Test Case Generation for Ultimately Periodic Paths

Saddek Bensalem, Doron Peled, Hongyang Qu, Stavros Tripakis, Lenore Zuck

Haifa Verification Conference, 2007

Outline

- Background (flow charts, path preconditions)
- Conditions for ultimately periodic paths
- Test case generation methodology
- Implementation
- Conclusion

Flow Charts

- A graphical representation of structure of a program
- Three kinds of nodes
  - Ellipse (beginning, end)
  - Box (assignment)
  - Diamond (condition)
- Two kinds of edges
  - Outgoing from ellipse or box nodes (no labels)
  - Outgoing from diamond nodes (labelled as *yes* or *no*)

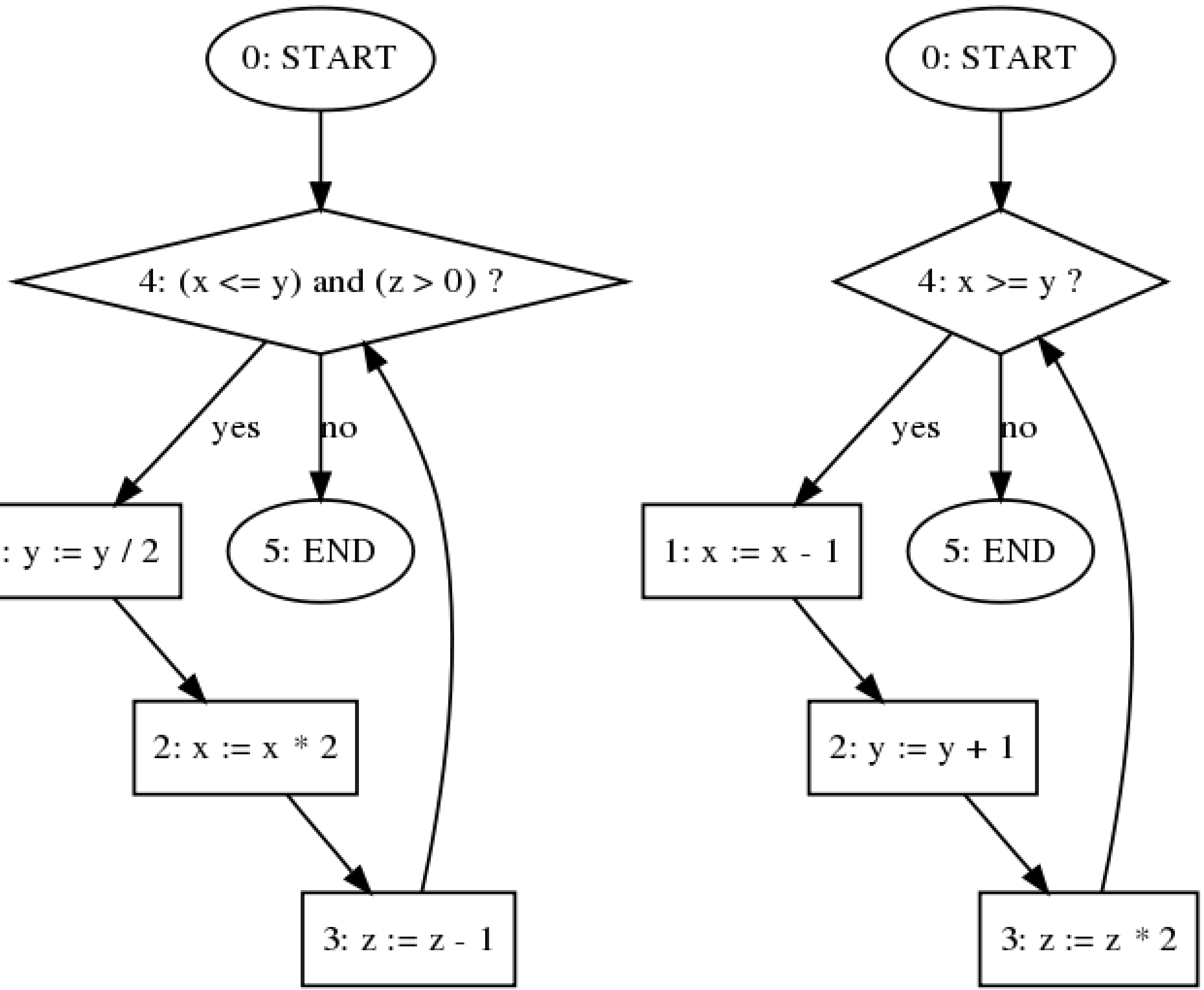
An example

- Program 1

```
while (x<=y && z>0) {
  y := y / 2;
  x := x * 2;
  z := z – 1;
}
```

- Program 2

```
while (x>=y) {
  x := x - 1;
  y := y + 1;
  z := z * 2
 }
```



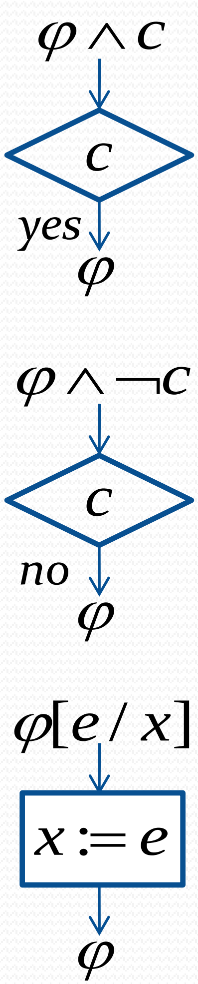
Path conditions

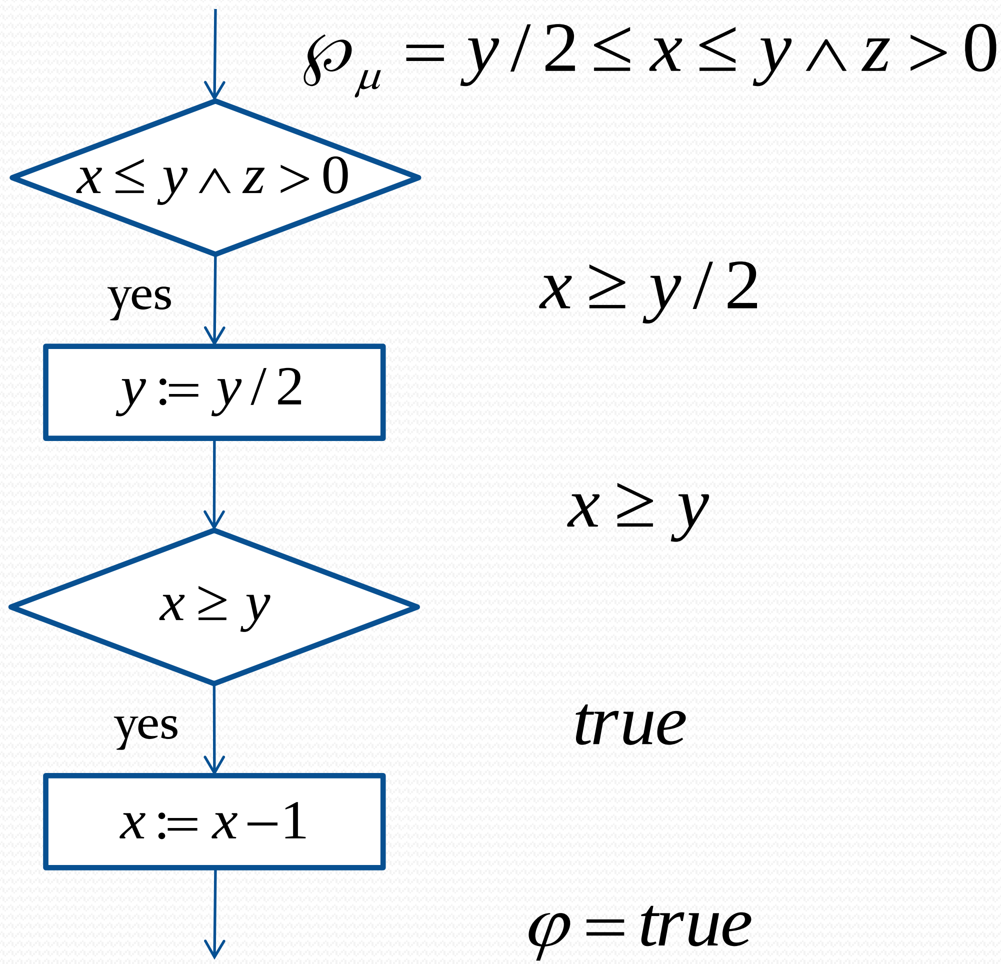
- A path condition $\wp_\mu(\varphi)$ is a first order predicate that expresses the condition to execute the path μ and satisfy the predicate φ at the end of the execution.

- Sometime we write $\wp_\mu$ for $\wp_\mu(true)$.

Computing path conditions

1.
2.
3.





Properties of path conditions

- Compositionality

$$\wp_{\sigma\rho}(\varphi) = \wp_{\sigma}(\wp_{\rho}(\varphi))$$

- Distribution over conjunction

$$\wp_{\mu}(\varphi \wedge \psi) = \wp_{\mu}(\varphi) \wedge \wp_{\mu}(\psi)$$

- Monotonicity

$$\text{if } \varphi \rightarrow \psi \text{ then } \wp_{\mu}(\varphi) \rightarrow \wp_{\mu}(\psi)$$

How to calculate a path condition for an ultimately periodic path?

- This is the subject of this work.
- In general this is an undecidable problem.

Some conditions for ultimately periodic paths

- **Equality condition**

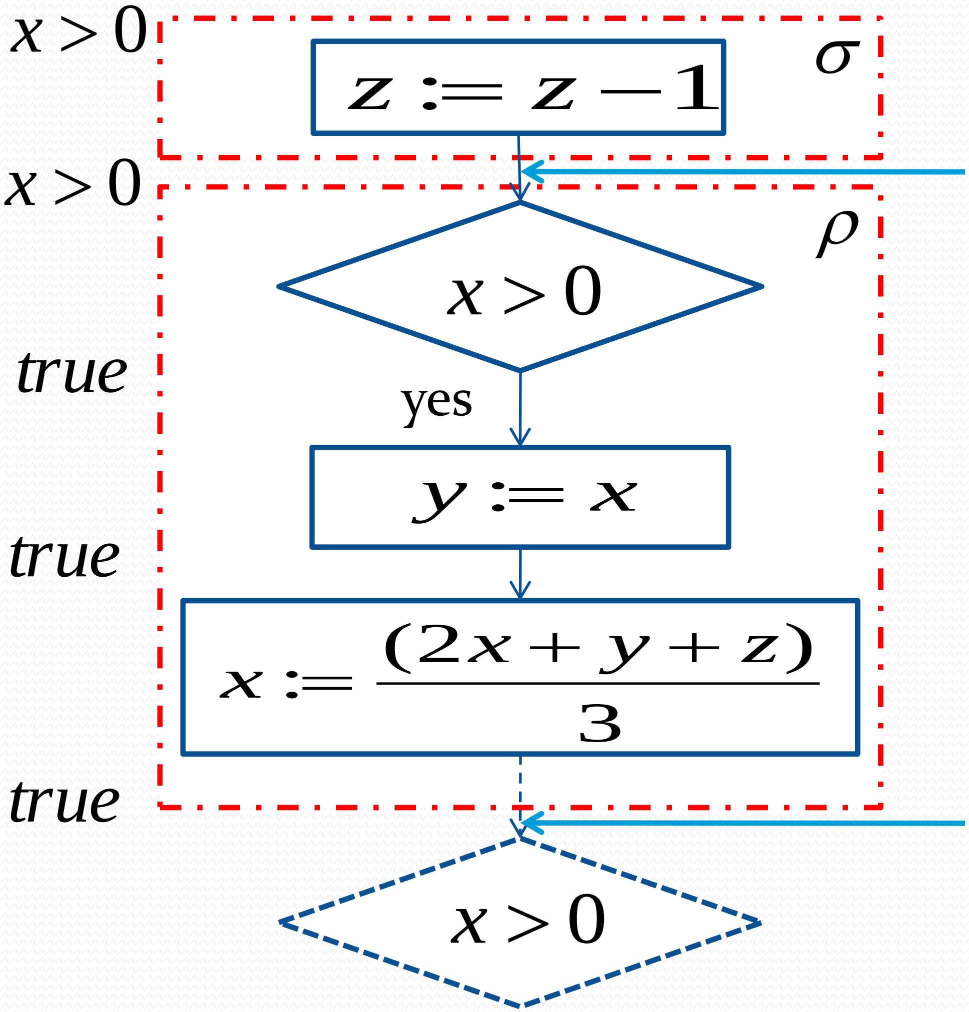
  computed using equality method
- **Monotonicity condition**

  computed using monotonicity method
- Condition for not completely ultimately periodic paths

Equality method

- We are looking for the condition to execute a loop ρ *indefinitely*, after a finite prefix σ, where in each iteration, the variables obtain *the same values*.
- Executing the periodic part ρ once when $\wp_\rho \wedge X = tr_\rho(X)$.
- Executing it after the prefix σ is when $\wp_{\sigma\rho} \wedge \wp_\sigma(\wp_\rho \wedge X = tr_\rho(X))$.
- Simplifying: $\wp_{\sigma\rho} \wedge \wp_\sigma(X = tr_\rho(X))$.

Example (1)



(x, y, z)

$$x = \frac{3x + z}{3} \Rightarrow z = 0$$

$$y = x \Rightarrow y = x$$

$$\wp_{\sigma}(z = 0 \wedge y = x) \equiv z - 1 = 0 \wedge y = x$$

$$\wp_{\sigma\rho}(true) \equiv x > 0$$

$$x > 0 \wedge z - 1 = 0 \wedge y = x$$

$(\frac{3x + z}{3}, x, z)$

Monotonicity Method

- It is sufficient to find a loop invariant such that $I \to \wp_\rho(I)$.
- The weakest such invariant I is $I = \wp_\rho(true)$.
- Proof:

  $I \to true$ for each I.

  By monotonicity of $\wp$, $\wp_\rho(I) \to \wp_\rho(true)$.

  Since $I \to \wp_\rho(I)$, it holds that $I \to \wp_\rho(true)$, independently of I.

Deriving an ultimately periodic condition

- We set $I = \wp_\rho(true)$ in the implication $I \rightarrow \wp_\rho(I)$, obtaining $\wp_\rho(true) \rightarrow \wp_\rho(\wp_\rho(true))$.

- This can be rewritten as $\wp_\rho(true) \rightarrow \wp_\rho(true)[tr_\rho(X)/X]$.

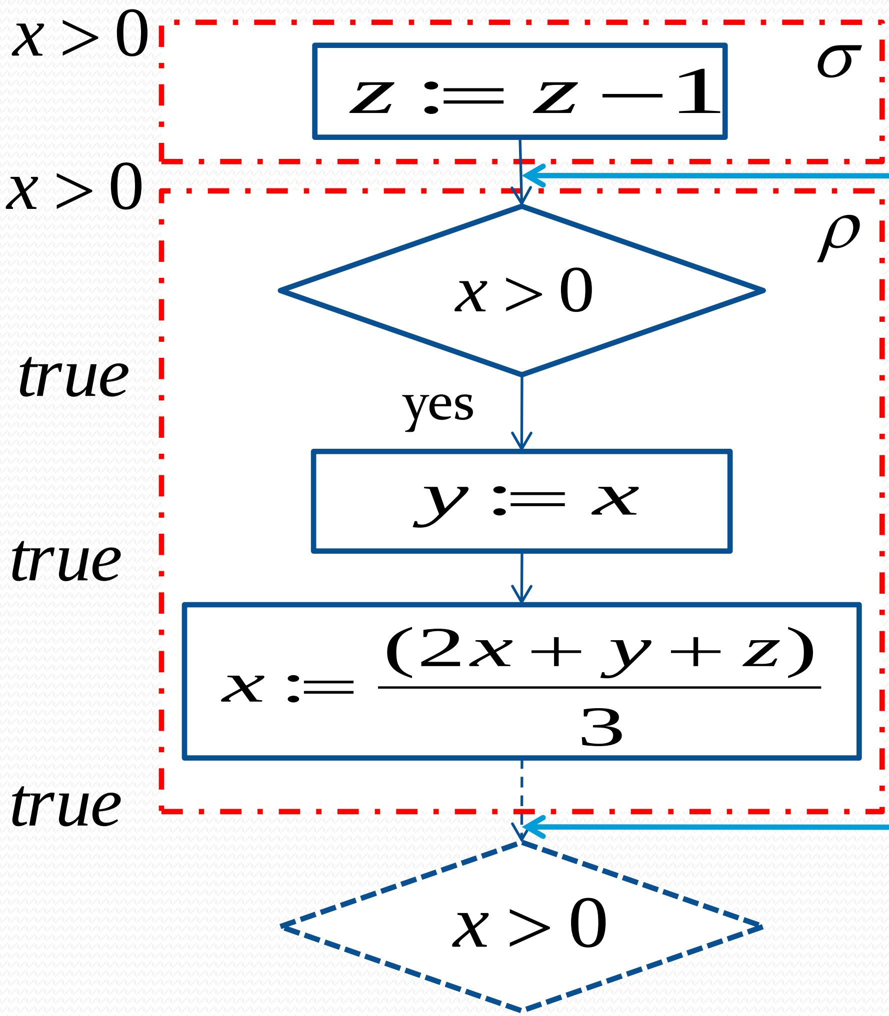
- Applying the $\wp$ of the prefix, we obtain $\wp_\sigma(\wp_\rho(true)) \rightarrow \wp_\sigma(\wp_\rho(true)[tr_\rho(X)/X])$.

- The next slide will deal with the 2nd bullet (and then we need to remember to apply the 3rd).

The case where $\wp_\rho(true)$ is $e \geq 0$ (or $e>0$)

- Set $e' = e[tr_\rho(X)/X]$.
- Bullet 2 from previous slide becomes $e \geq 0 \rightarrow e' \geq 0$.
- A *sufficient* condition is $e' \geq e$.
- Other cases: when we have a condition $\wp_\rho(true) \equiv g \geq f$, we take $e = g - f$.
- **Conjunction principle**: In case $\wp_\rho(true) \equiv g \geq 0 \wedge f \geq 0$, we have condition $g' \geq g \wedge f' \geq f$.
- **Disjunction principle**: In case $\wp_\rho(true) \equiv g \geq 0 \vee f \geq 0$, it is sufficient that we strengthen to either $g' \geq g$ or $f' \geq f$.
- An equality can be transformed into two inequalities and the disjunction case is applied.

Example (2)



$$\wp_\rho(true) \equiv x > 0 \Rightarrow \frac{3x+z}{3} > x \Rightarrow z > 0$$

$$(x, y, z)$$

$$\wp_\sigma(\omega_\rho) \to \wp_\sigma(\wp_\rho[tr_\rho(X)/X]) \equiv$$

$$x > 0 \wedge z > 1$$

$$\left(\frac{3x+z}{3}, x, z\right)$$

Some mixed and not completely ultimately periodic paths

```
While x>1 do
 begin
      if PowerTwo(x-1) then
              x:=4*(x-1)
      else
              x:=x-1
  end.
```

Example: 4→3→8→7→6→5→16→15…

Computing the condition

- Shrinking the loo body to a new transition *t:*

$$\wp_{\sigma t}(\varphi) = \bigvee_i (c_i \wedge \wp_{\sigma}(\varphi)[\overline{e}_i / \overline{x}_i])$$

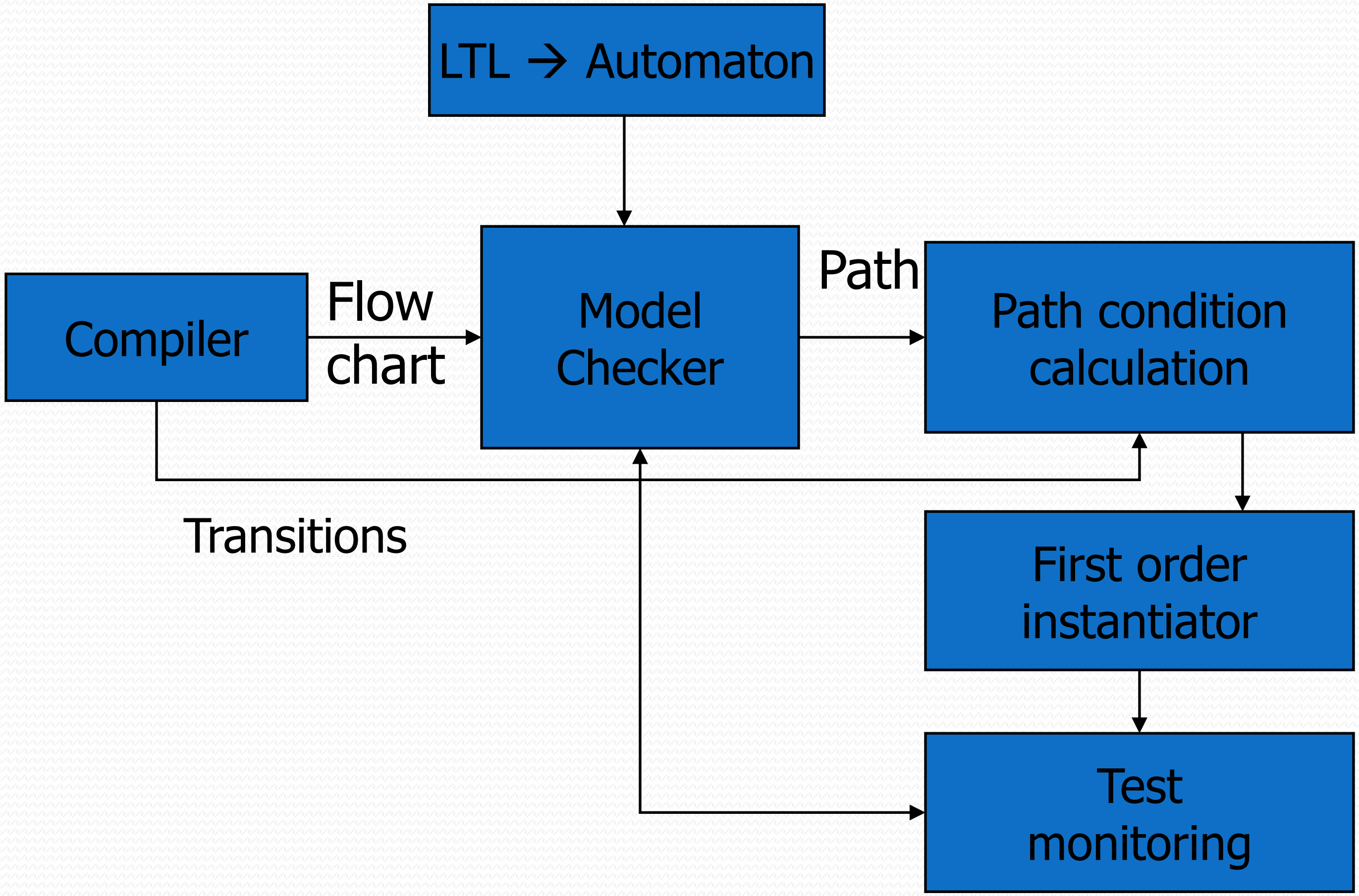
- **Example**:

$$t : \mathrm{PowerTwo}(x-1) \mapsto x := 4(x-1) \oplus \neg \mathrm{PowerTwo}(x-1) \mapsto x := x-1$$

$$\wp_{\sigma t} = (\mathrm{PowerTwo}(x-1) \wedge 4(x-1) > 1) \vee (\neg \mathrm{PowerTwo}(x-1) \wedge x-1 > 1)$$

$$\wp_{\sigma t} \to x > 1$$

Test case generation



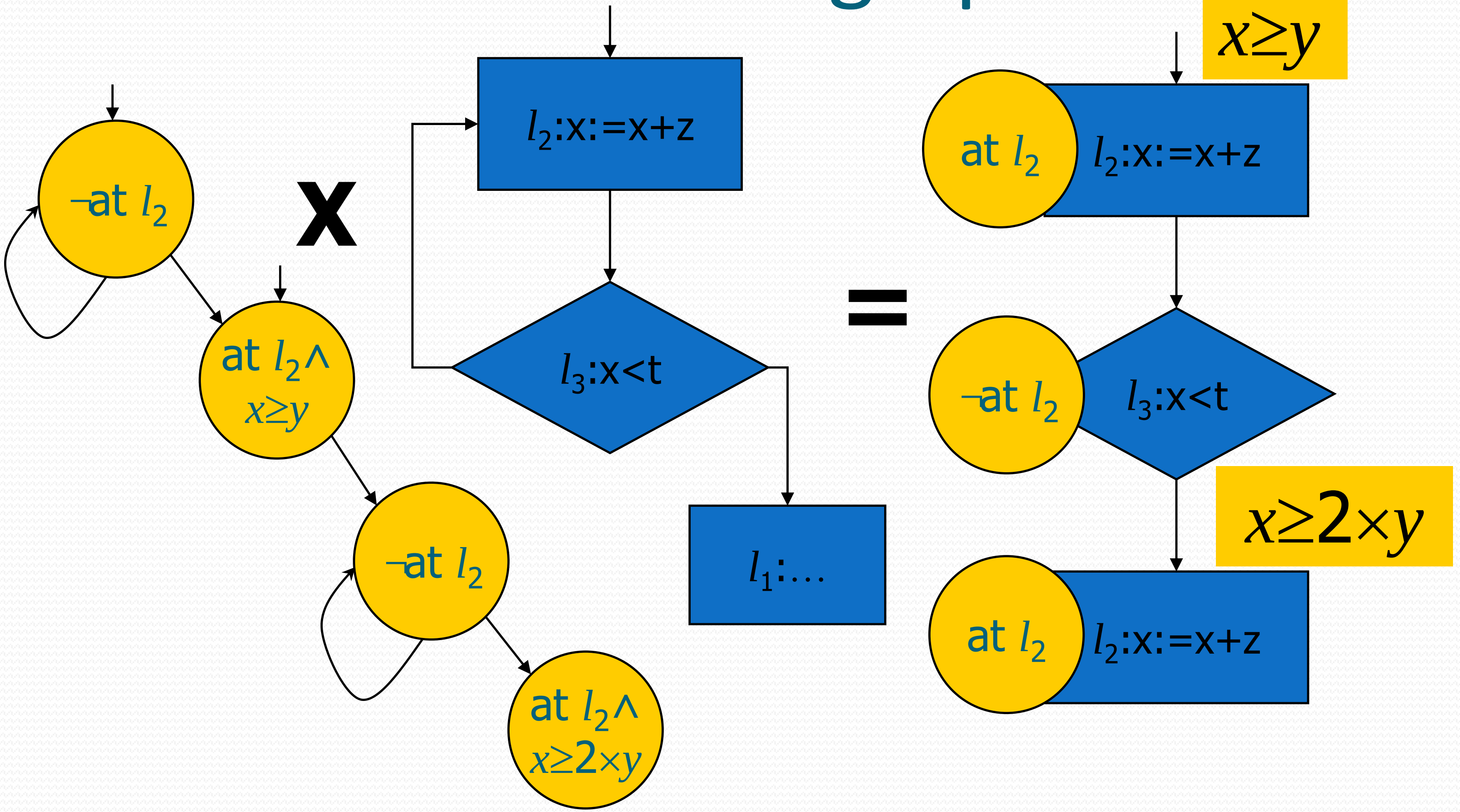
Goals

- Verification of *software*.
- Compositional verification. *Use only a unit of code* instead of the whole code.
- *Parameterized verification*. Verifies a procedure with any value of parameters in "one shot"
- Generating test cases via *path conditions*: A truth assignment satisfying the path condition. Helps derive the demonstration of errors.
- Generating appropriate values to missing parameters.

How to generate test cases

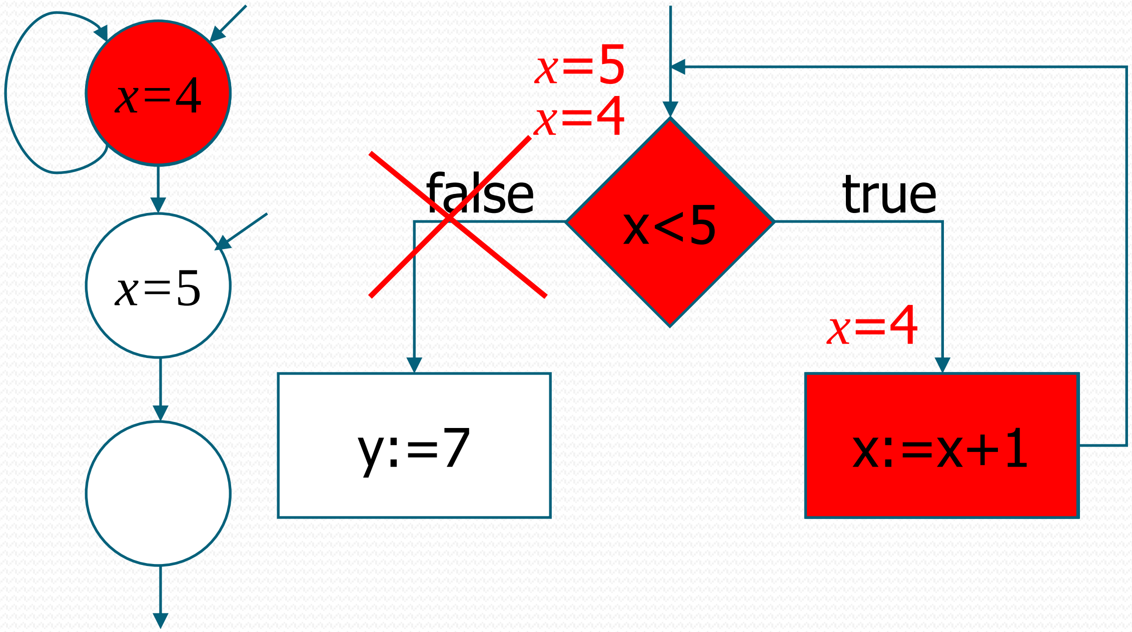
- Take the intersection of an LTL automaton with the flow graph.
  Some paths would be eliminated for not satisfying the assertions on the program counters.
- Seeing same flow chart node does not mean a loop: program variables may value. Use *iterative deepening.*
- For each initial path calculate the path condition. Backtrack if condition simplifies to false.
- Report path condition based on flow graph path+LTL assertions.
- Always simplify conditions!

intersection of the property automaton and flow graph



How the LTL formula directs the search

- Spec: $(x=4)U(x=5 \wedge \mathrm{O}\ldots)$



Implementation

- Implemented in Java
- Using *Mathematica* to simplify conditions.
- Detecting identical states
- Heuristic match

Conclusion

- An approach for generating test cases automatically.
- Also: verification of infinite state systems.
- Path by path verification rather than state by state.
- Challenge: the weakest precondition for ultimately periodic sequences in infinite state systems.
- We suggested several methods (e.g., the equality and monotonicity methods, etc.)
- Not all of the infinite executions are ultimately periodic.