Predicate Abstraction

Hossein Hojjat

LARA

April 1, 2011

Abstraction and Concretization

- $\mathcal{P} = \{P_0, \cdots, P_{n-1}\}$ finite set of predicates on program variables
- Example $\mathcal{P} = \{P_0, P_1, P_2\}$
  - $P_0 = \mathit{false}$, $P_1 = x > 0$, $P_2 = x < y$
- $\mathcal{S}$: concrete state of the program
- $\mathcal{A} = 2^{\mathcal{P}}$: abstract domain
- Abstraction function $\alpha : 2^{\mathcal{S}} \rightarrow \mathcal{A}$

$$\alpha(\psi) = \{P_i | \psi \rightarrow P_i \text{ is valid } \}$$

- Concretization function $\gamma : \mathcal{A} \rightarrow 2^{\mathcal{S}}$

$$\gamma(a) = \{c | \bigwedge_{p \in a} p(c)\}$$

- There are $2^{|\mathcal{P}|}$ abstract states
- Predicate abstraction is normally an over-approximation

Abstraction

Example

- Let $\mathcal{P} = \{x > y, x = 2\}$
- What is the abstraction after executing the following piece of code?

```
{true}
val x: Int
val y: Int
x = y + 1
```

- $\forall x, y, x', y' \in \mathbb{Z}.(x' = y + 1) \wedge (y' = y) \rightarrow (x' > y')$ (valid)
- $\forall x, y, x', y' \in \mathbb{Z}.(x' = y + 1) \wedge (y' = y) \rightarrow (x' = 2)$ (satisfiable)

The abstraction is $\{(x > y)\}$

Abstraction

Example

- Let $\mathcal{P} = \{x < 2\}$
- What is the abstraction after executing the following piece of code?

```
{x = 2}
if (x > 2) x = x - 1
```

- $\forall x, x'.(x = 2) \wedge (x > 2) \wedge (x' = x - 1) \rightarrow (x' < 2)$
  $\equiv \forall x, x'. \perp \rightarrow (x' < 2) \quad$ (valid)
- The abstraction in this case :$\{(x < 2)\}$
- How can we solve the problem?

Abstract Reachability Tree

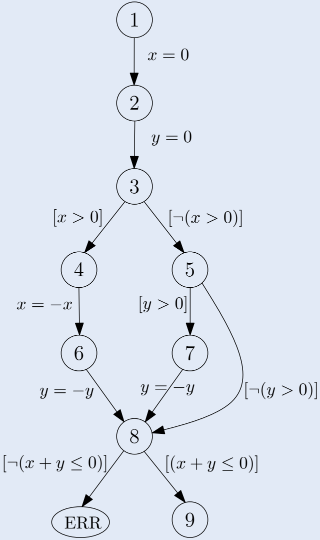
- Abstract state (l, ψ)
  - l: location in control flow graph
  - ψ: predicate abstraction

- Abstract reachability tree (ART) is a tree $G = (V_{\mathcal{A}}, \rightarrow, I)$
- $V_{\mathcal{A}}$ is a set of abstract states
- $\rightarrow \subseteq V_{\mathcal{A}} \times V_{\mathcal{A}}$ is the transition relation
  - Let c be the command between l_i and l_j in the CFG
  - $((l_i, \psi), (l_j, \phi)) \in \rightarrow$ if $\phi = sp^{\#}(\psi, c)$
- $I \in V_{\mathcal{A}}$ is the initial abstract state

- (l_i, ψ) is leaf if there exists another node (l_j, ϕ) in the tree such that $\phi \rightarrow \psi$

ART Example

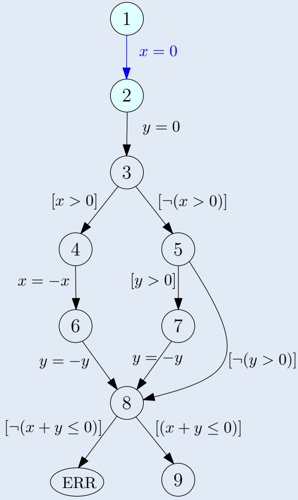
Control Flow Graph

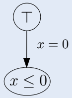



```
var x = 0
var y = 0
if( x > 0) {
  x = -x
  y = -y
} else {
    if( y > 0) y = -y
}
assert(x + y <= 0)
```

ART Example

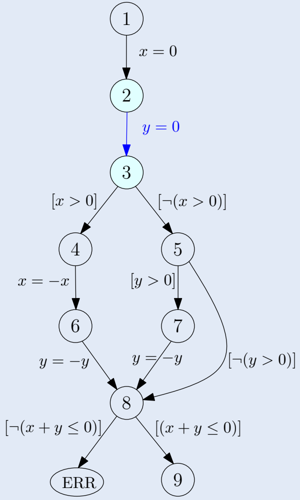
$$\mathcal{P} = \{\bot, (x <= 0), (x + y <= 0), (x - y <= 0)\}$$

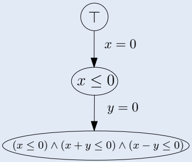




ART Example

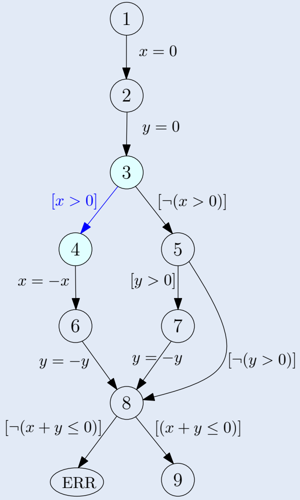
$$\mathcal{P} = \{\bot, (x <= 0), (x + y <= 0), (x - y <= 0)\}$$

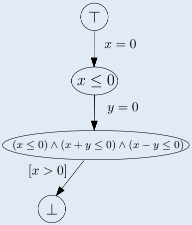




ART Example

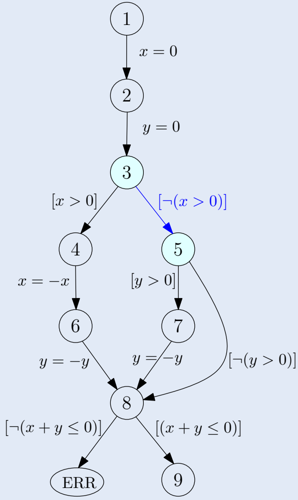
$$\mathcal{P} = \{\bot, (x <= 0), (x + y <= 0), (x - y <= 0)\}$$

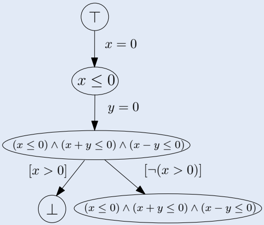




ART Example

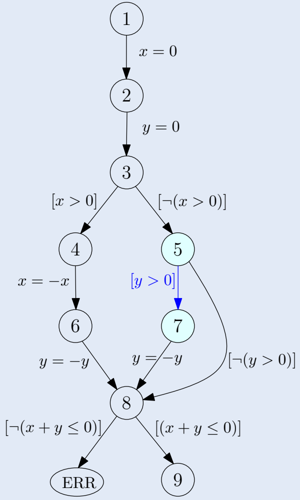
$$\mathcal{P} = \{\bot, (x <= 0), (x + y <= 0), (x - y <= 0)\}$$

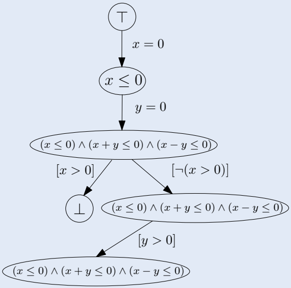




ART Example

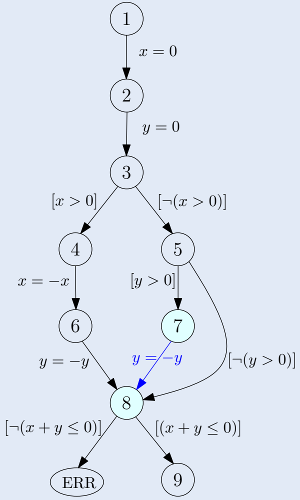
$$\mathcal{P} = \{\bot, (x <= 0), (x + y <= 0), (x - y <= 0)\}$$

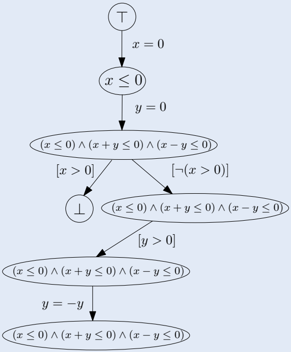




ART Example

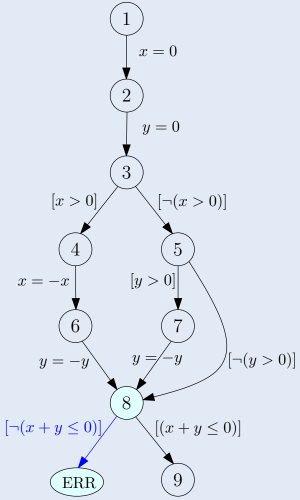
$$\mathcal{P} = \{\bot, (x <= 0), (x + y <= 0), (x - y <= 0)\}$$

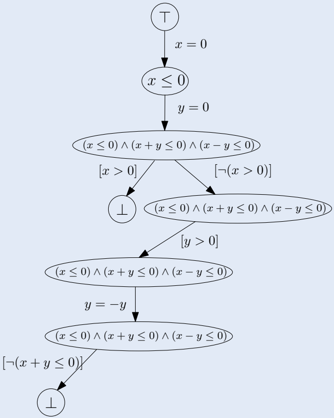




ART Example

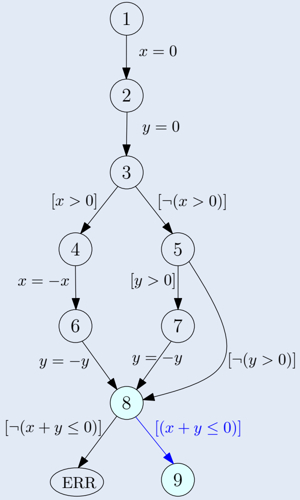
$$\mathcal{P} = \{\perp, (x <= 0), (x + y <= 0), (x - y <= 0)\}$$

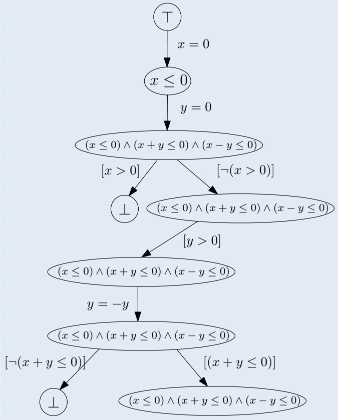




ART Example

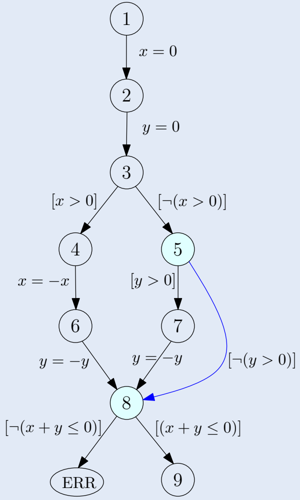
$$\mathcal{P} = \{\bot, (x <= 0), (x + y <= 0), (x - y <= 0)\}$$

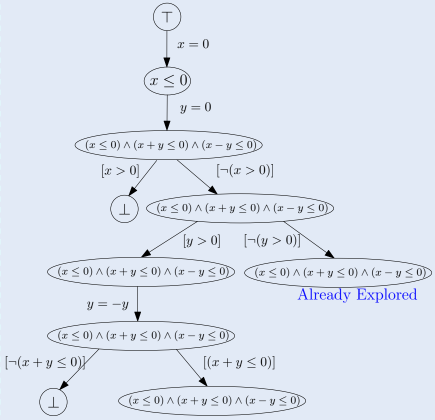




ART Example

$$\mathcal{P} = \{\bot, (x <= 0), (x + y <= 0), (x - y <= 0)\}$$



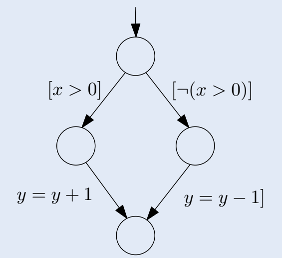


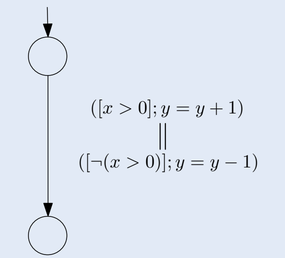
Large Block Encoding

- Idea: Squeeze loop-free subparts of program to single ART edges
- Reduce the size of ART
- Less theorem prover calls

- Fixpoint application of two summarizing rules
  - Sequence rule
  - Choice rule

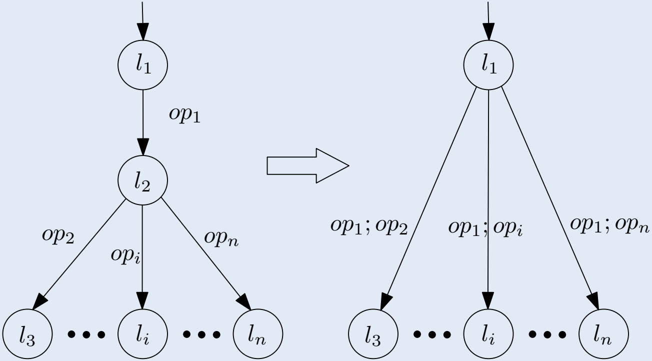
```
if( x > 0)
  y = y + 1
else   y = y - 1
```



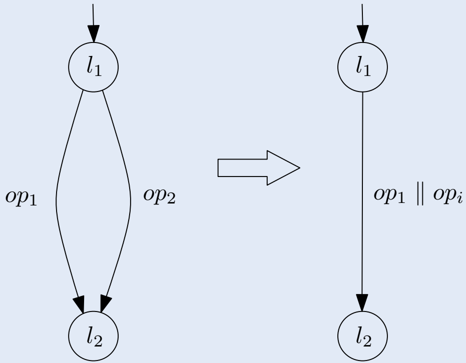


Sequence Rule

- $l_1 \neq l_2$
- No other incoming edges to l_2



Choice Rule



Demo