

# Promises are meant to be broken

## Strictness Analysis for R

Author 1<sup>\*</sup>

Position1

Department1

Institution1

Street1 Address1

City1, State1 Post-Code1, Country1

first1.last1@inst1.edu

Author 2<sup>†</sup>

Position2a

Department2a

Institution2a

Street2a Address2a

City2a, State2a Post-Code2a, Country2a

first2.last2@inst2a.com

Position2b

Department2b

Institution2b

Street3b Address2b

City2b, State2b Post-Code2b, Country2b

first2.last2@inst2b.org

### Abstract

R evaluates function arguments lazily. Arguments are stored in “promises” which are “forced” when the corresponding parameter is used in an evaluation context. A promise also contains a reference to the environment in which the argument has to be evaluated and a slot to cache the result of evaluation. This mechanism can potentially improve speed of R programs by preventing the evaluation of arguments that are not used. However, benchmarks obtained by instrumenting the R interpreter indicate that lazy evaluation is futile for majority of R programs and detrimental for their performance. R has a functional core and even most the basic control structures are implemented as functions. This results in the creation of a huge number of promises which get forced immediately but continue storing the unevaluated arguments until they are garbage collected. Furthermore, accessing the argument’s value requires the promise to look up the cache which introduces a level of indirection, increasing the possibility of cache misses.

In this paper, we present a static analysis technique on an intermediate representation of R code that identifies the promises that are forced and the order in which they are forced. This information can then be used by an optimizer to prevent the creation of promises and evaluate arguments in the correct order, ensuring unchanged program behavior. We validate the results of our static analysis from dynamic runs on a number of widely used R libraries.

---

<sup>\*</sup>with author1 note

<sup>†</sup>with author2 note

---

with paper note.

PL’17, New York, NY, USA

2017. 978-x-xxxx-xxxx-x/YY/MM...\$15.00

DOI: 10.1145/nnnnnnnn.nnnnnnn

**CCS Concepts** •Theory of computation → Program analysis; •Software and its engineering → Procedures, functions and subroutines; Multiparadigm languages;

**Keywords** R, Strictness analysis, Control-Flow analysis, Static analysis, Program analysis, Performance optimization

### ACM Reference format:

Author 1 and Author 2. 2017. Promises are meant to be broken. In *Proceedings of ACM SIGPLAN Conference on Programming Languages, New York, NY, USA, January 01–03, 2017 (PL’17)*, 2 pages. DOI: 10.1145/nnnnnnnn.nnnnnnn

## 1 Introduction

Text of paper ...

## 2 Motivating Example

### 2.1 Background on R/Strictness

### 2.2 Example/IR representation

## 3 Algorithm

## 4 Implementation

## 5 Experiments

## 6 Related Work

## 7 Conclusions

## Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. nnnnnnnn and Grant No. mmmmmmm. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

## **A Appendix**

Text of appendix ...