# Corpus based approach in Constraint solving

Kihong Heo

Programming Research Laboratory
Seoul National University

# Reference

- S. Gulwani, S. Srivastava, and R. Venkatesan, Program analysis as constraint solving, PLDI'08

# Contents

# Motivation

- How is the corpus-based (or enumeration based) approach used well in constraint solving?

# Background

- Discovering program invariant
  - fixed-point computation based approach (e.g. abstract interpretation)
  - constraint-based invariant generation approach
    - program ⟶ satisfiability constraints

# Background

- Two advantages of Constraint-based approach
  - more efficient
  - more precise

# Constraint solving

- Applications
  - program verification
  - strongest postcondition generation
  - weakest precondition generation

# Program verification

- Goal
  - verifying  assertions in program are valid

# Program verification

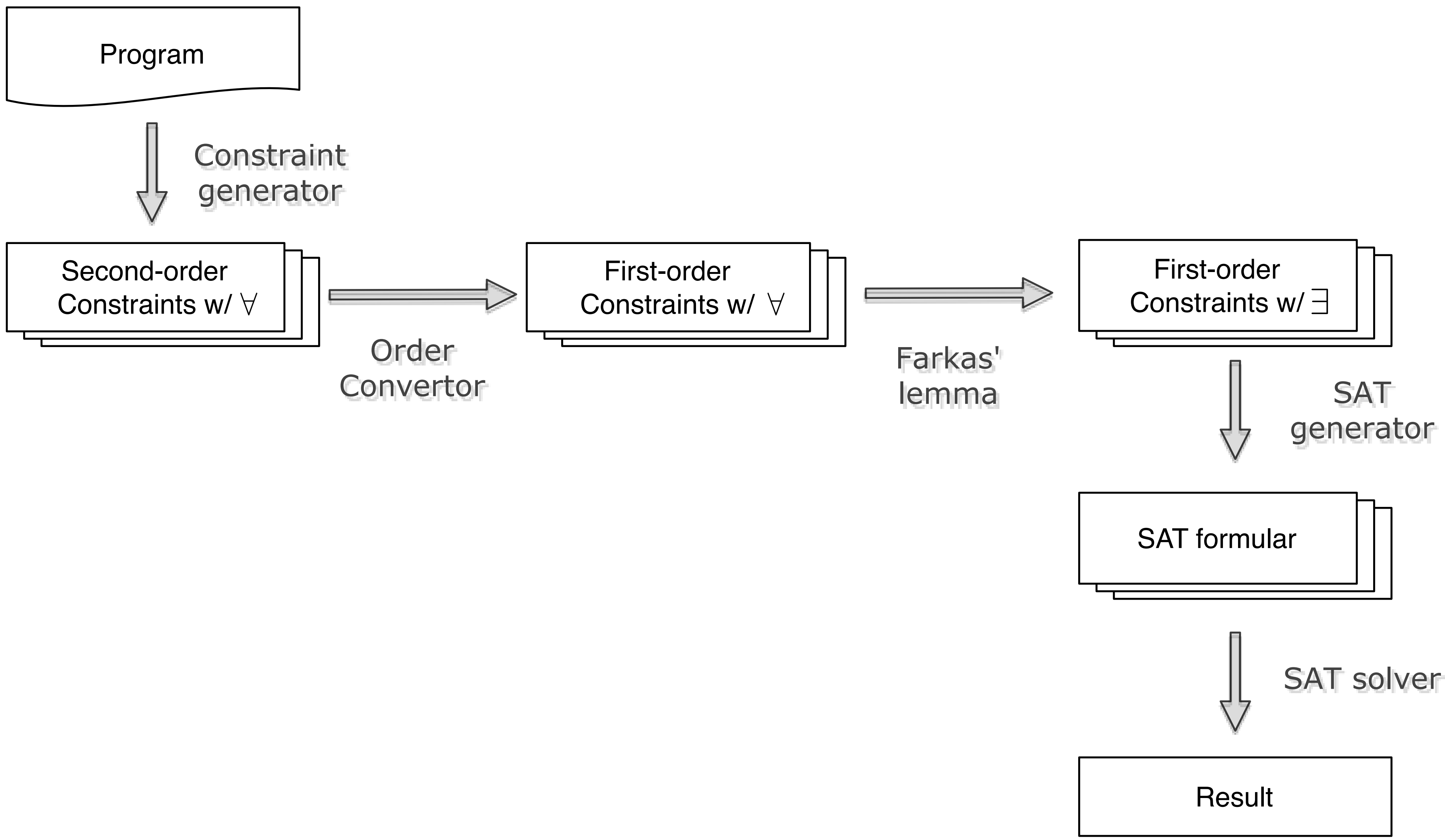
```
PV1 (int y) {
    x := -50;
    while (x < 0) {
      x := x + y;
      y++;
    }
    assert(y > 0)
}
```

# Program verification

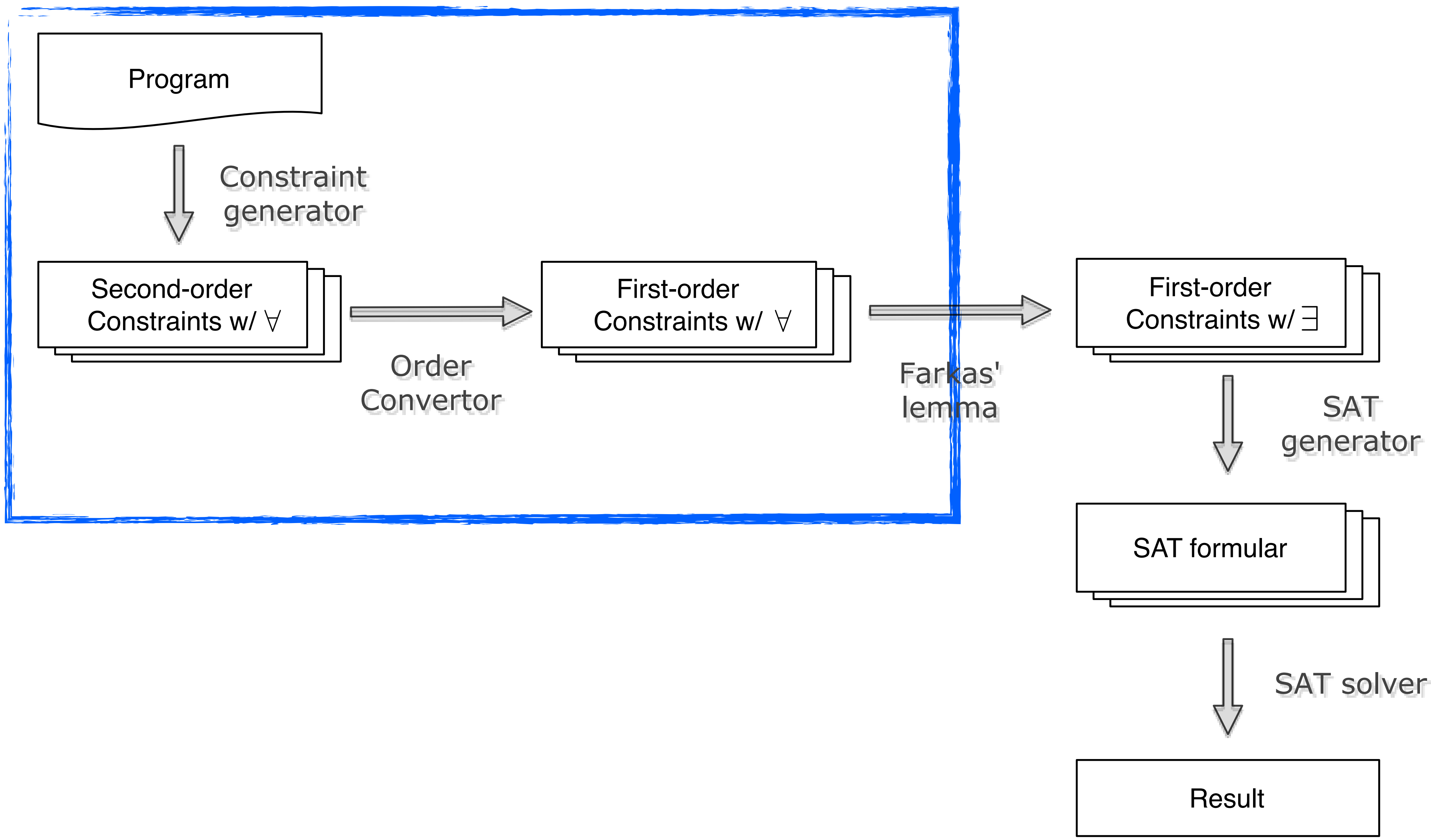
```
PV1 (int y) {
    x := -50;
    while (x < 0) {
        x := x + y;
        y++;
    }
    assert(y > 0)
}
```

Valid or Not?

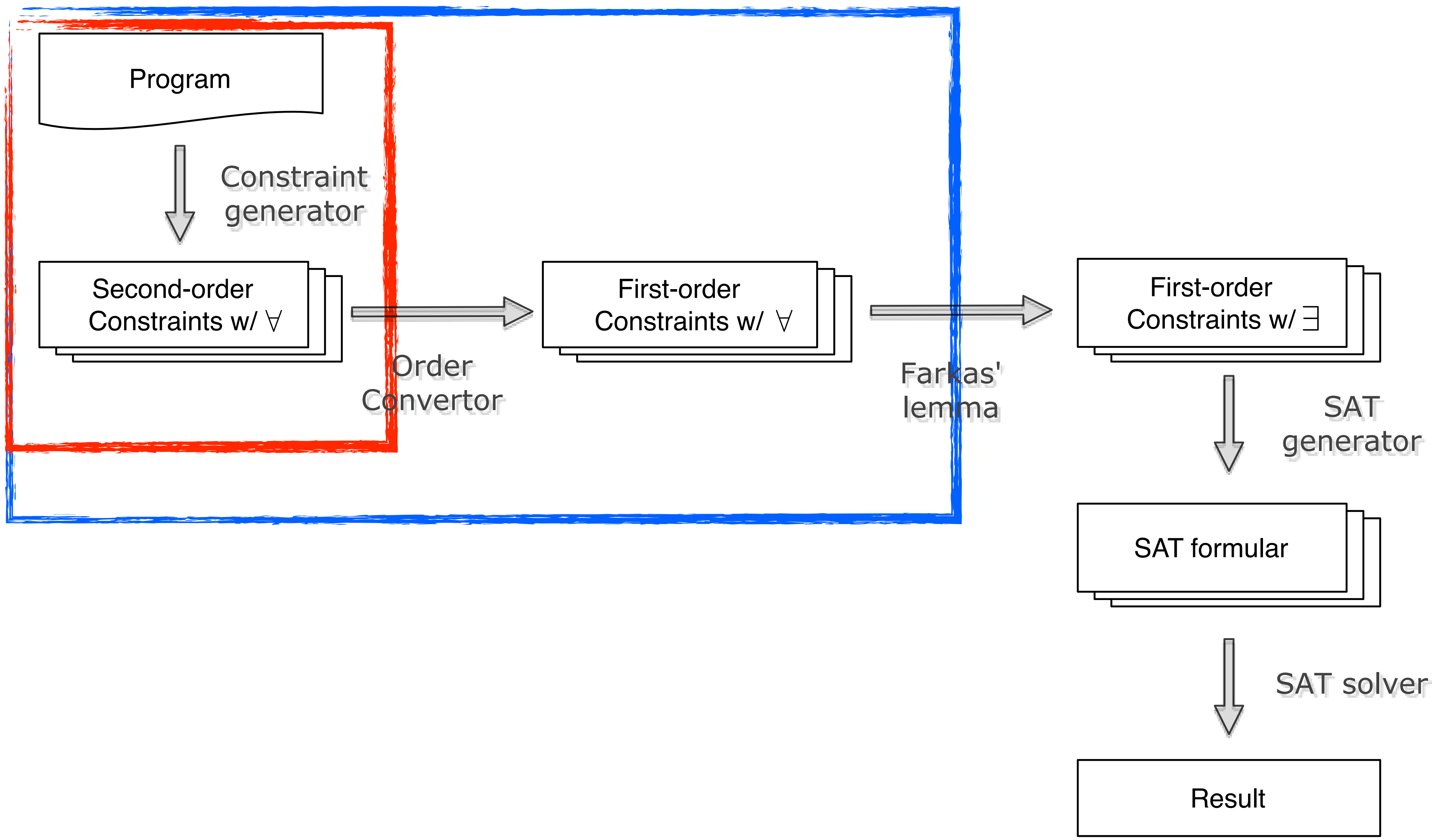
# Bird's-eye view



# Bird's-eye view

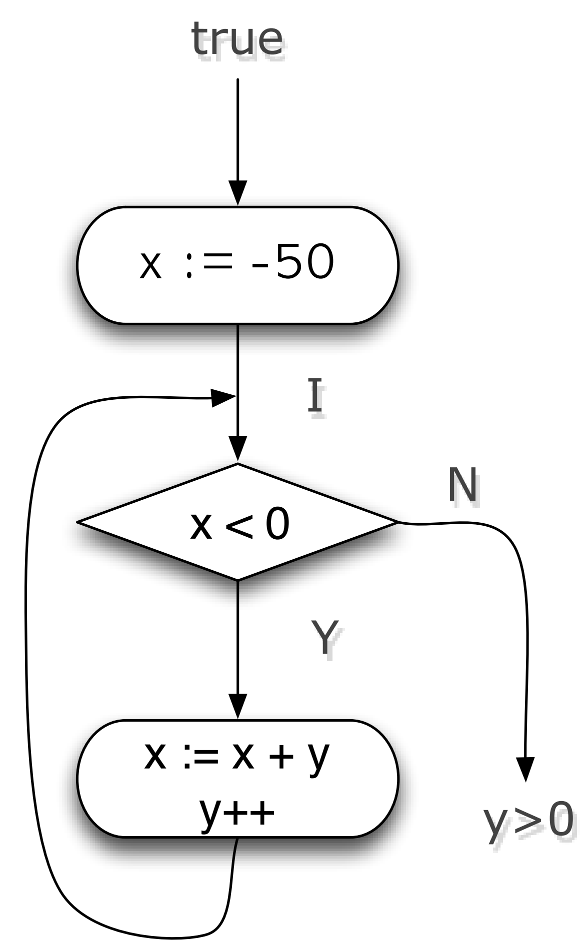


# Bird's-eye view



# Constraint generator

```
PV1 (int y) {
    x := -50;
    while (x < 0) {
       x := x + y;
       y++;
    }
    assert(y > 0)
}
```

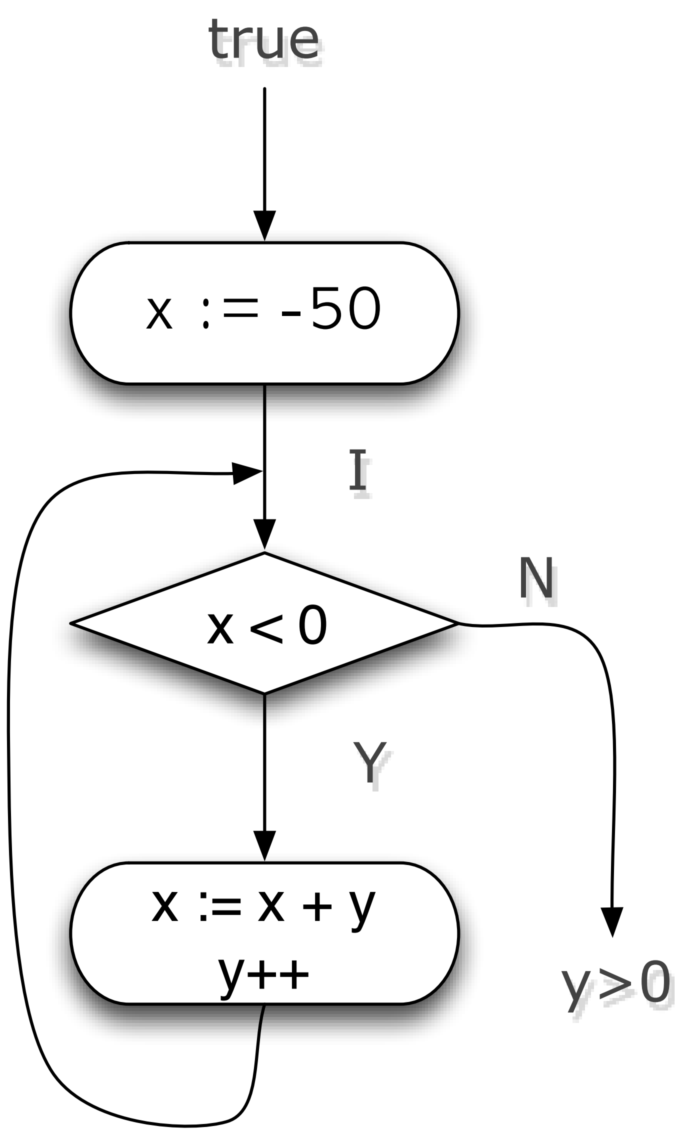



$$\forall_{x,y}(I):$$

$$\begin{array}{rcl} \texttt{true} & \Rightarrow & I[-50/x] \\ I \wedge x < 0 & \Rightarrow & I[(y+1)/y, (x+y)/x] \\ I \wedge x \geq 0 & \Rightarrow & y > 0 \end{array}$$

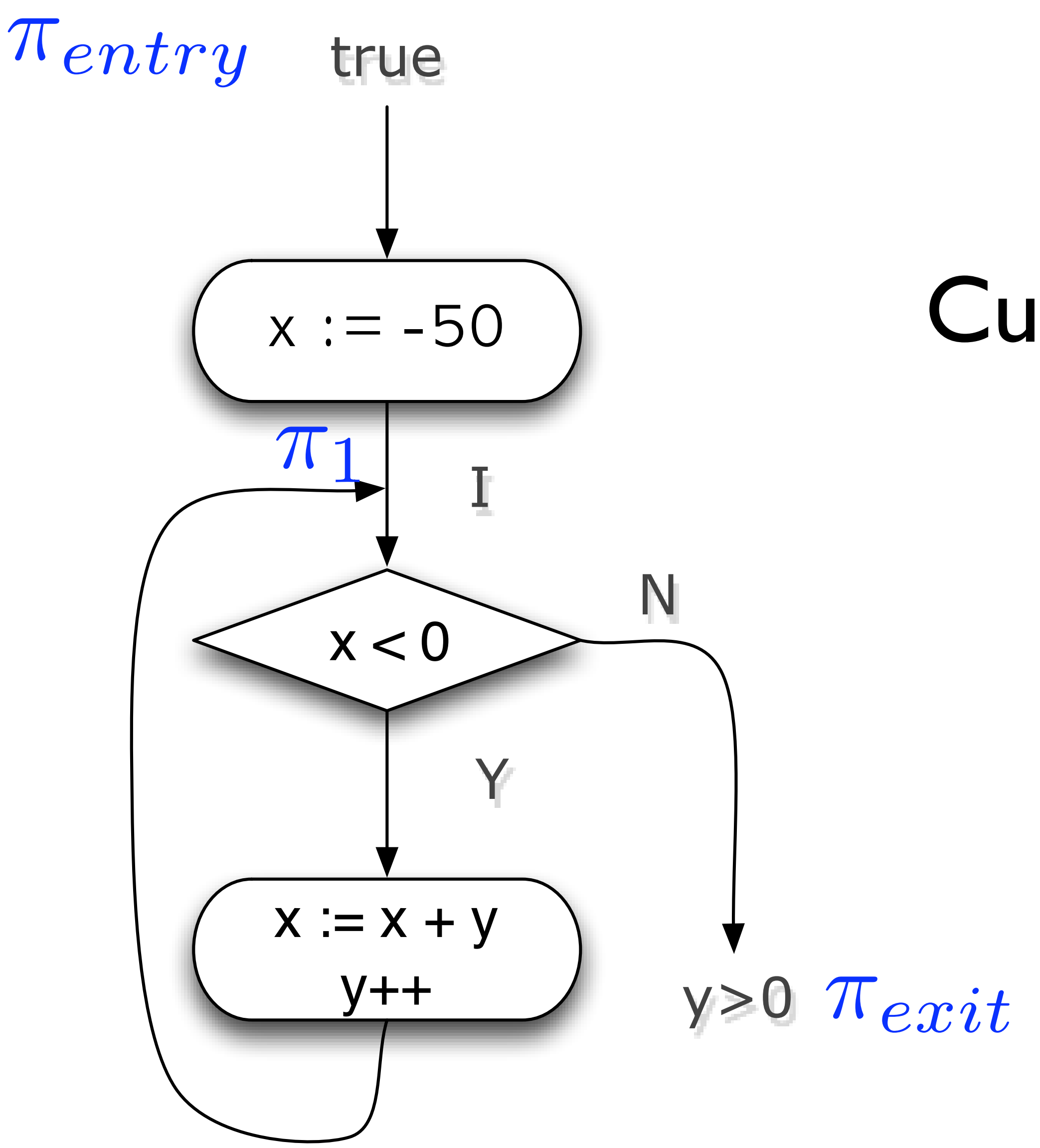
*taken from the reference paper

# Constraint generator



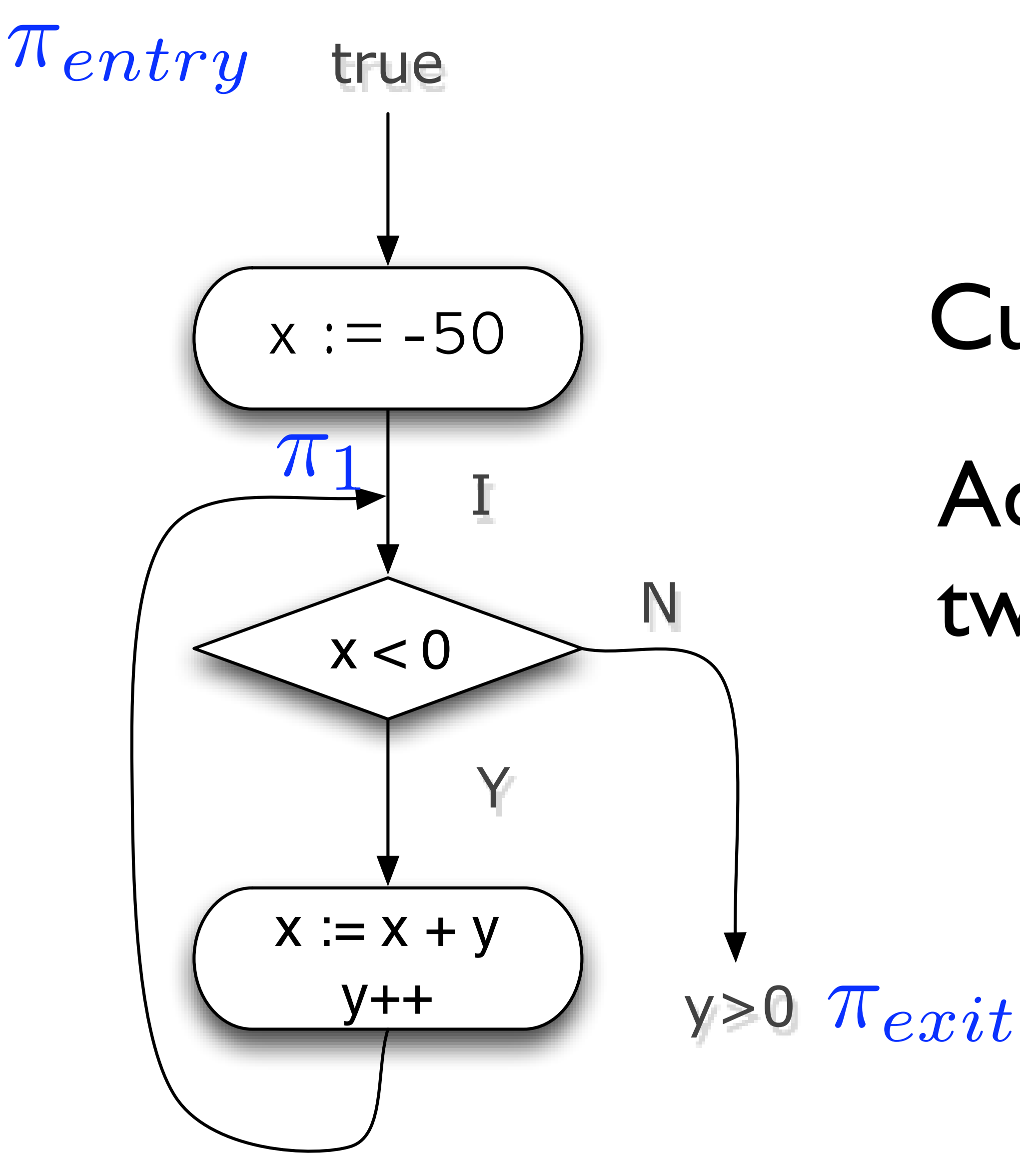

Cut-set : {entry point, exit point, cut-points}

# Constraint generator




Cut-set : { $\pi_{entry}$ , $\pi_{exit}$ , $\pi_1$ }

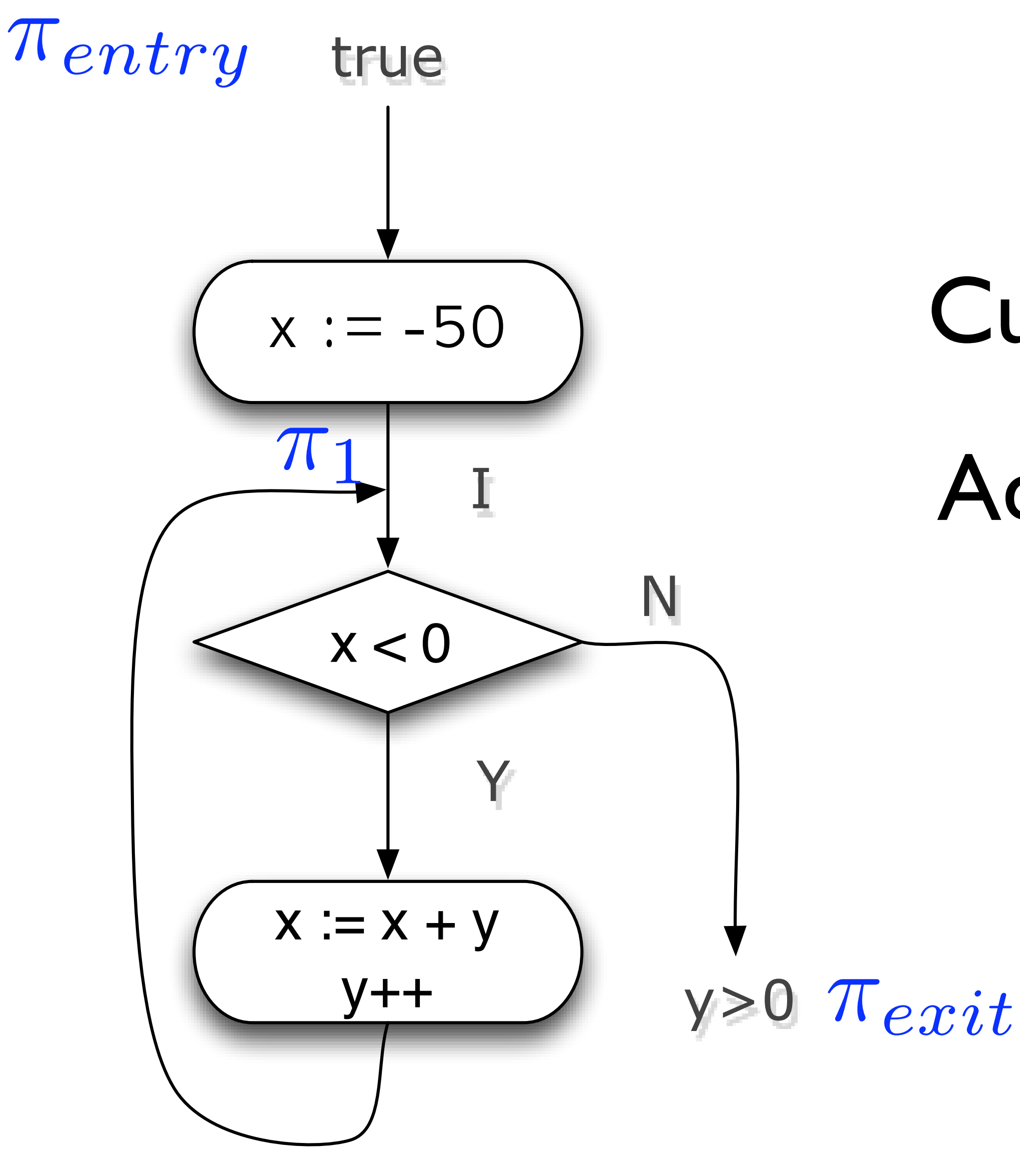
# Constraint generator




Cut-set : { $\pi_{entry}$ , $\pi_{exit}$ , $\pi_1$ }

Adjacent cut-points :
two cut-points directly connected on a path

# Constraint generator

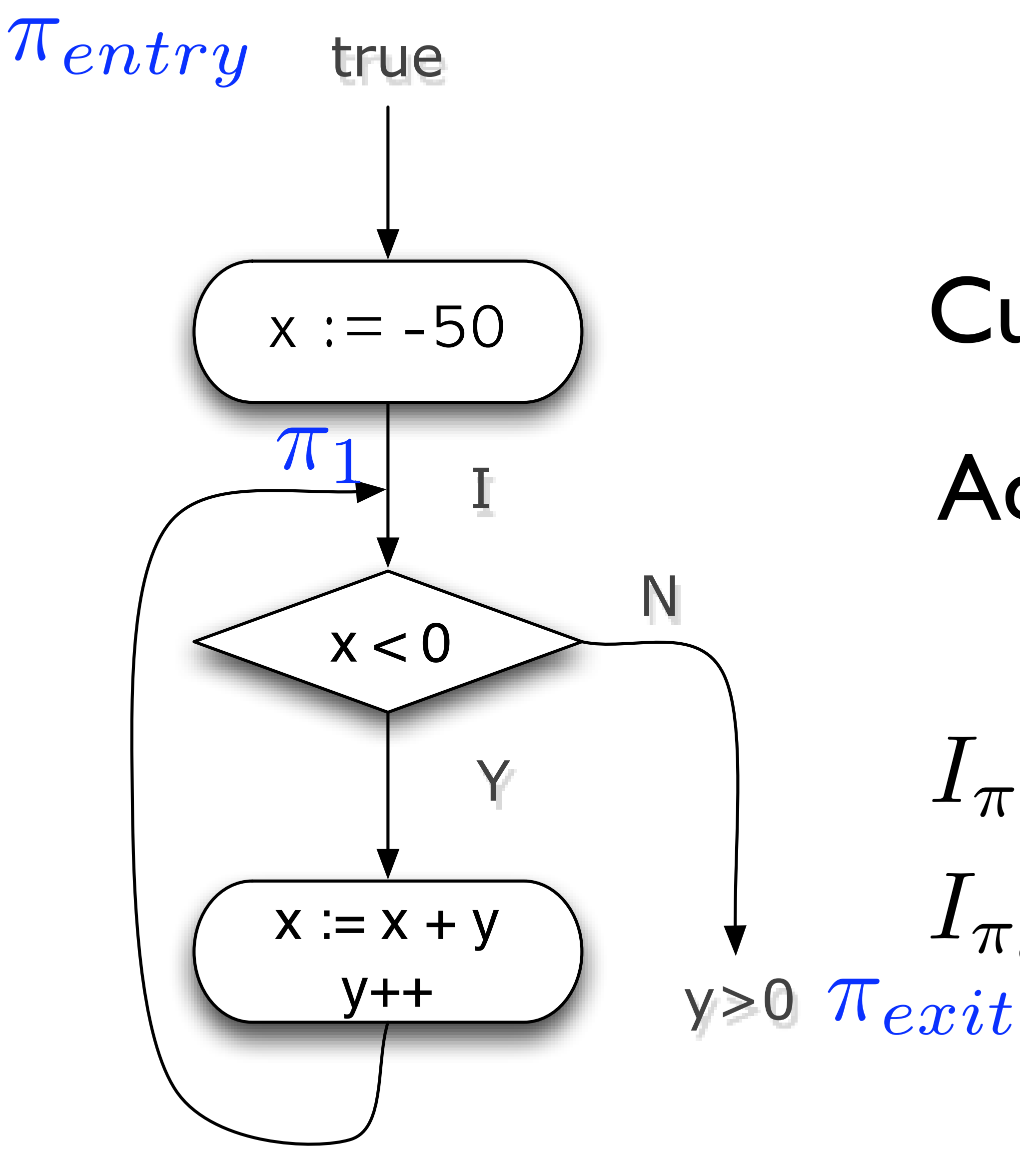



Cut-set : { $\pi_{entry}$ , $\pi_{exit}$ , $\pi_1$ }

Adjacent cut-points :

$$(\pi_{entry}, \pi_1), (\pi_1, \pi_1), (\pi_1, \pi_{exit})$$

# Constraint generator




Cut-set : { $\pi_{entry}$ , $\pi_{exit}$ , $\pi_1$ }

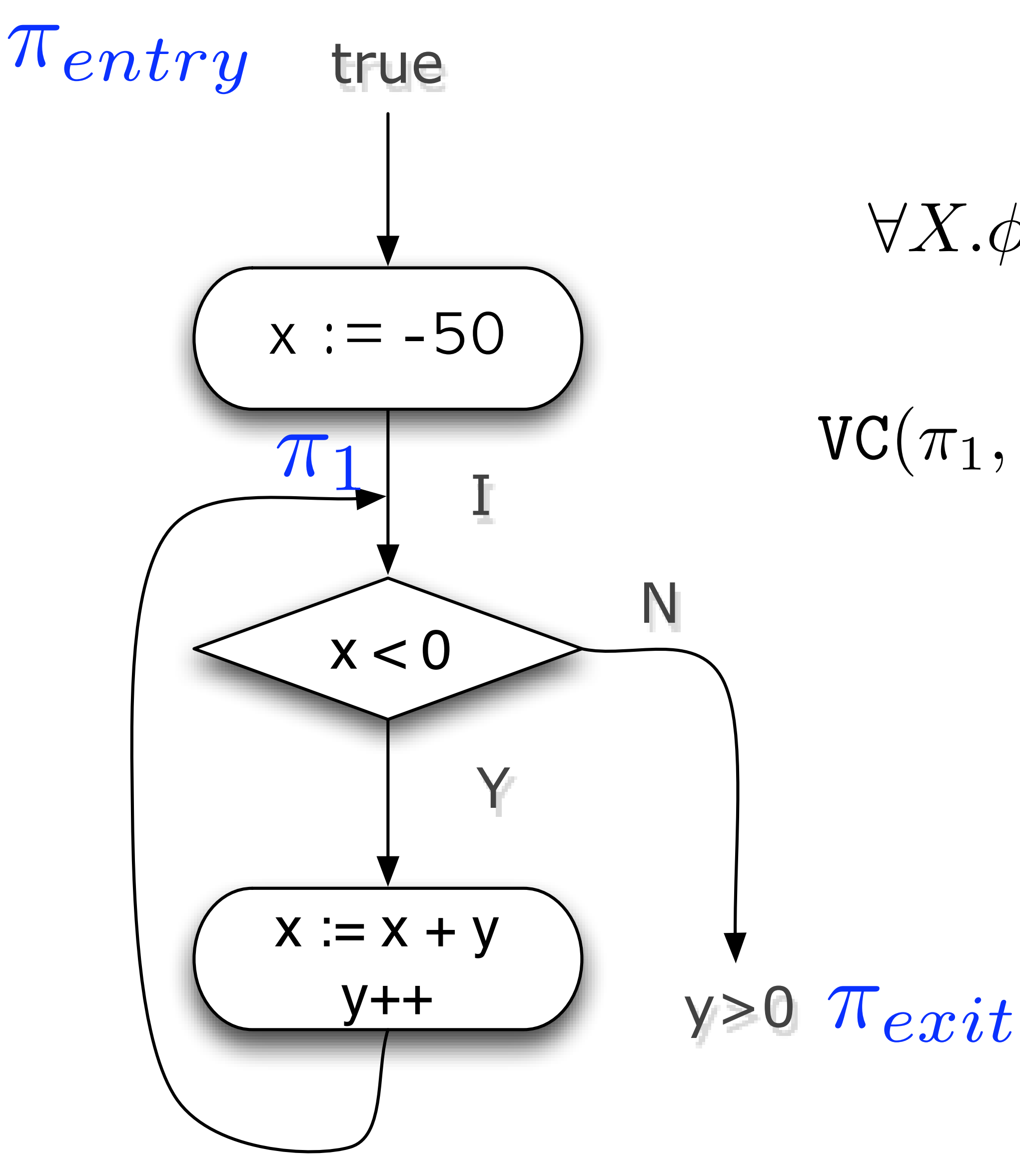
Adjacent cut-points :

$$(\pi_{entry}, \pi_1), (\pi_1, \pi_1), (\pi_1, \pi_{exit})$$

$I_\pi$ : a relation over variables that are live at $\pi$

$I_{\pi_{entry}} = \texttt{true}$ $\quad$ $I_{\pi_{exit}} = \texttt{true}$

# Constraint generator

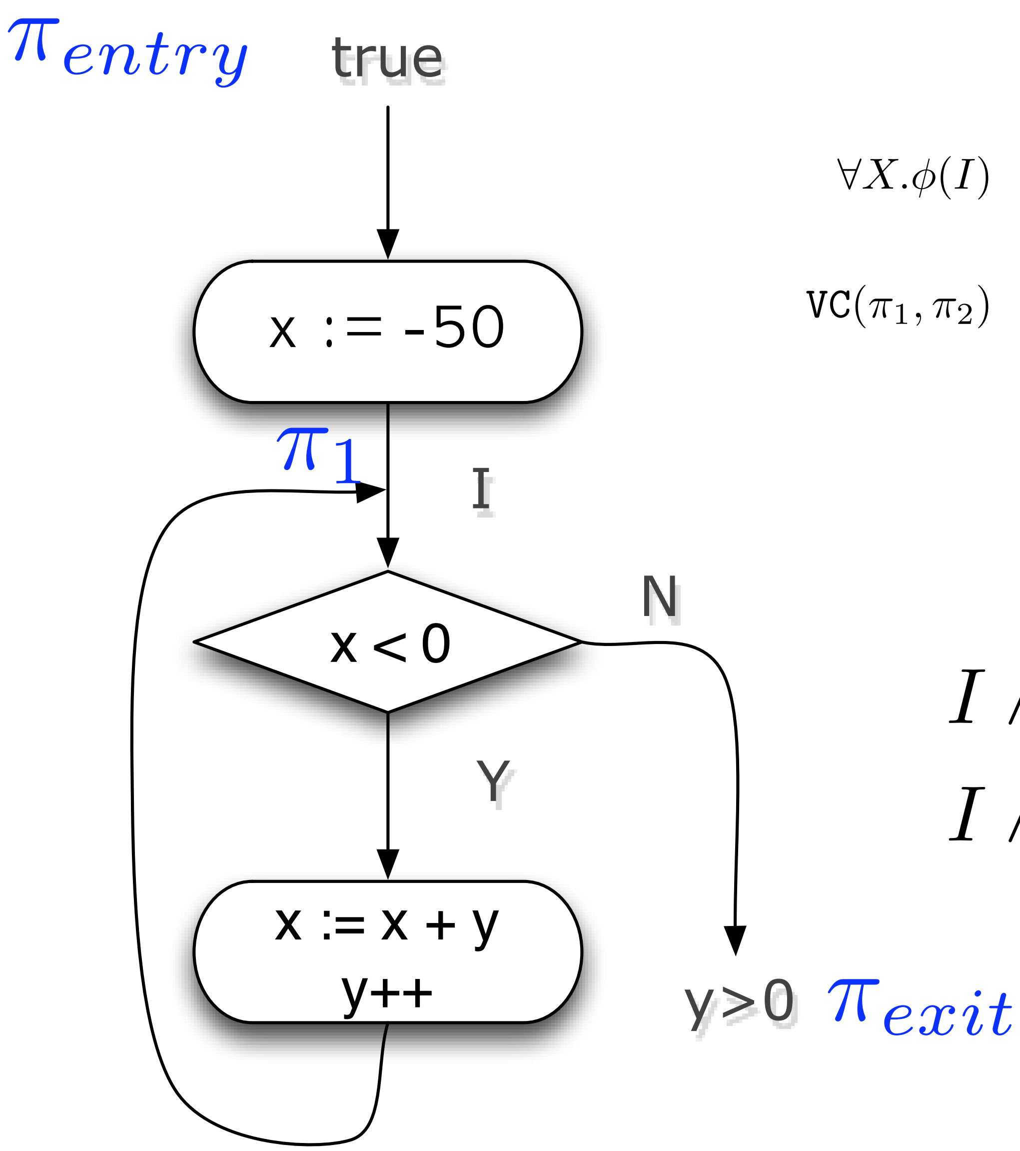



$$\forall X.\phi(I) = \bigwedge_{(\pi_1,\pi_2)\in Adjacent} VC(\pi_1,\pi_2)$$

$$\mathtt{VC}(\pi_1,\pi_2) = \forall X( \bigwedge_{\tau\in\mathtt{Paths}(\pi_1,\pi_2)} (I_{\pi_1} \Rightarrow \omega(\tau, I_{\pi_2})))$$

$$\begin{aligned}
\omega(\mathtt{skip}, I) &= I \\
\omega(x \ := \ e, I) &= I[e/x] \\
\omega(\mathtt{assume} p, I) &= p \Rightarrow I \\
\omega(\mathtt{assert} p, I) &= p \wedge I \\
\omega(S_1; S_2, I) &= \omega(S_1, \omega(S_2, I))
\end{aligned}$$

# Constraint generator



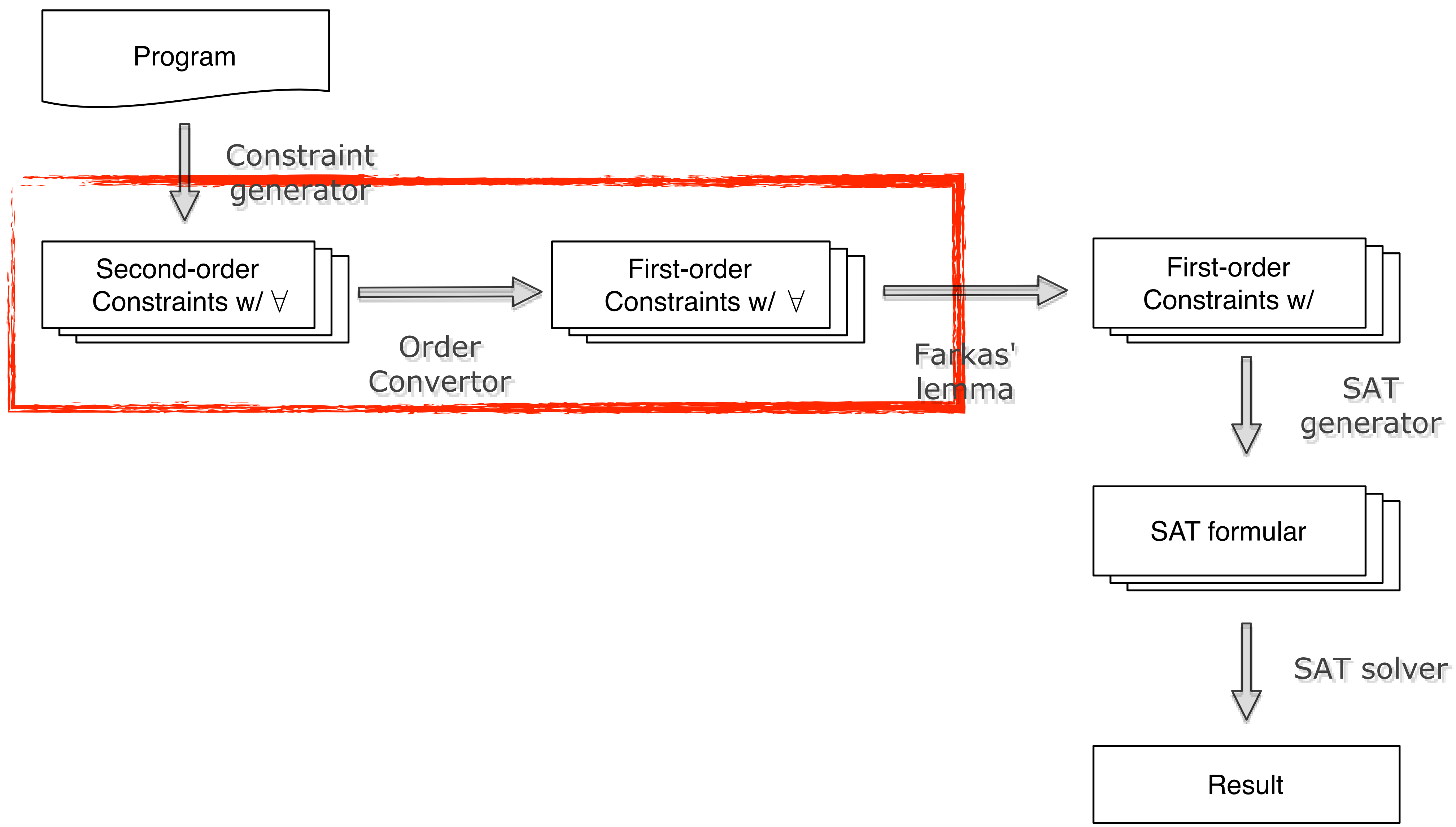

$$\forall X.\phi(I) = \bigwedge_{(\pi_1,\pi_2)\in Adjacent} VC(\pi_1,\pi_2)$$

$$\mathtt{VC}(\pi_1,\pi_2) = \forall X(\bigwedge_{\tau\in \mathtt{Paths}(\pi_1,\pi_2)} (I_{\pi_1} \Rightarrow \omega(\tau, I_{\pi_2})))$$

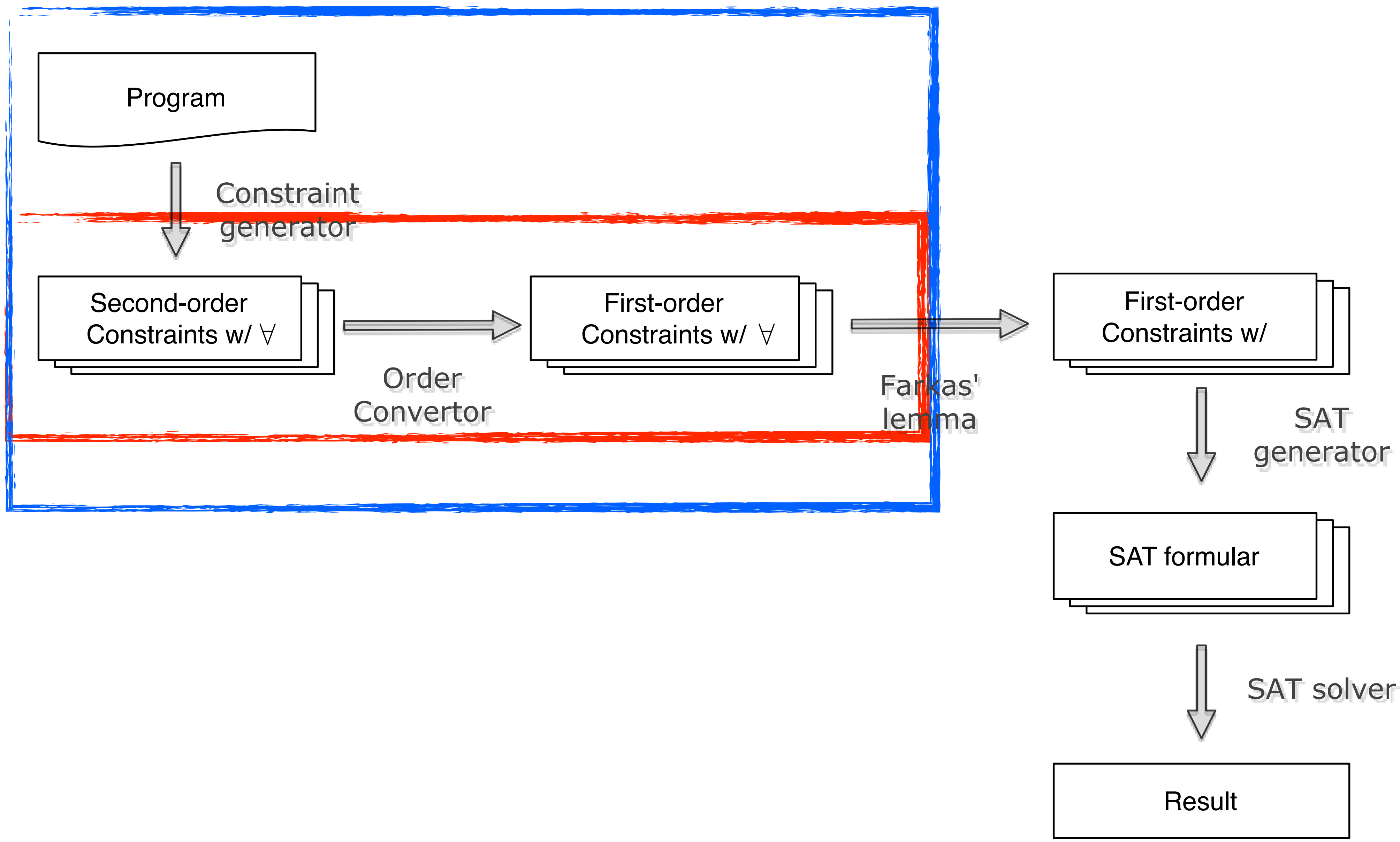
$$\begin{aligned}
\omega(\mathtt{skip}, I) &= I \\
\omega(x := e, I) &= I[e/x] \\
\omega(\mathtt{assume}\, p, I) &= p \Rightarrow I \\
\omega(\mathtt{assert}\, p, I) &= p \wedge I \\
\omega(S_1; S_2, I) &= \omega(S_1, \omega(S_2, I))
\end{aligned}$$

$$\begin{aligned}
\mathtt{true} &\Rightarrow I[-50/x] \\
I \wedge x < 0 &\Rightarrow I[(y+1)/y, (x+y)/x] \\
I \wedge x \geq 0 &\Rightarrow y > 0
\end{aligned}$$

# Bird’s-eye view



# Bird's-eye view



# Order convertor

$$
\begin{array}{rcl}
\forall_{x,y}(I): & & \\
\texttt{true} & \Rightarrow & I[-50/x] \\
I \wedge x < 0 & \Rightarrow & I[(y+1)/y, (x+y)/x] \\
I \wedge x \geq 0 & \Rightarrow & y > 0
\end{array}
$$

- Convert second-order constraint to first-order

# Order convertor

- Assume $\mathrm{I}$ has some form, e.g., $\sum_j a_j x_j \geq 0$
- Make some invariant templates
- Enumerate templates and find a right first-order constraint

# Order convertor

- Assume $I$ has some form, e.g., $\sum_j a_j x_j \geq 0$
- Make some invariant templates
- Enumerate templates and find a right first-order constraint

**Similar to our approach**

# Order convertor

$$
\begin{aligned}
\forall_{x,y}(I):& \\
\texttt{true} &\Rightarrow I[-50/x] \\
I \wedge x < 0 &\Rightarrow I[(y+1)/y, (x+y)/x] \\
I \wedge x \geq 0 &\Rightarrow y > 0
\end{aligned}
$$

For example, 3 kinds of templates

$$a_1x + a_2y + a_3 \geq 0$$

$$
\begin{aligned}
& a_1x + a_2y + a_3 \geq 0 \\
\vee\ & a_4x + a_5y + a_6 \geq 0
\end{aligned}
$$

$$
\begin{aligned}
& a_1x + a_2y + a_3 \geq 0 \\
\vee\ & a_4x + a_5y + a_6 \geq 0 \\
\vee\ & a_7x + a_8y + a_9 \geq 0
\end{aligned}
$$

# Order convertor

$$\begin{array}{rcl} \forall_{x,y}(I): & & \\ \texttt{true} & \Rightarrow & I[-50/x] \\ I \wedge x < 0 & \Rightarrow & I[(y+1)/y, (x+y)/x] \\ I \wedge x \geq 0 & \Rightarrow & y > 0 \end{array}$$

For example, 3 kinds of templates

$$a_1 x + a_2 y + a_3 \geq 0$$

$$\begin{array}{rcl} a_1 x + a_2 y + a_3 & \geq & 0 \\ \vee\ a_4 x + a_5 y + a_6 & \geq & 0 \end{array}$$

$$\begin{array}{rcl} a_1 x + a_2 y + a_3 & \geq & 0 \\ \vee\ a_4 x + a_5 y + a_6 & \geq & 0 \\ \vee\ a_7 x + a_8 y + a_9 & \geq & 0 \end{array}$$

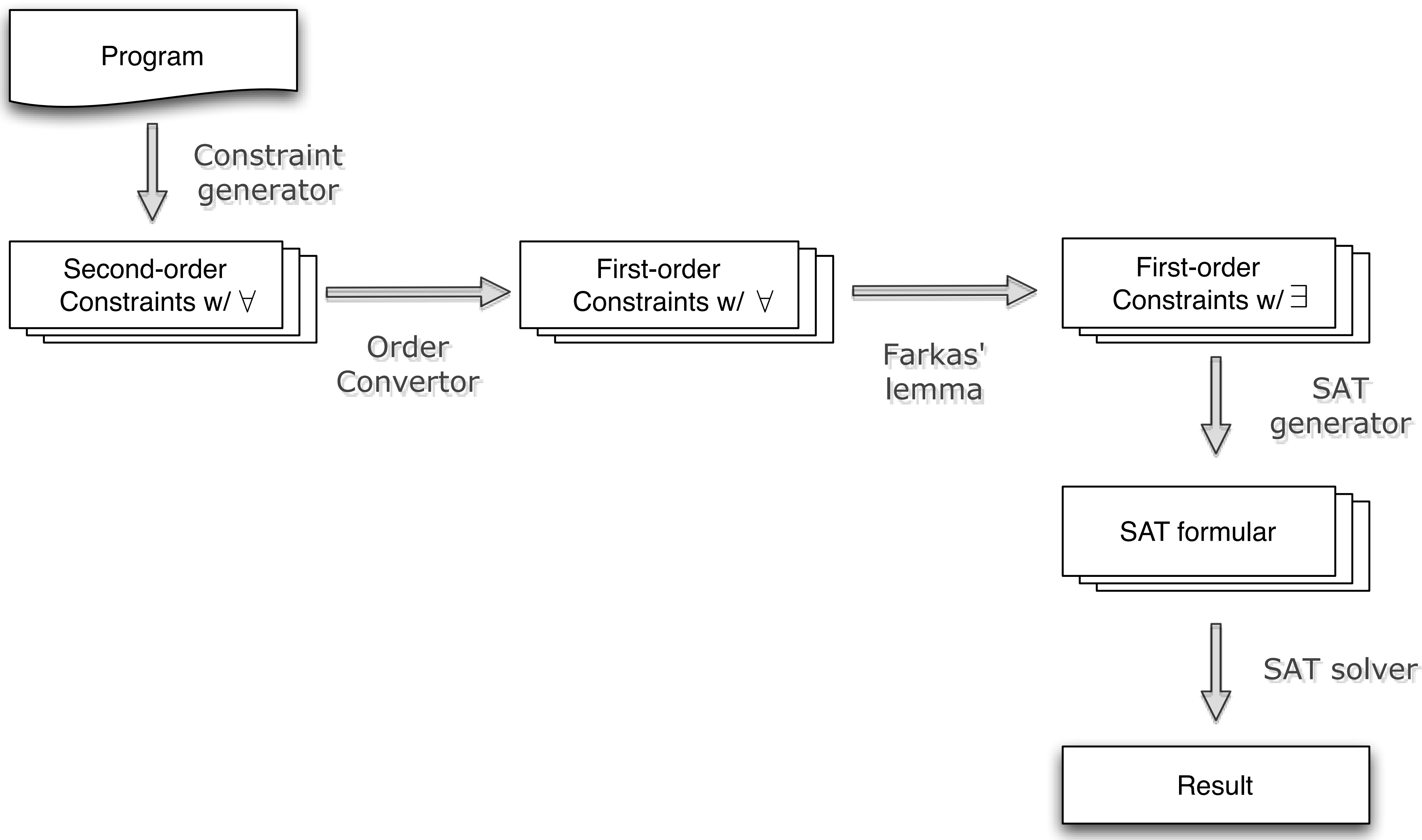
# Order convertor

$$
\begin{aligned}
&\forall_{x,y}(I): \\
\texttt{true} &\Rightarrow I[-50/x] \\
I \wedge x < 0 &\Rightarrow I[(y+1)/y, (x+y)/x] \\
I \wedge x \geq 0 &\Rightarrow y > 0
\end{aligned}
$$

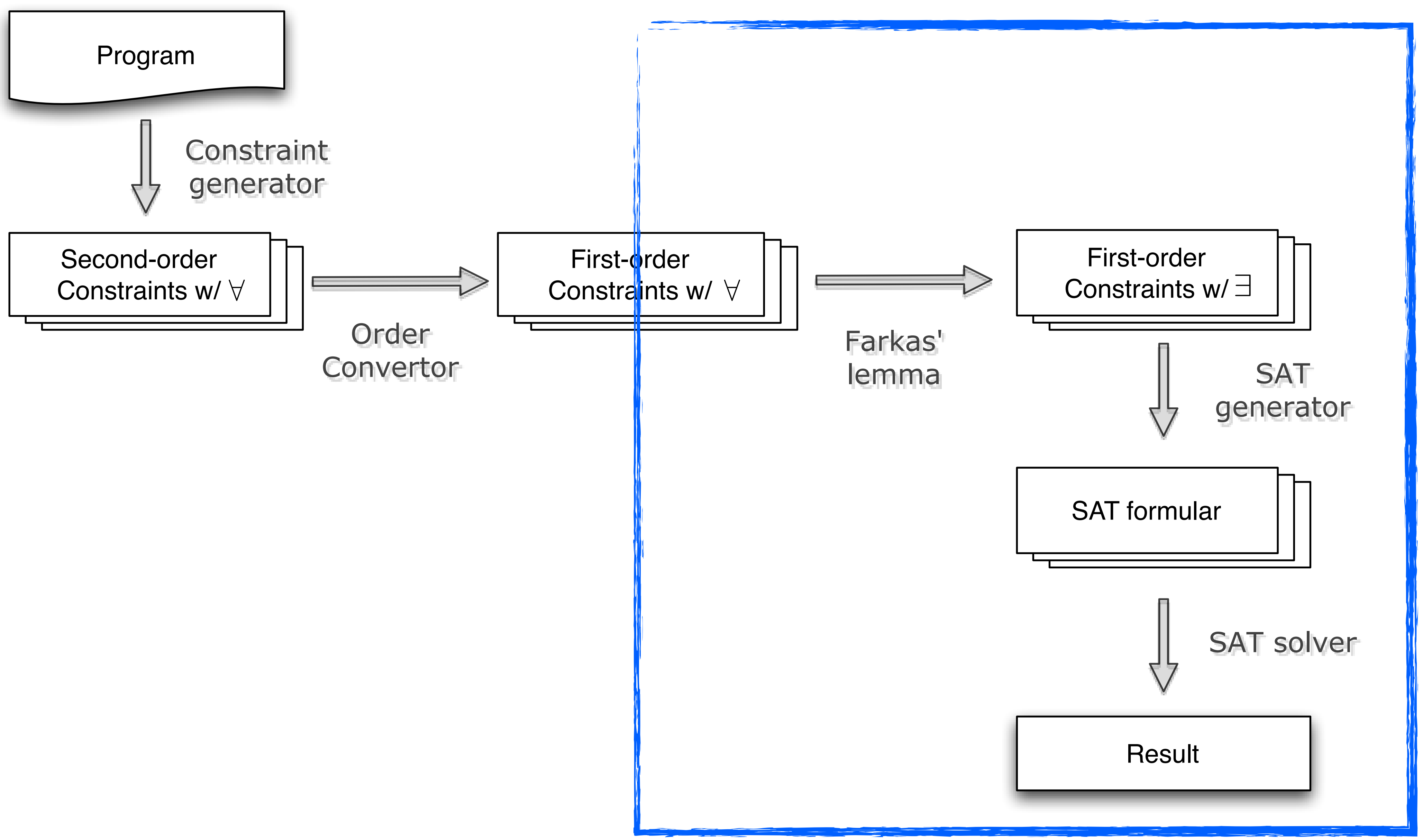
Assume I :
$$
\begin{aligned}
a_1 x + a_2 y + a_3 &\geq 0 \\
\vee\ a_4 x + a_5 y + a_6 &\geq 0
\end{aligned}
$$

$$
\begin{aligned}
&\forall_{x,y}\phi(a_j): \\
\texttt{true} &\Rightarrow -50a_1 + a_2 y + a_3 \\
&\quad \vee -50a_4 + a_5 y + a_6 \\
I \wedge x < 0 &\Rightarrow a_1(x+y+1) + a_2(y+1) + a_3 \\
&\quad \vee\ a_4(x+y+1) + a_5(y+1) + a_6 \\
I \wedge x \geq 0 &\Rightarrow y > 0
\end{aligned}
$$

# Bird's-eye view



# Bird's-eye view



# Experiment result

| Name | This paper | BLAST | SLAM | GR06 | ARMC |
|---|---|---|---|---|---|
| barbr | 0.41 | ! | 43.9 | 10.5 | 674 |
| berkeley | 3.00 | ! | 2.90 | 0.10 | 4 |
| bk-nat | 5.30 | ! | ! | 0.43 | 3.25 |
| seesaw | 3.23 | ! | 1.0 | 0.82 | >2000 |

! : inability of tool to discover new predicates

# Conclusion

- Very simple but powerful
  - don't need fix-point calculation
- Real-world invariants : Not that complicate
  - Real-world data structure : Not that complicate (in our case)

# Thank you