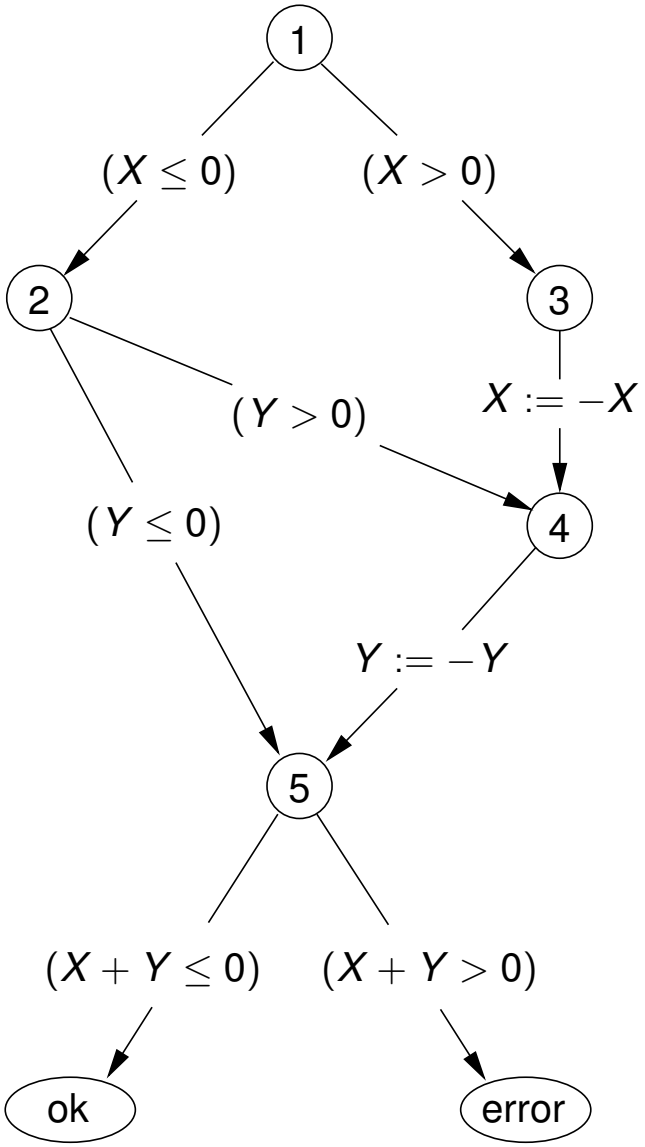
Part III: Abstraction Refinement

Javier Esparza

Technische Universität München

Example



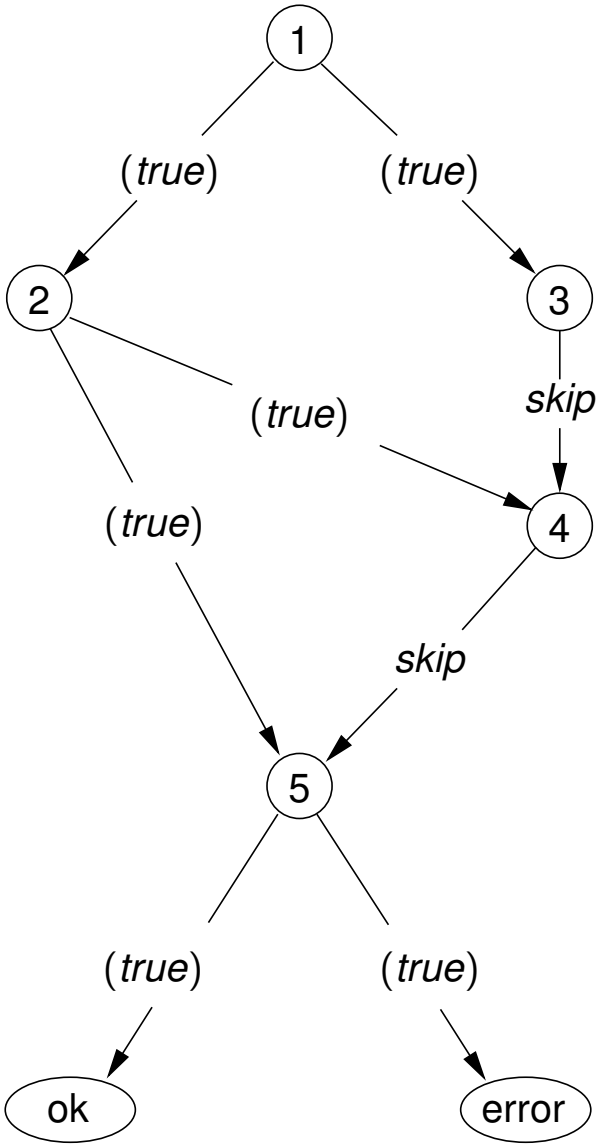
The problem:

- Is the error label reachable?

The approach:

- Upgrade a BDD checker with abstraction refinement

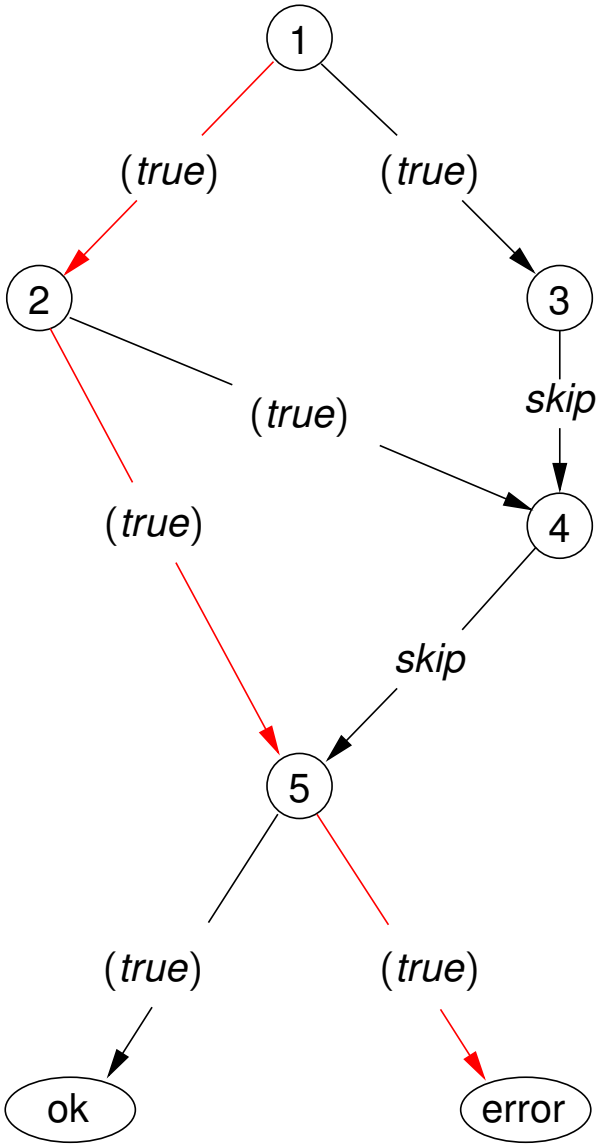
Example



Model-check the abstract program:

- Is the error label reachable considering only control flow?

Example

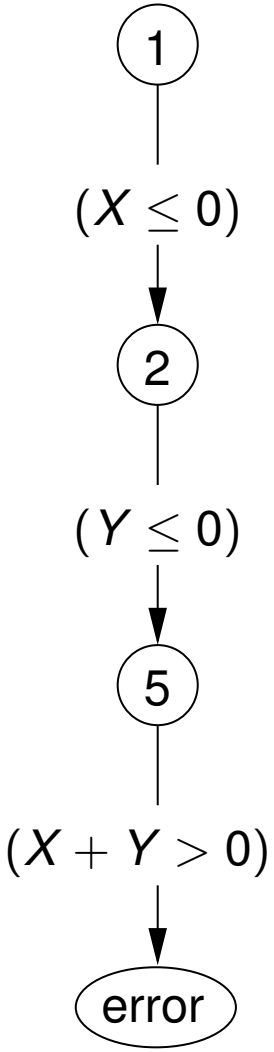


Model-check the abstract program:

- Is the error label reachable considering only control flow?

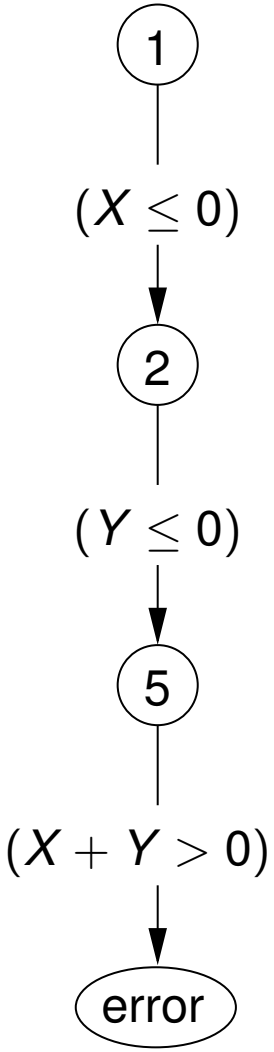
Yes!

Example



The concrete instructions are inserted again.

Example

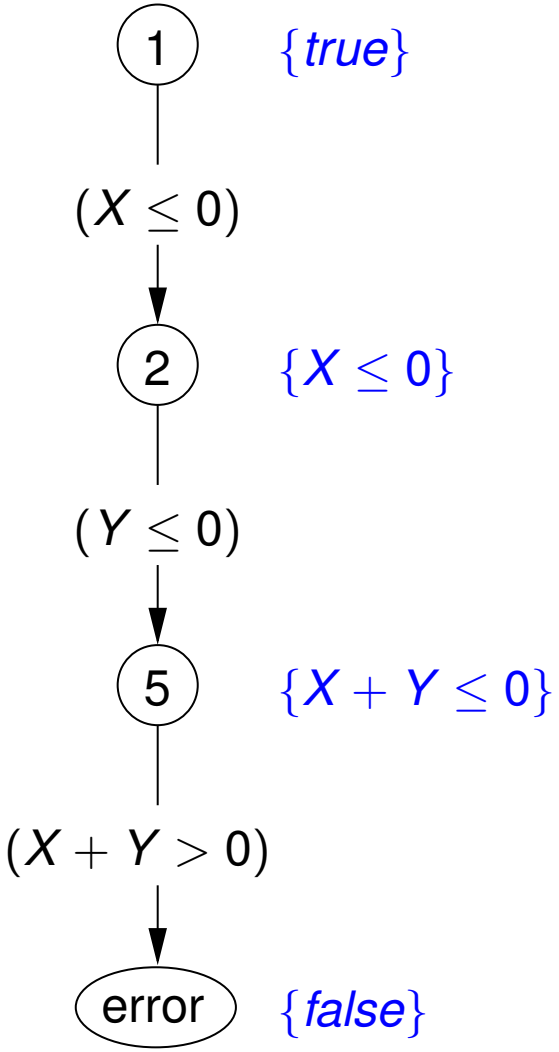


The concrete instructions are inserted again.

Analysis of the trace

- Is it real or spurious?

Example

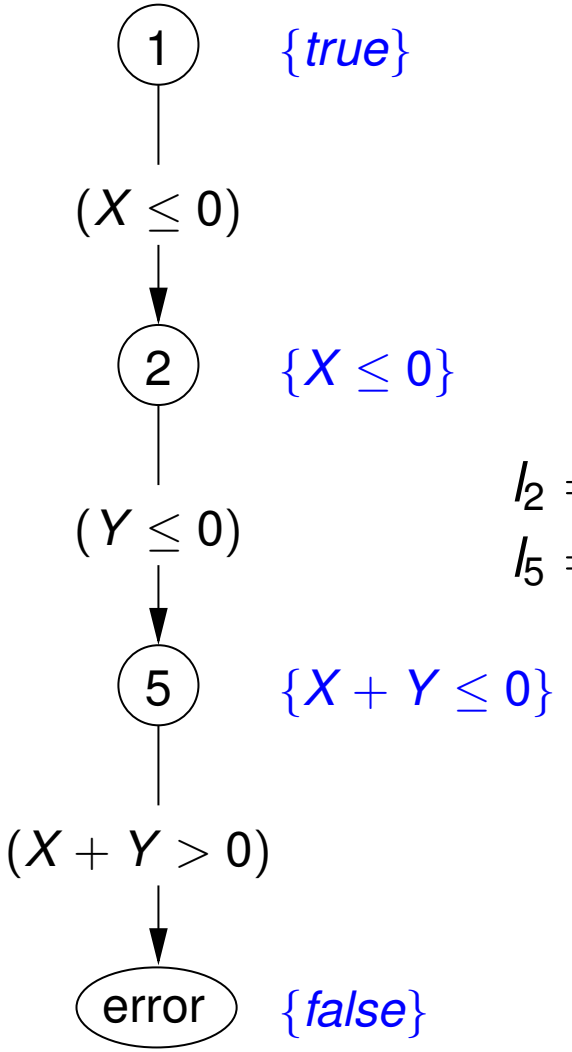


The concrete instructions are inserted again.

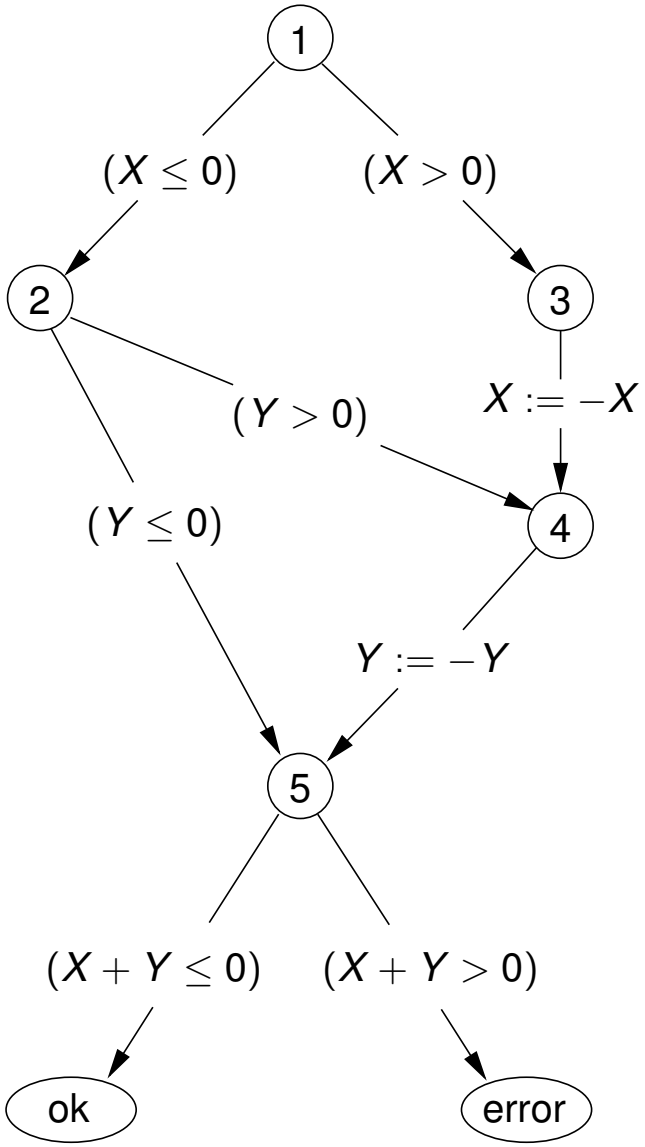
Analysis of the trace

- Is it real or spurious?
- Spurious! $\Rightarrow$ Hoare proof

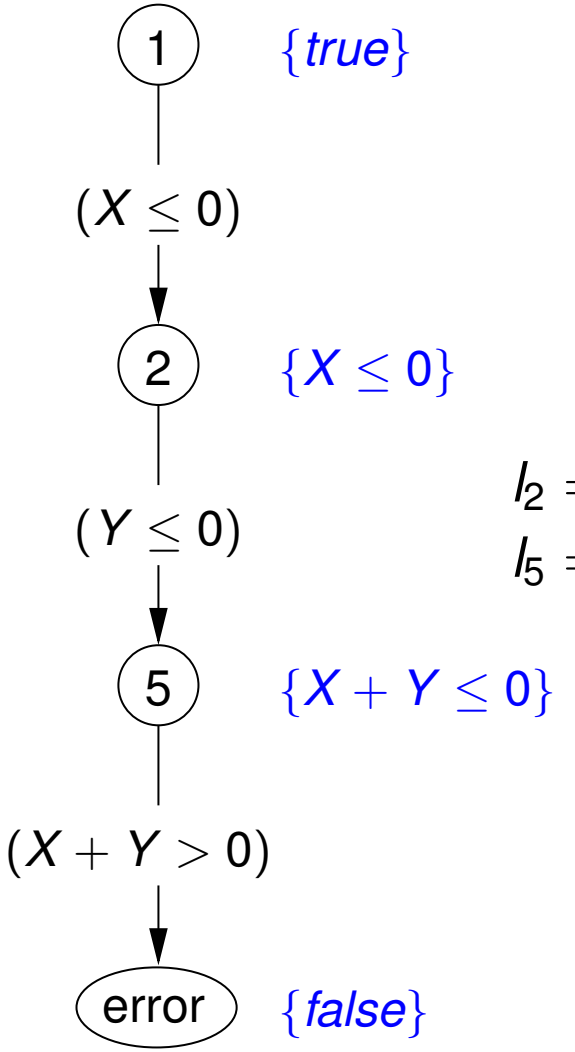
Example



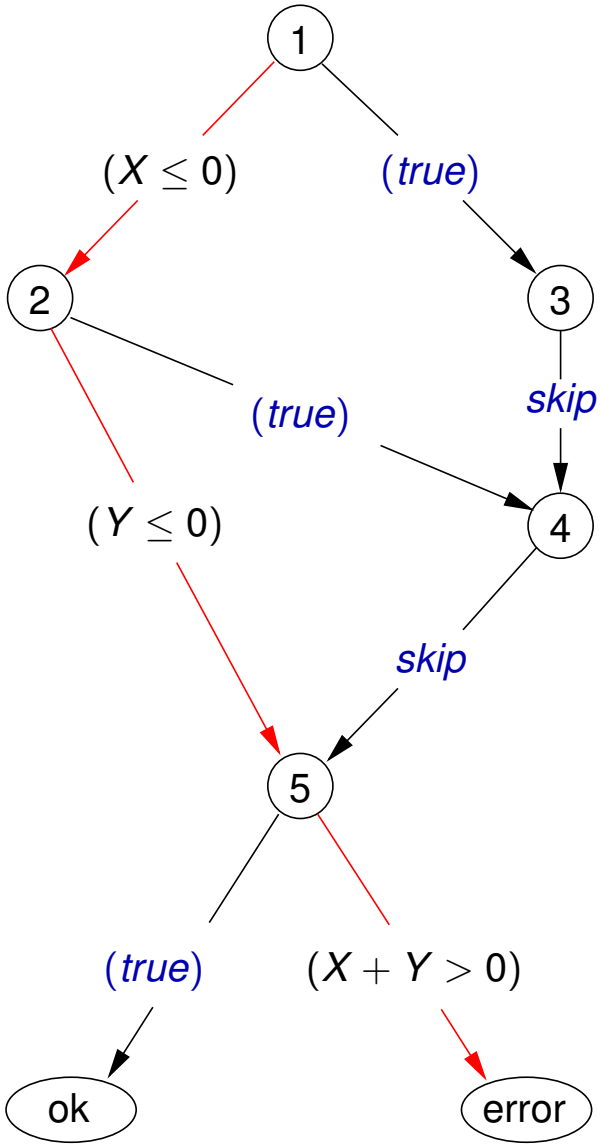
$$I_2 = (X \leq 0)$$
$$I_5 = (X + Y \leq 0)$$



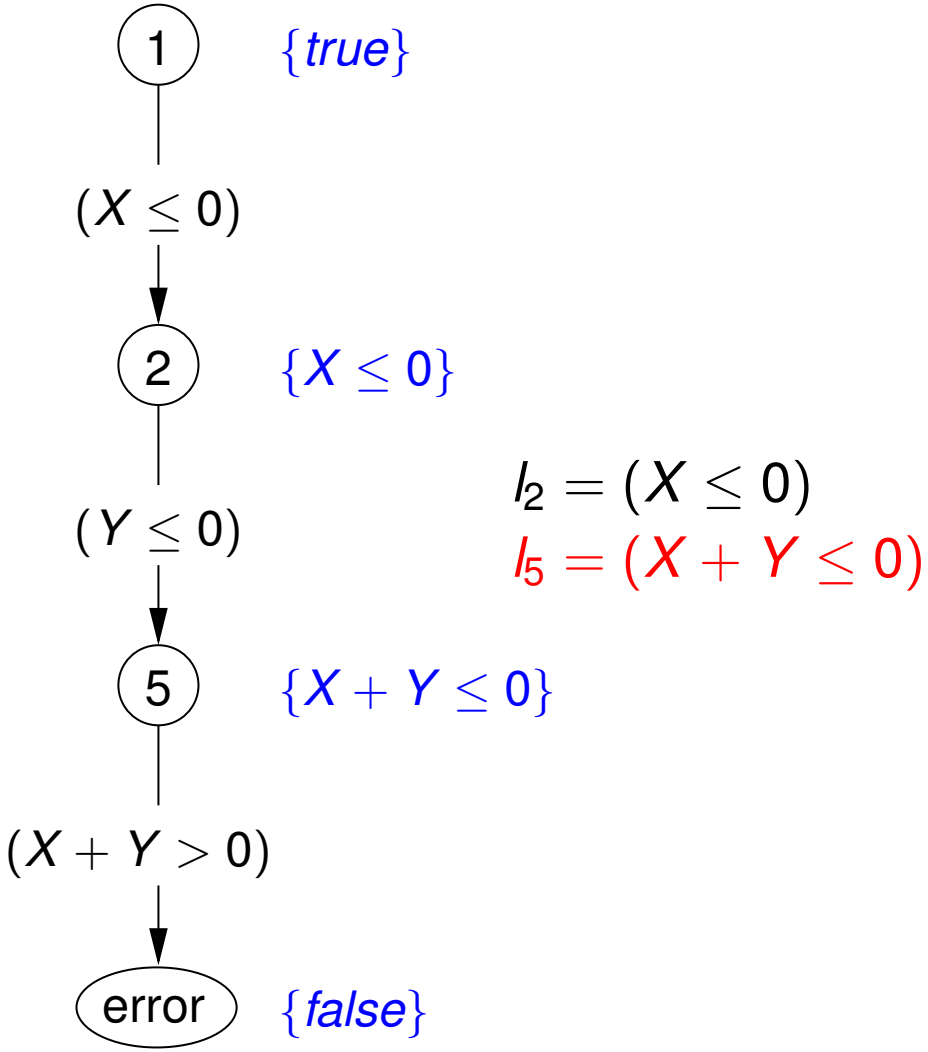
Example



$$l_2 = (X \leq 0)$$
$$l_5 = (X + Y \leq 0)$$

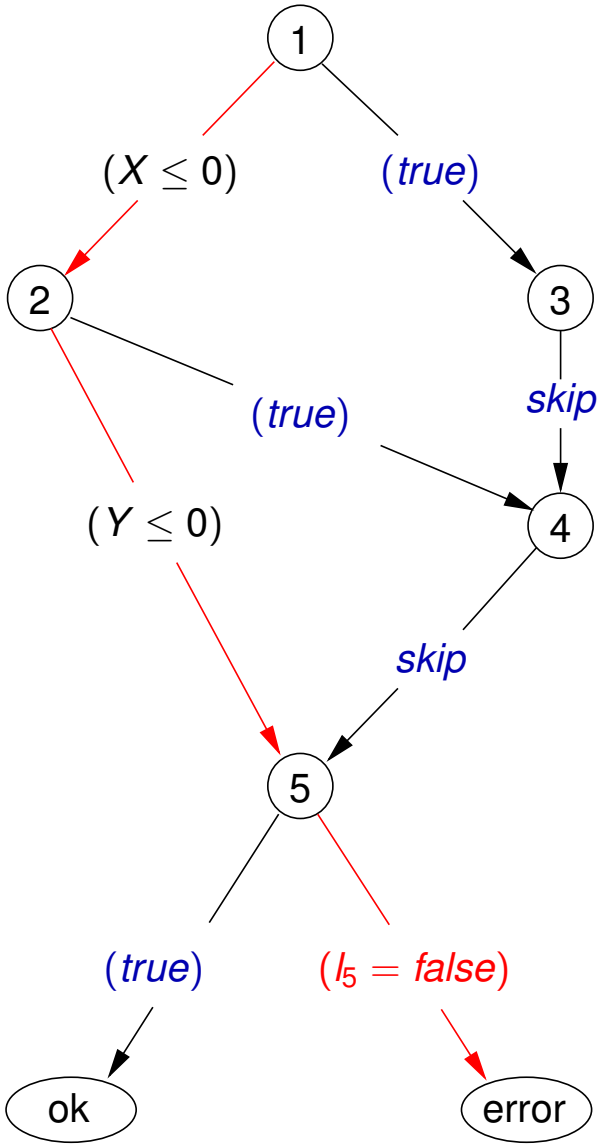


Example

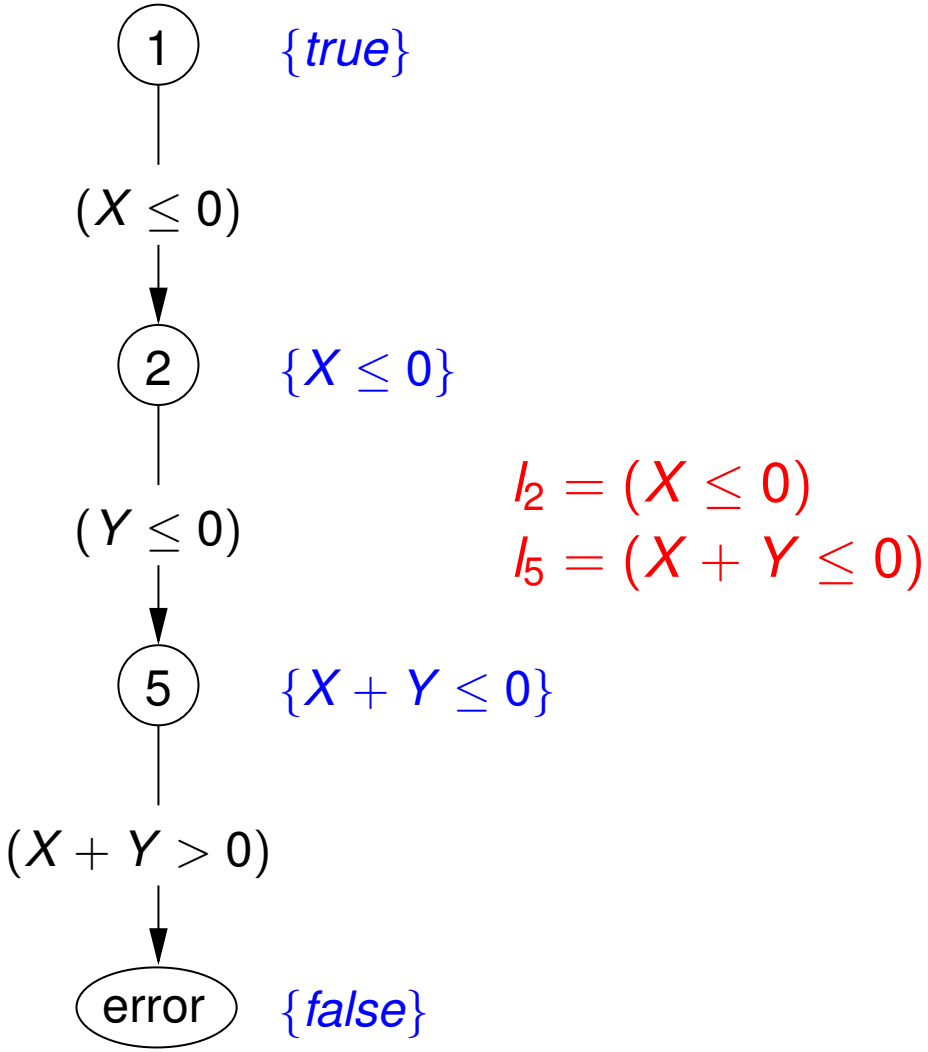


$l_2 = (X \leq 0)$

$l_5 = (X + Y \leq 0)$

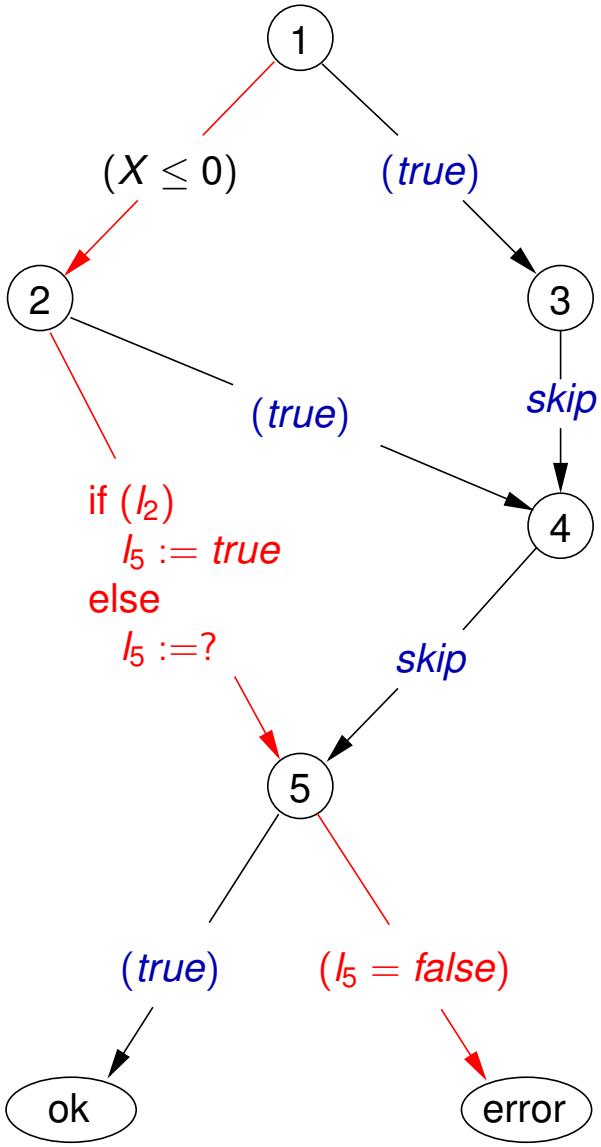


Example

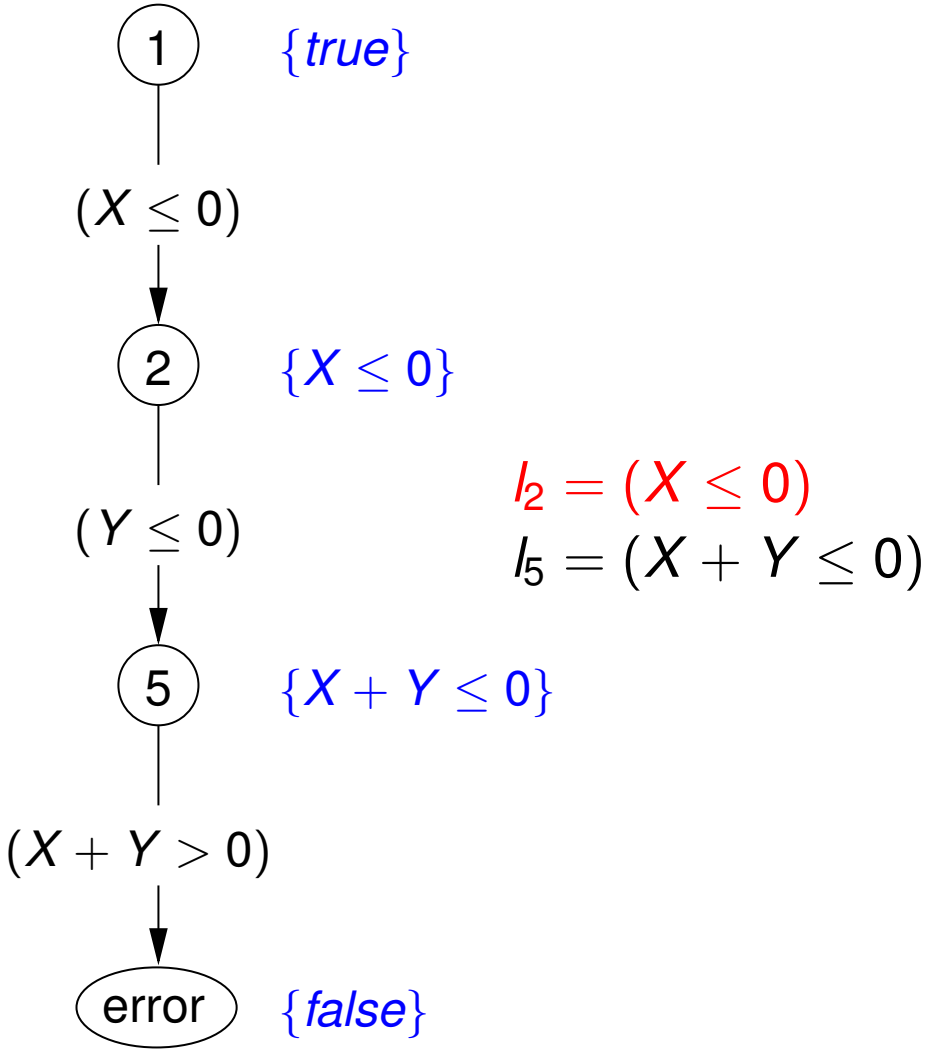


$l_2 = (X \leq 0)$

$l_5 = (X + Y \leq 0)$

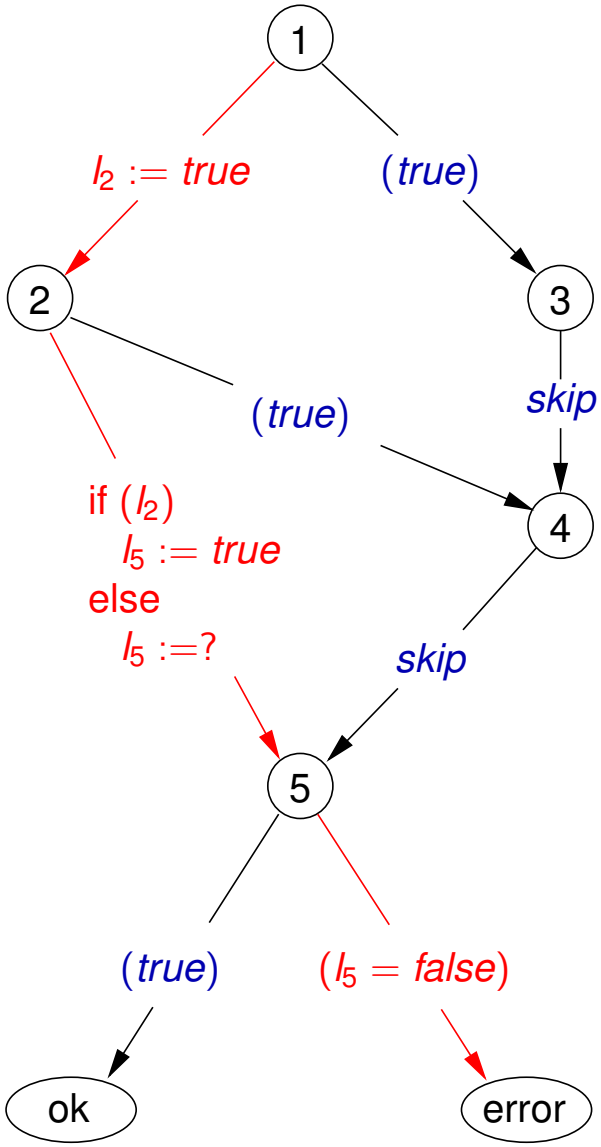


Example

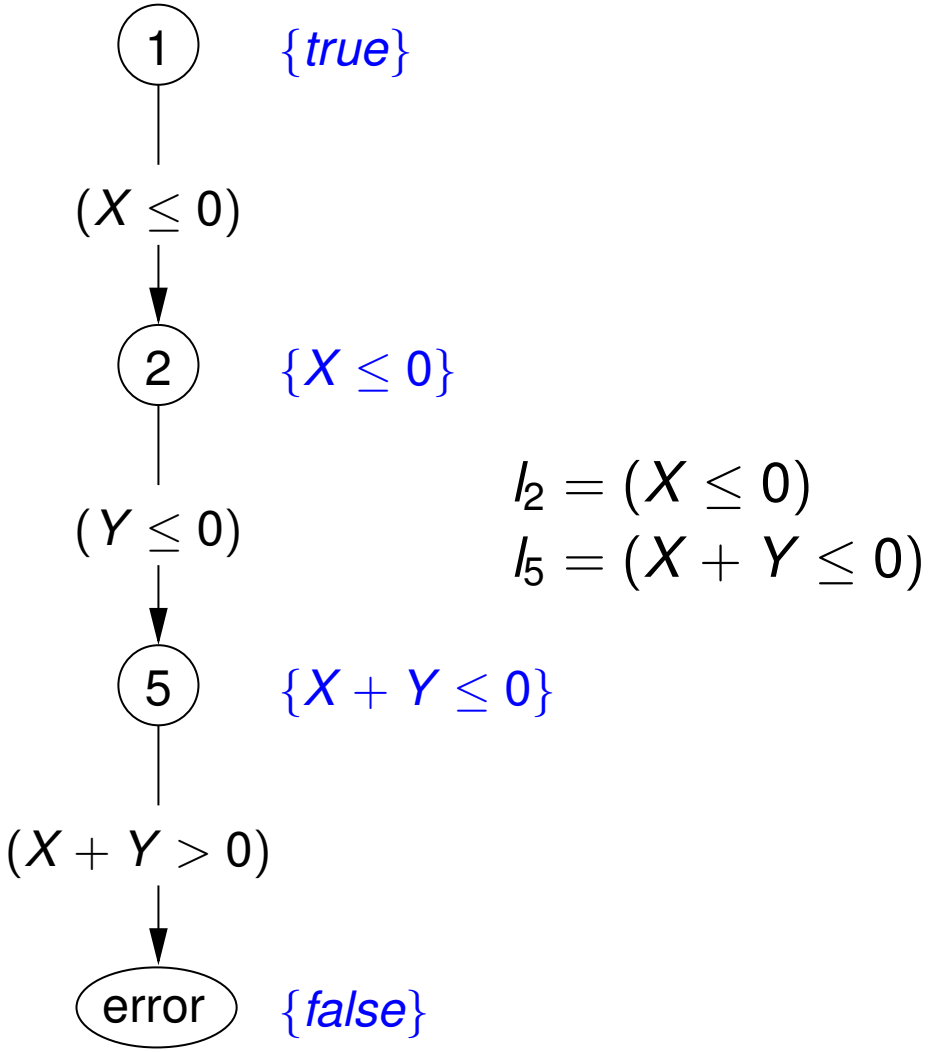


$l_2 = (X \leq 0)$

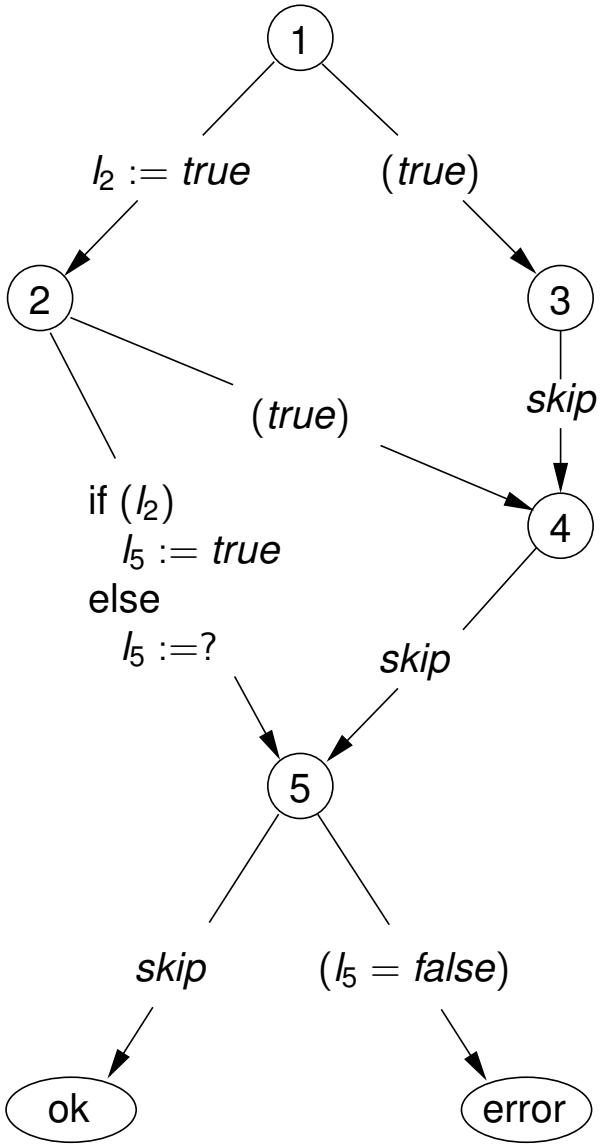
$l_5 = (X + Y \leq 0)$



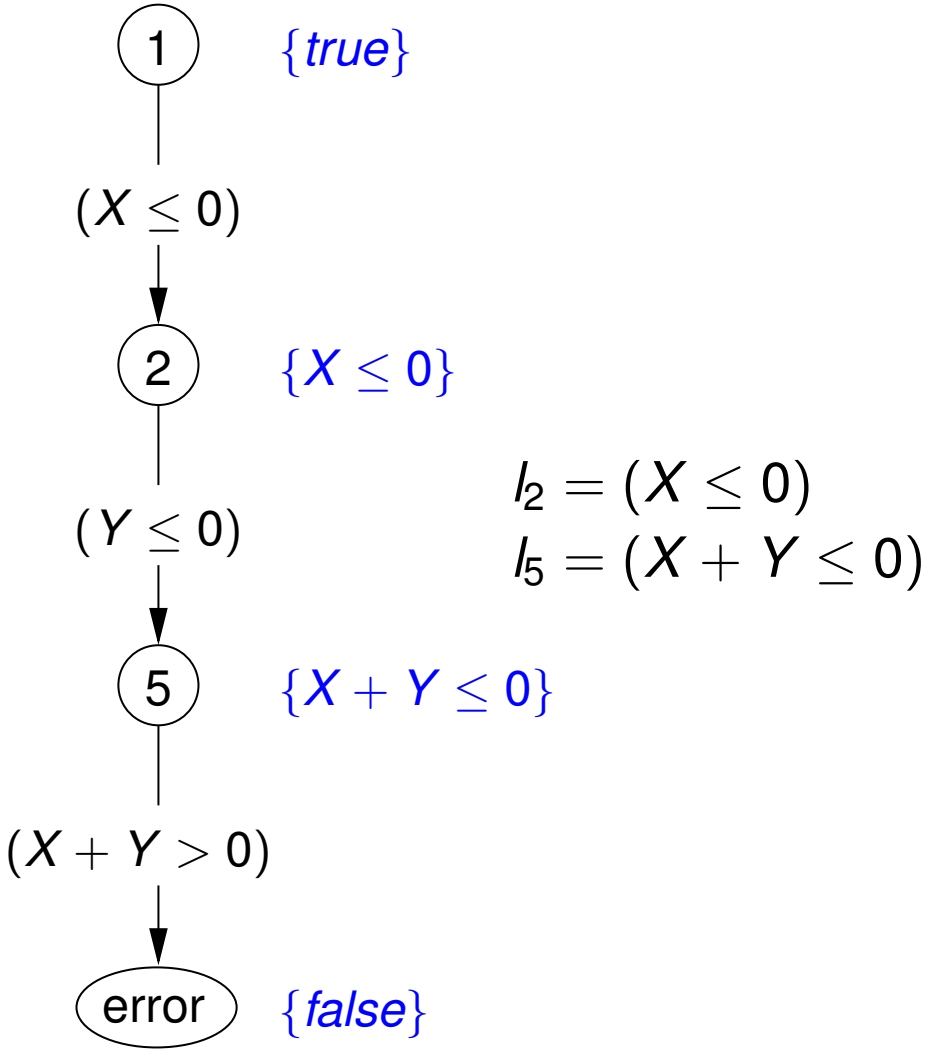
Example



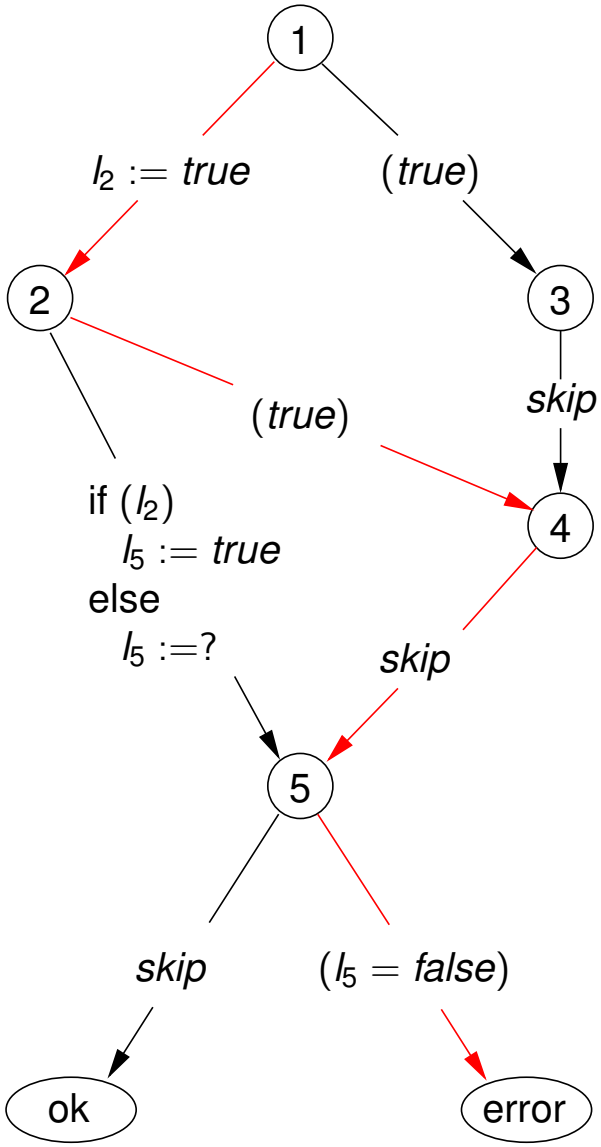
$$l_2 = (X \leq 0)$$
$$l_5 = (X + Y \leq 0)$$



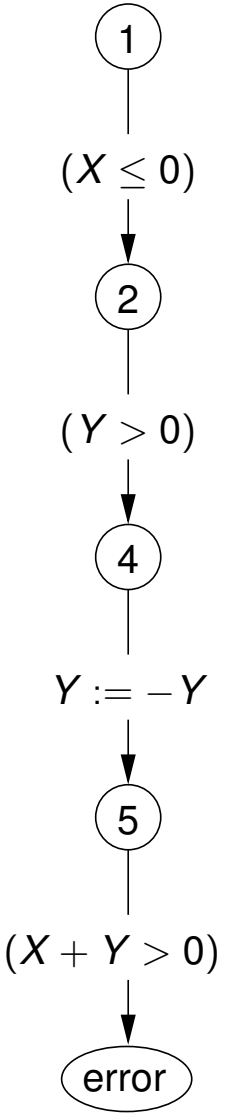
Example



$$l_2 = (X \leq 0)$$
$$l_5 = (X + Y \leq 0)$$

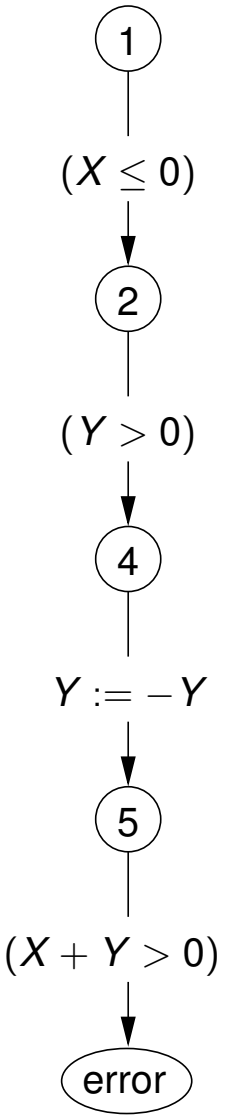


Example



The concrete instructions are inserted again.

Example

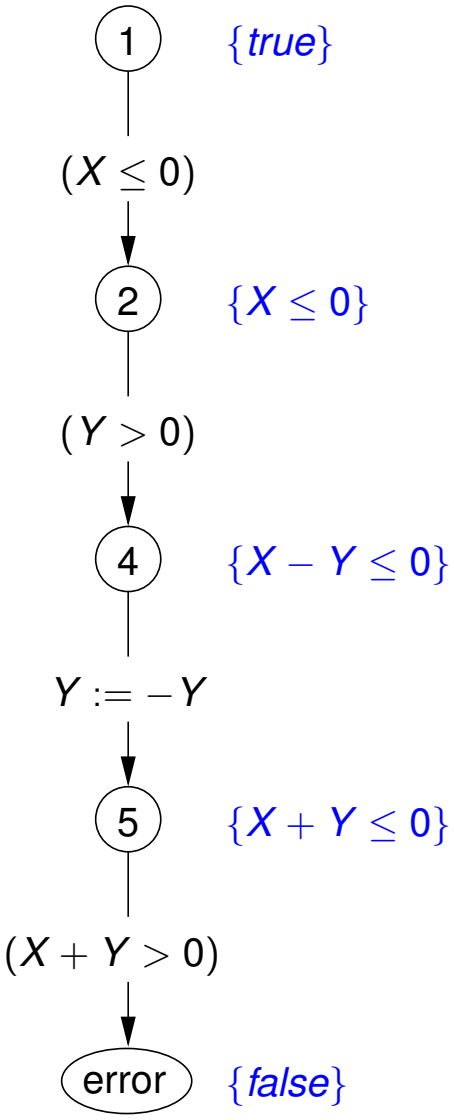


The concrete instructions are inserted again.

Analysis of the trace

- Is it real or spurious?

Example

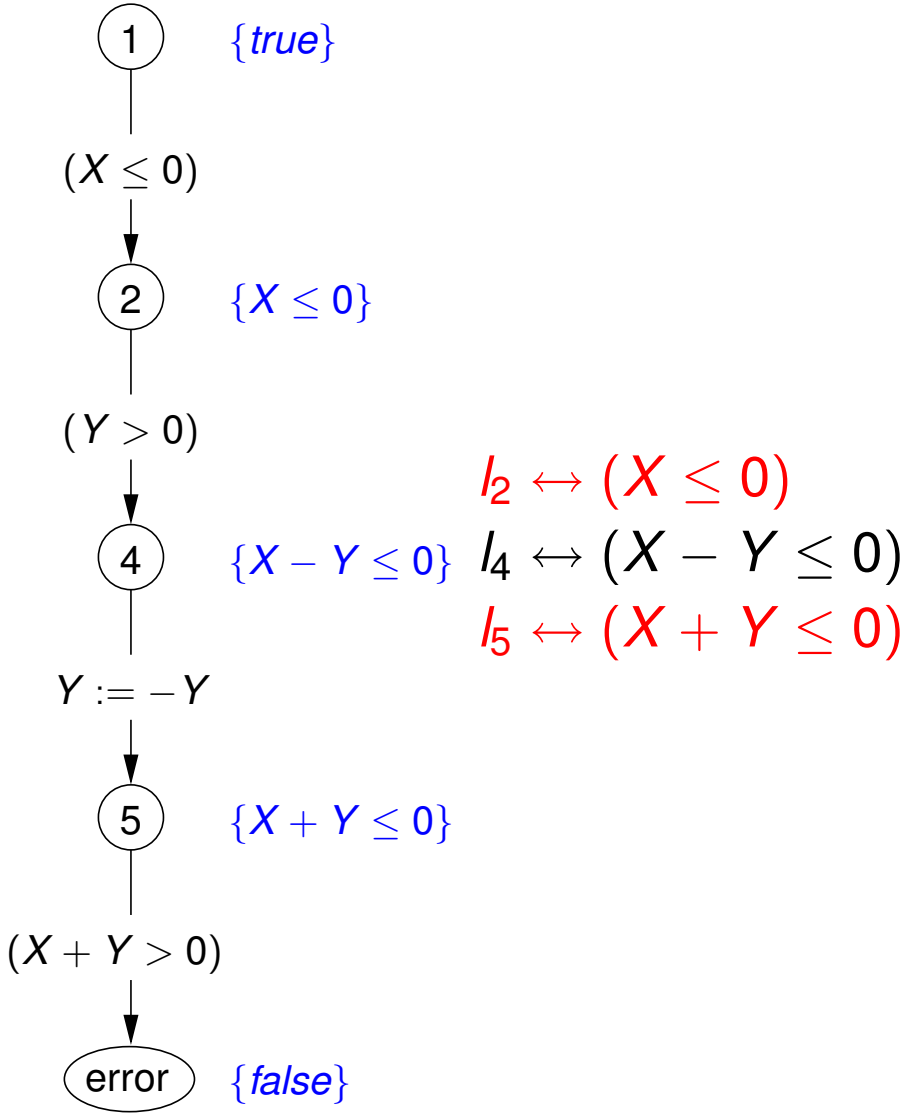


The concrete instructions are inserted again.

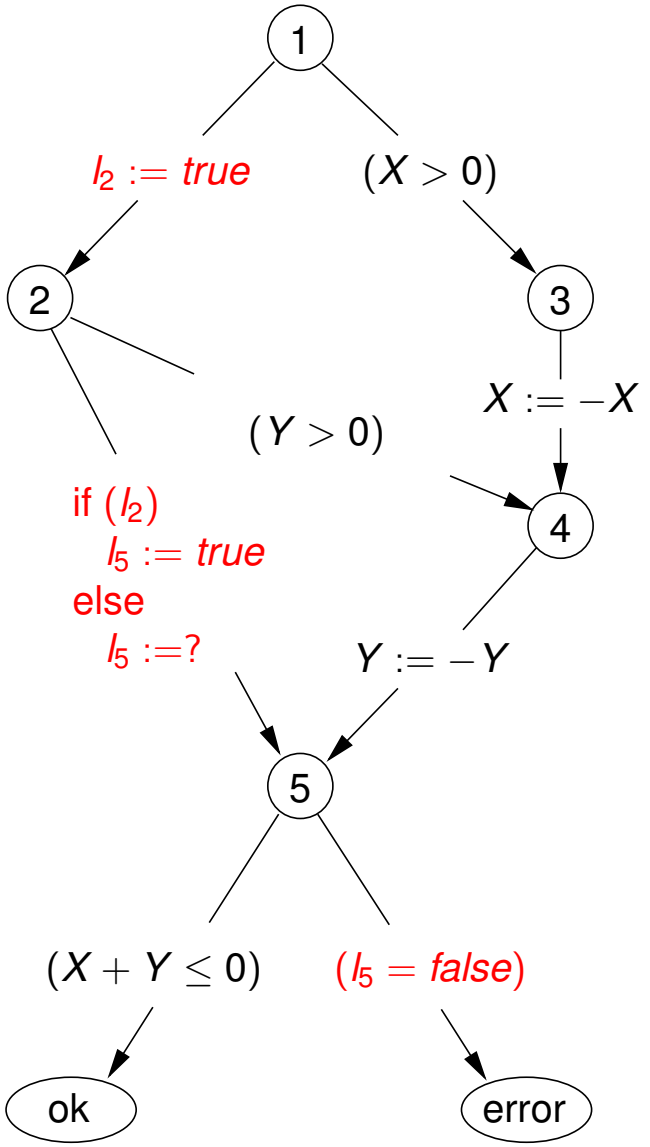
Analysis of the trace

- Is it real or spurious?
- Spurious! $\Rightarrow$ Hoare-like proof

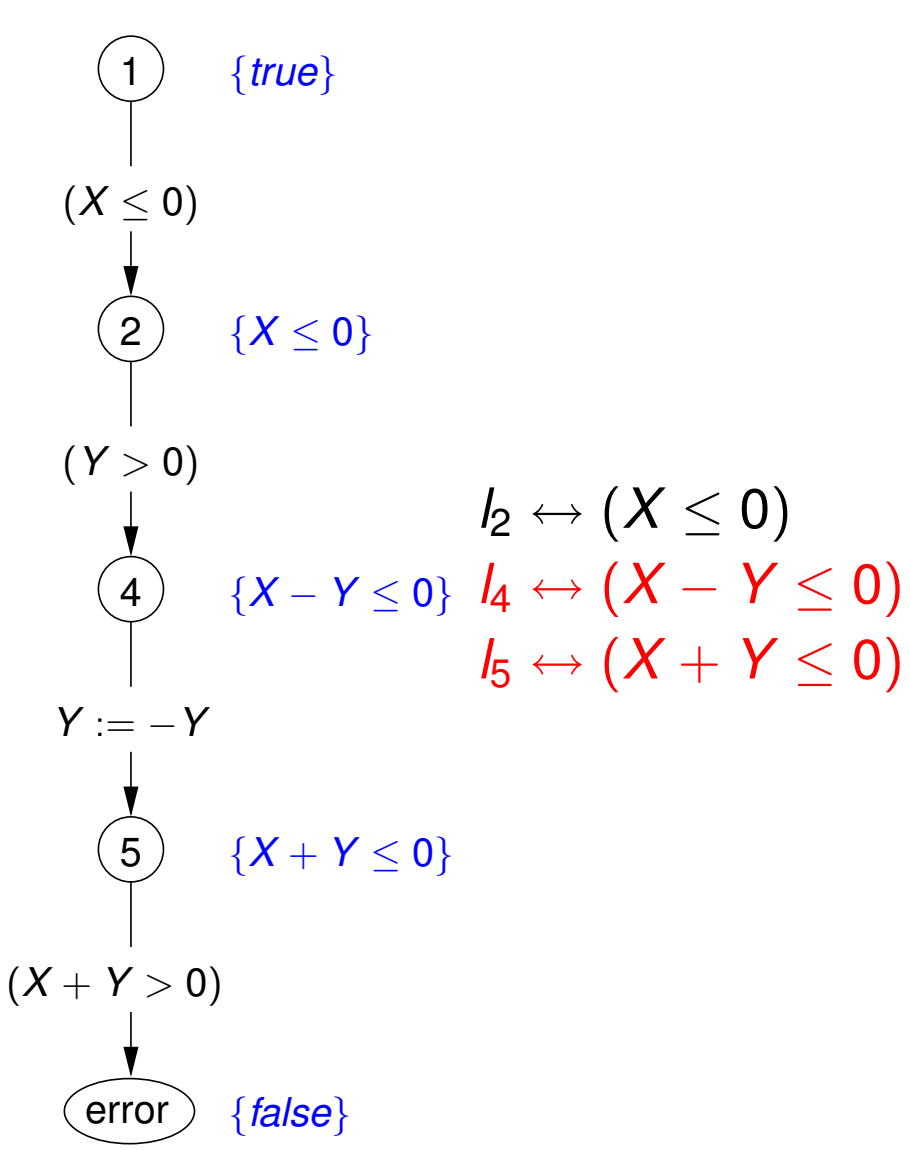
Example



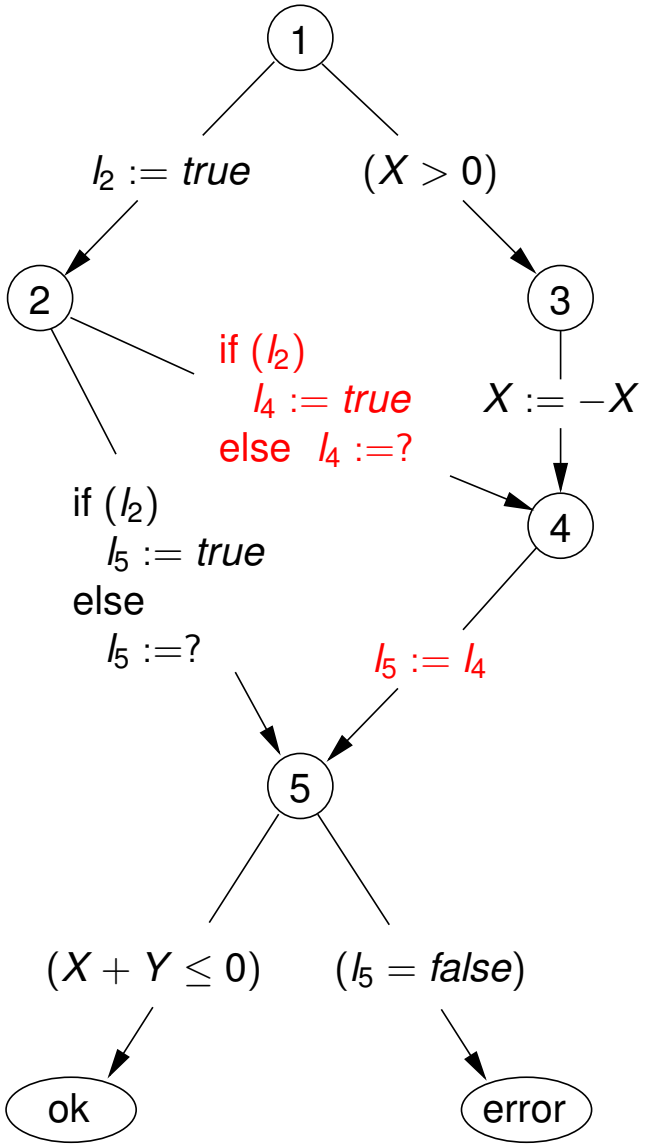
$$l_2 \leftrightarrow (X \leq 0)$$
$$l_4 \leftrightarrow (X - Y \leq 0)$$
$$l_5 \leftrightarrow (X + Y \leq 0)$$



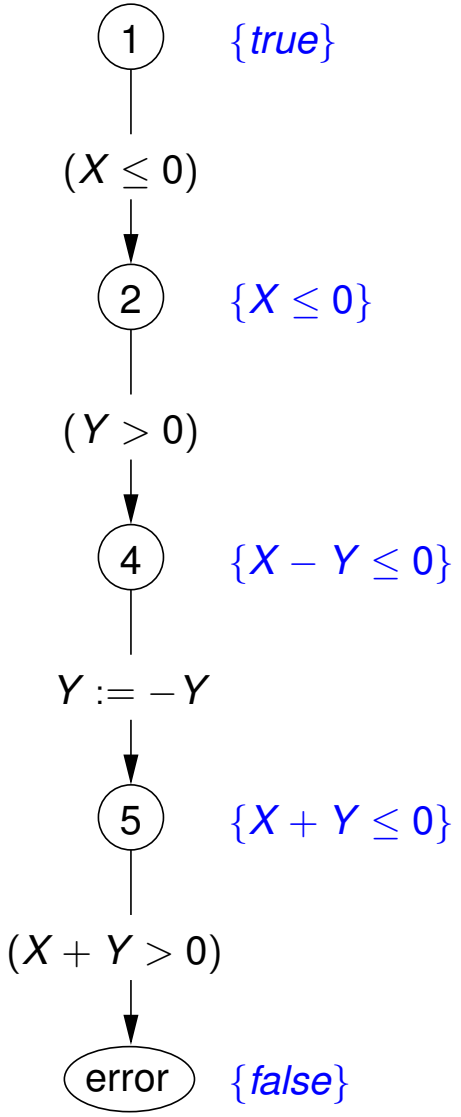
Example



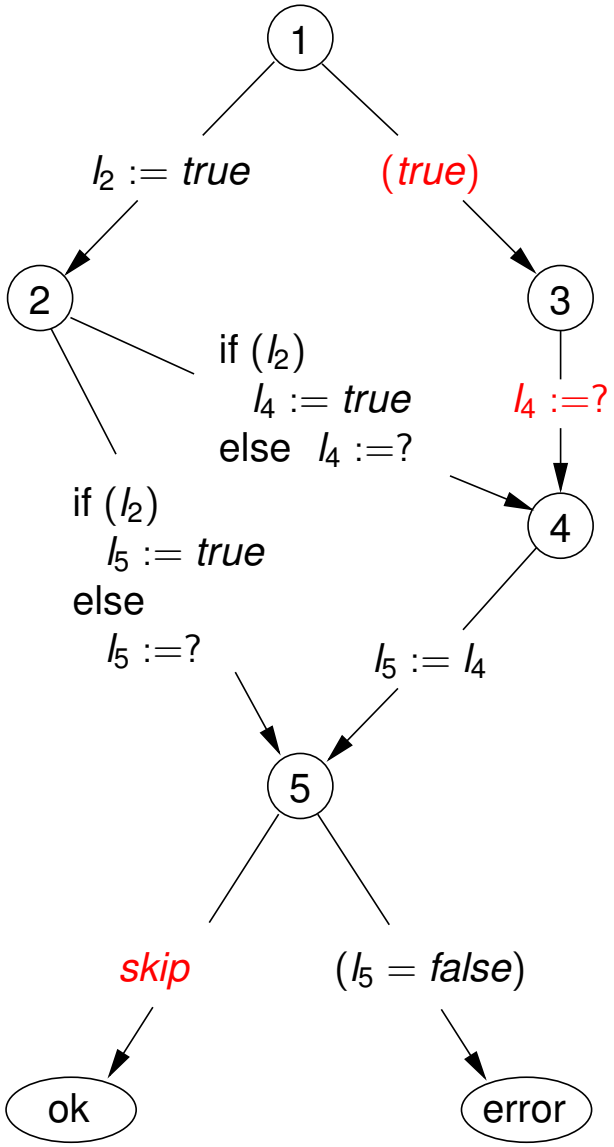
$$l_2 \leftrightarrow (X \leq 0)$$
$$l_4 \leftrightarrow (X - Y \leq 0)$$
$$l_5 \leftrightarrow (X + Y \leq 0)$$



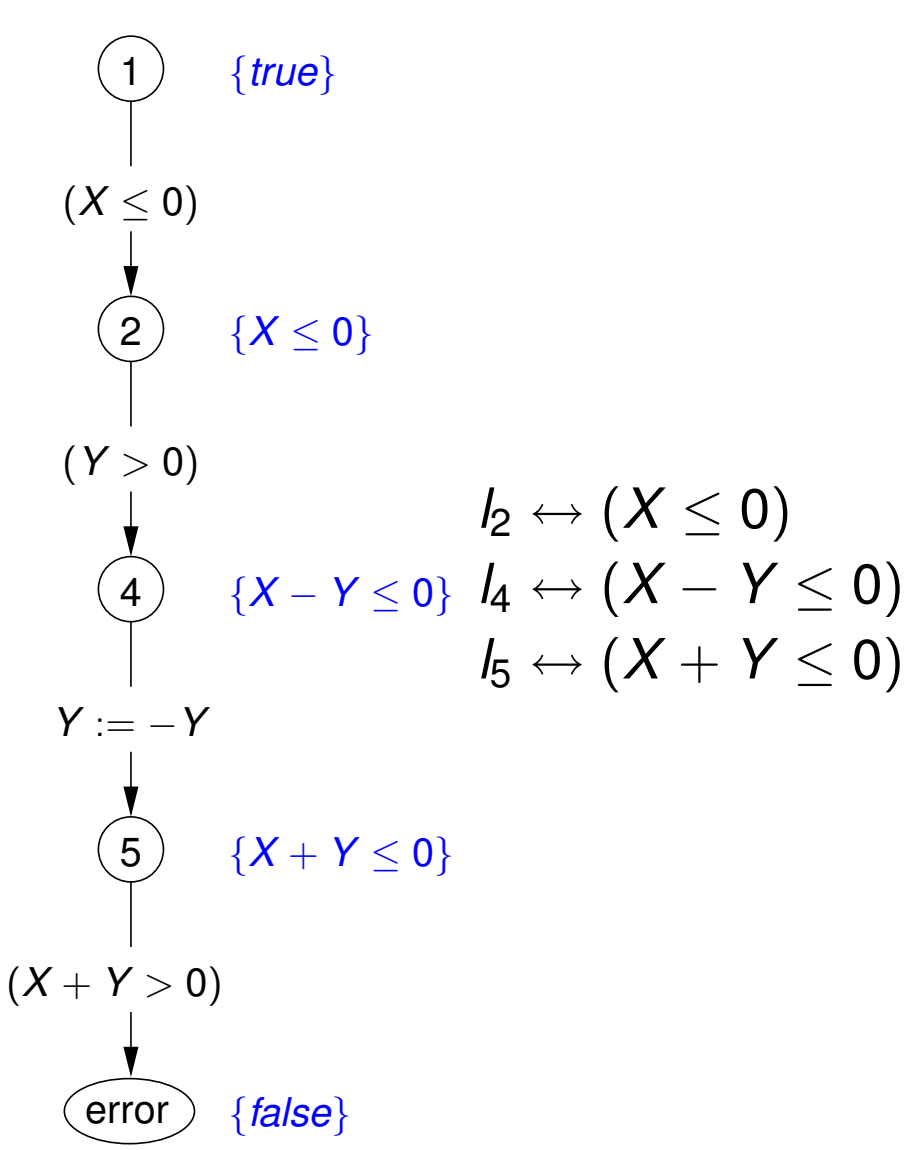
Example



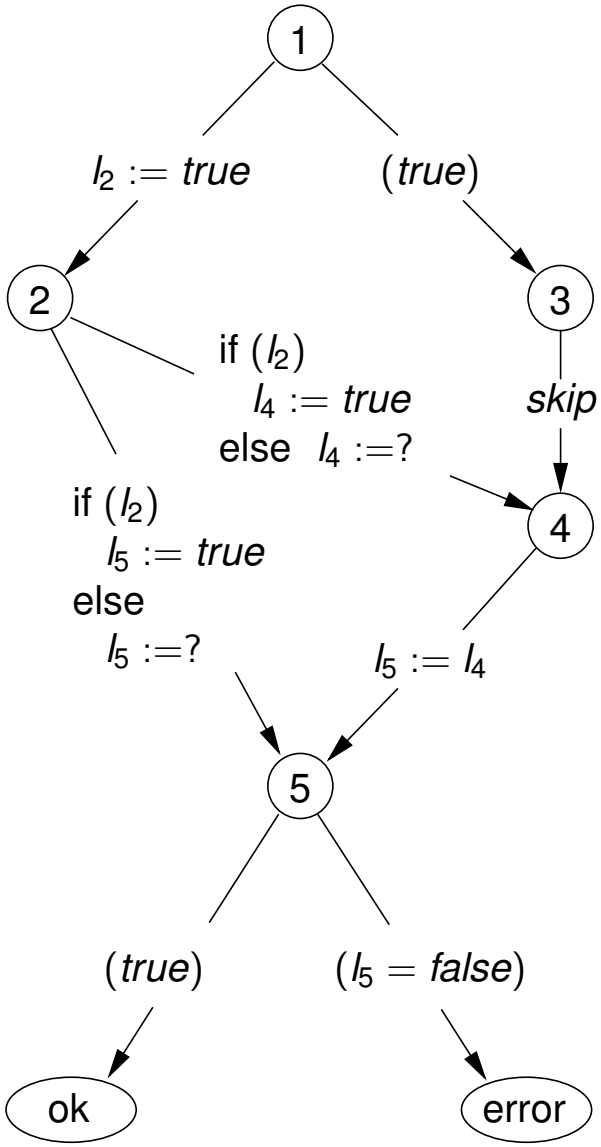
$$l_2 \leftrightarrow (X \leq 0)$$
$$l_4 \leftrightarrow (X - Y \leq 0)$$
$$l_5 \leftrightarrow (X + Y \leq 0)$$



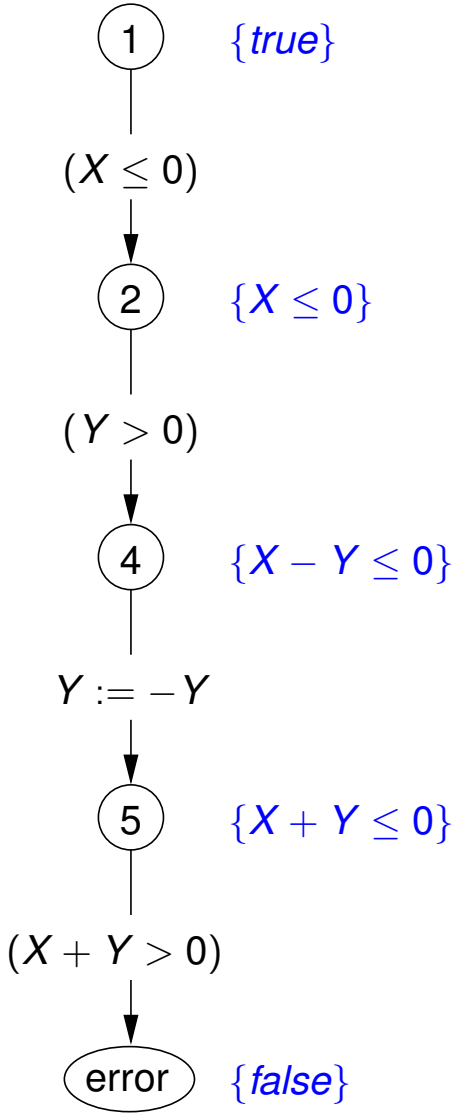
Example



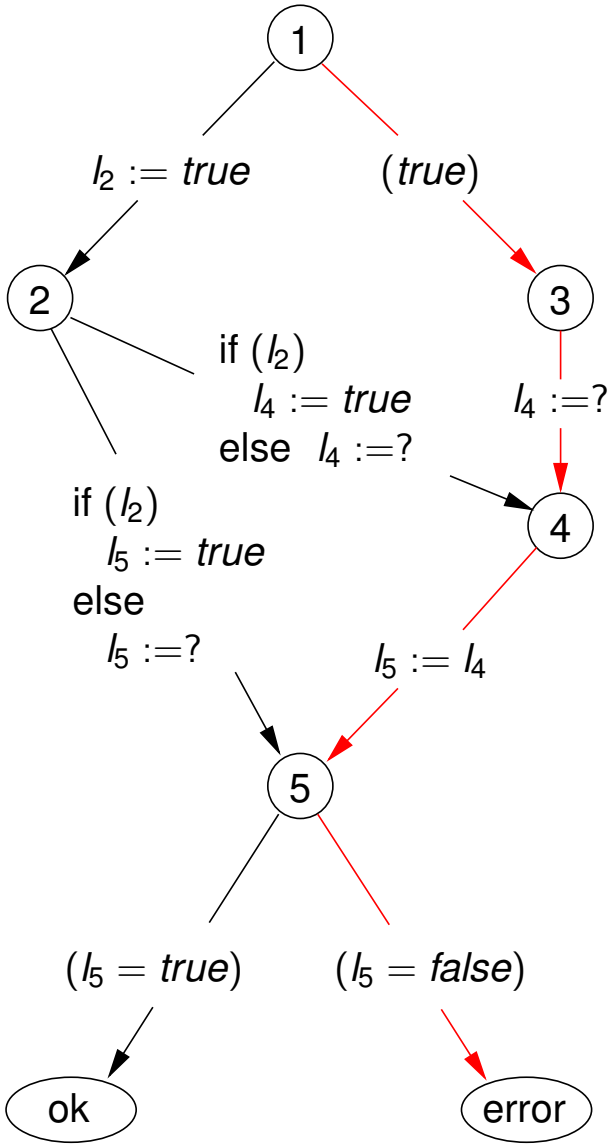
$$l_2 \leftrightarrow (X \leq 0)$$
$$l_4 \leftrightarrow (X - Y \leq 0)$$
$$l_5 \leftrightarrow (X + Y \leq 0)$$



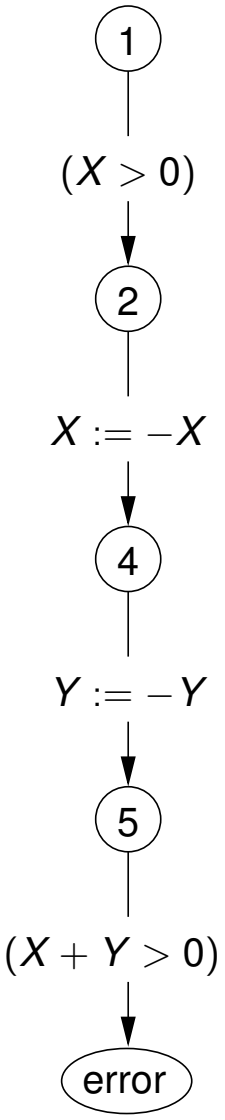
Example



$$l_2 \leftrightarrow (X \leq 0)$$
$$l_4 \leftrightarrow (X - Y \leq 0)$$
$$l_5 \leftrightarrow (X + Y \leq 0)$$

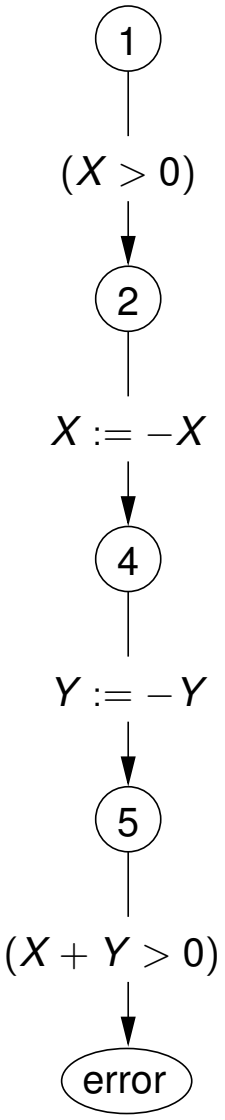


Example



The concrete instructions are inserted again.

Example

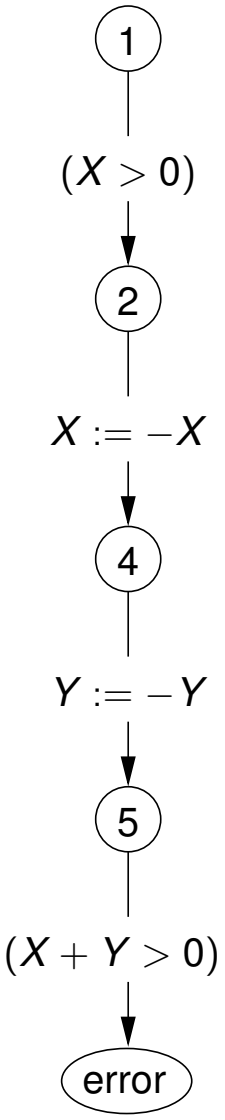


The concrete instructions are inserted again.

Analysis of the trace

- Is it real or spurious?

Example

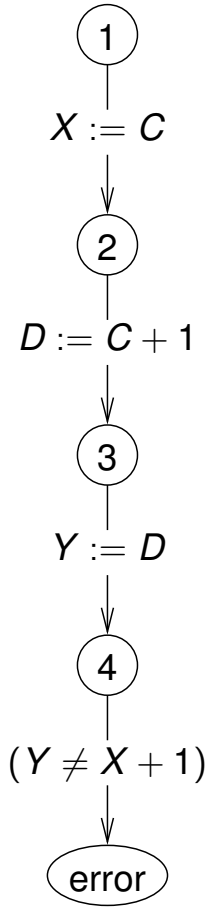


The concrete instructions are inserted again.

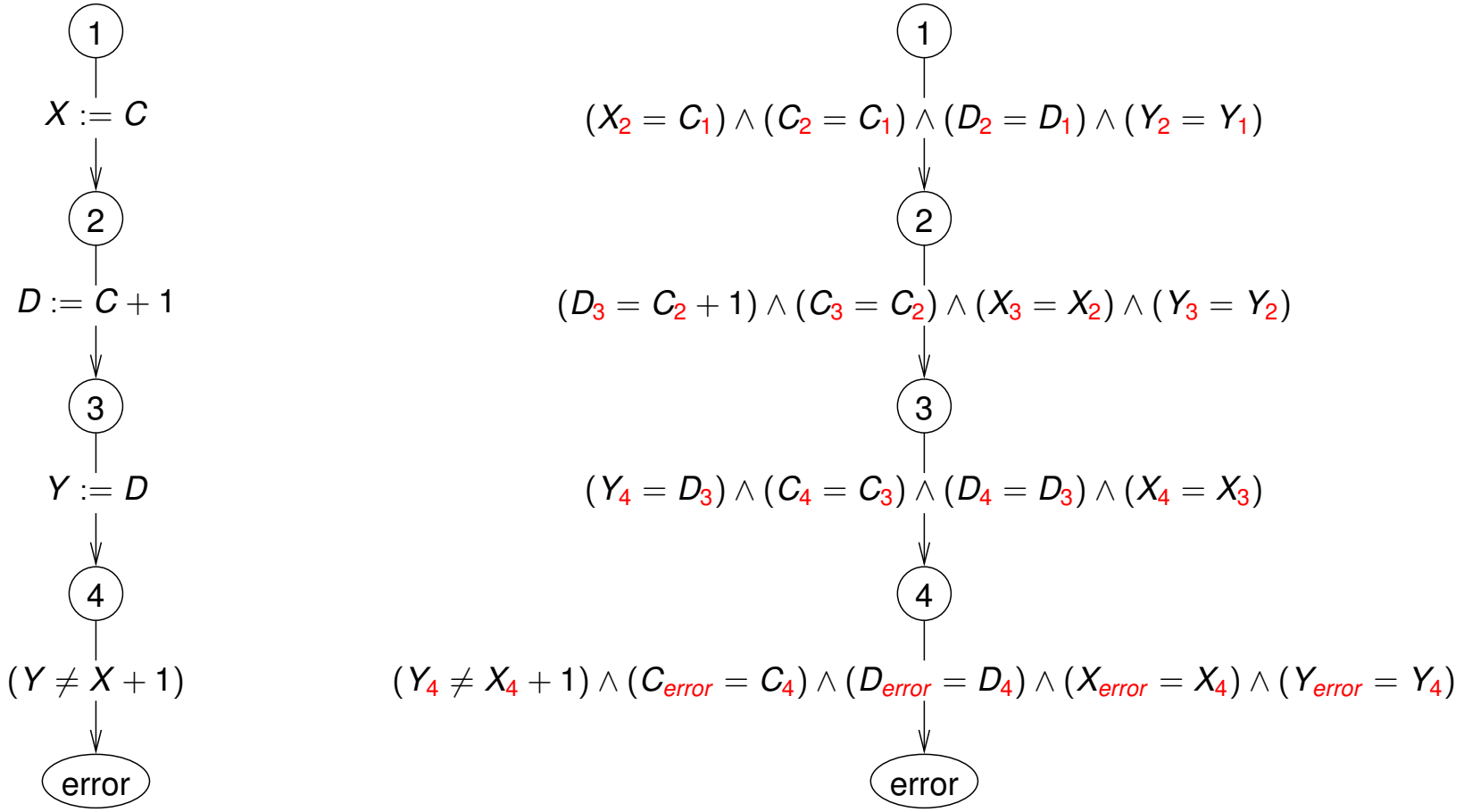
Analysis of the trace

- Is it real or spurious?
- Real! $\Rightarrow$ Report it to the user!

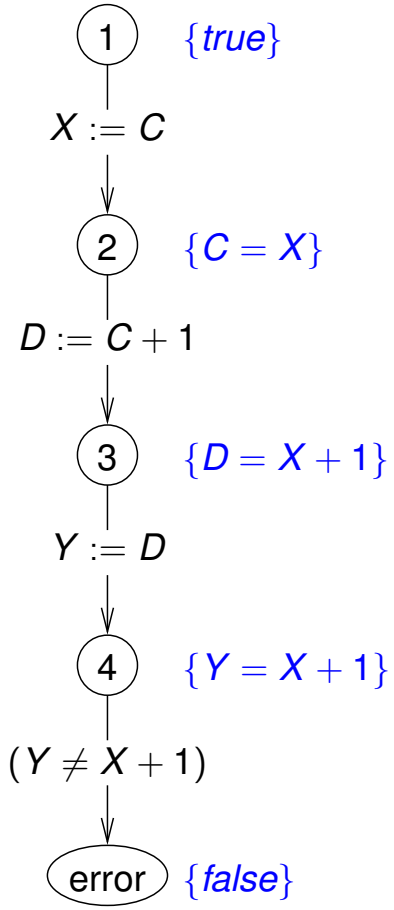
A Spurious Trace is an Unsatisfiable Formula.



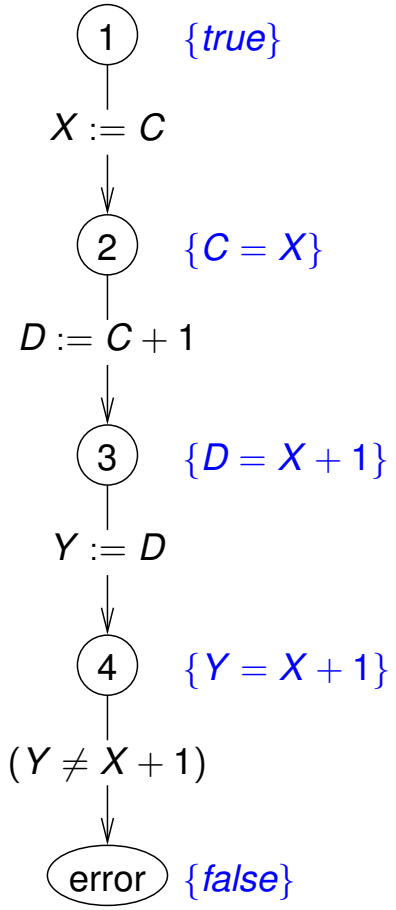
A Spurious Trace is an Unsatisfiable Formula.

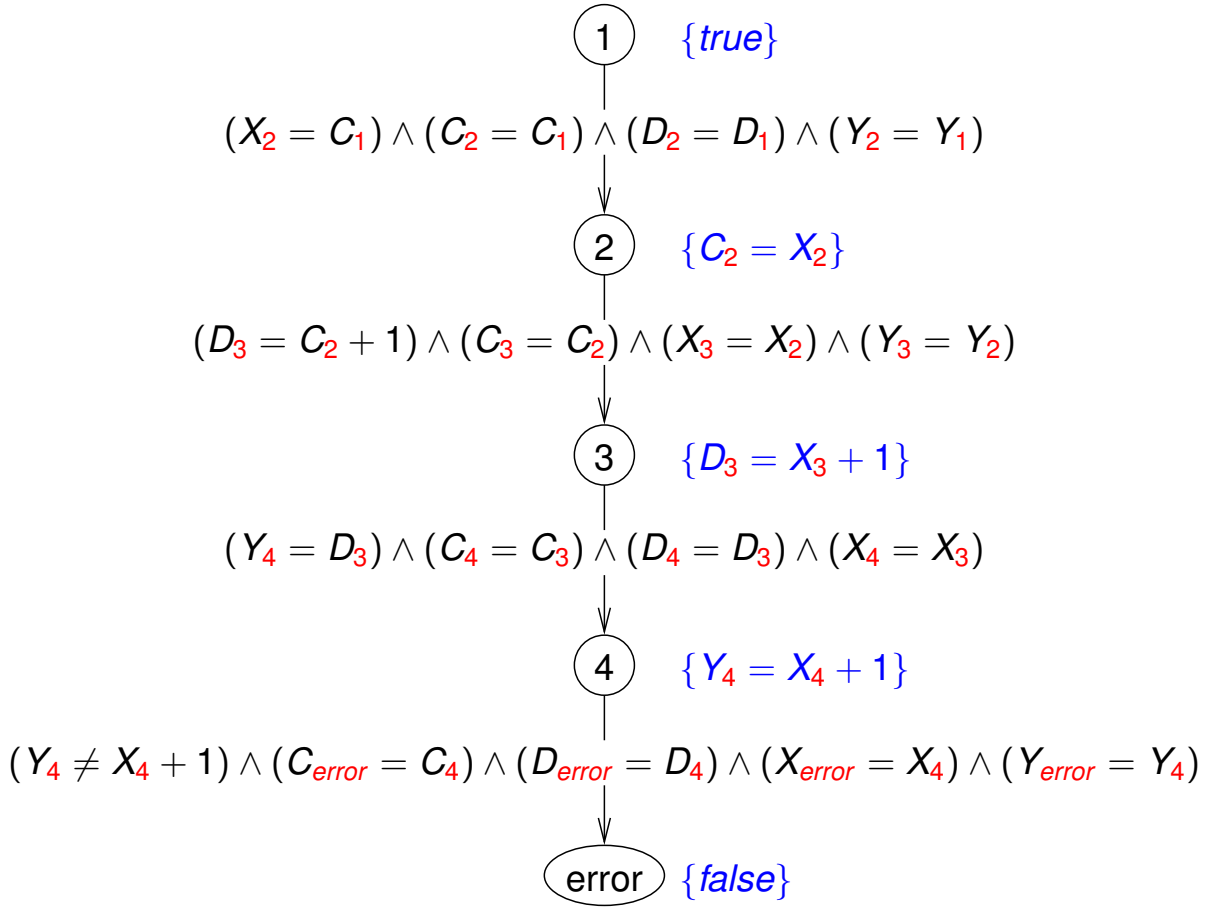


What is a Hoare-Proof of Spuriousness?



What is a Hoare-Proof of Spuriousness?





What is a Hoare Proof of Spuriousness?

Observations

- A blue predicate $\{\cdots\}$ is implied by the conjunction of the **instructions above**.

What is a Hoare Proof of Spuriousness?

Observations

- A blue predicate $\{\cdots\}$ is implied by the conjunction of the **instructions above**.
- A blue predicate is unsatisfiable together with the conjunction of the **instructions below**.

What is a Hoare Proof of Spuriousness?

Observations

- A blue predicate $\{\cdots\}$ is implied by the conjunction of the **instructions above**.
- A blue predicate is unsatisfiable together with the conjunction of the **instructions below**.
- A blue predicate, together with the **next instruction**, implies the next blue predicate.

The last property is called Tracking Property.

Craig Interpolation in Propositional Logics

Definition (Craig interpolant)

Let (F, G) be a pair of formulas with $F \wedge G$ unsatisfiable.
An interpolant for (F, G) is a formula I
with the following properties:

- $F \models I$,
- $I \wedge G$ is unsatisfiable and
- I refers only to the common variables of F and G.

Craig Interpolation in Propositional Logics

Definition (Craig interpolant)

Let (F, G) be a pair of formulas with $F \wedge G$ unsatisfiable.
An interpolant for (F, G) is a formula I
with the following properties:

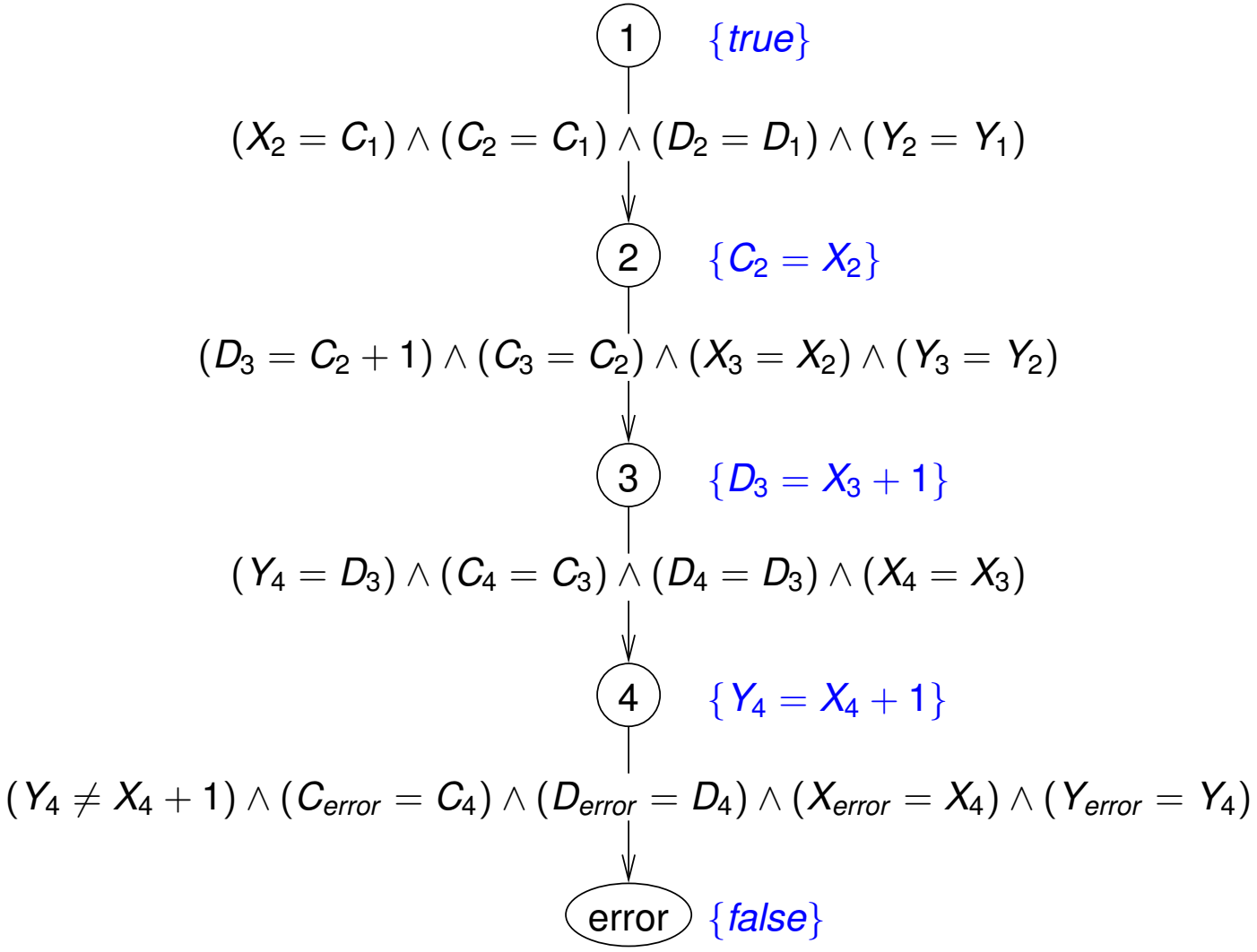
- $F \models I$,
- $I \wedge G$ is unsatisfiable and
- I refers only to the common variables of F and G.

Example

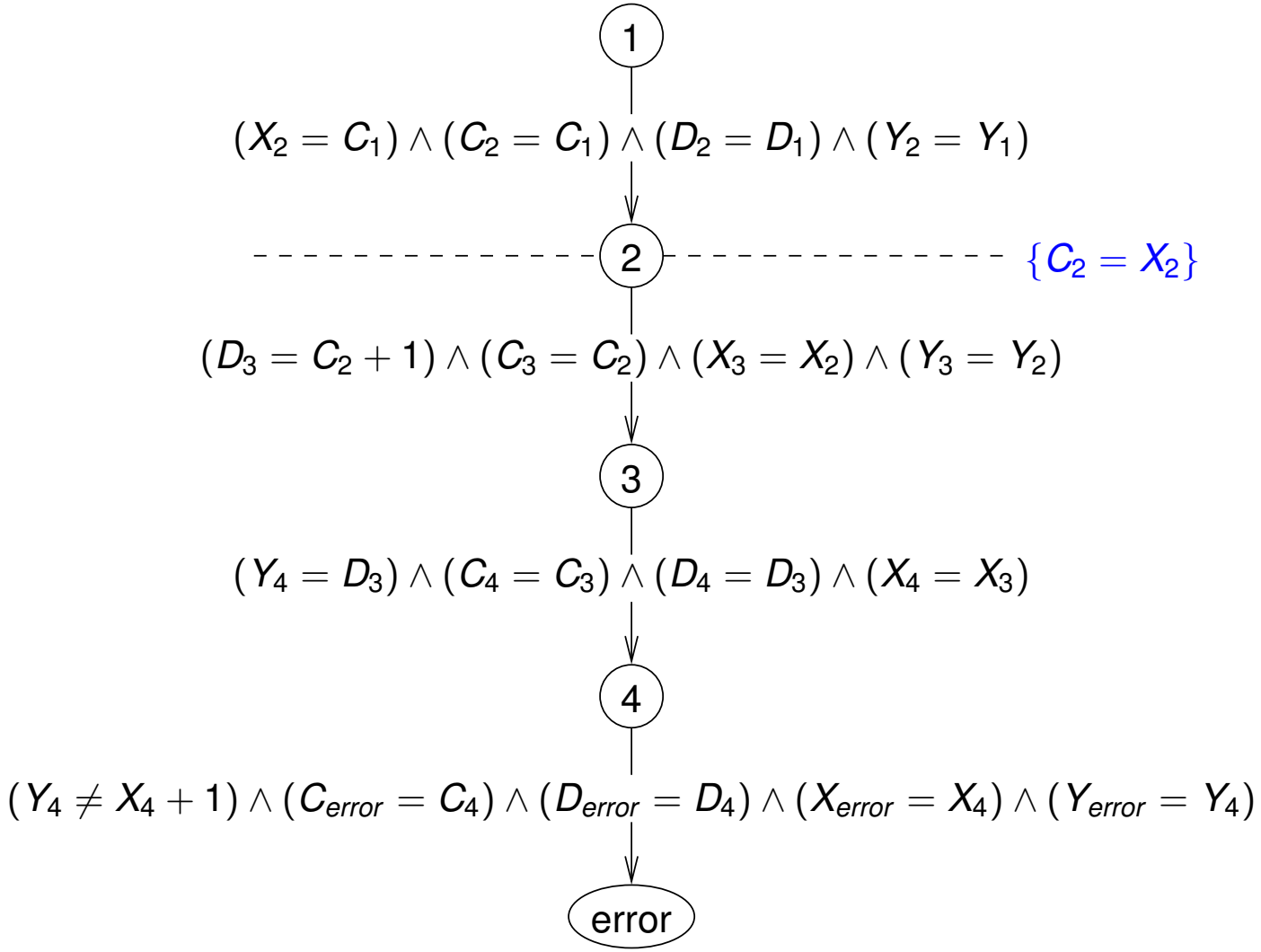
$$F = x \wedge y \qquad G = \neg x \wedge z$$

$I = x$ is an interpolant for (F, G).

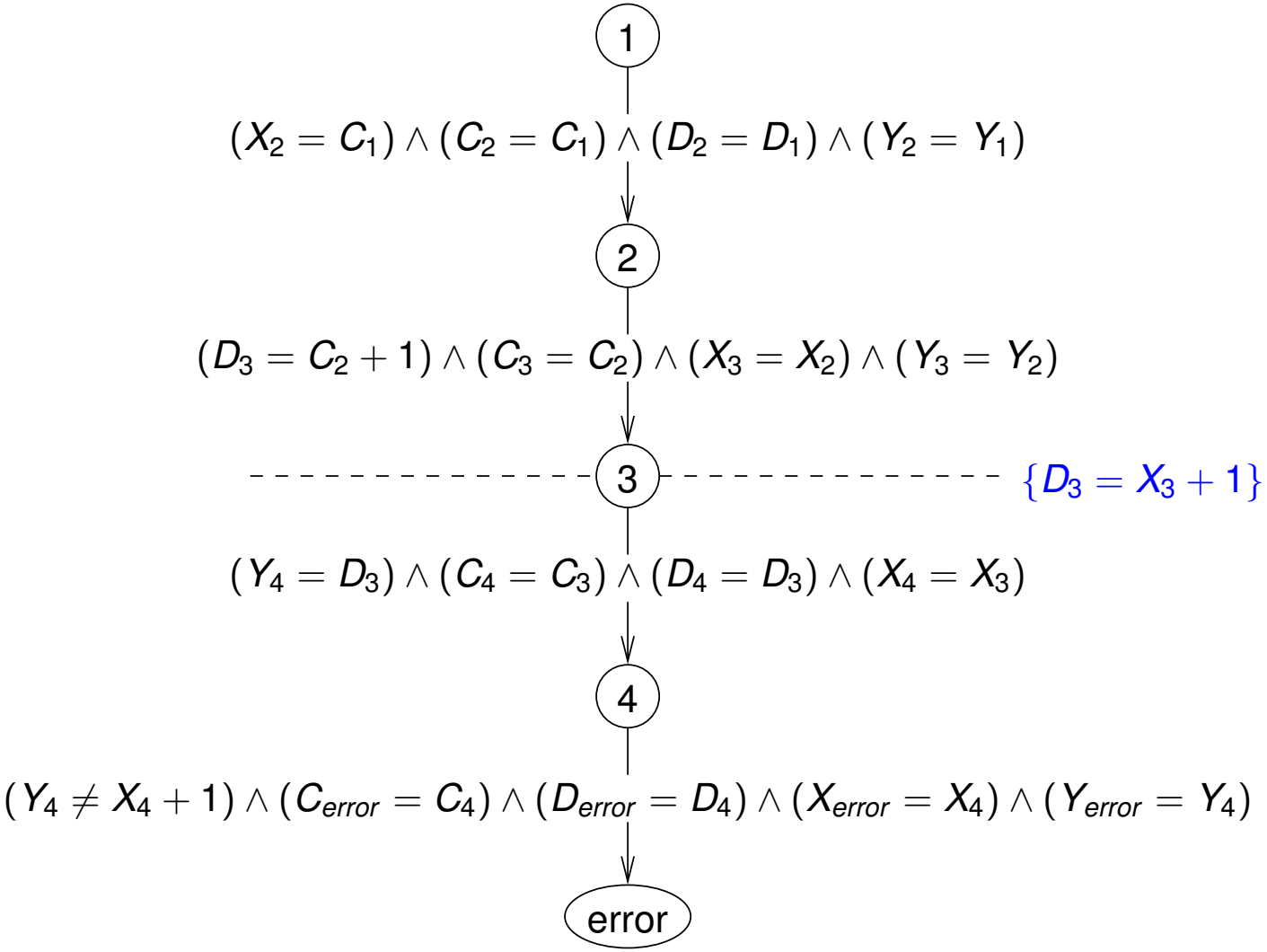
Craig Interpolation: Our Application



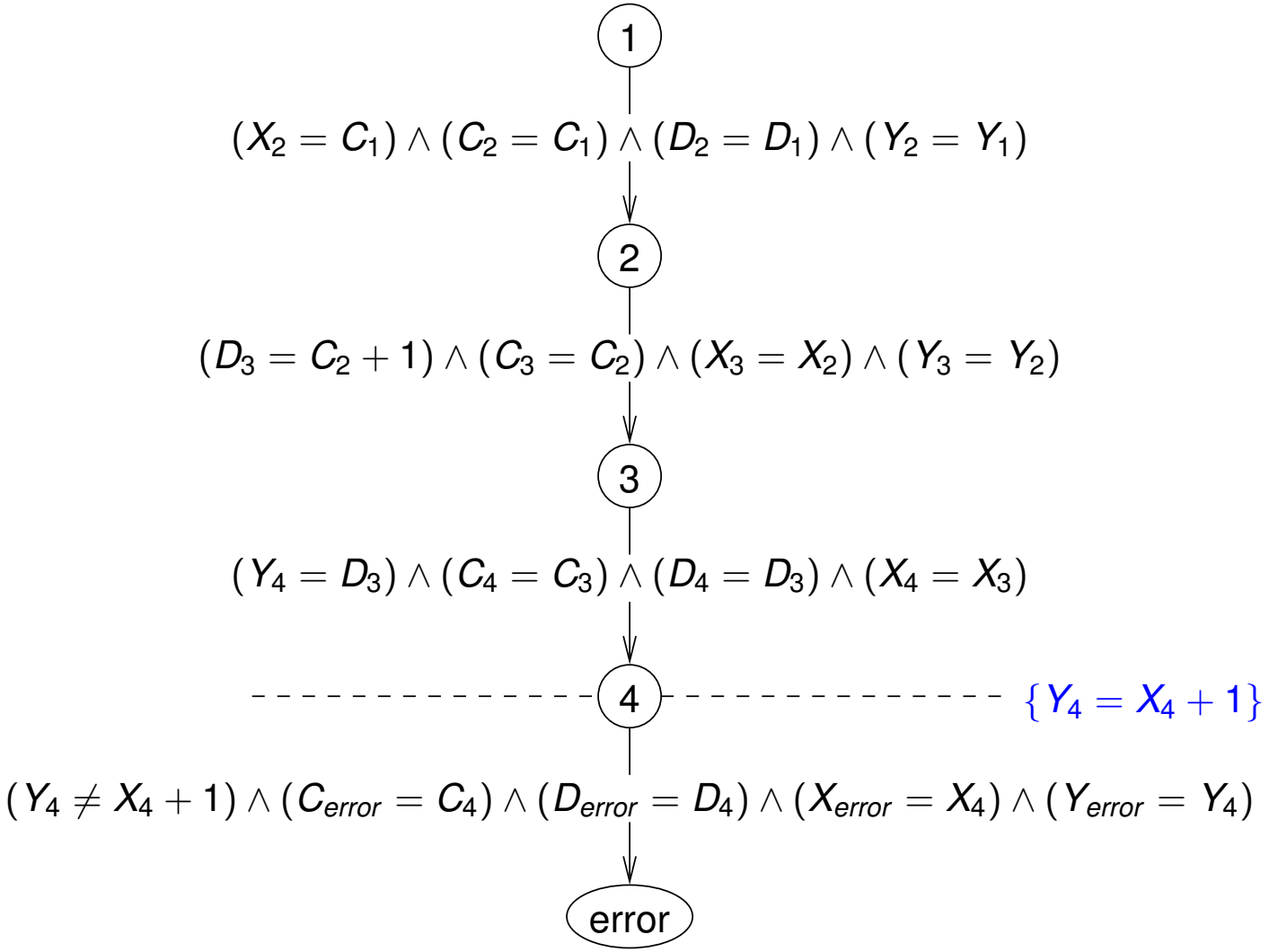
Craig Interpolation: Our Application



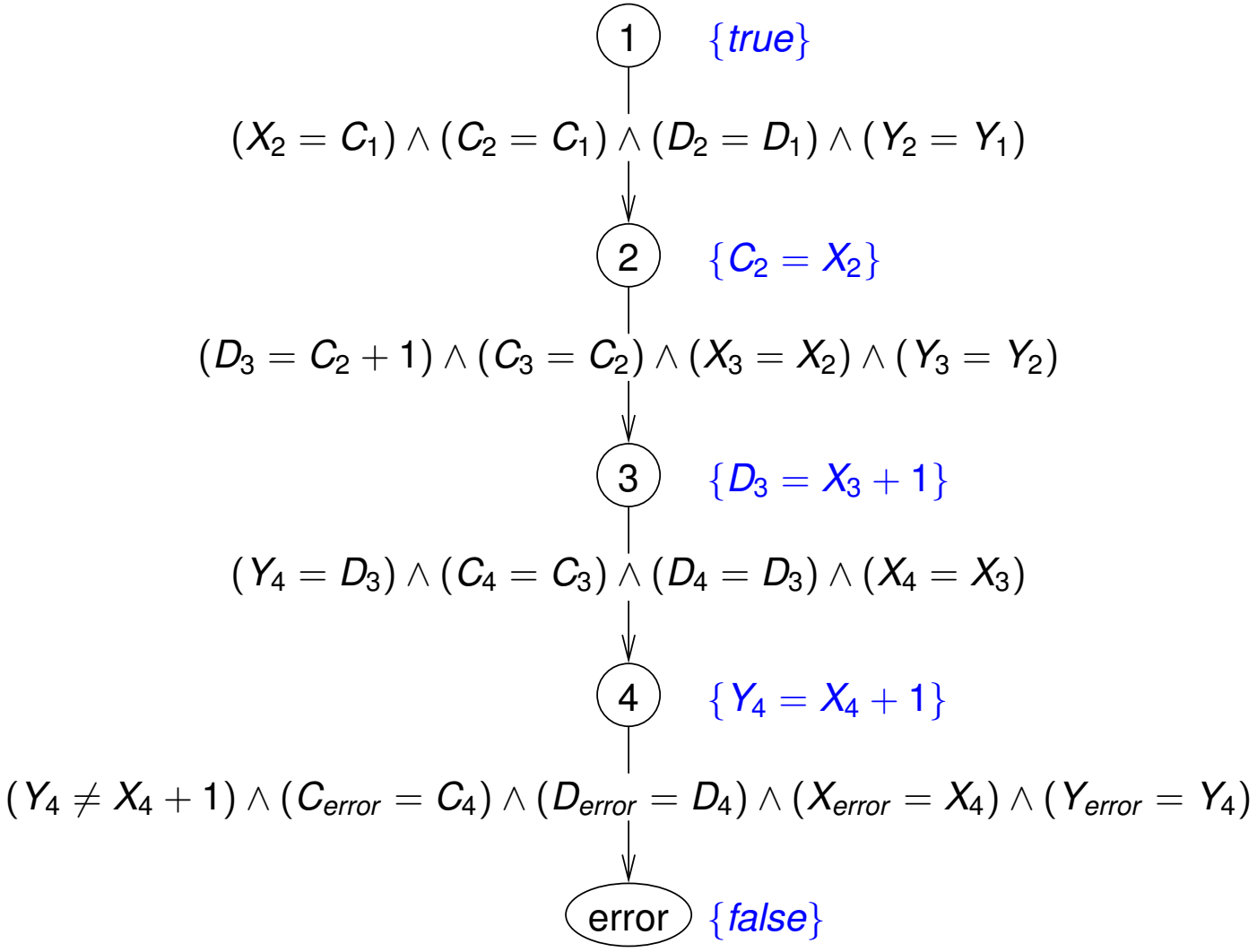
Craig Interpolation: Our Application



Craig Interpolation: Our Application



Craig Interpolation: Our Application



Summary

- Spurious traces $\leftrightarrow$ unsatisfiable formula
- Craig interpolants satisfying the tracking property $\rightarrow$ Hoare proofs of spuriousness
- 'Clean' Hoare proofs of spuriousness $\rightarrow$ Craig interpolants

Weakest and Strongest Interpolants

Definition (weakest interpolant)

The weakest interpolant for (F, G) is the interpolant for (F, G) that is implied by all interpolants for (F, G).
It is denoted by $WI(F, G)$.

Definition (strongest interpolant)

The strongest interpolant for (F, G) is the interpolant for (F, G) that implies all interpolants for (F, G).
It is denoted by $SI(F, G)$.

We show how to compute them and that they satisfy the tracking property.

A Charaterization of Weakest Interpolants

Theorem (weakest interpolant)

- *Let* (F, G) *be a pair of formulas with* $F \wedge G$ *unsatisfiable.*
- *Let* Z *be the variables that occur in* G*, but not in* F*.*

Then $WI(F, G) \equiv \forall Z.\neg G$*.*

A Charaterization of Weakest Interpolants

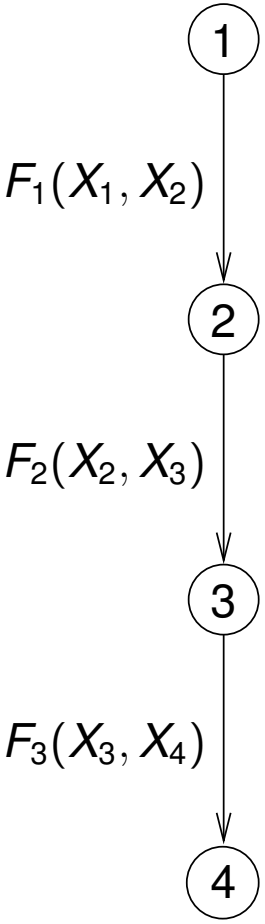
Theorem (weakest interpolant)

- *Let* (F, G) *be a pair of formulas with* $F \wedge G$ *unsatisfiable.*
- *Let* Z *be the variables that occur in* G*, but not in* F*.*

Then $WI(F, G) \equiv \forall Z.\neg G$*.*

Very adequate for computation with BDDs.

Efficient Computation of Weakest Interpolants



Theorem

- *Let $F_1 \wedge F_2 \wedge F_3$ be unsatisfiable.*
- *Let X_3 be the variables that occur in F_2, but not in F_1.*

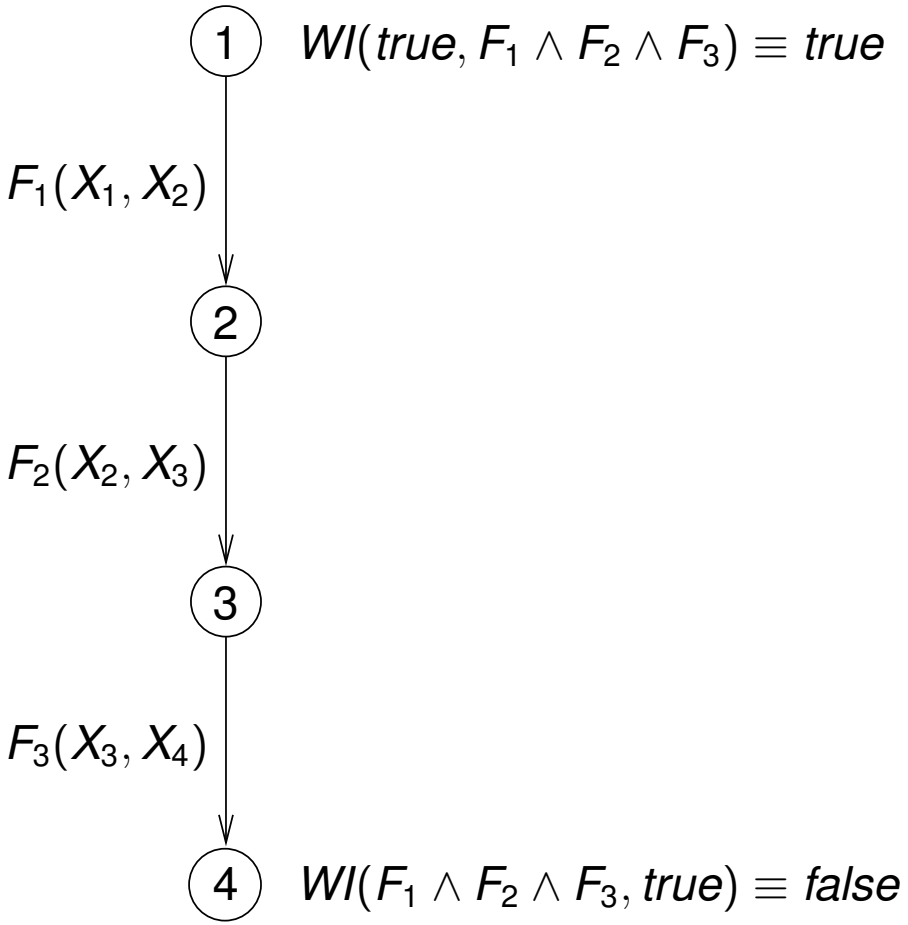
Then

$$WI(F_1, F_2 \wedge F_3) \equiv \forall X_3 (F_2 \rightarrow WI(F_1 \wedge F_2, F_3)).$$

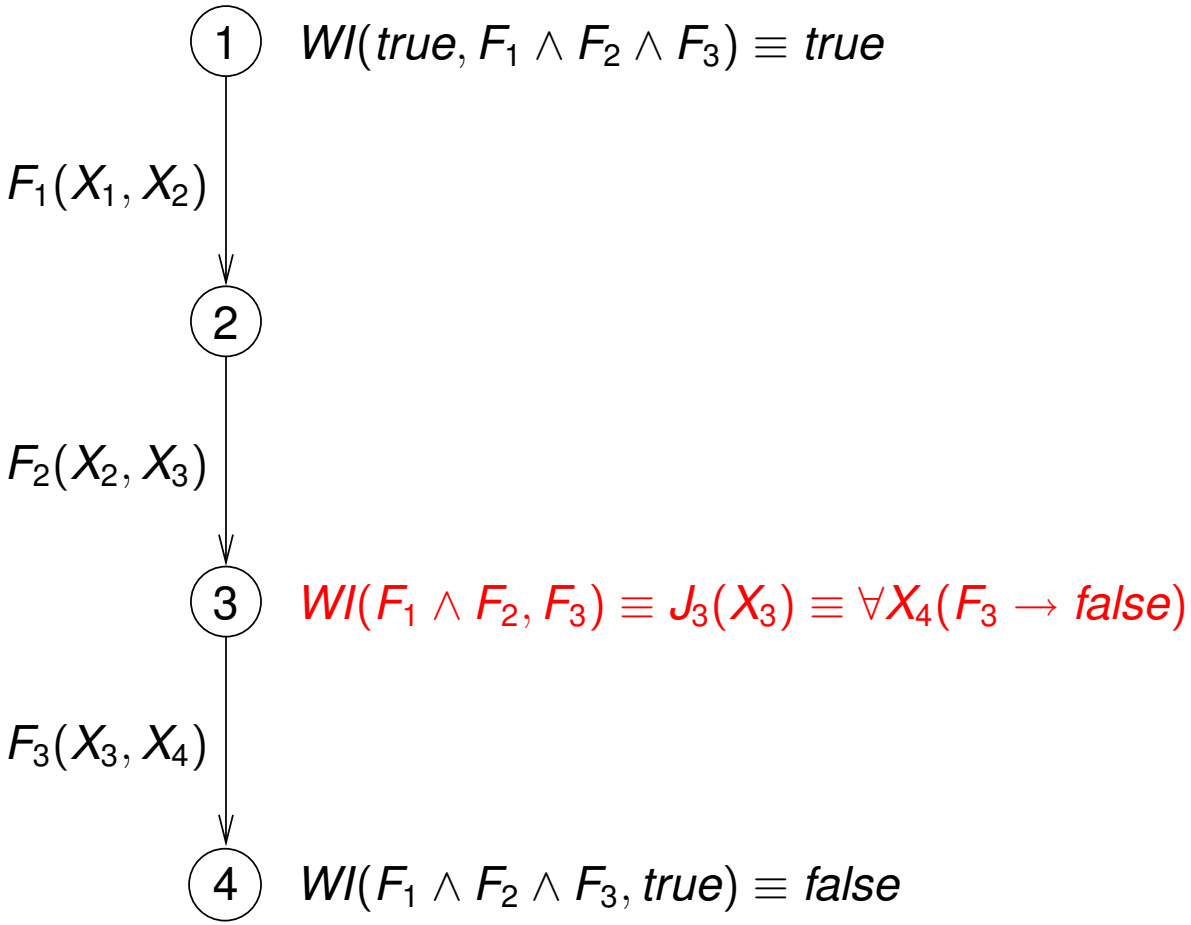
Corollary (Tracking Property)

$$WI(F_1, F_2 \wedge F_3) \wedge F_2 \quad \models \quad WI(F_1 \wedge F_2, F_3).$$

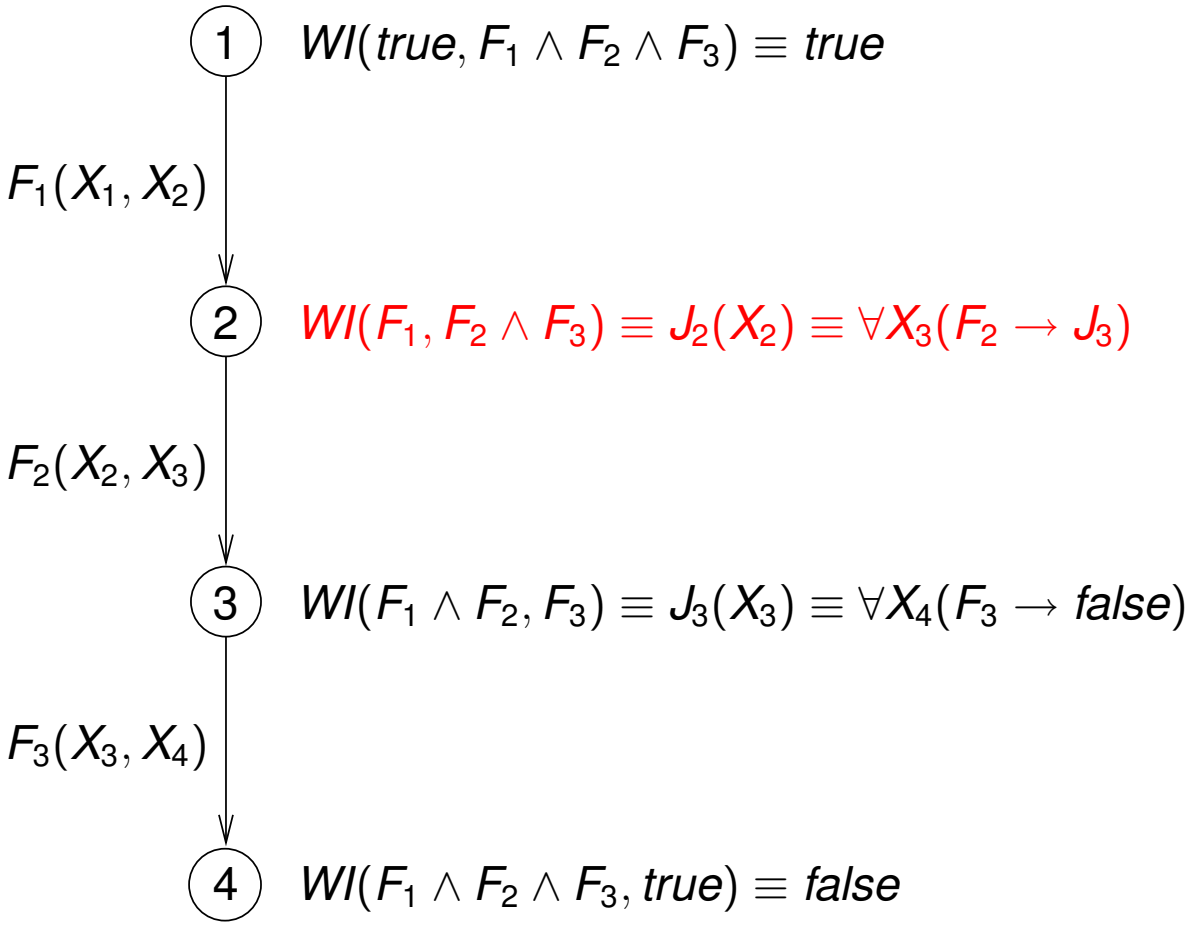
Interpolants Computation for a Spurious Trace



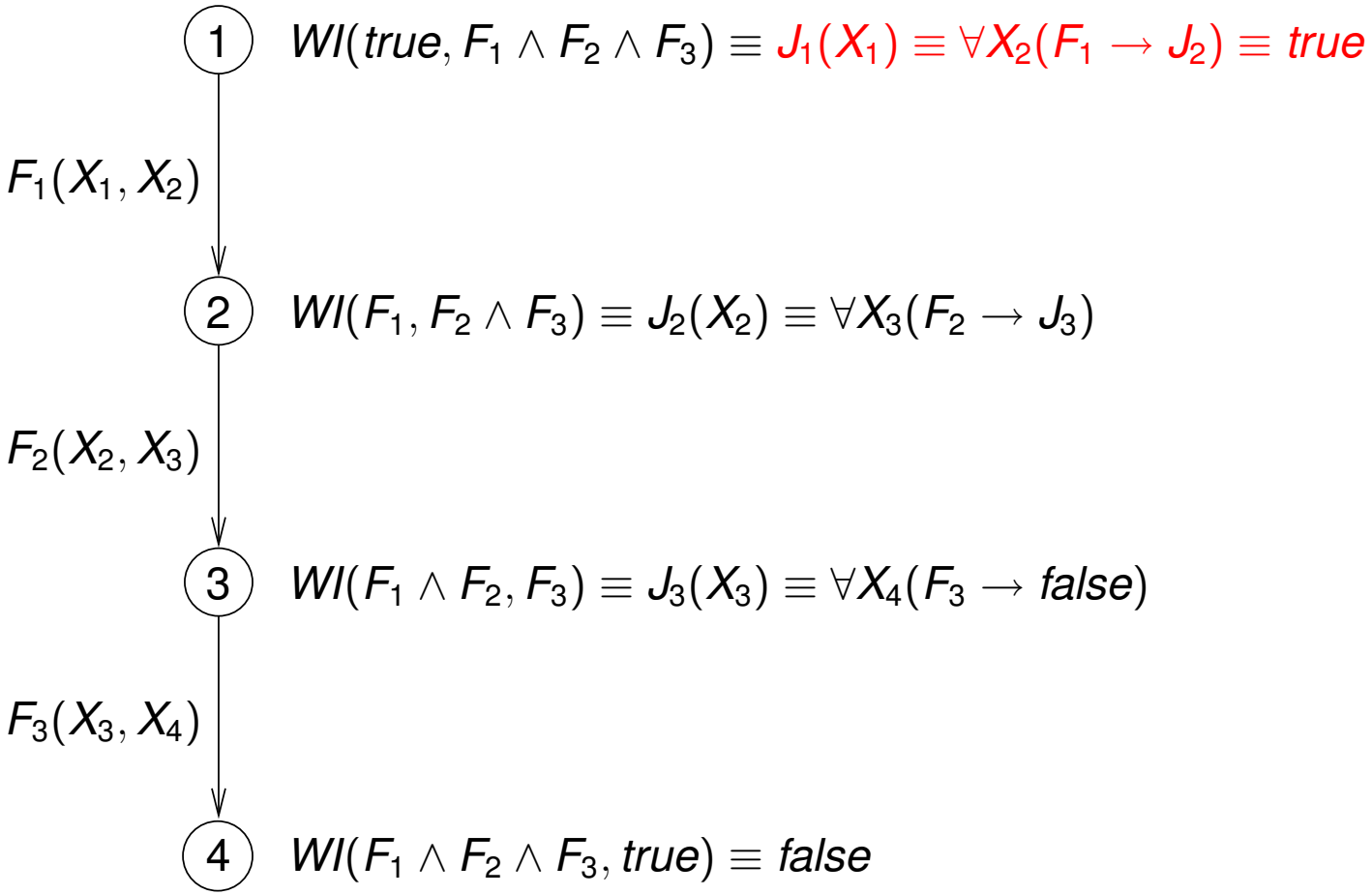
Interpolants Computation for a Spurious Trace



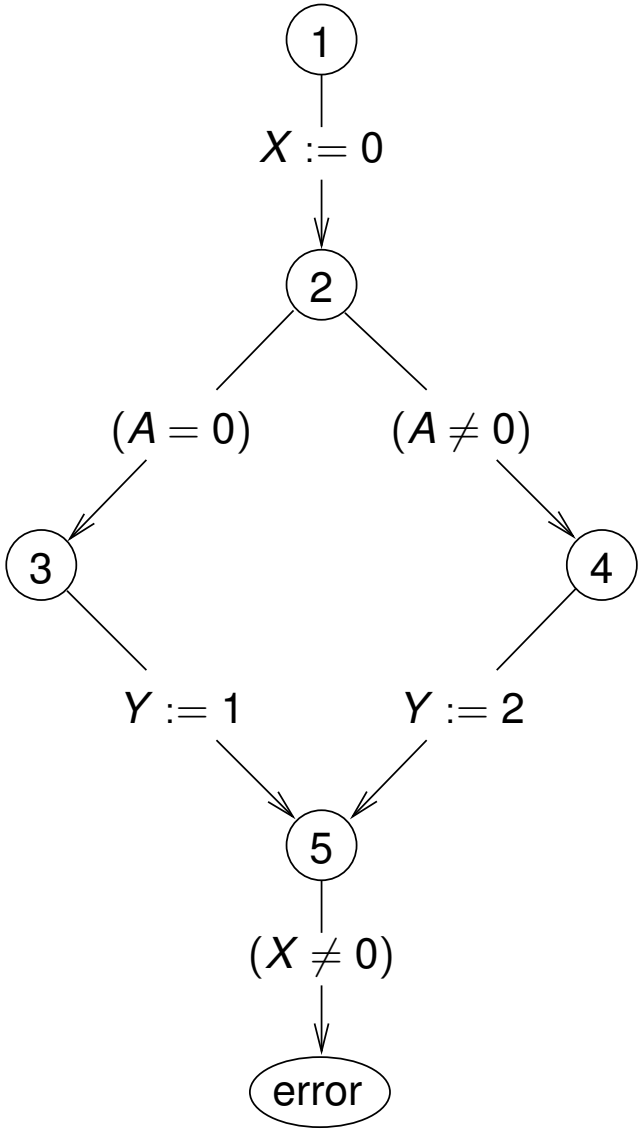
Interpolants Computation for a Spurious Trace



Interpolants Computation for a Spurious Trace



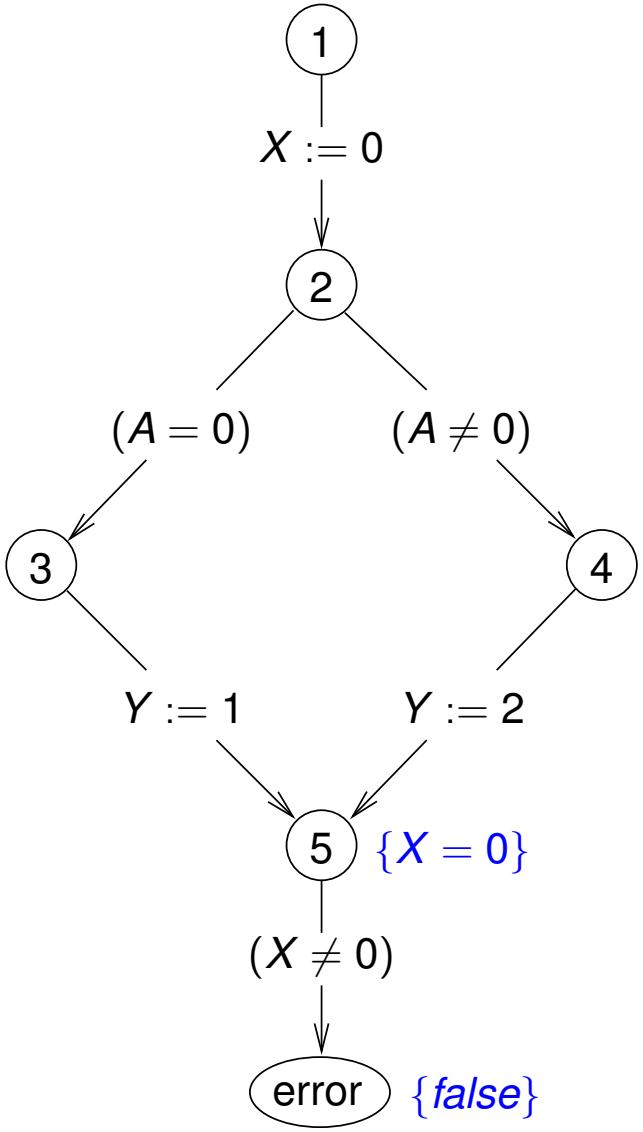
DAGs of Spurious Counterexamples



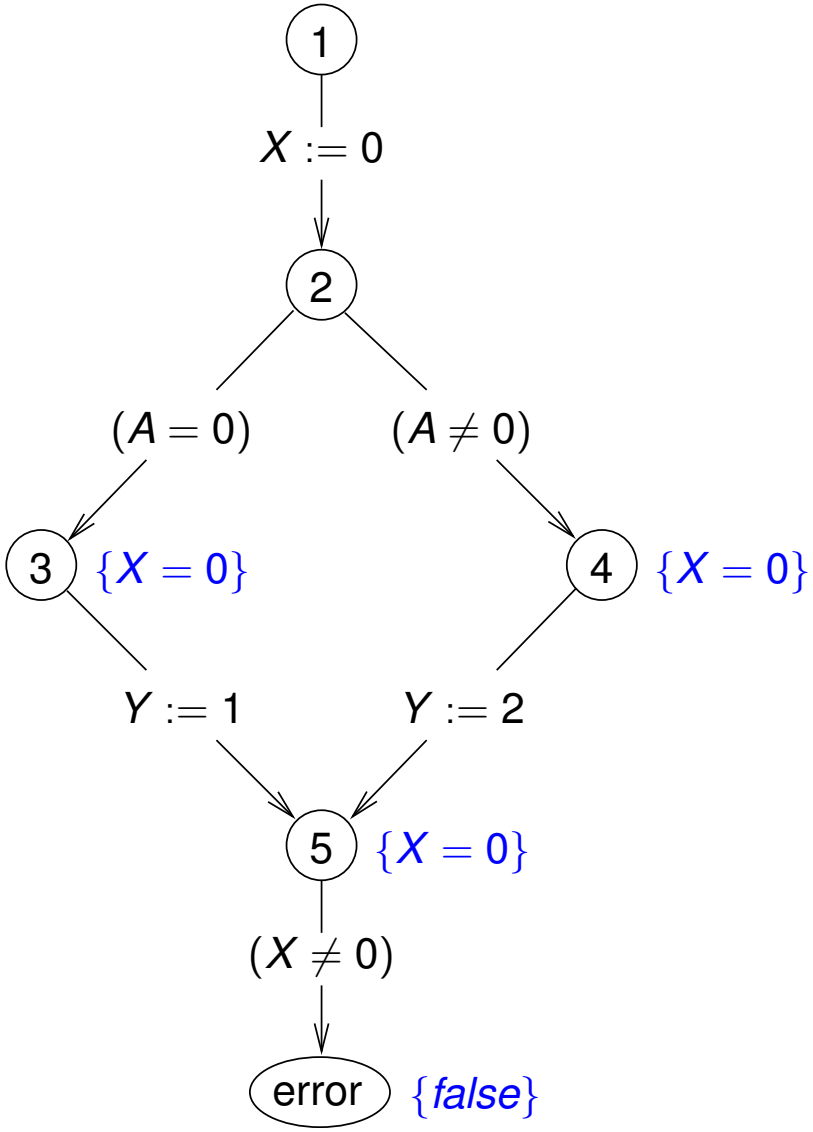
Spurious Counterexample DAGs

- Each path through the DAG is a spurious counterexample.
- Each path through the DAG corresponds to an unsatisfiable formula.
- The disjunction of the trace formulas is unsatisfiable.

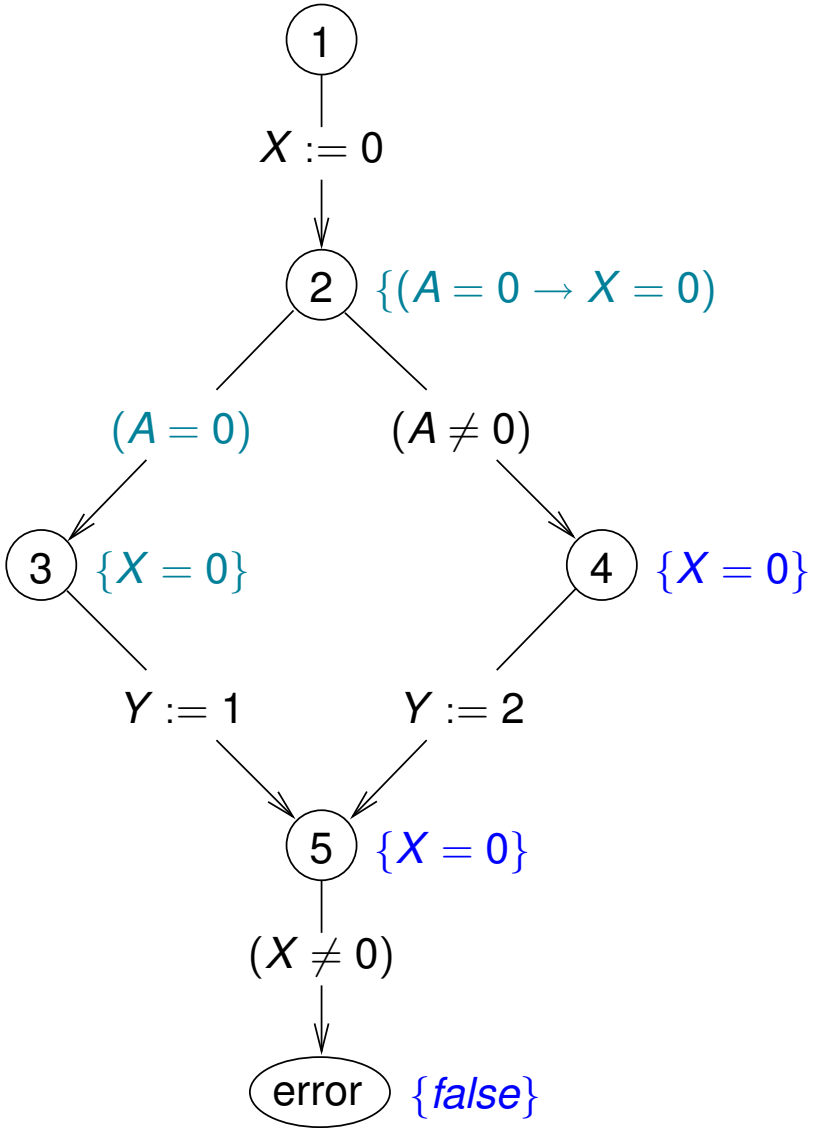
DAGs of Spurious Counterexamples



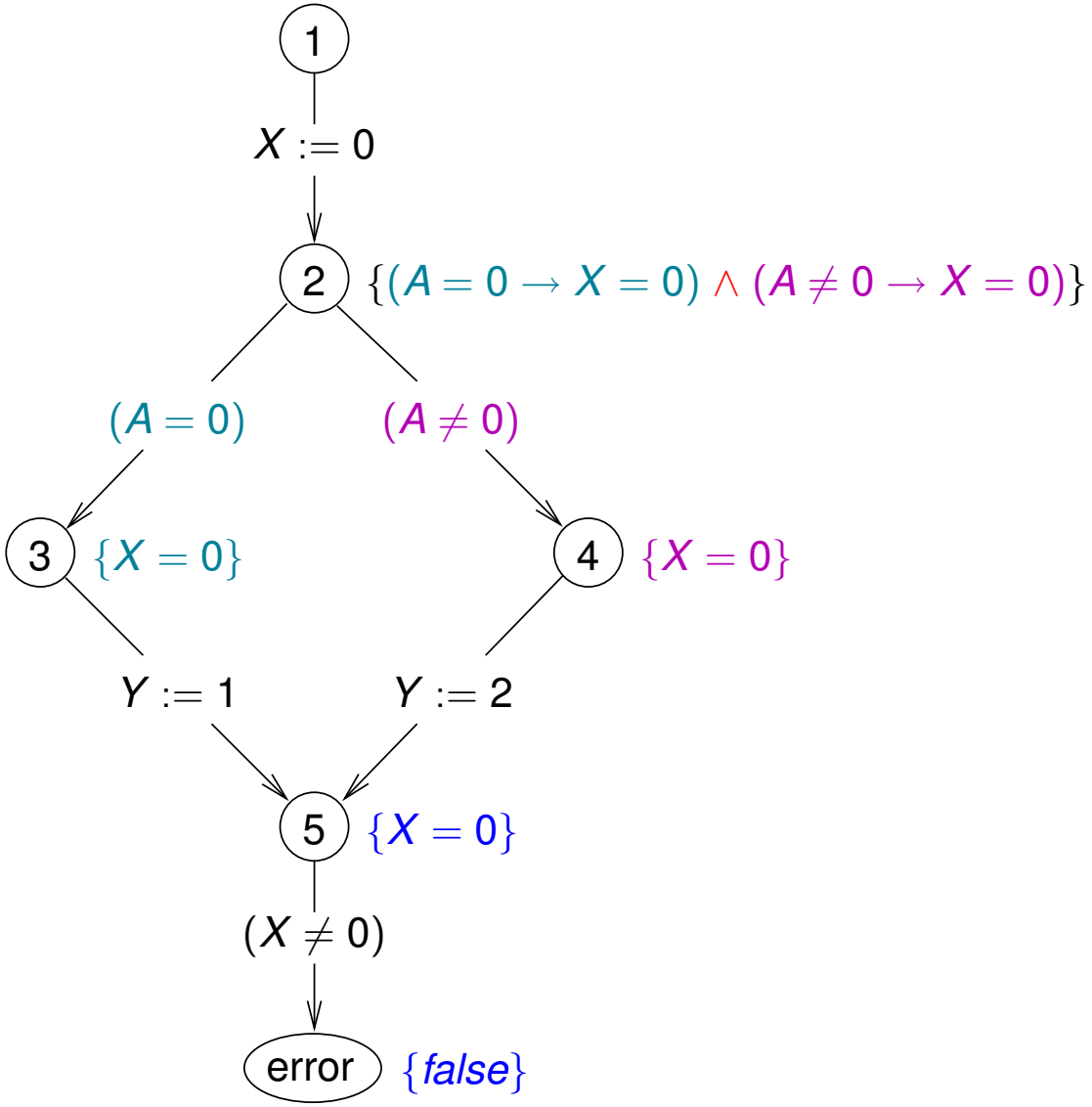
DAGs of Spurious Counterexamples



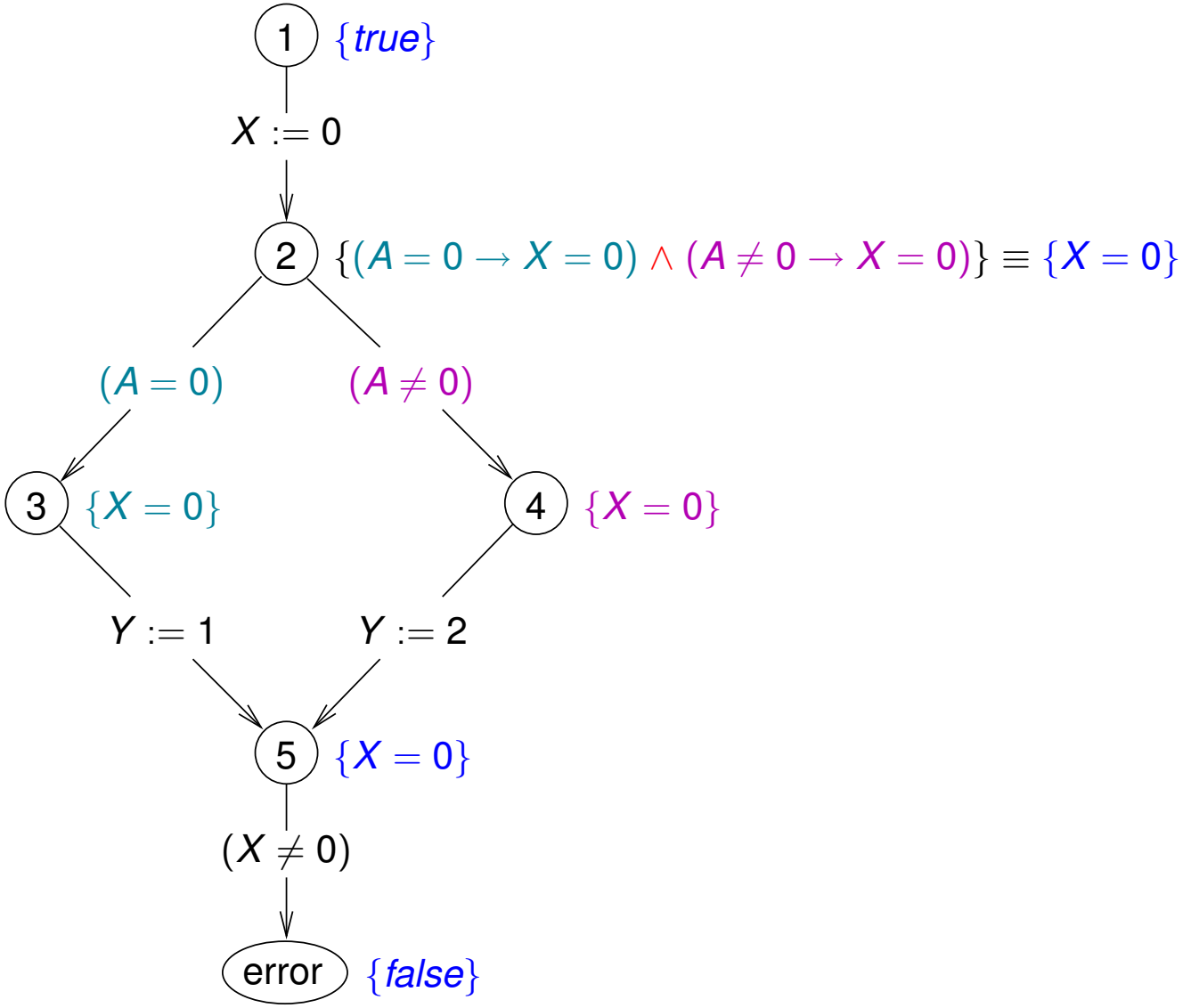
DAGs of Spurious Counterexamples



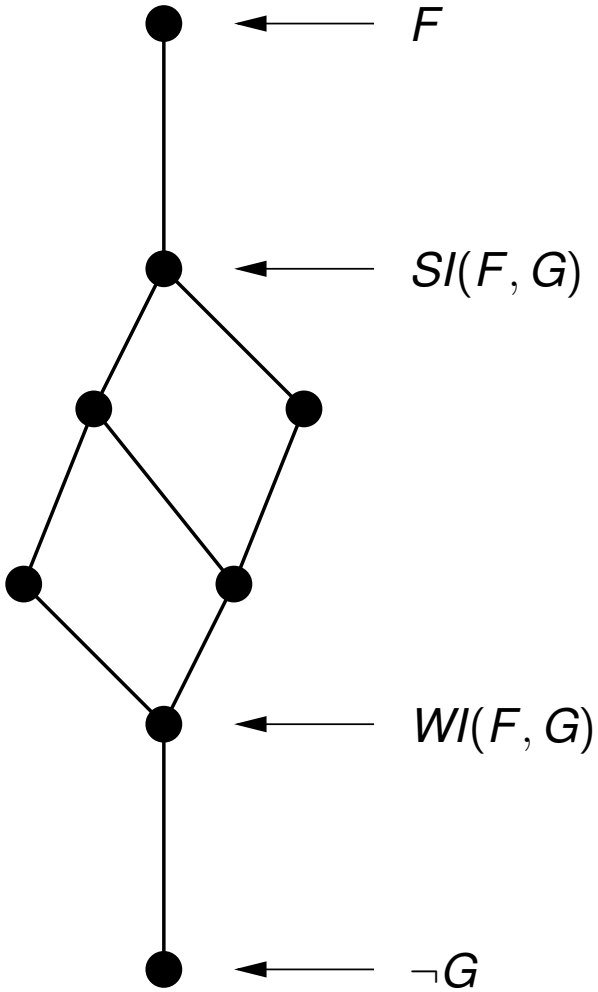
DAGs of Spurious Counterexamples



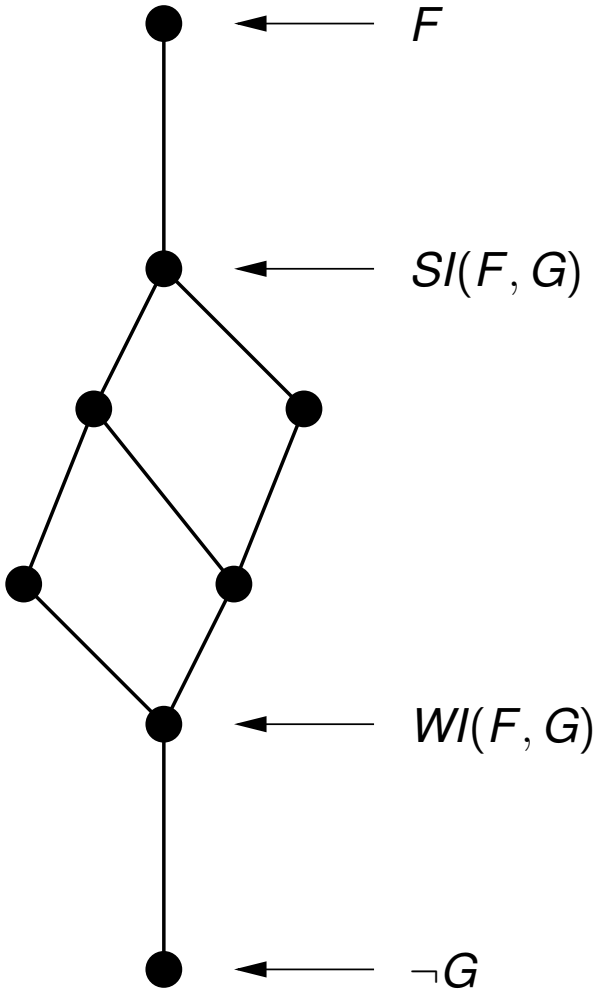
DAGs of Spurious Counterexamples



There Are Many Interpolants.



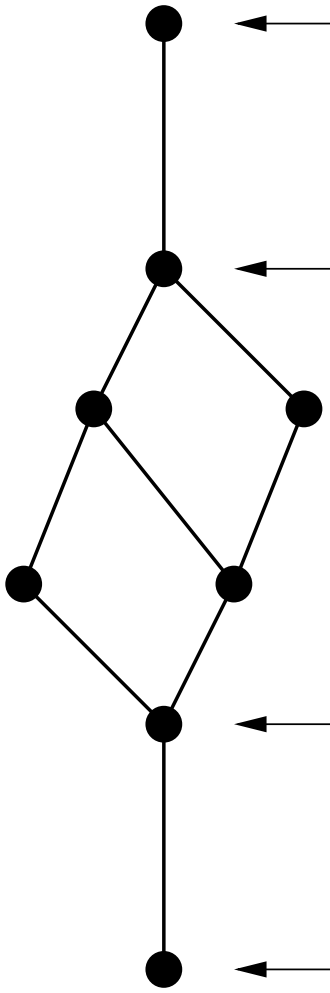
There Are Many Interpolants.



$\equiv x \wedge (y \vee z)$

$\equiv x \vee y \vee (z \wedge w)$

There Are Many Interpolants.



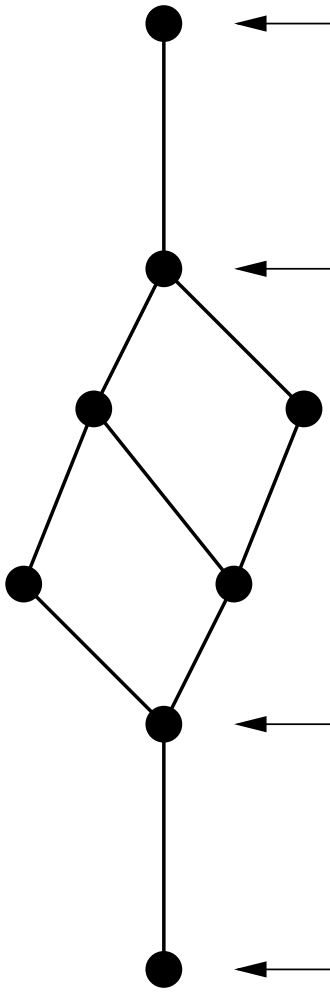
$F \equiv x \wedge (y \vee z)$

$SI(F, G) \equiv \exists\{\}.F \equiv x \wedge (y \vee z)$

$WI(F, G) \equiv \forall\{w\}.\neg G \equiv x \vee y$

$\neg G \equiv x \vee y \vee (z \wedge w)$

There Are Many Interpolants.



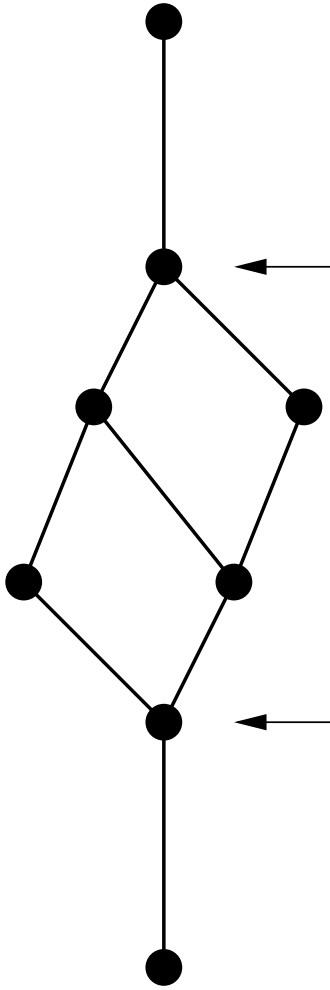
$F \equiv x \wedge (y \vee z)$

$SI(F, G) \equiv \exists\{\}.F \equiv x \wedge (y \vee z)$

$WI(F, G) \equiv \forall\{w\}.\neg G \equiv x \vee y$

$\neg G \equiv x \vee y \vee (z \wedge w)$

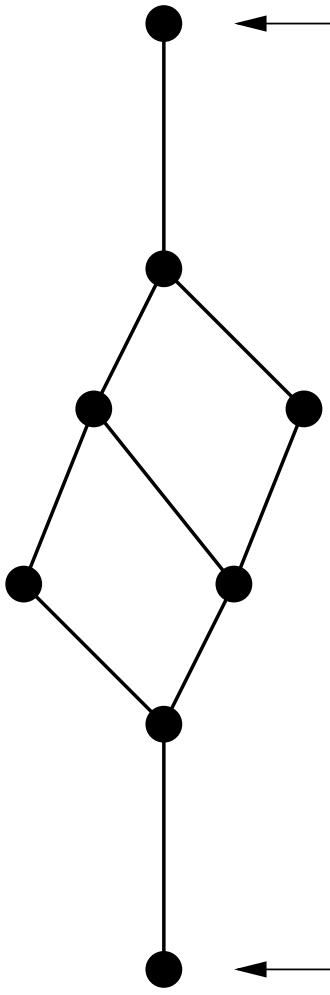
There Are Many Interpolants.



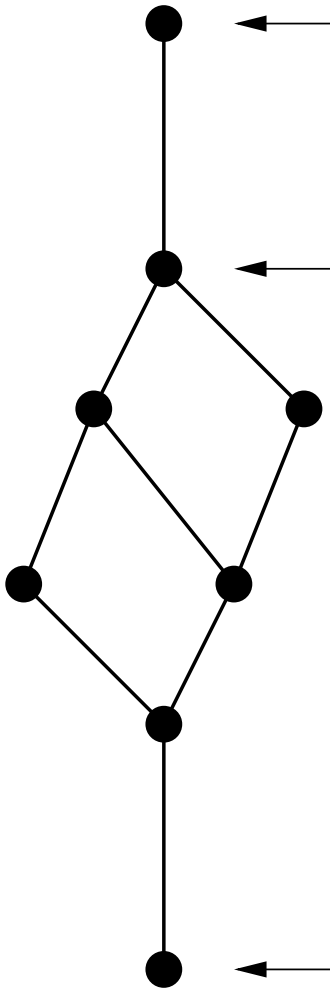
$SI(F, G)$ $\equiv \exists\{\}.F \equiv x \wedge (y \vee z)$

$WI(F, G)$ $\equiv \forall\{w\}.\neg G \equiv x \vee y$

There Are Many Interpolants.



There Are Many Interpolants.

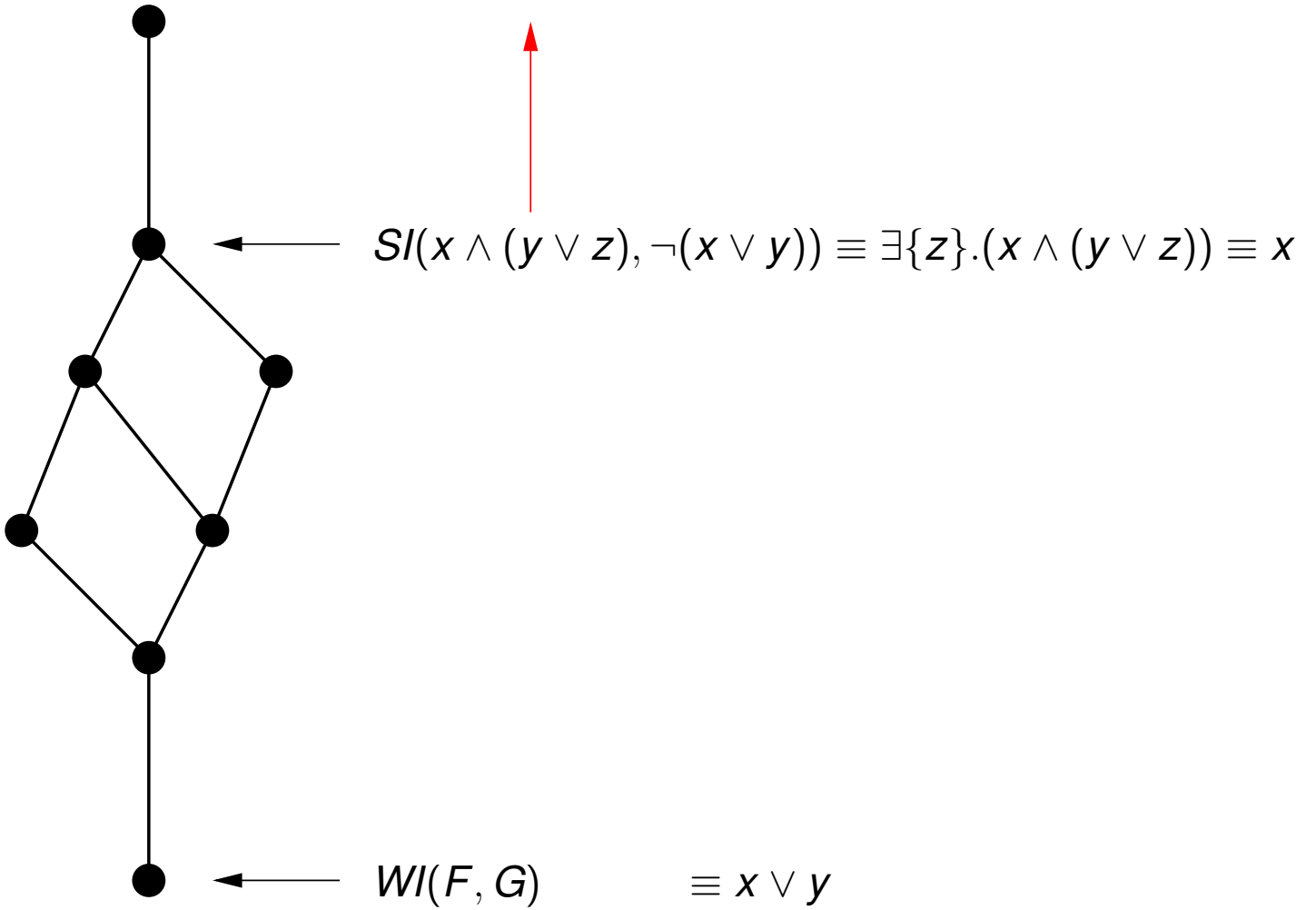


$SI(F, G) \equiv x \wedge (y \vee z)$

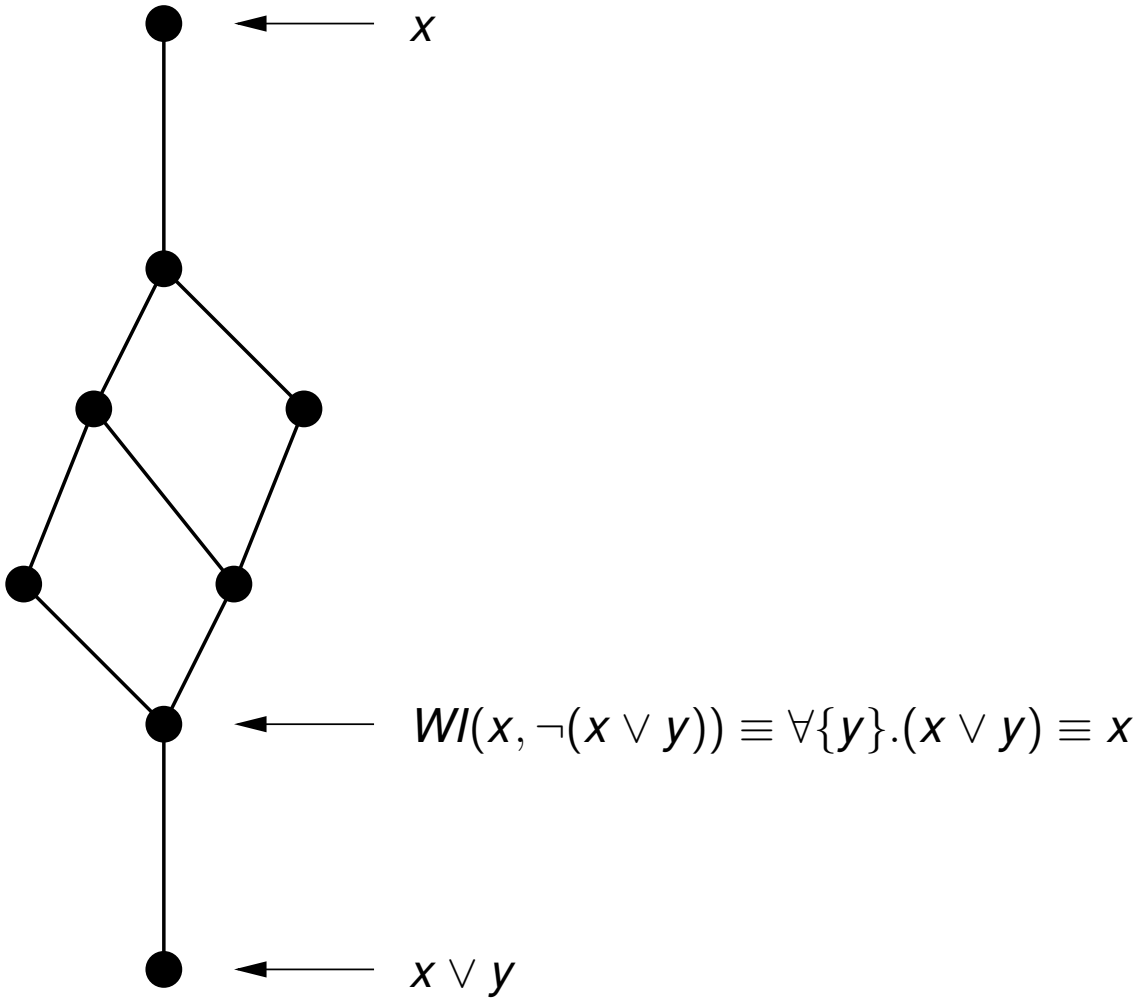
$SI(x \wedge (y \vee z), \neg(x \vee y)) \equiv \exists\{z\}.(x \wedge (y \vee z)) \equiv x$

$WI(F, G) \equiv x \vee y$

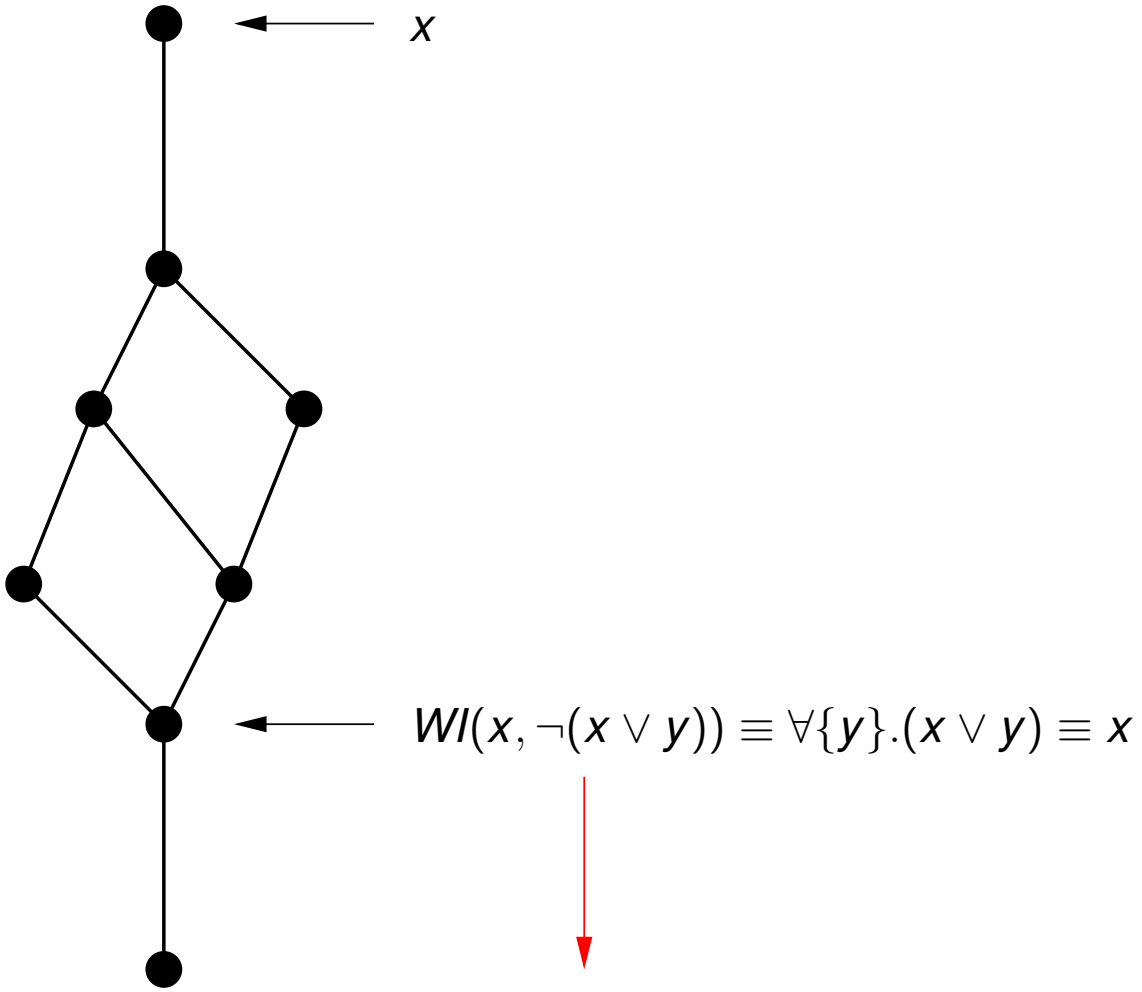
There Are Many Interpolants.



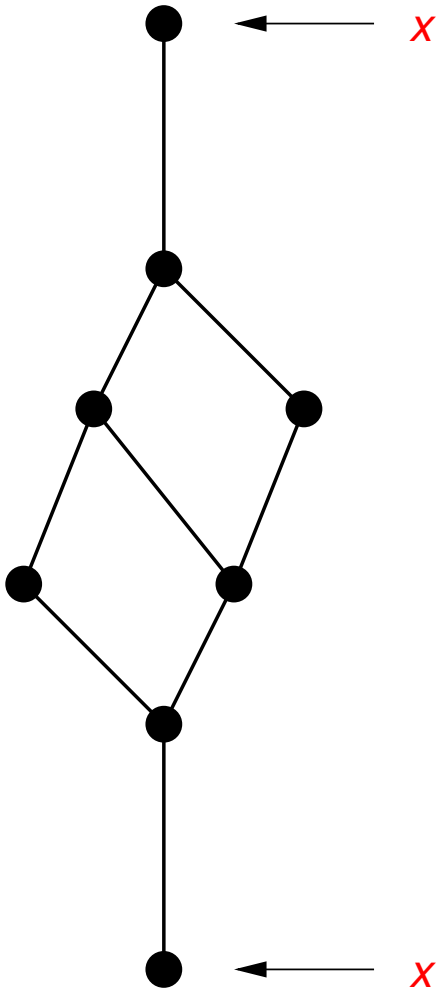
There Are Many Interpolants.



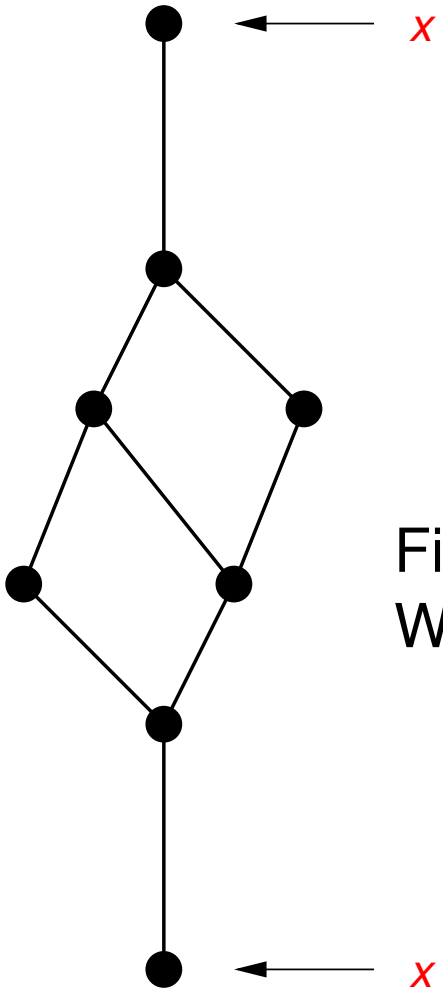
There Are Many Interpolants.



There Are Many Interpolants.



There Are Many Interpolants.



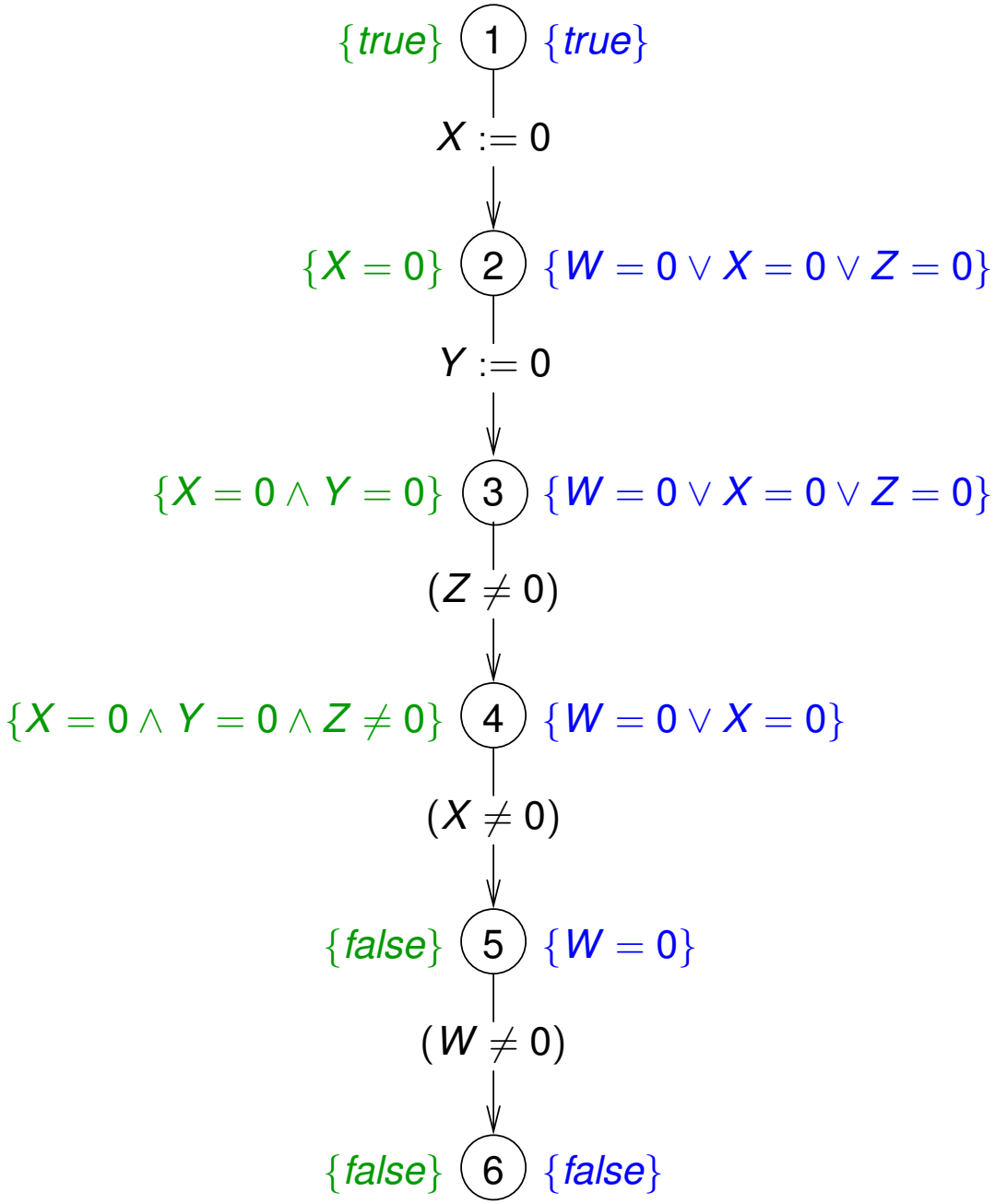
Fixed points have been reached.
We call them conciliated interpolants.

Conciliated Interpolants

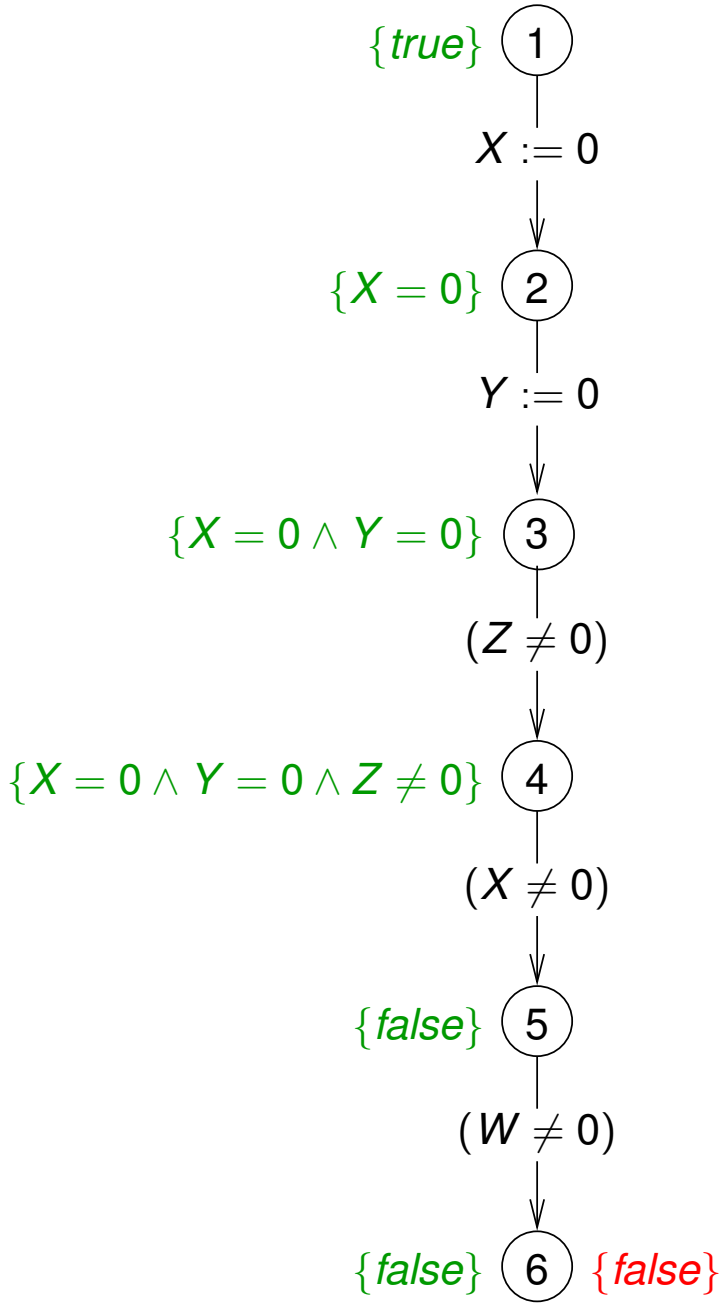
What about the Tracking Property?

- Conciliated interpolants by themselves
  do not necessarily satisfy the tracking property.
- Therefore, we
  1. apply a strongest interpolants computation (forward),
  2. apply a backward computation and
     conciliate after each step with the strongest interpolant.
- The resulting interpolants satisfy the tracking property.

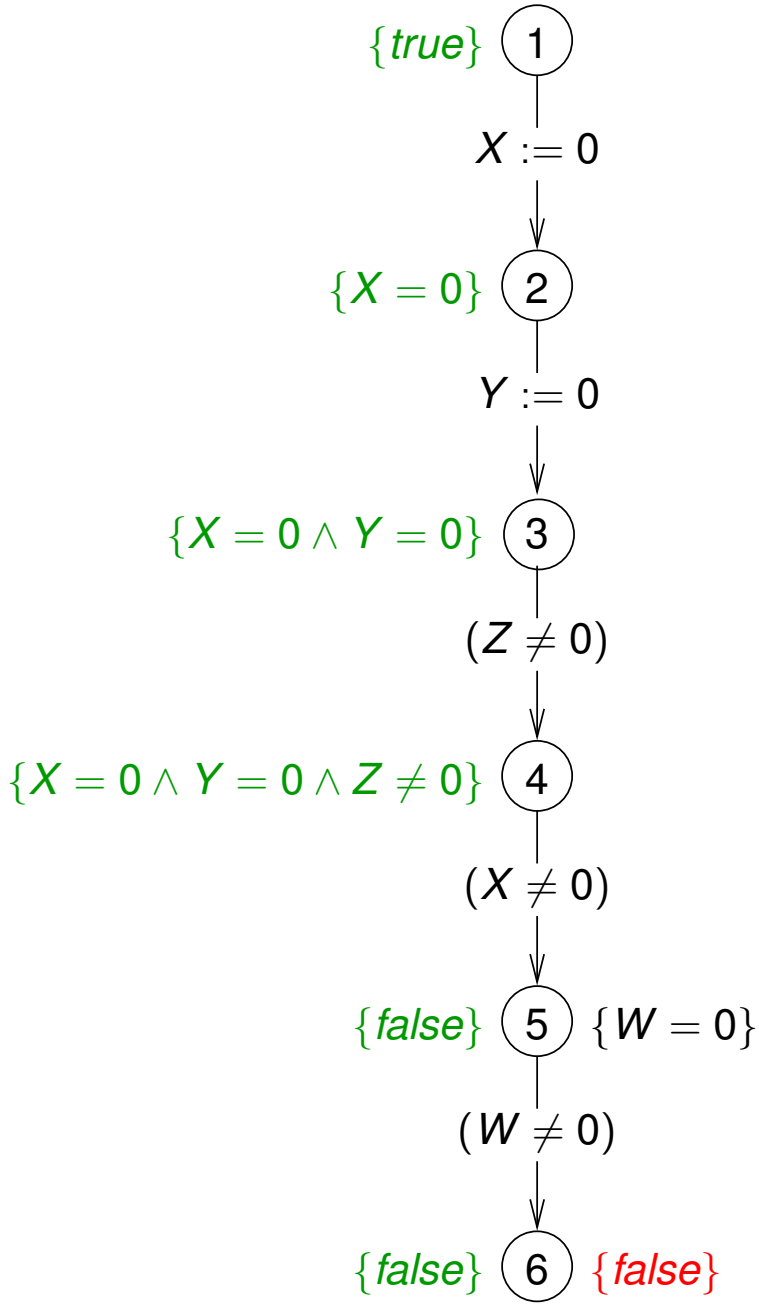
Conciliated Interpolants as a Simplification Procedure



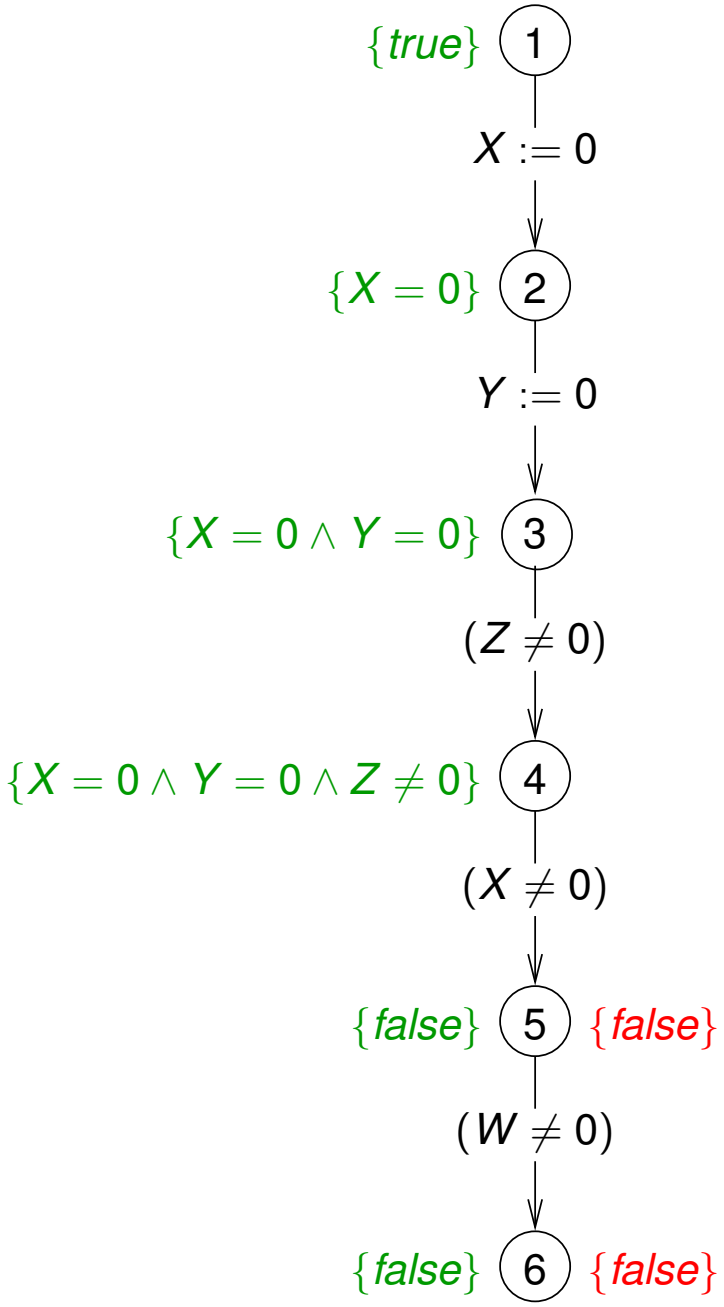
Conciliated Interpolants as a Simplification Procedure



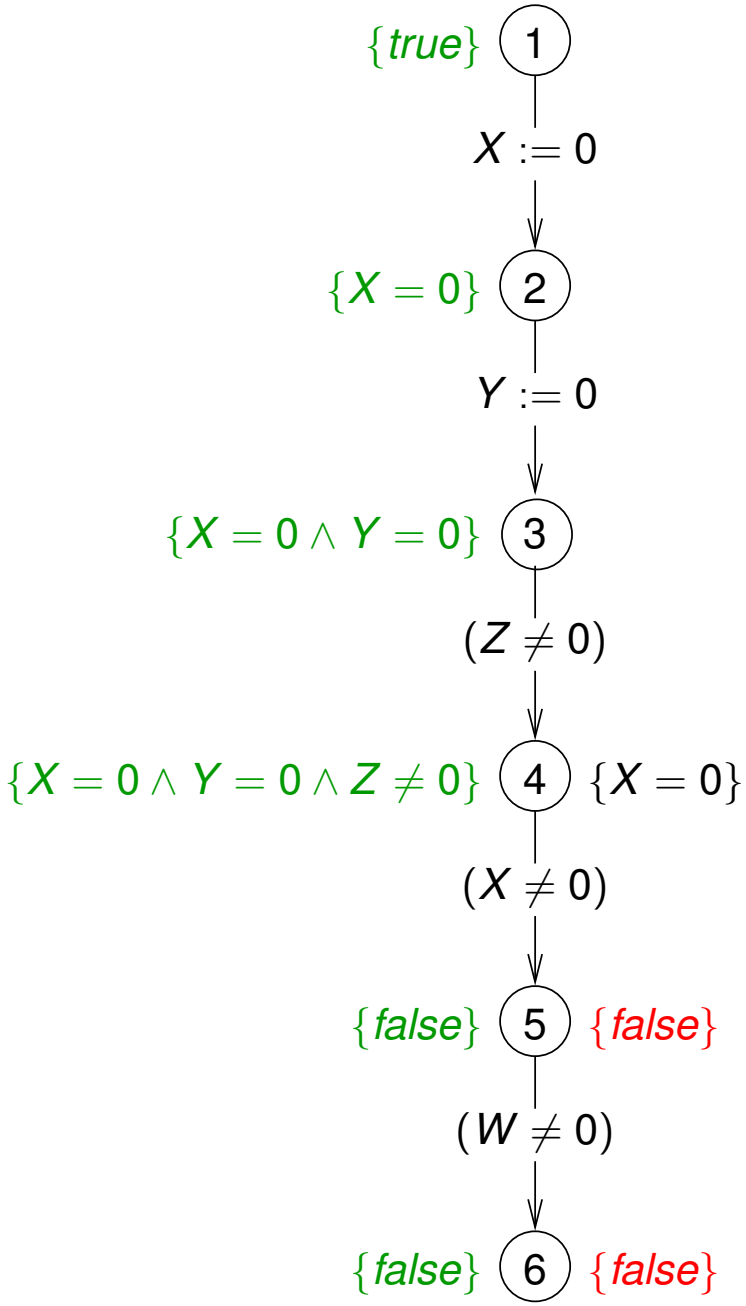
Conciliated Interpolants as a Simplification Procedure



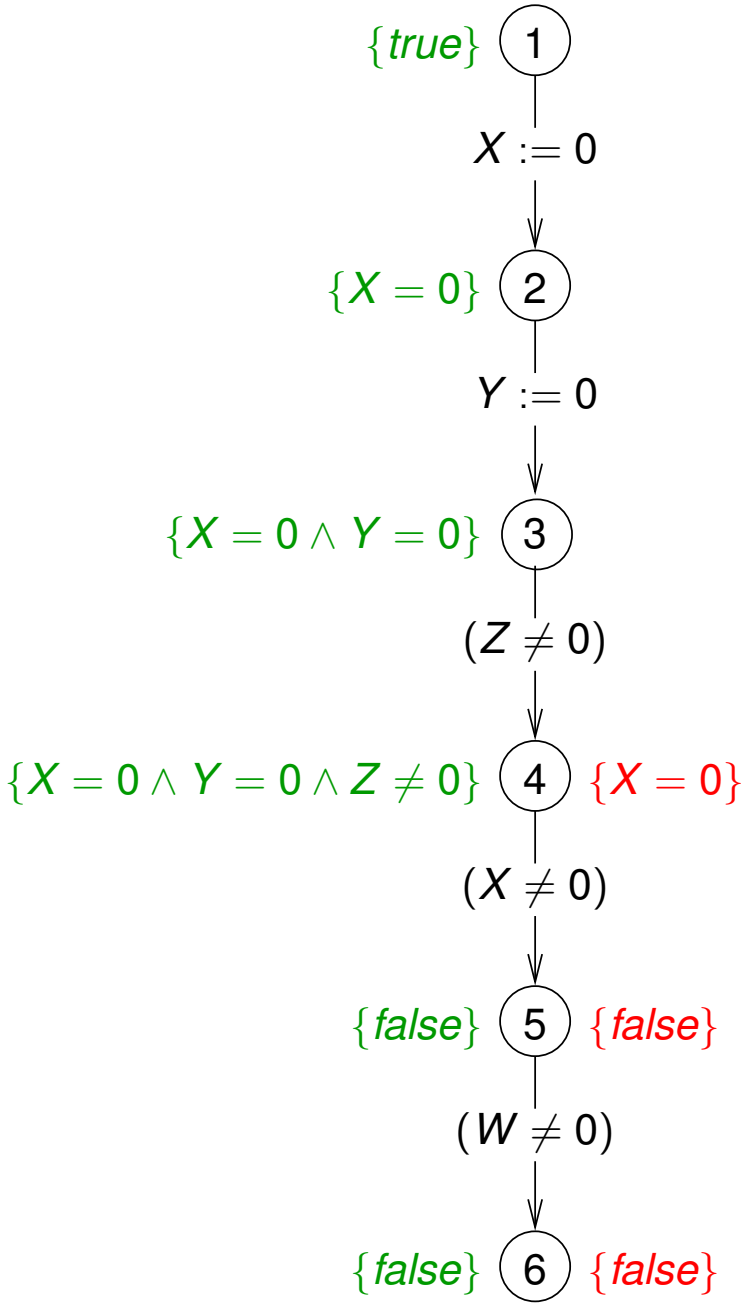
Conciliated Interpolants as a Simplification Procedure



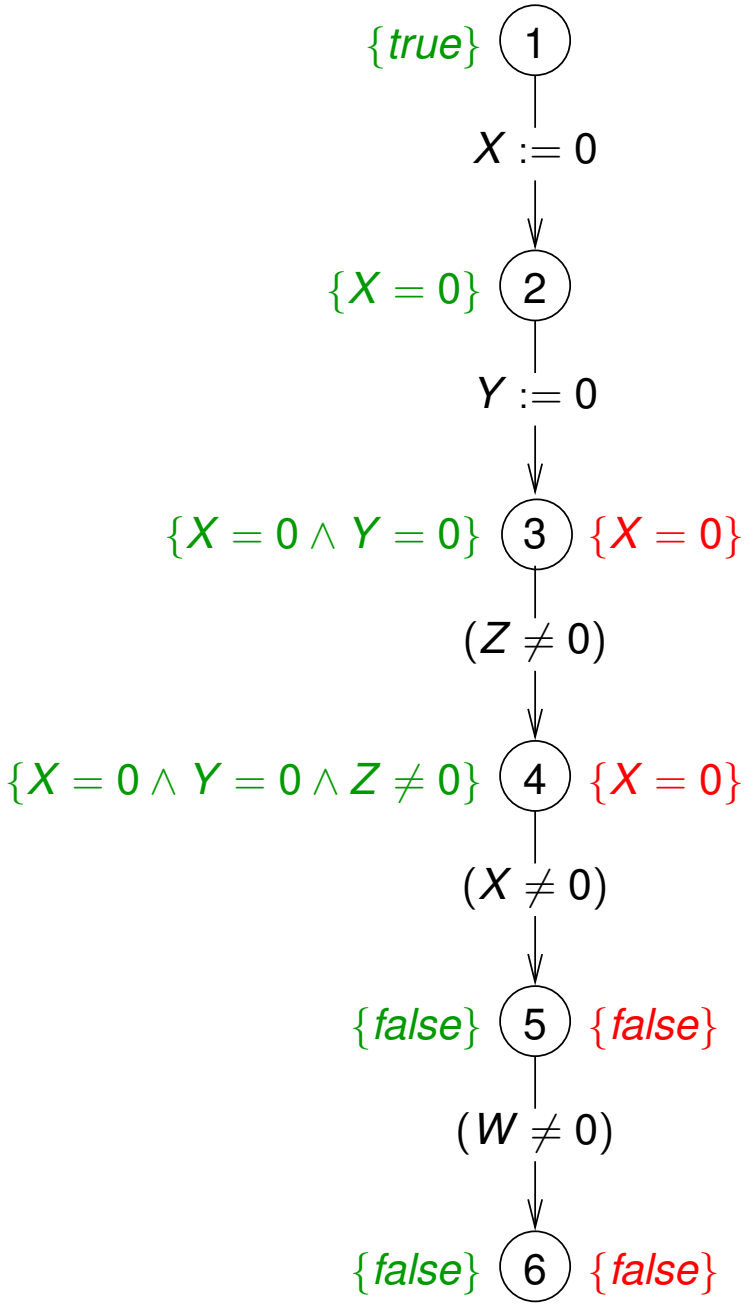
Conciliated Interpolants as a Simplification Procedure



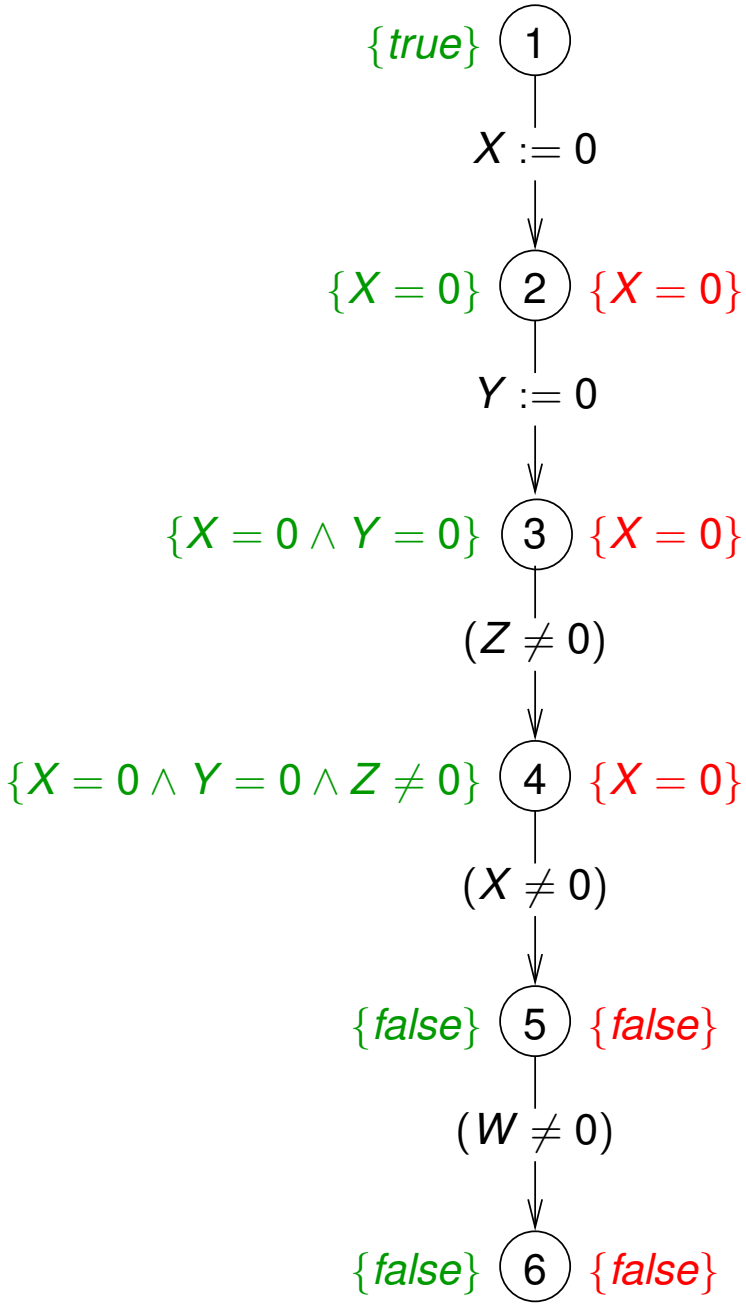
Conciliated Interpolants as a Simplification Procedure



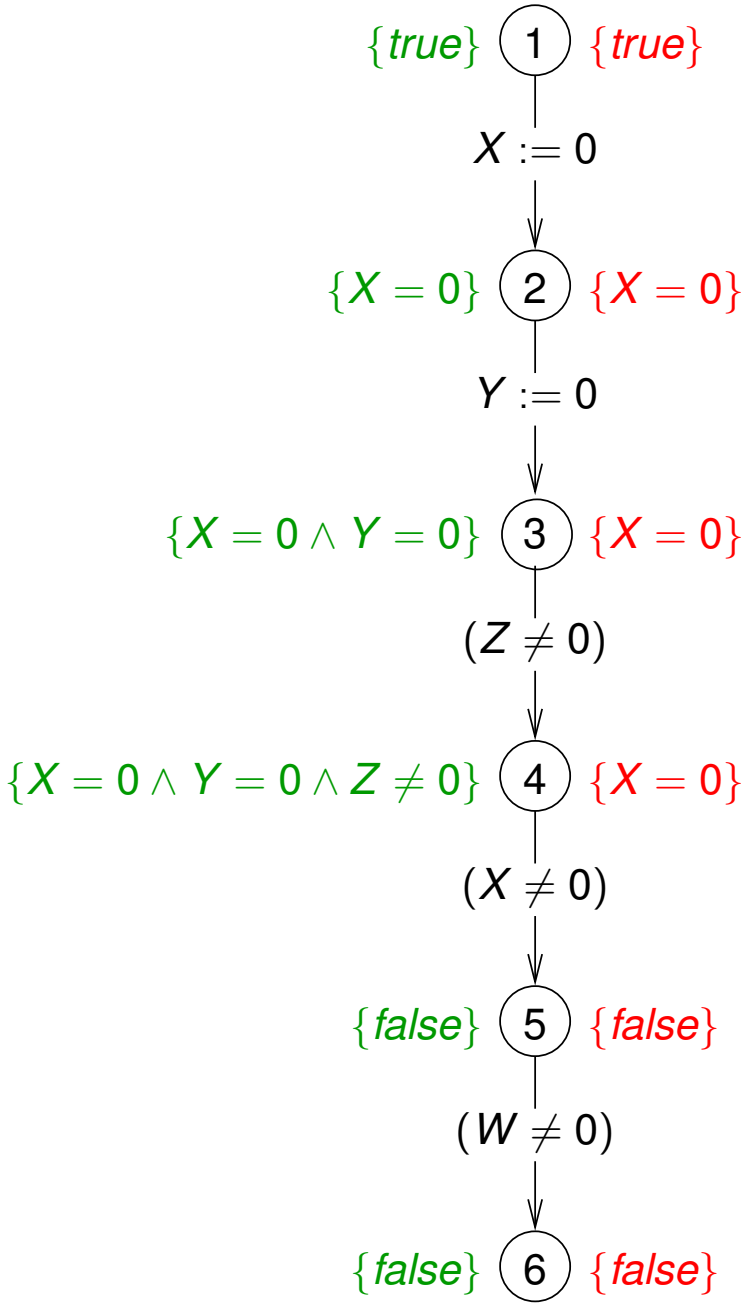
Conciliated Interpolants as a Simplification Procedure



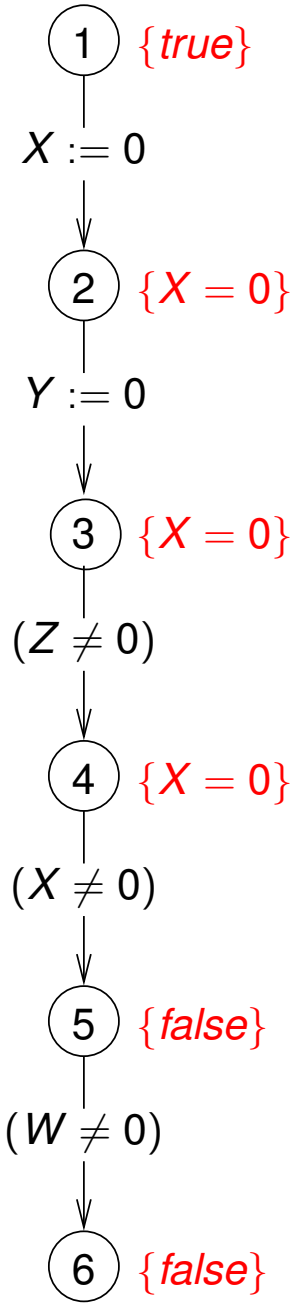
Conciliated Interpolants as a Simplification Procedure



Conciliated Interpolants as a Simplification Procedure



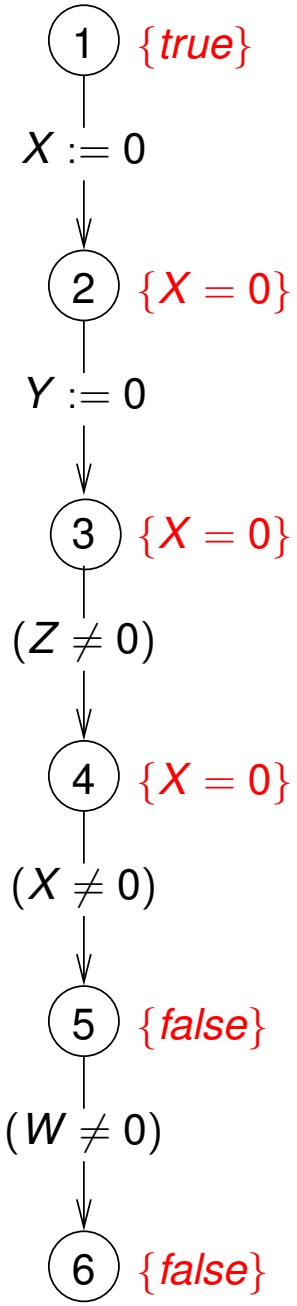
Conciliated Interpolants



Conciliated Interpolants

lead to predicates on fewer variables

Conciliated Interpolants

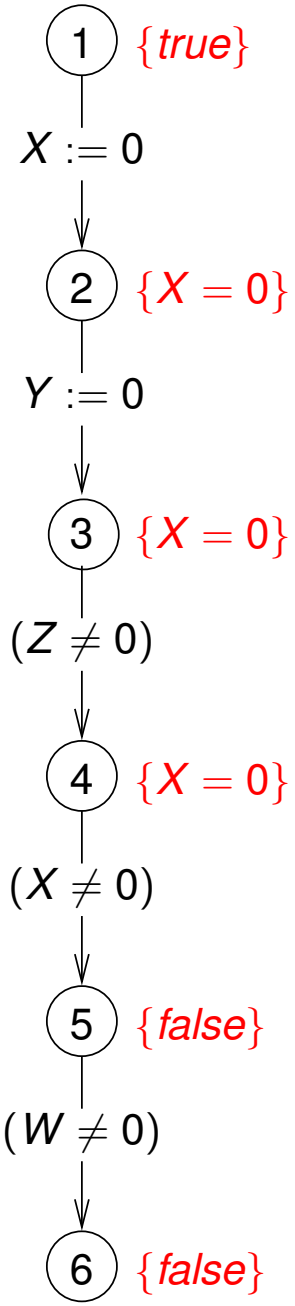


Conciliated Interpolants

lead to predicates on fewer variables

$\Rightarrow$ faster computation

Conciliated Interpolants



Conciliated Interpolants

lead to predicates on fewer variables

$\Rightarrow$ faster computation

$\Rightarrow$ more meaningful predicates

A Locking Example

```
struct file {
        bool locked;
        int pos;
};
open(file f) {
        assert(¬f.locked);
        f.locked = true;
        f.pos = 0;
}
close(file f) {
        assert(f.locked ∨
                f.pos==0);
        f.locked = false;
}
```

```
rw(file f) {
        assert(f.locked ∨ f.pos==0);
        f.pos = f.pos + 1;
}
main() {
        file f1,f2;
        f1.locked = f2.locked = false;
        open(f1);
        while(*) {
                open(f2);
                while(*) { rw(f2); rw(f1); }
                close(f2);
        }
        close(f1);
}
```

Experimental Results

	time/s	memory (BDD nodes)	# cycles
w/o abstraction	460	440482	n/a
weakest interp.	0.43	89936	14
concil. interp.	0.29	80738	10

Summary

- Craig interpolation goes well with CEGAR if the program is given in terms of BDDs.
- Multiple counterexamples can be excluded at once.
- There are heuristics to enhance predicate generation.
- The model-checking process can be speeded up.