Abstraction techniques for Floating-Point Arithmetic

Angelo Brillout\textsuperscript{1}, Daniel Kroening\textsuperscript{2} and Thomas Wahl\textsuperscript{2}

\textsuperscript{1}ETH Zurich, \textsuperscript{2}Oxford University
Floating-Point Arithmetic (FPA)

- Used for embedded and safety critical systems
- Finite representation of real numbers
  - Rounding
  - Deviation causes unintuitive results
  - Deviation can change control flow
- Behavior of floating-point programs hard to predict
Contributions

→ New effective approximation techniques
  - Over- and underapproximation for FPA
  - Bit-precise

→ Precise and sound decision procedure for FPA:
  - Based on CBMC model checking engine
  - SAT solver as the back-end
Floating-Point Arithmetic (FPA)

- Numerical representation of a subset of the reals
- Floating-point format: IEEE-754 standard
  - Triple \((s, e, f)\) stands for the number \((-1)^s \cdot f \cdot 2^e\)
  - Represented by a bit-vector

\[
\begin{array}{ccccccc}
 s & e_{r-1} & \cdots & e_0 & f_0 & \cdots & f_{p-1} \\
 \downarrow & \downarrow & \cdots & \downarrow & \downarrow & \cdots & \downarrow \\
 1 & r & & & p & & \\
\end{array}
\]

- Representable numbers \(\mathbb{F}_p\)
- Floating-point operations \(\oplus \ominus \circlearrowright\)
  - Differ from real arithmetic. E.g.:
  \[
  (a \oplus b) \oplus c \neq a \oplus (b \oplus c)
  \]
Floating-Point Arithmetic (FPA)

- Result of FP-operation not always representable

→ Approximations:

\[
[x]_p := \max\{f \in \mathbb{F}_p : f \cdot x\}, \quad \text{and}
\]

\[
[x]_p := \min\{f \in \mathbb{F}_p : f \geq x\}.
\]

→ Rounding function:

\[
rd_p(x) \in \{[x]_p, [x]_p\}
\]

- Rounding based on least significant bits of fraction
Floating-Point Arithmetic (FPA)

- Floating-point operations defined as:
  \[ x \odot_p y := r d_p (x \odot y) \]

- Verification of FPA programs:
  - Naïve method: Bit-vector model of an FPU and bit-blasting
  - BMC (Unrolling, Bit-blasting, SAT-solving)

→ Does not scale for FPA
FPA Verification

- FPU-Implementation of Add/Sub

  - Align: mantissa shifted, rendering exponents equal
  - Add/Sub: resulting mantissas are added/subtracted
  - Round: shortening mantissa to obtain a number in $\mathbb{F}_p$
FPA Verification

- FPU-Implementation of Add/Sub

<table>
<thead>
<tr>
<th>Precision</th>
<th>ALIGN</th>
<th>ADD/SUB</th>
<th>ROUND</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>p = 5</td>
<td>295</td>
<td>168</td>
<td>572</td>
<td>1035</td>
</tr>
<tr>
<td>p = 23</td>
<td>687</td>
<td>420</td>
<td>1447</td>
<td>2554</td>
</tr>
<tr>
<td>p = 52</td>
<td>1404</td>
<td>826</td>
<td>2923</td>
<td>5153</td>
</tr>
</tbody>
</table>
FPA Verification

- FPU-Implementation of Mul/Div

- Add/Sub: exponents added/subtracted (Mul/Div)

- Mul/Div: mantissas multiplied/divided (Mul/Div)
FPA Verification

- FPU-Implementation of Mul/Div

<table>
<thead>
<tr>
<th>Precision</th>
<th>MUL/DIV</th>
<th>ADD/SUB</th>
<th>ROUND</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p = 5$</td>
<td>280</td>
<td>94</td>
<td>674</td>
<td>1048</td>
</tr>
<tr>
<td>$p = 23$</td>
<td>3898</td>
<td>94</td>
<td>2258</td>
<td>6550</td>
</tr>
<tr>
<td>$p = 52$</td>
<td>19268</td>
<td>94</td>
<td>5742</td>
<td>25104</td>
</tr>
</tbody>
</table>
FPA Verification

→ Need for approximate FP-operations

Can we approximate FP-operations by reducing the precision $p$ ?
Approximation techniques

- Reducing the precision $p' < p$
  - Least significant bits are lost

- Overapproximation by **open** rounding:
  \[ \overline{rd}_{p,p'}(X) := \left[ \left\lfloor X \right\rfloor_{p'}, \left\lceil X \right\rceil_{p'} \right] \cap \mathbb{F}_p \]

- New FP-operations
  \[ X \odot_{p,p'} Y := \overline{rd}_{p,p'}(X \circ Y) \]

- Replace $\odot_p$ by $\overline{\odot}_{p,p'}$ for some precision $p' < p$
Approximation techniques

- Overapproximation: visualization

\[ \overline{rd}_{p,p'}(\{x\}) = \left[ \left[ x \right]_{p'}, \left[ x \right]_{p'} \right] \cap \mathbb{F}_p \]

precision \( p' < p \)

\[ \overline{rd}_{p,p'}(\{x\}) = \{*,*,*,*,*\} \]
Approximation techniques

- Reducing the precision $p' < p$
  - Least significant bits are lost
- Underapproximation by inhibiting rounding:
  $$r_{d_{p,p'}}(X) := X \cap \mathbb{F}_{p'}$$
- New FP-operations
  $$X \odot_{p,p'} Y := r_{d_{p,p'}}(X \circ Y)$$
- Replace $\odot_p$ by $\odot_{p,p'}$ for some precision $p' < p$
Approximation techniques

- **Underapproximation: visualization**

  \[ \overline{rd}_{p,p'}(\{x\}) = \{x\} \cap \mathbb{F}_{p'} \]

  precision \( p' < p \)

  \[ \overline{rd}_{p,p'}(\{x\}) = \{x\} \text{ if } x \in \mathbb{F}_{p'}, \emptyset \text{ otherwise} \]
Alternating abstractions for FPA

- **Over-approximation**
  - Permits *more* execution traces than original program
  - SAT: no conclusion, UNSAT: assertions OK

- **Under-approximation**
  - Permits *less* execution traces than original program
  - SAT: assertion violated, UNSAT: no conclusion

- **Refinement:** increase $p$

→ Alternation yields complete procedure
Alternating abstractions for FPA

Select small precision \( p \)

\[ \phi \]

Generate Underapproximation \( \phi \)

(increase \( p \) using \( \alpha \))

\[ \alpha \]

\[ \phi \]

\(\phi\)

SAT ?

yes

SAT, ass. \( \alpha \)

yes

\(\alpha\) satisfies \( \phi \)

yes (ass. \( \alpha \))

no

\( \phi \)

\( \phi \)

no

(proof \( P \))

P valid for \( \phi \) ?

no

\( \overline{\phi} \)

Generate Overapproximation

(increase \( p \) using \( P \))

\( \phi \)

\( \phi \)

no

\( \overline{\phi} \)

UNSAT, proof \( P \)

no

SAT ?

\( P \) valid for \( \phi \) ?

proof \( P \)

\( \overline{\phi} \)

\( \phi \)
Alternating abstractions for FPA

Refinement for FPA:

- **Spuriously SAT:**
  
  \[ r \text{ result of } \bigcap_{p, p'} \text{. If } r \neq \bigcap_p \text{ then increase precision} \]

- **Spuriously UNSAT:**
  
  - **Recall:**
    
    \[ \overline{r d}_{p, p'}(X) := X \cap F_{p'} \]
  
  - If the constraint \( X \cap F_{p'} \) occurs in \( P \), then increase precision
Summary

- Model Checking with FPA
  - Effective over- and underapproximation hard to find
  - Slow (model checking)
    - Fully automatic
    - Provides counterexample

→ Implemented in CBMC
State of the Art

- Proof assistants
  - Very powerful
  - Require interaction
  - No counterexample

- Interval arithmetic \[ [1, 2] + [4, 6] = [5, 8] \]
  - Fully automated
  - Too coarse
  - No counterexample
Issues

- E.g. the formula \((a \oplus b) \oplus c \neq a \oplus (b \oplus c)\) is SAT
  - Every overapproximation based on \(\bar{\ominus}\) is SAT
  - Every underapproximation based on \(\ominus\) is UNSAT

\[\rightarrow \text{Some formulae do not have effective over- or underapproximations}\]
Conclusion

- New algorithm for iteratively approximating complex FPA –formulae
  - New under- and over-approximations for FP-operations
- Ability to generate counterexamples
  - Debugging
  - Automated test-vector generation
- Promising experiments, future work

Thank you!