The Name of the Title Is Hope

BEN TROVATO* and G.K.M. TOBIN*, Institute for Clarity in Documentation, USA
LARS THØRVÄLD, The Thørväld Group, Iceland
VALERIE BÉRANGER, Inria Paris-Rocquencourt, France
APARNA PATEL, Rajiv Gandhi University, India
HUIFEN CHAN, Tsinghua University, China
CHARLES PALMER, Palmer Research Laboratories, USA
JOHN SMITH, The Thørväld Group, Iceland
JULIUS P. KUMQUAT, The Kumquat Consortium, USA

Fig. 1. Seattle Mariners at Spring Training, 2010.

Write useful abstract

CCS Concepts: • **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

Additional Key Words and Phrases: datasets, neural networks, gaze detection, text tagging

ACM Reference Format:
Ben Trovato, G.K.M. Tobin, Lars Thørväld, Valerie Béranger, Aparna Patel, Huifen Chan, Charles Palmer, John Smith, and Julius P. Kumquat. 2022. The Name of the Title Is Hope. In . ACM, New York, NY, USA, 2 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

1.1 Context

Using Refinement Types allows us to define contracts for function types that define pre-conditions on the function arguments and post-conditions on it's return values. These contracts can then be checked statically at compile time. One implementation of this is LiquidHaskell. When combining

*Both authors contributed equally to this research.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

function calls, it can be verified that the post-condition of the inner function will also fulfill all pre-conditions of the outher function. Listing 1 shows an example of this.

Listing 1. Contracts in LiquidHaskell

```
-- Refinement type that describes a non-empty list
{-@ type NonEmpty a = {v:[a] | len v > 0 } @-}

-- Define a head function that only works on non-emtpy lists
{-@ head :: NonEmpty Int -> Int @-}
head :: [Int] -> Int
head (x:_) = x

{-@ ensureNonEmpty :: [Int] -> NonEmpty Int -}
ensureNonEmpty :: [Int] -> [Int]
ensureNonEmpty [] = [0]
ensureNonEmpty x = x

i1 :: Int
i1 = head []                        -- Not allowed
i1 = head (ensureNonEmpty [])       -- Allowed
```

Using contracts, the programmer can clearly define the expectations and assurances that are made to the functions they write. Ensuring their fulfillment before code execution can have a great impact on a programs correctness and stability.

1.2 Motivation

1.3 Problem Definition

1.4 Impact

1.5 Approach

1.6 Contributions

2 RELATED WORK

3 BACKGROUND

4 IMPLEMENTATION

5 EVALUATION

6 CONCLUSION

7 ACKNOWLEDGMENTS

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009