Denotational Semantics for Abadi and Leino's Logic of Objects

Bernhard Reus
Jan Schwinghammer

Denotational Semantics for Abadi and Leino's Logic of Objects

B[illegible] R[illegible] and J[illegible] S

1 Introduction

When Hoare presented his seminal work about an *axiomatic basis of computer programming* [7], high-level languages had just started to gain broader acceptance. While programming languages are evolving ever more rapidly, verification techniques seem to be struggling to keep up. For object-oriented languages several formal systems have been proposed, e.g. [2, 6, 13, 12, 5, 20, 17]. A "standard" comparable to the Hoare-calculus for imperative While-languages [4] has not yet emerged. Nearly all the approaches listed above are designed for class-based languages (usually a sub-language of sequential Java), where method code is known statically.

One notable exception is Abadi and Leino's work [2] where a logic for an object-based language is introduced that is derived from the imperative object calculus with first-order types, **impς**, [1]. In object-based languages, every object contains its own suite of methods. Operationally speaking, the store for such a language contains code (and is thus called *higher-order store*) and modularity is for free simply by the fact that all programs can depend on the objects' code in the store. We therefore consider object-based languages ideal for studying modularity issues that occur also in class-based languages. Class-based programs can be compiled into object-based ones (see [1]), and object-based languages can naturally deal with classes defined on-the-fly, like inner classes and classes loaded at run-time (cf. [14, 15]).

Abadi and Leino's logic is a Hoare-style system, dealing with partial correctness of object expressions. Their idea was to enrich object types by method specifications, also called *transition relations*, relating pre- and post-execution states of program statements, and *result specifications* describing the result in case of program termination. Informally, an object satisfies such a specification

$$A \equiv [\mathsf{f}_i\colon A_i{}^{i=1\ldots n},\ \mathsf{m}_j\colon \varsigma(y_j)B_j{::}T_j{}^{j=1\ldots m}]$$

if it has fields f_i satisfying A_i and methods m_j that satisfy the transition relation T_j and, in case of termination of the method invocation, their result satisfies B_j. However, just as a method can use the *self*-parameter, we can assume that an object a itself satisfies A in both B_j and T_j when establishing that A holds for a. This yields a powerful and convenient proof principle for objects.[1]

We are going to present a new proof using a (untyped) denotational semantics (of the language and the logic) to define validity. Every program and every specification have a meaning, a *denotation*. Those of specifications are simply predicates on (the domain of) objects. The properties of these predicates provide

[1]This also works for class-based languages. But an easier solution for those is to interpret class specifications as mutually defined predicates over classes (and their class names).

a description of inherent limitations of the logic. Such an approach is not new, it has been used e.g. in LCF, a logic for functional programs [10].

The difficulty in this case is to establish predicates that provide the powerful reasoning principle for objects. Reus and Streicher have outlined in [16] how to use some classic domain theory [11] to guarantee existence and uniqueness of appropriate predicates on (isolated) objects. In an object-calculus program, however, an object may depend on other objects (and its methods) in the store. So object specifications must depend on specifications of other objects in the store which gives rise to "store specifications" (already present in the work of Abadi and Leino).

For the reasons given above, this paper is not "just" an application of the ideas in [16]. Much care is needed to establish the important invariance property of Abadi-Leino logic, namely that proved programs preserve store specifications. Our main achievement, in a nutshell, is that we have successfully applied the ideas of [16] to the logic of [2] to obtain a soundness proof that can be used to *analyse this logic* and to *develop similar but more powerful program logics* as well.

Our soundness proof is not just "yet another proof" either. We consider it complementary (if not superior) to the one in [2] which relies on the operational semantics of the object calculus and does not assign proper "meaning" to specifications. Our claim is backed up by the following reasons:

- By using denotational semantics we can introduce a clear notion of validity with no reference to derivability. This helps clarifying *what the proof is actually stating* in the first place.
- We can extend the logic easily e.g. for recursive specifications. This has been done for the Abadi-Leino logic in [8] but for a slightly different language with nominal subtyping.
-

a, b	$::=$	x	variable
	$\mid$	$\texttt{true} \mid \texttt{false}$	booleans
	$\mid$	$\texttt{if}\ x\ \texttt{then}\ a\ \texttt{else}\ b$	conditional
	$\mid$	$\texttt{let}\ x = a\ \texttt{in}\ b$	let
	$\mid$	$[\mathsf{f}_i = x_i{}^{i=1\ldots n}, \mathsf{m}_j = \varsigma(y_j)b_j{}^{j=1\ldots m}]$	object construction
	$\mid$	$x.\mathsf{f}$	field selection
	$\mid$	$x.\mathsf{f} := y$	field update
	$\mid$	$x.\mathsf{m}$	method invocation

Table 1. Syntax

recursive specifications can be introduced (Section 6) and discuss the benefits of the denotational approach (Section 7).

When presenting the language and logic, we deliberately keep close to the original presentation [2].

2 The Object Calculus

Below, we review the language of [2], which is based on the imperative object calculus of Abadi and Cardelli [1]. Following [16] we give a denotational semantics in Section 2.2.

2.1 Syntax

Let Var, M and F be pairwise disjoint, countably infinite sets of *variables*, *method names* and *field names*, respectively. Let x, y range over Var, let $\mathsf{m} \in \mathsf{M}$ and $\mathsf{f} \in \mathsf{F}$. The language is defined by the grammar in Tab. 1.

Variables are (immutable) identifiers, the semantics of booleans and conditional is as usual. The object expression $\texttt{let}\ x = a\ \texttt{in}\ b$ first evaluates a and then evaluates b with x bound to the result of b.

Object construction $[\mathsf{f}_i = x_i{}^{i=1\ldots n}, \mathsf{m}_j = \varsigma(y_j)b_j{}^{j=1\ldots m}]$ allocates new storage let construct[2]

$$
\begin{aligned}
[\![x]\!]\rho\sigma &= \begin{cases} (\rho(x), \sigma) & \text{if } x \in \mathrm{dom}(\rho) \\ (\mathrm{error}, \sigma) & \text{otherwise} \end{cases} \\
[\![\texttt{true}]\!]\rho\sigma &= (\mathit{true}, \sigma) \\
[\![\texttt{false}]\!]\rho\sigma &= (\mathit{false}, \sigma) \\
[\![\texttt{if}\ x\ \texttt{then}\ b_1\ \texttt{else}\ b_2]\!]\rho\sigma &= \begin{cases} [\![b_1]\!]\rho\sigma' & \text{if } [\![x]\!]\rho\sigma = (\mathit{true}, \sigma') \\ [\![b_2]\!]\rho\sigma' & \text{if } [\![x]\!]\rho\sigma = (\mathit{false}, \sigma') \\ (\mathrm{error}, \sigma') & \text{if } [\![x]\!]\rho\sigma = (v, \sigma') \text{ for } v \notin \mathrm{BVal} \end{cases} \\
[\![\texttt{let}\ x = a\ \texttt{in}\ b]\!]\rho\sigma &= \textbf{let } (v, \sigma') = [\![a]\!]\rho\sigma \textbf{ in } [\![b]\!]\rho[x := v]\sigma' \\
[\![[\mathsf{f}_i = x_i{}^{i=1\ldots n}, \mathsf{m}_j = \varsigma(y_j)b_j{}^{j=1\ldots m}]]\!]\rho\sigma & \\
&= \begin{cases} (l, \sigma[l := (o_1, o_2)]) & \text{if } x_i \in \mathrm{dom}(\rho) \end{cases}
\end{aligned}
$$

We will also use a projection to the part of the store that contains data in Val only (i.e., forget about the closures that live in the store), $\pi_{\mathsf{Val}} : \mathsf{St} \to \mathsf{St}_{\mathsf{Val}}$ defined by $(\pi_{\mathsf{Val}}\, \sigma).l.\mathsf{f} = \sigma.l.\mathsf{f}$, where $\mathsf{St}_{\mathsf{Val}} = \mathsf{Rec}_{\mathsf{Loc}}(\mathsf{Rec}_{\mathsf{F}}(\mathsf{Val}))$. We refer to $\pi_{\mathsf{Val}}(\sigma)$ as the *flat part* of σ.

Example 2.1. *We extend the syntax with integer constants and operations, and consider an object-based modelling of a bank account as an example:*

```
acc(x) ≡ [balance = 0,
          deposit10 = ς(y)let z = y.balance+10 in y.balance:=z,
          interest = ς(y)let r = x.manager.rate in
                         let z = y.balance□r/100 in y.balance:=z]
```

Note how the self parameter y is used in both methods to access the `balance` *field. Object acc depends on a "managing" object x in the context that provides the interest rate, through a field* `manager` components, which will be justified by our semantics.

Intuitively, `true` and `false` satisfy *Bool*, and an object satisfies the specification $A \equiv [\mathsf{f}_i{:}\,A_i{}^{i=1\ldots n}, \mathsf{m}_j{:}\,\varsigma(y_j)B_j{::}T_j{}^{j=1\ldots m}]$ if it has fields f_i satisfying A_i and methods m_j that satisfy the transition relation T_j and, in case of termination of the method invocation, their result satisfies B_j. Corresponding to the fact that a method m_j can use the *self*-parameter y_j, in both T_j and B_j it is possible to refer to the ambient object y_j.

Let Γ range over *specification contexts* $x_1{:}A_1, \ldots, x_n{:}A_n$. A specification context is *well-formed* if no variable x_i occurs more than once, and the free variables of A_k are contained in the set $\{x_1, \ldots, x_{k-1}\}$. In writing $\Gamma, x{:}A$ we will always assume that x does not appear in Γ. Sometimes we write $\square$ for the empty context. Given Γ, we write $[\Gamma]$ for the list of variables occurring in Γ:

$$[x_1{:}A_1, \ldots, x_n{:}A_n] = x_1, \ldots, x_n$$

If clear from context, we use the notation $\overline{x}$ for a sequence $x_1, \ldots, x_n$, and similarly $\overline{x} : \overline{A}$ for $x_1{:}A_1, \ldots, x_n{:}A_n$. To make the notions of well-formed specifications and well-formed specification contexts formal, there are judgements for

- well-formed transition relations:

$$x_1, \ldots, x_n \;\square\; T, \qquad {}_1, \ldots, x_n$$

is covariant along method specifications and transition relations, and invariant in field specifications. Observe that $\overline{x} \vdash A_1 <: A_2$ in particular implies $\overline{x} \vdash A_i$ for $i = 1, 2$.

In the logic, judgements of the form $\Gamma \vdash a{:}A{::}T$ can be derived, where Γ is a well-formed specification context, a is an object expression, A is a specification, and T is a transition relation. The rules guarantee that all the free variables of a, A and T appear in $[\Gamma]$. We use the following transition relations in the rules:

$$
\begin{aligned}
T_{\mathsf{res}}(e) &\equiv \mathsf{result} = e \\
&\quad \wedge \forall x, f.\mathsf{alloc}_{pre}(x) \leftrightarrow \mathsf{alloc}_{post}(x) \wedge \mathsf{sel}_{pre}(x, f) = \mathsf{sel}_{post}(x, f) \\
T_{\mathsf{obj}}(\mathsf{f}_i = x_i)^{i=1\ldots n} &\equiv \neg\mathsf{alloc}_{pre}(\mathsf{result}) \wedge \mathsf{alloc}_{post}(\mathsf{result}) \\
&\quad \wedge \forall x, f.x \neq \mathsf{result} \rightarrow \\
&\qquad (\mathsf{alloc}_{pre}(x) \leftrightarrow \mathsf{alloc}_{post}(x) \wedge \mathsf{sel}_{pre}(x, f) = \mathsf{sel}_{post}(x, f)) \\
&\quad \wedge \mathsf{sel}_{post}(\mathsf{result}, \mathsf{f}_1) = x_1 \wedge \cdots \wedge \mathsf{sel}_{post}(\mathsf{result}, \mathsf{f}_n) = x_n \\
T_{\mathsf{upd}}(x, f, e) &\equiv \forall x'.\mathsf{alloc}_{pre}(x') \leftrightarrow \mathsf{alloc}_{post}(x') \wedge \mathsf{sel}_{post}(x, f) = e \\
&\quad \wedge \forall x', f'.(x' \neq x \vee f' \neq f) \rightarrow \mathsf{sel}_{pre}(x', f') = \mathsf{sel}_{post}(x', f') \\
&\quad \wedge
\end{aligned}
$$

9.963 Tf 5.539 0 Td[(].)-310(W)80(e)-250(use)-250(the)-250s 3825039]TJ/F104 n Td[1iT0 Td[(].)- 5.9u.Td[

subsumption

$$\frac{[\Gamma] \vdash A <: A' \quad \Gamma \vdash a{:}A{::}T \quad [\Gamma] \vdash A' \quad [\Gamma] \vdash T' \quad \vdash_{\mathsf{fo}} T \rightarrow T'}{\Gamma \vdash a{:}A'{::}T'}$$

variable

$$\frac{\Gamma \vdash \mathsf{ok} \quad x{:}A \text{ in } \Gamma}{\Gamma \vdash x{:}A{::}T_{\mathsf{res}}(x)}$$

booleans

$$\frac{\Gamma \vdash \mathsf{ok}}{\Gamma \vdash \mathtt{false}{:}Bool{::}T_{\mathsf{res}}(\mathsf{false})} \qquad \frac{\Gamma \vdash \mathsf{ok}}{\Gamma \vdash \mathtt{true}{:}Bool{::}T_{\mathsf{res}}(\mathsf{true})}$$

conditional

$$\frac{\begin{array}{c} A[\mathsf{true}/x] \equiv A_t[\mathsf{true}/x] \text{ and } A[\mathsf{false}/x] \equiv A_f[\mathsf{false}/x] \\ T[\mathsf{true}/x] \equiv T_t[\mathsf{true}/x] \text{ and } T[\mathsf{false}/x] \equiv T_f[\mathsf{false}/x] \\ \Gamma \vdash x{:}Bool{::}T_{\mathsf{res}}(x) \quad \Gamma \vdash a{:}A_t{::}T_t \quad \Gamma \vdash b{:}A_f{::}T_f \end{array}}{\Gamma \vdash \mathtt{if}\ x\ \mathtt{then}\ a\ \mathtt{else}\ b{:}A{::}T}$$

let

$$\frac{\begin{array}{c} \Gamma \vdash a{:}A'{::}T' \quad \Gamma, x{:}A' \vdash b{:}B{::}T'' \quad [\Gamma] \vdash B \quad [\Gamma] \vdash T \\ \vdash_{\mathsf{fo}} T'[\mathsf{sel}_{int}(\cdot,\cdot)/\mathsf{sel}_{post}(\cdot,\cdot), \mathsf{alloc}_{int}(\cdot)/\mathsf{alloc}_{post}(\cdot), x/\mathsf{result}] \\ \wedge\ T''[\mathsf{sel}_{int}(\cdot,\cdot)/\mathsf{sel}_{pre}(\cdot,\cdot), \mathsf{alloc}_{int}(\cdot)/\mathsf{alloc}_{pre}(\cdot)] \rightarrow T \end{array}}{\Gamma \vdash \mathtt{let}\ x = a\ \mathtt{in}\ b{:}B{::}T}$$

object construction

$$\frac{\begin{array}{c} A \equiv [\mathsf{f}_i{:}\,A_i{}^{i=1\ldots n},\ \mathsf{m}_j{:}\,\varsigma(y_j)B_j{::}T_j{}^{j=1\ldots m}] \\ \Gamma \vdash x_i{:}A_i{::}T_{\mathsf{res}}(x_i)^{i=1\ldots n} \quad \Gamma, y_j{:}A \vdash b_j{:}B_j{::}T_j^{j=1\ldots m} \end{array}}{\Gamma \vdash [\mathsf{f}_i = x_i{}^{i=1\ldots n}, \mathsf{m}_j = \varsigma(y_j)b_j{}^{j=1\ldots m}]{:}A{::}T_{\mathsf{obj}}(\mathsf{f}_i = x_i{}^{i=1\ldots n})}$$

field selection

$$\frac{\Gamma \vdash x{:}[\mathsf{f}{:}A]{::}T_{\mathsf{res}}(x)}{\Gamma \vdash x.\mathsf{f}{:}A{::}T_{\mathsf{res}}(\mathsf{sel}_{pre}(x,\mathsf{f}))}$$

field update

$$\frac{\begin{array}{c} A \equiv [\mathsf{f}_i{:}\,A_i{}^{i=1\ldots n},\ \mathsf{m}_j{:}\,\varsigma(y_j)B_j{::}T_j{}^{j=1\ldots m}] \\ \Gamma \vdash x{:}A{::}T_{\mathsf{res}}(x) \quad \Gamma \vdash y{:}A_k{::}T_{\mathsf{res}}(y) \end{array}}{\Gamma \vdash x.\mathsf{f}_k := y{:}A{::}T_{\mathsf{upd}}(x, \mathsf{f}_k, y)} \quad 1 \leq k \leq n$$

method invocation

$$\frac{\Gamma \vdash x{:}[\mathsf{m}{:}\varsigma(y)A{::}T]{::}T_{\mathsf{res}}(x)}{\Gamma \vdash x.\mathsf{m}{:}A[x/y]{::}T[x/y]}$$

Table 3. Inference rules of Abadi-Leino logic

$$
\begin{aligned}
T_{\texttt{deposit}}(y) \equiv\ & \exists z.z = \mathsf{sel}_{pre}(y, \texttt{balance}) \\
& \wedge T_{\mathsf{upd}}(y, \texttt{balance}, z + 10)
\end{aligned}
$$

$$
\begin{aligned}
T_{\texttt{interest}}(x, y) \equiv\ & \exists z.z = \mathsf{sel}_{pre}(y, \texttt{balance}) \\
& \wedge \exists m.m = \mathsf{sel}_{pre}(x, \texttt{manager}) \\
& \wedge \exists r.r = \mathsf{sel}_{pre}(m, \texttt{rate}) \\
& \wedge T_{\mathsf{upd}}(y, \texttt{balance}, z * r/100)
\end{aligned}
$$

$$
T_{\texttt{create}}(x) \equiv T_{\mathsf{obj}}(\texttt{balance} = 0)
$$

$$
\begin{aligned}
A_{\texttt{Account}}(x) \equiv [&\texttt{balance} : \mathit{Int}, \\
& \texttt{deposit10} : \varsigma(y)[\,] :: T_{\texttt{deposit}}(y), \\
& \texttt{interest} : \varsigma(y)[\,] :: T_{\texttt{interest}}(x, y)]
\end{aligned}
$$

$$
\begin{aligned}
A_{\texttt{AccFactory}} \equiv [&\texttt{manager} : [\texttt{rate} : \mathit{Int}], \\
& \texttt{create} : \varsigma(x) A_{\texttt{Account}}(x) :: T_{\texttt{create}}(x)]
\end{aligned}
$$

$$
\begin{aligned}
A_{\texttt{Manager}} \equiv [&\texttt{rate} : \mathit{Int}, \\
& \texttt{accFactory} : A_{\texttt{AccFactory}}
\end{aligned}
$$

$$[\![\bar{x} \vdash e]\!] : \mathsf{Env}^+ \to \mathsf{St}_{\mathsf{Val}} \to \mathsf{Val} \to \mathsf{St}_{\mathsf{Val}} \rightharpoonup (\mathsf{Val} + \mathsf{F})$$

$$
\begin{array}{ll}
[\![\bar{x} \vdash x]\!]\rho\sigma v\sigma' & = \begin{cases} \rho(x) & \text{if } x \in \mathrm{dom}(\rho) \\ \text{undefined} & \text{otherwise} \end{cases} \\
[\![\bar{x} \vdash \mathsf{f}]\!]\rho\sigma v\sigma' & = \mathsf{f} \\
[\![\bar{x} \vdash \mathsf{result}]\!]\rho\sigma v\sigma' & = v \\
[\![\bar{x} \vdash \mathsf{true}]\!]\rho\sigma v\sigma' & = \mathit{true} \\
[\![\bar{x} \vdash \mathsf{false}]\!]\rho\sigma v\sigma' & = \mathit{false} \\
[\![\bar{x} \vdash \mathsf{sel}_{pre}(e_0, e_1)]\!]\rho\sigma v\sigma' & = \begin{cases} \sigma.l.\mathsf{f} & \text{if } [\![\bar{x} \vdash e_0]\!]\rho\sigma v\sigma' = l \in \mathsf{Loc} \text{ and} \\ & \quad [\![\bar{x} \vdash e_1]\!]\rho\sigma v\sigma' = \mathsf{f} \in \mathsf{F} \text{ are defined} \\ \text{undefined} & \text{otherwise} \end{cases} \\
[\![\bar{x} \vdash \mathsf{sel}_{post}(e_0, e_1)]\!]\rho\sigma v\sigma' & = \begin{cases} \sigma'.l.\mathsf{f} & \text{if } [\![\bar{x} \vdash e_0]\!]\rho\sigma v\sigma' = l \in \mathsf{Loc} \text{ and} \\ & \quad [\![\bar{x} \vdash e_1]\!]\rho\sigma v\sigma' = \mathsf{f} \in \mathsf{F} \text{ are defined} \\ \text{undefined} & \text{otherwise} \end{cases}
\end{array}
$$

$$[\![\bar{x} \vdash T]\!] : \mathsf{Env}^+ \to \mathcal{P}(\mathsf{St}_{\mathsf{Val}} \times \mathsf{Val} \times \mathsf{St}_{\mathsf{Val}})$$

$$
\begin{array}{lll}
(\sigma, v, \sigma') \in [\![\bar{x} \vdash e_0 = e_1]\!]\rho & \textit{iff} & \text{both } [\![\bar{x} \vdash e_0]\!]\rho\sigma v\sigma' \text{ and } [\![\bar{x} \vdash e_1]\!]\rho\sigma v\sigma' \text{ are defined} \\
& & \quad \text{and equal, or both undefined} \\
(\sigma, v, \sigma') \in [\![\bar{x} \vdash \mathsf{alloc}_{pre}(e)]\!]\rho & \textit{iff} & [\![\bar{x} \vdash e]\!]\rho\sigma v\sigma' \in \mathrm{dom}(\sigma) \\
(\sigma, v, \sigma') \in [\![\bar{x} \vdash \mathsf{alloc}_{post}(e)]\!]\rho & \textit{iff} & [\![\bar{x} \vdash e]\!]\rho\sigma v\sigma' \in \mathrm{dom}(\sigma') \\
(\sigma, v, \sigma') \in [\![\bar{x} \vdash \forall x.T]\!]\rho & \textit{iff} & \text{for all } u \in \mathsf{Val} + \mathsf{F} \,.\, (\sigma, v, \sigma') \in [\![\bar{x}, x \vdash T]\!]\rho[x := u]
\end{array}
$$

Table 4. Meaning of expressions and transition relations

the transition relations is untyped, the types of the free variables are not relevant.The interpretation of object specifications $\bar{x} \vdash A$,

$$[\![\bar{x} \vdash A]\!] : \mathsf{Env} \to \mathcal{P}(\mathsf{Val} \times \mathsf{St})$$

is given in Tab. 5.

We begin with a number of observations about the interpretation.

Lemma 3.2. *For all specifications $\bar{x} \vdash A$, all $\sigma \in \mathsf{St}$ and environments ρ we have* $(\mathsf{error}, \sigma) \notin [\![\bar{x} \vdash A]\!]\rho$.

Proof. Immediate from the definition of $[\![\bar{x} \vdash A]\!]\rho$. □

Lemma 3.3 (Soundness of Subspecification). *Suppose $\bar{x} \vdash A <: B$. Then, for all environments ρ, $[\![\bar{x} \vdash A]\!]\rho \subseteq [\![\bar{x} \vdash B]\!]\rho$ for values $\bar{v}$.*

Proof. This follows by induction on the derivation of $\bar{x} \vdash A <: B$. The cases for reflexivity and transitivity are immediate. For the case where both A and B are object specifications we need a similar lemma for transition relations:

If $\bar{x} \vdash T$ and $\bar{x} \vdash T'$ then $\vdash_{\mathsf{fo}} T \to T'$ implies

$$[\![\bar{x} \vdash T]\!]\rho \subseteq [\![\bar{x} \vdash T']\!]\rho \tag{3}$$

$$[\![\bar{x} \;\square\; A]\!] : \mathrm{Env} \to \mathrm{P}(\mathrm{Val} \times \mathrm{St})$$

$$[\![\bar{x} \;\square\; Bool]\!]\rho = \mathrm{BVal} \times \mathrm{St}$$

$$[\![\bar{x} \;\square\; [\mathrm{f}_i : A_i{}^{i=1\ldots n}, \mathrm{m}_j : \varsigma(y_j) y$$

A store specification Σ assigns *closed* specifications □ A to (a finite set of) locations:

Definition 4.1 (Store Specification). *A record* $\Sigma \in \mathsf{Rec}_{\mathsf{Loc}}(\mathit{Spec})$ *is a* store specification *if for all* $l \in \mathit{dom}(\Sigma)$, $\Sigma.l = A$ *is a closed object specification.*

Because we focus on closed specifications in the following, we need a way to turn the components B_j of a specification $[\mathsf{f}_i\colon A_i{}^{i=1\ldots n}, \mathsf{m}_j\colon \varsigma(y_j)B_j{::}T_j{}^{j=1\ldots m}]$ (which closed1

Observe that for all A, if $\Sigma' \leq \Sigma$ then $\|A\|_{\Sigma} \subseteq \|A\|_{\Sigma'}$. We obtain the following lemma about *context extensions*.

Lemma 4.5 (Context Extension). *If* $\rho \in \|\Gamma\|_{\Sigma}$ *and* $\Gamma, x{:}A \vdash \mathit{ok}$ *and* $v \in \|A[\rho/\Gamma]\|_{\Sigma}$ *then* $\rho[x := v] \in \|\Gamma, x{:}A\|_{\Sigma}$.

Proof. The result follows immediately from the definition once we show $\rho[x := v] \in \|\Gamma\|_{\Sigma}$. This can be seen to hold since $x \notin \mathsf{dom}(\Gamma)$, hence for all $y{:}B$ in Γ we know that x is not free in B and we must have $B[\rho[x := v]/\Gamma] \equiv B[\rho/\Gamma]$. ∎

We want to interpret store specifications as predicates over stores, as follows.

Definition 4.6 (Store Predicate, Tentative). *Let* $\mathsf{P} = \mathsf{P}(\mathsf{St})^{\mathsf{Rec}_{\mathsf{Loc}}(\mathit{Spec})}$ *denote the collection of predicates on* St*, indexed by store specifications. We define a functional* $\Phi : \mathsf{P}^{op} \times \mathsf{P} \to \mathsf{P}$ *as follows.*

$\sigma \in \Phi(Y, X)_{\Sigma} \;:\Leftrightarrow$
$\quad \forall l \in \mathit{dom}(\Sigma)$ *where* $\Sigma.l = [f_i{:}\,A_i{}^{i=1\ldots n},\ m_j{:}\,\varsigma(y_j)B_j{::}T_j{}^{j=1\ldots m}]$:
$\quad\quad$ **(F)** $\sigma.l.f_{i}$

resulting object has to be allocated in the store, and so a proper extension of the original store specification Σ has to be found.

So let $A_0 \equiv [\mathsf{m}_1 : \varsigma(y)[]::\mathit{False}]$ and $A_{i+1} \equiv [\mathsf{m}_1 : \varsigma(y)A_i::\mathit{True}]$. In particular, this means that the method m_1 of objects satisfying A_0 *must* diverge. The method m_1 of an object satisfying A_i returns an object satisfying A_{i-1}. Hence, for such objects x, it is possible to have method calls $x.\mathsf{m}_1.\mathsf{m}_1 \ldots \mathsf{m}_1$ at most i times, of which the i-th call must necessarily diverge (the others may or may not terminate). The example below uses the fact that we can construct an ascending chain of objects for which the first $i-1$ calls indeed terminate, and therefore do *not* satisfy A_{i-1}. Then, the limit of this chain is an object x for which an arbitrary number of calls $x.\mathsf{m}_1.\mathsf{m}_1 \ldots \mathsf{m}_1$ terminates, and which therefore does not satisfy *any* of the A_i:

Set $\Sigma_i'' \equiv \Sigma, l : A_i$ and let $\underline{\sigma} \in [\![\Sigma]\!]$ denote some store satisfying Σ. Moreover, define

$$\sigma_i = \{\!|l_0 = \{\!|\mathsf{m}_0 = \lambda\_.(l, \underline{\sigma} + \sigma_i'')|\!\}|\!\}$$

where $\sigma_0'' = \{\!|l = \{\!|\mathsf{m}_1 = \lambda\_.\bot|\!\}|\!\}$ and $\sigma_{i+1}'' = \{\!|l = \{\!|\mathsf{m}_1 = \lambda\_.(l, \underline{\sigma} + \sigma_i'')|\!\}|\!\}$, and let $\sigma = \bigsqcup_i \sigma_i$. Finally, define $X, Y \in \mathsf{P}$ by

$$\begin{aligned} X_{\Sigma_i''} &= \{\underline{\sigma} + \sigma_i''\}, \text{ for } i \in \mathsf{N} \\ X_{\hat{\Sigma}} &= \emptyset, \text{ for all other } \hat{\Sigma} \\[1ex] Y_\Sigma &= \{\underline{\sigma}\} \\ Y_{\hat{\Sigma}} &= \emptyset, \text{ for all other } \hat{\Sigma} \end{aligned}$$

By construction, both X and Y are admissible in every component $\hat{\Sigma}$. By induction one obtains $\sigma_0'' \sqsubseteq \sigma_1'' \sqsubseteq \ldots$, therefore $\sigma_0 \sqsubseteq \sigma_1 \sqsubseteq \ldots$ in $\Phi(Y, X)_\Sigma$. Hence we must show $\sigma \in \Phi(Y, X)_\Sigma$. But this is not the case, since it would entail, by **(M2)** and

$$\sigma.l.\mathsf{m}_0(\underline{\sigma}) = \bigsqcup_i \sigma_i.l.\mathsf{m}_0(\underline{\sigma}) = (l, \underline{\sigma} + \bigsqcup_i \sigma_i'')$$

that there exists $\Sigma'' \geq \Sigma$ such that $\underline{\sigma} + \bigsqcup_i \sigma_i'' \in X_{\Sigma''}$. Clearly this is not the case, since $\underline{\sigma} + \bigsqcup_i \sigma_i''$ is *strictly* greater than every $\underline{\sigma} + \sigma_i''$ and therefore not in any of the $X_{\Sigma_i''}$.

4.3 A Refined Semantics of Store Specifications

We refine the definition of store predicates by replacing the existential quantifier in **(M2)** of Definition 4.6 by a *Skolem function*, as follows: We call the elements of the (recursively defined) domain

$$\varphi \in \mathsf{RSF} = \mathsf{Rec}_{\mathsf{Loc}}(\mathsf{Rec}_{\mathsf{M}}(\mathsf{St} \times \mathsf{RSF} \times \mathit{Spec} \rightharpoonup \mathit{Spec} \times \mathsf{RSF})) \tag{4}$$

choice functions, or *Skolem Functions*. The intuition is that, given a store $\sigma \in [\![\Sigma]\!]$, if $\sigma' \in [\![\Sigma']\!]$ with choice function φ', for some extension $\Sigma' \geq \Sigma$ and the method invocation $\sigma.l.\mathsf{m}(\sigma')$ terminates, then $\varphi.l.\mathsf{m}(\sigma', \varphi', \Sigma') = (\Sigma'', \varphi'')$ yields a store specification $\Sigma'' \supseteq \Sigma'$ such that $\sigma'' \in [\![\Sigma'']\!]$ (and φ'' is a choice function for the extension Σ'' of Σ). This is again an abstraction of the actual store σ, this time abstracting the *dynamic effects* of methods wrt. allocation, on the level of store specifications. Note that the argument store σ' is needed in general to determine the resulting extension of the specification, since allocation behaviour may depend on the actual values of fields, for example.

We use the domain RSF of choice functions explicitly in the interpretation of store specifications below. This has the effect of constraining the existential quantifier to work *uniformly* on the elements of increasing chains, hence precluding the counter-example to admissibility of the previous subsection.

Definition 4.7 (Store Predicate). *Let* $\mathsf{P} = \mathcal{P}(\mathsf{St} \times \mathit{RSF})^{\mathsf{Rec}_{\mathsf{Loc}}(\mathit{Spec})}$ *denote the collection of families of subsets of* $\mathsf{St} \times \mathit{RSF}$*, indexed by store specifications. We define a functional* $\Phi : \mathsf{P}^{op} \times \mathsf{P} \to \mathsf{P}$ *as follows.*

$(\sigma, \varphi) \in \Phi(Y, X)_\Sigma :\Leftrightarrow$

(1) $\mathit{dom}(\Sigma) = \mathit{dom}(\varphi)$ *and* $\forall l \in \mathit{dom}(\Sigma).\ \mathit{dom}(\pi_2(\Sigma.l)) = \mathit{dom}(\varphi.l)$*, and*

(2) $\forall l \in \mathit{dom}(\Sigma)$ *where* $\Sigma.l = [f_i : A_i{}^{i=1\ldots n},\ m_j : \varsigma(y_j)B_j :: T_j{}^{j=1\ldots m}$

we obtain $\sigma.l.\mathsf{f}_i \in \|A_i\|_\Sigma$ by assumption $(\sigma_j, \varphi_j) \in \Phi(Y, X)_\Sigma$. Next, suppose $\Sigma' \supseteq \Sigma$, $(\sigma', \varphi') \in Y_{\Sigma'}$ and $\sigma.l.\mathsf{m}_j(\sigma') = (v, \sigma'')\downarrow$. By definition of σ as $\bigsqcup_k \sigma_k$ and continuity, we must have $\sigma_k.l.\mathsf{m}_j(\sigma') = (v, \sigma''_k)\downarrow$ for sufficiently large k, and

$$(v, \sigma'') = \bigsqcup_k \sigma_k.l.\mathsf{m}_j(\sigma') = \bigsqcup_k (v, \sigma''_k)$$

By assumption, for all sufficiently large k, $\varphi_k.l.\mathsf{m}_j(\sigma', \varphi', \Sigma') = (\Sigma''_k, \varphi''_k)$ with $\Sigma''_k \supseteq \Sigma'$ and

- $(\pi_{\mathrm{Val}}(\sigma'), v, \pi_{\mathrm{Val}}(\sigma''_k)) \in [\![T_j[l/\mathsf{T}$

$F_{\mathrm{St,RSF}}(e, e)(\sigma, \varphi) \square \Phi(Y^{\square}, X^{\square})_{\Sigma}$ which proves (

5 Soundness

5.1 Preliminaries

Recall from the previous section that the semantics of store specifications is defined in terms of the semantics $\|A\|_\Sigma$ for result specifications A that does not mention St at all. The following key lemma establishes the relation between store specifications and object specifications $[\![\vdash A]\!]$ as defined in Section 3.3:

Lemma 5.1. *For all object specifications A, store specifications Σ, stores σ, and locations l, if $\sigma \models [\![\Sigma]\!]$ and $l \in \mathit{dom}(\Sigma)$ such that $\vdash \Sigma.l <: A$ then $(l, \sigma) \in [\![A]\!]$.*

Proof. By induction on the structure of A. Because A is an object specification it is necessarily of the form

$$A \equiv [\mathsf{f}_i \colon A_i{}^{i=1\ldots n},\ \mathsf{m}_j \colon \varsigma(y_j)B_j{::}T_j{}^{j=1\ldots m}]$$

We have to show that $(l, \sigma) \in [\![\vdash A]\!]$, i.e., that

- $(\sigma.l.\mathsf{f}_i, \sigma) \in [\![\vdash A_i]\!]$ for all $1 \le i \le n$ and
- if $\sigma.l.\mathsf{m}_j(\sigma) = (v, \sigma')$ then $(v, \sigma') \in [\![y_j \vdash B_j]\!](y_j \mapsto l)$ and $(\pi_{\mathrm{Val}}\, \sigma, v, \pi_{\mathrm{Val}}\, \sigma') \in [\![y_j \vdash T_j]\!](y_j \mapsto l)$ for all $1 \le j \le m$.

From the subtyping relation and $\Sigma.l <: A$ we find

$$\Sigma.l \equiv [\mathsf{f}_i{:}A_i{}^{i=1\ldots n+p}, \mathsf{m}_j{:}\varsigma(y_j)B'_j{::}T'_j{}^{j=1\ldots m+p}]$$

where $y_j \vdash B'_j <: B_j$ and $y_j \vdash_{\mathsf{fo}} T'_j \to T_j$.

For the first part, by Definition 4.7 **(F)** and $\sigma \models [\![\Sigma]\!]$ we have $\sigma.l.\mathsf{f}_i \in \|A_i\|_\Sigma$. If A_i is *Bool* then from $\|Bool\|_\Sigma = \mathsf{BVal}$, hence, $(\sigma.l.\mathsf{f}_i, \sigma) \in [\![\vdash Bool]\!]$. Otherwise A_i is an object specification and the definition of $\|A_i\|_\Sigma$ implies

$$\vdash \Sigma.(\sigma.l.\mathsf{f}_i) <: A_i$$

again by Definition 4.7 **(F)**. Hence by induction hypothesis we obtain $(\sigma.l.\mathsf{f}_i, \sigma) \in [\![\vdash A_i]\!]$ as required.

For the second part, suppose that $\sigma.l.\mathsf{m}_j(\sigma) = (v, \sigma'')$. From Definition 4.7 part **(M2)** and **(M3)**, and the assumption $\sigma \models [\![\Sigma]\!]$, we find $v \in \|B'_j[l/y_j$

$[\![\vdash B'_j[l/y_j]]\!]$. Thus,

$$
\begin{aligned}
(v, \sigma'') \in [\![\vdash B'_j[l/y_j]]\!] &= [\![y_j \vdash B'_j]\!](y_j \mapsto l) && \text{Lemma 4.2} \\
&\subseteq [\![y_j \vdash B_j]\!](y_j \mapsto l) && \text{Lemma 3.3}
\end{aligned}
$$

as required.

Finally, by Definition 4.7 **(M1)** we obtain

$$
\begin{aligned}
(\pi_{\mathrm{Val}}\,\sigma, v, \pi_{\mathrm{Val}}\,\sigma'') \in [\![\vdash T'_j[l/y_j]]\!] &= [\![y_j \vdash T'_j]\!](y_j \mapsto l) && \text{Lemma 4.2} \\
&\subseteq [\![y_j \vdash T_j]\!](y_j \mapsto l) && \text{soundness of } \vdash_{\mathrm{fo}}
\end{aligned}
$$

This concludes the proof. ∎

We can now define the semantics of judgements of Abadi-Leino logic and prove the key lemma.

Definition 5.2 (Validity). $\Gamma \models a : A :: T$ *if and only if for all store specifications* $\Sigma \in \mathrm{Rec}_{\mathrm{Loc}}(\mathit{Spec})$*, for all* $\rho \in \|\Gamma\|_{\Sigma}$ *and all* $\sigma \in [\![\Sigma]\!]$*, if* $[\![a]\!]\rho\sigma = (v, \sigma')$

Thus,

$$(\pi_{\mathrm{Val}}\, \sigma, v', \pi_{\mathrm{Val}}\, \sigma', \pi_{\mathrm{Val}}\, \sigma'') \in$$
$$[\![\overline{x}, x \vdash T'[\mathsf{sel}_{int}(\cdot,\cdot)/\mathsf{sel}_{post}(\cdot,\cdot), \mathsf{alloc}_{int}(\cdot)/\mathsf{alloc}_{post}(\cdot), x/\mathsf{result}]]\!]\rho[x := v]$$

since there are no occurrences of $\mathsf{sel}_{post}(\cdot,\cdot)$, $\mathsf{alloc}_{post}(\cdot)$ *post* post

Note that by the subtyping rules, $A \equiv Bool$ if and only if $A' \equiv Bool$. In this case *(S3)* follows directly from *(S3')*. In the case where A' is an object specification, assumption [Γ

(S1) $\varphi(\sigma', \varphi', \Sigma') = (\Sigma', \varphi')$

(S2) $(\sigma'', \varphi') = (\sigma', \varphi') \in \mathit{fix}(\Phi)_{\Sigma'}$

(S3) $v = \mathit{true} \in \mathsf{BVal} = \|A\|_{\Sigma'}$

(S4) $(\pi_{\mathsf{Val}}(\sigma'), \mathit{true}, \pi_{\mathsf{Val}}(\sigma'')) \in [\![[\Gamma] \vdash T_{\mathsf{res}}(\mathit{true})]\!]\rho$ by definition

as required. The case where a is `false` is analoguous.

- **Conditional**
By a case distinction, depending on whether the value of the guard x is *true* or *false*.

- **Let**
Suppose *(H1)* $\Gamma \vdash a : A :: T$ has been derived by an application of the (Let) rule. Hence, a is `let` $x = a_1$ `in` a_2. Assume that

(H2) Σ is a store specification, and

(H3) $\rho \in \|\Gamma\|_{\Sigma}$

Now recall the rule for this case,

$$\frac{\begin{array}{c}\Gamma \vdash a_1{:}A_1{::}T_1 \quad \Gamma, x{:}A_1 \vdash a_2{:}A{::}T_2 \quad [\Gamma] \vdash A \quad [\Gamma] \vdash T \\ \vdash_{\mathsf{fo}} T_1[\mathsf{sel}_{int}(\cdot,\cdot)/\mathsf{sel}_{post}(\cdot,\cdot), \mathsf{alloc}_{int}(\cdot)/\mathsf{alloc}_{post}(\cdot), x/\mathsf{result}] \\ \wedge\, T_2[\mathsf{sel}_{int}(\cdot,\cdot)/\mathsf{sel}_{pre}(\cdot,\cdot), \mathsf{alloc}_{int}(\cdot)/\mathsf{alloc}_{pre}(\cdot)] \to T\end{array}}{\Gamma \vdash \texttt{let } x = a_1 \texttt{ in } a_2{:}A{::}T}$$

By the premiss of this rule we must have

(H1') $\Gamma \vdash a_1 : A_1 :: T_1$

(H1") $\Gamma, x{:}A_1 \vdash a_2 : A :: T_2$

By induction hypothesis applied to *(H1')* there is $\varphi_1 \in \mathsf{SF}$ s.t. for all $\Sigma' \supseteq \Sigma$, $(\sigma', \varphi') \in \mathit{fix}(\Phi)_{\Sigma'}$ with $[\![a_1]\!]\rho\sigma' = (\hat{v}, \hat{\sigma})$, the conclusions of the lemma hold:

(S1') there exists $\hat{\Sigma} \supseteq \Sigma'$ and $\hat{\varphi} \in \mathsf{RSF}$ s.t. $\varphi(\sigma', \varphi', \Sigma') = (\hat{\Sigma}, \hat{\varphi})$

(S2') $(\hat{\sigma}, \hat{\varphi}) \in \mathit{fix}(\Phi)_{\hat{\Sigma}}$

σ σ all 1 Td[(^)]TJ/F103 9.963 Tf -1.37 -6.763 4d[(^)]TJ/F90 9.963 Tf 5.48 0 Td[(,1)]TJ/F101 9.963 Tf
(S3') v̂1(,)]TJ/J/F103 9.963 Tf 3.318lTJ/F101 9.9663 .317 0 Td[(;)13 9.963 Tf 2.491 0 Td[(;)]TJ/F27 9.963 Tf 4.151 0 T

(S1") there exists $\Sigma'' \in \hat{\Sigma}$ and $\varphi'' \in \mathsf{RSF}$ s.t. $\varphi_{\hat{v}}(\hat{\sigma}, \hat{\varphi}, \hat{\Sigma}) = (\Sigma'', \varphi'')$

(S2") $(\sigma'', \varphi'') \in \mathit{fix}(\Phi)_{\Sigma''}$

(S3") $v \in \|A[\rho[x := v]/\Gamma, x{:}A_1]\|_{\Sigma''}$

(S4") $(\pi_{\mathrm{Val}}(\hat{\sigma}), v, \pi_{\mathrm{Val}}(\sigma'')) \in [\![[\Gamma, x{:}A_1] \to T_2]\!]\rho$

Now define $\varphi \in \mathsf{SF}$ for all σ', φ' and Σ' by

$$\varphi(\sigma', \varphi', \Sigma') = \left\{ \right.$$

(S4) $(\pi_{\mathrm{Val}}(\sigma'), v, \pi_{\mathrm{Val}}(\sigma'')) \in [\![\Gamma \vdash T]\!]\rho$, by (9)

as required.

- **Object**
  Suppose *(H1)*: $\Gamma \vdash a : A :: T$ has been derived by an application of rule the (object construction) rule. Necessarily $a \equiv [\mathsf{f}_i = x_i{}^{i=1..n}, \mathsf{m}_j = \varsigma(y_j)b_j{}^{j=1..m}]$. Suppose that

(H2) Σ is a store specification

(H3) $\rho \in \|\Gamma\|_\Sigma$

We recall the object introduction rule,

$$\frac{\begin{array}{c} A \equiv [\mathsf{f}_i{:}\,A_i{}^{i=1...n},\ \mathsf{m}_j{:}\,\varsigma(y_j)B_j{::}T_j{}^{j=1...m}] \\ \Gamma \vdash x_i{:}A_i :: T_{\mathsf{res}}(x_i)^{i=1...n} \quad \Gamma, y_j{:}A \vdash b_j{:}B_j{::}T_j{}^{j=1...m} \end{array}}{\Gamma \vdash [\mathsf{f}_i = x_i{}^{i=1...n}, \mathsf{m}_j = \varsigma(y_j)b_j{}^{j=1...m}]{:}A{::}T_{\mathsf{obj}}(\mathsf{f}_1 = x_1 \ldots \mathsf{f}_n = x_n)}$$

from which we see that A is $[\mathsf{f}_i{:}A_i, \mathsf{m}_j{:}B_j{::}T_j]$, that T is $T_{\mathsf{obj}}(\mathsf{f}_1 = x_1 \ldots \mathsf{f}_n = x_n)$ and that

(H1') $\Gamma \vdash x_i : A_i :: T_{\mathsf{res}}(x_i)$ for $1 \leq i \leq n$

(H1") $\Gamma, y_j{:}A \vdash b_j : B_j :: T_j$ for $1 \leq j \leq m$

We have to show that there is $\varphi \in \mathsf{SF}$ s.t. for all $\Sigma' \sqsupseteq \Sigma$, $(\sigma', \varphi') \in \mathit{fix}(\Phi)_{\Sigma'}$ with $[\![a]\!]\rho\sigma' = (v, \sigma'')$, *(S1)-(S4)* hold.

From *(H3)* and Lemma 4.5 we know that for all $\hat{\Sigma} \sqsupseteq \Sigma$ and $l_0 \notin \mathsf{dom}(\hat{\Sigma})$,

$$\rho[y_j := l_0] \in \|\Gamma, y_j{:}A\|_{\hat{\Sigma}, l_0:A}$$

Hence by induction hypothesis on *(H1")*, there is $\varphi^j_{l_0} \in \mathsf{SF}$ for all $1 \leq j \leq m$ s.t. for all $\Sigma_1 \sqsupseteq (\hat{\Sigma}, l_0{:}A[\rho/\Gamma])$, for all $(\sigma_1, \varphi_1) \in \mathit{fix}(\Phi)_{\hat{\Sigma}, l_0:A[\rho/\Gamma]}$ with $[\![b_j]\!]\rho[y_j := l_0]\sigma_1 = (v_2, \sigma_2) \downarrow$, we obtain the conclusions *(S1)-(S4)* of the lemma, i.e.,

(S1') there exists $\Sigma_2 \sqsupseteq \Sigma_1$ and $\varphi_2 \in \mathsf{RSF}$ s.t. $\varphi^j_{l_0}(\sigma_1, \varphi_1, \Sigma_1) = (\Sigma_2, \varphi_2)$

(S2') $(\sigma_2, \varphi_2) \in \mathit{fix}(\Phi)_{\Sigma_2}$

(S3') $v_2 \in \|B_j[\rho[y_j := l_0]/\Gamma, y_j{:}A]\|_{\Sigma_2}$

(S4') $(\pi_{\mathrm{Val}}(\sigma_1), v_2, \pi_{\mathrm{Val}}(\sigma_2)) \in [\![[\Gamma, y_j{:}A] \vdash T_j]\!]\rho[y_j := l_0]$

We have $\{l_0 = \{\mathsf{m}_j = \varphi^j_{l_0}\}^{j=1...m}\} \in \mathsf{RSF}$, therefore we can define $\varphi \in \mathsf{SF}$ by

φ(σ′, φ′, Σ′T((5.54 0 Td[(]),)-327(1 0 Td[(=)]TJ/F351)]TJ/F90 9.963 Td[(:)]TJ/F1636

We show that *(S1)–(S4)* hold. Let $\Sigma' \subseteq \Sigma$, $(\sigma', \varphi') \in \mathit{fix}(\Phi)_{\hat{\Sigma}}$ and suppose $[\![a]\!]\rho\sigma' = (v, \sigma'')$. By9J/F90 9.963 T050S1)

(F) By assumption *(H1')* and $\rho \in \|\Gamma\|_\Sigma$ we know that there is $\vdash A_i' <: A_i$ for all $1 \le i \le n$ s.t. $x_i{:}A_i'$ in Γ. Hence,

$$\sigma''.l_0.\mathsf{f}_i = \rho(x_i) \in \left(\!\left| A_i' \right|\!\right)_\Sigma \subseteq \|A_i\|_\Sigma \subseteq \|A_i\|_{\Sigma''}$$

(M) Let $1 \le j \le m$. Suppose $\Sigma_1 \geq \Sigma''$, let $(\sigma_1, \varphi_1) \in \mathit{fix}(\Phi)_{\Sigma_1}$ and suppose $\sigma''.l_0.\mathsf{m}_j(\sigma_1) = (v_2, \sigma_2)$. Since $\sigma''.l_0.\mathsf{m}_j = [\![b_j]\!]\rho[y_j := l_0]\sigma_1$ and $\Sigma_1 \geq \Sigma'$, the assumption $(\sigma_1, \varphi_1) \in \mathit{fix}(\Phi)_{\Sigma'}$ and the construction of φ'' give Σ_2 and φ_2 s.t.

- $\varphi''.l_0.\mathsf{m}_j(\sigma_1, \varphi_1, \Sigma_1) = \varphi^j_{l_0}(\sigma_1, \varphi_1, \Sigma_1) = (\Sigma_2, \varphi_2)$, by *(S1')*
- $(\sigma_2, \varphi_2) \in \mathit{fix}(\Phi)_{\Sigma_2}$, by *(S2')*
- $v_2 \in \left(\!\left| B_j[\rho[y_j := l_0]/\Gamma, y_j{:}A] \right|\!\right)_{\Sigma_2} = \left(\!\left| B_j[\rho/\Gamma][l_0/y_j] \right|\!\right)_{\Sigma_2}$, by *(S3')* and the substitution lemma, Lemma 4.2
- $(\pi_{\mathrm{Val}}(\sigma_1), v_2, \pi_{\mathrm{Val}}(\sigma_2)) \in [\![[\Gamma, y_j{:}A] \vdash T_j]\!]\rho[y_j := l_0]$ which equals $[\![T_j[\rho/\Gamma][l_0/y_j]]\!]$, by *(S4')* and the substitution lemma

Thus we have shown $(\sigma'', \varphi'') \in \mathit{fix}(\Phi)_{\Sigma''}$, i.e., *(S2)* holds.

- **Method Invocation**

Suppose $\Gamma \vdash a : A :: T$ is derived by an application of the method invocation rule:

$$\frac{\Gamma \vdash x{:}[\mathsf{m}{:}\varsigma(y)A'{::}T']{::}T_{\mathrm{res}}(x)}{\Gamma \vdash x.\mathsf{m}{:}A'[x/y]{::}T'[x/y]}$$

Necessarily a is of the form $x.\mathsf{m}$ and there are A' and T' s.t. $A \equiv A'[x/y]$ and $T \equiv T'[x/y]$. So suppose

(H1) $\Gamma \vdash a : A'[x/y] :: T'[x/y]$

(H2) Σ is a store specification

(H3) $\rho \in \|\Gamma\|_\Sigma$

Define $\varphi \in \mathsf{SF}$ using "self-application" of the argument,

$$\varphi(\sigma', \varphi', \Sigma') = \varphi'.\rho(x).\mathsf{m}(\sigma', \varphi', \Sigma') \tag{12}$$

Now let $\Sigma' \geq \Sigma$, $(\sigma', \varphi') \in \mathit{fix}(\Phi)_{\Sigma'}$ and suppose $[\![a]\!]\rho\sigma' = \sigma'.\rho(x).\mathsf{m}(\sigma') = (v, \sigma'')$ terminates. We show that *(S1)-(S4)* hold.

By the hypothesis of the method invocation rule,

$$\Gamma \vdash x{:}[\mathsf{m}{:}\varsigma(y)A'{::}T']{::}T_{\mathrm{res}}(x) \tag{H1'}$$

Since this implies $x{:}B \in \Gamma$ for some $[\Gamma] \vdash B <: [\mathsf{m} : \varsigma(y)A' :: T']$, by assumption *(H3)* this entails

$$\vdash \Sigma.(\rho(x)) <: [\mathsf{m} : \varsigma(y)A' :: T']\,[\rho/\Gamma]$$

i.e., there are A_i, A'', B_j and T_j, T'' such that

$$\vdash \Sigma.\rho(x) \equiv [\mathsf{f}_i{:}A_i, \mathsf{m}_j{:}\varsigma(y_j)B_j :: T_j, \mathsf{m}{:}\varsigma(y)A''{::}T'']$$

where

$$y \vdash A'' <: A'[\rho/\Gamma] \text{ and } \vdash_{\mathsf{fo}} T'' \rightarrow T'[\rho/\Gamma] \tag{13}$$

Now assumption $(\sigma', \varphi') \in \mathit{fix}(\Phi)_{\Sigma'}$ with equation (12) implies that there are Σ'', φ'' s.t.

(S1) $\varphi(\sigma', \varphi', \Sigma') = \varphi'.(\rho(x)).\mathsf{m}(\sigma', \varphi', \Sigma') = (\Sigma'', \varphi'')$

(S2) $(\sigma'', \varphi'') \in \mathit{fix}(\Phi)_{\Sigma''}$

(S3') $v \in \|A''[\rho(x)/y]\|_{\Sigma''}$

(S4') $(\pi_{\mathrm{Val}}(\sigma'), v, \pi_{\mathrm{Val}}(\sigma'')) \in [\![\vdash T''[\rho(x)/y]]\!]$

By transitivity of $<:$, equation (13), Lemma 4.2 and *(S3')*

$$v \in \|A'[\rho/\Gamma][\rho(x)/y]\|_{\Sigma''}$$

Since $A'[\rho/\Gamma, \rho(x)/y] \equiv A'[x/y][\rho/\Gamma]$ we also have

(S3) $v \in \|A'[x/y][\rho/\Gamma]\|_{\Sigma''} = \|A[\rho/\Gamma]\|_{\Sigma''}$

Similarly, by (13) and *(S4')*,

$$\begin{aligned}(\pi_{\mathrm{Val}}(\sigma'), v, \pi_{\mathrm{Val}}(\sigma'')) \in [\![T''[\rho(x)/y]]\!] &\subseteq [\![T'[\rho/\Gamma][\rho(x)/y]]\!] \\ &= [\![[\Gamma] \vdash T[x/y]]\!]\rho \end{aligned} \tag{S4}$$

which was to show.

- **Field Selection**
  Similar. φ can be chosen as $\varphi(\sigma', \varphi', \Sigma') = (\varphi', \Sigma')$.

- **Field Update**
  Suppose

(H1) $\Gamma \vdash a$

In particular, a is of the form $x.\mathsf{f}_k := y$ and T is $T_{\mathsf{upd}}(x, \mathsf{f}_k, y)$. From the semantics of $[\![a]\!]\rho\sigma'$, this means $v = \rho(x) \in \mathsf{Loc}$ and

$$\sigma'' = \sigma'[v := \sigma'.v[\mathsf{f}_k := \rho(y)]] \tag{14}$$

We show that *(S1)*-*(S4)* hold.

By *(H3)*, $\rho(x) \in \|A[\rho/\Gamma]\|_{\Sigma} \subseteq \|A[\rho/\Gamma]\|_{\Sigma'}$. Then by construction of φ, and (14),

(S1) $\varphi(\sigma', \varphi', \Sigma') = (\Sigma', \varphi')$

(S3) $v = \rho(x) \in \|A[\rho/\Gamma]\|_{\Sigma'}$

(S4) $(\pi_{\mathrm{Val}}(\sigma'), v, \pi_{\mathrm{Val}}(\sigma'')) \in [\![[\Gamma] \vdash T]\!]\rho$, from the semantics given in Tab. 4

It remains to show *(S2)*, $(\sigma'', \varphi') \in \mathit{fix}(\Phi)_{\Sigma'}$.

By assumption $(\sigma', \varphi') \in \mathit{fix}(\Phi)_{\Sigma'}$, condition (1) of Definition 4.7 is satisfied. As for condition (2), suppose $l \in \mathsf{dom}(\Sigma')$ s.t.

$$\Sigma'.l \equiv [\mathsf{g}_i{:}A_i'^{\,i=1\ldots p}, \mathsf{n}_j{:}\varsigma(y_j)B_j' :: T_j'^{\,j=1\ldots q}]$$

(F) We distinguish two cases:

- Case $l = \rho(x)$ and $\mathsf{g}_i = \mathsf{f}_k$. Then, by (14), $\sigma''.l.\mathsf{g}_i = \rho(y)$. By *(H3)*, $\rho(x) \in \|A[\rho/\Gamma]\|_{\Sigma} \subseteq \|A[\rho/\Gamma]\|_{\Sigma'}$, which entails

  $$\vdash \Sigma'.l <: A[\rho/\Gamma]$$

  and in particular, by the definition of the subspecification relation, $A_k' \equiv A_k[\rho/\Gamma]$. Note that *invariance of subspecification* in the field components is needed to conclude this. Now again by *(H3)*,

  $$\rho(y) \in \|A_k[\rho/\Gamma]\|_{\Sigma} \subseteq \|A_k[\rho/\Gamma]\|_{\Sigma'} = \|A_k'\|_{\Sigma'}$$

  Hence, $\sigma''.l.\mathsf{g}_i \in \|A_i'\|_{\Sigma'}$ as required.

- Case $l \neq \rho(x)$ or $\mathsf{g}_i \neq \mathsf{f}_k$. Then $\sigma''.l.\mathsf{g}_i = \sigma'.l.\mathsf{g}_i$, by (14). Hence, by assumption $(\sigma', \varphi') \in \mathit{fix}(\Phi)_{\Sigma'}$, we have $\sigma''.l.\mathsf{g}_i \in \|A_i'\|_{\Sigma'}$.

(M) Let $\Sigma'' \sqsupseteq \Sigma'$, let $(\sigma_1, \varphi_1) \in \mathit{fix}(\Phi)_{\Sigma''}$ and suppose $\sigma''.l.\mathsf{n}_j(\sigma_1) = (v_2, \sigma_2)$. Then, by assumption $(\sigma', \varphi') \in \mathit{fix}(\Phi)_{\Sigma'}$ and the fact that $\sigma''.l.\mathsf{n}_j = \sigma'.l.\mathsf{n}_j$ by (14), we obtain that $\varphi'.l.\mathsf{n}_j(\sigma_1, \varphi_1, \Sigma'') = (\Sigma_2, \varphi_2)$ s.t. $\Sigma_2 \sqsupseteq \Sigma''$ and

(M1) $(\pi_{\mathrm{Val}}(\sigma_1), v_2, \pi_{\mathrm{Val}}(\sigma_2)) \in [\![T_j'[l/y_j]]\!]$

(M2) $(\sigma_2, \varphi_2) \in \mathit{fix}(\Phi)_{\Sigma_2}$

(M3) $v_2 \in \|B_j'[l/y_j]\|_{\Sigma_2}$

as required.

which concludes the proof. ∎

5.3 Soundness Theorem

With Lemma 5.1 and Lemma 5.4, proved in Subsections 5.1 and 5.2, it is now easy to establish our main result:

Theorem 5.5 (Soundness). *If* $\Gamma \vdash a : A :: T$ *then* $\Gamma \models a : A :: T$.

Proof. Suppose $\Gamma \vdash a : A :: T$, and let $\Sigma \in \mathsf{Rec}_{\mathsf{Loc}}(\mathit{Spec})$ be a store specification and suppose $\rho \in \mathsf{Env}$ s.t. $\rho \models$

ness".

$$\underline{A}, \underline{B} ::= \top \mid Bool \mid [\mathsf{f}_i{:}\, A_i{}^{i=1\ldots n},\ \mathsf{m}_j{:}\, \varsigma(y_j)B_j{::}T_j{}^{j=1\ldots m}] \mid \mu(X)\underline{A}$$
$$A, B ::= \underline{A} \mid X$$

where X ranges over an infinite set *TyVar* of specification variables. X is bound in $\mu(X)A$, and as usual we identify specifications up to the names of bound variables.

In addition to specification contexts Γ we introduce contexts Δ that contain specification variables with an upper bound, $X <: A$, where A is either another variable or $\top$. In the rules of the logic we replace $\Gamma \vdash \ldots$ by $\Gamma; \Delta \vdash \ldots$, and the definitions of well-formed specifications and well-formed specification contexts are extended, similar to the case of recursive types [1].

$$\frac{\Gamma; \Delta \vdash Y \quad X \notin \Gamma}{\Gamma; \Delta, X <: Y \vdash \mathsf{ok}} \qquad \frac{\Gamma; \Delta \vdash \mathsf{ok} \quad X \notin \Gamma}{\Gamma; \Delta, X <: \top \vdash \mathsf{ok}}$$

and

$$\frac{\Gamma; \Delta, X <: A, \Delta' \vdash \mathsf{ok}}{\Gamma; \Delta, X <: A, \Delta' \vdash X} \qquad \frac{\Gamma; \Delta, X <: \top \vdash A}{\Gamma; \Delta \vdash \mu(X)A} \qquad \frac{\Gamma; \Delta \vdash \mathsf{ok}}{\Gamma; \Delta \vdash \top}$$

and we often write Δ, X for $\Delta, X <: \top$.

Subspecifications for recursive specifications are obtained by the "usual" recursive subtyping rule [3], and $\top$ is the greatest specification,

$$\frac{\Gamma; \Delta, Y <: \top, X <: Y \vdash A <: B}{\Gamma; \Delta \vdash \mu X.A <: \mu Y.B} \qquad \frac{\Gamma; \Delta \vdash A}{\Gamma; \Delta \vdash A <: \top}$$

As will be seen from the semantics below, in our model a recursive specification and its unfolding are not just isomorphic but equal, i.e., $[\![\mu X.A]\!] = [\![A[(\mu X.A)/X]]\!]$. Because of this, we do not need to introduce *fold* and *unfold* terms: We can deal with (un)folding of recursive specifications through the subsumption rule once we add the following subspecifications,

$$\text{fold}\ \frac{\Gamma; \Delta \vdash \mu X.A}{\Gamma; \Delta \vdash A[(\mu X.A)/X] <: \mu X.A} \qquad \text{unfold}\ \frac{\Gamma; \Delta \vdash \mu X.A}{\Gamma; \Delta \vdash \mu X.A <: A[(\mu X.A)/X]}$$

We will prove their soundness below.

6.1 Existence of Store Specifications

Next, we adapt our notion of store specification to recursive specifications. The existence proof is very similar to the one given in Section 4, however, for completeness we spell it out in detail below.

Definition 6.1. *A store specification is a record $\Sigma \in \mathsf{Rec}_{\mathsf{Loc}}(Spec)$ such that for each $l \in dom(\Sigma)$,*

$$\Sigma.l = \mu(X)[f_i{:}\, A_i{}^{i=1\ldots n},\ m_j{:}\, \varsigma(y_j)B_j{::}T_j{}^{j=1\ldots m}]$$

is a closed (recursive) object specification.

i.e., $f(\bigsqcap_i x_i) = \bigsqcap_i f(x_i)$, the greatest fixed point can be obtained as

$$\mathsf{gfp}(f) = \bigsqcap\{f^n(\top) \mid n \in \mathbb{N}\} \tag{15}$$

where $\top$ is the greatest element of L: Writing $\alpha = \bigsqcap\{f^n(\top) \mid n \in \mathbb{N}\}$ it is immediate that $f(\alpha) = \bigsqcap\{f^{n+1}(\top) \mid n \in \mathbb{N}\} =$

Preservation of Meets. $[\![\Gamma;\Delta \vdash A]\!]$ *preserves meets of descending chains:*

$$\eta_0 \geq \eta_1 \geq \dots \implies [\![\Gamma;\Delta \vdash A]\!]\rho(\textstyle\bigsqcap_i \eta_i) = \textstyle\bigsqcap_i [\![\Gamma;\Delta \vdash A]\!]\rho\eta_i$$

In particular, this lemma shows that the greatest fixed point used in Definition 6.4 exists, by the observations made above.

Proof. We can show both properties simultaneously by induction on the structure of A. The only interesting case is where A is $\mu(X)B$.

To show the first part, **Monotonicity**, note that the assumption $\eta_1 \leq \eta_2$ entails

$$\eta_1[X = \chi_1] \leq \eta_2[X = \chi_2] \quad \text{for all } \chi_1 \sqsubseteq \chi_2 \in \mathsf{A}dm(\mathsf{Val} \times \mathsf{St})$$

So for $f_i : \mathsf{A}dm(\mathsf{Val} \times \mathsf{St}) \to \mathsf{A}dm(\mathsf{Val} \times \mathsf{St})$ defined by

$$f_i(\chi) = [\![\Gamma;\Delta, X \vdash B]\!]\rho\eta_i[X = \chi], \qquad i = 1, 2$$

we obtain from the induction hypothesis on B that f_i is monotonic, preserves meets, and $f_1 \leq f_2$. By the observations made above, gfp is monotonic which yields $\mathsf{gfp}(f_1) \sqsubseteq \mathsf{gfp}(f_2)$. Thus

$$[\![\Gamma;\Delta \vdash \mu(X)B]\!]\rho\eta_1 = \mathsf{gfp}(f_1) \sqsubseteq \mathsf{gfp}(f_2) = [\![\Gamma;\Delta \vdash \mu(X)B]\!]\rho\eta_2$$

which concludes the proof □

Lemma 6.6 (Substitution). *For all* $\Gamma; \Delta, X \vdash A$, $\Gamma; \Delta \vdash B$, ρ *and* η,

$$[\![\Gamma; \Delta, X \vdash A]\!]\rho(\eta[X{=}[\![\Gamma; \Delta \vdash B]\!]\rho\eta]) = [\![\Gamma; \Delta \vdash A[B/X]]\!]\rho\eta$$

Proof. By induction on A. □

6.3 Syntactic Approximations

Recall the statement of Lemma 5.1, one of the key lemmas in the proof of the soundness theorem:

$$\text{for all } \sigma, \Sigma, l \text{ and } \vdash A,\ \text{ if } \sigma \in [\![\Sigma]\!] \text{ and } \vdash \Sigma.l <: A \text{ then } (l, \sigma) \in [\![\vdash A]\!] \tag{18}$$

In Section 5 this was proved by induction on the structure of A. This inductive proof cannot be extended directly to prove a corresponding result for recursive specifications: The recursive unfolding in cases **(F)** and **(M3)** of Definition 6.2 would force a similar unfolding of A in the inductive step, thus not necessarily decreasing the size of A.

Instead, we consider finite approximations as in [3], where we get rid of recursion by unfolding a finite number of times and then replacing all remaining occurrences of recursion by □. We call a specification *non-recursive* if it does not contain any occurrences of specifications of the form $\mu(X)B$.

Definition 6.7 (Approximations). *For each* A *and* $k \in \mathbb{N}$, *we define* $A|^k B$.

Proof.

$$\frac{\begin{array}{lll} & & \Gamma, y_j \vdash T_j{}^{j=1\ldots m+q} \\ \Gamma \quad ; \Delta \vdash A_i{}^{i=1\ldots n+p} & \Gamma \quad ; \Delta \vdash A_i <: A_i'{}^{i=1\ldots n} & \Gamma, y_j \vdash T_j'{}^{j=1\ldots m} \\ \Gamma, y_j ; \Delta \vdash B_j{}^{j=m+1\ldots m+q} & \Gamma, y_j ; \Delta \vdash B_j <: B_j'{}^{j=1\ldots m} & \vdash_{\mathsf{fo}} T_j \to T_j'{}^{j=1\ldots m} \end{array}}{\Gamma ; \Delta \vdash [\mathsf{f}_i \colon A_i{}^{i=1\ldots n+p},\ \mathsf{m}_j \colon \varsigma(y_j) B_j :: T_j{}^{j=1\ldots m+q}] <: [\mathsf{f}_i \colon A_i'{}^{i=1\ldots n},\ \mathsf{m}_j \colon \varsigma(y_j) B_j' :: T_j'{}^{j=1\ldots m}]}$$

Table 6. The generalised object subspecification rule

invariance in field specifications. For example, if $A \equiv [\mathsf{f}_1 : X, \mathsf{f}_2 : Bool]$ then

$$\begin{aligned} \mu(X)\mu(Y)A|^2 &= [\mathsf{f}_1 : \mu(X)\mu(Y)A, \mathsf{f}_2 : Bool]|^2 \\ &= [\mathsf{f}_1 : \mu(X)\mu(Y)A|^1, \mathsf{f}_2 : Bool|^1] \\ &= [\mathsf{f}_1 : [\mathsf{f}_1 : \mu(X)\mu(Y)A, \mathsf{f}_2 : Bool]|^1, \mathsf{f}_2 : Bool] \\ &= [\mathsf{f}_1 : [\mathsf{f}_1 : \mu(X)\mu(Y)A|^0, \mathsf{f}_2 : Bool|^0], \mathsf{f}_2 : Bool] \\ &= [\mathsf{f}_1 : [\mathsf{f}_1 : \Box, \mathsf{f}_2 : \Box], \mathsf{f}_2 : Bool] \end{aligned}$$

By inspection of the rules, $\vdash \mu(X)\mu(Y)A <: \mu(X)\mu(Y)A|^2$ requires to show

$$\Gamma ; \Delta \vdash [\mathsf{f}_1 : [\mathsf{f}_1 : \mu(X)\mu(Y)A, \mathsf{f}_2 : Bool], \mathsf{f}_2 : Bool] <: [\mathsf{f}_1 : [\mathsf{f}_1 : \Box, \mathsf{f}_2 : \Box], \mathsf{f}_2 : Bool]$$

for appropriate Γ and Δ. But subspecifications of object specifications can only be derived for equal components f_1 with the rules of Sect. 3.

Therefore we consider the more generous subspecification relation that also

- A is $\mu(X)B$. Then, by induction hypothesis,

$$\Gamma;\Delta \vdash B[A/X] <:^{\square} B[A/X]|^k$$

By definition of approximations, the latter equals $A|^k$. Moreover,

$$\Gamma;\Delta \vdash A <:^{\square} B[A/X]$$

by the (unfold) rule, and transitivity then yields $\Gamma;\Delta \vdash A <:^{\square} A|^k$.

- A is $[\mathsf{f}_i{:}\, A_i{}^{i=1\ldots n},\ \mathsf{m}_j{:}\,\varsigma(y_j)B_j{::}T_j{}^{j=1\ldots m}]$. By definition,

$$A|^k = [\mathsf{f}_i : A_i|^{k-1}, \mathsf{m}_j : B_j|^{k-1}]$$

By induction hypothesis we obtain that $\Gamma;\Delta \vdash A_i <:^{\square} A_i|^{k-1}$ and that $\Gamma, y_j;\Delta \vdash B_j <:^{\square} B_j|^{k-1}$ which entails

$$\Gamma;\Delta \vdash [\mathsf{f}_i : A_i, \mathsf{m}_j : B_j :: T_j] <:^{\square} [\mathsf{f}_i : A_i, \mathsf{m}_j : B_j :: T_j]|^k$$

by the (modified) subspecification rule, as required.

$\square$

Soundness of Subspecification

Soundness of subspecification is easily established:

Lemma 6.9 (Soundness of $<:^{\square}$). *If* $\Gamma;\Delta \vdash A <:^{\square} B$, $\rho \in \mathsf{Env}$ *and* $\eta \models \Delta$ *then* $[\![\Gamma;\Delta \vdash A]\!]\rho\eta \subseteq [\![\Gamma;\Delta \vdash B]\!]\rho\eta$.

Proof. By induction on the derivation of $\Gamma;\Delta \vdash A <:^{\square} B$.

- (Reflexivity) and (Transitivity) are immediate, as is (Top).
- (Fold) and (Unfold) follow from the fact that the denotation of $\mu(X)A$ is indeed a fixed point,

$$\begin{aligned}
[\![\Gamma;\Delta \vdash \mu(X)A]\!]\rho\eta &= \mathsf{gfp}(\lambda\chi.[\![\Gamma;\Delta, X \vdash A]\!]\rho\eta[X=\chi]) && \text{by definition}\\
&= [\![\Gamma;\Delta, X \vdash A]\!]\rho(\eta[X = [\![\Gamma;\Delta \vdash \mu(X)A]\!]\rho\eta]) && \text{fixed point}\\
&= [\![\Gamma;\Delta \vdash A[\mu(X)A/X]]\!]\rho\eta && \text{Lemma 6.6}
\end{aligned}$$

- For the case of (Object), we must have

$$A = [\mathsf{f}_i : A_i{}^{i=1\ldots n+p}, \mathsf{m}_j : \varsigma(y_j)B_j :: T_j{}^{j=1\ldots m+q}]$$

and

$$B = [\mathsf{f}_i : A_i'{}^{i=1\ldots n}, \mathsf{m}_j : \varsigma(y_j)B_j' :: T_j{}^{j=1\ldots m}]$$

such that $\Gamma;\Delta \vdash A_i <:^{\square} A_i'$ and $\Gamma, y_j;\Delta \vdash B_j <:^{\square} B_j'$ and $\vdash_{\mathsf{fo}} T_j \to T_j'$. By induction hypothesis,

$$[\![\Gamma;\Delta \vdash A_i]\!]\rho\eta \subseteq [\![\Gamma;\Delta \vdash A_i']\!]\rho\eta$$

and

$$[\![\Gamma, y_j; \Delta \vdash B_j]\!](\rho[y_j := l])\eta \subseteq [\![\Gamma, y_j; \Delta \vdash B'_j]\!](\rho[y_j := l])\eta$$

for all $1 \leq i \leq n$, $1 \leq j \leq m$ and $l \in \mathsf{Loc}$. Moreover, by soundness of $\vdash_{\mathsf{fo}}$ we know

$$[\![[\Gamma], y_j \vdash T_j]\!](\rho[y_j := \text{‘}$$

Proof. By induction on the lexicographic order on l and the number of μ in head position.

- $l = 0$. Clearly $\Gamma; \Delta \vdash A[B/X]|^0 <: \top = A[B|^k/X]|^0$.
- $l > 0$. We consider possible cases for A.
  - A is X. Then $\Gamma; \Delta \vdash A[B/X]|^l = B|^l <: B|^k|^l = A[B|^k/X]|^l$.
  - A is $\top$, *Bool* or $Y \neq X$. Then $\Gamma; \Delta \vdash A[B/X]|^l = A|^l <: A|^l = A[B|^k/X]|^l$.
  - A is $[\mathsf{f}_i : A_i{}^{i=1\ldots n}, \mathsf{m}_j : \varsigma(y_j)B_j :: T_j{}^{j=1\ldots m}]$. Then, by induction hypothesis,

    $$\Gamma; \Delta \vdash A_i[B/X]|^{l-1} <: A_i[B|^k/X]|^{l-1}$$

    and

    $$\Gamma, y_j{:}A; \Delta \vdash B_j[B/X]|^{l-1} <: B_j[B|^k/X]|^{l-1}$$

    for all $1 \leq i \leq n$ and $1 \leq j \leq m$. Hence,

    $$\Gamma; \Delta \vdash A[B/X]|^l <: [\mathsf{f}_i : A_i[B|^k/X], \mathsf{m}_j : B_j[B|^k/X]]|^l = A[B|^k/X]|^l$$
  - A is $\mu(Y)C$, without loss of generality Y not free in B. Then by induction hypothesis we find $\Gamma; \Delta \vdash C[A/Y][B/X]|^l <: C[A/Y][B|^k/X]|^l$. Using properties of syntactic substitutions, we calculate

    $$\begin{aligned} A[B/X]|^l &= \mu(Y)(C[B/X])|^l \\ &= C[B/X][(\mu(Y)(C[B/X]))/Y]|^l \\ &= C[B/X][(A[B/X])/Y]|^l \\ &= C[A/Y][B/X]|^l \end{aligned}$$

    and analogously $C[A/Y][B|^k/X]|^l = A[B|^k/X]|^l$, which entails the result.

$\square$

Lemma 6.11 (Approximation of Specifications). *For all* $\Gamma; \Delta \vdash A$, $\rho \in \mathsf{Env}$ *and environments* $\eta \models \Delta$,

$$[\![\Gamma; \Delta \vdash A]\!]\rho\eta = \bigcap_{k \in \mathsf{N}} [\![\Gamma; \Delta \vdash A|^k]\!]\rho\eta$$

Proof. By (19), all that remains to show is $[\![\Gamma; \Delta \vdash A]\!]\rho\eta \supseteq \bigcap_{k \in \mathsf{N}} [\![\Gamma; \Delta \vdash A|^k]\!]\rho\eta$. We proceed by induction on the lexicographic order on pairs (M, A) where M is an upper bound on the number of μ-binders in A. For the base case, $M = 0$, by Lemma 6.8(3) there exists $n \in \mathsf{N}$ such that for all $k \geq n$, $A|^k = A$, and so in fact

$$[\![\Gamma; \Delta \vdash A]\!]\rho\eta = [\![\Gamma; \Delta \vdash A|^n]\!]\rho\eta \supseteq \bigcap_{k \in \mathsf{N}} [\![\Gamma; \Delta \vdash A|^k]\!]\rho\eta$$

Now suppose that A contains at most $M + 1$ μ binders. We consider cases for A.

6.4 Soundness

After the technical development in the preceding subsection we can now prove (18). From this result the soundness proof of the logic extended with recursive specifications then follows, along the lines of the proof presented in Section 5 for finite specifications.

Lemma 6.12. *For all* σ, Σ, l *and* □ A*, if* σ □ $[\![\Sigma]\!]$ *and* □ $\Sigma.l <:^{□} A$ *then* (l, σ) □ $[\![A]\!]$.

Proof. The proof proceeds by considering finite specifications first. This can be proved by induction on A

arbitrary extensions $\Sigma' \supseteq \Sigma$. This will account for the (specifications of) objects allocated between definition time and call time.

Clearly, not every predicate on stores is preserved. As we lack a semantic characterisation of those specifications that are syntactically definable (as Σ), specification syntax appears in the definition of $\sigma \in [\![\Sigma]\!]$ (Def. 4.7). More annoyingly, field update requires subspecifications to be invariant in the field components, otherwise even type soundness is invalidated. We do not know how to express this property of object specifications semantically (on the level of predicates) and need to use the inductively defined subspecification relation instead.

The proof of Theorem 4.8, establishing the existence of store predicates, provides an explanation why transition relations of the Abadi-Leino logic express properties of the flat part of stores only: Semantically, a (sufficient) condition is that transition relations are upwards and downwards closed in their first and second store argument, respectively.

Abadi and Leino's logic is peculiar in that verified programs need to preserve store specifications. Put differently, only properties which are in fact preserved can be expressed in the logic. In particular, specifications of field values are limited such that properties like e.g. $\mathsf{self.hd} \leq \mathsf{self.tail.hd}$, stating that a list is sorted, cannot be expressed. In future work we thus plan to investigate how a logic can be set up where

- methods are specified by pre-/post-conditions that explicitly state invariance properties during execution of the method code.
- methods can be specified by pre-/post-conditions that can refer to other methods. This is important for simulating methods that act like higher-order functions (e.g. the map function for lists).
- methods can have additional parameters.
- method update is allowed. In the setting of Abadi and Leino this would require that the new method body satisfies the old specification (in order to establish invariance). More useful would be a "behavioural" update where result and transition specifications of the overriding method are subspecifications of the original method.

The results established in this paper pave the way for the above line of research.

Acknowledgement We wish to thank Thomas Streicher for discussions and comments.

References

[1] M. Abadi and L. Cardelli. *A Theory of Objects*. Springer, New York, 1996.

[2] M. Abadi and K. R. M. Leino. A logic of object-oriented programs. In N. Dershowitz, editor, *Verification: Theory and Practice*, pages 11–41. Springer, 2004.

[3] R. M. Amadio and L. Cardelli. Subtyping recursive types. *ACM Transactions on Programming Languages and Systems*, 15(4):575–631, 1993.

[4] K. R. Apt. Ten years of Hoare's logic: A survey — part I. *ACM Transactions on Programming Languages and Systems*, 3(4):431–483, Oct. 1981.

[5] F. S. de Boer. A WP-calculus for OO. In W. Thomas, editor, *Foundations of Software Science and Computation Structures*, volume 1578 of *Lecture Notes in Computer Science*, pages 135–149, 1999.

[6] U. Hensel, M. Huisman, B. Jacobs, and H. Tews. Reasoning about classes in object-oriented languages: Logical models and tools. In C. Hankin, editor, *Programming Languages and Systems—ESOP'98, 7th European Symposium on Programming*, volume 1381 of *Lecture Notes in Computer Science*, pages 105–121, Mar. 1998.

[7] C. A. R. Hoare. An Axiomatic Basis of Computer Programming. *Communications of the ACM*, 12:576–580, 1969.

[8] K. R. M. Leino. Recursive object types in a logic of object-oriented programs. In C. Hankin, editor, *7th European Symposium on Programming*, volume 1381 of *Lecture Notes in Computer Science*, pages 170–184, Mar. 1998.