# Program verification 2019

## Lecture 5: Abstract Model checking

Instructor: Ashutosh Gupta

IITB, India

Compile date: 2019-01-15

# Where are we?

- SMT solving
- Lattice theory
- Abstract interpretation

# Lecture plan

- Model checking with finite abstraction
- Predicate abstraction
- Abstract reachability graph
- Model checking vs. Abstract interpretation
- Abstract counterexample
- Abstraction refinement
- Counter example guided abstraction refinement

Topic 5.1

# Model checking

# Abstract program

### Definition 5.1

*Let us consider a finite abstraction $D$ and a program $P = (V, L, \ell_0, \ell_e, E)$. An* abstract program *$P^{\#} = \textsc{Abstract}(P, D)$ is $(V, L, \ell_0, \ell_e, E^{\#})$ where $E^{\#}$ is defined as follows. If $(\ell, \rho, \ell') \in E$ then $(\ell, \rho^{\#}, \ell') \in E^{\#}$, where*

$$\rho^{\#} = \{\gamma(d) \times \gamma(d') | d' = sp^{\#}(d, \rho)\}.$$

We assume $D$ and $P$ allow $\rho^{\#}$ to be easily representable in a computer.

### Theorem 5.1

$\forall d \in D \exists d' \in D.\ sp(\gamma(d), \rho^{\#}) = \gamma(d')$

### Theorem 5.2

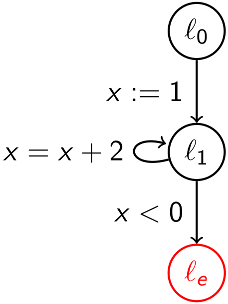
*If $P^{\#}$ is safe then $P$ is safe.*

# Example: abstract program

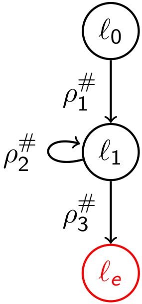
## Example 5.1

*Consider the following program and sign abstraction* $D = \{\top, -, 0, +, \bot\}$.

*Program:*




*Abstract program:*




$$
\begin{aligned}
\rho_1^{\#} &= \{(-,+),(0,+),(+,+),(\top,+)\}\\
\rho_2^{\#} &= \{(-,+),(-,0),(-,+),(0,+),(+,+)\\
&\quad (\top,\top)\}\\
\rho_3^{\#} &= \{(-,-),(\top,-)\}
\end{aligned}
$$

We have only listed pairs that do not have $\bot$ as second component.

# Abstract reachability graph

Since $D = (\sqsubseteq, \top, \bot)$ is finite, symbolic execution of $P^{\#} = \textsc{Abstract}(P, D)$ will only produce finitely many symbolic states, which are called abstract states.

## Definition 5.2

*Abstract reachability graph(ARG)* $(reach, R)$ *is the smallest directed graph such that*

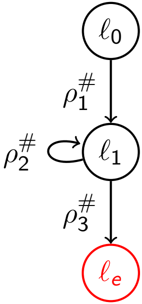
- $reach \subseteq L \times D$
- $(\ell_0, \top) \in reach$
- *If* $((\ell, d), (\ell', d')) \in R$ *iff* $\exists(\ell, \rho^{\#}, \ell') \in E^{\#}.\ \ d' = sp(d, \rho^{\#})$

## Theorem 5.3

*If* $\forall d.\ d \neq bot \wedge (l_e, d) \notin reach$ *then* $P^{\#}$ *is safe.*

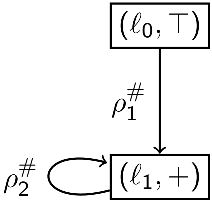
# Example: abstract reachability graph

Abstract program:




$\rho_1^\# = \{(-,+),(0,+),(+,+),(\top,+)\}$
$\rho_2^\# = \{(-,+),(-,0),(-,+),(0,+),(+,+)$
$(\top,\top)\}$
$\rho_3^\# = \{(-,-),(\top,-)\}$

Abstract reachability graph:




We are not showing abstract states with $\bot$.

# Model checking

The word model checking originated from the area of modal logic, where finding a model that satisfies a formula is called model checking.

In our situation, we have a logical statement $P^{\#}$ is not safe

We search for a model of the statement, i.e., a path in the abstract reachability graph that reaches to error location.

If no model found, then $P^{\#}$ is safe.

Abstract reachability graph may be large.

In contrast, abstract interpretation does not construct large objects.

# Abstract model checking

**Algorithm 5.1:** $\textsc{AbstractMC}(P^{\#} = (V, L, \ell_0, \ell_e, E^{\#}),\ D = (\sqsubseteq, \top, \bot))$

**Output:** $\textsc{Correct}$ if $P^{\#}$ is safe, abstract counterexample otherwise

$\mathit{worklist} := \{(\ell_0, \top)\}$; $\mathit{reach} := \emptyset$; $\mathit{covered} := \emptyset$;

$\mathit{parent} : \mathit{reach} \cup \mathit{worklist} \to \mathit{reach} \cup \mathit{worklist} := \{((\ell_0, \top), (\ell_0, \top))\}$;

$\mathit{path} : \mathit{reach} \cup \mathit{worklist} \to$ (sequences of $E^{\#}$) $:= \{((\ell_0, \top), \epsilon)\}$;

**while** $\mathit{worklist} \neq \emptyset$ **do**

- choose $(\ell, d) \in \mathit{worklist}$; $\mathit{worklist} := \mathit{worklist} \setminus \{(\ell, d)\}$;
- **if** $d = \bot$ *or* $\exists s \in \mathit{parent}^*((\ell, d)).\ (\_, s) \in \mathit{covered}$ **then** **continue**;
- **if** $\ell = \ell_e$ **then** **return** $\textsc{Counterexample}(\mathit{path}(\ell, d))$ ;
- $\mathit{reach} := \mathit{reach} \cup \{(\ell, d)\}$;
- **if** $\exists (\ell, d') \in \mathit{reach} - \mathit{range}(\mathit{covered}).\ d \sqsubseteq d'$ **then**
  - $\mathit{covered} := \mathit{covered} \cup \{((\ell, d'), (\ell, d))\}$
- **else**
  - **if** $\exists (\ell, d') \in \mathit{reach} - \mathit{range}(\mathit{covered}).\ d' \sqsubseteq d$ **then**
    - $\mathit{covered} := \mathit{covered} \cup \{((\ell, d), (\ell, d'))\}$
  - **foreach** $(\ell, \rho^{\#}, \ell') \in E^{\#}$ **do**  
    $P^{\#}$ accessed only once
    - $d' := sp(d, \rho^{\#})$; $\mathit{worklist} := \mathit{worklist} \cup \{(\ell', d')\}$;
    - $\mathit{parent}((\ell', d')) = (\ell, d)$; $\mathit{path}((\ell', d')) = \mathit{path}((\ell, d)).(\ell, \rho^{\#}, \ell')$;

**return** $\textsc{Correct}$

# On the fly abstraction

In $\textsc{AbstractMC}$, we only access $P^{\#}$ to compute post operator over $d$.This suggests, $\textsc{AbstractMC}$ can be implemented in the following two ways.

- Precompute $P^{\#}$ and run $\textsc{AbstractMC}$ as presented.
- On the fly construction of $P^{\#}$. We construct transitions of $P^{\#}$ as we need them

## Exercise 5.1

*Discuss benefits of both the approaches*

# Finite abstraction

The following abstractions are widely used in modelcheckers

- Cartesian predicate abstraction
- Boolean predicate abstraction

# Finite abstraction example : Cartesian predicate abstraction

Cartesian predicate abstraction is defined by a set of predicates

$Preds = \{p_1, \dots, p_n\}$

$C = \mathfrak{p}(\mathbb{Q}^{|V|})$

$D = \bot \cup \mathfrak{p}(Preds)$ // $\emptyset$ represents $\top$

$\bot \sqsubseteq S_1 \sqsubseteq S_2$ if $S_2 \subseteq S_1$

$\alpha(c) = \{p \in P | c \Rightarrow p\}$

$\gamma(S) = \bigwedge S$

## Example 5.2

$V = \{\mathtt{x}, \mathtt{y}\}$

$P = \{\mathtt{x} \leq 1, -\mathtt{x} - \mathtt{y} \leq -1, \mathtt{y} \leq 5\}$

$\alpha(\{(0, 0)\}) = \{\mathtt{x} \leq 1, \mathtt{y} \leq 5\}$

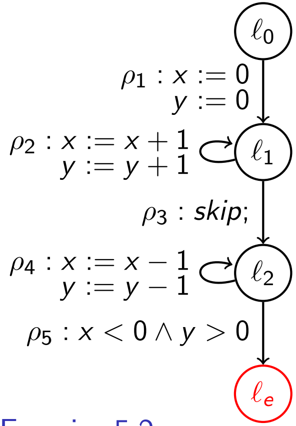
$\alpha((\mathtt{x} - 1)^2 + (\mathtt{y} - 3)^2 = 1) = \{-\mathtt{x} - \mathtt{y} \leq -1, \mathtt{y} \leq 5\}$

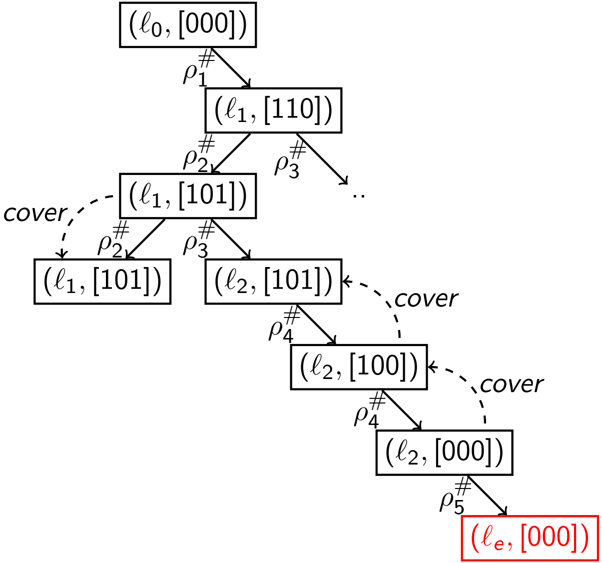
We represent abstract state as bit vectors, e.g., [101] represents $\mathtt{x} \leq 1 \wedge \mathtt{y} \leq 5$

# Example: ARG with Cartesian predicate abstraction

$Preds = \{x \geq 0, y \leq 0, x \geq 1\}$.

Program:




## Exercise 5.2

*Complete the ARG*

# Spurious counterexample

$\textsc{AbstractMC}(P^{\#}, D)$ may fail to prove $P^{\#}$ correct and return a path $e_1^{\#} \dots e_m^{\#}$, which is called abstract counterexample.

Let $e_1 \dots e_m$ be the corresponding path in $P$. Now we have two possibilities.

- $e_1 \dots e_m$ is feasible. Then, we have found a bug
- $e_1 \dots e_m$ is not feasible. Then, we call $e_1 \dots e_m$ as spurious counterexample.

We need to fix our abstraction such that we do not get the spurious counter example.

# Example : spurious counterexample

## Example 5.3

$$\boxed{(\ell_0, [000])} \xrightarrow{\rho_1^{\#}} \boxed{(\ell_1, [110])} \xrightarrow{\rho_2^{\#}} \boxed{(\ell_1, [101])} \xrightarrow{\rho_3^{\#}} \boxed{(\ell_2, [101])} \xrightarrow{\rho_4^{\#}} \boxed{(\ell_2, [100])} \xrightarrow{\rho_4^{\#}} \boxed{(\ell_2, [000])} \xrightarrow{\rho_5^{\#}} \boxed{(\ell_e, [000])}$$

*Since we cannot execute $\rho_1\rho_2\rho_3\rho_4\rho_4\rho_5$, the path is a spurious counterexample.*

*We check the feasibility of the path using satisfiability of path constraints.*

# Refinement relation

### Definition 5.3

*Consider abstractions*

$$(C, \subseteq) \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} (D_1, \sqsubseteq_1) \quad \textit{and} \quad (C, \subseteq) \underset{\alpha_1}{\overset{\gamma_2}{\leftrightarrows}} (D_2, \sqsubseteq_2).$$

*$D_2$ refines $D_1$ if*

$$\forall c \in C.\ \gamma_1(\alpha_1(c)) \subseteq \gamma_2(\alpha_2(c))$$

### Exercise 5.3

*$\gamma_1 \circ \alpha_2$ is order embedding.*

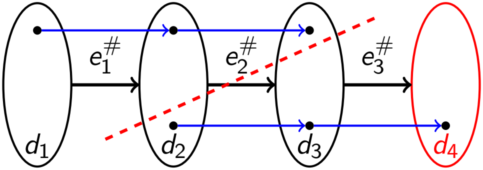
# Abstraction refinement

## Theorem 5.4

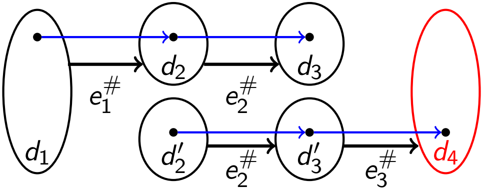
*If* $\text{ABSTRACT}(P, D_1)$ *exhibits a spurious counterexample then there is an abstraction* $D_2$ *such that* $D_2$ *refines* $D_1$ *and* $\text{ABSTRACT}(P, D_2)$ *does not exhibit the same counter example.*

## Proof sketch.

Spurious counterexample:




Refined abstraction:




We say the refinement to $D_2$ from $D_1$ ensures progress, i.e., counterexamples are not repeated if ARG is build again with $D_2$

# Refinment Strategy for predicate abstraction

**General refinement strategy**
Split abstract states such that the spurious counterexample is disconnected.

In predicate abstraction, we only need to add more predicates. The new abstraction will certainly be refinement.

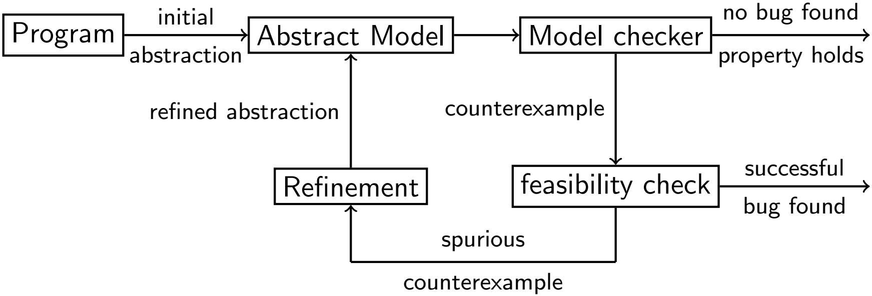
# Example: refinement

Adding predicate $y \leq -1$ will remove the spurious counterexample.

$\mathit{Preds} = \{x \geq 0, y \leq 0, x \geq 1, y \leq 1\}$

$$\boxed{(\ell_0, [0000])} \xrightarrow{\rho_1^\#} \boxed{(\ell_1, [1101])} \xrightarrow{\rho_2^\#} \boxed{(\ell_1, [1011])} \xrightarrow{\rho_3^\#} \boxed{(\ell_2, [1011])} \xrightarrow{\rho_4^\#} \boxed{(\ell_2, [1101])} \xrightarrow{\rho_4^\#} \boxed{(\ell_2, [0101])} \xrightarrow{\rho_5^\#} \boxed{(\ell_e, \bot)}$$

# CEGAR: CounterExample Guided Abstraction Refinement



# Computing refinement

In order to automate CEGAR, we need an effective method for computing new predicates that result in the desired refinement.

Here, we discuss the following two methods for refinement of a predicate abstraction.

- Syntax based refinement
- Interpolation based refinement

Let us first define(remind) some notations.

# Path constraints for spurious counterexample (reminder)

$V_i$ be the vector of variables obtained by adding subscript $i$ after each variable in $V$.

## Definition 5.4

*For a spurious counterexample $e_1 \dots e_n$, path constraints $pathCons(e_1 \dots e_n)$ is*

$$\bigwedge_{i \in 1..n} e_i(V_{i-1}, V_i)$$

*A path is feasible if corresponding path constraints is satisfiable.*

Note: Path constraints are also known as "SSA formulas".

# Syntax based refinement

$core = \text{unsatCore}(\ pathCons(e_1 \ldots e_n)\ )$.

$preds$ = atoms of *core* after erasing subscripts in its variables

Add *preds* in the predicate domain to obtain the refined abstract domain

# Interpolation

## Definition 5.5

*Let $A$ and $B$ be formulas such that $A \wedge B$ is unsat. An interpolant $\mathrm{I}$ between $A$ and $B$ is a formula such that*

- $A \Rightarrow \mathrm{I}$
- $B \wedge \mathrm{I} \Rightarrow \bot$
- $vars(\mathrm{I}) \subseteq vars(A) \cap vars(B)$

## Theorem 5.5 (Craig interpolation theorem)

*Interpolant always exists.*

## Example 5.4

*Consider:*

$A = x_1 + x_2 \leq 2 \wedge x_3 - x_2 \leq 0$

$B = 6x_4 - 2x_1 \leq -8 \wedge -3x_4 - x_3 \leq 0$

$vars(A) = \{x_1, x_2, x_3\}$ $\quad vars(B) = \{x_1, x_3, x_4\}$ $\quad vars(\mathrm{I}) \subseteq \{x_1, x_3\}$

$\mathrm{I} = x_1 + x_3 \leq 2$

# Interpolation chain

We can extend the definition of interpolant to our setting

## Definition 5.6

*Consider unsat formula* $\bigwedge_{i \in 1..m} e_i(V_{i-1}, V_i)$. *An* *interpolant chain* *is a sequence of formulas such that* $\mathrm{I}_0 \ldots \mathrm{I}_m$ *such that*

- $\mathrm{I}_0 = \top$
- $\forall i \in 1..m \; \mathrm{I}_{i-1} \wedge e_i(V_{i-1}, V_i) \Rightarrow \mathrm{I}_i$
- $\mathrm{I}_m = \bot$
- $vars(\mathrm{I}_i) \subseteq V_i$

# Interpolation for refinement

We compute interpolation chain $\mathrm{I}_0 \dots \mathrm{I}_m$ for *pathCons*(*cons*)

*preds* $=$ atoms in $\mathrm{I}_0 \dots \mathrm{I}_m$ after erasing subscripts in its variables

Add *preds* in the predicate domain to obtain the refined abstract domain

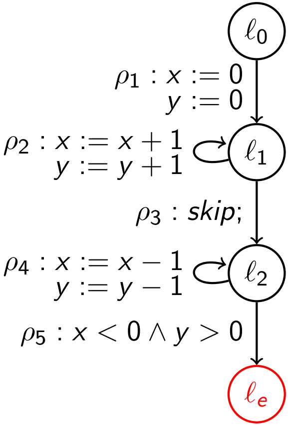
**Theorem 5.6**

*The new abstract domain eliminates spurious counterexample* $e_1 \dots e_n$

# Example: interpolation for refinement

Spurious counterexample: $\rho_1\rho_2\rho_3\rho_4\rho_4\rho_5$.

Program:




$\top$

$\rho_1(x_0, y_0, x_1, y_1) = (x_1 = 0 \land y_1 = 0)$

$\mathrm{I}_1 = y_1 \leq 0$

$\rho_2(x_1, y_1, x_2, y_2) = (x_2 = x_1 + 1 \land y_2 = y_1 + 1)$

$\mathrm{I}_2 = y_2 \leq 1$ $\leftarrow$ New predicate

$\rho_3(x_2, y_2, x_3, y_3) = (x_3 = x_2 \land y_3 = y_2)$

$\mathrm{I}_3 = y_3 \leq 0$

$\rho_4(x_3, y_3, x_4, y_4) = (x_4 = x_3 - 1 \land y_4 = y_3 - 1)$

$\mathrm{I}_4 = y_4 \leq 0$

$\rho_4(x_4, y_4, x_5, y_5) = (x_5 = x_4 - 1 \land y_5 = y_4 - 1)$

$\mathrm{I}_5 = y_5 \leq 0$

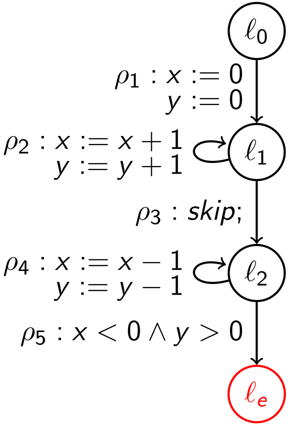
$\rho_5(x_5, y_5, x_6, y_6) = (x_5 < 0 \land y_5 > 0)$

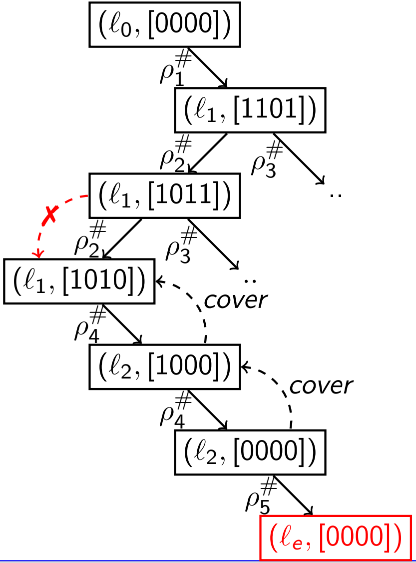
$\bot$

# Example: refined reachability graph

$$Preds = \{x \geq 0, y \leq 0, x \geq 1, y \leq 1\}$$

Program:



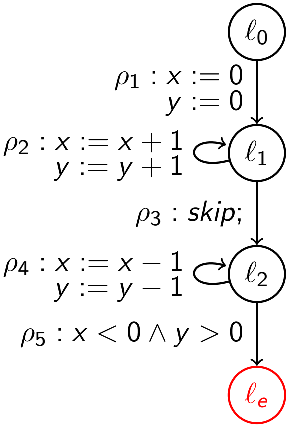




**Exercise 5.4**

*Complete the ARG*

# Example: good refinement

Program:




Consider the earlier spurious counterexample again: $\rho_1\rho_2\rho_3\rho_4\rho_4\rho_5$.

$\top$

$\rho_1(x_0, y_0, x_1, y_1) = (x_1 = 0 \wedge y_1 = 0)$

$\mathrm{I}_1 = y_1 \leq x_1$

$\rho_2(x_1, y_1, x_2, y_2) = (x_2 = x_1 + 1 \wedge y_2 = y_1 + 1)$

$\mathrm{I}_2 = y_2 \leq x_2$ $\leftarrow$ New predicate

$\rho_3(x_2, y_2, x_3, y_3) = (x_3 = x_2 \wedge y_3 = y_2)$

$\mathrm{I}_3 = y_3 \leq x_3$

$\rho_4(x_3, y_3, x_4, y_4) = (x_4 = x_3 - 1 \wedge y_4 = y_3 - 1)$

$\mathrm{I}_4 = y_4 \leq x_4$

$\rho_4(x_4, y_4, x_5, y_5) = (x_5 = x_4 - 1 \wedge y_5 = y_4 - 1)$

$\mathrm{I}_5 = y_5 \leq x_5$

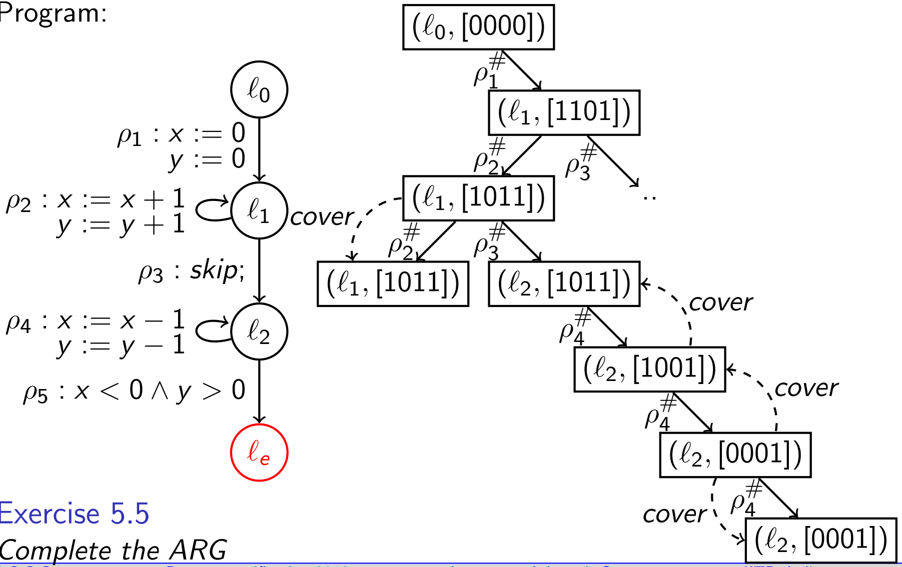
$\rho_5(x_5, y_5, x_6, y_6) = (x_5 < 0 \wedge y_5 > 0)$

$\bot$

# Example: ARG without spurious counterexample

$$Preds = \{x \geq 0, y \leq 0, x \geq 1, y \leq x\}$$

Program:




Exercise 5.5

*Complete the ARG*

# Observation on CEGAR

- Bad predicates waste our time
- We may reuse the ARG computed in last iteration
- Potential exponential blowup

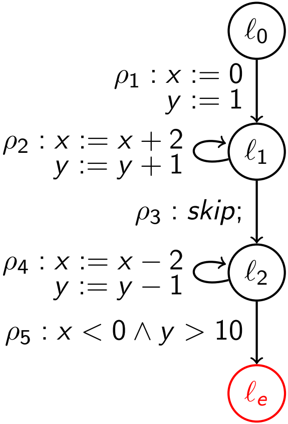
Topic 5.2

# Problems

# Abstract reachability graph

## Exercise 5.6

*Choose a set of predicates that will prove the following program correct and show the ARG of the program using the predicates.*



# CPAchecker

## Exercise 5.7

*Download CPAchecker:* `https://cpachecker.sosy-lab.org/`
*Apply the tool on the following example and report the generated ARG.*

```
int x=0; y=0; z=0; w=0;
while( * )) {
  if( * ) {
     x = x+1;
     y = y+100;
  }else if ( * ) {
    if (x >= 4) {
      x = x+1;
      y = y+1;
    }
  }else if (y > 10*w && z >= 100*x) {
     y = -y;
  }
  w = w+1;
  z = z+10;
}
if (x >= 4 && y <= 2)
  error();
```

# LTL to Bübhi

### Exercise 5.8

*Convert the following LTL formula into a Büchi automatom*

$$\Box\Diamond a \wedge \Diamond\Box b$$

# End of Lecture 5