UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING
Department of Electrical & Computer Engineering

Prioritizing Lemma Pushing

Hari Govind V K[1], Arie Gurfinkel[1], Yakir Vizel[2] and Vijay Ganesh[1]

[1]Department of Electical and Computer Enginerring, University of Waterloo
[2]Technion - Israel Institute of Technology

OBSERVATION

In IC3 based algorithms, order of pushing lemmas has significant effect on performance

RESEARCH CHALLENGE

Finding a good ordering heuristic : static, dynamic, ML based?

SAFETY

Given an initial state *Init* and a transition relation *Tr*, check whether all reachable states satisfy property P

e.g. : adding up numbers

Init : $count = 0$, $sum = 0$ Tr : count++; sum+=count; if(count==10) count=0; sum=0 P : $sum \leq 45$

To prove safety, many algorithms construct an inductive invariant *Inv* such that $Init \Rightarrow Inv$ $\quad Inv \wedge Tr \Rightarrow Inv$ $\quad Inv \Rightarrow P$

$sum = count * (count + 1) * 0.5 \wedge count \geq 0 \wedge count < 10$

INCREMENTAL PROOFS

IC3 based Model Checking algorithms construct *Inv* incrementally

- Prove P at some bound
  - Conjoin several lemmas to prove the property

$sum = count * (count + 1) * 0.5 \wedge count < 10$ proves P at a bound

- Add supporting lemmas to push the lemmas to higher frames

adding $count \geq 0$ makes it inductive

SUPPORT SET

Set of lemmas required to push a lemma to the next higher frame

$SS(sum = count * (count + 1) * 0.5)$: $\{\ sum = count * (count + 1) * 0.5,\ count \geq 0\ \}$

Consider pushing 2 lemmas C_1 and $C_2 \subsetneq C_1$ in two different orders

- C_1 first then C_2 : 2 different proofs, wasted effort
- C_2 first then C_1 : since $C_2 \Rightarrow C_1$ We get C_1 for free after the first proof

Pushing $count \geq 0$ makes $count \geq 0 \wedge sum = count * (count + 1) * 0.5$ an inductive invariant

Without $count \geq 0$, $sum = count * (count + 1) * 0.5$ would have to be pushed to all bounds

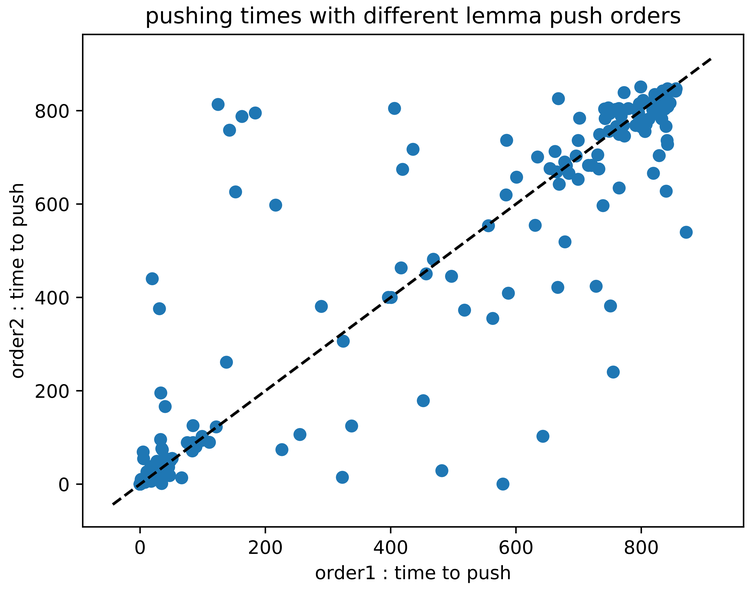

Figure 1: AVY with 2 different static orderings. Along each axis is the time (in seconds) the solver spent on pushing. Order of pushing has a lot of impact on pushing time.

WHICH LEMMA TO PUSH FIRST ?

Prioritize lemmas with

- high utility
- low effort to push

UTILITY OF A LEMMA

Define

- Utility(P) =1
- Utility$(L) = \Sigma_{\{l|L \in SS(l)\}} Utility(l)$

Dynamic : can change every time a lemma is pushed

EFFORT TO PUSH

Pushing lemmas from frame to frame is a repetition of previous proofs at higher bounds

Estimate the effort required from past experiences of pushing the lemma

UTILITY AND SUPPORT SETS

Different ways of computing support sets $\Rightarrow$ different utility values

Consider pushing the lemma $count \leq 10$ in two different ways

1) $count \leq 10 \wedge count \neq 10 \wedge Tr \Rightarrow count' \leq 10$
   SS $= \{count \neq 10, count \leq 10\}$
2) $count \leq 10 \wedge Tr \wedge count' \leq 10 \wedge Tr \Rightarrow count'' \leq 10$
   SS $= \{count \leq 10, count' \leq 10\}$

By leveraging k-induction, the number of distinct lemmas in the support set can be minimized

Greedily minimize support set $\not\Rightarrow$ minimal aggregate push effort
Aiming for lemma re-use is better

REFERENCES

- Aaron R. Bradley *SAT-based model checking without unrolling.* In VMCAI 2011
- Ryan Berryhill, Alexander Ivrii, Neil Veira, Andreas G. Veneris *Learning support sets in IC3 and Quip: The good, the bad, and the ugly.* In FMCAD 2017
- Yakir Vizel, Arie Gurfinkel *Interpolating Property Directed Reachability.* In CAV 2014
- Arie Gurfinkel, Alexander Ivrii *Pushing to the Top.* In FMCAD 2015

hgvedira@uwaterloo.ca