

Kripke semantics for full ground references (work in progress)

Levy, Paul

License:

None: All rights reserved

Document Version

Peer reviewed version

Citation for published version (Harvard):

Levy, P 2016, 'Kripke semantics for full ground references (work in progress)', LOLA 2016 - Syntax and Semantics of Low-Level Languages, New York, United States, 10/07/16 - 10/07/16. <<http://lola.cse.buffalo.edu/>>

[Link to publication on Research at Birmingham portal](#)

Publisher Rights Statement:

Checked for eligibility: 05/09/2018

General rights

Unless a licence is specified above, all rights (including copyright and moral rights) in this document are retained by the authors and/or the copyright holders. The express permission of the copyright holder must be obtained for any use of this material other than for purposes permitted by law.

- Users may freely distribute the URL that is used to identify this publication.
- Users may download and/or print one copy of the publication from the University of Birmingham research portal for the purpose of private study or non-commercial research.
- User may use extracts from the document in line with the concept of 'fair dealing' under the Copyright, Designs and Patents Act 1988 (?)
- Users may not further distribute the material nor use it for the purposes of commercial gain.

Where a licence is displayed above, please note the terms and conditions of the licence govern your use of this document.

When citing, please reference the published version.

Take down policy

While the University of Birmingham exercises care and attention in making items available there are rare occasions when an item has been uploaded in error or has been deemed to be commercially or otherwise sensitive.

If you believe that this is the case for this document, please contact UBIRA@lists.bham.ac.uk providing details and we will remove access to the work immediately and investigate.

Kripke semantics for full ground references (work in progress)

Paul Blain Levy, University of Birmingham

“Full ground references” means references to integers and to other references, but not functions or thunks. Game semantics for full ground references was given in [1]. This work aims to give Kripke semantics.

Language We fix a set \mathcal{S} of *sorts*. We extend the call-by-push-value types with a reference type Ref_s for each sort s . We then define

$$\text{full ground types } D ::= 0 \mid D + D \mid \sum_{i \in \mathbb{N}} D_i \mid 1 \mid D \times D \mid \text{Ref}_s \ (s \in \mathcal{S})$$

and fix, for each sort s , a full ground type D_s , intended to be the type of values stored in a reference cell of sort s .

A *world* is a finite set over \mathcal{S} . (Alternatively: a finite sequence of sorts.) The judgements are $w, \Gamma \vdash^v V : A$ for values and $w, \Gamma \vdash^c M : \underline{B}$ for computations, where w is a world and Γ a typing context (finite set of identifiers with an associated value type). Term syntax and operational semantics is defined as usual. Evaluation terminates because of the restriction to full ground references.

Denotational semantics A value type A will denote a functor from the category Inj of worlds and injections to **Set**. Intuitively, $\llbracket A \rrbracket w$ is a semantic domain for closed values in world w , and such values can be renamed along an injection $w \rightarrow w'$. In particular $\llbracket \text{Ref}_s \rrbracket w$ is the set of s -sorted cells in w .

A w -store associates to each cell l in w a value $w \vdash^v s_l : D_{\text{sort}(l)}$. The semantic domain for these is

$$Sw \stackrel{\text{def}}{=} \prod_{l \in w} \llbracket D_{\text{sort}(l)} \rrbracket w$$

Although for ground references S is a functor $\text{Inj}^{\text{op}} \rightarrow \mathbf{Set}$, that is not so for full ground references.

Computation types have more subtle semantics. To understand it, say that a w -store s associates to each cell l in w a value $w \vdash^v s_l : D_{\text{sort}(l)}$. An *SC-configuration* $\Gamma \vdash^{\text{sc}} x, s, M : \underline{B}$ consists of

- a world x —we think of cells in x as local/private, whereas those in w are global/public
- a $w + x$ -store s
- and a computation $w + x, \Gamma \vdash^c M : \underline{B}$.

These arise in the operational semantics of a language that has both w -many global cells and generation of local cells.

We want $\llbracket \underline{B} \rrbracket w$ to be a semantic domain for closed SC-configurations in world w . What category should $\llbracket \underline{B} \rrbracket$ be a functor from?

Let’s start with the ground ref setting. An *initialization* $(i, p) : w \rightarrow w'$ consists of

- an injection $i : w \rightarrow w'$ —we write $\text{new}(i)$ for the cells in w' not in the range of i
- and an element $p \in \prod_{l \in \text{new}(i)} \llbracket D_{\text{sort}(l)} \rrbracket$

These form a category Init . A *partial initialization* is defined similarly, except that i is a partial injection. The latter form a category PInit that contains both Inj^{op} and Init , and indeed is freely generated by these subcategories modulo two equations.

A partial initialization $i : w \rightarrow w'$ converts an SC-configuration in world w to one in world w' , by

- hiding the cells in w that are not in the domain of i
- renaming each cell in the domain of i to one in the range
- creating each cell $l \in \text{new}(i)$ with value p_l .

So a computation type \underline{B} should denote a functor $\mathbf{PInit} \rightarrow \mathbf{Set}$.

Turning to full ground references, an initialization $(i, p) : w \rightarrow w'$ consists of

- an injection $i : w \rightarrow w'$
- and an element $p \in \prod_{l \in \text{new}(i)} \llbracket D_{\text{sort}(l)} \rrbracket(w + \text{new}(i))$

These form a category \mathbf{Init} , and S is a functor $\mathbf{Init} \rightarrow \mathbf{Set}$. But partial initializations are more subtle. Let's first say that a *stateful value* $\Gamma \vdash^{\text{sv}} x, s, V : A$ consists of

- a world x —again, cells in x are local/private, whilst those in x are global/public
- for each local cell l in x , a value $w + x \vdash^v s_l : D_{\text{sort}(l)}$
- and a value $w + x \vdash^v V : A$.

For any functor $A : \mathbf{Inj} \rightarrow \mathbf{Set}$, define

$$(\Psi A) w \stackrel{\text{def}}{=} \int^{x \in \mathbf{Init}} \prod_{l \in x} \llbracket D_{\text{sort}(l)} \rrbracket(w + x) \times \llbracket A \rrbracket(w + x)$$

so that $(\Psi \llbracket A \rrbracket)w$ is a semantic domain for closed stateful values of type A in world w . If A is a constant functor, then $(\Psi A)w \cong Aw$. A *partial initialization* $(i, p) : w \rightarrow w'$ consists of

- a partial injection $i : w \rightarrow w'$
- and an element $p \in (\Psi \prod_{l \in \text{new}(i)} \llbracket D_{\text{sort}(l)} \rrbracket)(w + \text{new}(i))$.

These form a category \mathbf{PInit} that contains both \mathbf{Inj}^{op} and \mathbf{Init} and indeed is freely generated by these subcategories modulo two equations. A computation type \underline{B} should denote a functor $\mathbf{PInit} \rightarrow \mathbf{Set}$. Also, if $A : \mathbf{Inj} \rightarrow \mathbf{Set}$ then $\Psi A : \mathbf{PInit}^{\text{op}} \rightarrow \mathbf{Set}$.

Semantics of judgements:

- A value $w, \Gamma \vdash^v V : A$ denotes a family of functions $\llbracket \Gamma \rrbracket(w + x) \rightarrow \llbracket A \rrbracket(w + x)$, natural in $x \in \mathbf{Inj}$.
- A computation $w, \Gamma \vdash^c M : \underline{B}$ denotes a family of functions $S(w + x) \times \llbracket \Gamma \rrbracket(w + \bar{x}) \rightarrow \llbracket \underline{B} \rrbracket(w + x)$, natural in $x \in \mathbf{Init}$. Here overline represents the forgetful functor $\mathbf{Init} \rightarrow \mathbf{Inj}$.

Semantics of types:

$$\begin{aligned} \llbracket FA \rrbracket w &\stackrel{\text{def}}{=} \int^{x \in \mathbf{Init}} S(w + x) \times \llbracket A \rrbracket(w + \bar{x}) \\ \llbracket U \underline{B} \rrbracket w &\stackrel{\text{def}}{=} \int_{x \in \mathbf{Init}} S(w + x) \rightarrow \llbracket \underline{B} \rrbracket(w + x) \\ \llbracket \prod_{i \in I} \underline{B}_i \rrbracket w &\stackrel{\text{def}}{=} \prod_{i \in I} \llbracket \underline{B}_i \rrbracket w \\ \llbracket A \rightarrow \underline{B} \rrbracket w &\stackrel{\text{def}}{=} \int_{x \in \mathbf{Init}} \llbracket A \rrbracket(w + \bar{x}) \rightarrow \llbracket \underline{B} \rrbracket(w + x) \\ \llbracket U \prod_{i \in I} (A_i \rightarrow \underline{B}_i) \rrbracket w &\cong \int_{x \in \mathbf{Init}} S(w + x) \rightarrow \prod_{i \in I} (\llbracket A_i \rrbracket(w + \bar{x}) \rightarrow \llbracket \underline{B}_i \rrbracket(w + x)) \end{aligned}$$

References

- [1] Andrzej S. Murawski and Nikos Tzevelekos. Algorithmic games for full ground references. In Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer, editors, *ICALP (2)*, volume 7392 of *Lecture Notes in Computer Science*, pages 312–324. Springer, 2012.