Generating Propagation Complete Encodings in Haskell

Arunothia Marappan

Under Guidance of Graeme Gange

Submitted to Prof. Harald Sondergaard as a part of Research Internship (May'16 - July'16)

Abstract

THE UNIVERSITY OF
MELBOURNE

Computing and Information Systems
University of Melbourne

1 Introduction

In this project, we implemented the algorithm given in [1] in Haskell.

2 Methodology

We define Partial Assignments as Maybe[PAValue] where PAValue is the data type that can either be True, False or Question. The ordering amongst Partial Assignments are established as - Nothing (contradicting Partial Assignment) as the least and the more the undefined the partial assignment is, higher it is in the ordering. For the implementation of Priority Queues, we use Haskell Package Data.Heap. From the ordering of Partial Assignments mentioned, it is clear that we used maxHeap to mimic the priority queue desired. Sat Solver for the implementation has been taken from [2].

3 Examples

3.1 If-then-Else Gadget

$$x = if\ b\ then\ y\ else\ z$$

Which can be encoded as -

$$(b \longrightarrow (x \longleftrightarrow y)) \wedge (\neg b \longrightarrow (x \longleftrightarrow z))$$

Which in the CNF form corresponds to

$$(\neg b \vee \neg x \vee y) \wedge (\neg b \vee \neg y \vee x) \wedge (b \vee \neg x \vee z) \wedge (b \vee \neg z \vee x)$$

References

[1] Martin Brain, Liana Hadarean, Daniel Kroening, and Ruben Martins. Automatic generation of propagation complete sat encodings. In *Verification, Model Checking, and Abstract Interpretation (VMCAI)*, volume 9583 of *Lecture Notes in Computer Science*, pages 536–556. Springer, 2016.

[2] gatlin. sat.hs. https://gist.github.com/gatlin/1755736, 2012. [Online; accessed June-2016].