From beetles to ladybugs...

Delta debugging #SAT model counters

{ David N. Coroian }
{ Supervisor: Dr. Anna L.D. Latour }

TU Delft

1. Counting SAT

#SAT - counting variant of SAT

Example: $p \vee q$

POV:

- SAT - satisfiable
- #SAT = 3 solutions

Solvers - recent BOOM in development

Scope: unweighted model counting

2. Delta Debugging

Minimising input...

$$(p \vee q \vee r) \wedge (o \vee (\neg z)) \wedge (b \vee d)$$

$$\downarrow$$

$$(b \vee d)$$

...whilst keeping bugs.

3. State of the art?

- cnfdd [Brummayer *et al.*, 2010], SAT delta debugger
  - Based on dd-min [Zeller *et al.*, 2002]
  - Integrating domain-specific knowledge
- TestMC [Usman *et al.*, 2020], proposed #SAT delta debugger
  - Based on dd-min
  - Code not available

Contact Information

Web www.tudelft.nl/ewi
Email D.N.Coroian@student.tudelft.nl

Δ 🐞

[a]Picture generated using `https://deepai.org`

Scan me for paper

4. Approach

1. Implement Probabilistic Delta Debugging (prob-dd) [Wang *et al.*, 2021]
2. Apply cnfdd to model counters
3. Compare performance

5. Heuristics

Initial prob-dd probabilities:

- H1 - equal values of 0.1
- H2 - based on number of literals
- H3 - based on rarity of literals

6. No real bugs

Three solvers

+

100.000 instances

=

Only timeouts, no crashes or wrong counts

7. Results

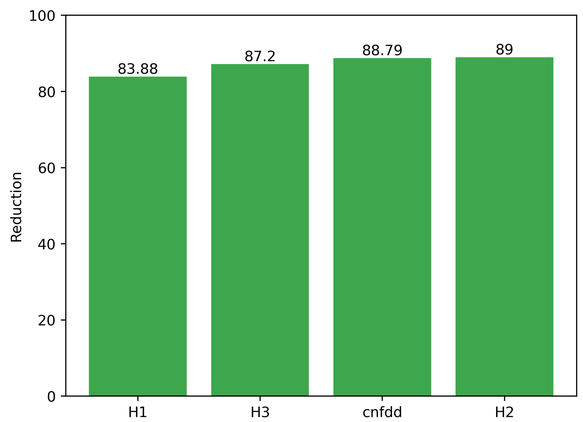

Figure 1: Average input reduction (%).

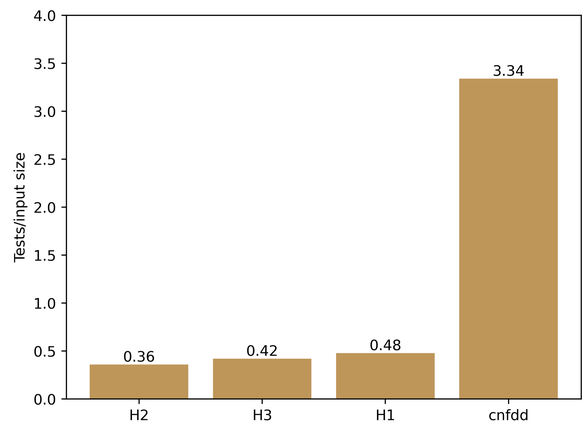

Figure 2: Average ratio between number of delta-debugging tests and input size.

Conclusion: H2 - similar reduction to cnfdd, performing $\sim 10\times$ less tests