Instructions for NAACL-HLT 2021 Proceedings

First Author
Affiliation / Address line 1
Affiliation / Address line 2
Affiliation / Address line 3
email@domain

Second Author
Affiliation / Address line 1
Affiliation / Address line 2
Affiliation / Address line 3
email@domain

Abstract

Debloating Abstract

1 Introduction

Introduction

2 Tools

Explanation of all tools

2.1 CHISEL

The input to the **CHISEL** tool is a C/C++ source file (test.c) which is to be debloated. Along with the source file , we give a specialized script which contains the high level specification of the desired output. The CHISEL tool then generates binary of the source code(test.c.chisel.c). Both the source and debloated program are then compiled by g++ or any other compiler and the binaries generated by the compiler are given to the ROP gadgets or Gadget Set Analyser(GSA) . The ROP gadget outputs the count of the gadgets before and after debloating respectively.

CHISEL works on the Delta Debugging algorithm.The delta debugging algorithm optimizes the global functions calls,etc. first and then further approach local level reduction for the optimization. The process is continued till 1-minimal of a program is reached. Delta debugging ensures that there is no scope for further debloating.

Markov Decision Process for delta debugging is deployed to build a statistical model to get 1-minimal solution with lesser number of iterations than delta debugging alone.

2.2 INST2VEC

Inst2vec is the technique of processing the source code into the features vector for modeling the Reinforcement Learning. It follows an approach similar to skip-gram model in Natural Language Processing of pre-defined context size. The source code is first compiled into LLVM IR code. The LLVM IR is in the form of a static single assignment.As LLVM IR is independent of machine or hardware architecture and programming language , it becomes easy to train the program embeddings.The approach is similar to word2vec model.

With the LLVM IR , the contextual flow graph (XFG)is created. XFG takes into account both the data flow and control flow of the program. With these XFGs generated from LLVM IR , the consecutive statement pairs are made of pre-defined context size . These statement pairs are then checked for duplication. The XFGs pairs are made by constructing a dual graph with statement as nodes and removing duplicate edges. The process is then followed by removing statements of negligible presence. We get an inst2vec after subsampling of frequent pairs which can be optimized and trained. XFGs ensures that the semantics of statements are preserved.

2.3 TRIMMER

The source code is converted to LLVM IR and given as an input to the trimmer tool along with the manifest file consisting of user defined configuration. The TRIMMER first performs input specialization with respect to the user defined configuration . The second part is loop unrolling . The loop unrolling becomes important step to make interprocedural constant propagation easier. Constant Propagation is the final stage in the tool. The specialized code processed by the TRIMMER is then given as input to the linker . The linker also reads the linker flags from the manifest file and generates a final specialized binary executable file.

3 Overview

How these tools solve debloating

```
\documentclass[11pt]{article}
```

4 Setup

Setup for running the pipelines.

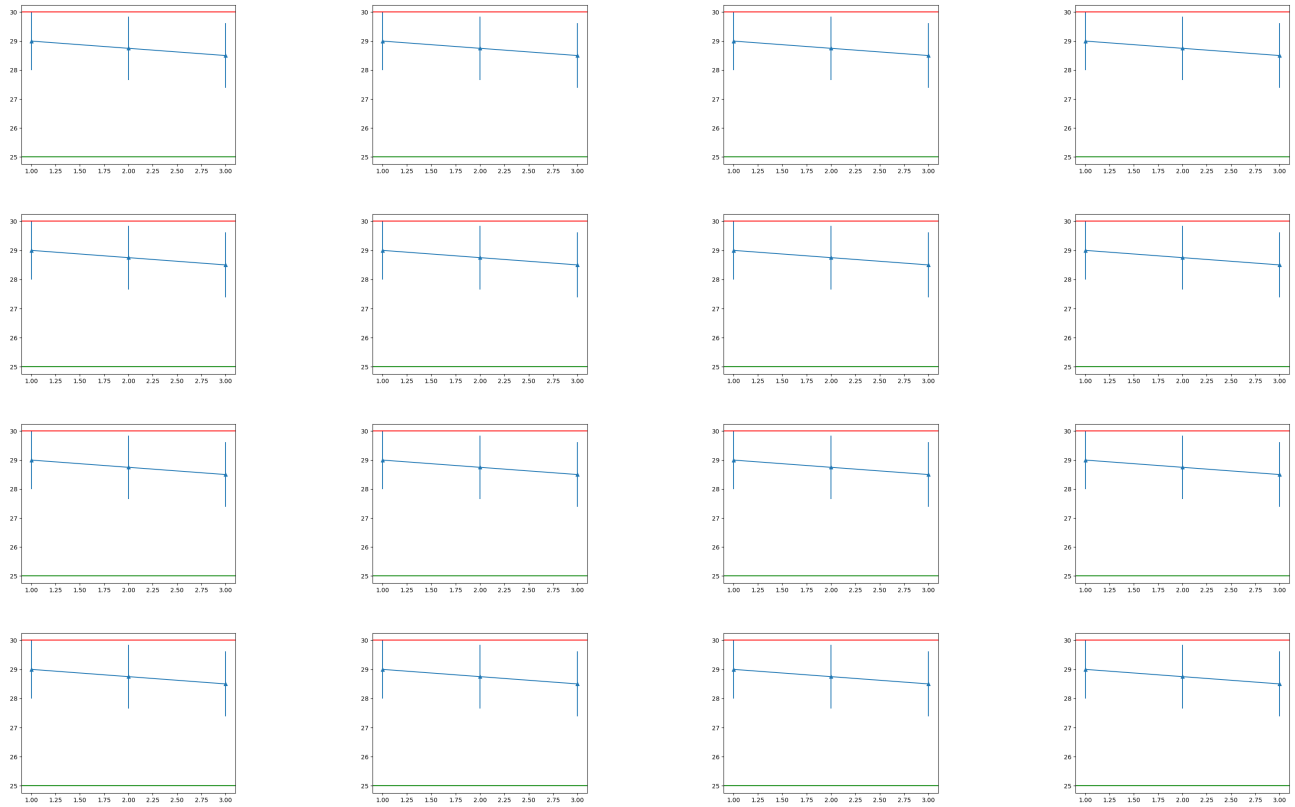
4.1 Aim

Footnotes are inserted with the `\footnote` command.[1]

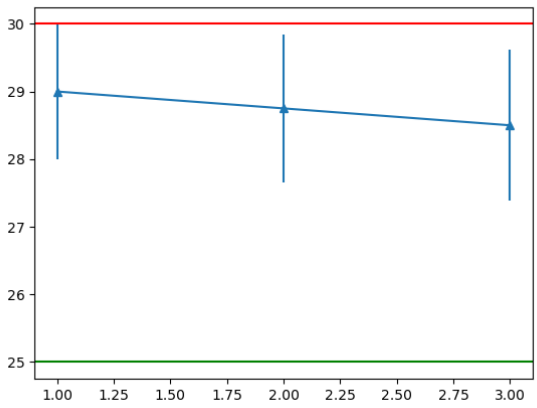
4.2 Tables and figures

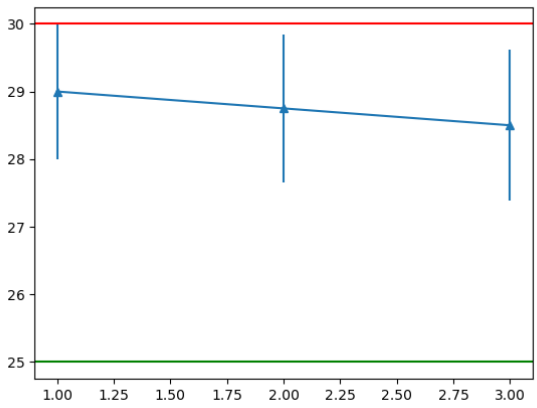
See Table 1 for an example of a table and its caption. **Do not override the default caption sizes.**

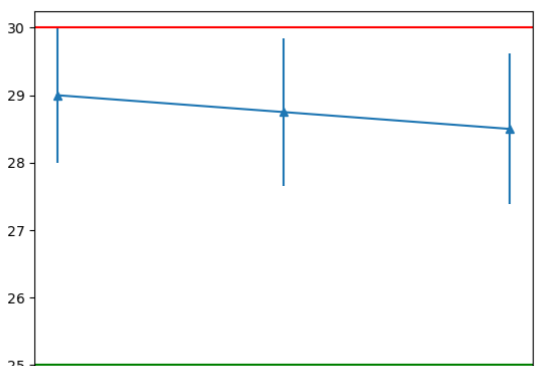
[1] This is a footnote.

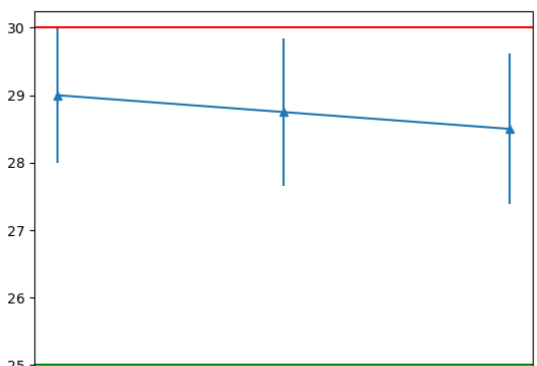


Write the description here. Dummy description .Write the description here. Dummy description Write the description here. Dummy description Write the description here. Dummy description Write the description here. Dummy description Write the description here. Dummy description Write the description here. Dummy description Write the description here. Dummy description Write the description here. Dummy description Write the description here. Dummy description Write the description here. Dummy description Write the description here. Dummy description Write the description here. Dummy description.









Command	Output	Command	Output
{\"a}	ä	{\c c}	ç
{\^e}	ê	{\u g}	ğ
{\`i}	ì	{\l}	ł
{\.I}	İ	{\~n}	ñ
{\o}	ø	{\H o}	ő
{\'u}	ú	{\v r}	ř
{\aa}	å	{\ss}	ß

Table 1: Example commands for accented characters, to be used in, *e.g.*, BibTeX entries.

4.3 Observations

Users of older versions of LaTeX may encounter the following error during compilation:

```
\pdfendlink ended up in different
nesting level than \pdfstartlink.
```

This happens when pdfLaTeX is used and a citation splits across a page boundary. The best way to fix this is to upgrade LaTeX to 2018-12-01 or later.

4.4 Inferences

Table 2 shows the syntax supported by the style files. We encourage you to use the natbib styles. You can use the command `\citet` (cite in text) to get "author (year)" citations, like this citation to a paper by Gusfield (1997). You can use the command `\citep` (cite in parentheses) to get "(author, year)" citations (Gusfield, 1997). You can use the command `\citealp` (alternative cite without parentheses) to get "author, year" citations, which is useful for using citations within parentheses (e.g. Gusfield, 1997).

4.5 Failures

The LaTeX and BibTeX style files provided roughly follow the American Psychological Association format. If your own bib file is named `custom.bib`, then placing the following before any appendices in your LaTeX file will generate the references section for you:

```
\bibliographystyle{acl_natbib}
\bibliography{custom}
```

You can obtain the complete ACL Anthology as a BibTeX file from https://aclweb.org/anthology/anthology.bib.gz. To include both the Anthology and your own .bib file, use the following instead of the above.

```
\bibliographystyle{acl_natbib}
\bibliography{anthology,custom}
```

Please see Section 5 for information on preparing BibTeX files.

4.6 Future Runs

See Appendix A for more artifacts

5 Final Result

6 Comparision Insights

7 Comparision Insights

8 Comparision Insights

9 Comparision Insights

10 Template Adaptation

This document has been adapted by Steven Bethard, Ryan Cotterell and Rui Yan from the instructions for earlier ACL and NAACL proceedings, including those for ACL 2019 by Douwe Kiela and Ivan Vulić, NAACL 2019 by Stephanie Lukin and Alla Roskovskaya, ACL 2018 by Shay Cohen, Kevin Gimpel, and Wei Lu, NAACL 2018 by Margaret Mitchell and Stephanie Lukin, BibTeX suggestions for (NA)ACL 2017/2018 from Jason Eisner, ACL 2017 by Dan Gildea and Min-Yen Kan, NAACL 2017 by Margaret Mitchell, ACL 2012 by Maggie Li and Michael White, ACL 2010 by Jing-Shin Chang and Philipp Koehn, ACL 2008 by Johanna D. Moore, Simone Teufel, James Allan, and Sadaoki Furui, ACL 2005 by Hwee Tou Ng and Kemal Oflazer, ACL 2002 by Eugene Charniak and Dekang Lin, and earlier ACL and EACL formats written by several people, including John Chen, Henry S. Thompson and Donald Walker. Additional elements were taken from the formatting instructions of the *International Joint Conference on Artificial Intelligence* and the *Conference on Computer Vision and Pattern Recognition*.

References

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

Galen Andrew and Jianfeng Gao. 2007. Scalable training of L1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 33–40.

Dan Gusfield. 1997. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, UK.

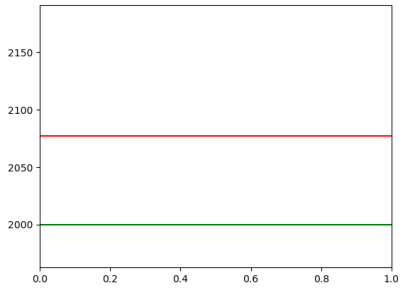
Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *Computing Research Repository*, arXiv:1503.06733. Version 2.

A Example Appendix

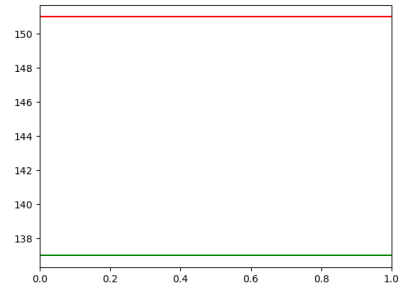
This is an appendix.

Output	natbib command	Old ACL-style command
(Gusfield, 1997)	\citep	\cite
Gusfield, 1997	\citealp	no equivalent
Gusfield (1997)	\citet	\newcite
(1997)	\citeyearpar	\shortcite

Table 2: Citation commands supported by the style file. The style is based on the natbib package and supports all natbib citation commands. It also supports commands defined in previous ACL style files for compatibility.

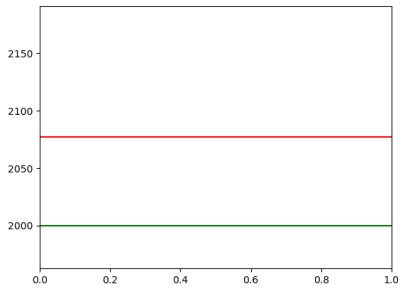


(a) OOCAM Run : **`bzip2`**

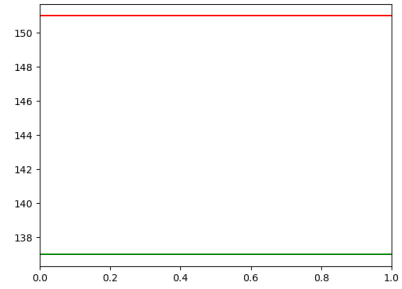


(b) OOCAM Run : **`module`** example

Figure 1: OCCAM Tool runs on **`bzip2`** & **`module`**. **Before** and **After** debloating

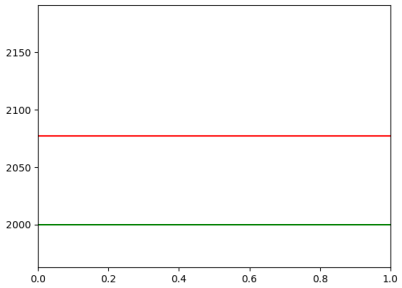


(a) OOCAM Run : **`bzip2`**

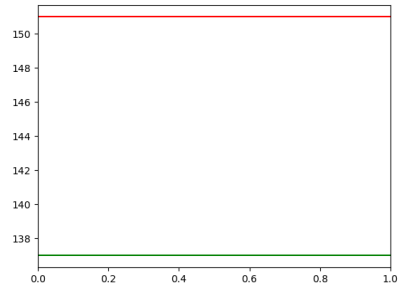


(b) OOCAM Run : **`module`** example

Figure 2: OCCAM Tool runs on **`bzip2`** & **`module`**. **Before** and **After** debloating

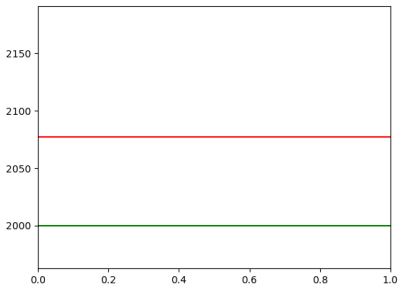


(a) OOCAM Run : **`bzip2`**

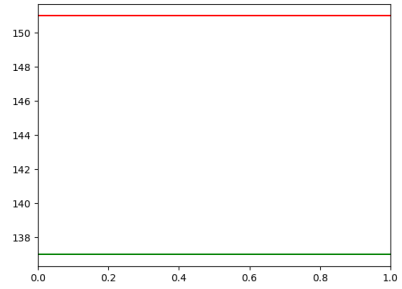


(b) OOCAM Run : **`module`** example

Figure 3: OCCAM Tool runs on **`bzip2`** & **`module`**. **Before** and **After** debloating

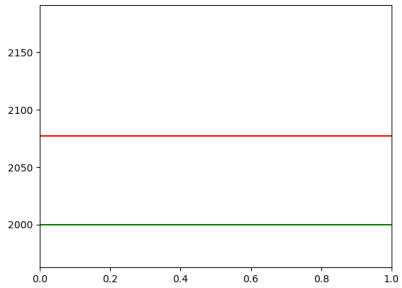


(a) OOCAM Run : **`bzip2`**

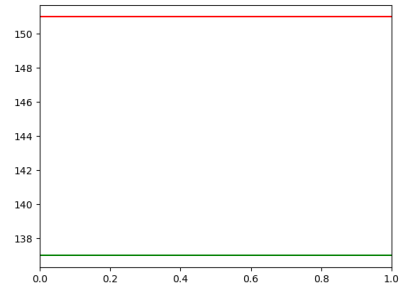


(b) OOCAM Run : **`module`** example

Figure 4: OCCAM Tool runs on **`bzip2`** & **`module`**. **Before** and **After** debloating

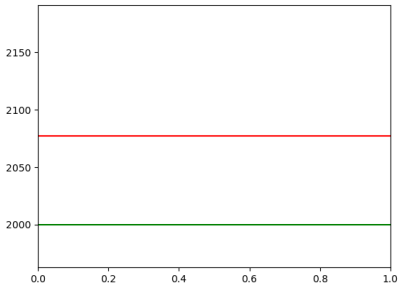


(a) OOCAM Run : **`bzip2`**

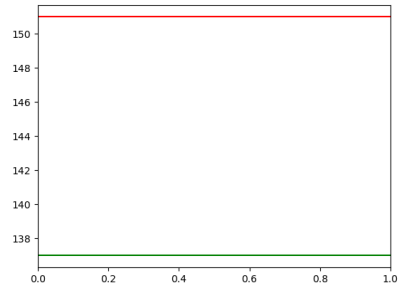


(b) OOCAM Run : **`module`** example

Figure 5: OCCAM Tool runs on **`bzip2`** & **`module`**. **Before** and **After** debloating

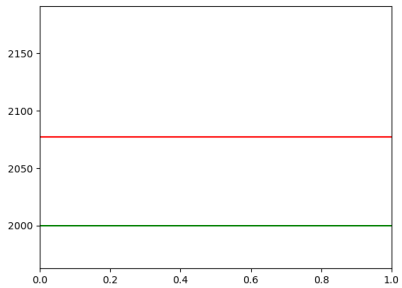


(a) OOCAM Run : **`bzip2`**

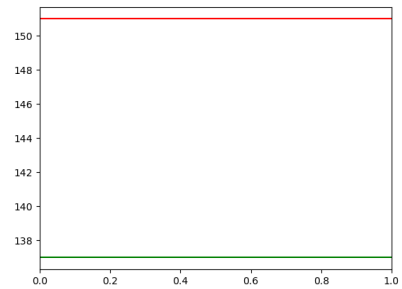


(b) OOCAM Run : **`module`** example

Figure 7: OCCAM Tool runs on **`bzip2`** & **`module`**. **Before** and **After** debloating

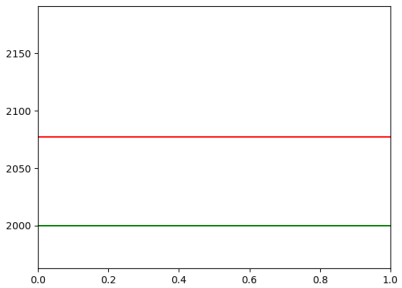


(a) OOCAM Run : **`bzip2`**

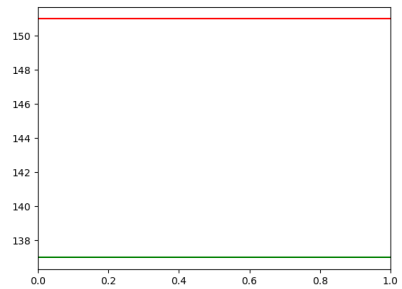


(b) OOCAM Run : **`module`** example

Figure 6: OCCAM Tool runs on **`bzip2`** & **`module`**. **Before** and **After** debloating

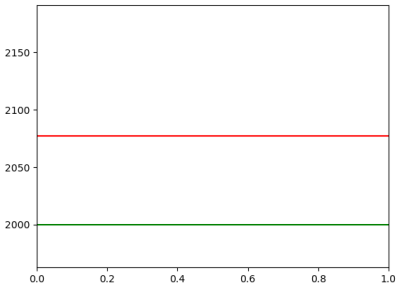


(a) OOCAM Run : **`bzip2`**

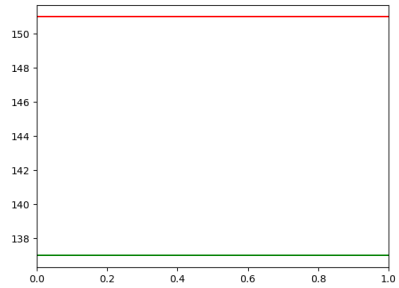


(b) OOCAM Run : **`module`** example

Figure 8: OCCAM Tool runs on **`bzip2`** & **`module`**. **Before** and **After** debloating

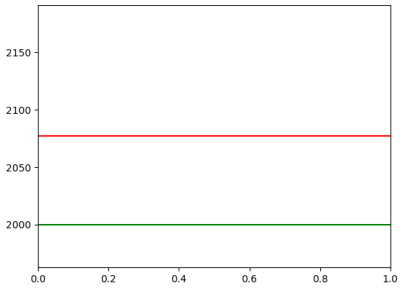


(a) OOCAM Run : **`bzip2`**

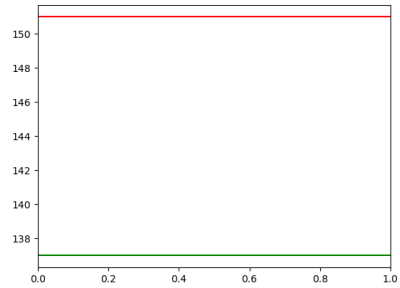


(b) OOCAM Run : **`module`** example

Figure 9: OCCAM Tool runs on **`bzip2`** & **`module`**. **Before** and **After** debloating

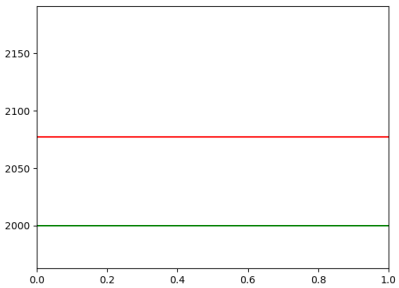


(a) OOCAM Run : **`bzip2`**

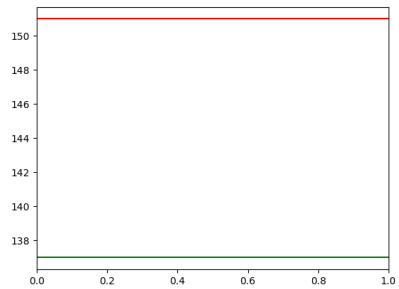


(b) OOCAM Run : **`module`** example

Figure 11: OCCAM Tool runs on **`bzip2`** & **`module`**. **Before** and **After** debloating

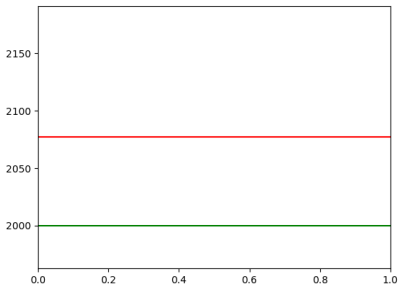


(a) OOCAM Run : **`bzip2`**

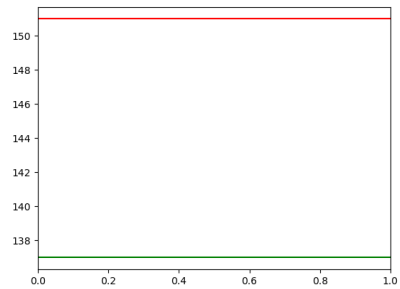


(b) OOCAM Run : **`module`** example

Figure 10: OCCAM Tool runs on **`bzip2`** & **`module`**. **Before** and **After** debloating

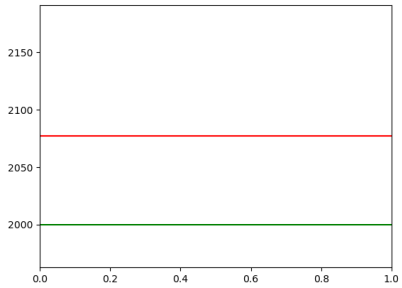


(a) OOCAM Run : **`bzip2`**

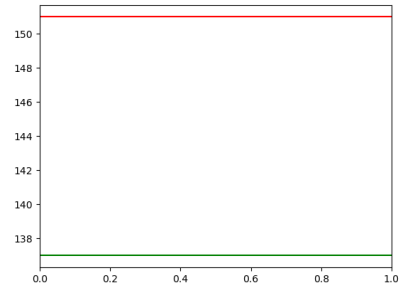


(b) OOCAM Run : **`module`** example

Figure 12: OCCAM Tool runs on **`bzip2`** & **`module`**. **Before** and **After** debloating

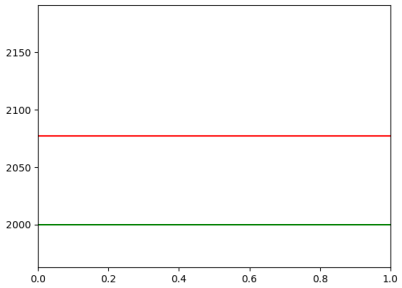


(a) OOCAM Run : **`bzip2`**

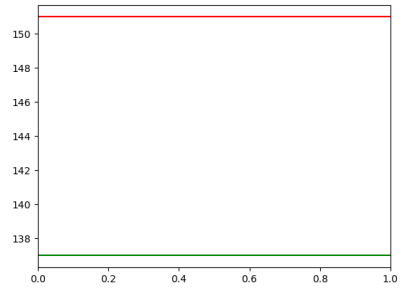


(b) OOCAM Run : **`module`** example

Figure 13: OCCAM Tool runs on **`bzip2`** & **`module`**. **Before** and **After** debloating

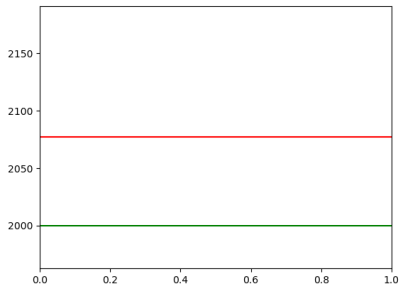


(a) OOCAM Run : **`bzip2`**

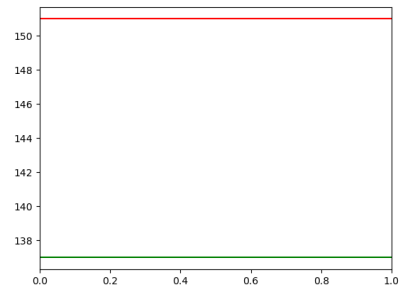


(b) OOCAM Run : **`module`** example

Figure 15: OCCAM Tool runs on **`bzip2`** & **`module`**. **Before** and **After** debloating

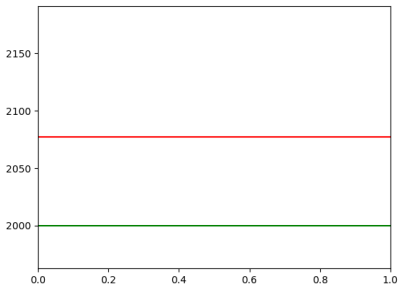


(a) OOCAM Run : **`bzip2`**

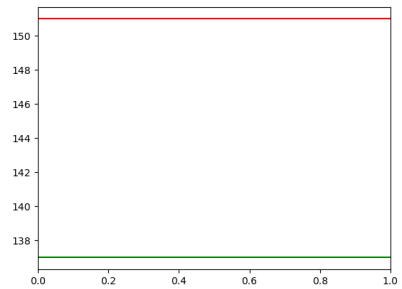


(b) OOCAM Run : **`module`** example

Figure 14: OCCAM Tool runs on **`bzip2`** & **`module`**. **Before** and **After** debloating



(a) OOCAM Run : **`bzip2`**



(b) OOCAM Run : **`module`** example

Figure 16: OCCAM Tool runs on **`bzip2`** & **`module`**. **Before** and **After** debloating