Advanced Software Security

13. Automatic Verification using CHC

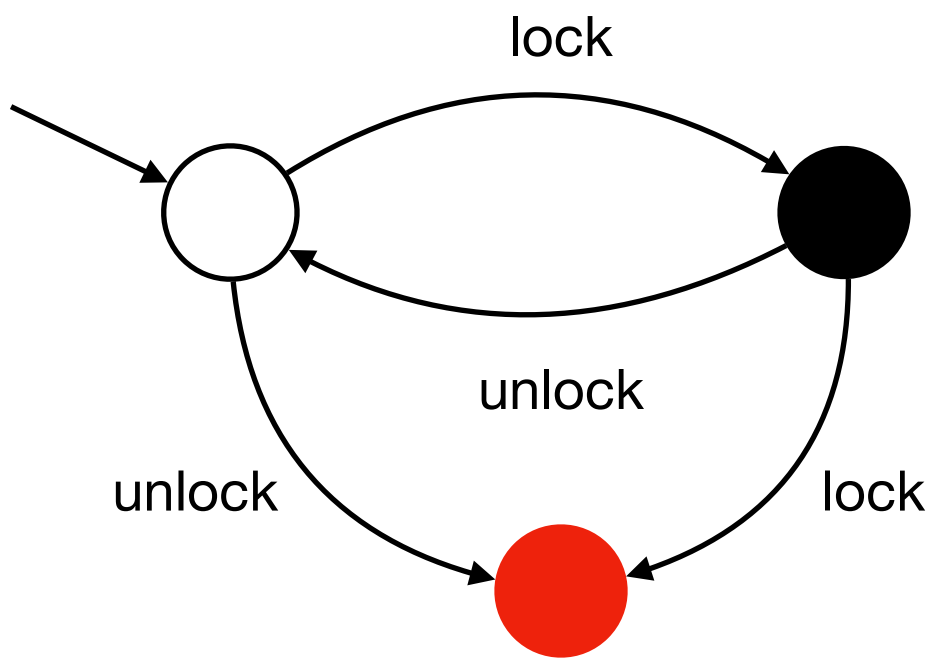
Kihong Heo

KAIST

Towards Fully Automated Verification

- Assumption so far: a user provides inductive invariants
- Fully automated verification: combined with automated invariant generation methods
- Example:
  - Program analysis [CS524]: automatic, terminating, but may not be exact
  - Model checking: automatic, exact, but may not terminate
- This lecture: model checking using Constrained Horn Clause (CHC)

Model Checking



Check: calls to lock and unlock must alternate

- Automatic technique to verify if a model satisfies a specification
  - Model of the target program (finite automata)
  - Specification written in logical formula
  - Verification via reachability
- **Sound and complete with respect to the model**
  - May not terminate
- Example: SLAM (MS Windows device driver verifier)

Horn Clause

- Clause: a disjunction of literals
  - E.g., $p \lor \neg q \lor \neg r$
- Horn clause: a clause with at most one positive literal
  - E.g., $\neg p \lor \neg q \lor r$ which is equivalent to $p \land q \implies r$
- Horn clause logic: basis of logic programming languages such as Prolog and Datalog

Constrained Horn Clause (CHC)

- A fragment of first-order logic

$$\underbrace{\varphi}_{\textbf{Constraint}} \wedge \underbrace{p_1(X_1) \wedge \cdots \wedge p_n(X_n) \implies h(X)}_{\textbf{Datalog rule}}$$

- φ: a constraint in a background theory (e.g., linear)

Example

- Is this CHC formula satisfiable? If so, what is P ?

$$P(0)$$
$$\forall x, x' . P(x) \land x < 10 \land x' = x + 1 \implies P(x')$$

$$\forall x . x \leq 0 \implies P(x)$$
$$\forall x, x' . P(x) \land x < 5 \land x' = x + 1 \implies P(x')$$

Program Verification via CHC

- Given a program and a specification, generate verification conditions using CHC
- Check the satisfiability of the CHC formula
  - E.g., Z3

Language

- Program = control flow graph

- Node = basic block = list of commands (end with jump)

$$
\begin{aligned}
C &\rightarrow \texttt{skip} \mid x := E \mid x := \texttt{input}() \mid \texttt{br}\ B\ l_1\ l_2 \\
&\mid\ \texttt{goto}\ l \mid \texttt{assume}(E) \mid \texttt{assert}(E) \\
E &\rightarrow n \mid x \mid E + E \mid E - E \mid E \times E \mid E\ /\ E \\
B &\rightarrow \mathsf{true} \mid \mathsf{false} \mid E < E \mid E = E \mid \neg B
\end{aligned}
$$

- Example:

```
Entry:
  x := input()
  assume(x > 1)
  assert(x == 0)
```

```
Entry:
  x := input()
  y := x - 1
  br x / 2 != 0 L1 L2
L1:
  assert(y != 0)
L2:
  skip
```

Specification

- Annotated in programs using assertions
- Checking an assertion = checking a reachability
  - Assertion is false = error state is reachable
- Example

```
Entry:
  x := input()
  y := x - 1
  assert(y != 0)
```

```
Entry:
  x := input()
  y := x - 1
  br y != 0 L1 L2
L1:
  skip
L2:
  assert false
```

State

- A relation (predicate) parameterized by values of variables defined so far
    - One relation per basic block
- Example

```
Entry:
  x := input()
  y := x - 1
  br y != 0 L1 L2
L1:
  skip
L2:
  assert false
```

Relations: Entry, L1(x, y), L2(x, y)

Reachable states:
Entry,
L1(2, 1), L1(3,2), L1(4, 3), …
L2(1, 0)

Verification Condition

- CHC formula: the relationship between all nodes + unreachability of the error node
- Loop invariants will be computed by the underlying solver (But not always! Why?)
- Example

```
Entry:
  x := input()
  y := x - 1
  assert(y != 0)
```

⬇

```
Entry:
  x := input()
  y := x - 1
  br y != 0 L1 L2
L1:
  skip
L2:
  assert false
```

The condition is SATISFIABLE
iff
there exists an erroneous input

$$
\begin{array}{l}
Entry \\
\forall x, y \,.\, Entry \land y = x - 1 \land y \neq 0 \implies L_1(x, y) \\
\forall x, y \,.\, Entry \land y = x - 1 \land y = 0 \implies L_2(x, y) \\
\exists x, y \,.\, L_2(x, y)
\end{array}
$$

Example

```
x := input();
assume(x < 10);
while(x < 10) {
  x++;
}
assert(x == 0);
```

```
Entry:
  x0 := input()
  assume(x < 10)
  goto Cond
Cond:
  x1 := ϕ [x0, Entry] [x2, Body]
  br (x1 < 10) Body End
Body:
  x2 := x1 + 1
  goto Cond
End:
  br (x1 = 10) Then Else
Then
  skip
Else:
  assert false
```

The condition is SATISFIABLE
iff
there exists an erroneous input

$$Entry$$
$$\forall x . Entry \wedge x < 10 \implies Cond(x)$$
$$\forall x . Cond(x) \wedge x < 10 \implies Body(x)$$
$$\forall x . Cond(x) \wedge x >= 10 \implies End(x)$$
$$\forall x . Body(x) \wedge x' = x + 1 \implies Cond(x')$$
$$\forall x . End(x) \wedge x = 10 \implies Else(x)$$
$$\exists x . Else(x)$$

Summary

- Model checking: Automatically check if a model satisfies a specification
  - I.e., reachability of error states
- Constrained Horn clause: a fragment of FOL
- Program verification using CHC
  - Verification condition = unreachability of error states
- Automatically solved by theorem provers