Solver-aided DSL with ROSETTE

Rui Zhao (rz2290)
Solver-aided Programming

- Software is widely used
- We all want to build programs, not only software engineers
Solver-aided Programming

- 1960 - Software crisis
- 1970 - Program logics
- 1980 - Mechanization of logic
- 1990 - Mechanized tools
Solver-aided Programming

- We all want to build programs
  - less code
  - less time
  - less effort
Solver-aided Programming

- 1960 - Software crisis
- 1970 - Program logics
- 1980 - Mechanization of logic
- 1990 - Mechanized tools
- 2000 - Solvers and tools, eg. SAT, SMT
- 2010 - Solver-aided Languages
Four Elementary Queries

- **S**: synthesize a code fragment
- **V**: checking that an implementation satisfies a desired property
- **L**: localizing code fragments that cause an undesired behavior
- **A**: asking an angelic oracle to divine values that make the execution satisfy a specification
Programming

• Specification

P(x) {
  ...
  ...
  ...
}


Programming

• I have test cases

```plaintext
P(x) {
  ...
  ...
}
assert(safe(p(2)))
```
Programming with Solver-aided tools

• I do not have test cases

```plaintext
P(x) {
...
...
}
assert(safe(p(X)))
```

translate....
solver
Programming with Solver-aided tools

• Verification

\[ P(x) \{ \]
\[ \quad \ldots \]
\[ \quad \ldots \]
\[ \} \]
\[ \text{assert(safe(p(X)))} \]

\[ \exists x . \neg \text{safe}(P(x)) \]

\[ \text{solver} \]
Programming with Solver-aided tools

• Find a value that fails the program

\[ P(x) \{
\ldots
\ldots
\}
\]

assert(safe(p(X))))

\[ \exists x . \neg \text{safe}(P(x)) \]
solver
Programming with Solver-aided tools

- Debugging

\[ P(x) \{ \]
\[ \quad v = x + 2; \]
\[ \quad ... \]
\[ \}
\[ \text{assert(safe(p(X)))} \]

\[ x = 42 \land \text{safe(P(x))} \]

solver
Programming with Solver-aided tools

• Debugging

```
P(x) {
  v = choose();
  ...
}
assert(safe(p(X)))
```

```
x = 42 \land \text{safe}(P(x))
```

```
solver
```
Programming with Solver-aided tools

• Find the pair that fails the execution

\[
P(x) \{ \\
v = \text{choose}(); \\
... \\
\}
\]

\[
\text{assert}(\text{safe}(p(X)))
\]

\[
\exists v . \text{safe}(P(42, v))
\]

\[
\text{solver}
\]
Programming with Solver-aided tools

• Synthesis

\[ P(x) \{ \]
\[ \begin{align*}
    v &= ? \quad ; \\
    \ldots \\
    \}
\]
\[ \text{assert(safe}(p(X))) \]

\[ \exists v \cdot \text{safe}(P(42, v)) \]

\[ \text{solver} \]
Programming with Solver-aided tools

• Synthesis

```
P(x) { 
v = x - 2;
...
}
assert(safe(p(X)))
```

\[
\exists e. \forall x. \text{safe}(P_e(x))
\]

solver
Current Problems

- It’s very hard to write a solver-aided tool / PL

1. Learn the problem domain
2. Design a domain language
3. Build a symbolic compiler from the domain language to constraints
Solution

- It’s very hard to write a solver-aided tool / PL

  1. Learn the problem domain
  2. Design a domain language
  3. Get a symbolic compiler for free, implement an interpreter for the new PL
Languages

- Layers

Diagram:

- DSL
  - Interpreter
  - library

- Host Language
Solver-aided Languages

- Layers

- SDSL
  - Interpreter
  - library

- Solver-aided Host Language

- Symbolic Virtual Machine
What is Rosette

• Solver-aided host language
• A framework for designing solver-aided programming languages
• Rosette itself is a solver-aided programming language embedded in Racket
• Frees designers from having to compile the new language to constraints
How does Rosette work

• Take BV as an example

```python
def bvmax(r0, r1):
    r2 = bvge(r0, r1)
    r3 = bvneg(r2)
    r4 = bvxor(r0, r2)
    r5 = bvand(r3, r4)
    r6 = bvxor(r1, r5)
    return r6
```
How does Rosette work

• Take BV as an example

```python
def bvmax(r0, r1):
    r2 = bvge(r0, r1)
    r3 = bvneg(r2)
    r4 = bvxor(r0, r2)
    r5 = bvand(r3, r4)
    r6 = bvxor(r1, r5)
    return r6
```

```scheme
(define bvmax
  `((2 bvge 0 1)
     (3 bvneg 2)
     (4 bvxor 0 2)
     (5 bvand 3 4)
     (6 bvxor 1 5))
)
```
How does Rosette work

• Take BV as an example

```
(define bvmax
  `(((2 bvge 0 1)
    (3 bvneg 2)
    (4 bvxor 0 2)
    (5 bvand 3 4)
    (6 bvxor 1 5))
)
```

```
(define (interpret prog inputs)
  (make-registers prog inputs)
  (for ([stmt prog])
    (match stmt
      [(list out opcode in ...)]
      (define op (eval opcode))
      (define args (map load in))
      (store out (apply op args))
    )
  )

(load(last))
```

> bvmax (-1,-2)
How does Rosette work

• Take BV as an example

\[
\text{(define bvmax `((2 bvge 0 1) (3 bvneg 2) (4 bvxor 0 2) (5 bvand 3 4) (6 bvxor 1 5)))}
\]

> bvmax (-1,-2)
How does Rosette work

• Take BV as an example

```
(define bvmax
  `(2 bvge 0 1)
    (3 bvneg 2)
    (4 bvxor 0 2)
    (5 bvand 3 4)
    (6 bvxor 1 5))
)
```

```
(define-symbolic n0 n1 number?)
(define inputs (list n0 n1))
(verify
  (assert (= (interpret bvmax inputs)
              (apply max inputs)))
)
```

> verify (bvmax, max)
  (0, -2)
> bvmax(0, -2)
  -1
How does Rosette work

• Take BV as an example

(define bvmax
  `((2 bvge 0 1)
    (3 bvneg 2)
    (4 bvxor 0 2)
    (5 bvand 3 4)
    (6 bvxor 1 5)))

(verify
  (assert (= (interpret bvmax inputs)
              (apply max inputs)))
)

> verify (bvmax, max)
  (0, -2)
> bvmax(0, -2)
  -1
How does Rosette work

• Take BV as an example

> debug (bvmax, max, (0, -2))

```
(define bvmax
  `((2 bvge 0 1)
    (3 bvneg 2)
    (4 bvxor 0 2)
    (5 bvand 3 4)
    (6 bvxor 1 5))
)
```

```
(define inputs (list 0 -2))
(define [input-register?]
  (assert (= (interpret bvmax inputs)
             (apply max inputs)))
)
```
How does Rosette work

• Take BV as an example

> debug (bvmax, max, (0, -2) )

```
(define bvmax
  '((2 bvge 0 1)
    (3 bvneg 2)
    (4 bvxor 0 2)
    (5 bvand 3 4)
    (6 bvxor 1 5))
)
```

```
(define inputs (list 0 -2))
(debug [input-register?]
  (assert (= (interpret bvmax inputs)
              (apply max inputs)))
)
```
How does Rosette work

• Take BV as an example

```scheme
(define bvmax
  `((2 bvge 0 1)
    (3 bvneg 2)
    (4 bvxor ? ?)
    (5 bvand 3 ?)
    (6 bvxor ? ?))
)
```

> synthesize (bvmax, max)

```scheme
(define inputs (list 0 -2))
(debug [input-register?]
  (assert (= (interpret bvmax inputs)
              (apply max inputs))
  )
)
```
How does Rosette work

• Take BV as an example  

> synthesize (bvmax, max)

```
(define bvmax
  `((2 bvge 0 1)
    (3 bvneg 2)
    (4 bvxor 0 1)
    (5 bvand 3 4)
    (6 bvxor 1 5))
)
```

```
(define inputs (list 0 -2))
(debug [input-register?]
  (assert (= (interpret bvmax inputs)
    (apply max inputs)))
)
```
How does Rosette work

- Take BV as an example

```python
def bvmax(r0, r1):
    r2 = bvge(r0, r1)
    r3 = bvneg(r2)
    r4 = bvxor(r0, r1)
    r5 = bvand(r3, r4)
    r6 = bvxor(r1, r5)
    return r6
```

```scheme
(define bvmax
  `((2 bvge 0 1)
     (3 bvneg 2)
     (4 bvxor 0 1)
     (5 bvand 3 4)
     (6 bvxor 1 5)))
)
```
References

• Growing Solver-Aided languages with ROSETTE slides: https://excape.cis.upenn.edu/documents/rosette_Emina.pdf
• Github Repository for Rosette: https://github.com/emina/rosette

• Images and code fragments I used in this slides are from the papers and slides above
Thank you!