A Practical Tool for Finding Provable Arithmetic Bounds

Sol Swords

ACL2 Workshop 2023

intel

Problem context

Problem: Prove that a hardware design implements some arithmetic algorithm.

Inevitable challenges:

- Intermediate values are assumed to fit in certain bit widths
- If you don't have tight enough bounds for A, B, C, won't be able to show A*B+C fits in its assumed width
- Other HW tricks – e.g. the MSBs will always be 101, so don't bother storing them.
- *Need "good enough" bounds on the computed values.*

Finding Bounds

- Bitblasting can find exact bounds
  - But doesn't scale
- ACL2 has linear/nonlinear decision procedures
  - But they won't tell you what's the best bound they can prove
  - Also sometimes don't scale
- Algorithm might suggest theoretical bounds
  - But math is hard.
- Practical approach: Find the best bound you can easily prove. If insufficient, go back and try harder.

Tool for the practical approach: `def-bounds`

Find bounds that we can prove, then prove them.

Features

- Initial simplification: phased rewriting and case splitting
- Core algorithm: abstract interpretation based on ranges
- Linear rules, user hints, typeset reasoning provide base bounds for abstract interpretation
- Runs once to find the bounds, produces defthm to prove them

Abstract interpretation sketch

$$a \in [a_l, a_u], b \in [b_l, b_u] \Rightarrow a + b \in [a_l + b_l, a_u + b_u]$$

$$a \in [a_l, a_u], b \in [b_l, b_u]$$
$$\Rightarrow ab \in [\min(a_l b_l, a_l b_u, a_u b_l, a_u b_u), \max(a_l b_l, a_l b_u, a_u b_l, a_u b_u)]$$

$$a \in [a_l, a_u]$$
$$\Rightarrow a^2 \in \left[\max\left(\min\left(a_l^2, a_l a_u, a_u^2\right), 0\right), \max\left(a_l^2, a_l a_u, a_u^2\right)\right]$$

(Bounds may be infinite as well.)

Example – first try

```
(defund foo (x)
  (- (* x x) (* 3 x)))

(def-bounds foo-bounds
  (foo x)
  :hyp (and (rationalp x)
            (<= 2 x)
            (<= x 4))
  :simp-hints ((:in-theory (enable foo))))
```

$$x \in [2,4] \Rightarrow x^2 \in [4,16], 3x \in [6,12], x^2 - 3x \in [-8,10]$$

Actual range is [-2, 4] – [-8, 10] isn't a great result…

Example – better

```
(defund foo (x)
  (- (* x x) (* 3 x)))

(defthmd my-factor
  (equal (+ (- (* 3 x)) (* x x))
         (* x (- x 3))))

(def-bounds foo-bounds
 …
 :simp-hints ((:in-theory (enable foo))
              (:in-theory (enable my-factor))))
```

$$x \in [2,4] \Rightarrow x - 3 \in [-1,1], x(x-3) \in [-4,4]$$

Better…

Example – extreme

```
(defund foo (x)
  (- (* x x) (* 3 x)))

(defthmd my-factor …)
(def-bounds foo-bounds
 …
 :cases ((:ranges-from-to-by x 2 4 1/64)))
```

Result after 128-way case split: $\left[-\frac{129}{64}, 4\right]$ -- close to the actual range

Conclusion

- Practical tool for finding and proving bounds
- Allows for various levels of effort depending how tight a bound is needed
  - Rewrite term toward formulation that yields narrowest bounds
  - Case split to mitigate imprecision due to correlations between subterms
- Successfully used at Intel for FP SQRT verification
- Community books -- centaur/misc/def-bounds