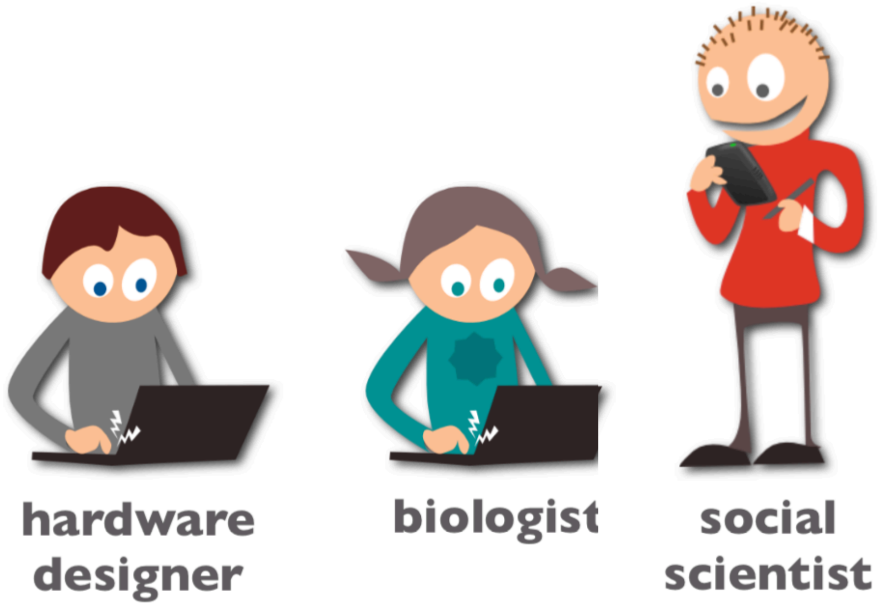
Solver-aided DSL with ROSETTE

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

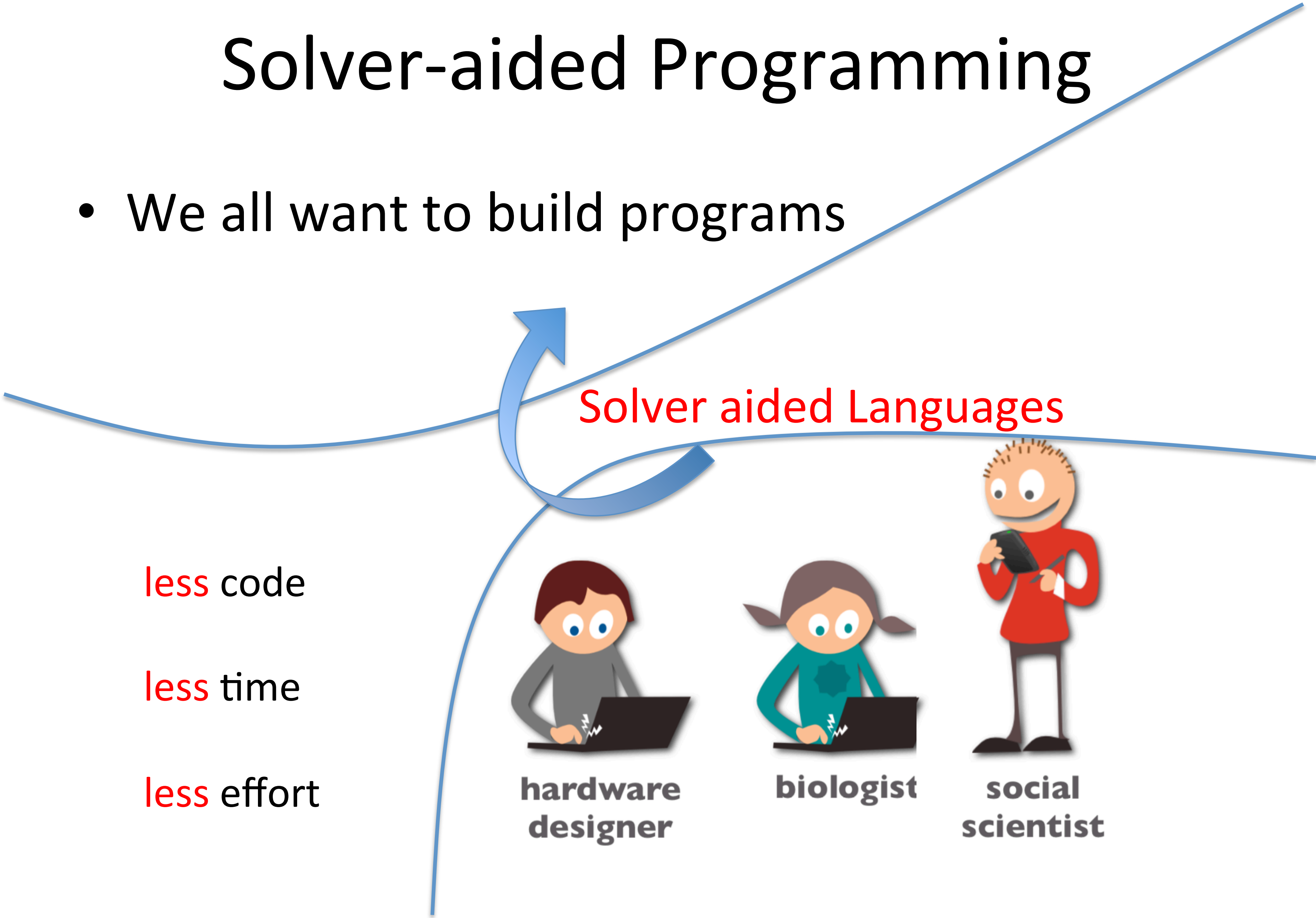
Solver-aided Programming

- 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

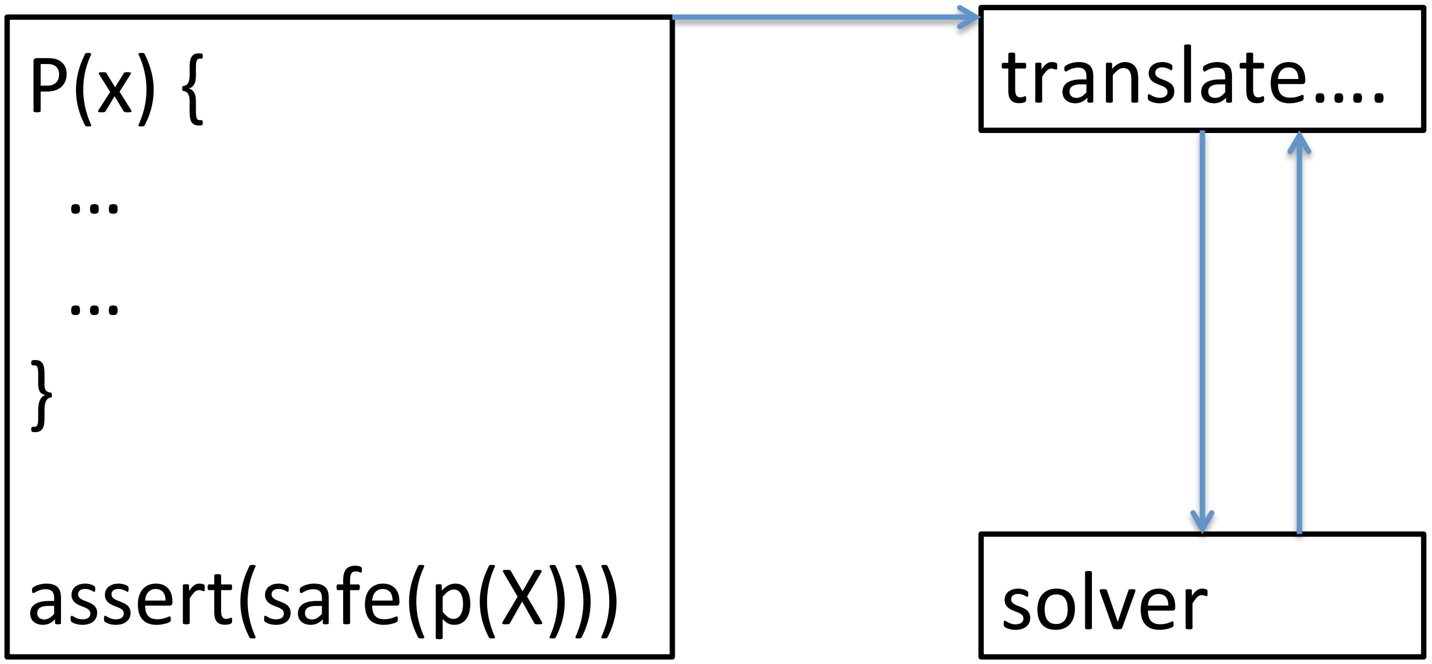
- I have test cases

```
P(x) {
  …
  …
}

assert(safe(p(2)))
```

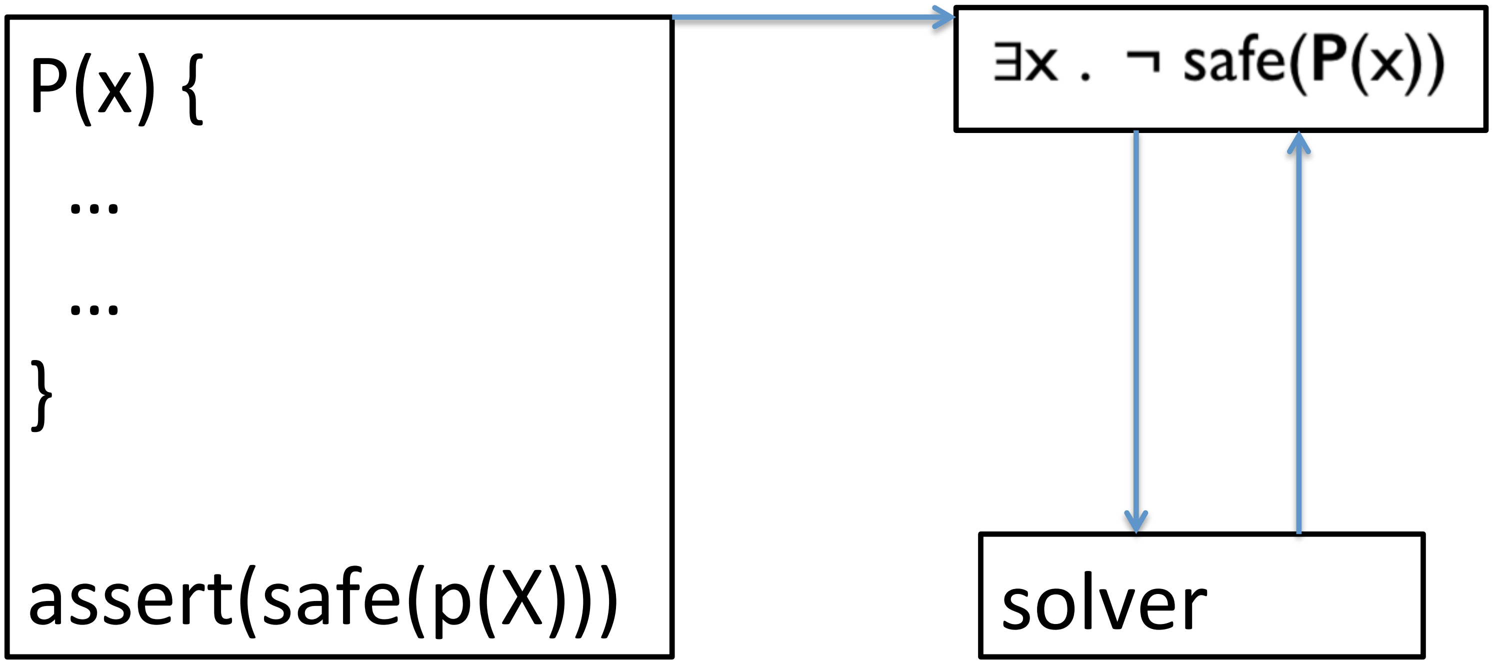
Programming with Solver-aided tools

- I do not have test cases



Programming with Solver-aided tools

- Verification

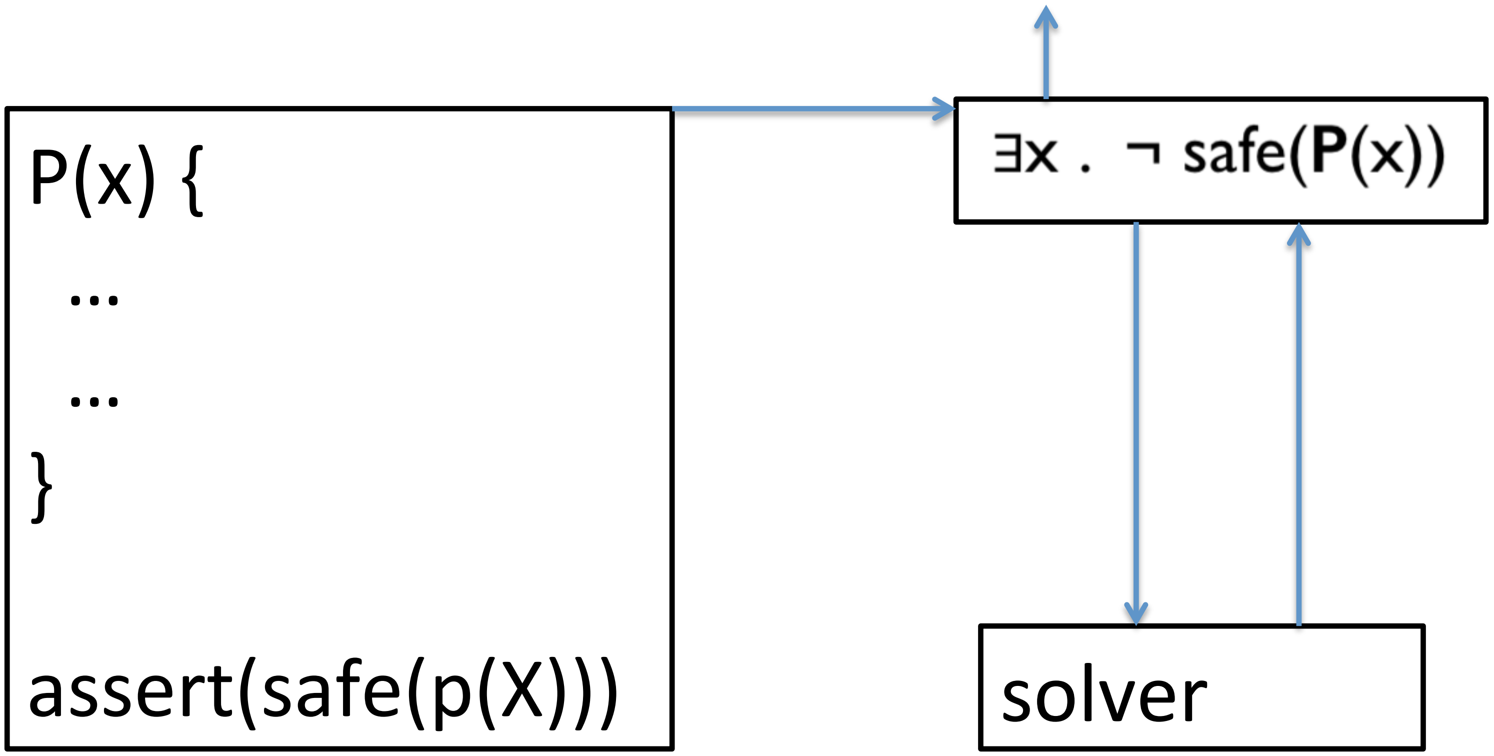


```
P(x) {
  …
  …
}

assert(safe(p(X)))
```

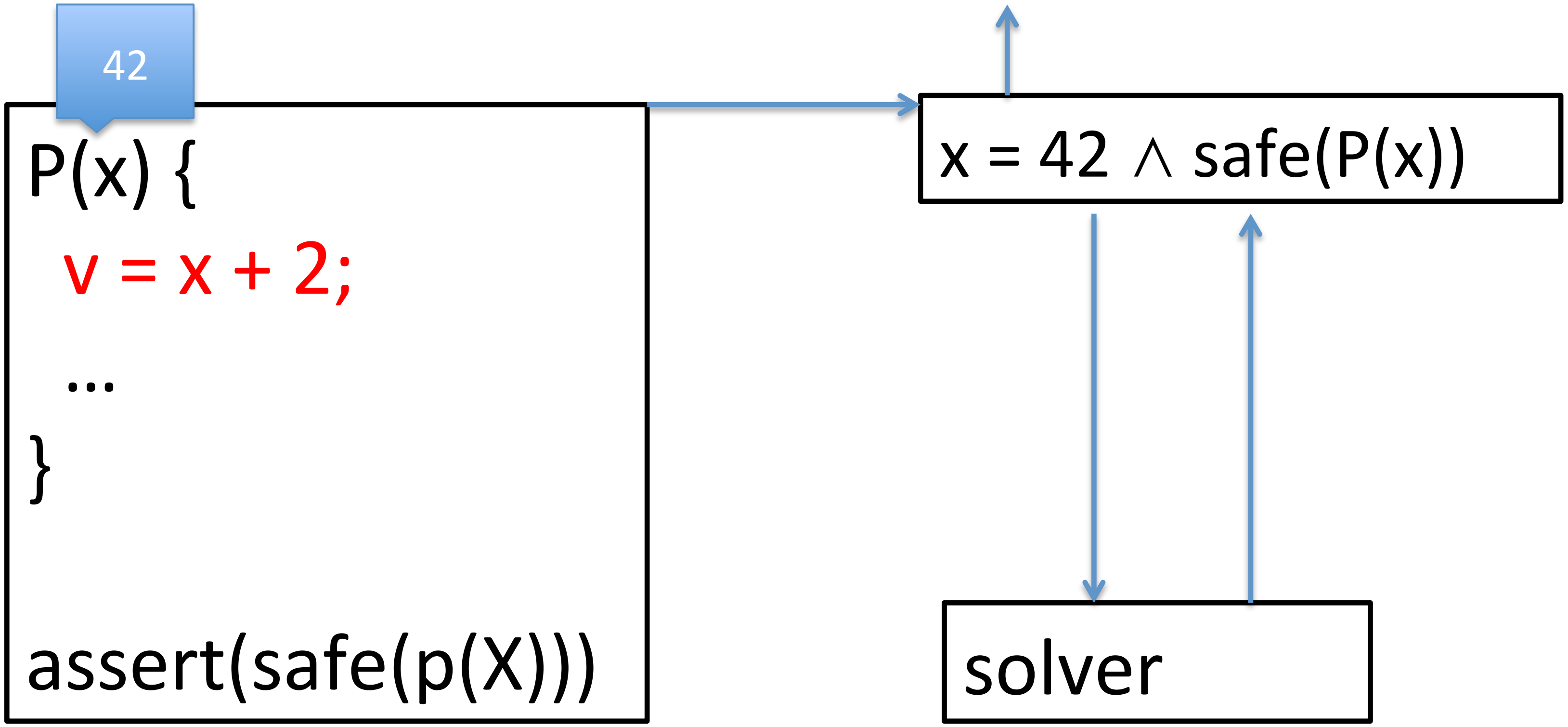
Programming with Solver-aided tools

- Find a value that fails the program



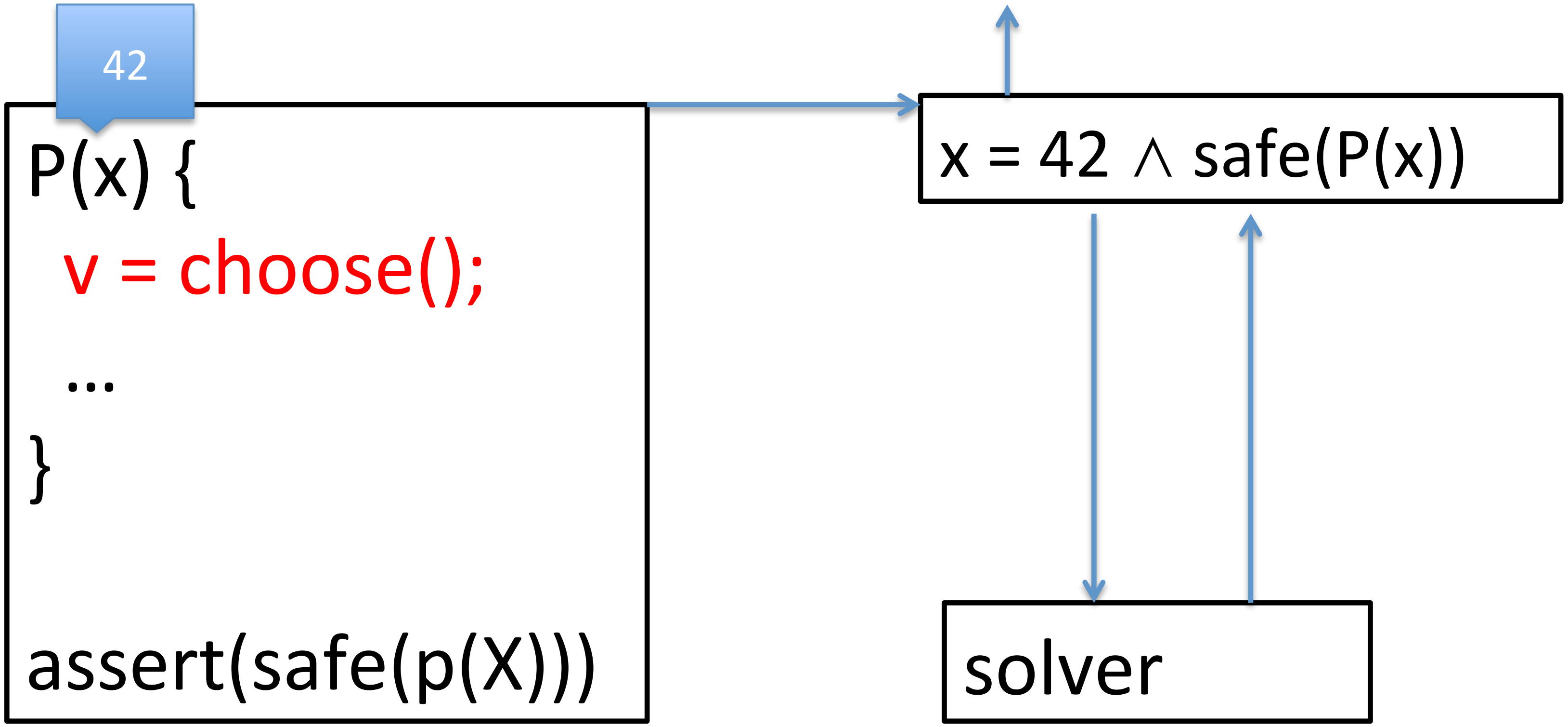
Programming with Solver-aided tools

- Debugging



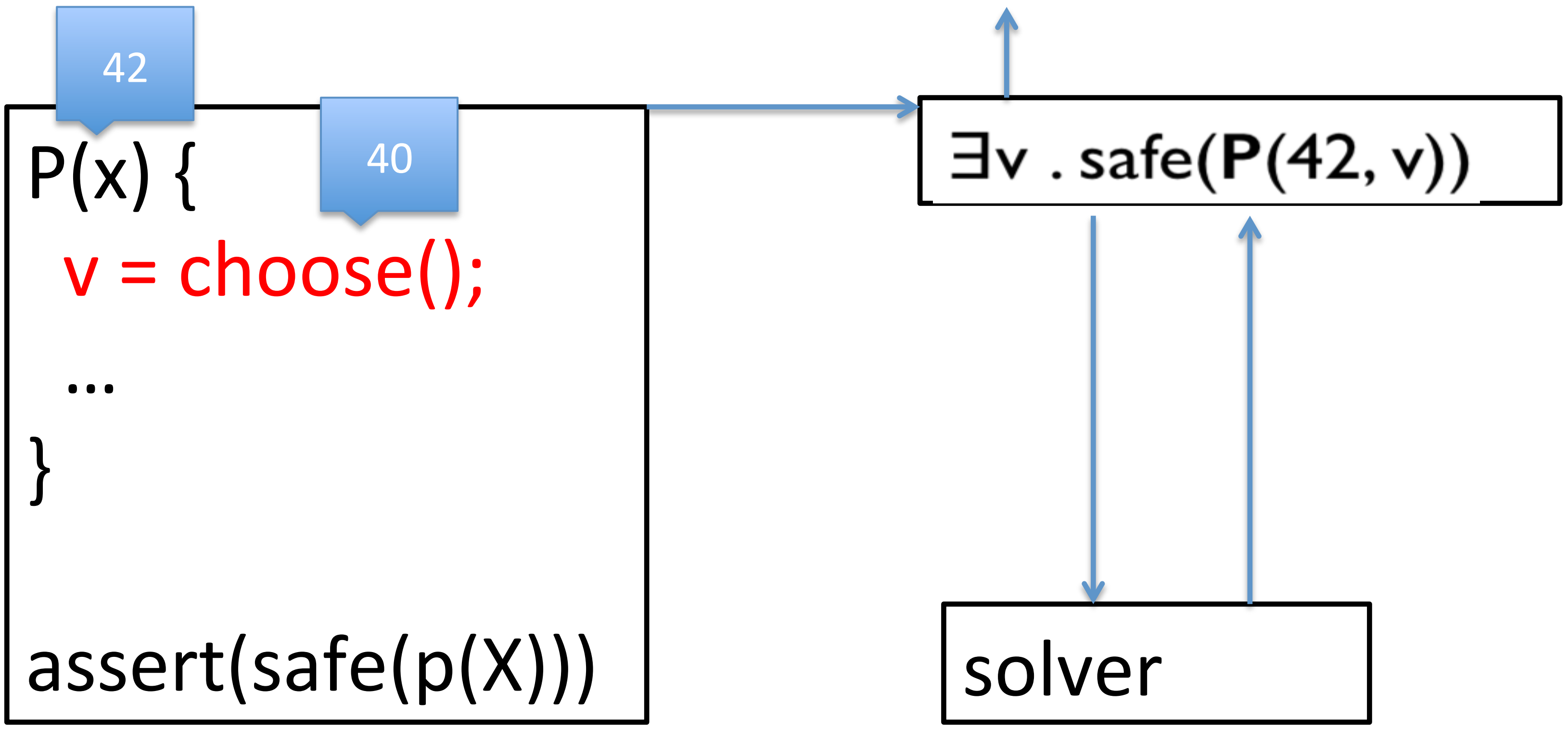
Programming with Solver-aided tools

- Debugging



Programming with Solver-aided tools

- Find the pair that fails the execution

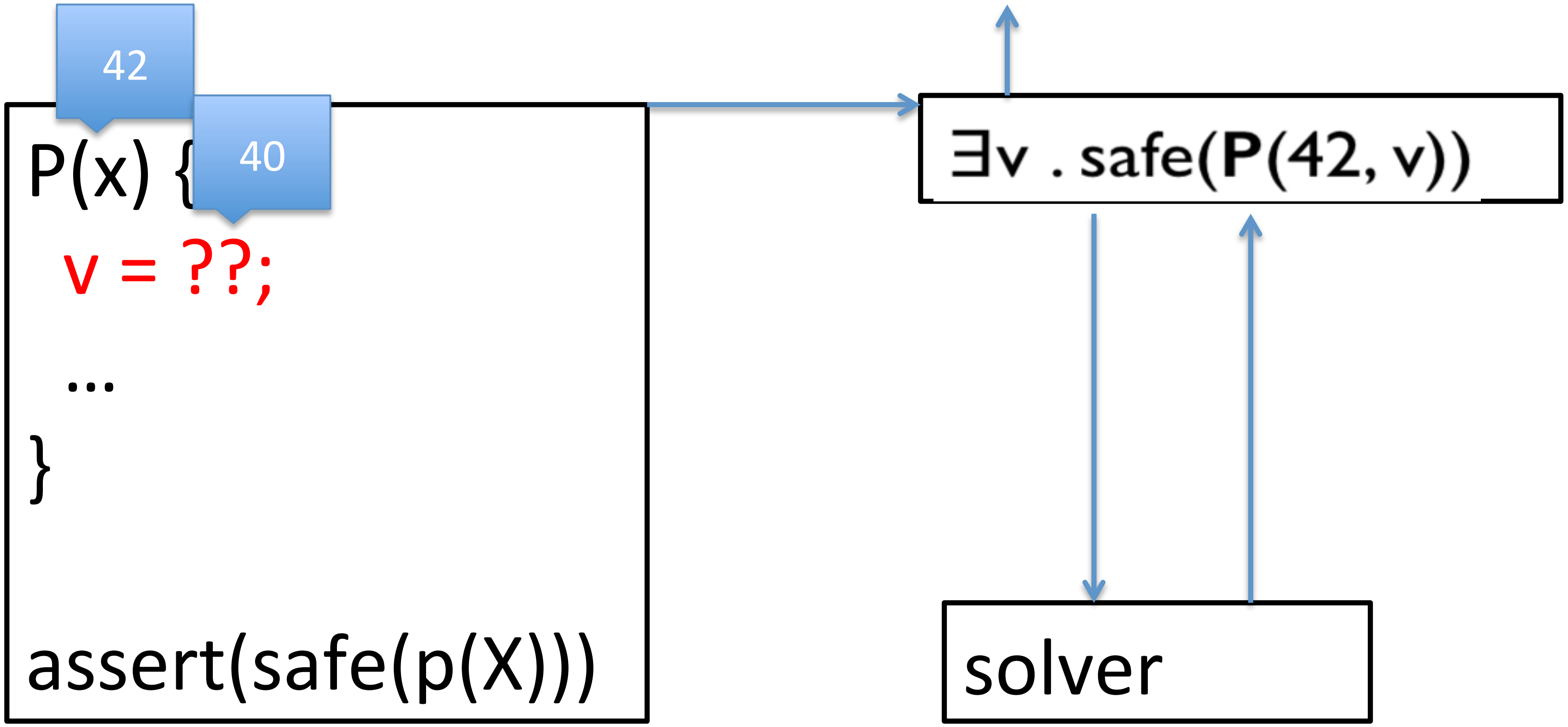


```
P(x) {
  v = choose();
  ...
}

assert(safe(p(X)))
```

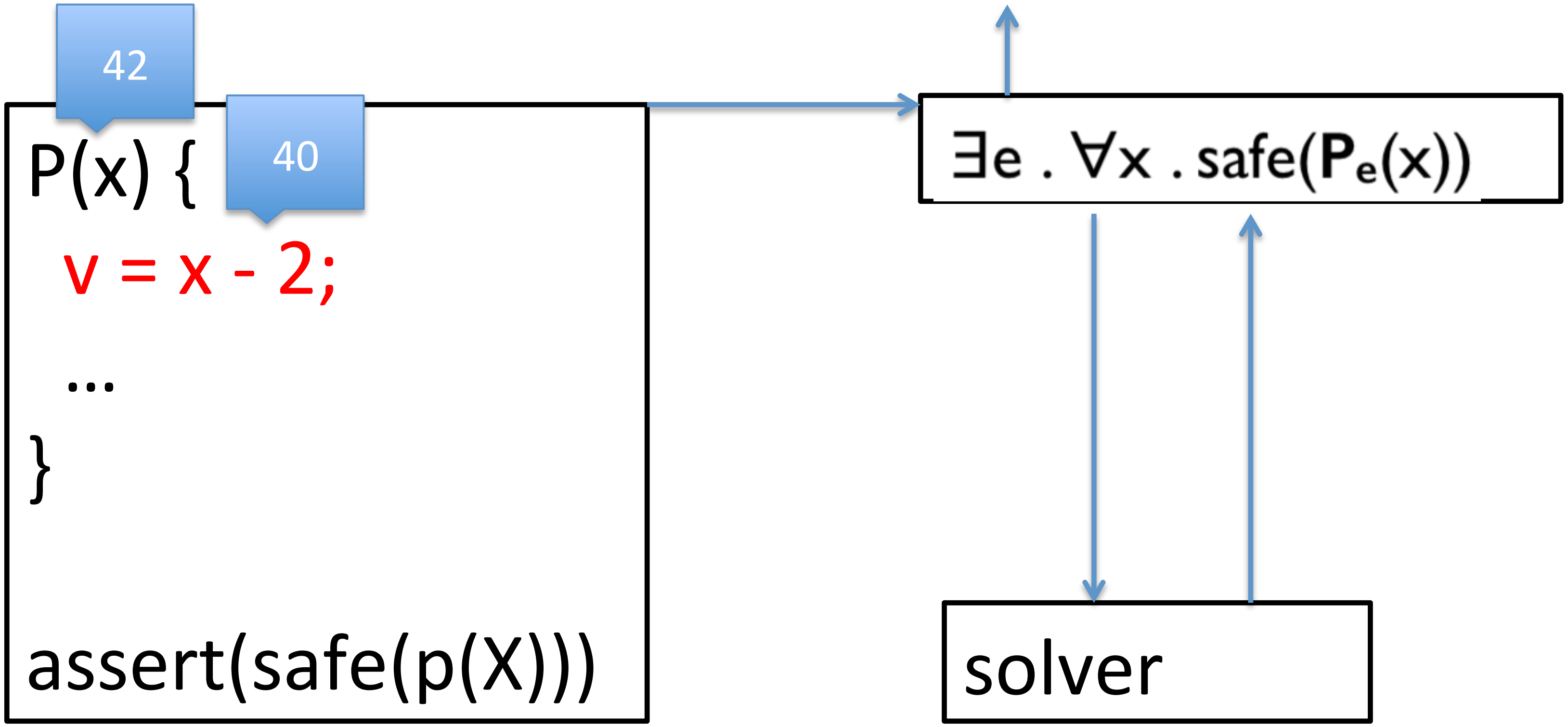
Programming with Solver-aided tools

- Synthesis



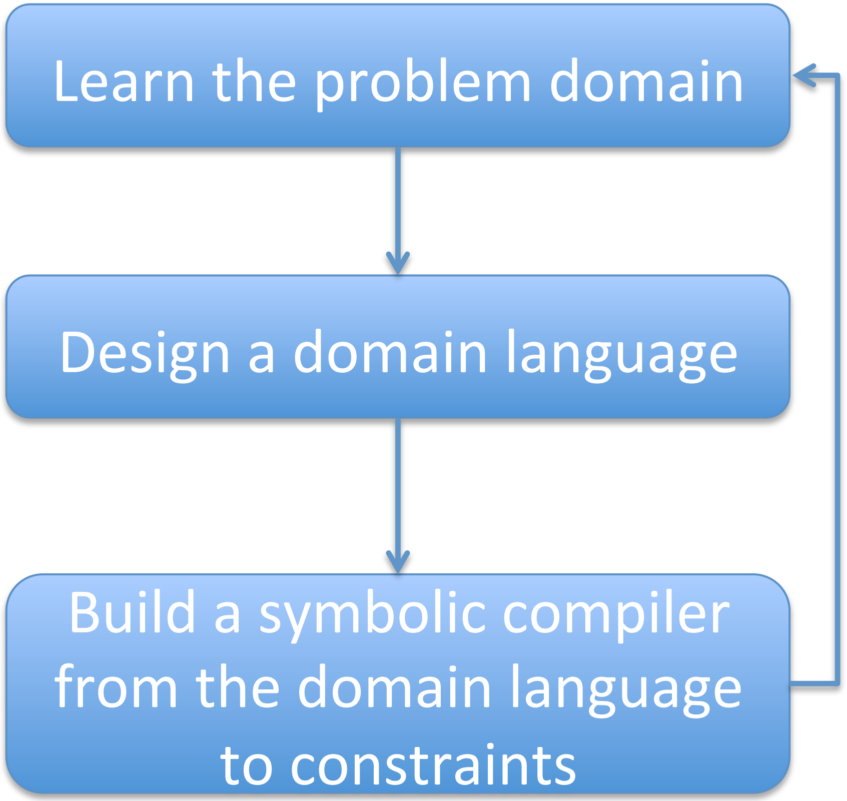
Programming with Solver-aided tools

- Synthesis



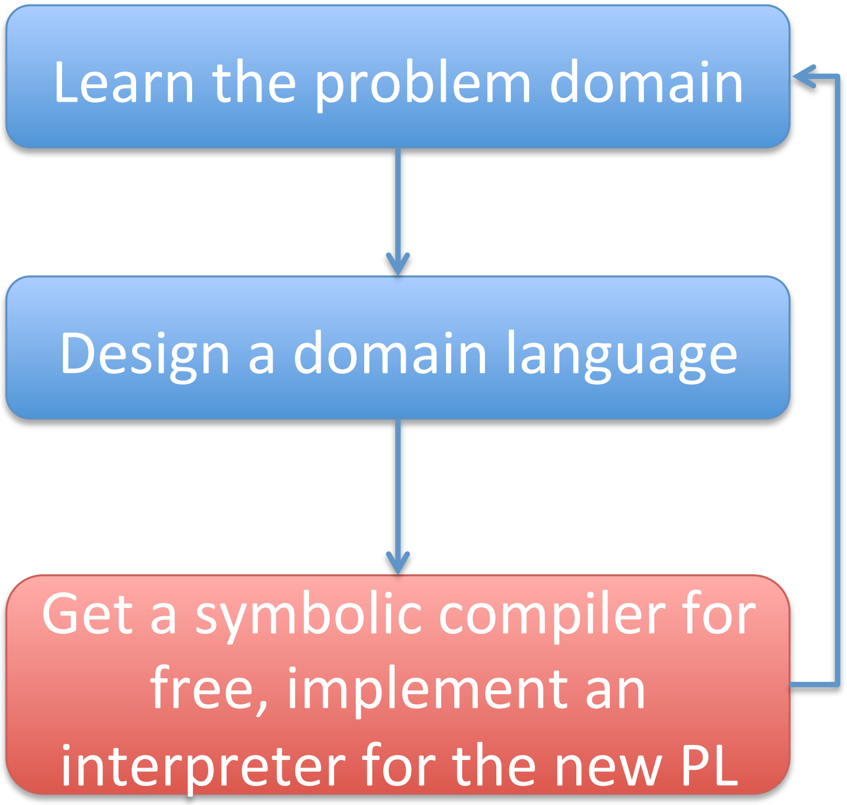
Current Problems

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


Learn the problem domain
Design a domain language
Build a symbolic compiler from the domain language to constraints

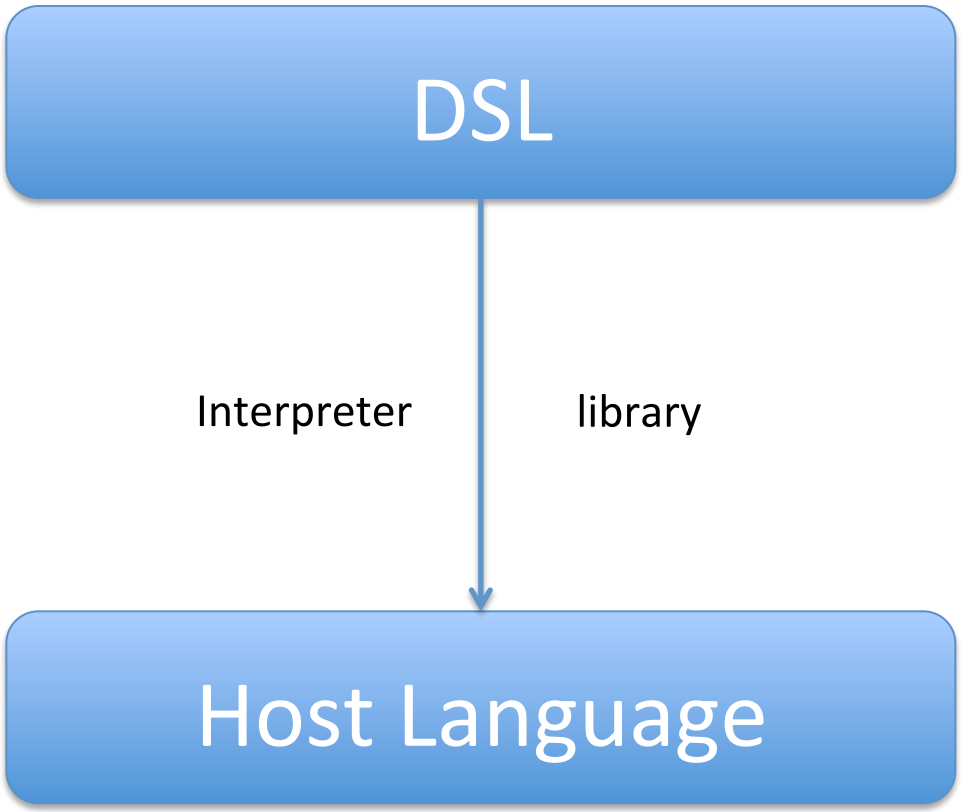
Solution

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


Learn the problem domain
Design a domain language
Get a symbolic compiler for free, implement an interpreter for the new PL

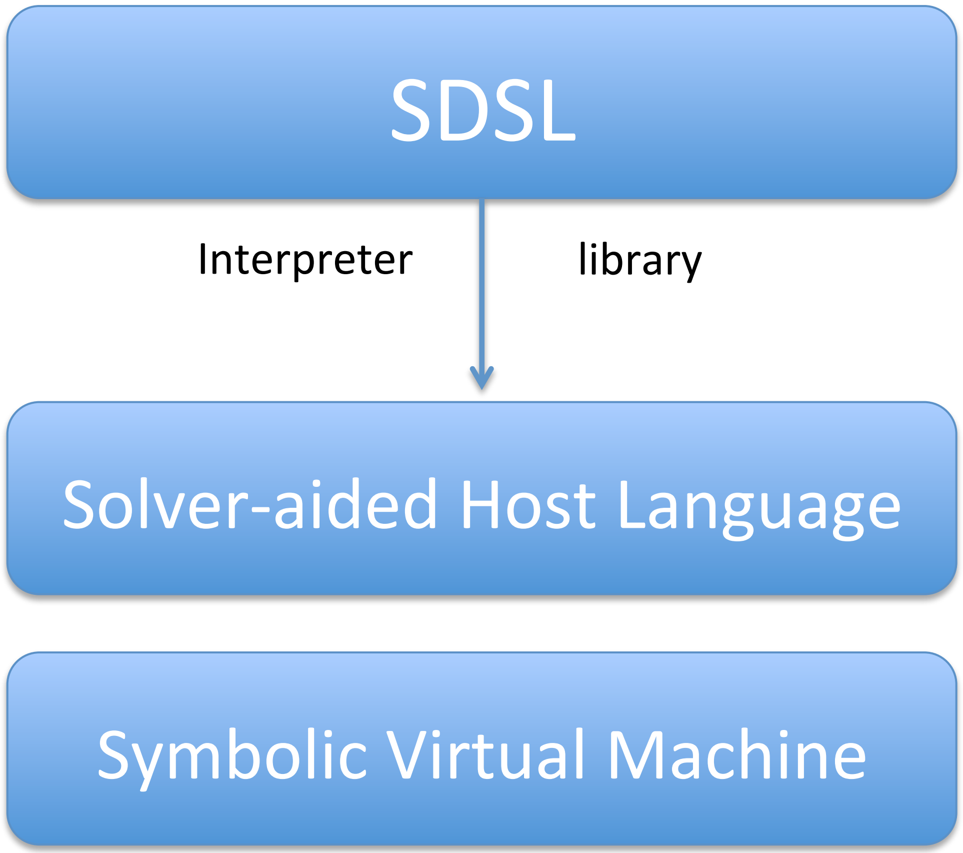
Languages

- Layers



Solver-aided Languages

- Layers



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 compiler the new language to constraints

How does Rosette work

- Take BV as an example

```
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

> bvmax (-1,-2)

```
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
```

```
(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

> bvmax (-1,-2)

```
(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)
)
```

How does Rosette work

- Take BV as an example

0	1	2	3	4	5	6
-2	-1	0	0	-2	0	-1

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

> bvmax (-1,-2)

```
(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)
)
```

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))
)
```

```
(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

> 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

> 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

> synthesize (bvmax, max)

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

```
(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

```
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
```

```
(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
- Growing Solver-Aided languages with ROSETTE paper: http://homes.cs.washington.edu/~emina/pubs/rosette.onward13.pdf
- A Lightweight Symbolic Virtual Machine for Solver-Aided Host Languages: http://homes.cs.washington.edu/~emina/pubs/rosette.pldi14.pdf
- Github Repository for Rosette: https://github.com/emina/rosette
- Programming for everyone: http://fm.csl.sri.com/SSFT14/rosette-lecture.pdf

- Images and code fragments I used in this slides are from the papers and slides above

Thank you !