbf{B}_j{\Rightarrow}\mathbf{B}_j)_{i,j}$$

The intended meaning is that instances of a class of this type are objects with type $[\mathsf{f}_{\mathbf{i}}{:}A_{\mathbf{i}}\ \ \mathsf{m}_{\mathbf{j}}{:}B_{\mathbf{j}}\ \mathbf{)}\ B'_{\mathbf{j}}]_{\mathbf{i;j}}$. For the root class there is the obvious introduction rule,

$$\frac{}{.\ \mathsf{Root} : \mathsf{class}()}$$

and we have a type inference rule for subclassing as given in Table 11. Here the object type B is the type of instances of this class; it is used as type of the self parameter this when typing the method bodies $e_{\mathbf{j}}$. More precisely, the record type B^* is used for this purpose (recall that object types are interpreted as record types, replacing each field declaration f:A by f:ref A). Finally, note that refinement of argument and result type of methods during method redefinition is allowed (“specialisation”).

Arising from the informal interpretation of classes and objects outlined at the beginning of this subsection, the semantics of these class types is already forced upon us:

$$\begin{aligned} &\mathsf{class}(\bar{\mathsf{f}}{:}\overline{\mathbf{A}};\ \mathsf{m}_j{:}\mathbf{B}_j{\Rightarrow}\mathbf{B}_j)_j \\ &\qquad\qquad \equiv \quad (\overline{\mathbf{A}}{\Rightarrow}\{\mathsf{m}_j : \mathbf{B} \Rightarrow \mathbf{B}_j \Rightarrow \mathbf{B}_j\}_j{\Rightarrow}\mathbf{B}) \times \{\mathsf{m}_j : \mathbf{B} \Rightarrow \mathbf{B}_j \Rightarrow \mathbf{B}_j\}_j \end{aligned}$$

where $B \qquad [\bar{\mathsf{f}}{:}\overline{A}\ \ \mathsf{m}_{\mathbf{j}}{:}B_{\mathbf{j}}\ \mathbf{)}\ B'_{\mathbf{j}}]_{\mathbf{j}}$ stands for the type of instances. The first component of this pair will contain the function instantiating objects from the record of pre-methods, i.e., the second component. We reuse the recursive functions $\mathsf{Meth}_{\mathsf{B}}$ of Section 8.2 for this purpose. Formally, the semantics of class expressions is obtained by providing a translation of derivations into the procedural language of Section 2. For simplicity, we omit the types here, since the class type of a class expression is in fact uniquely determined. Thus,

$$\mathsf{Root} \quad \equiv \quad \langle\ \ \_{:}M \quad\ {}_{[]}\{\};\{\}\rangle$$

for the root

this use of reflexive domains seems unavoidable is witnessed by programs using recursion through the store, such as the factorial example of Section 8.3. However, the store parameter remains implicit in the semantics; in particular, it does not appear in the source-level type of the methods of an object and thus does not interfere with subtyping.

9 Polymorphism

We extend the language and the type system with (explicit) predicative, prenex- (or "let"-) polymorphism, similar to the (implicit) polymorphism found in Standard ML [32] and Haskell [38]. Essentially, the type system is stratified into simple types and type schemes, with universally quantified type variables ranging over simple (non-polymorphic) types only; moreover, the quantification occurs only on top-level. In particular, function arguments must have simple types. In contrast to ML, and in line with subtyping on simple types considered in previous sections, we actually consider bounded universal quantification. The universal quantification of ML can be recovered by using a trivial upper bound, >, of which every type is a subtype.

While this form of polymorphic typing may seem fairly restricted, it has proved very popular and useful in practice: It provides a good compromise between expressiveness and type inference that is tractable in many relevant cases, witnessed by the ML and Haskell languages.

Our theory goes through without any unexpected complications: After presenting the syntax and type inference rules, the semantics of bounded quantification is given using coercion maps (following [12]). Coherence of the extended system is proved by a logical relations theorem and introducing bracketing maps, as in Sects. 5 and 6. In the last part of this Section we introduce a polymorphic allocation operator. It is used in another short case study where generic classes are considered.

9.1 Syntax and Typing

We assume a countably infinite set of type variables, ranged over by identifiers , and a type > in order to denote trivial upper type substitution is an assignment of monotypes for type variables. By a monotype instance of a type scheme we mean a substitution instance without free type variables.

Contexts Γ may now contain subtype constraints of the form $: A$, with at most one of these occurring for every . Hence the derivations of subtypings may depend on the context, and there is the obvious rule to derive the subtyping Γ $: A$ from ordered pointwise, and with the action on morphisms given by restriction.

The type > is interpreted as the one-element cpo, $[\![>]\!]_w = f \quad g$. Further let $\top$

TABLE 12. Semantics of type abstraction and application

TABLE **14.** **Semantics of terms**

$$[\![\Lambda \quad \prec{:}\mathbf{A}{:}\,\mathbf{e}]\!]_{\theta} = \langle \;\; ; \quad {}_{B}\mathbf{:} \;\; \langle \quad ;\mathbf{v}\rangle\mathbf{:}\,[\![\mathbf{e}]\!]_{\theta[\alpha:=B]} \qquad \rangle$$

$$[\![\mathbf{x}_{B}]\!]_{\theta} = \begin{cases} \mathbf{p}_{(B\theta)}(\quad ;\{\![\,]\!\}) \text{ if } [\![\mathbf{x}]\!]_{\theta} = \langle\, \mathbf{Q};\mathbf{p}\rangle \in \mathsf{St} \times \quad {}_{B}(\mathsf{St} \times \mathsf{Val} \;\; \mathbf{*} \;\; \mathsf{St} \times \mathsf{Val}) \\ \text{undefined} \quad \text{otherwise} \end{cases}$$

9.3 Coherence of the Polymorphic System

We extend the coherence proof to the enriched language. For the untyped[2] semantics we introduce a

We prove the analogue of Lemma 5.5 with respect to environments.

Lemma 9.1 (Subtype Monotonicity). Let be a monotype substitution. Suppose that 2 $[\![\Gamma]\!]$ w and $'$. If h i 2 $^{\mathsf{A}}_{\mathsf{w}'}$ and **P** $(\Gamma \quad A \quad : B)$ then h($[\![$**P** $(\Gamma \quad A \quad : B)]\!]$ w $)_{\mathsf{w}'}(\ )$ i 2 $^{\mathsf{B}}_{\mathsf{w}'}$.

Proof. We consider the new case, where the derivation **P** $(\Gamma \quad A \quad : B)$ ends with an application of the rule for type variables. Thus, A is a type variable and

$$\left[\!\!\left[\frac{}{\Gamma;\quad \prec:\mathbf{B};\Gamma\quad.\quad \prec:\ \mathbf{B}}\right]\!\!\right]_{\theta,w} \quad = \quad (\mathsf{c}_\alpha)$$

The assumption 2 $[\![\Gamma]\!]$ w

either $[\![\Gamma \quad e : A]\!]$ w ' " and $[\![e]\!]$, ", or

there are ′ ′ ′ s.t. $[\![\Gamma \quad e : A]\!]$ w ' $=$ h ′ h ′ ii # and $[\![e]\!]$, $=$ h ′ i # s.t. h ′ ′i 2 $\mathbf{w}'$ and h i 2 $\mathbf{w}'$.

Proof. We consider the new cases, for type abstraction and type application.

(Type Abstraction) From the semantics it is immediate that both

$$[\![\Gamma \;.\; \mathbf{e} : \mathbf{A}]\!]_{\theta,w} \quad \mathbf{s}\downarrow \text{ and } [\![\mathbf{e}]\!]_{\theta} \quad \downarrow$$

and we must show h i 2 (immediate

TABLE 15. Bracketing maps

$$ {}^{\forall\alpha\ \ :A.\tau}_{w}(\mathbf{a}) = \ {}_{B}\ \langle\ ;\mathbf{v}\rangle\!: \begin{cases} \langle\ {}^{St}_{w''}(\mathbf{s});\ {}^{\tau[B/\alpha]}_{w''}(\mathfrak{b})\rangle \\ \qquad \text{if } \mathbf{B} \prec: \mathbf{A};\ \mathrm{dom}(\ ) = \mathbf{w}\ ;\ {}^{St}_{w'}(\ )\downarrow \text{ and} \\ \qquad \mathbf{a}_{w'B}(\ {}^{St}_{w'}(\ );\ {}_{w'}) = \langle \mathbf{w}\ ;\langle \mathbf{s};\mathfrak{b}\rangle\rangle \\ \text{undefined otherwise}\end{cases}$$

where ${}_{w} = {}_{w'}\ {}_{w}$: ${}^{B}_{w'} \circ {}^{A}_{w'}$ is the unique coercion map in $[\![\mathbf{A} \multimap \mathbf{B}]\!]_w$ (see Corollary 9.2)

$$ {}^{\forall\alpha\ \ :A.\sigma}_{w}(\mathbf{u}) = {}_{w'}\ {}_{w}\ \ {}_{B}\ {}_{:A}\ \ \langle \mathbf{s};\ \ \rangle\!: \begin{cases} \langle\ {}^{St}_{w'}(\ \ );\ {}^{\tau[B/\alpha]}_{w'}(\mathbf{v})\rangle \\ \qquad\text{if } \ {}^{St}_{w}(\mathbf{s}) = \ \ ; \\ \qquad \mathbf{u}_B(\ ;\{\!|\,|\!\}) = \langle\ \ ;\mathbf{v}\rangle \\ \qquad \mathrm{dom}(\ ) \ = \mathbf{w}\ ;\text{ and} \\ \qquad {}^{St}_{w'}(\ \ )\downarrow;\ {}^{\tau[B/\alpha]}_{w'}(\mathbf{v})\downarrow \\ \text{undefined otherwise}\end{cases}$$

that either $f_{\mathbf{wB}}$ ( ) " and ( )$_{\mathbf{B}}$ (fjg) " are both undefined, or there are ′ ′ ′ such that

$$\mathbf{f}_{w\ B\theta}(\mathbf{s};\ \ ) = \langle \mathbf{w}\ ;\langle \mathbf{s}\ ;\mathfrak{b}\rangle\rangle \text{ and } \ (\mathbf{x})_{B\theta}(\ ;\{\!|\,|\!\}) = \langle\ \ ;\mathbf{v}\rangle$$

with h ′ ′ ′ $\mathbf{w}'$ and $\mathbf{w}'$, which was to show.

The case for subsumption follows easily with Lemma 9.1. The remaining cases are proved as in Lemma w [illegible]

1. for all 2 $[\![\ ]\!]_{\mathbf{w}}$ h $_{\mathbf{w}}(\ )$i 2 $_{\mathbf{w}}$
2. for all h yi 2 $_{\mathbf{w}}$ $=$ $_{\mathbf{w}}(y)$

Proof. The proof is by induction on the number of universal quantiers in the type scheme . For simple types the claims are proved in Theorem 5.7. Now consider the case where is of the form 8 $:A$ $'$.

1. Recall that

$$\begin{array}{lll} & 8 & \\ & > & \langle\ {}^{St}_{w''}(\mathbf{s});\ {}^{\tau[B/\alpha]}_{w''}(\mathbf{b})\rangle \\ & < & \text{if } \mathbf{B} \prec: \mathbf{A};\mathsf{dom}(\ ) = \mathbf{w}\ ;\ {}^{St}_{w'}(\ )\downarrow\ \text{and} \\ {}^{\tau}_{w}(\mathbf{x}) = \ _{B}\ \langle\ ;\mathbf{v}\rangle: & > & \mathbf{x}_{w'B}(\ {}^{St}_{w'}(\ );\ _{w'}) = \langle \mathbf{w}\ ;\langle \mathbf{s};\mathbf{b}\rangle\rangle \\ & : & \text{undened otherwise} \end{array}$$

where $_{\mathbf{w'}} = \ _{\mathbf{w''}\geq\mathbf{w'}}\ {}^{\mathbf{B}}_{\mathbf{w''}}\ {}^{\mathbf{A}}_{\mathbf{w''}}$ 2 $[\![A \multimap B]\!]_{\mathbf{w}}$. Let $'$, B $:A$, let 2 $[\![B \multimap A]\!]_{\mathbf{w'}}$ and h i 2 $_{\mathbf{w'}}$. We note

$$\begin{array}{rll} \mathbf{s} = & {}^{St}_{w'}(\ ) & \text{(by Theorem 5.7)} \\ = & _{w'} & \text{(by Corollary 9.2)} \end{array}$$

Moreover, since $_{\mathbf{w''}}$ and ${}^{'[\mathbf{B}=\ ]}_{\mathbf{w''}}$ are total maps,

$$(\ {}^{\tau}_{w}(\mathbf{x}))_{B}(\ ;\{\!|\!|\!\})\uparrow \iff \mathbf{a}_{w'B}(\mathbf{s};\ )\uparrow$$

It remains to consider the case where both terms are dened. Suppose there are $''$, 2 $_{\mathbf{w''}}$ and 2 $[\![\ '[B.\ ]]\!]_{\mathbf{w''}}$,

$$\mathbf{a}_{w'B}(\mathbf{s};\ ) = \langle \mathbf{w}\ ;\langle \mathbf{s}\ ;\mathbf{b}\rangle\rangle$$

$$(\ {}^{\tau}_{w}(\mathbf{x}))_{B}(\ ;\{\!|\!|\!\}) = \langle\ {}^{St}_{w''}(\mathbf{s}\ );\ {}^{\tau'[B/\alpha]}_{w''}(\mathbf{b})\rangle$$

By Theorem 5.7, h $'$ $_{\mathbf{w''}}(\ ')$i 2 $_{\mathbf{w''}}$, and by induction hypothesis, h ${}^{'[\mathbf{B}=\ ]}_{\mathbf{w''}}(\ )$i 2 ${}^{'[\mathbf{B}=\ ]}_{\mathbf{w''}}(\ )$. Thus we have proved h $_{\mathbf{w}}(\ )$i 2 $_{\mathbf{w}}$.

2. Suppose h yi 2 $_{\mathbf{w}}$. By denition,

$$\begin{array}{lll} & 8 & \langle\ {}^{St}_{w'}(\\ & > & \\ & > & \\ & > & \qquad\qquad\qquad\qquad \wedge \\ & < & \\ {}^{\tau}_{w}(\mathbf{y}) = \ _{w'}\ _{w}\ \ _{B}\ _{:A}\ \langle \mathbf{s};\ \rangle: & > & \\ & > & \\ & > & \\ & : & \end{array}$$

where

$$\begin{aligned} \mathbf{f} &= \ \mathbf{w} \ \geq \mathbf{w} \ \ \mathbf{A} \ \ \langle \mathbf{s} ; \ \rangle \mathbf{:} [\![\ \mathbf{.} \ \ \mathbf{x:}\mathsf{new}_A\, \mathbf{x} : \mathbf{A} \Rightarrow \mathsf{ref}\ \mathbf{A}]\!]_{\theta w'}\, \mathbf{s} \\ \mathbf{g} &= \ \ \mathbf{A} \ \ \langle \ \ ;\mathbf{v}\rangle\mathbf{:} \ [\![\ \ \mathbf{x:}\mathsf{new}_A\, \mathbf{x}]\!]_{\theta} \end{aligned} \qquad \textbf{(20)}$$

To this end, suppose $'$, A is any monotype, 2 $[\![A \multimap >]\!]_{w'}$ is the unique coercion from A to >, and let h $'$ $'$i 2 $_{w'}$. By induction hypothesis and the fact that the term new$_A$ is a value it follows that

$$\begin{aligned} [\![\ \mathbf{.} \ \ \mathbf{x:}\mathsf{new}_A\, \mathbf{x} : \mathbf{A} \Rightarrow \mathsf{ref}\ \mathbf{A}]\!]_{\theta}\, \mathbf{s} &= \langle \mathbf{w} \ ; \langle \mathbf{s} \ ; \mathbf{a} \rangle\rangle \\ [\![\ \ \mathbf{x:}\mathsf{new}_A\, \mathbf{x}]\!]_{\theta} \ \ &= \langle \ \ \ ; \mathbf{u} \rangle \end{aligned}$$

with h $''$ $''$i 2 $_{w''}$ and h i 2 $^{\mathbf{A} \Rightarrow \mathsf{ref}\ \mathbf{A}}_{w''}$. Thus from the definition in (20), f and are in relation as required. □

AN APPLICATION: GENERIC CLASSES. The concept of polymorphism is not only used in functional languages, but more recently also in mainstream, object-oriented languages such as Java [11, 37], leading to parametric or generic classes. Indeed, the semantics of this section is sufficient to interpret parametric container classes: We will consider the case of objects implementing memory cells [1]; such objects can be instantiated from a class that is parametric in the type of the stored elements.

The type of memory cells storing values of type is

$$\mathbf{A}(\ \ ) \ \equiv \ [\mathsf{cont} : \ \ ; \mathsf{get} : 1 \Rightarrow \ \ ; \mathsf{set} : \ \ \Rightarrow 1]$$

providing just a field cont to store the data, and methods get and

10 Related Work

Apart from Levy's work [29, 30] which we built

11 Conclusions and Future Work

We have extended a model of general references with subtyping, to obtain a semantics of imperative objects. While the individual facts are much more intricate to prove than for the functional language considered in [45], the overall structure of the coherence proof is almost identical to loc. cit. This suggests it could be interesting to work out the general conditions needed for the construction (for example, using the setting of [35]).

In a different direction, we can extend the language with a more expressive type system: Recursive types and polymorphism feature prominently in the work on semantics of functional objects (see [14]). Here we have shown that the techniques to establish coherence scale well to the extension of the type system with ML-like (prenex) polymorphism [31, 50] – essentially because there is no interaction with the store. We are less optimistic about polymorphism in general; the combination of second-order lambda calculus and higher-order storage certainly appears to be challenging. In [30] it is suggested that the construction of the intrinsic model also works for a variant of recursive types. We haven't considered the combination with subtyping yet, but do not expect any difficulties.

Finally, we plan to develop (Hoare-style) logics, with pre- and post-conditions, for languages involving higher-order store. As a starting point, we are currently trying to adapt the program logic of [3] to the language considered here.

references. In *c -n .. Ann , Sy n L -c -n . Sc-nc*, pages 334–344. IEEE Computer Society Press, 1998.

[6] A. J. Ahmed, A. W. Appel, and R. Virga. A strati ed semantics of general references embeddable in higher-order logic. In *c -n .. Ann , Sy L -c -n . Sc-nc*, pages 75–86. IEEE Computer Society Press, 2002.

[7] N. Benton and B. Leperchey. Relational reasoning in a nominal semantics for storage. In *-n c -n .. S n. n n . -n , n nc n y L c , - n A , -c . - n . L A* , Lecture Notes in Computer Science. Springer, 2005.

[8] V. Bono and M. Bugliesi. Interpretations of extensible objects and types. In *c -n .. . n. Sy n n n. .-n* , volume 1684 of *L c. . -n . Sc-nc*, pages 112–123. Springer, 1999.

[9] V. Bono, A. J. Patel, V. Shmatikov, and J. C. Mitchell. A core calculus of classes and objects. In *.. n nc n.. M .. . -c , n . - n -n S n. -c*, volume 20 of *L c. n-c . -n . Sc-nc*, Apr. 1999.

[10] G. Boudol. The recursive record semantics of objects revisited. *n , nc. - n , -n* , 14(3):263–315, May 2004.

[11] G. Bracha, M. Odersky, D. Stoutamire, and P. Wadler. Making the future safe for the past: Adding genericity to the Java programming language. *A M S LA . -c*, 33(10):183–200, Oct. 1998.

[12] V. Breazu-Tannen, T. Coquand, G. Gunter, and A. Scedrov. Inheritance as implicit coercion. *n . - n n . . - n*, 93(1):172–221, July 1991.

[13] K. B. Bruce. A paradigmatic object-oriented programming language: Design, static typing and semantics. *n , nc. - n , -n* , 4(2):127–206, Apr. 1994.

[14] K. B. Bruce, L. Cardelli, and B. C. Pierce. Comparing object encodings. *n . - n n . . - n*, 155(1/2):108–133, Nov. 1999.

[15] P. Canning, W. Cook, W. Hill, W. Olthoff, and J. Mitchell. F-bounded polymorphism for object-oriented programming. In *c -n n. n . - n , n nc n nc . - n , -n L n n . A c. - . c* , pages 273–280. ACM Press, 1989.

[16] L. Cardelli, S. Martini, J. C. Mitchell, and A. Scedrov. An extension of System F with subtyping. *n . - n n . . - n*, 109(1–2):4–56, 1994.

[17] W. Cook and J. Palsberg. A denotational semantics of iinheritance and its correctness. *n . - n n . . - n*, 114(2):329–350, Nov. 1994.

[18] W. R. Cook. *A n . . - n , S n. -c n - nc*. Ph.D.

imperative higher-order functions. In [illegible], 2005. To appear.

[25] A. Jeffrey and J. Rathke. A fully abstract may testing semantics for concurrent objects. In [illegible] 17th [illegible], pages 101–112. IEEE Computer Society Press, 2002.

[26] S. N. Kamin and U. S. Reddy. Two semantic models of object-oriented languages. In C. A. Gunter and J. C. Mitchell, editors, [illegible], pages 464–495. MIT Press, 1994.

[27] J. Laird. A categorical semantics of higher-order store. In R. Blute and P. Selinger, editors, [illegible], volume 69 of [illegible], pages 1–18. Elsevier, 2003.

[28] P. J. Landin. The mechanical evaluation of expressions. [illegible], 6(4):308–320, Jan. 1964.

[29] P. B. Levy. Possible world semantics for general storage in call-by-value. In J. Bradfield, editor, [illegible], volume 2471 of [illegible]. Springer, 2002.

[30] P. B. Levy. [illegible], volume 2 of [illegible].

volume 3444 of *Lecture Notes in Computer Science*, pages 264–279. Springer, 2005.

[44] B. Reus and T. Streicher. Semantics and logic of object calculi. *Theoretical Computer Science*, 316:191–213, 2004.

[45] J. C. Reynolds. What do types mean? — From intrinsic to extrinsic semantics. In A. McIver and C. Morgan, editors, *Essays on Programming Methodology*. Springer, 2002.

[46] M. B. Smyth and G. D. Plotkin. The category-theoretic solution of recursive domain equations. *SIAM Journal on Computing*, 11(4):761–783, Nov. 1982.

[47] I. Stark. Names, equations, relations: Practical ways to reason about *new*. *Fundamenta Informaticae*, 33(4):369–396, April 1998.

[48] R. D. Tennent and D. R. Ghica. Abstract models of storage. *Higher-Order and Symbolic Computation*, 13(1–2):119–129, Apr. 2000.

[49] L. Thorup and M. Tofte. Object-oriented programming and standard ML. In *Proceedings of the ACM SIGPLAN Workshop on Standard ML and its Applications*, 1994.

[50] A. K. Wright. Simple imperative polymorphism. *Lisp and Symbolic Computation*, 8(4):343–355, Dec. 1995.

[51] Zhang and Nowak. Logical relations for dynamic name creation. In *Proceedings of Computer Science Logic, CSL 2003*, volume 2803 of *Lecture Notes in Computer Science*, pages 575–588. Springer, 2003.