Reasoning about Relaxed Programs

Michael Carbin

Massachusetts Institute of Technology
How do we verify the safety of relaxed programs?
Standard Program Model

One point in tradeoff space
When do we think like this?

Media Processing, Machine Learning, Search
When do we think like this?

Accuracy

0%

100%

Time/Resources/Cost

Highly Accurate, Expensive

Less accurate, Inexpensive
Relaxed programs can dynamically and automatically adapt.
Producing Relaxed Programs

**Task Skipping/Loop Perforation** - Rinard ICS ‘06, Misailovic ICSE ‘10

**Dynamic Knobs** - Hoffmann ASPLOS ‘11

**Approximate Memories** - Lui ASPLOS ‘11, Sampson PLDI ‘11

**Approximate Memoization** - Chaudhuri FSE ‘11

**Racy Parallelization** - Misailovic MIT-TR ‘10, Rinard RACES ‘12
General Model for Relaxed Programs

A general primitive for relaxed sequential programs [1]:

\[
\text{relax} \ (n) \ \text{st} \ (n \leq \text{old}(n));
\]

\[
\text{for} \ (\text{uint} \ i = 0; \ i < n; \ ++i) \ \{\ldots\}
\]

How do we verify the safety of relaxed programs?
Program Logic (Hoare Logic)

\[ \{P\} \ s \ \{Q\} \]

Standard Hoare Logic doesn’t capture what we want

\[ \{x = 1\} \ x = x + 1 \ \{x = 2\} \]
Applying Standard Hoare Logic

Note: relaxation doesn’t modify y

Why do we need to prove S? If S(y) holds in the original program, then it also holds in the relaxed
Alternative: Relational Program Logic

$$\{ P_{rel} \} \ s \ \{ Q_{rel} \}$$

\[
\begin{align*}
\{x<r> == x<o> \ & \land \ y<r> == y<o>\} \\
\text{relax (x) st (true);} \\
\{y<r> == y<o>\}
\end{align*}
\]
Applying Relational Program Logic

\[
\begin{align*}
\{x<r> &= x<o> && y<r> &= y<o> \}\\
\text{relax } (x) \text{ st } (\text{true});
\\
\{ y<r> &= y<o> \}
\end{align*}
\]

\[
\begin{align*}
\{R(x<r>, y<r>) && y<r> &= y<o> \}\\
\text{assert } R(x, y) && S(y) ;
\end{align*}
\]

If \( S(y<o>) \) is true and \( y<r> = y<o> \) then \( S(y<r>) \) is true

Relational reasoning is the bridge
Relative Safety

If original program satisfies all assertions, then the relaxed program satisfies all assertions

Established through any means: verification, testing, code review

In our PLDI paper:
• Full formalization of the relaxed programming model
• Primitives for reasoning about accuracy
• Examples from racy parallelization, approximate memory, and dynamic knobs
Takeaway

\[
\text{for } (\text{uint } i = 0; i < n; ++i) \{ \ldots \}
\]

\[
\downarrow
\]

\[
\text{for } (\text{uint } i = 0; i < n; i+=2) \{ \ldots \}
\]

Relax Semantics. Preserve Safety. Reuse Proofs