

Title

NIKI VAZOU, IMDEA Software Institute, Spain

This is the abstract

Additional Key Words and Phrases: refinement types, function extensionality, decidable verification, SMT

ACM Reference Format:

Niki Vazou. 2018. Title. *J. ACM* 37, 4, Article 111 (August 2018), 2 pages. <https://doi.org/10.1145/1122445.1122456>

1 INTRODUCTION

The contributions of this work are the following:

- We present why the natural axiom of function extensionality is unsound when encoded as a refinement type.
- We formalize a decidable and complete algorithm that reason about function extensionality and data types (§ 2).
- We implemented the algorithm on the pipeline of Liquid Haskell and evaluated on various benchmarks.

2 FORMALIZATION

We use $\text{PLE}(\Psi, \Phi, p)$ [1] that checks if the conjunction of Φ implies p . But, before we call PLE, we expand function extensionality using the below rules.

2.1 Interpretation of Type Constructors

Let's assume we want to prove

```
first :: (a → b) → (a, c) → (b, c)
first f (x, y) = (f x, y)
```

```
thm :: f:(a → b) → g:(a → b) → {first f == first g => f == g}
thm _ _ = ()
```

Then you need to prove

```
x:a, f:a→b, g:a→b; |- f x == g x
```

```
-----
x:a, f:a→b, g:a→b; first f == first g |- f x == g x
```

```
-----
f:a→b, g:a→b; first f == first g |- f == g
```

which gets stuck, since no pair exists in the binding environment.

Author's address: Niki Vazou, niki.vazou@imdea.org, IMDEA Software Institute, Madrid, Spain.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2018 Association for Computing Machinery.

0004-5411/2018/8-ART111 \$15.00

<https://doi.org/10.1145/1122445.1122456>

Terms	$t ::= \text{true} \mid \text{false} \mid 0, \pm 1, \pm 2, \dots$	Constants
	$\mid x$	Variables
	$\mid D \bar{t} \mid f \bar{t}$	Applications
Predicates	$p ::= t$	Terms
	$\mid p \Rightarrow p \mid p \wedge p$	Booleans
	$\mid p =_{\tau} p$	Equality
Sorts	$\tau ::= \text{Int} \mid \text{Bool}$	Constants
	$\mid \alpha$	Variables
	$\mid \tau \rightarrow \tau$	Functions
	$\mid T \bar{\tau}$	Type Constructors
Functions	$F ::= \lambda \bar{x}. \langle \overline{p \Rightarrow p} \rangle$	
Logical Environment	$\Phi ::= \emptyset \mid p, \Phi$	
Function Environment	$\Psi ::= \emptyset \mid f:F, \Psi$	
Binder Environment	$\Sigma ::= \emptyset \mid x:\tau, \Sigma$	

Fig. 1. Syntax

$$\begin{array}{c}
\frac{\text{PLE}(\Psi, \Phi, t)}{\Psi; \Sigma; \Phi \vdash t} \text{TERM} \quad \frac{\Psi; \Sigma; p_1, \Phi \vdash p_2}{\Psi; \Sigma; \Phi \vdash p_1 \Rightarrow p_2} \Rightarrow_R \quad \frac{\Psi; \Sigma; \Phi \vdash p_1 \quad \Psi; \Sigma; p_2, \Phi \vdash p}{\Psi; \Sigma; p_1 \Rightarrow p_2, \Phi \vdash p} \Rightarrow_{-L} \quad \Psi; \Sigma; \Phi \vdash p \\
\\
\frac{\Psi; \Sigma; \Phi \vdash p_2 \quad \Psi; \Sigma; \Phi \vdash p_2}{\Psi; \Sigma; \Phi \vdash p_1 \wedge p_2} \wedge - R \quad \frac{\Psi; \Sigma; p_1, p_2, \Phi \vdash p}{\Psi; \Sigma; p_1 \wedge p_2, \Phi \vdash p} \wedge - L \\
\\
\frac{\Psi; x:\tau_x \Sigma; \Phi \vdash t_L x =_{\tau} t_R x}{\Psi; \Sigma; \Phi \vdash t_L =_{\tau_x \rightarrow \tau} t_R} =\text{RFUN} \quad \frac{\text{PLE}(\Psi, \Phi, t_L =_{\tau} t_R) \quad \tau \neq \tau_1 \rightarrow \tau_2}{\Psi; \Sigma; \Phi \vdash t_L =_{\tau} t_R} =\text{RNotFUN} \\
\\
\frac{ts = \{x \mid x:\tau_x \in \Sigma\} \quad \Psi; \Sigma; \bigwedge_{t \in ts} t_L t =_{\tau} t_R t, \Phi \vdash p}{\Psi; \Sigma; t_L =_{\tau_x \rightarrow \tau} t_R, \Phi \vdash p} =\text{LFUN}
\end{array}$$

Fig. 2. Inference rules for Extensionality

REFERENCES

- [1] Niki Vazou, Anish Tondwalkar, Vikraman Choudhury, Ryan G. Scott, Ryan R. Newton, Philip Wadler, and Ranjit Jhala. 2018. Refinement reflection: complete verification with SMT. *PACMPL* 2, POPL (2018), 53:1–53:31. <https://doi.org/10.1145/3158141>