

Dead Code Elimination

Robbie Hammond
Case Western Reserve University,
School of Engineering
Cleveland, Ohio, USA
reh161@case.edu

Saketh Dendi
Case Western Reserve University,
School of Engineering
Cleveland, Ohio, USA
ssd74@case.edu

Milo Cassarino
Case Western Reserve University,
School of Engineering
Cleveland, Ohio, USA
mgc73@case.edu

ABSTRACT

Insert our abstract here.

KEYWORDS

compiler optimization, dead code

ACM Reference Format:

Robbie Hammond, Saketh Dendi, and Milo Cassarino. 2018. Dead Code Elimination. In *Proceedings of ACM Conference (Conference'17)*. ACM, New York, NY, USA, 2 pages. <https://doi.org/XXXXXXX.XXXXXXX>

1 INTRODUCTION

Dead code, when unaddressed, can be a major problem for a piece of software. If not eliminated by the compiler, dead code can make a program larger, especially if there is a substantial amount of it within a codebase. In addition, dead code can make software arbitrarily slower, with computational resources being devoted to declaring unused variables and empty functions. For these reasons, it is imperative that compilers, at a minimum, eliminate chunks of code that are clearly unneeded.

Dead code, as defined in this paper, consists of two related, but distinct types of code. Firstly, code can be considered dead if it is never actually executed at runtime. For example, code inside of the 'else' portion of an if-else block where the condition is always true would be dead. Secondly, a portion of code can be considered dead if the computation performed on those lines are never used anywhere else. For example, a function that is never written to or read from can be considered dead.

Our project focuses on adding basic dead code elimination strategies into the compiler created in the fourth class project. Our goal is to give the compiler support for eliminating unreachable code in if branches, for loops, and while loops. In addition, the compiler should be able to remove unused variables.

2 IMPLEMENTATION

In this section, we will outline how we implemented the aforementioned code removal functionality.

2.1 If Statements

We checked the if condition, if the compiled R value was 0, remove the if. If the compiled R value was 1, remove the else. Make this sound more complicated and smart.

2.2 While Loops

We checked the condition, if the compiled R value was 0, don't generate a loop at all. If the compiled R value was 1, Generate the loop, but heavily shorten the code after it. Make this sound more complicated and smart.

2.3 For Loops

Pretty much the same as the while loop.

2.4 Useless Variables

Hopefully we can get this implemented and talk about it. Worst case, we can just talk about things we considered and stuff we tried.

3 TESTING METHODOLOGY

I'm thinking our main metric should be file size in bytes, since this is the main purpose of dead code elimination. We can also measure speed as well, but make it clear that it isn't as significant.

4 RESULTS

We'll use tables, charts, and other stuff to make our basic testing look fancy. The better it looks, the more it'll look like we tried.

5 CONCLUSION

We can talk about our results and talk about how the simple examples could extend to bigger examples and that big code bases need dead code elim. Blah blah blah.

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

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, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Conference'17, July 2017, Washington, DC, USA

© 2018 Association for Computing Machinery.

ACM ISBN 978-1-4503-XXXX-X/18/06...\$15.00

<https://doi.org/XXXXXXX.XXXXXXX>

Table 1: Some Typical Commands

Command	A Number	Comments
<code>\author</code>	100	Author
<code>\table</code>	300	For tables
<code>\table*</code>	400	For wider tables