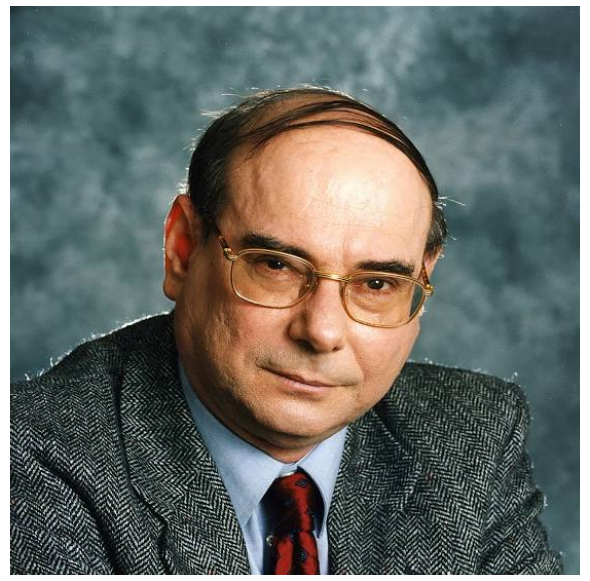


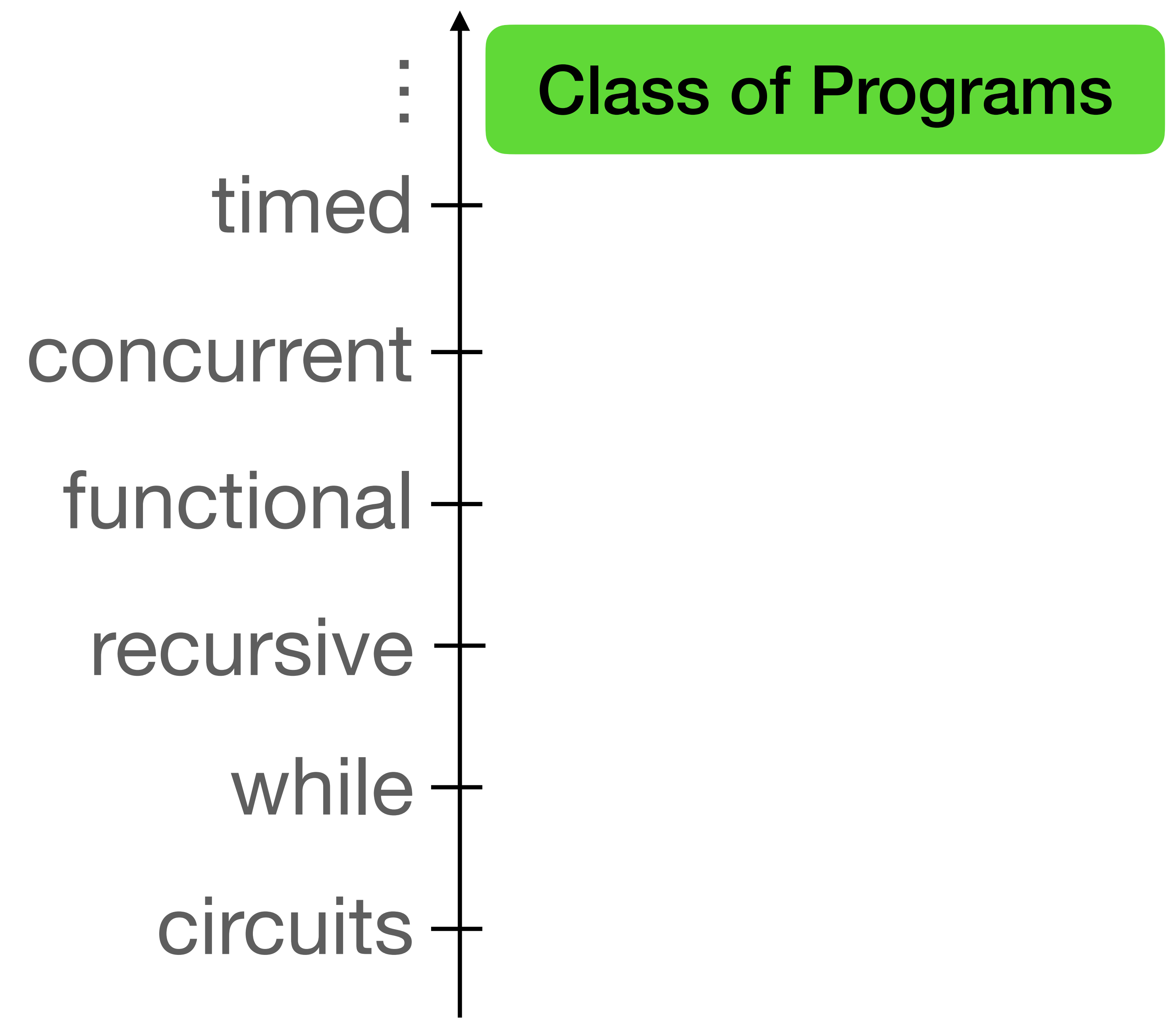
Color coding ahead!

IC3

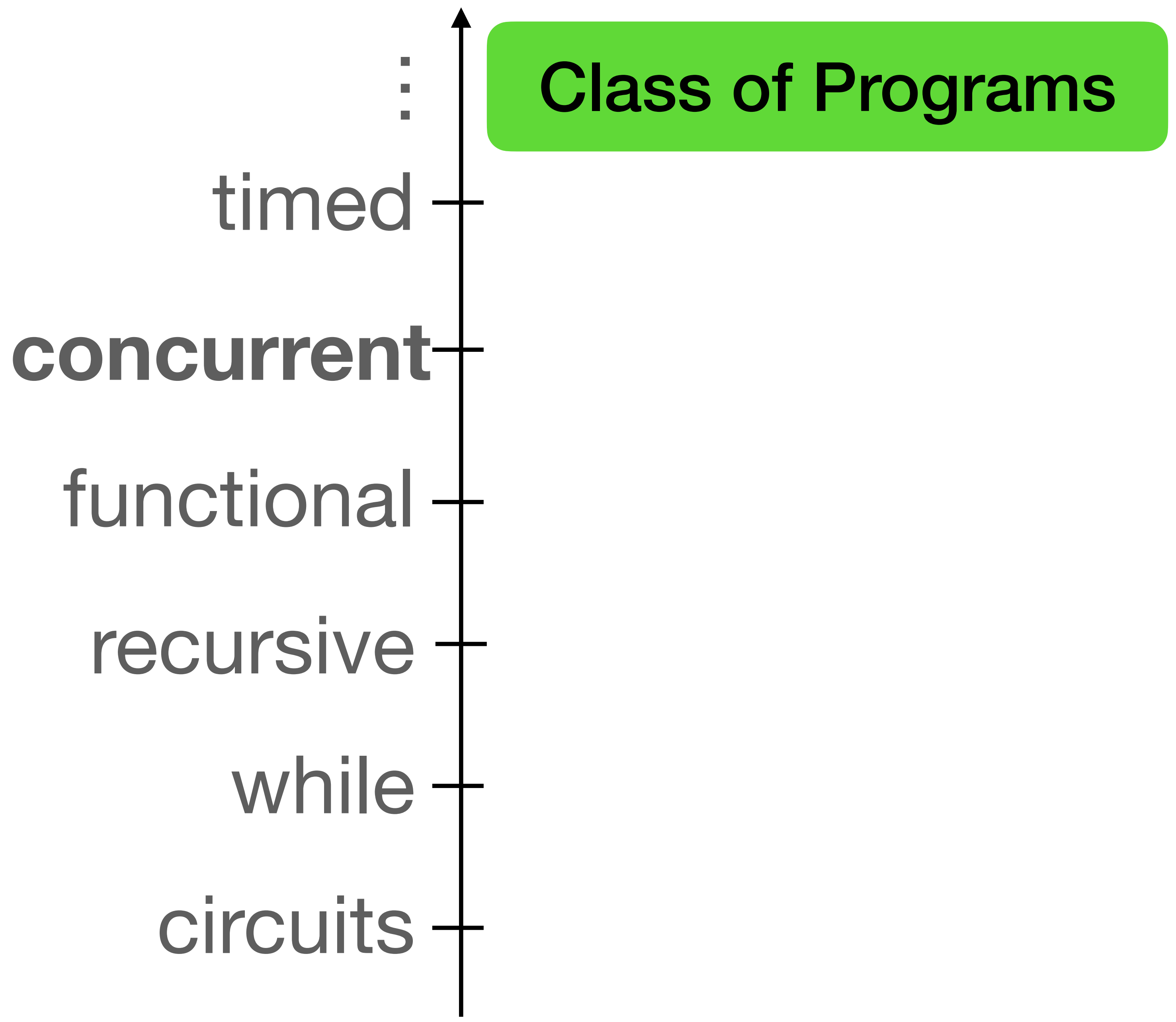
Incremental Construction of Inductive Clauses for Indubitable Correctness

Roland Meyer, TU Braunschweig

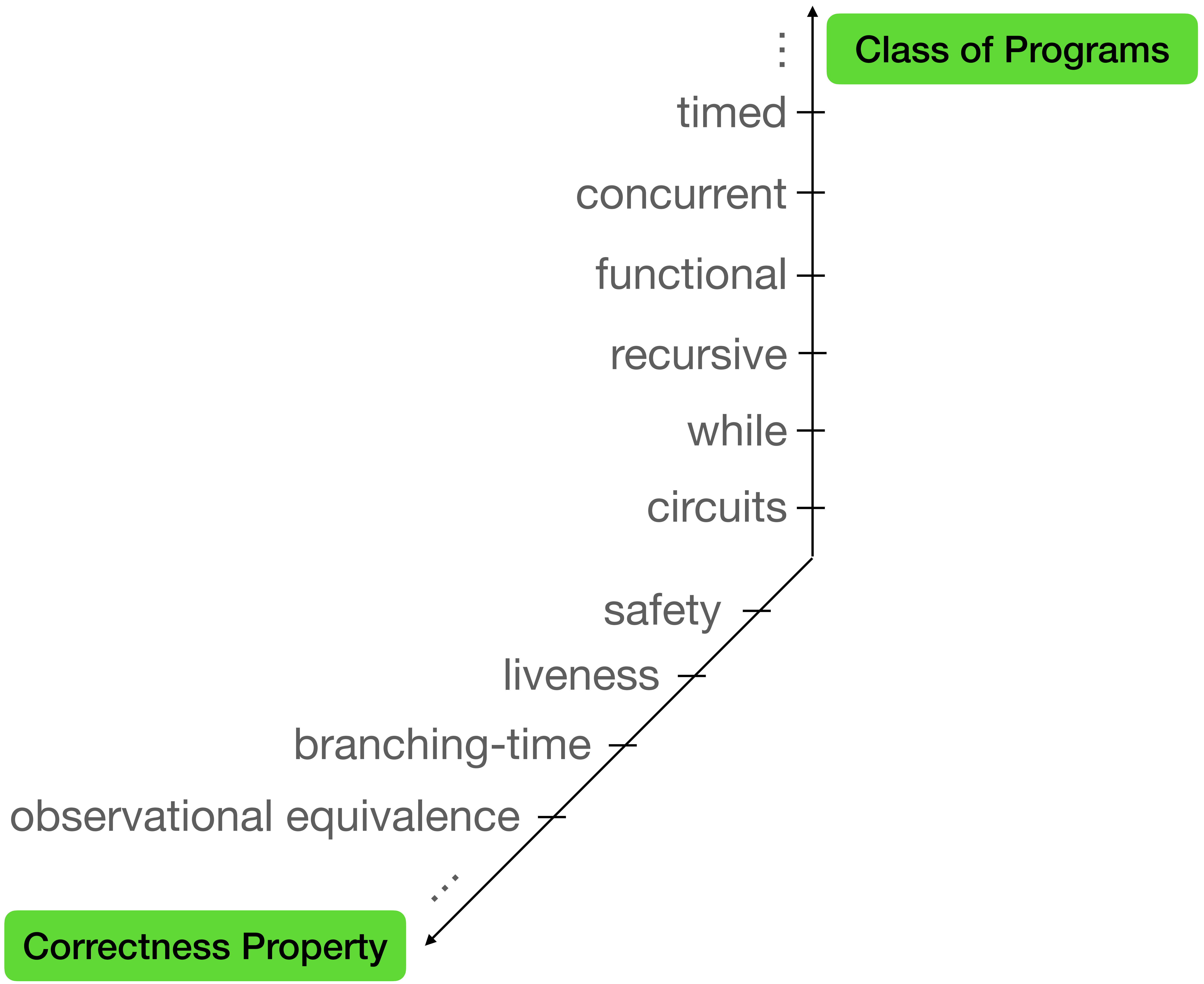
Program Verification



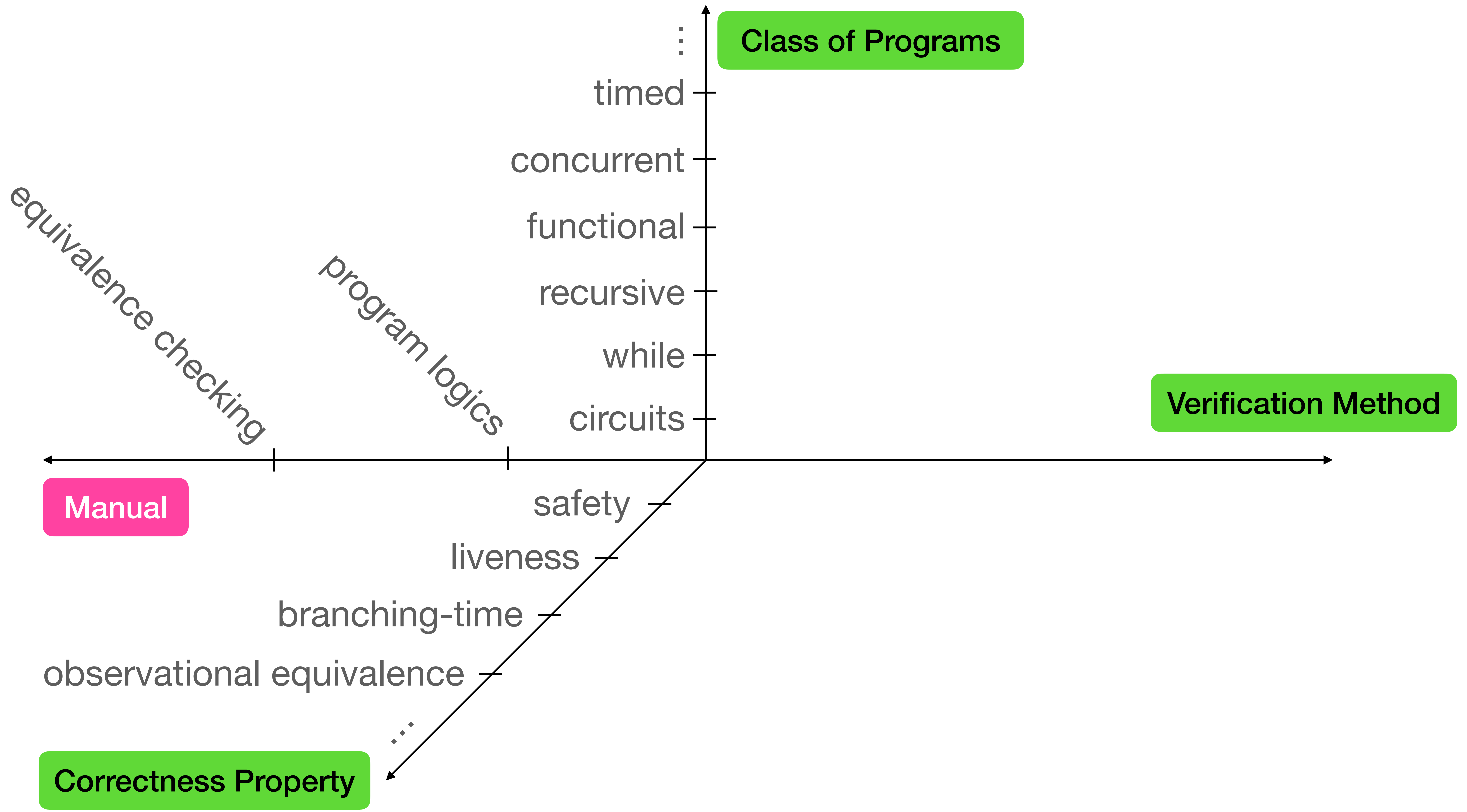
Program Verification



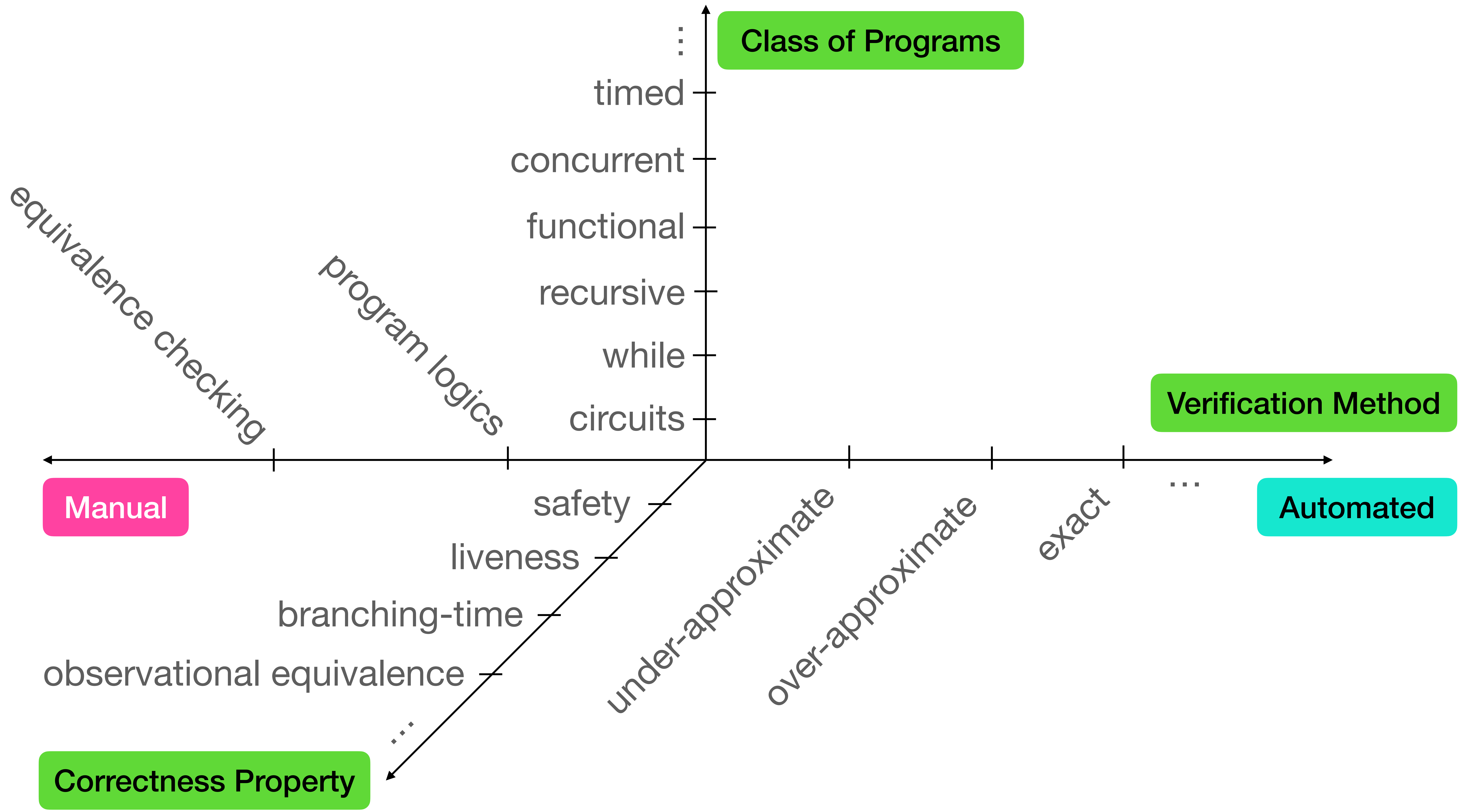
Program Verification



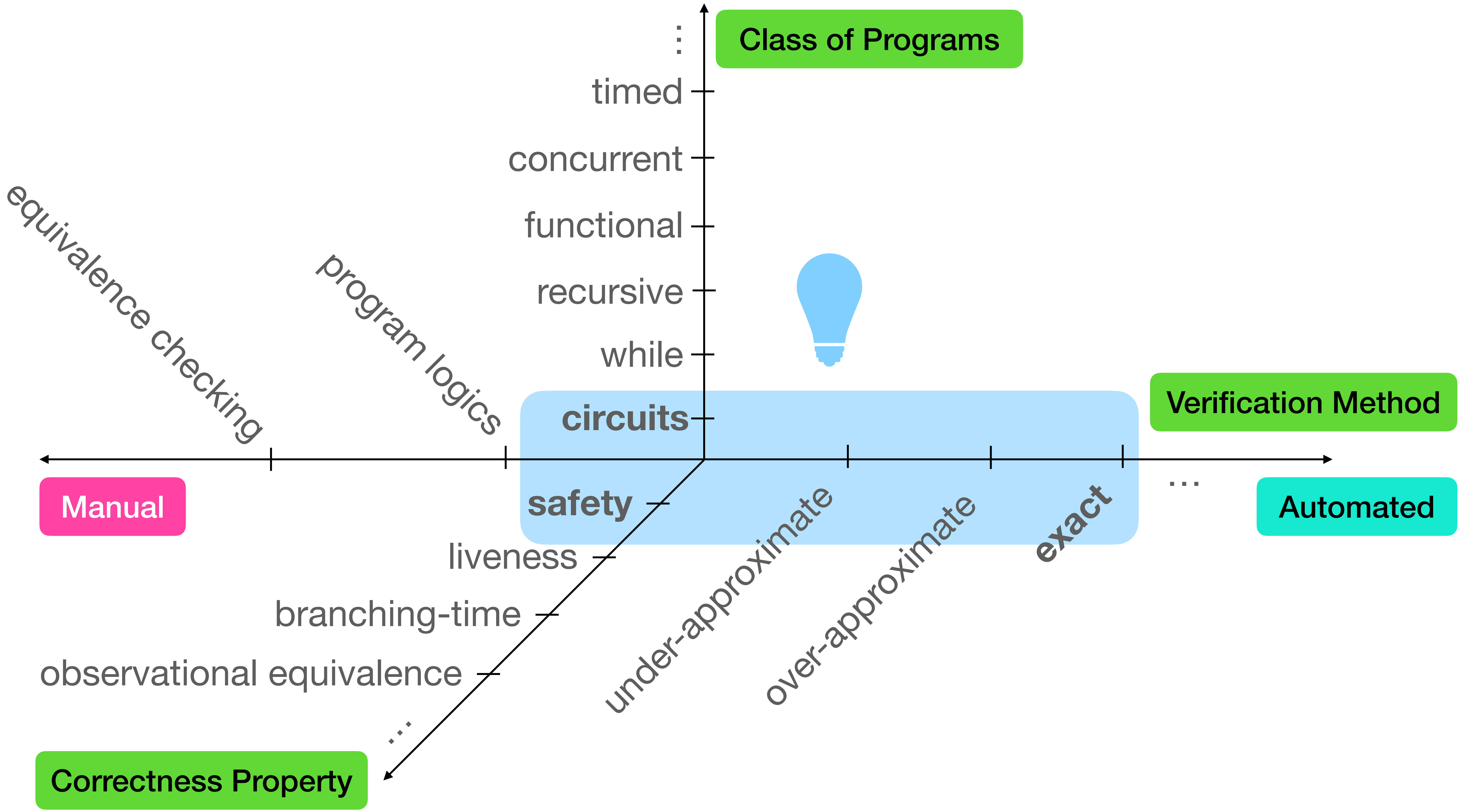
Program Verification



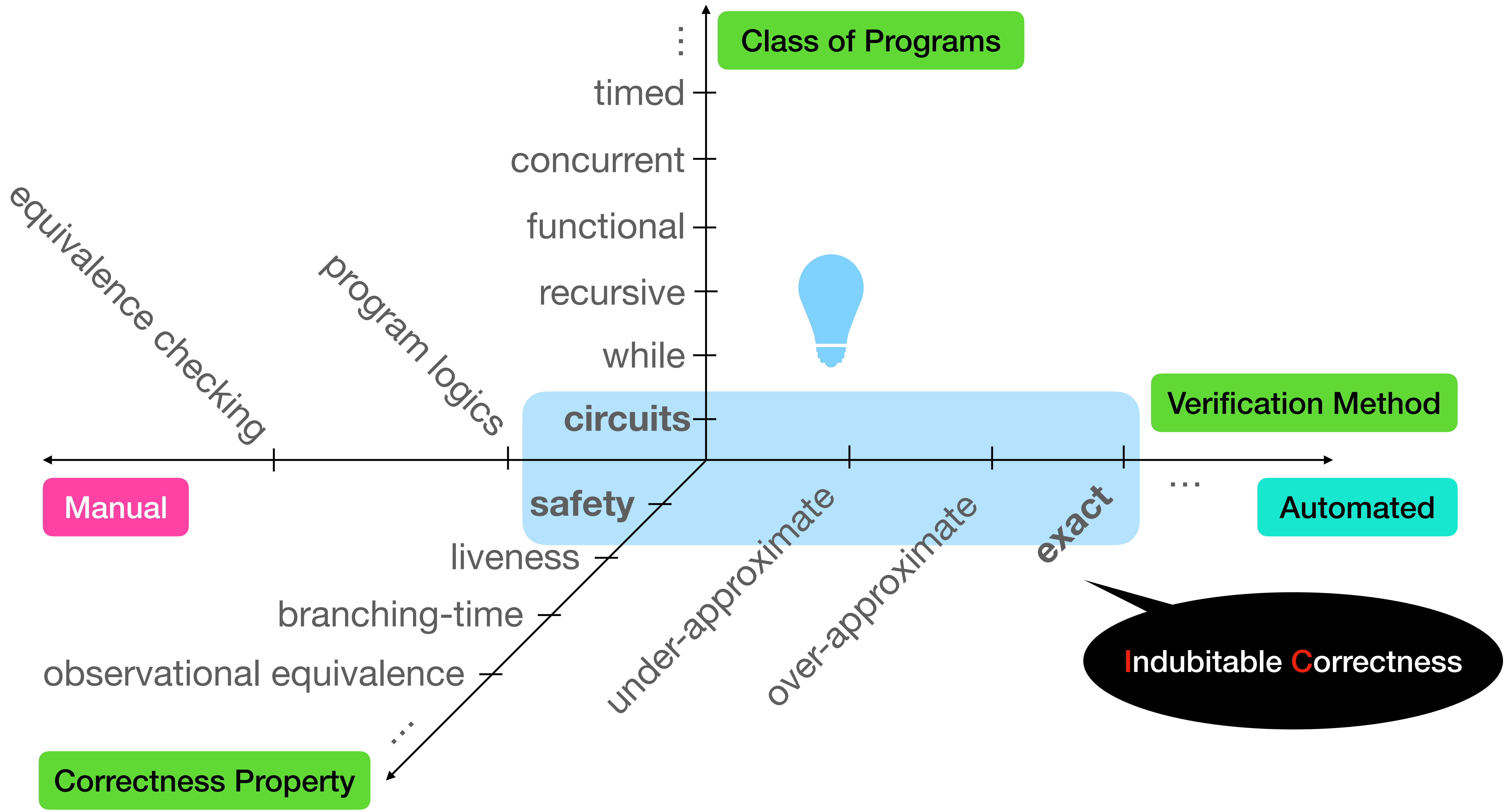
Program Verification



Program Verification

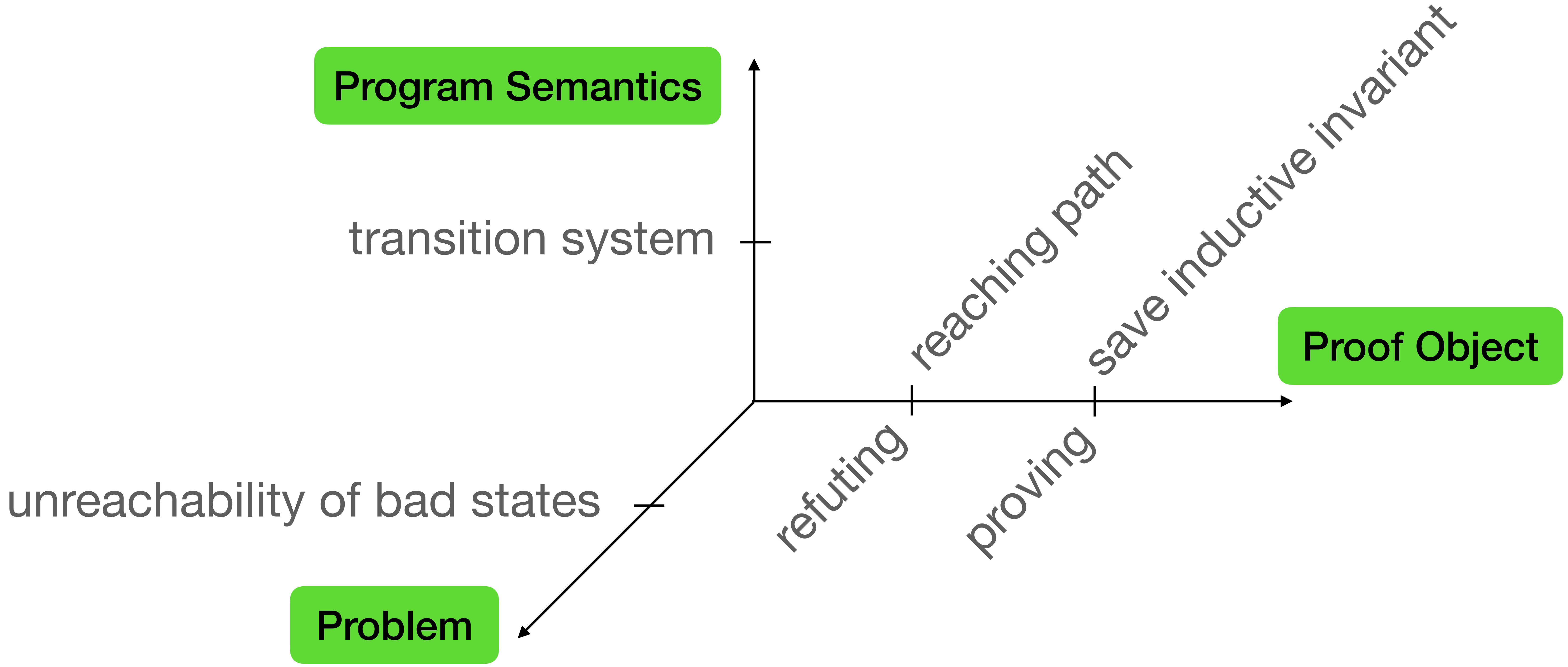


Program Verification



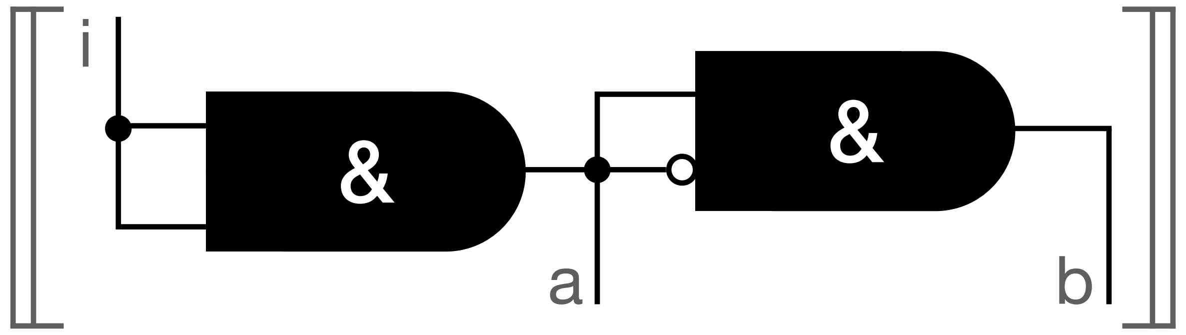
Automated Safety Verification

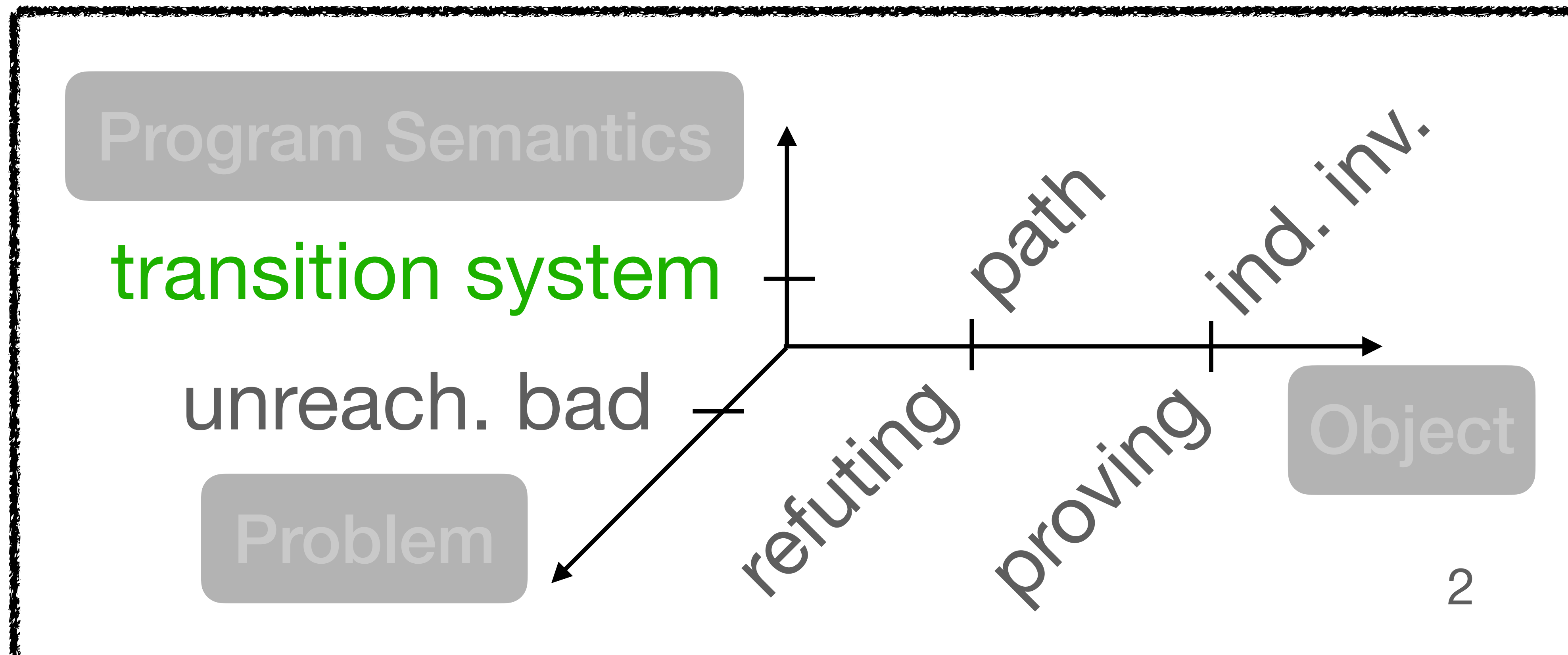
Recap



Automated Safety Verification

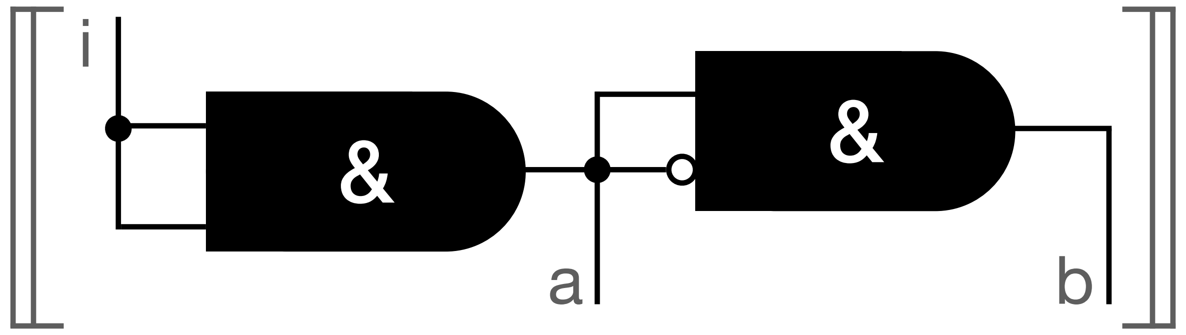
Transition System:



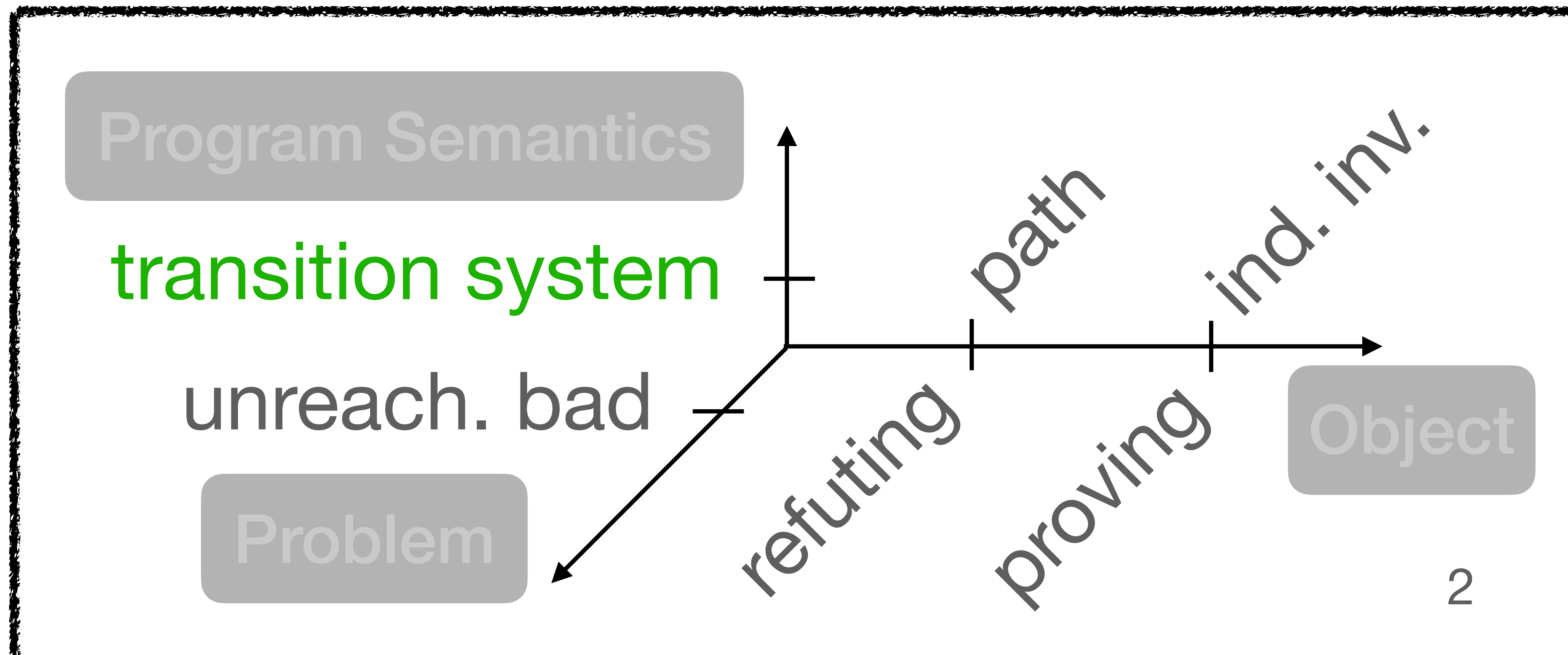


Automated Safety Verification

Transition System:

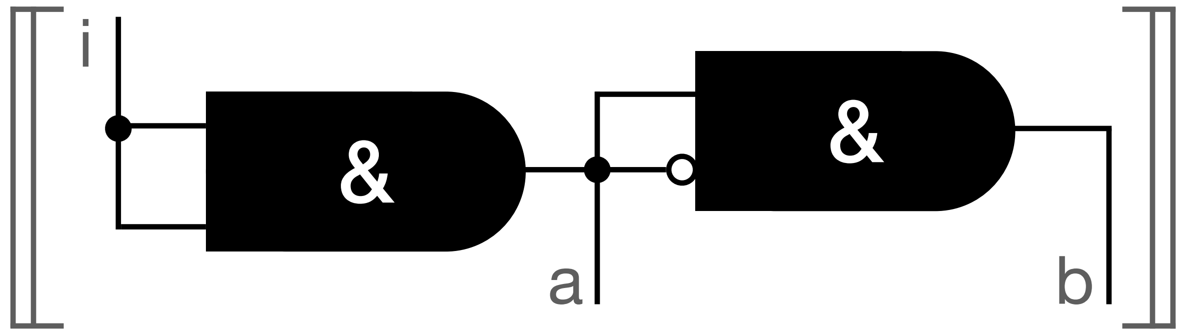


= → 000

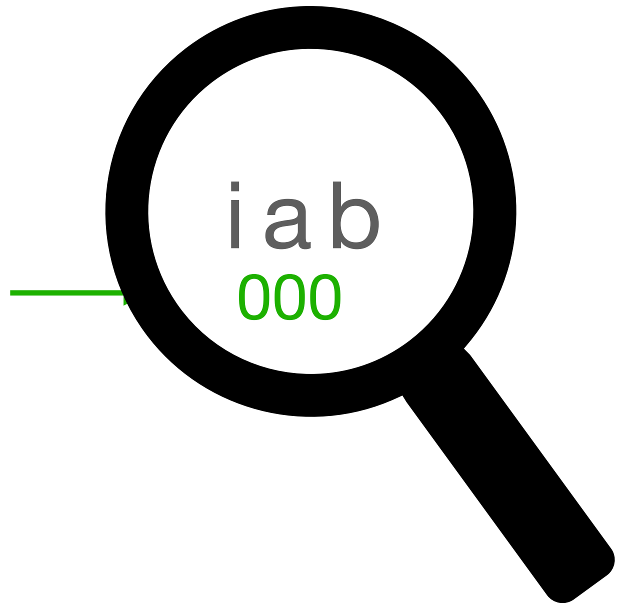


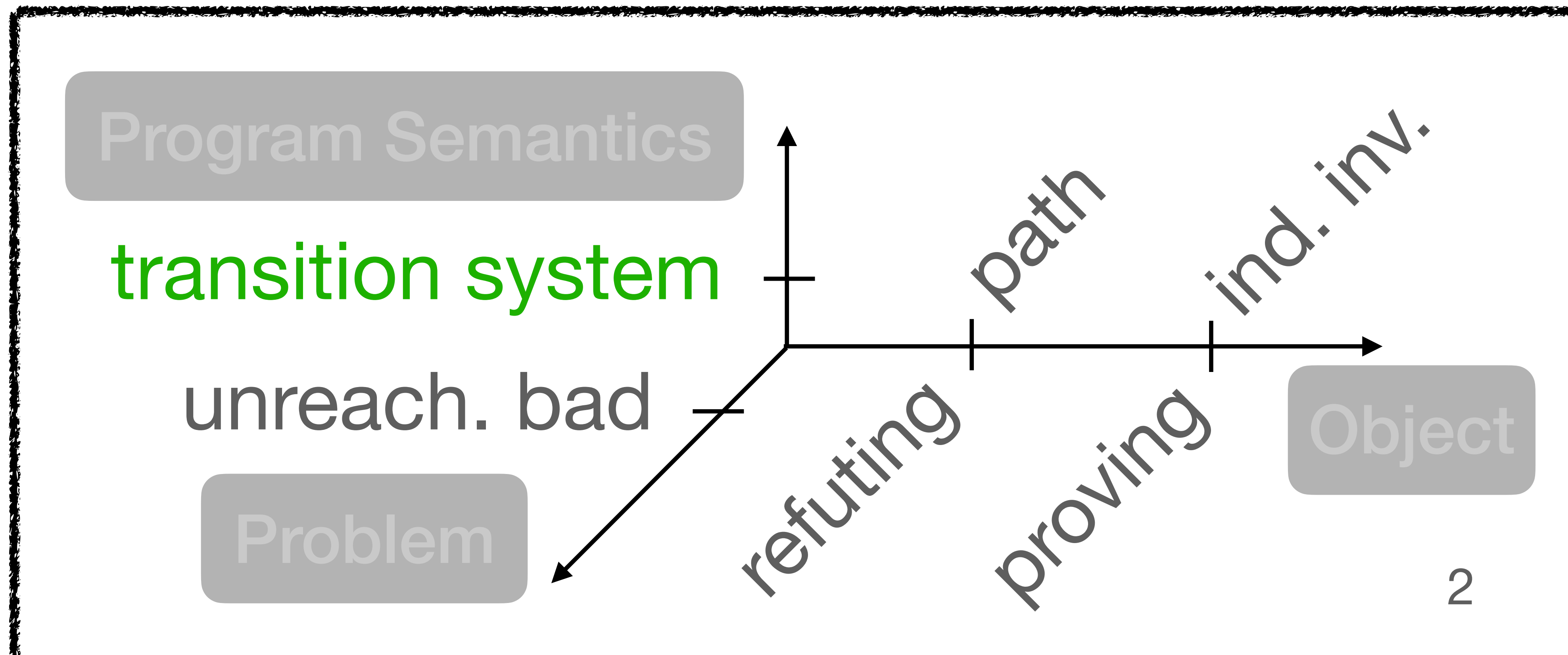
Automated Safety Verification

Transition System:



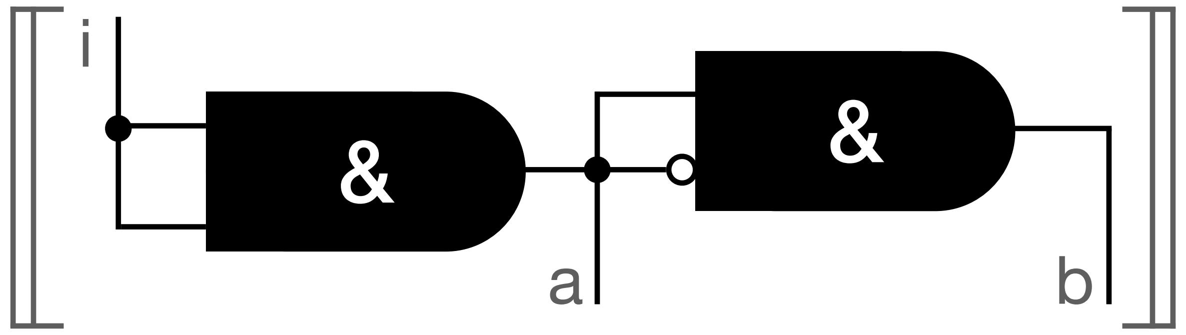
=



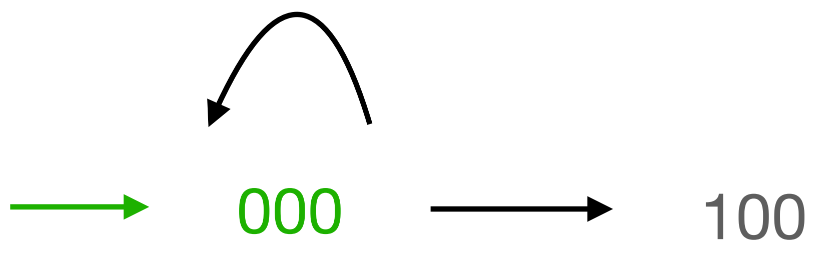


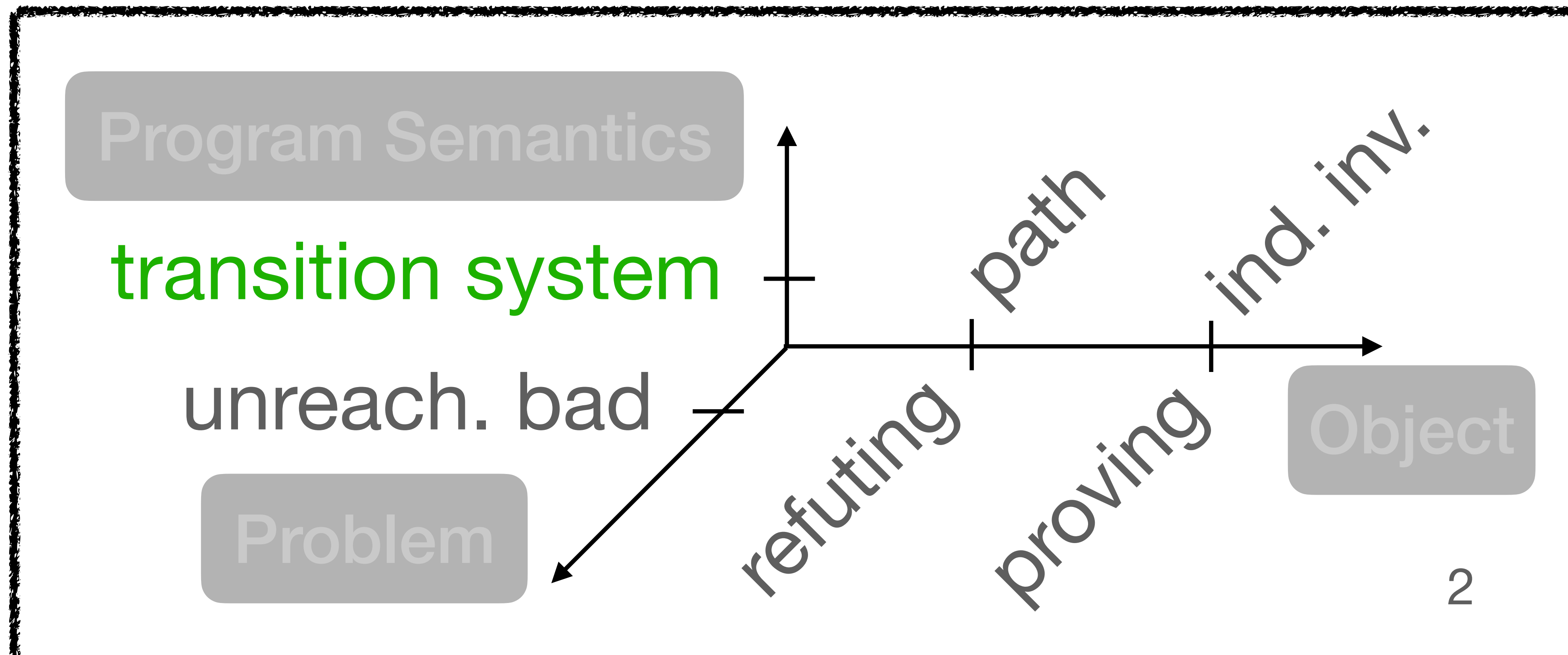
Automated Safety Verification

Transition System:



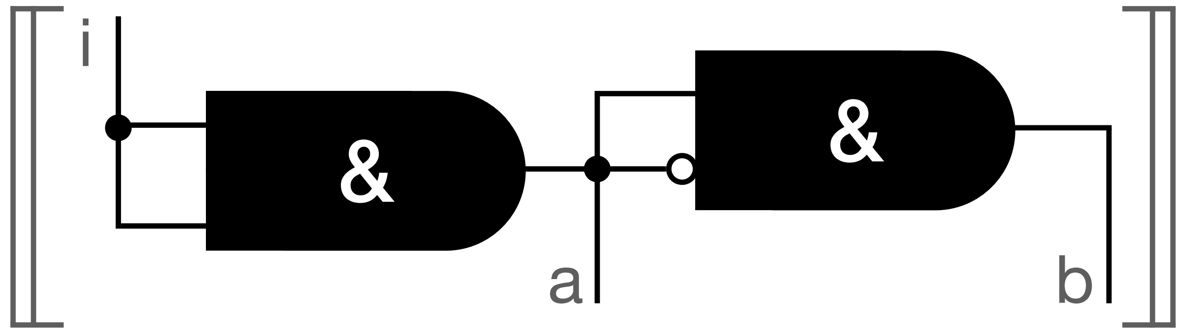
=

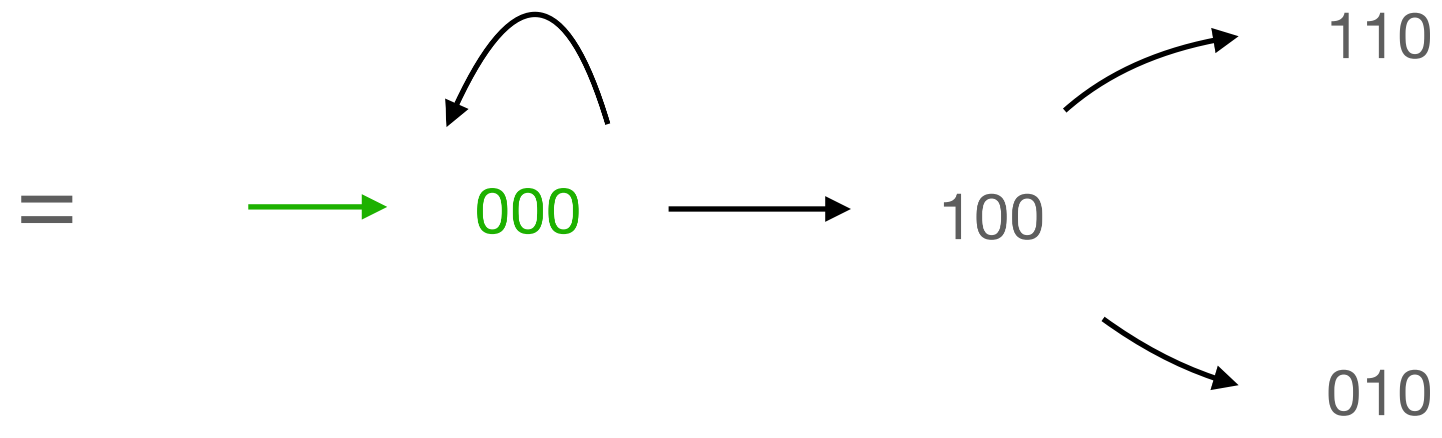


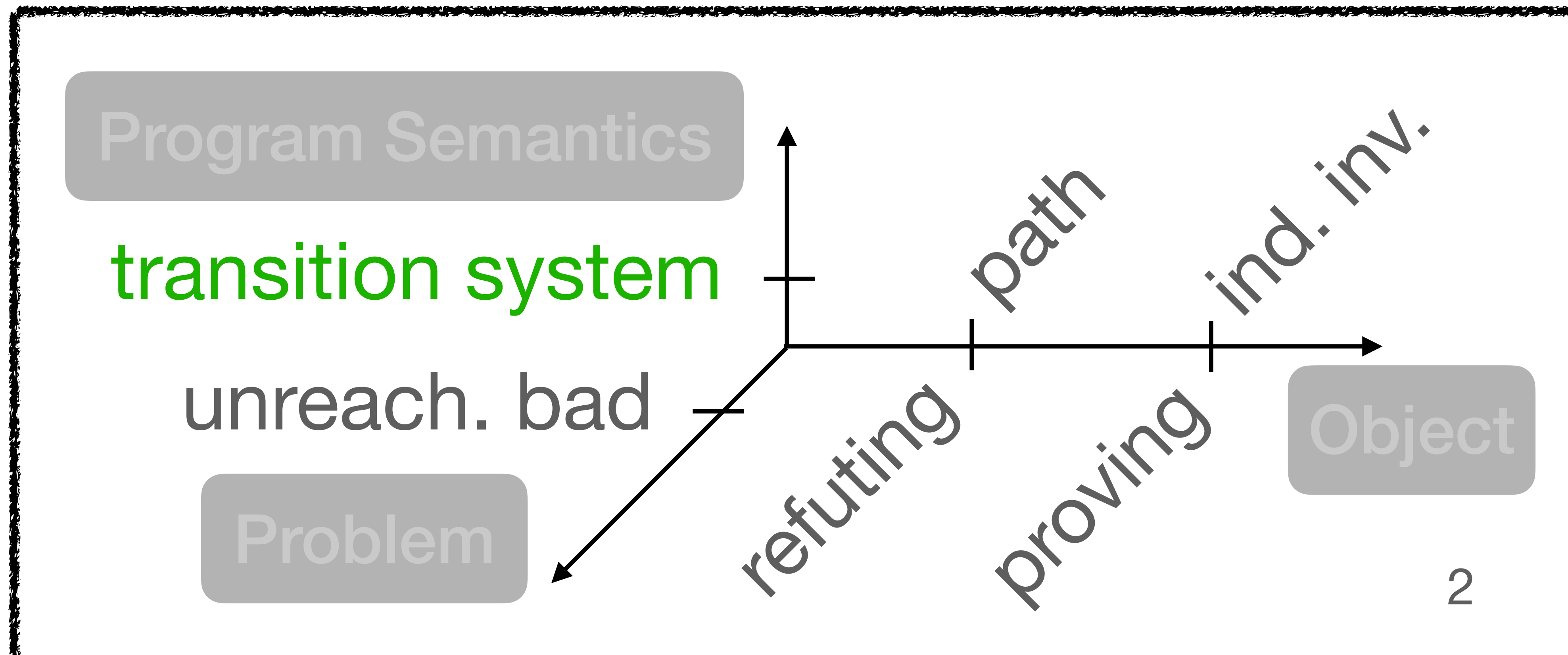


Automated Safety Verification

Transition System:

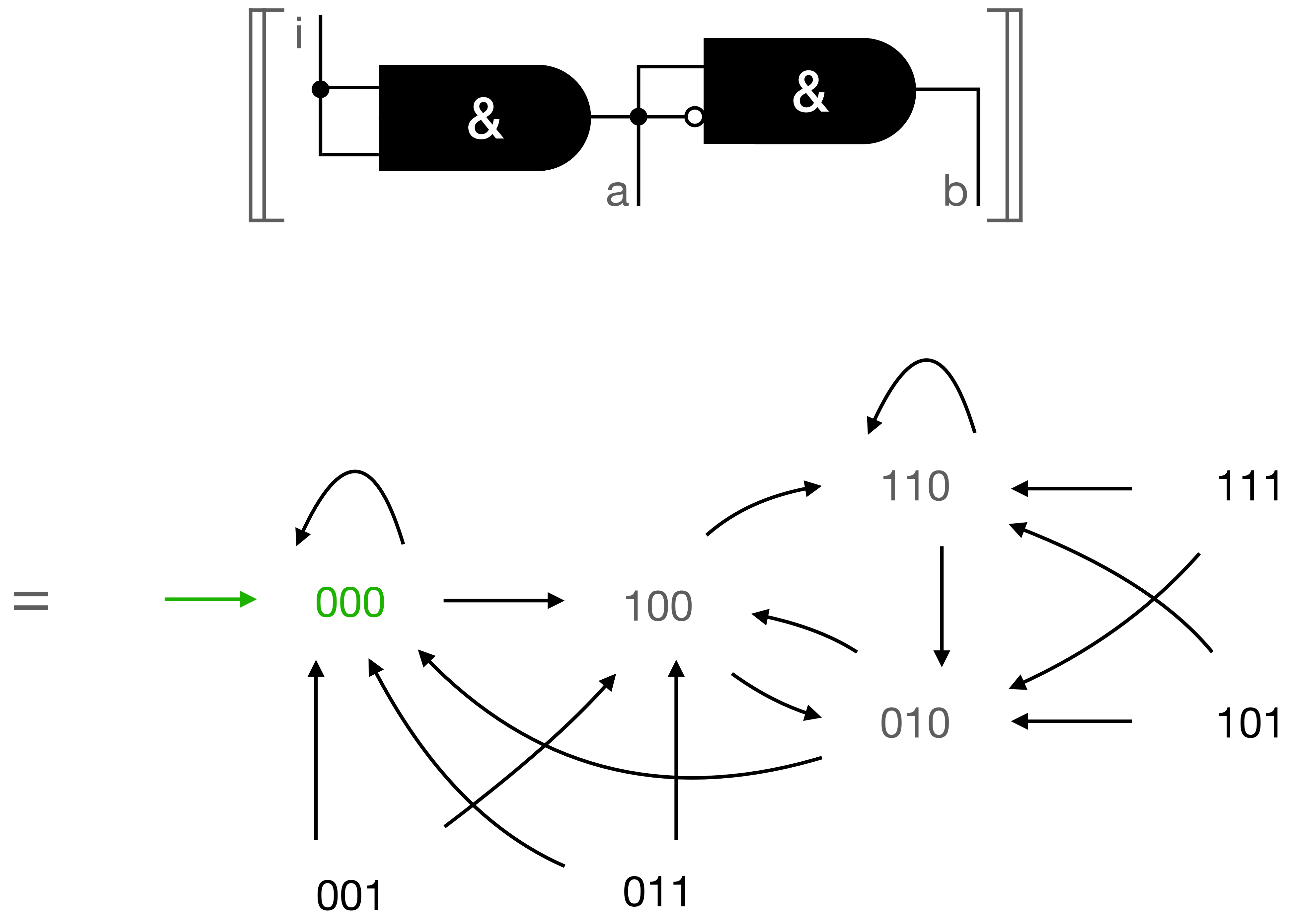


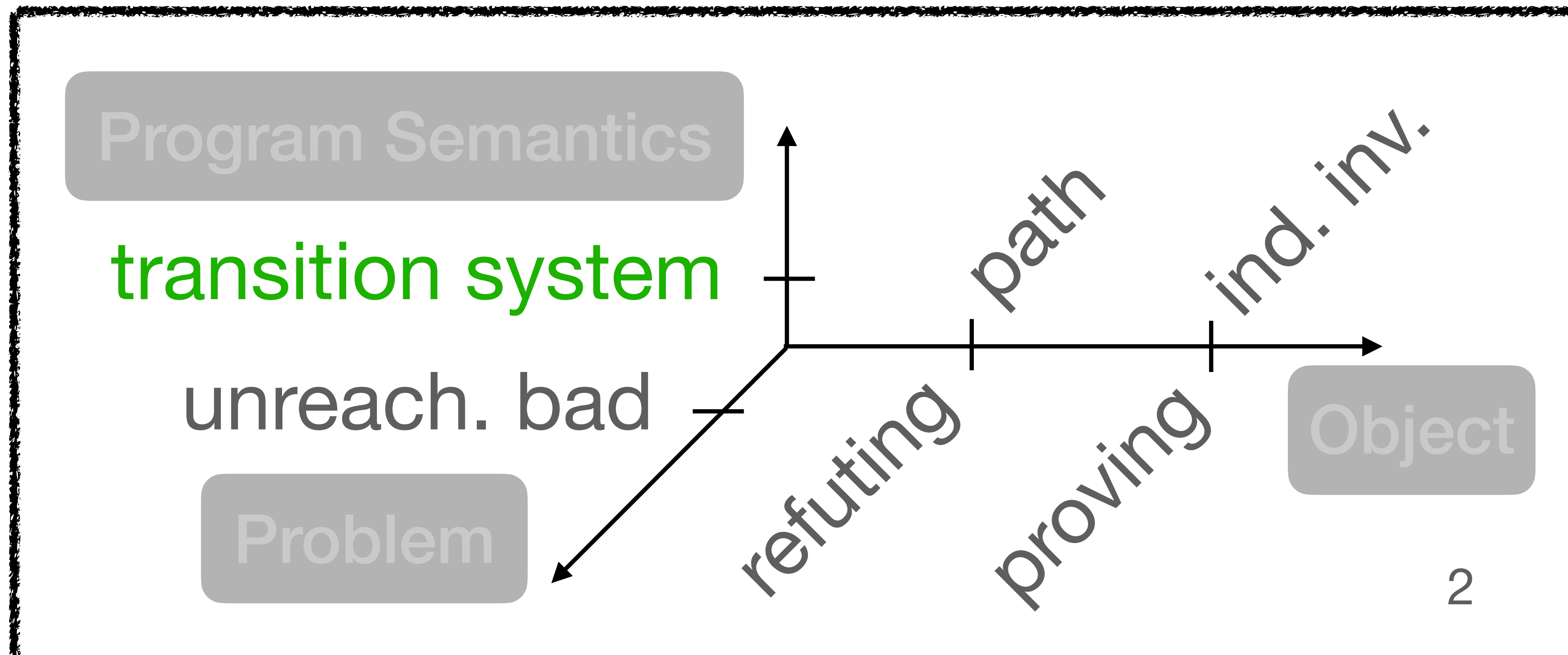




Automated Safety Verification

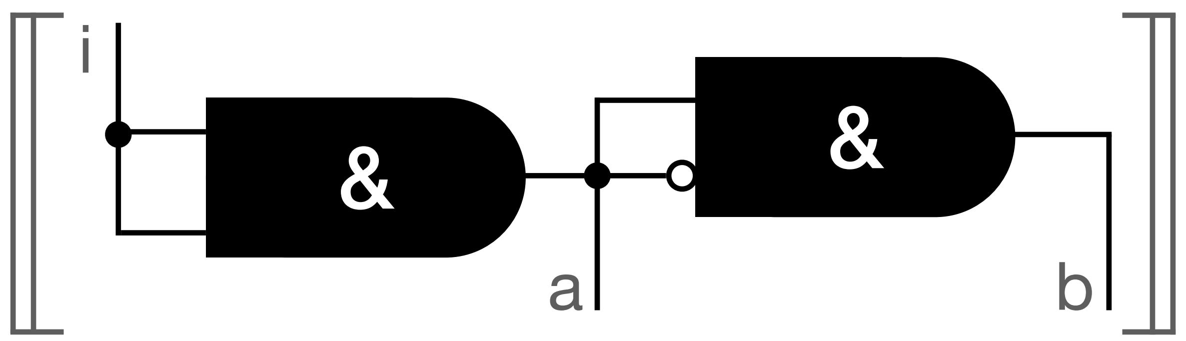
Transition System:

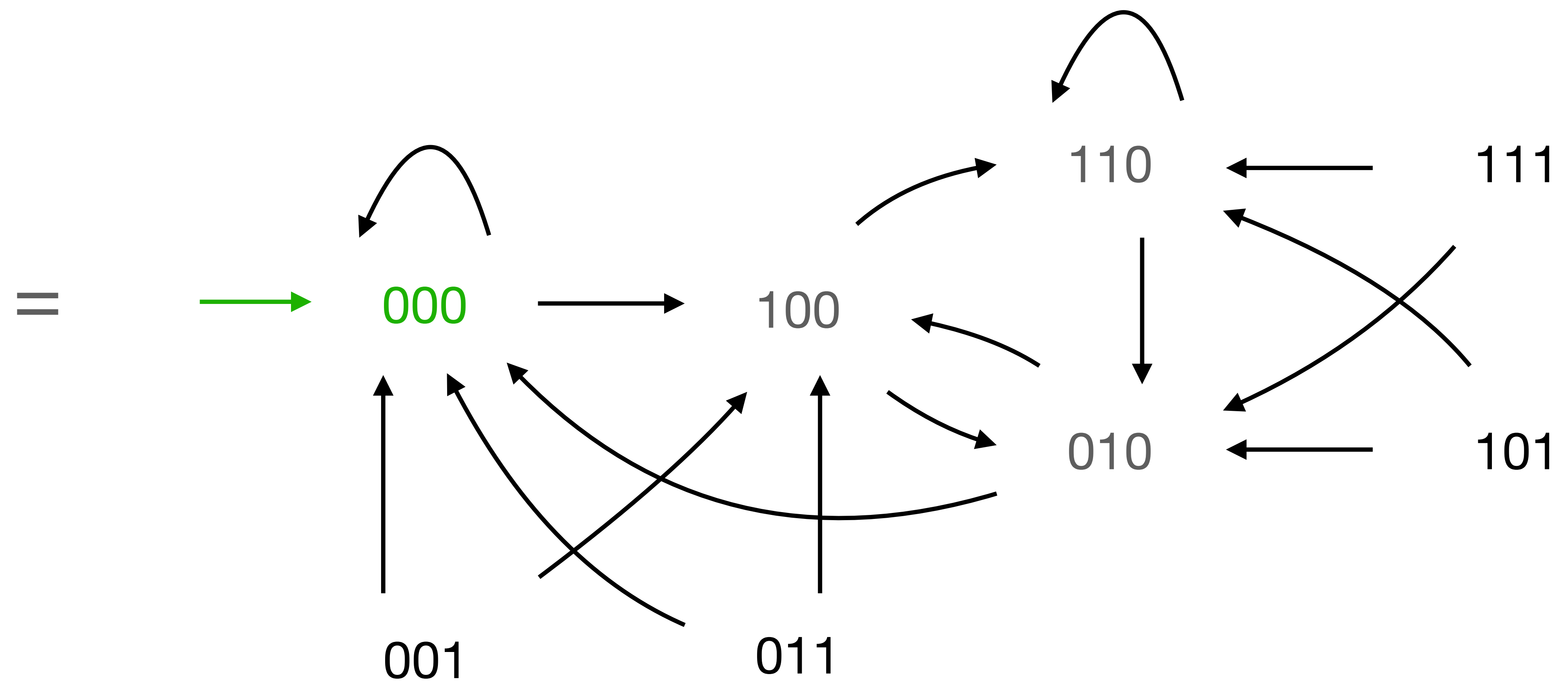




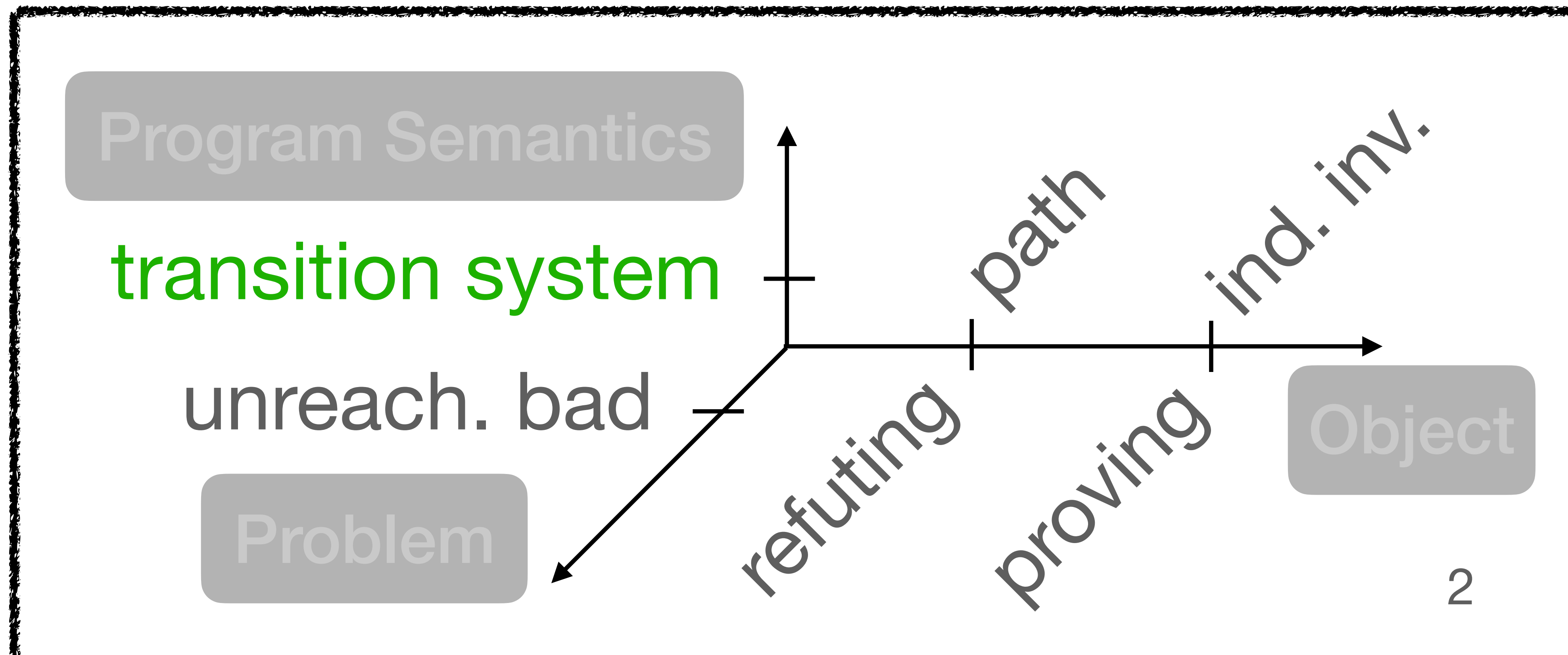
Automated Safety Verification

Transition System:



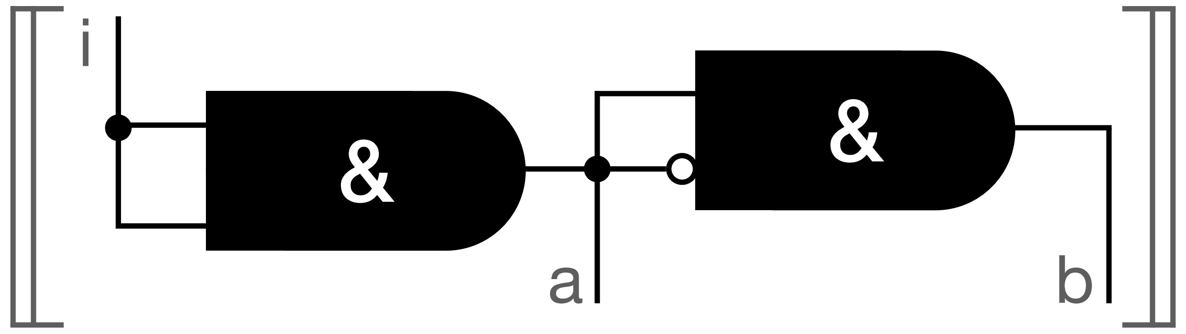


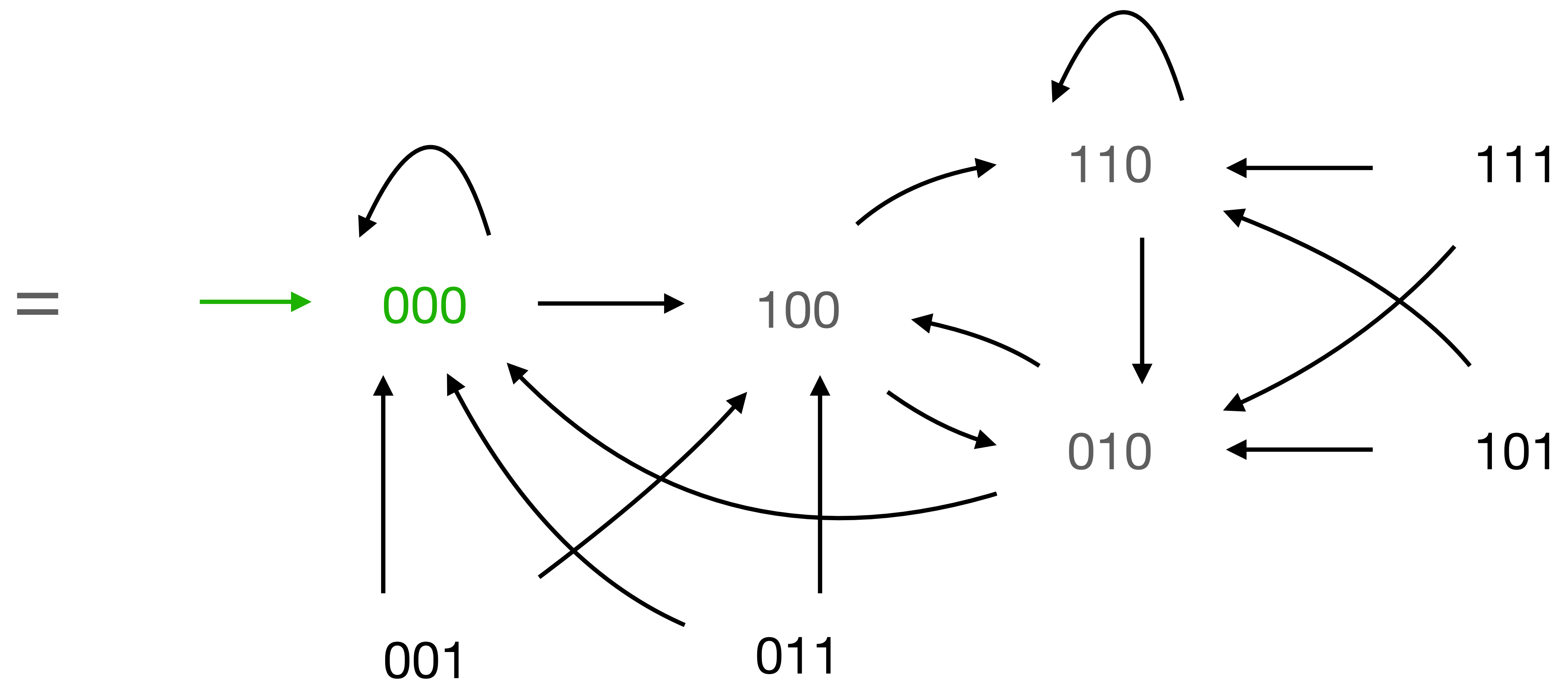
= (States, Init, post)



Automated Safety Verification

Transition System:

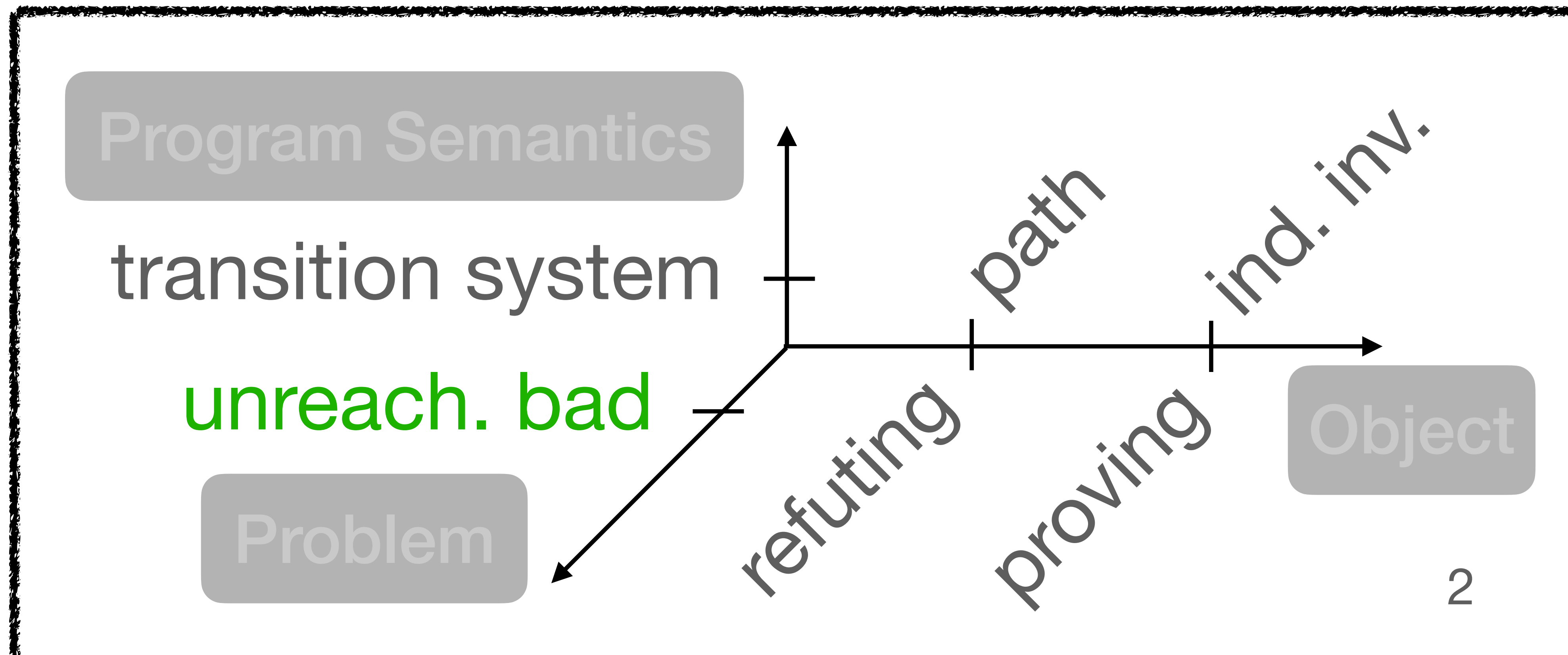




= (States, Init, post)

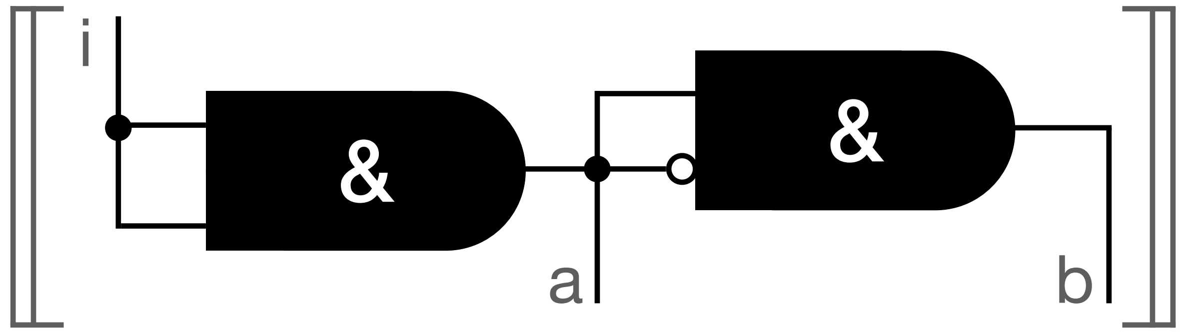
Reachable States:

$$\mathrm{post}^*(\mathrm{Init}) = \bigcup_{i \in \mathbb{N}} \mathrm{post}^i(\mathrm{Init}) \subseteq \mathrm{States}$$

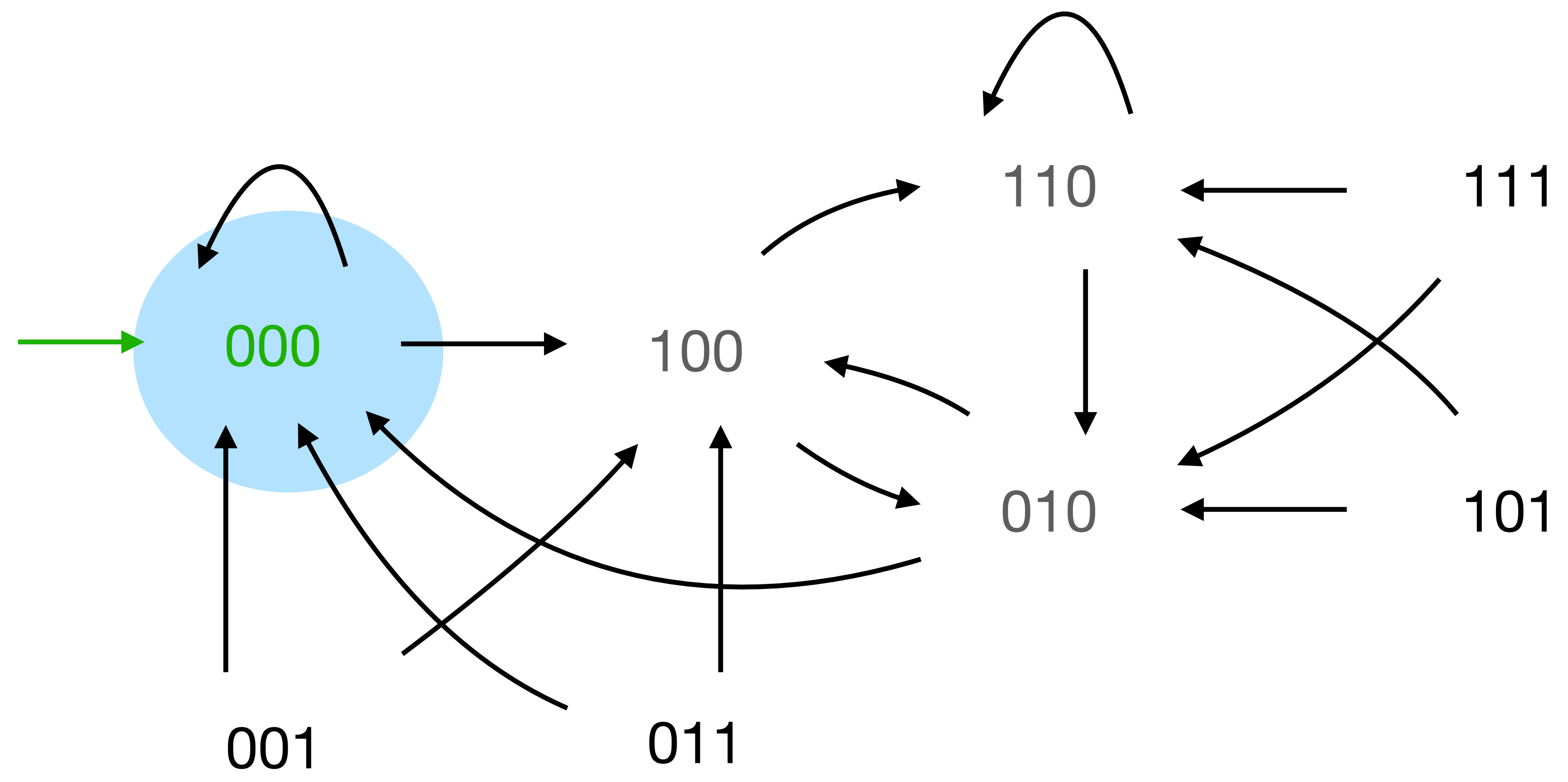


Automated Safety Verification

Transition System:



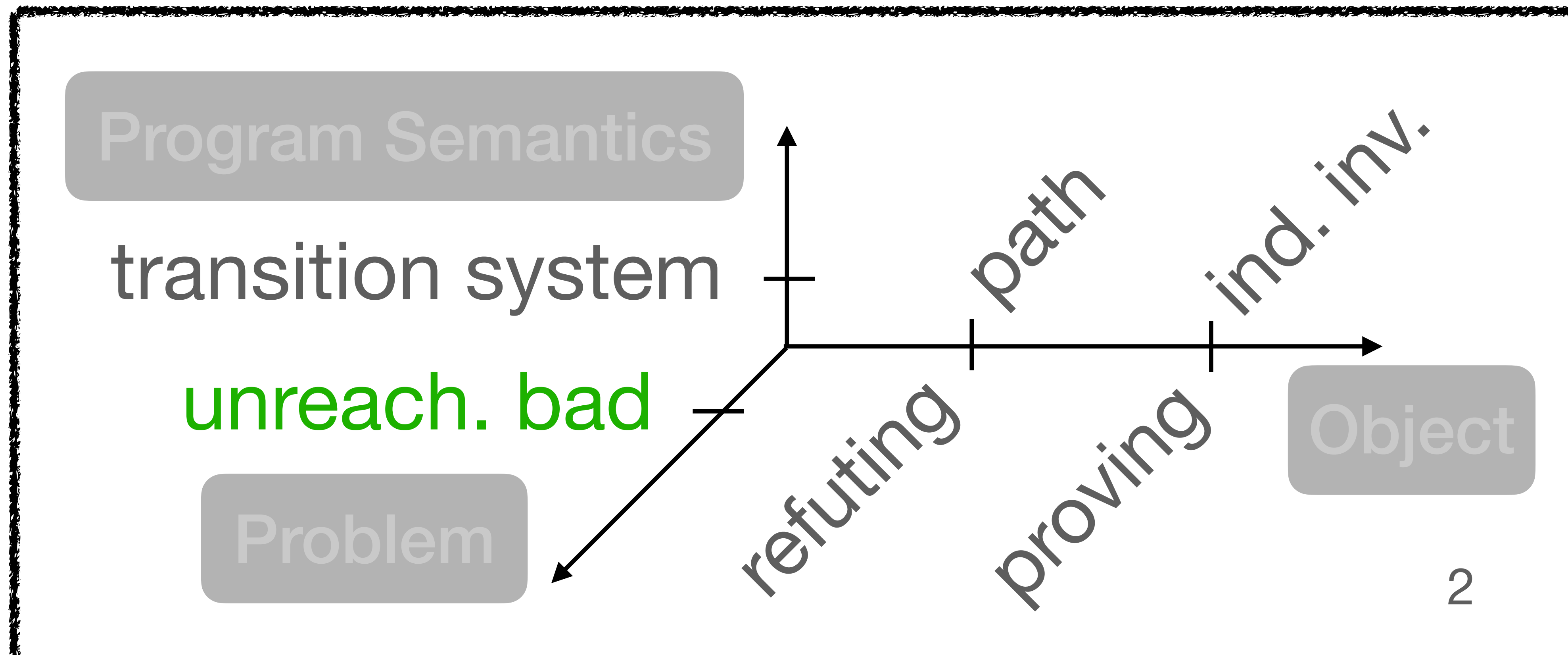
=



= (States, Init, post)

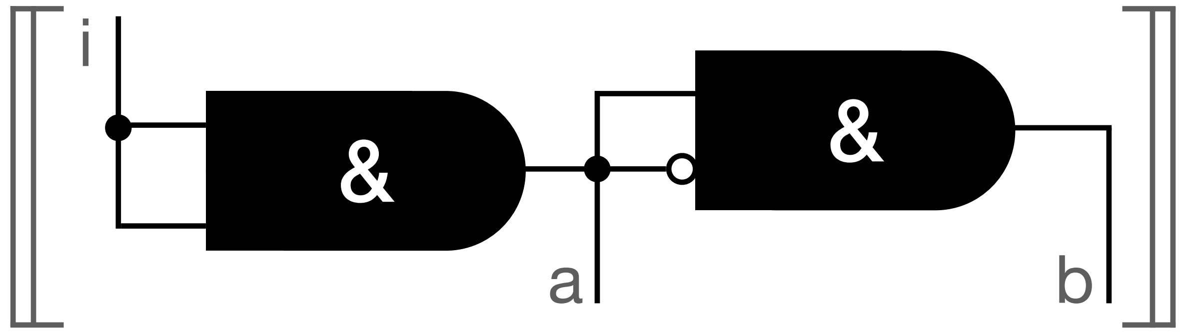
Reachable States:

$$\text{post}^*(\text{Init}) = \bigcup_{i\in\mathbb{N}} \text{post}^i(\text{Init}) \subseteq \text{States}$$

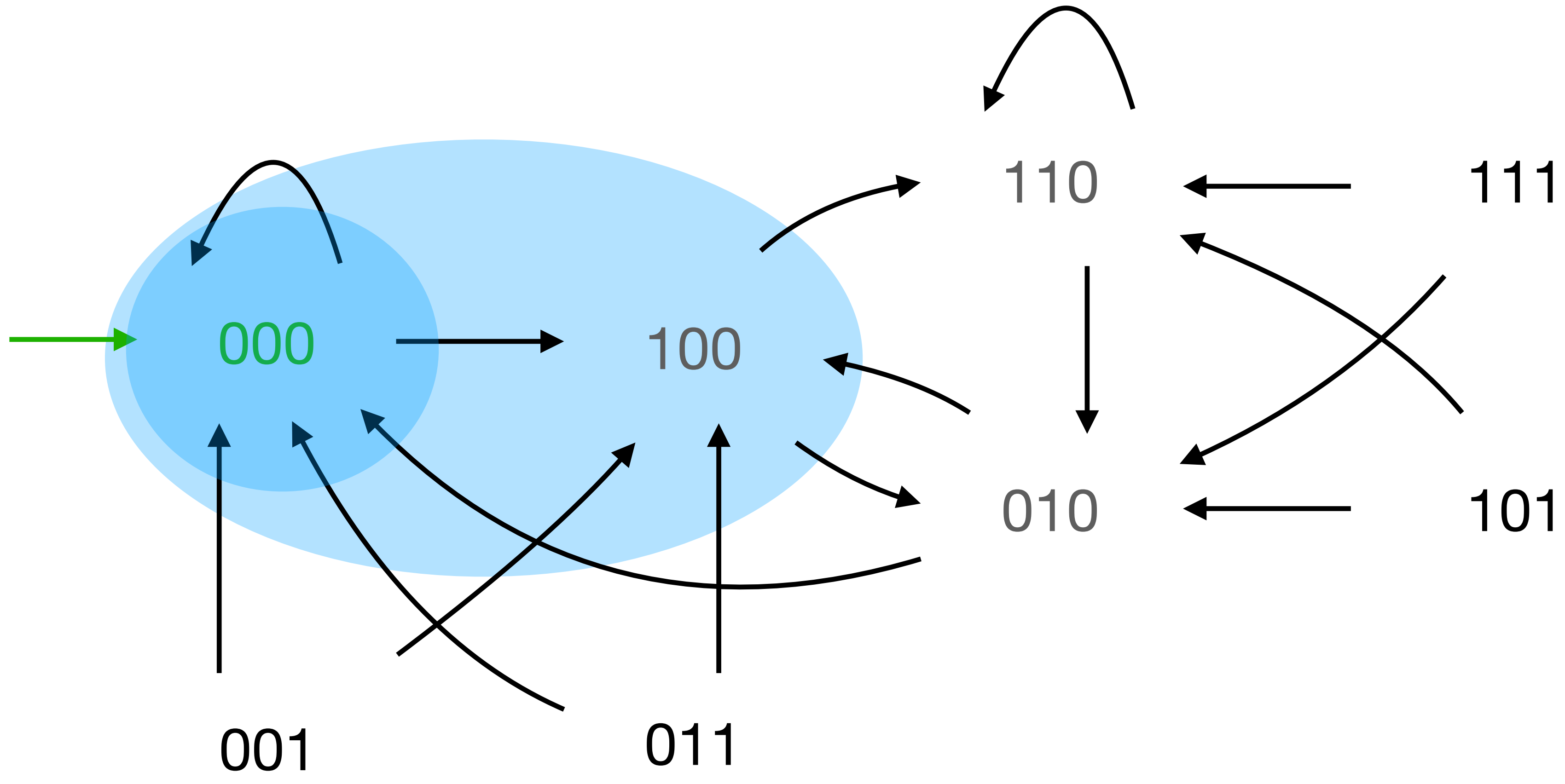


Automated Safety Verification

Transition System:



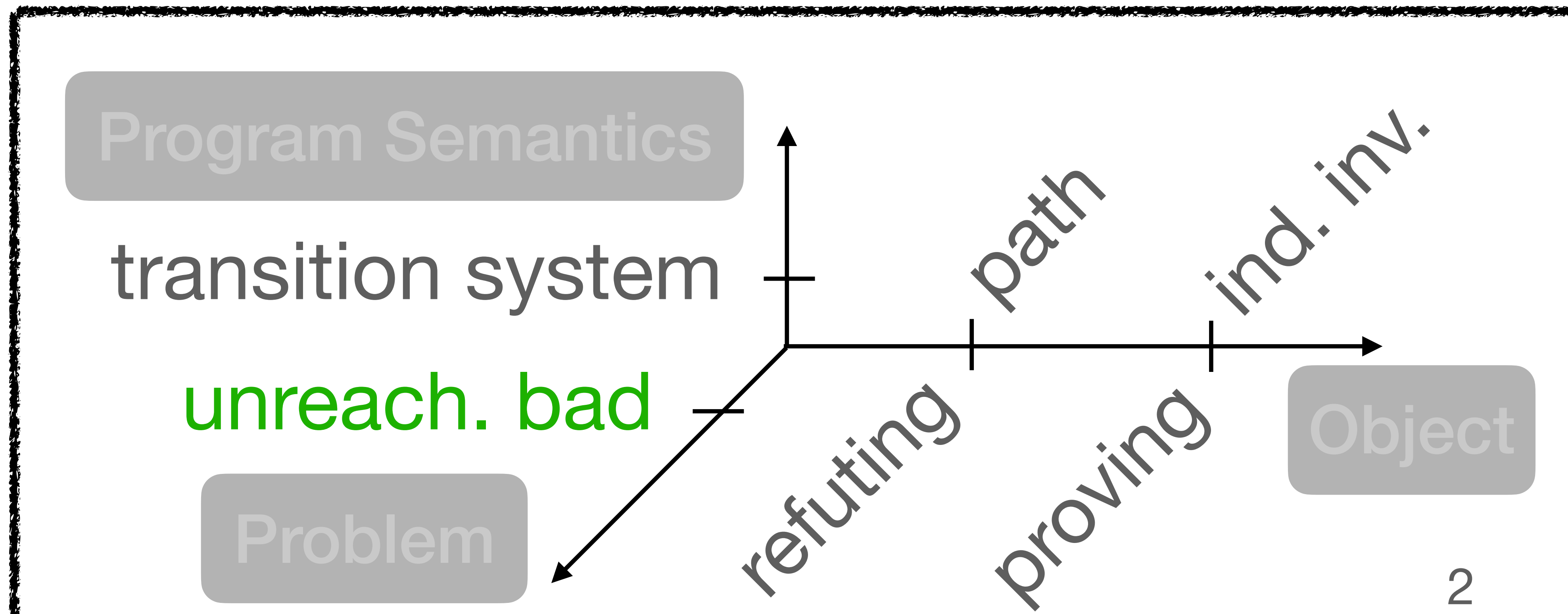
=



= (States, Init, post)

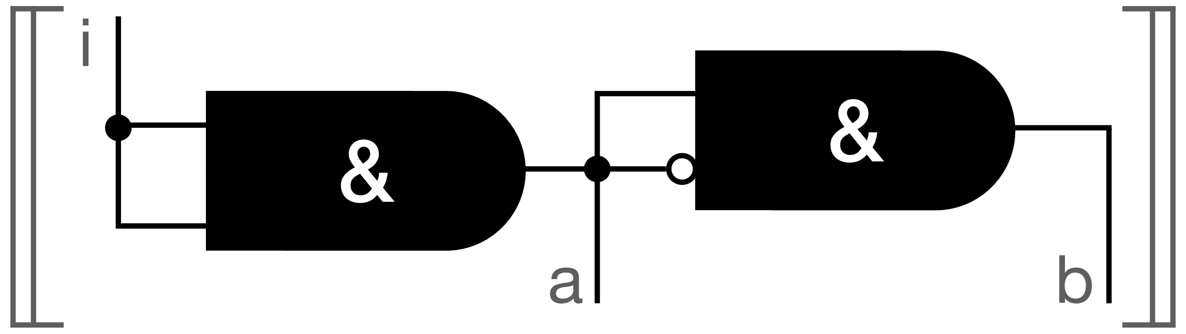
Reachable States:

$$\mathrm{post}^*(\mathrm{Init}) = \bigcup_{i \in \mathbb{N}} \mathrm{post}^i(\mathrm{Init}) \subseteq \mathrm{States}$$

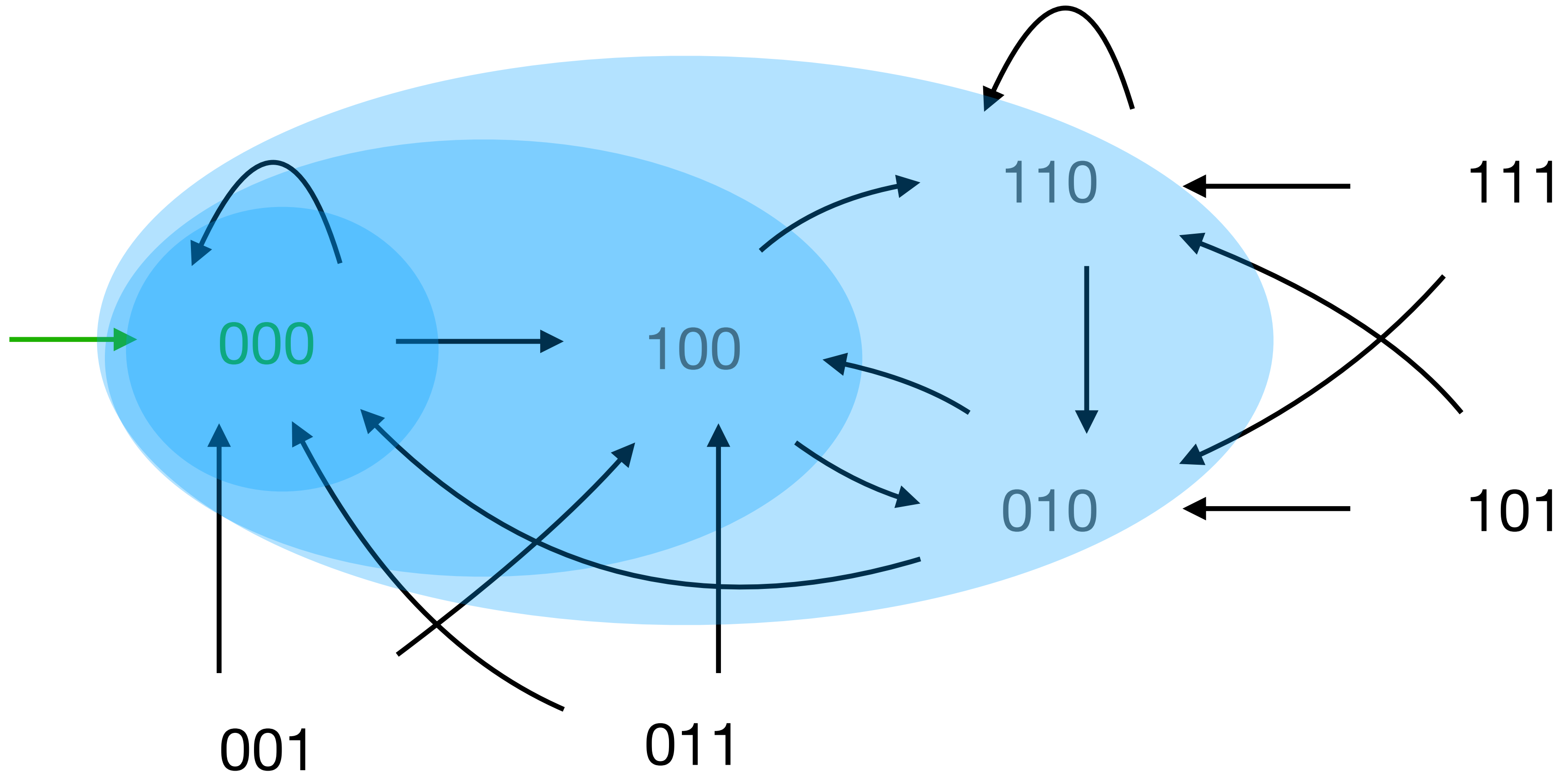


Automated Safety Verification

Transition System:



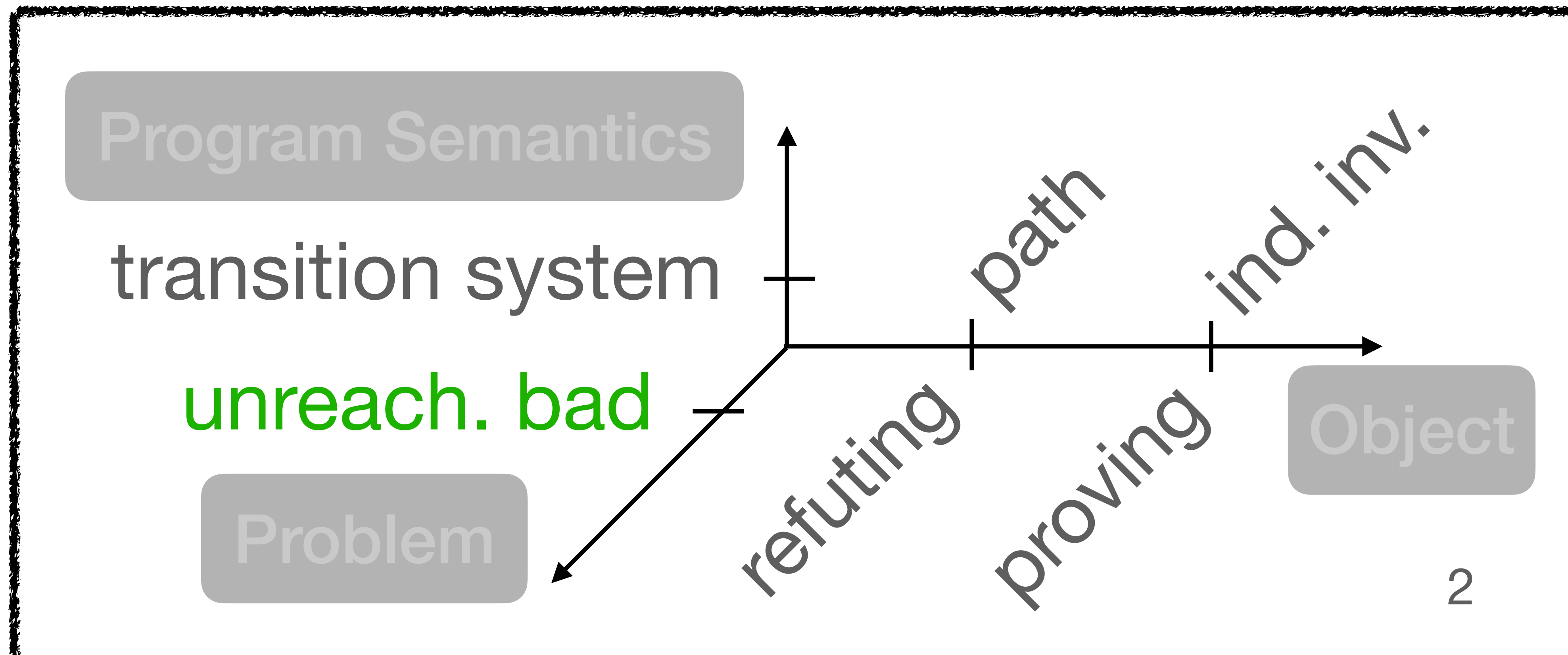
=



= (States, Init, post)

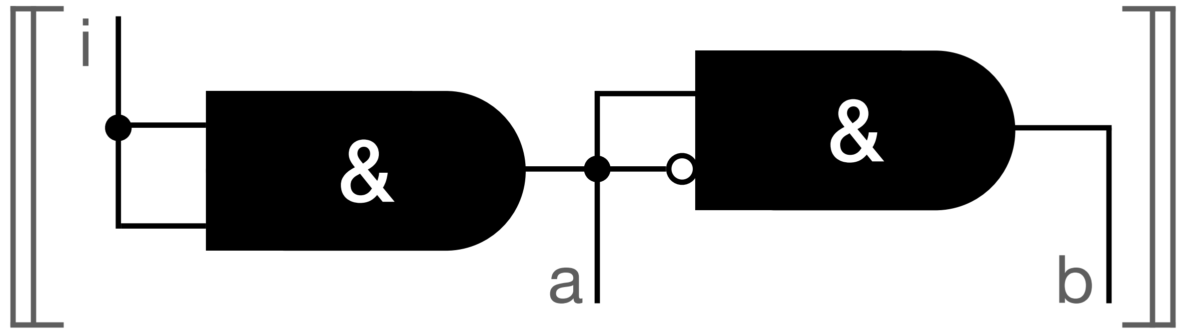
Reachable States:

$$\mathrm{post}^*(\mathrm{Init}) = \bigcup_{i\in\mathbb{N}} \mathrm{post}^i(\mathrm{Init}) \subseteq \mathrm{States}$$

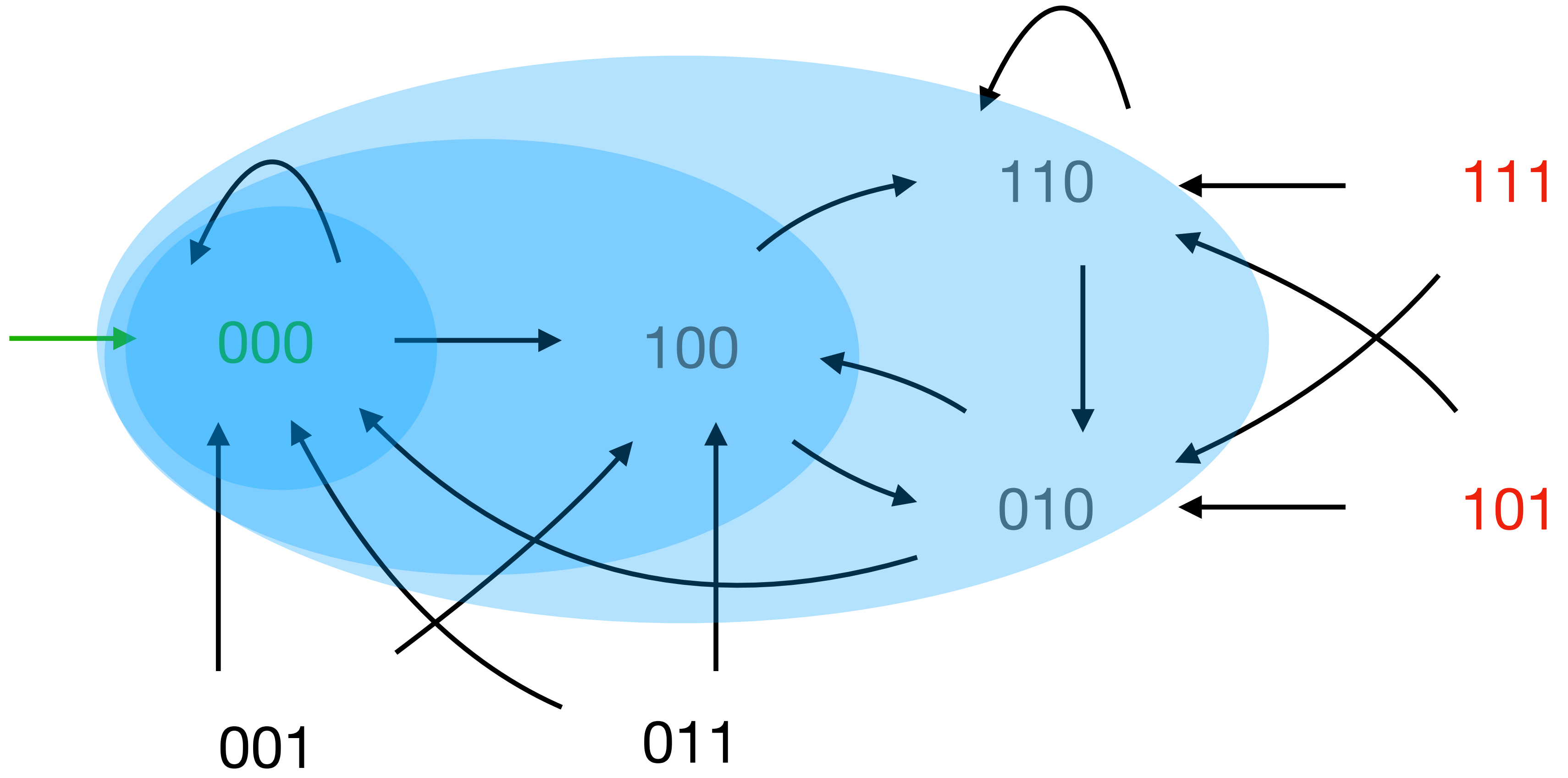


Automated Safety Verification

Transition System:



=



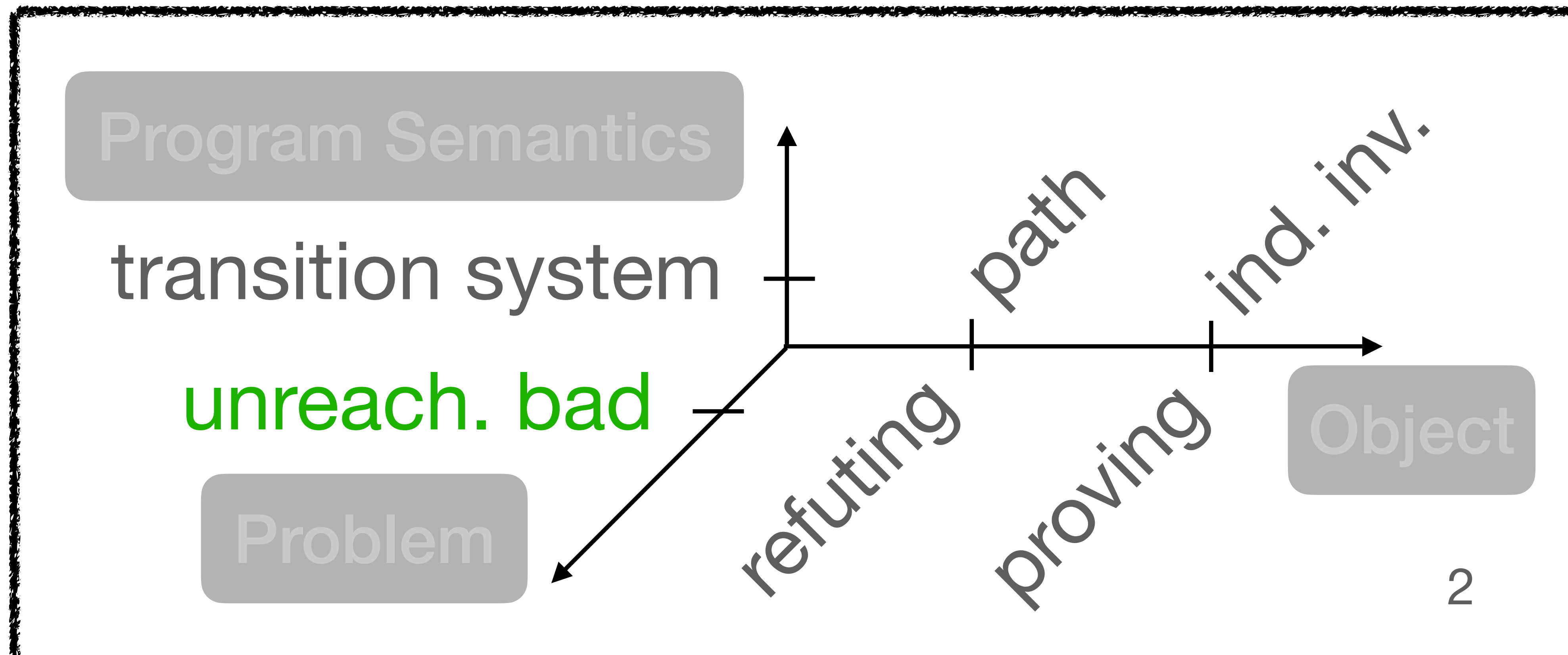
= (States, Init, post)

Reachable States:

$$\mathrm{post}^*(\mathrm{Init}) = \bigcup_{i \in \mathbb{N}} \mathrm{post}^i(\mathrm{Init}) \subseteq \mathrm{States}$$

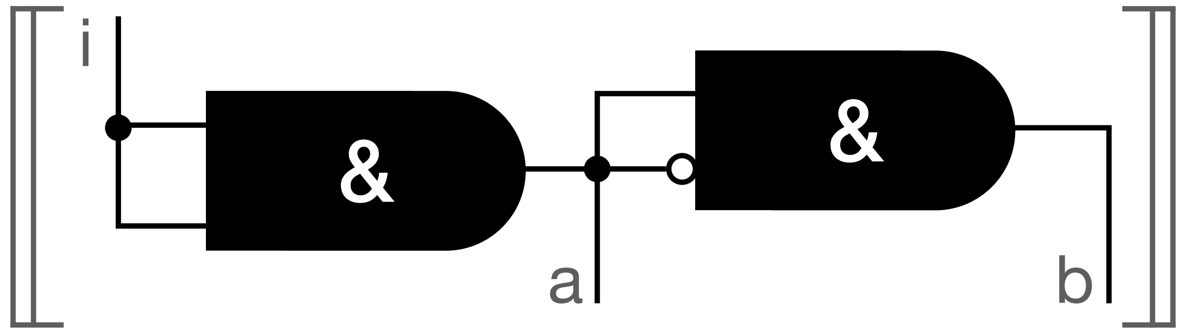
Problem:

Given: TS and Bad $\subseteq$ States.

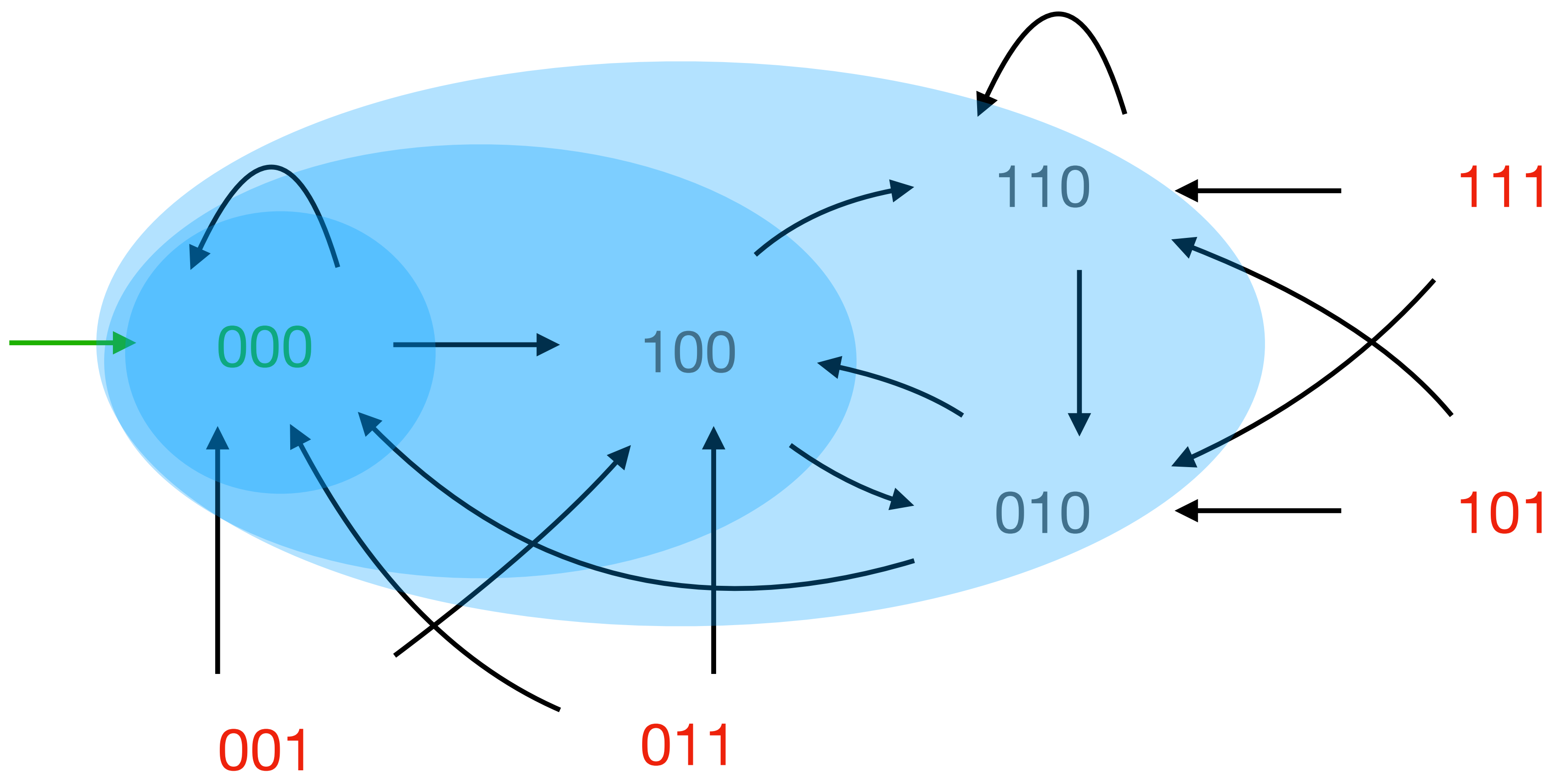


Automated Safety Verification

Transition System:



=



= (States, Init, post)

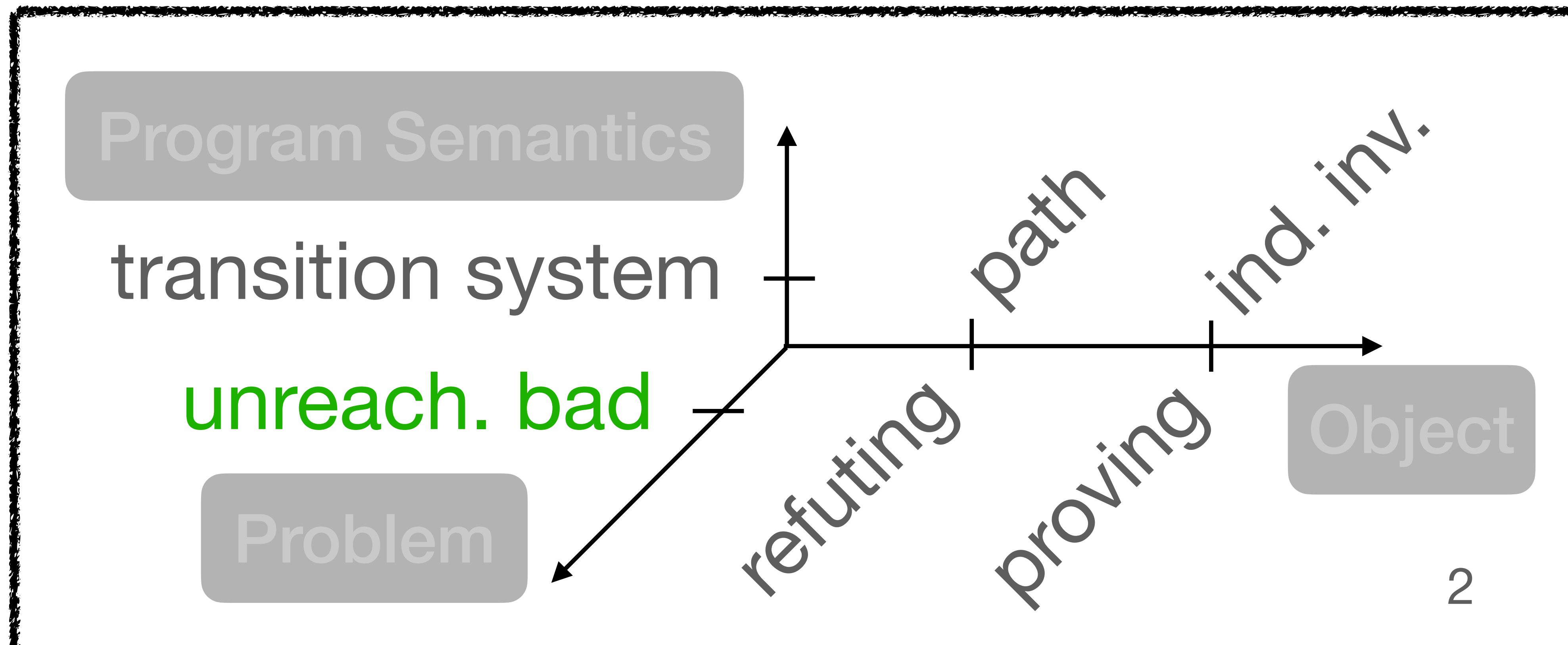
Reachable States:

$$\text{post}^*(\text{Init}) = \bigcup_{i\in\mathbb{N}} \text{post}^i(\text{Init}) \subseteq \text{States}$$

Problem:

Given: TS and Bad $\subseteq$ States.

Question: $\text{post}^*(\text{Init}) \cap \text{Bad} = \varnothing$?

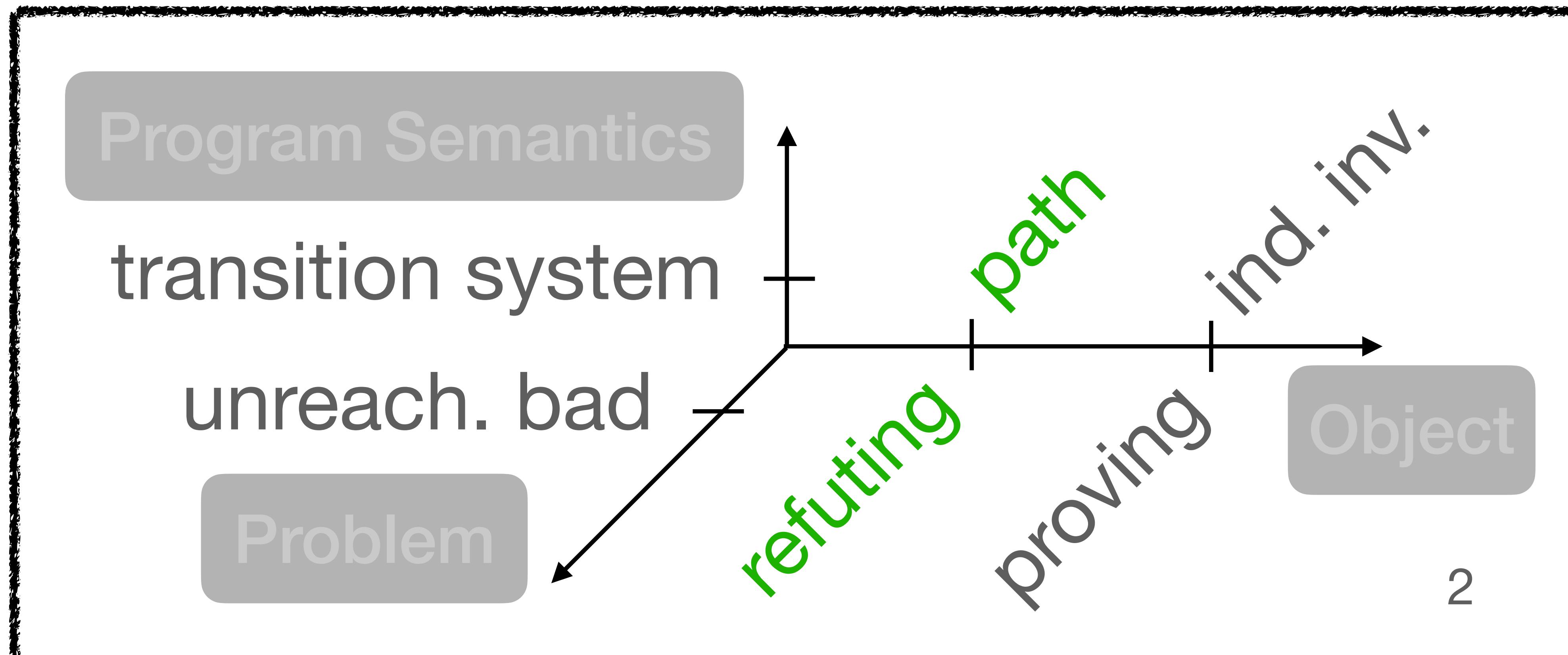


Automated Safety Verification

Path $p \in \text{States}^+$ with

$p_0 \in$ Init
$p_{i+1} \in \text{post}(p_i)$ for all $0 \leq i < |p|$.

Reaches Bad, if $\text{last}(p) \in$ Bad.

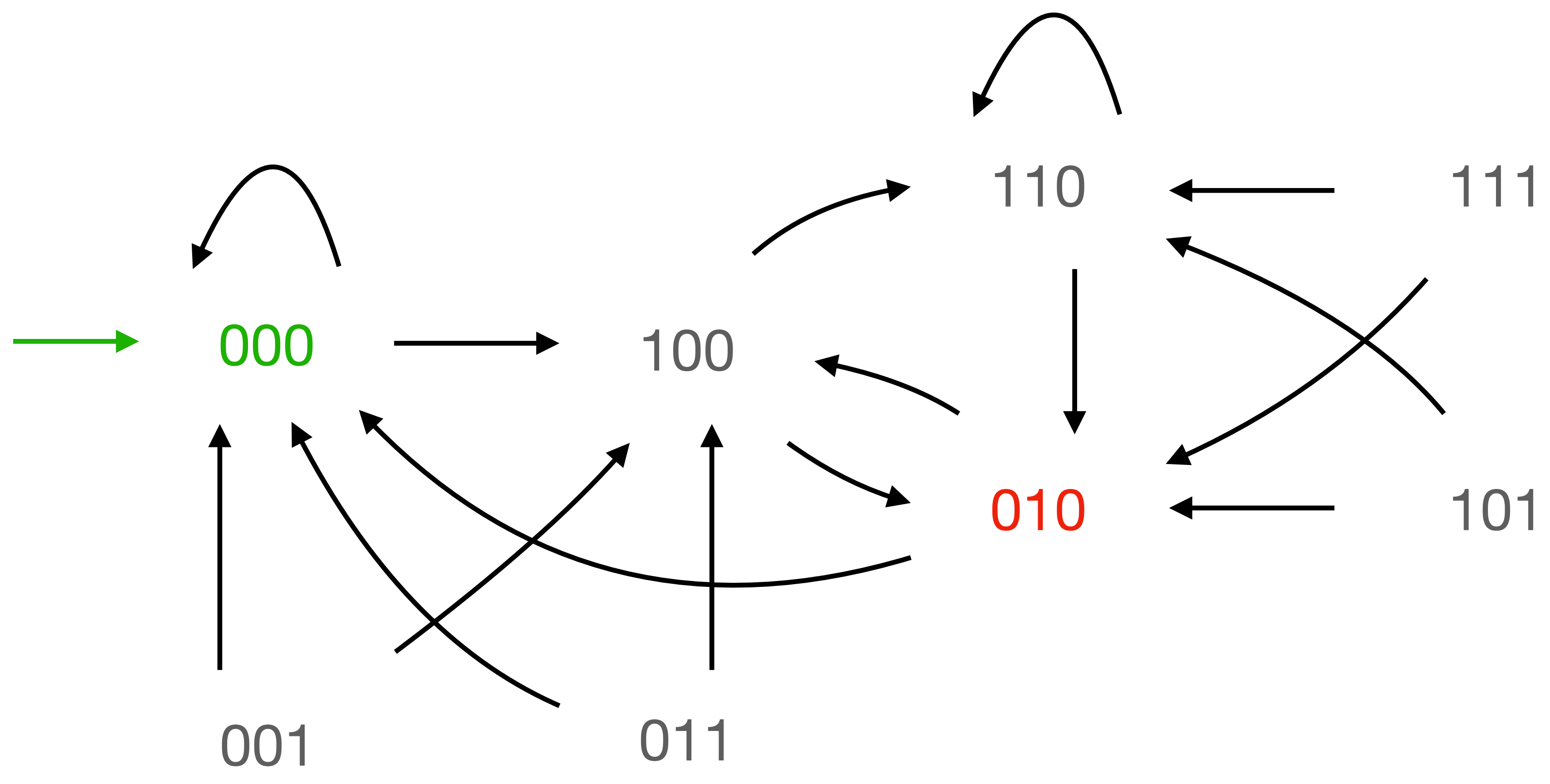


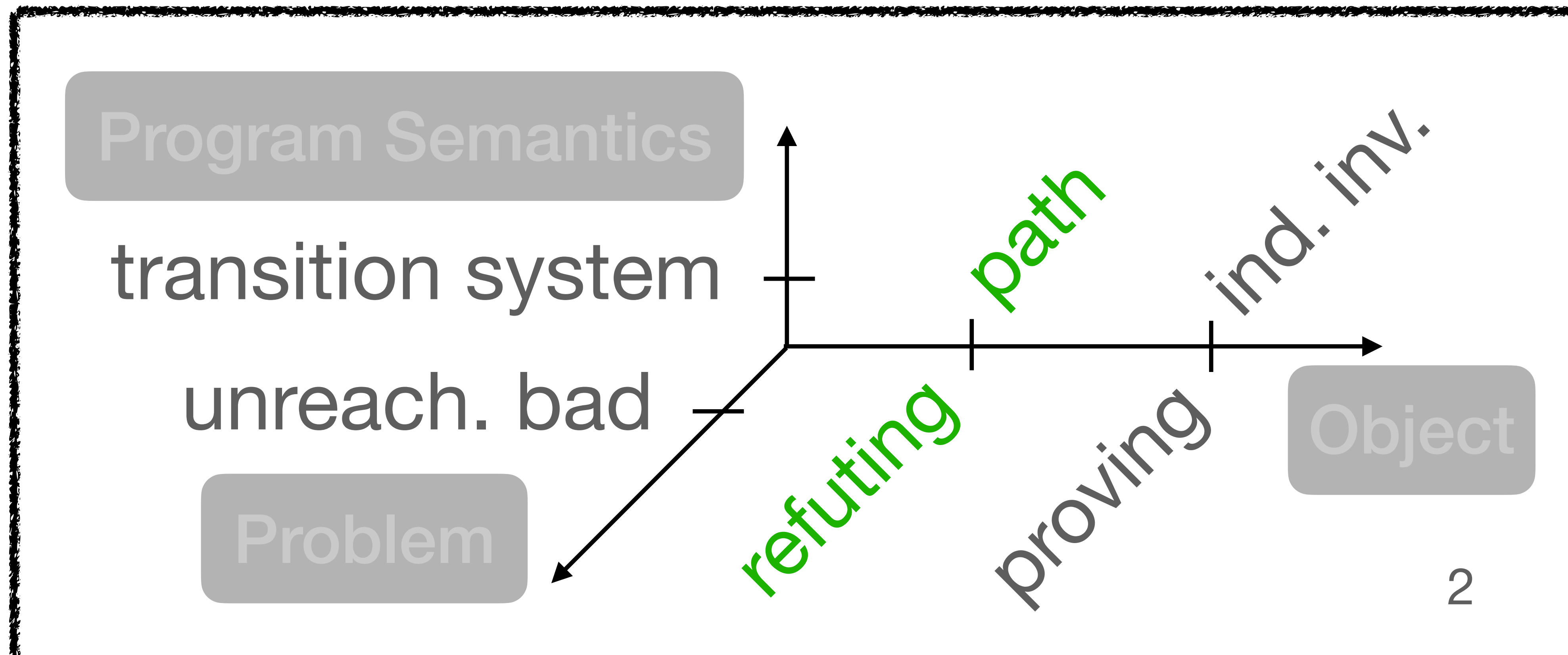
Automated Safety Verification

Path $p \in \text{States}^+$ with

$$p_0 \in \text{Init}$$
$$p_{i+1} \in \text{post}(p_i) \text{ for all } 0 \le i < |p|\ .$$

Reaches Bad, if $\text{last}(p) \in \text{Bad}$.



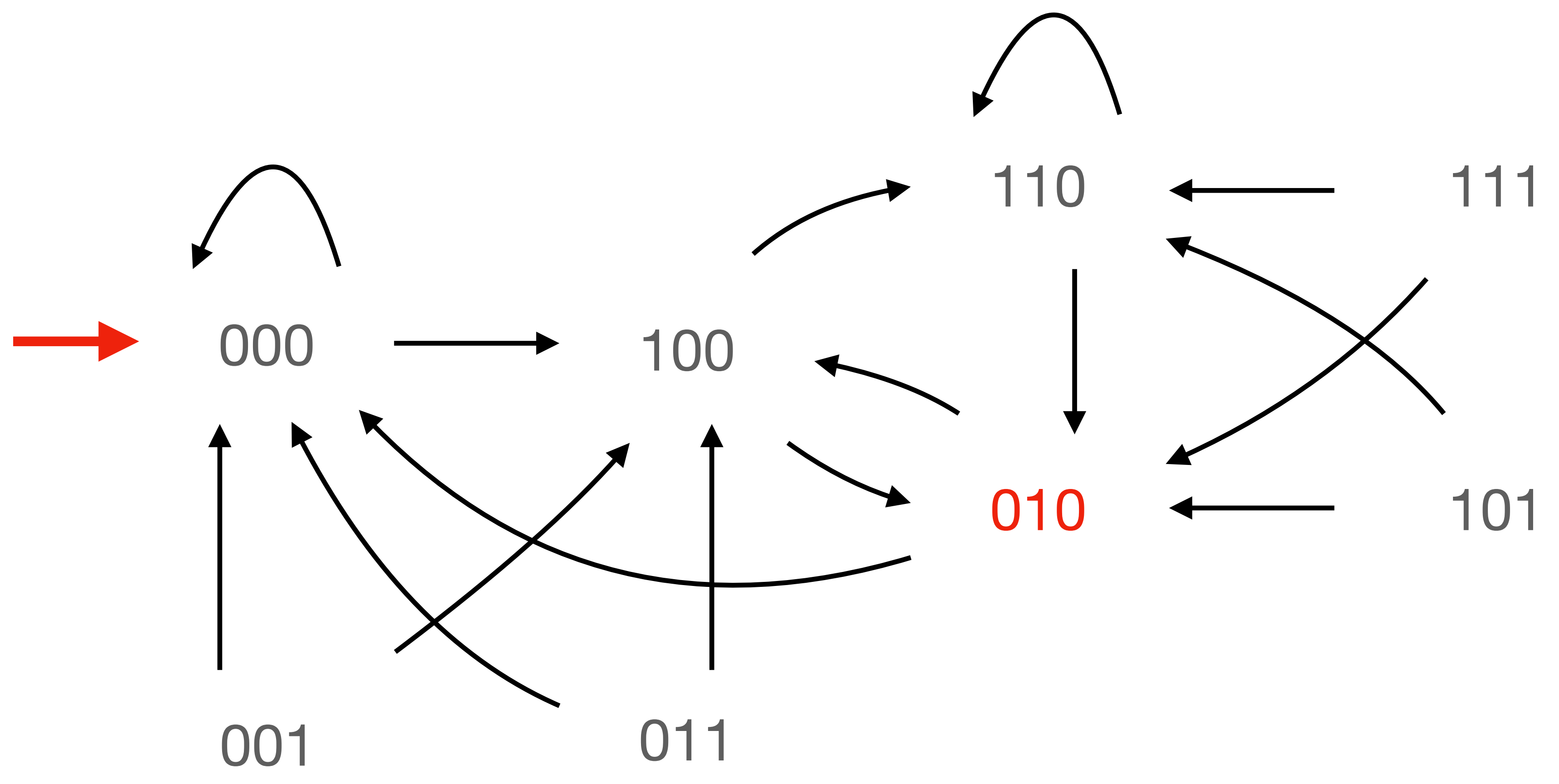


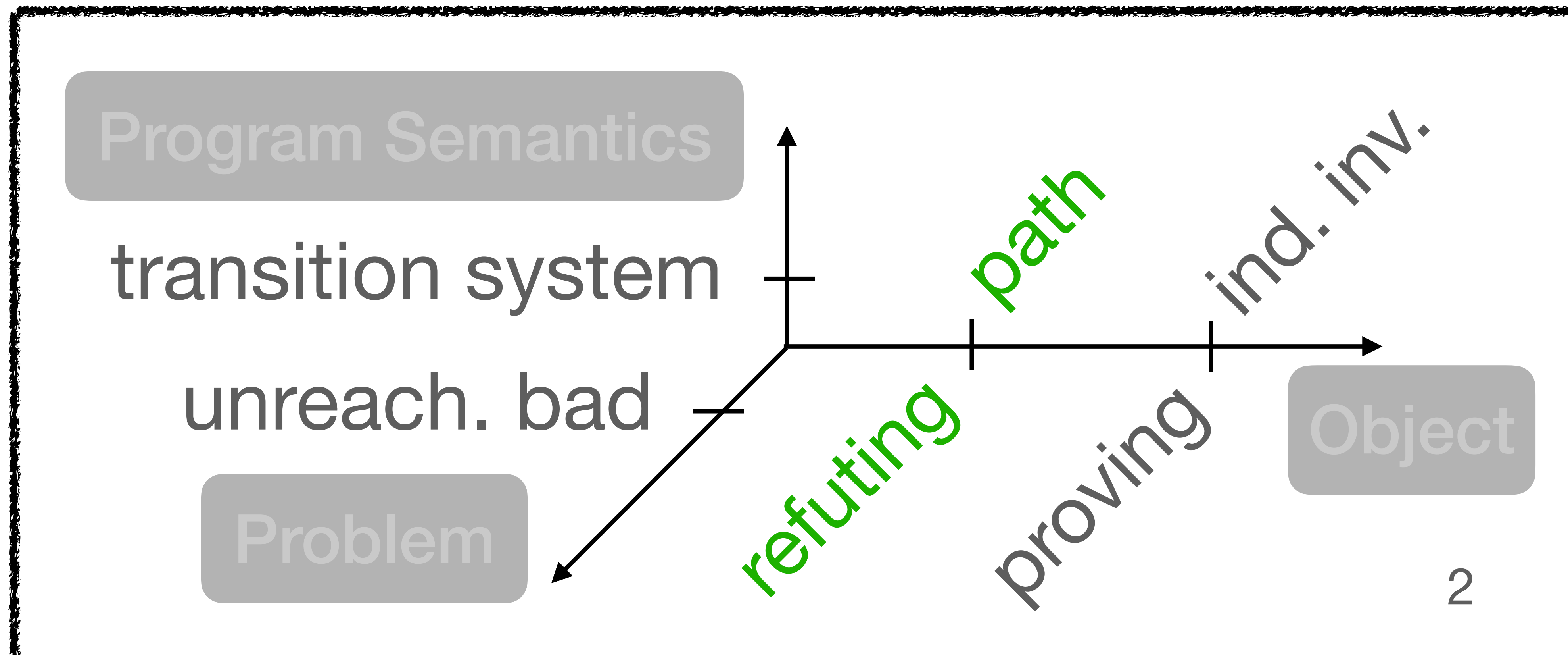
Automated Safety Verification

Path p $\in$ States$^+$ with

$p_0 \in$ Init
$p_{i+1} \in$ post(p_i) for all $0 \leq i < |p|$.

Reaches Bad, if last(p) $\in$ Bad.





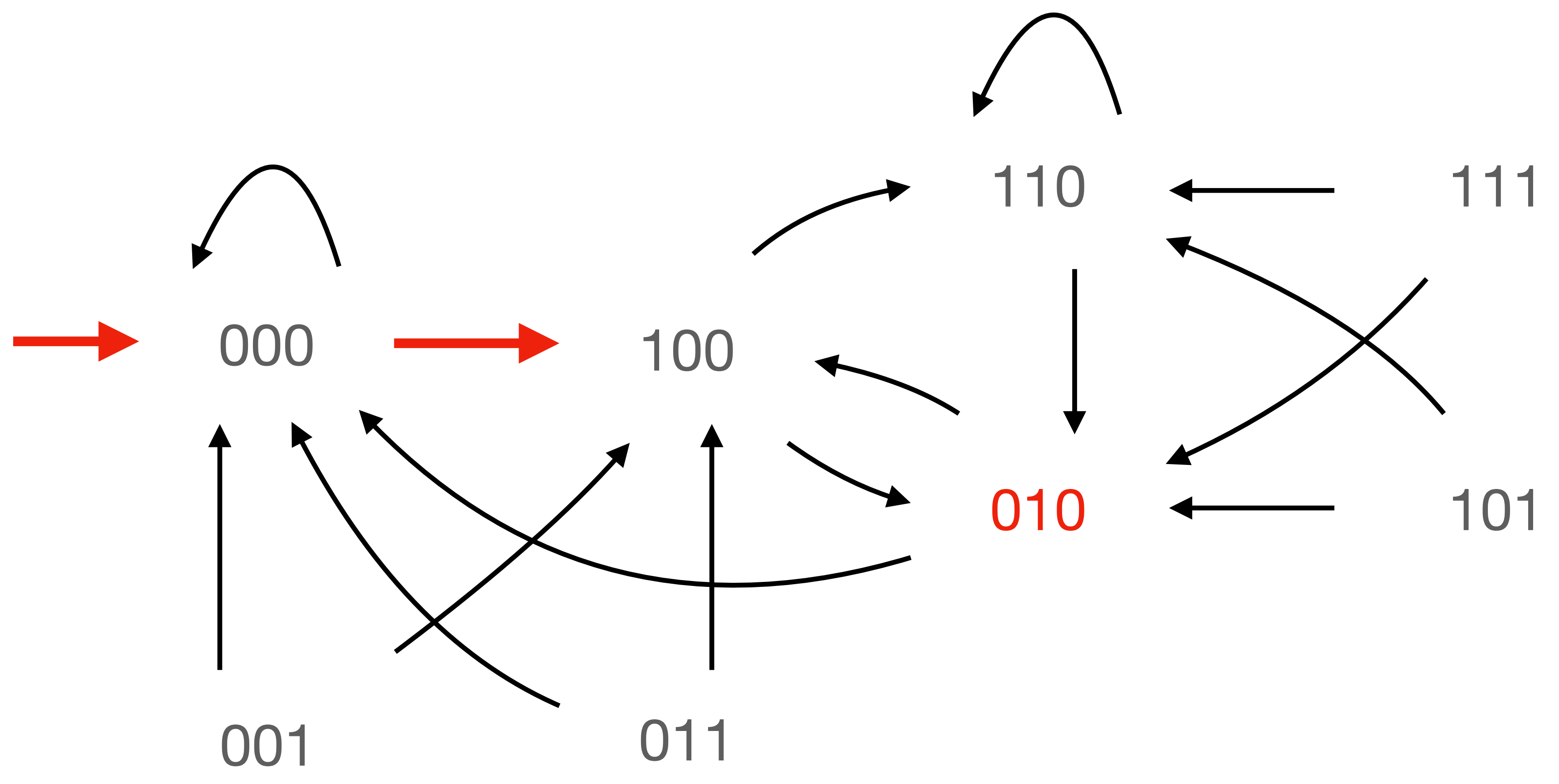
Automated Safety Verification

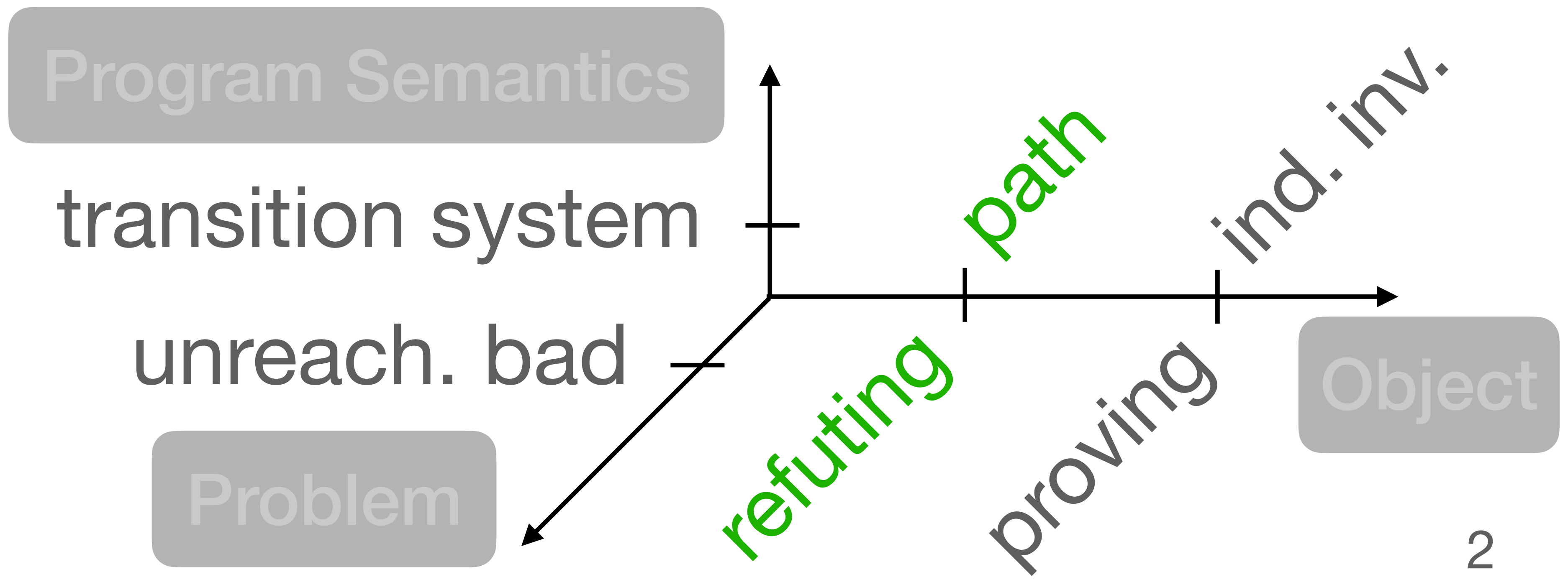
Path $p \in \text{States}^+$ with

$p_0 \in \text{Init}$

$p_{i+1} \in \text{post}(p_i)$ for all $0 \leq i < |p|$.

Reaches Bad, if $\text{last}(p) \in \text{Bad}$.



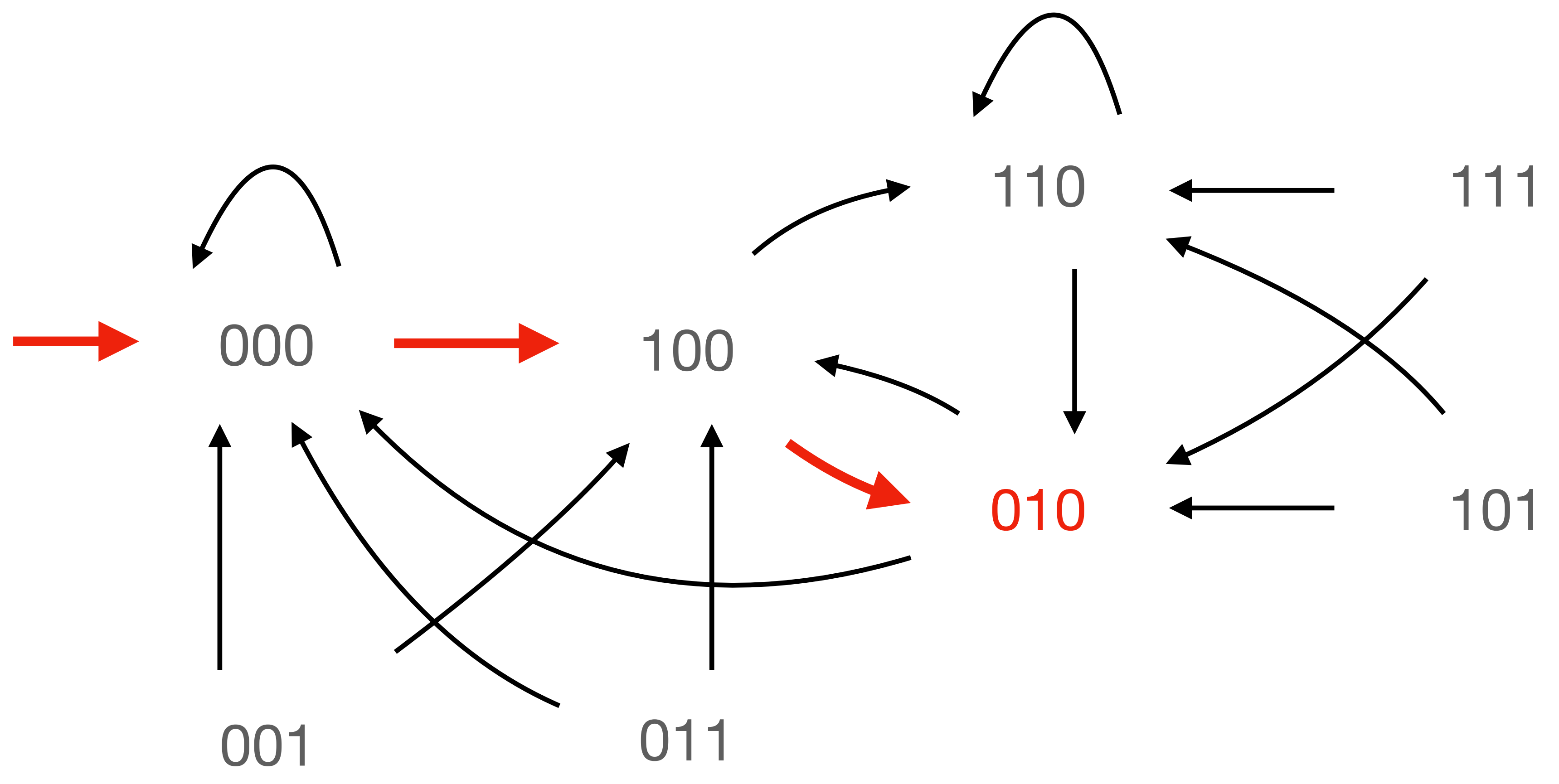


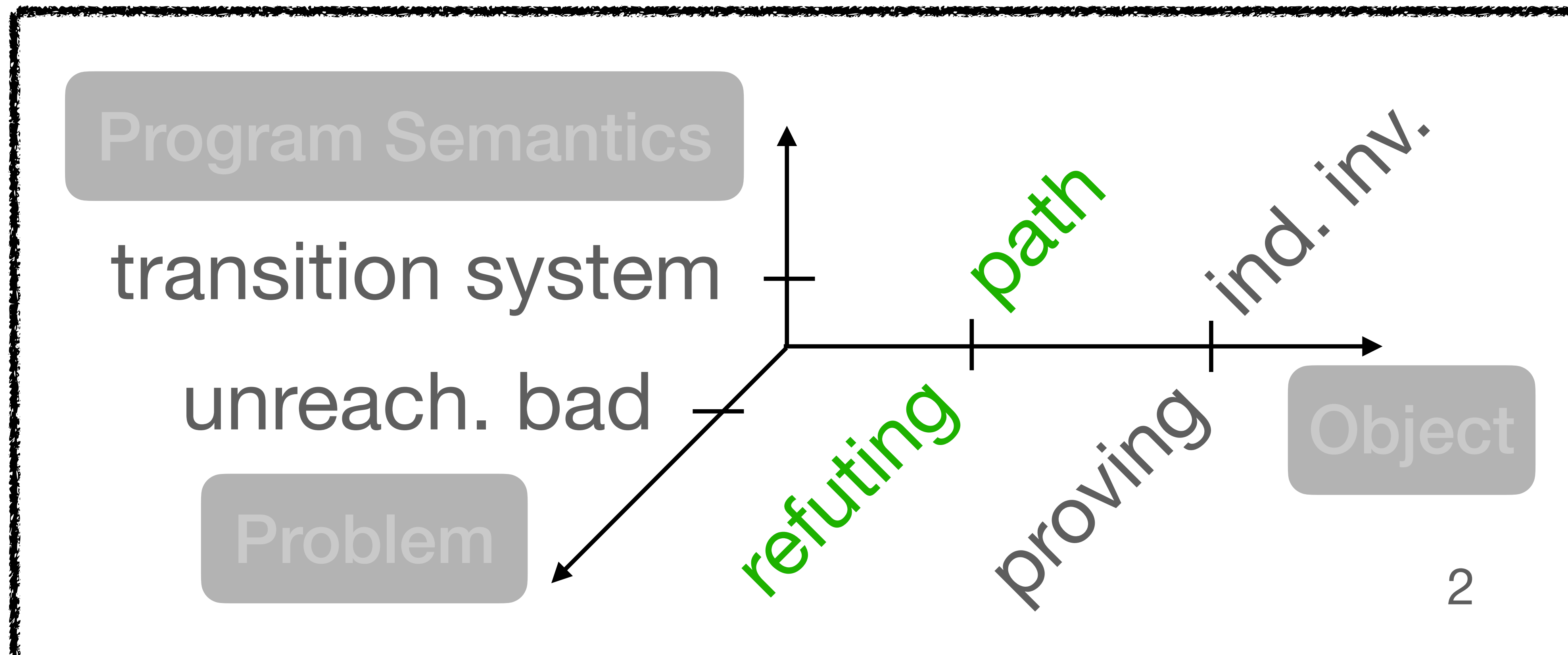
Automated Safety Verification

Path $p \in \text{States}^+$ with

$p_0 \in$ Init
$p_{i+1} \in \text{post}(p_i)$ for all $0 \leq i < |p|$.

Reaches Bad, if $\text{last}(p) \in$ Bad.



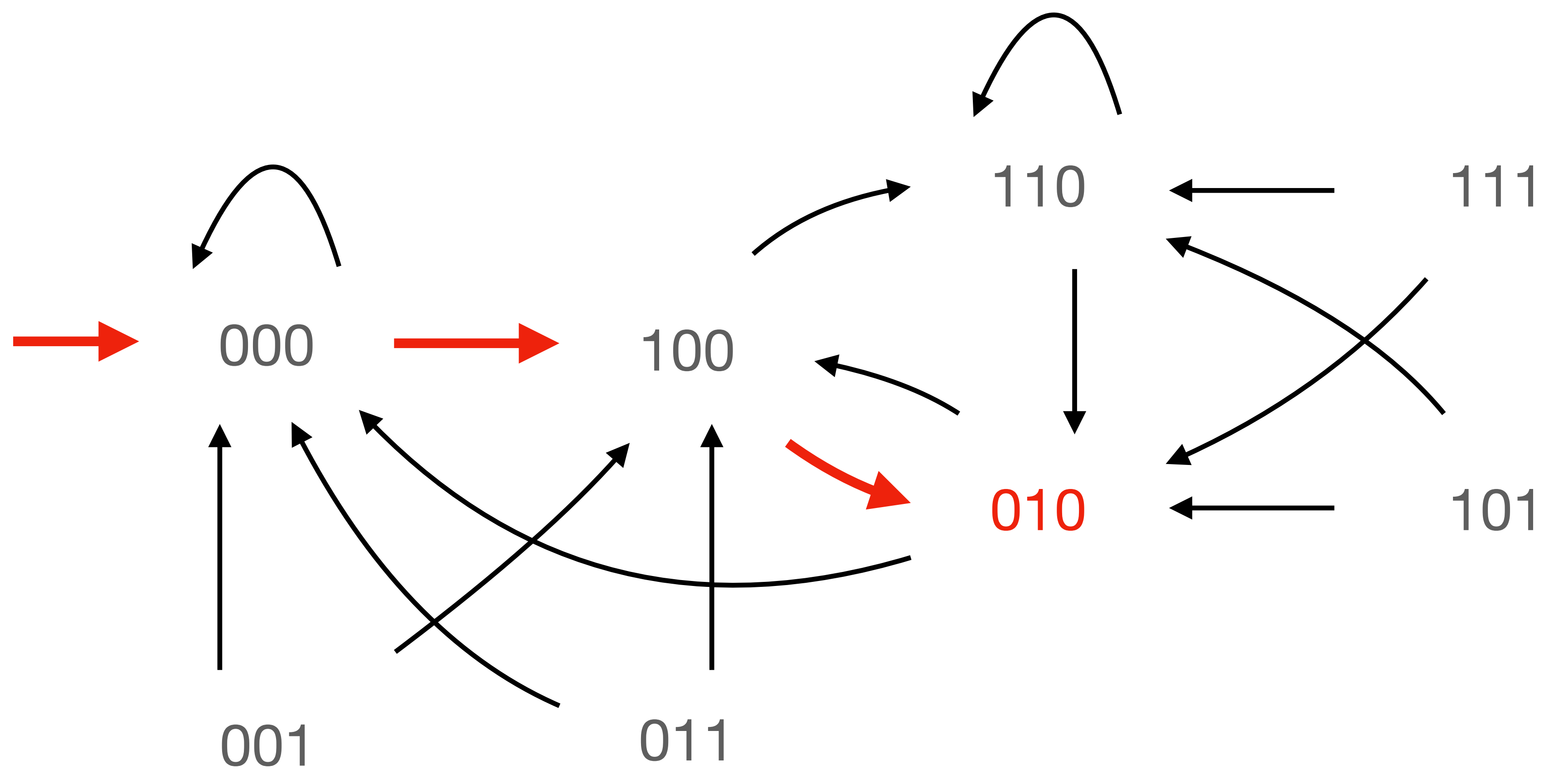


Automated Safety Verification

Path p $\in$ States$^+$ with

$p_0 \in$ Init
$p_{i+1} \in$ post(p_i) for all $0 \leq i < |p|$.

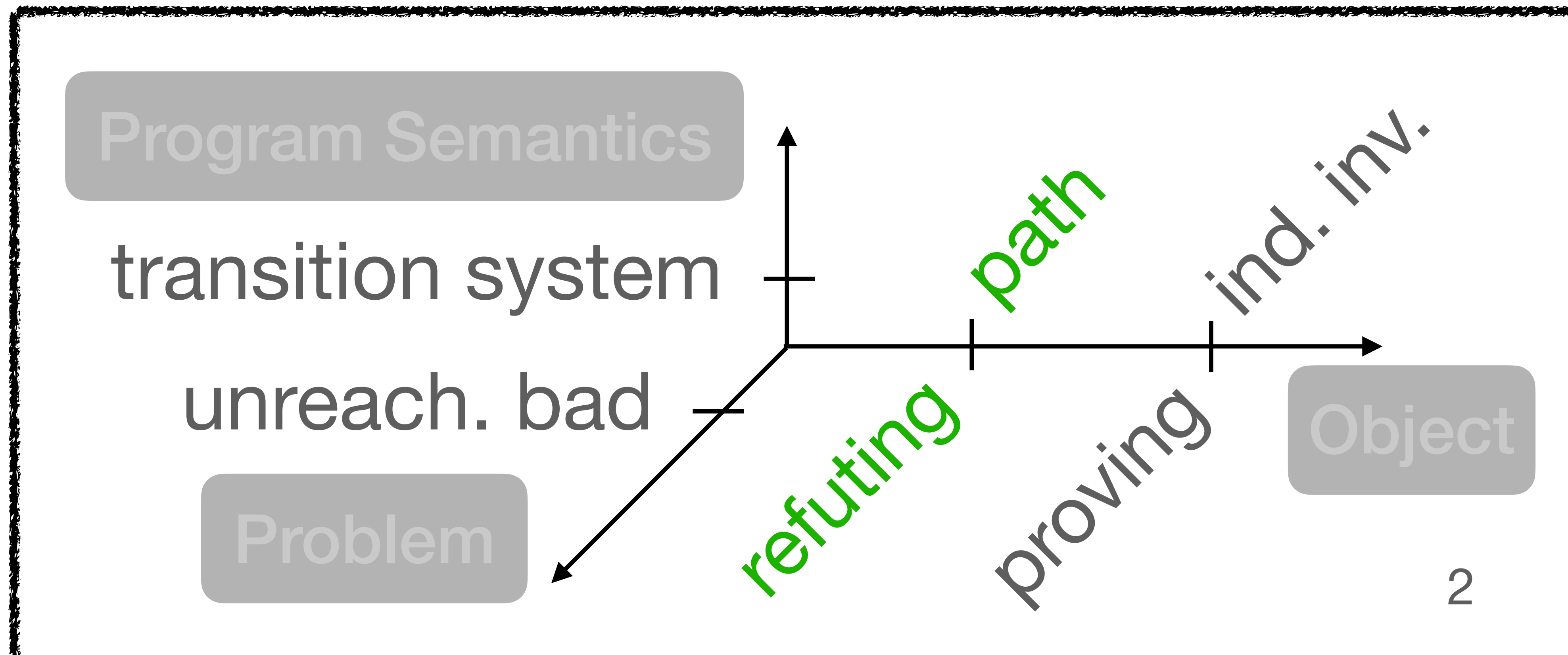
Reaches Bad, if last(p) $\in$ Bad.



Lemma (S+C for Finding Bugs):

post*(Init) $\cap$ Bad $\neq \varnothing$ if and only if

$\exists$ path that reaches Bad.

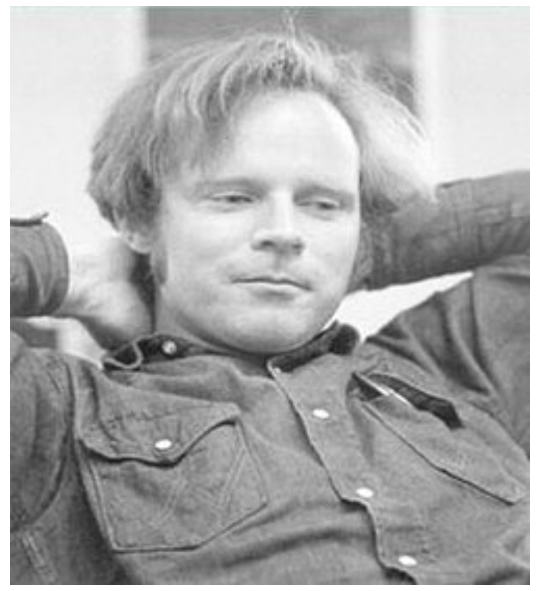


Automated Safety Verification: Invariants

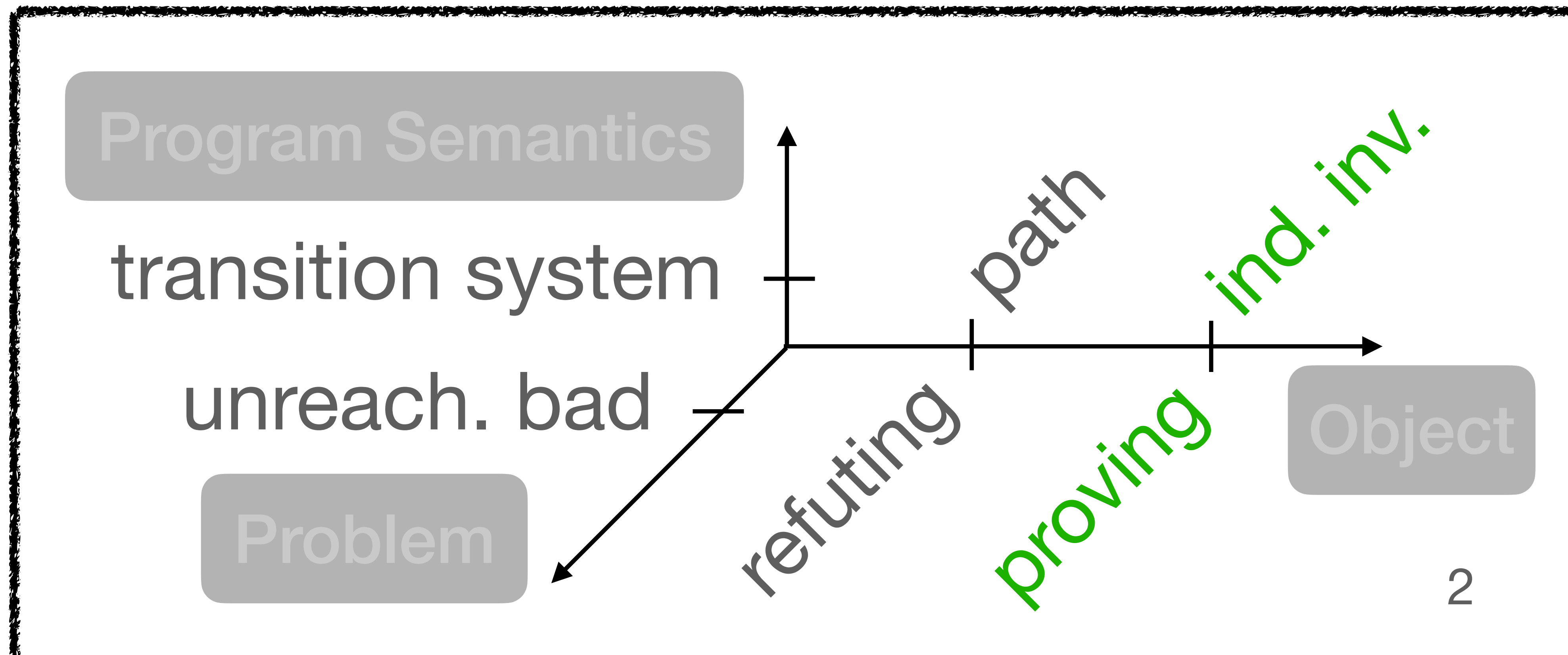
Inductive Invariant I $\subseteq$ States with

$$\text{Init} \subseteq I$$
$$\text{post}(I) \subseteq I\ .$$

Save wrt. Bad, if $I \cap \text{Bad} = \emptyset$.



1967

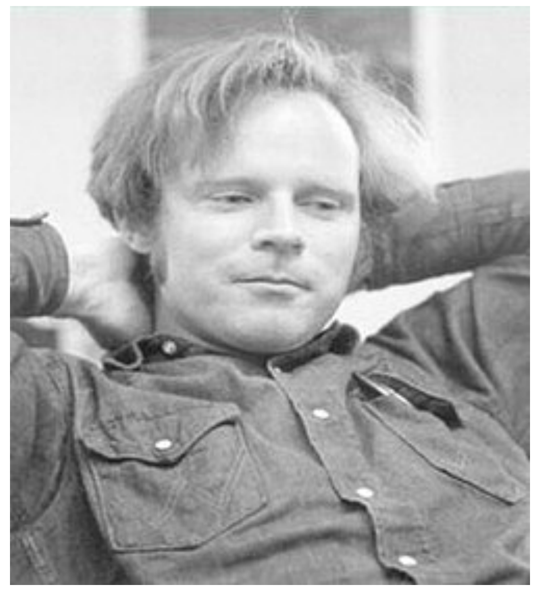


Automated Safety Verification: Invariants

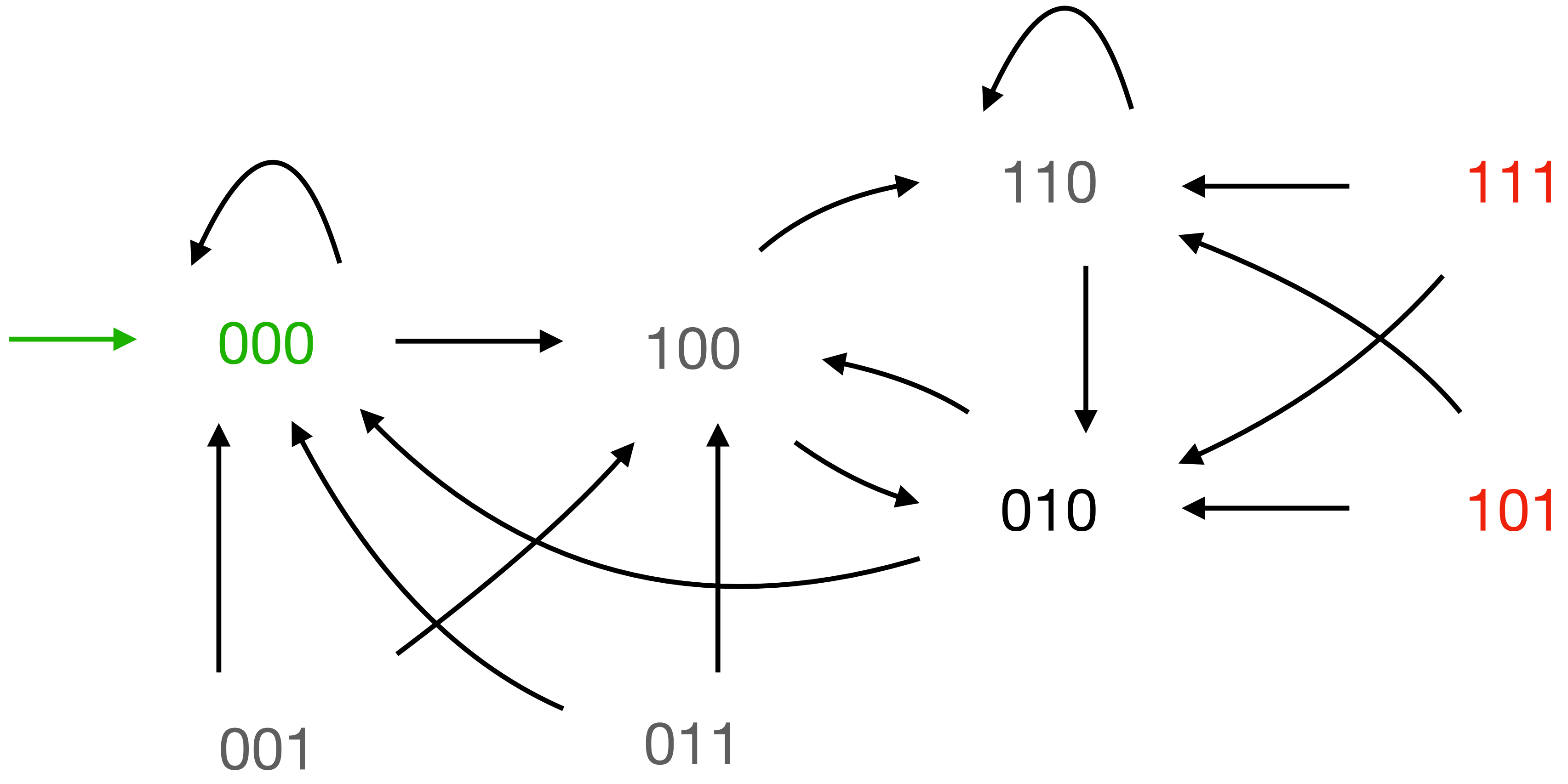
Inductive Invariant I $\subseteq$ States with

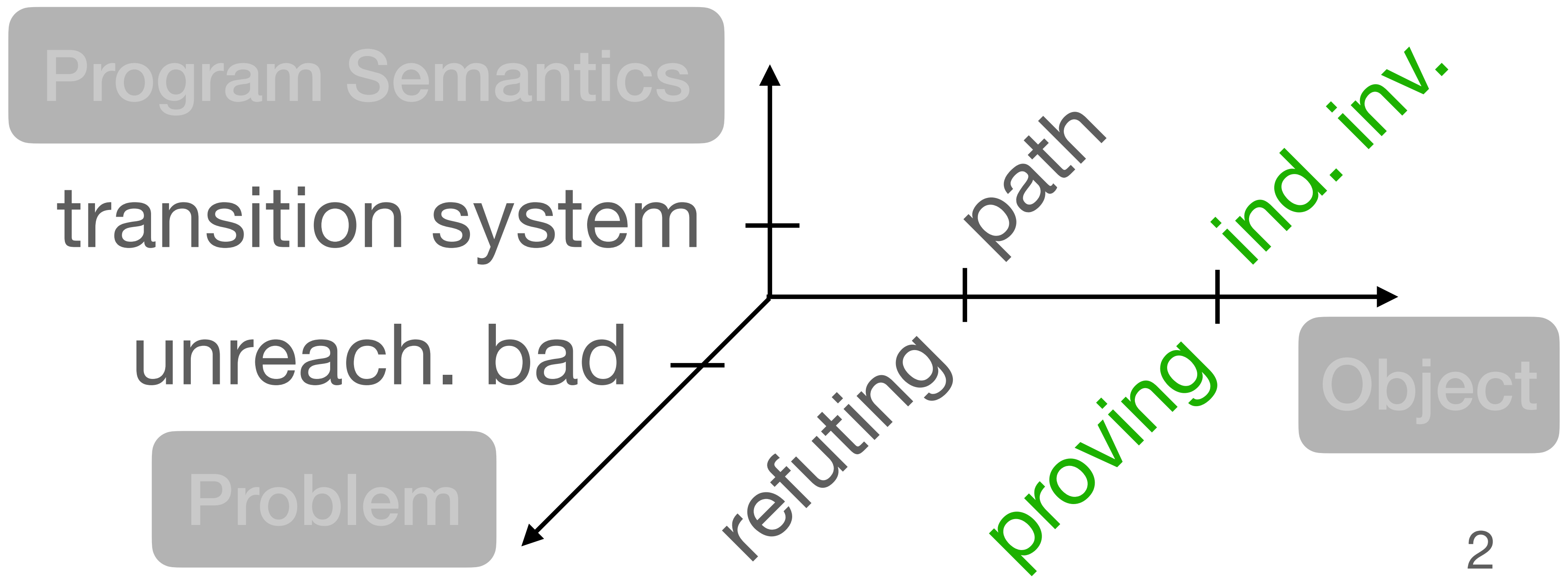
$$\text{Init} \subseteq \text{I}$$
$$\text{post(I)} \subseteq \text{I} \,.$$

Save wrt. Bad, if I $\cap$ Bad $= \emptyset$.



1967





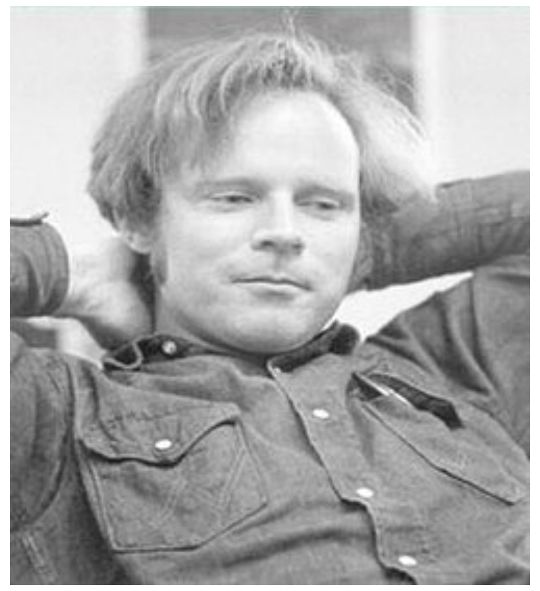
Automated Safety Verification: Invariants

Inductive Invariant I $\subseteq$ States with

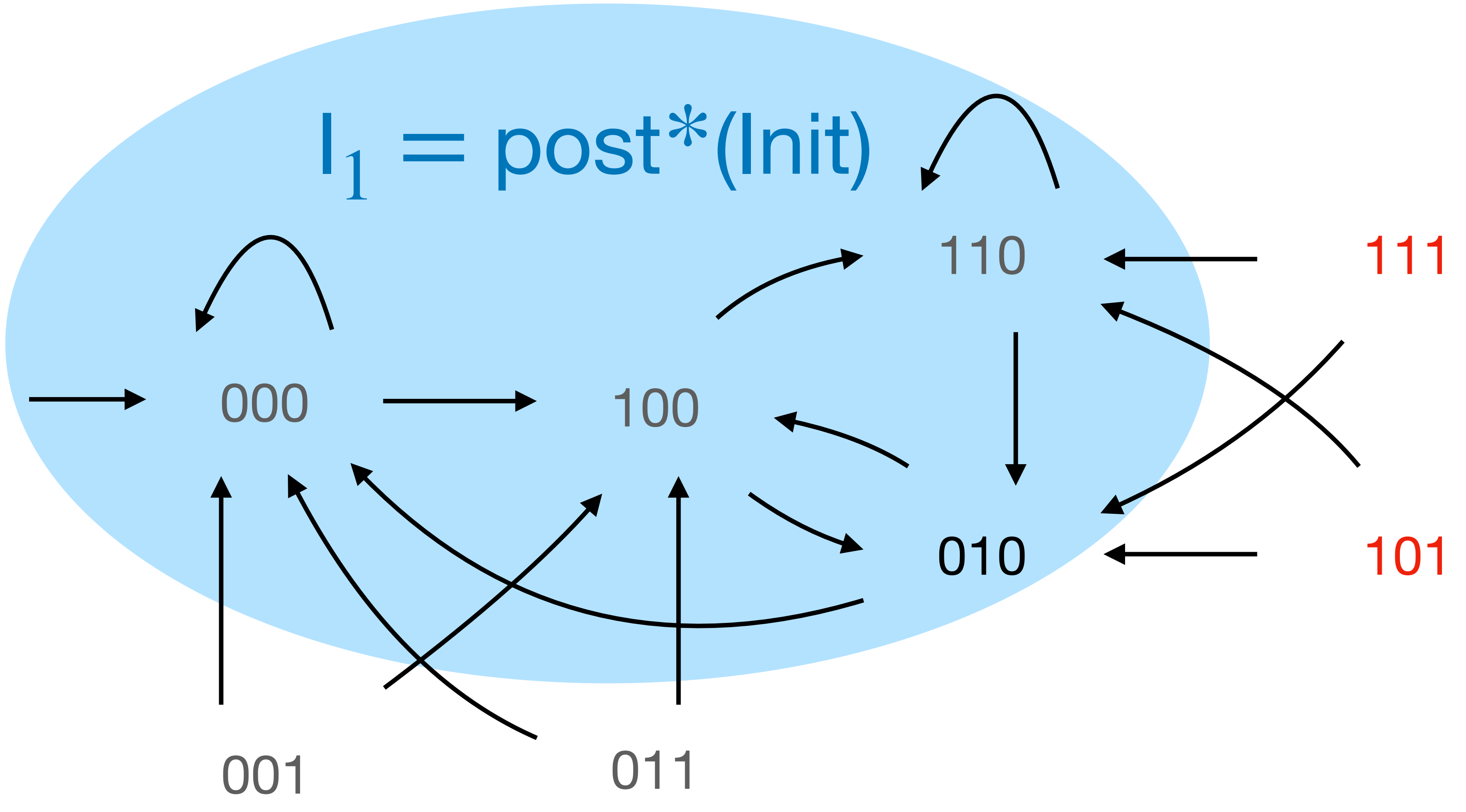
Init $\subseteq$ I

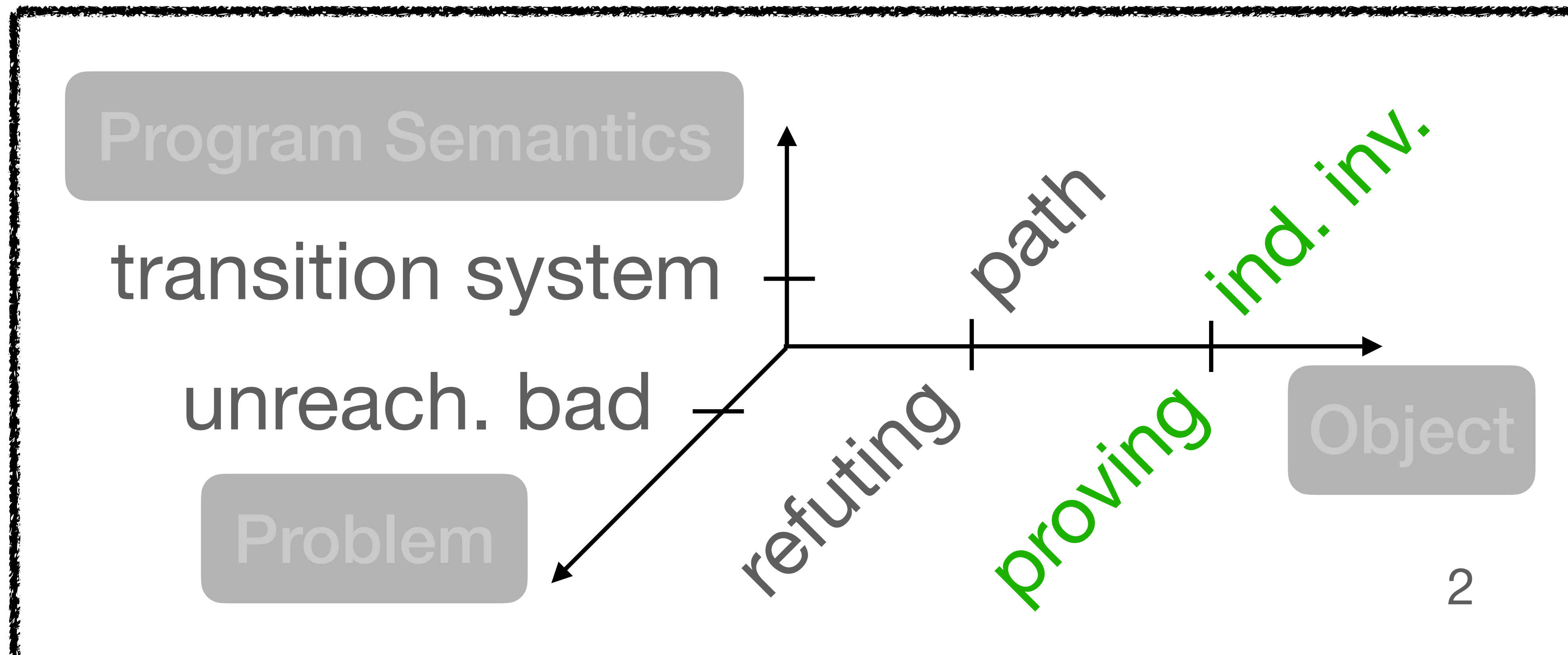
post(I) $\subseteq$ I .

Save wrt. Bad, if I $\cap$ Bad $= \emptyset$.



1967



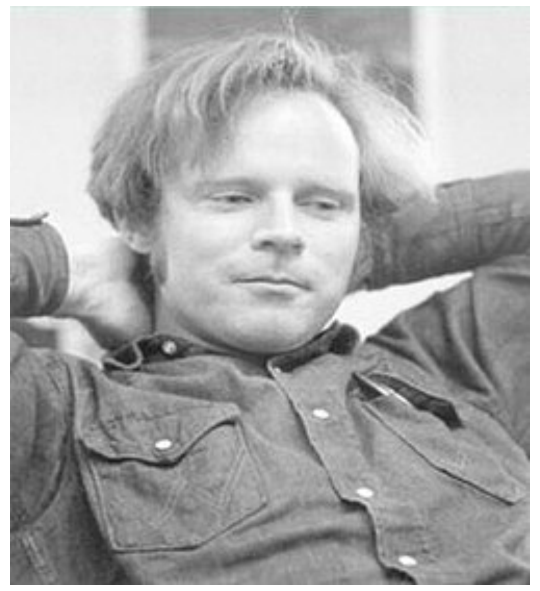


Automated Safety Verification: Invariants

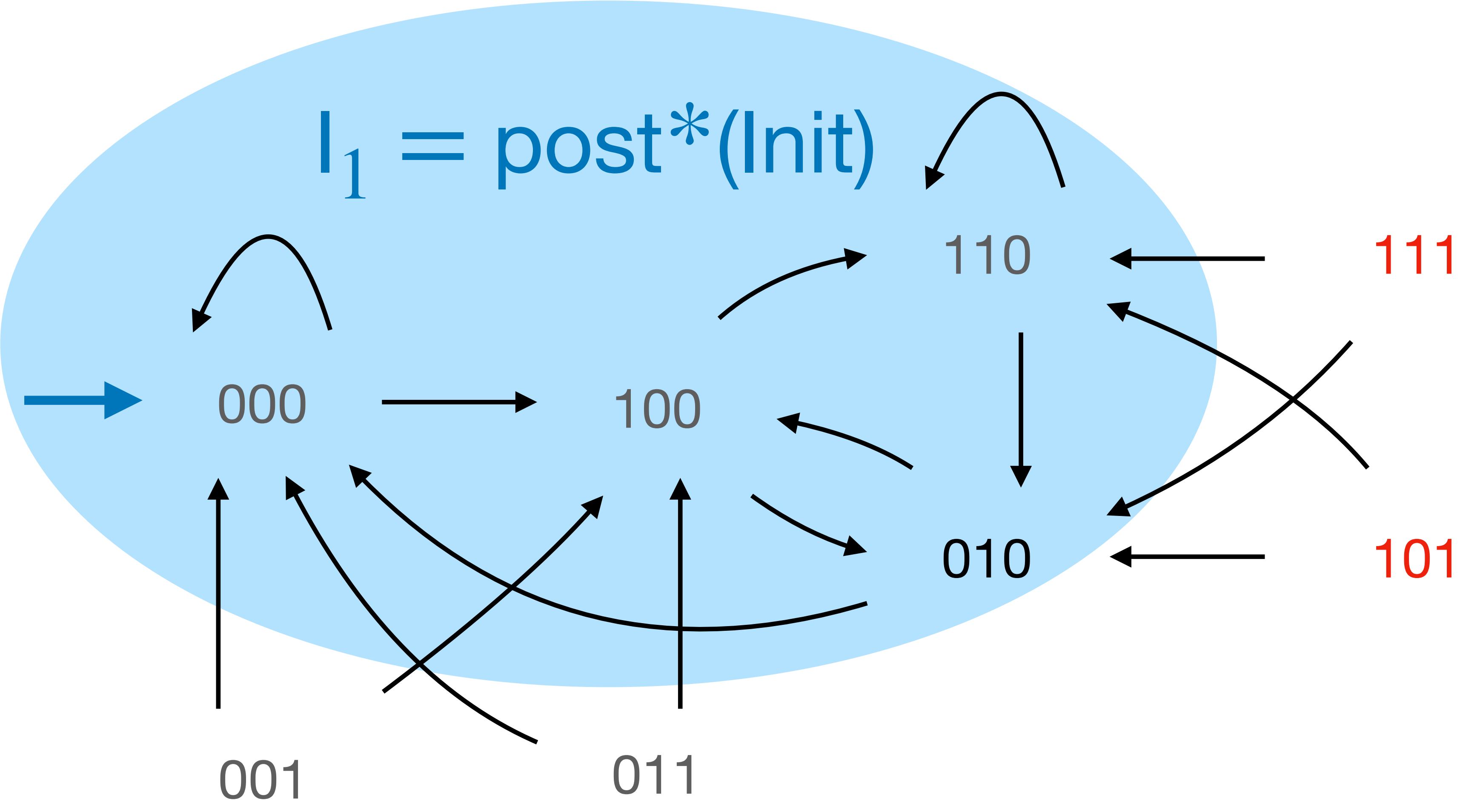
Inductive Invariant $I \subseteq$ States with

$$\mathrm{Init} \subseteq I$$
$$\mathrm{post}(I) \subseteq I\ .$$

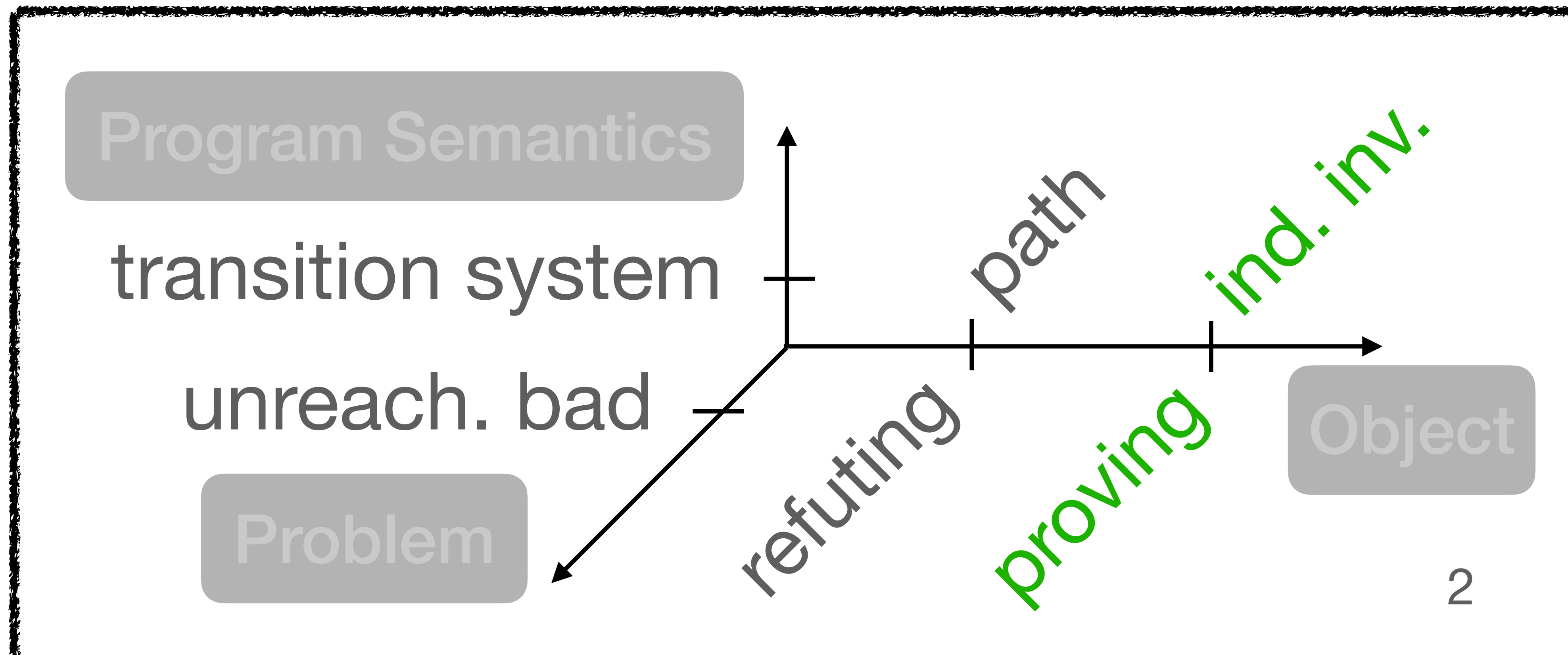
Save wrt. Bad, if $I \cap \mathrm{Bad} = \emptyset$.



1967



Inductiveness!

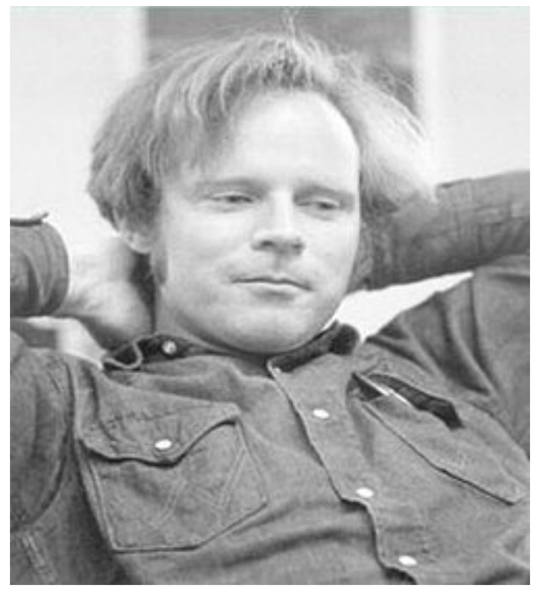


Automated Safety Verification: Invariants

Inductive Invariant I $\subseteq$ States with

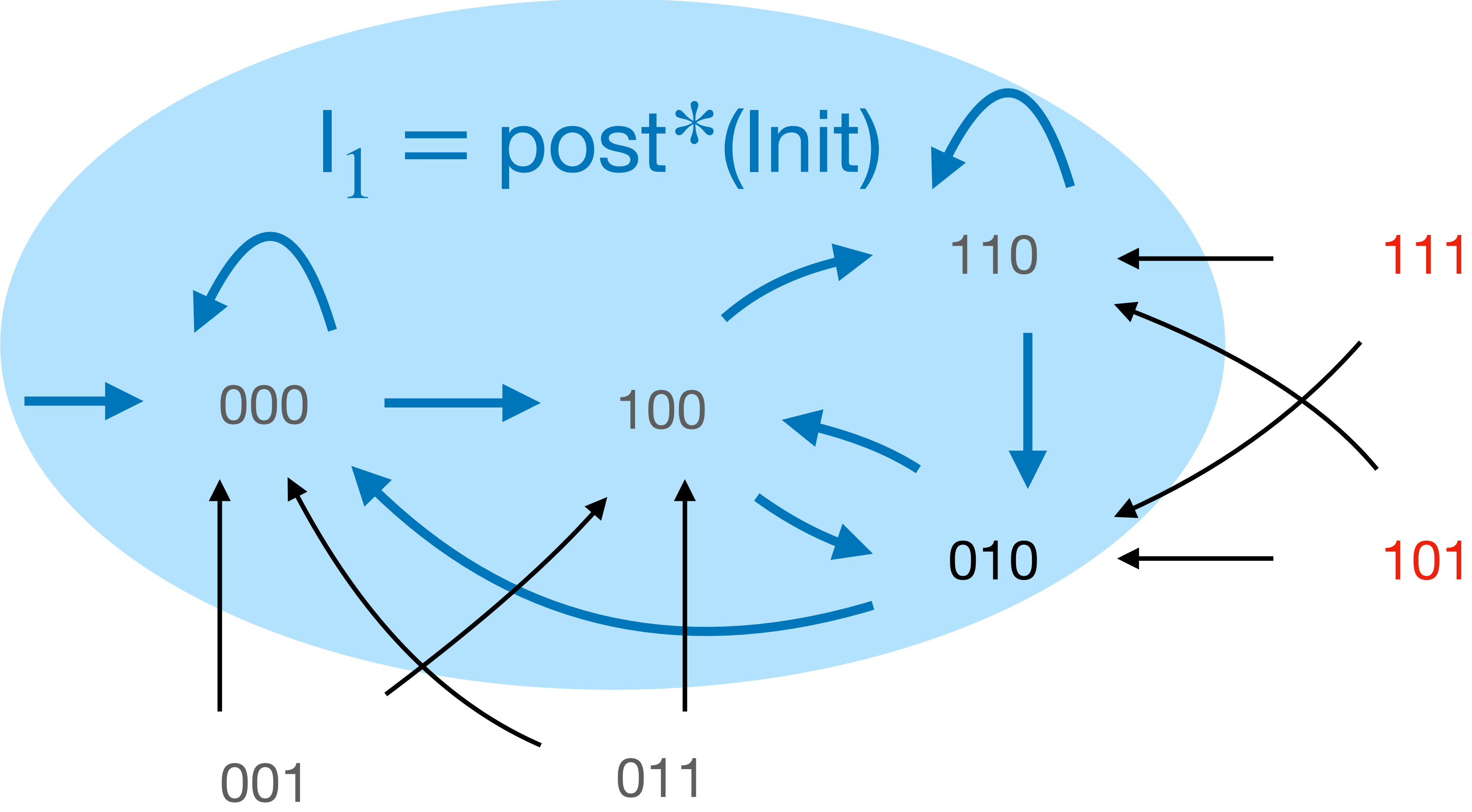
Init $\subseteq$ I
post(I) $\subseteq$ I .

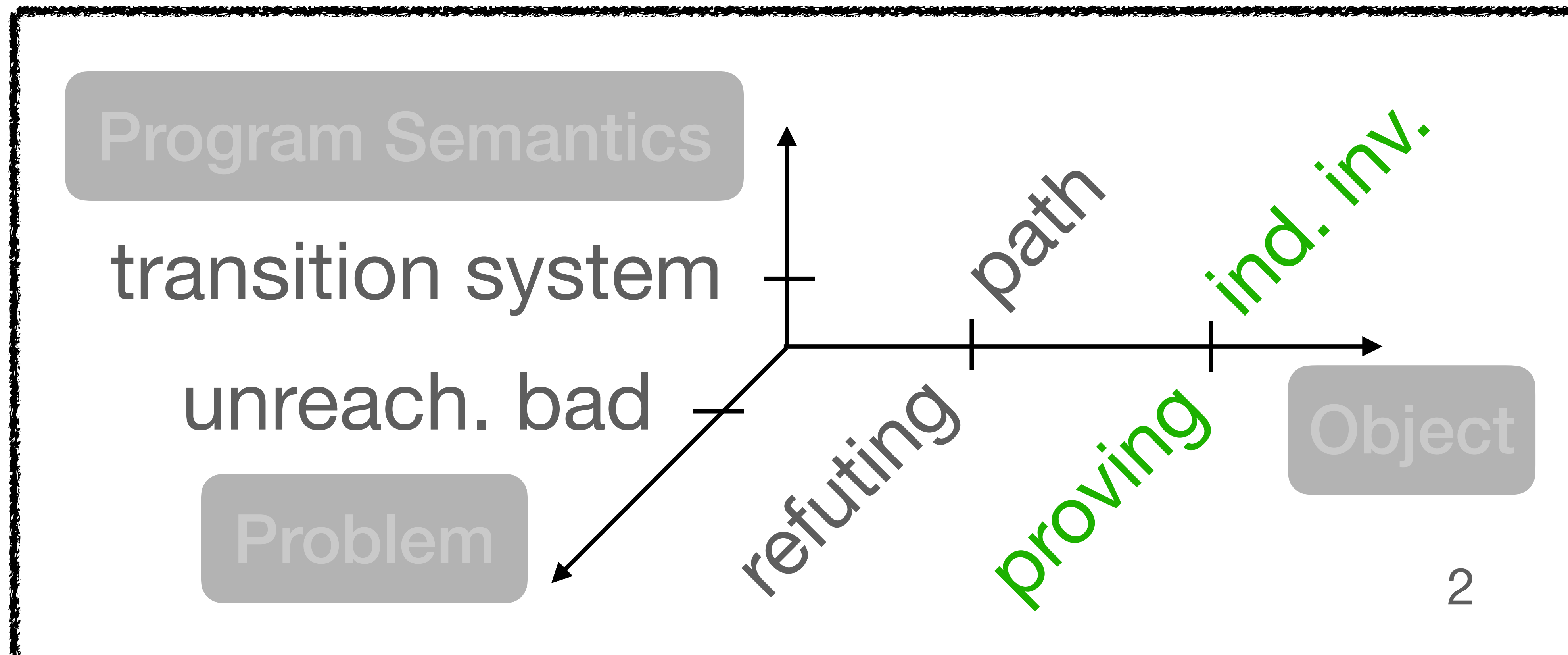
Save wrt. Bad, if I $\cap$ Bad $= \emptyset$.



1967

Inductiveness!



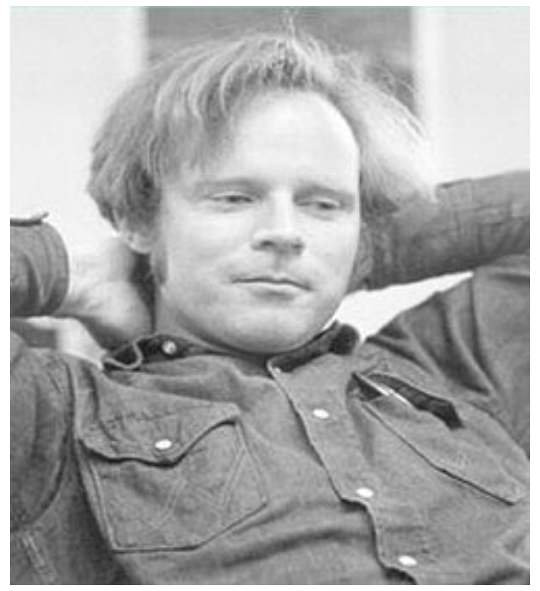


Automated Safety Verification: Invariants

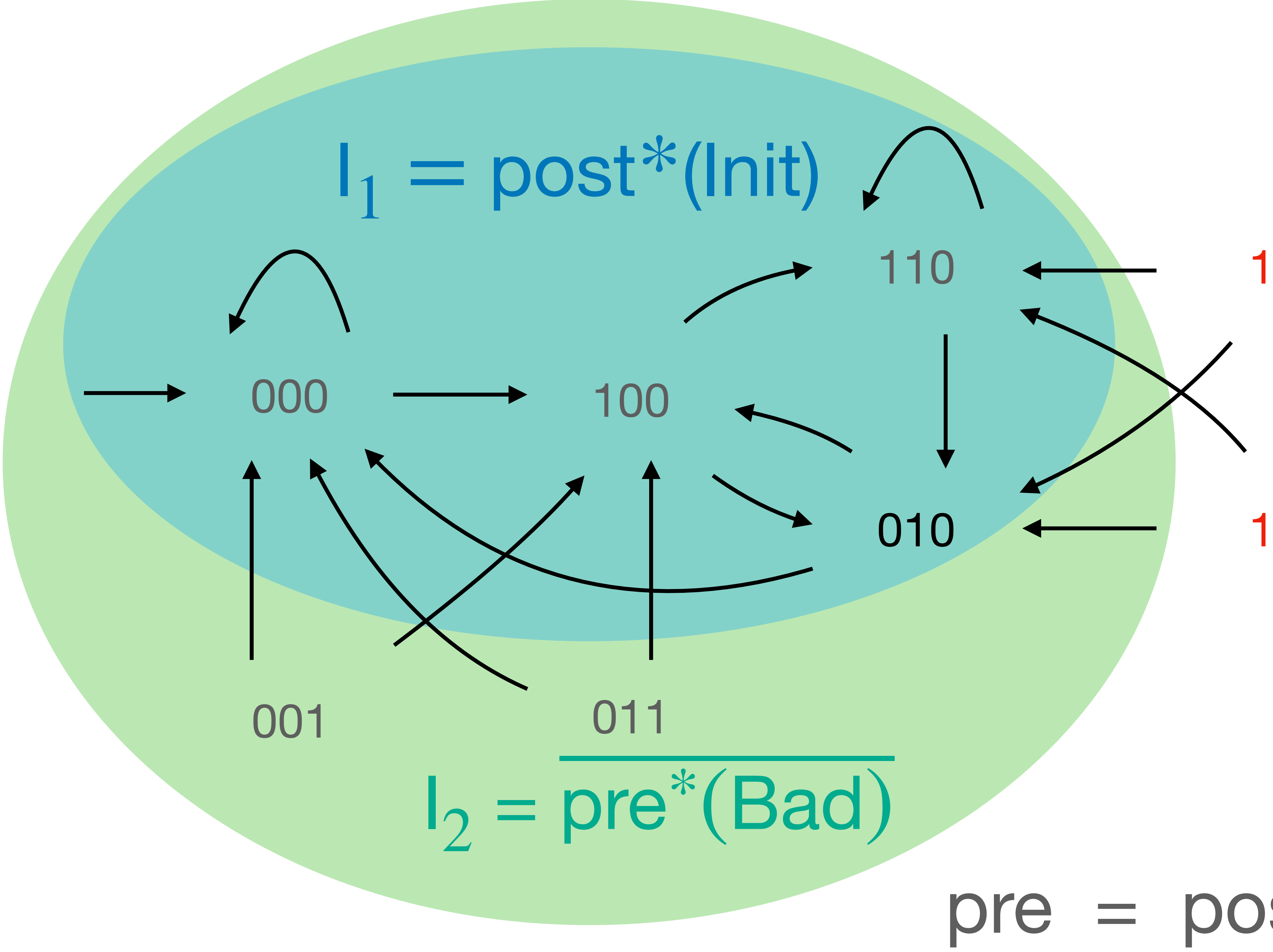
Inductive Invariant I $\subseteq$ States with

$$\text{Init} \subseteq \text{I}$$
$$\text{post(I)} \subseteq \text{I}\,.$$

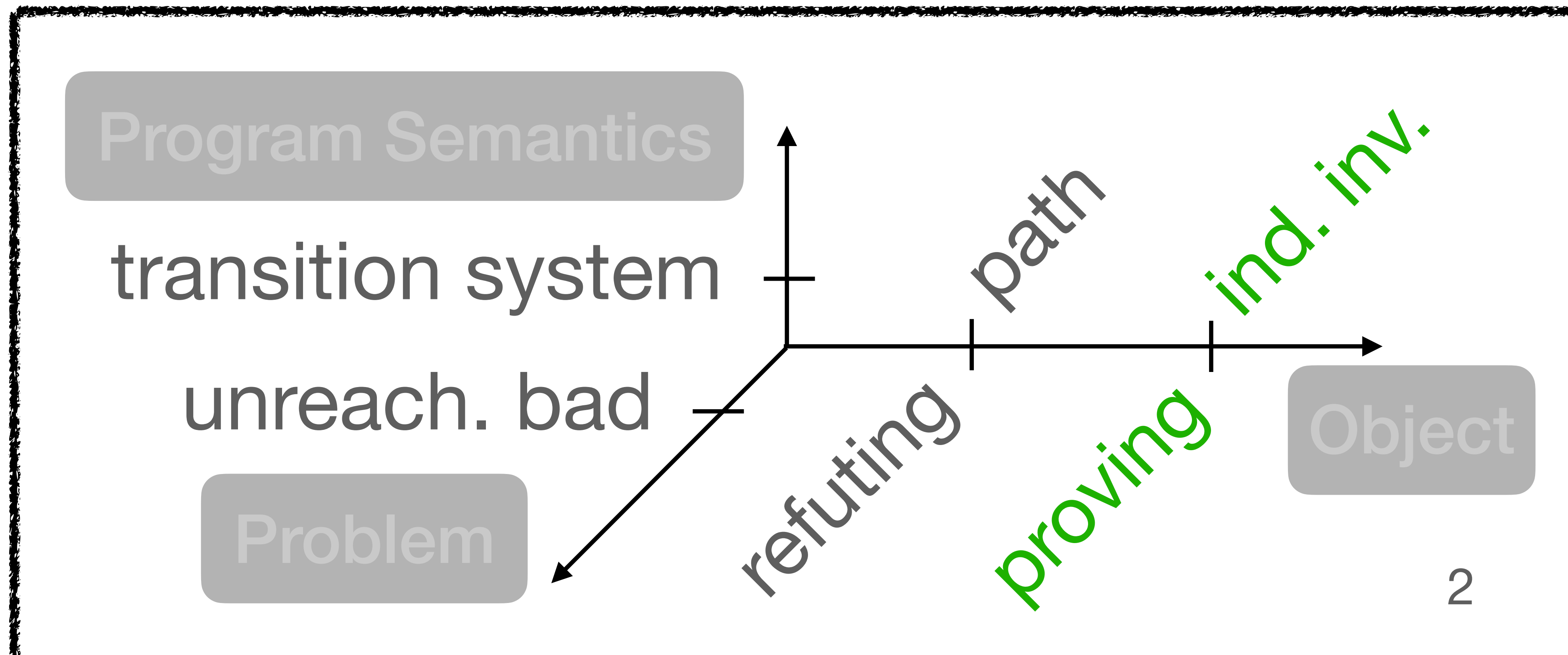
Save wrt. Bad, if I $\cap$ Bad $= \emptyset$.



1967



pre = post^{-1}



Automated Safety Verification: Invariants

Inductive Invariant $I \subseteq$ States with

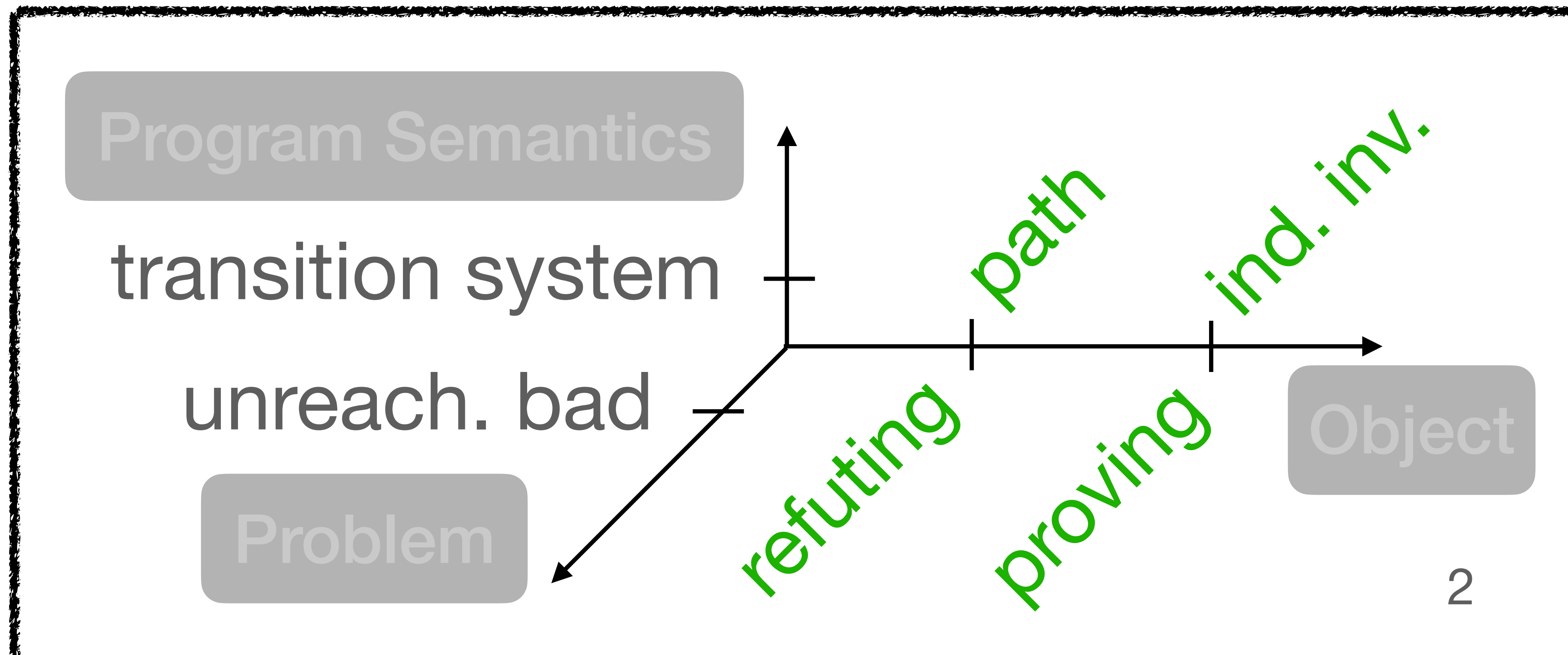
$$\text{Init} \subseteq I$$
$$\text{post}(I) \subseteq I\ .$$

Save wrt. Bad, if $I \cap \text{Bad} = \emptyset$.

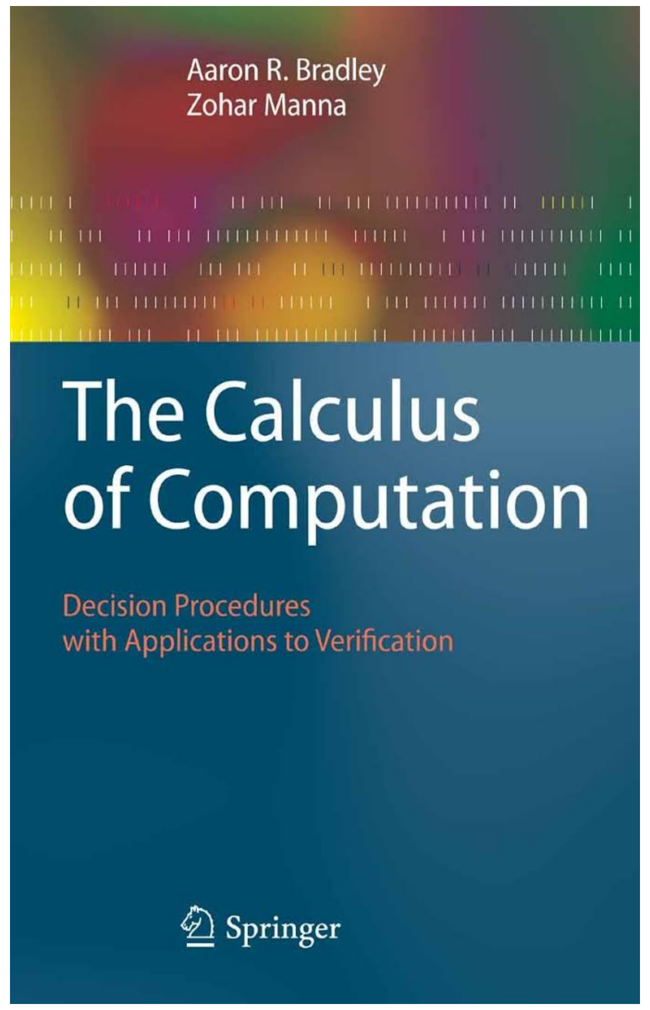
Path $p \in \text{States}^+$ with

$$p_0 \in \text{Init}$$
$$p_{i+1} \in \text{post}(p_i) \text{ for all } i\ .$$

Reaches Bad, if $\text{last}(p) \in \text{Bad}$.



IC3



[SAT-Based Model Checking without Unrolling, Bradley, VMCAI'11]

Importance

*``[…] the **first truly new** bit-level symbolic model checking **algorithm** **since** Ken McMillan's **interpolation** based model checking procedure in 2003.´´*

*``[…] the **most important contribution** to bit-level formal verification **in almost a decade**.´´*

[Een et al., FMCAD'11]

Importance

*``[…] the **first truly new** bit-level symbolic model checking **algorithm***
***since** Ken McMillan's **interpolation** based model checking procedure in 2003.´´*

*``[…] the **most important contribution** to bit-level formal verification **in almost a decade.**´´*

[Een et al., FMCAD'11]

They suggested the name
PDR for Bradley's method and
leave IC3 for the implementation!

Importance

*``[…] the **first truly new** bit-level symbolic model checking **algorithm***
* **since** Ken McMillan's **interpolation** based model checking procedure in 2003.´´*

*``[…] the **most important contribution** to bit-level formal verification **in almost a decade.**´´*

[Een et al., FMCAD'11]

They suggested the name
PDR for Bradley's method and
leave IC3 for the implementation!

HWMCC 2010: Bronze against mature tools.
HWMCC 2012: All tools IC3 based.

HWMCC 2012: **Award!**

HWMCC 2024: `avr_dp`, `fric3`, `nuXmv`, `Pono`, `rIC3`, `Satvik`.

New papers every year at major conferences!

IC3 in Essence

[Hasuo et al., CAV'22]

Invariants are sound (+c) for proving correctness.
Paths are sound (+c) for finding bugs.

IC3 in Essence

[Hasuo et al., CAV'22]

Invariants are sound (+c) for proving correctness.
Paths are sound (+c) for finding bugs.

Combine the two!

IC3 in Essence

[Hasuo et al., CAV'22]

Invariants are sound (+c) for proving correctness.
Paths are sound (+c) for finding bugs.

Combine the two!

IC3 maintains two sequences of sets of states

IC3 in Essence

[Hasuo et al., CAV'22]

Invariants are sound (+c) for proving correctness.
Paths are sound (+c) for finding bugs.

Combine the two!

IC3 maintains two sequences of sets of states

IC3 in Essence

[Hasuo et al., CAV'22]

Invariants are sound (+c) for proving correctness.
Paths are sound (+c) for finding bugs.

Combine the two!

IC3 maintains two sequences of sets of states

Candidate invariants that
over-approximate
forward reachability from Init!

IC3 in Essence

[Hasuo et al., CAV'22]

Invariants are sound (+c) for proving correctness.
Paths are sound (+c) for finding bugs.

Combine the two!

IC3 maintains two sequences of sets of states

Candidate invariants that over-approximate forward reachability from Init!

Candidate paths that under-approximate backward reachability from Bad!

IC3 in Essence

[Hasuo et al., CAV'22]

Invariants are sound (+c) for proving correctness.
Paths are sound (+c) for finding bugs.

Combine the two!

Incremental Construction

IC3 maintains two sequences of sets of states

Candidate invariants that over-approximate forward reachability from Init!

Candidate paths that under-approximate backward reachability from Bad!

IC3 in Essence

[Hasuo et al., CAV'22]

Over-approximate forwards

Under-approximate backwards

IC3 maintains sequences of sets of states

$F_0 \subseteq \ldots \subseteq F_k$ candidate invariants

$B_0 \subseteq \ldots \subseteq B_k$ candidate paths

satisfying

IC3 in Essence

[Hasuo et al., CAV'22]

Over-approximate forwards

Under-approximate backwards

IC3 maintains sequences of sets of states

$F_0 \subseteq \ldots \subseteq F_k$ candidate invariants

$B_0 \subseteq \ldots \subseteq B_k$ candidate paths

satisfying

Frames and proof obligations in the literature.

IC3 in Essence

[Hasuo et al., CAV'22]

Over-approximate forwards

Under-approximate backwards

IC3 maintains sequences of sets of states

$F_0 \subseteq \ldots \subseteq F_k$ candidate invariants

$B_0 \subseteq \ldots \subseteq B_k$ candidate paths

satisfying

1. $F_0 = \mathrm{Init}$
   $\mathrm{Bad} = B_0$

IC3 in Essence

[Hasuo et al., CAV'22]

Over-approximate forwards

Under-approximate backwards

IC3 maintains sequences of sets of states

$F_0 \subseteq \ldots \subseteq F_k$ candidate invariants

$B_0 \subseteq \ldots \subseteq B_k$ candidate paths

satisfying

1. $F_0 = \mathrm{Init}$

   $\mathrm{Bad} = B_0$

3. $F_k \cap B_k = \emptyset$

IC3 in Essence

[Hasuo et al., CAV'22]

Over-approximate forwards

Under-approximate backwards

IC3 maintains sequences of sets of states

$F_0 \subseteq \ldots \subseteq F_k$ candidate invariants

$B_0 \subseteq \ldots \subseteq B_k$ candidate paths

satisfying

1. $F_0 = \mathrm{Init}$

   $\mathrm{Bad} = B_0$

2. $F_i \cup \mathrm{post}(F_i) \subseteq F_{i+1}$

   $B_{i+1} \subseteq B_i \cup \mathrm{pre}(B_i)$

3. $F_k \cap B_k = \varnothing$

IC3 in Essence

[Hasuo et al., CAV'22]

Over-approximate forwards

Under-approximate backwards

IC3 maintains sequences of sets of states

$F_0 \subseteq \ldots \subseteq F_k$ candidate invariants

$B_0 \subseteq \ldots \subseteq B_k$ candidate paths

satisfying

1. $F_0 = \mathrm{Init}$
   $\mathrm{Bad} = B_0$
2. $F_i \cup \mathrm{post}(F_i) \subseteq F_{i+1}$ // $\mathrm{post}^{\leq i}(\mathrm{Init}) \subseteq F_i$
   $B_{i+1} \subseteq B_i \cup \mathrm{pre}(B_i)$
3. $F_k \cap B_k = \varnothing$

IC3 in Essence

[Hasuo et al., CAV'22]

Over-approximate forwards

Under-approximate backwards

IC3 maintains sequences of sets of states

$F_0 \subseteq \ldots \subseteq F_k$ candidate invariants

$B_0 \subseteq \ldots \subseteq B_k$ candidate paths

satisfying

1. $F_0 = \mathrm{Init}$

   $\mathrm{Bad} = B_0$

2. $F_i \cup \mathrm{post}(F_i) \subseteq F_{i+1}$ // $\mathrm{post}^{\leq i}(\mathrm{Init}) \subseteq F_i$

   $B_{i+1} \subseteq B_i \cup \mathrm{pre}(B_i)$ // $B_i \subseteq \mathrm{pre}^{\leq i}(\mathrm{Bad})$

3. $F_k \cap B_k = \varnothing$

f.a. i .

IC3 in Essence

[Hasuo et al., CAV'22]

Extend $F_0 \subseteq \ldots \subseteq F_k$ and $B_0 \subseteq \ldots \subseteq B_k$

Idea:

1. $\mathrm{Init} = F_0$
   $B_0 = \mathrm{Bad}$
2. $F_i \cup \mathrm{post}(F_i) \subseteq F_{i+1}$
   $B_{i+1} \subseteq B_i \cup \mathrm{pre}(B_i)$
3. $F_k \cap B_k = \varnothing$

IC3 in Essence

[Hasuo et al., CAV'22]

Extend $F_0 \subseteq \ldots \subseteq F_k$ and $B_0 \subseteq \ldots \subseteq B_k$

Idea:

$$F_{k+1} := \overline{B_k}$$
$$B_{k+1} := B_k$$

1. $\mathrm{Init} = F_0$
   $B_0 = \mathrm{Bad}$
2. $F_i \cup \mathrm{post}(F_i) \subseteq F_{i+1}$
   $B_{i+1} \subseteq B_i \cup \mathrm{pre}(B_i)$
3. $F_k \cap B_k = \varnothing$

IC3 in Essence

[Hasuo et al., CAV'22]

Extend $F_0 \subseteq \ldots \subseteq F_k$ and $B_0 \subseteq \ldots \subseteq B_k$

Idea:

$F_{k+1} := \overline{B_k}$ // $F_k \subseteq F_{k+1}$ **by 3.** ✔

$B_{k+1} := B_k$

1. Init $= F_0$
   $B_0 =$ Bad
2. $F_i \cup \text{post}(F_i) \subseteq F_{i+1}$
   $B_{i+1} \subseteq B_i \cup \text{pre}(B_i)$
3. $F_k \cap B_k = \varnothing$

IC3 in Essence

[Hasuo et al., CAV'22]

Extend $F_0 \subseteq \ldots \subseteq F_k$ and $B_0 \subseteq \ldots \subseteq B_k$

Idea:

$F_{k+1} := \overline{B_k}$ // $F_k \subseteq F_{k+1}$ by 3. ✔

$B_{k+1} := B_k$ // $B_{k+1} \subseteq B_k \cup \mathrm{pre}(B_k)$ ✔

1. $\mathrm{Init} = F_0$
   $B_0 = \mathrm{Bad}$
2. $F_i \cup \mathrm{post}(F_i) \subseteq F_{i+1}$
   $B_{i+1} \subseteq B_i \cup \mathrm{pre}(B_i)$
3. $F_k \cap B_k = \varnothing$

IC3 in Essence

[Hasuo et al., CAV'22]

Extend $F_0 \subseteq \ldots \subseteq F_k$ and $B_0 \subseteq \ldots \subseteq B_k$

Idea:

$F_{k+1} := \overline{B_k}$ // $F_k \subseteq F_{k+1}$ by 3. ✔

$B_{k+1} := B_k$ // $B_{k+1} \subseteq B_k \cup \mathrm{pre}(B_k)$ ✔

// $F_{k+1} \cap B_{k+1} = \varnothing$ ✔

1. $\mathrm{Init} = F_0$
   $B_0 = \mathrm{Bad}$
2. $F_i \cup \mathrm{post}(F_i) \subseteq F_{i+1}$
   $B_{i+1} \subseteq B_i \cup \mathrm{pre}(B_i)$
3. $F_k \cap B_k = \varnothing$

IC3 in Essence

[Hasuo et al., CAV'22]

Extend $F_0 \subseteq \ldots \subseteq F_k$ and $B_0 \subseteq \ldots \subseteq B_k$

Idea:

$$F_{k+1} := \overline{B_k}$$

$$B_{k+1} := B_k$$

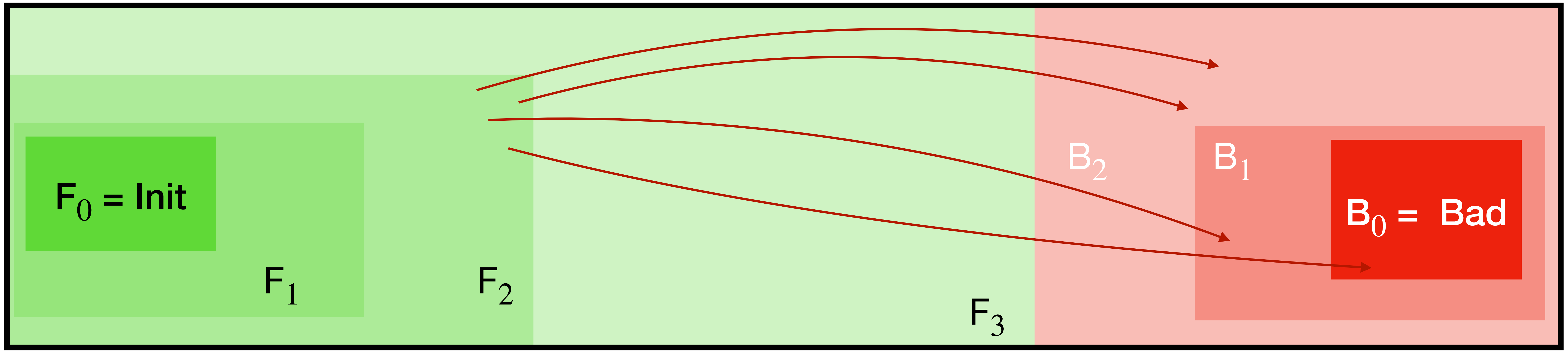
Problem:

$$\mathrm{post}(F_k) \not\subseteq F_{k+1}$$

1. $\mathrm{Init} = F_0$
   $B_0 = \mathrm{Bad}$
2. $F_i \cup \mathrm{post}(F_i) \subseteq F_{i+1}$
   $B_{i+1} \subseteq B_i \cup \mathrm{pre}(B_i)$
3. $F_k \cap B_k = \varnothing$

IC3 in Essence

[Hasuo et al., CAV'22]



Extend $F_0 \subseteq \ldots \subseteq F_k$ and $B_0 \subseteq \ldots \subseteq B_k$

Idea:

$F_{k+1} := \overline{B_k}$

$B_{k+1} := B_k$

Problem:

$\mathrm{post}(F_k) \nsubseteq F_{k+1}$

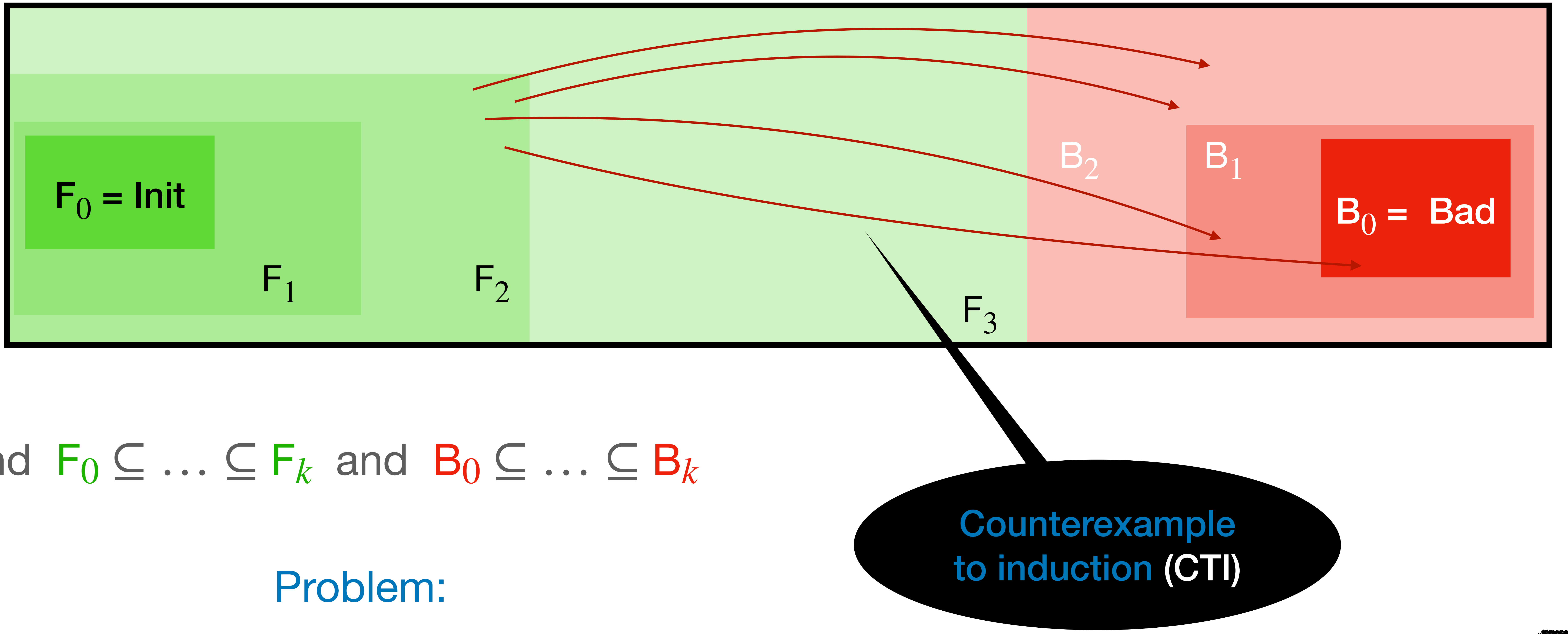
iff

$\mathrm{post}(F_k) \cap B_k \neq \emptyset$

1. $\mathrm{Init} = F_0$
   $B_0 = \mathrm{Bad}$
2. $F_i \cup \mathrm{post}(F_i) \subseteq F_{i+1}$
   $B_{i+1} \subseteq B_i \cup \mathrm{pre}(B_i)$
3. $F_k \cap B_k = \emptyset$

IC3 in Essence

[Hasuo et al., CAV'22]



Extend $F_0 \subseteq \ldots \subseteq F_k$ and $B_0 \subseteq \ldots \subseteq B_k$

Idea:

$F_{k+1} := \overline{B_k}$

$B_{k+1} := B_k$

Problem:

$\mathrm{post}(F_k) \not\subseteq F_{k+1}$

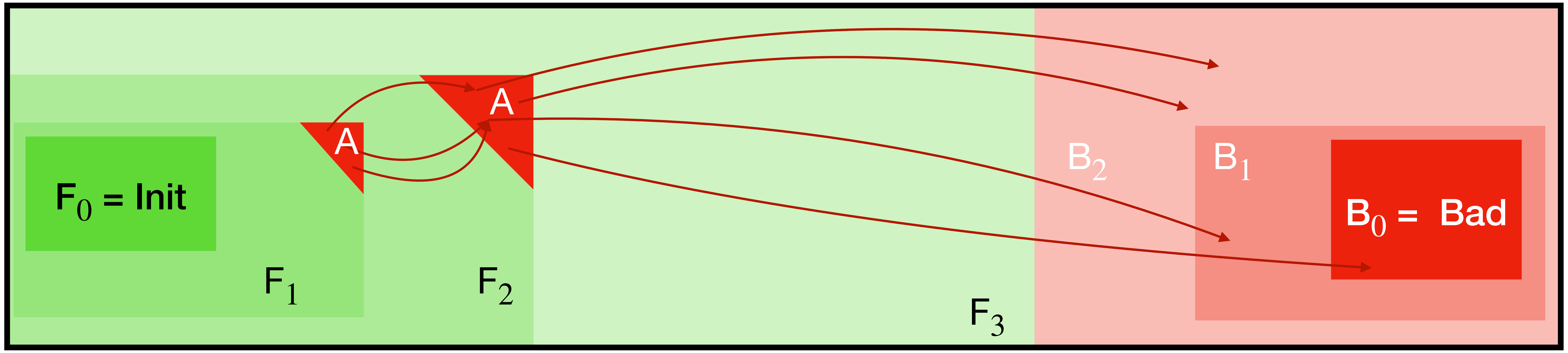
iff

$\mathrm{post}(F_k) \cap B_k \neq \emptyset$

1. $\mathrm{Init} = F_0$
   $B_0 = \mathrm{Bad}$
2. $F_i \cup \mathrm{post}(F_i) \subseteq F_{i+1}$
   $B_{i+1} \subseteq B_i \cup \mathrm{pre}(B_i)$
3. $F_k \cap B_k = \emptyset$

IC3 in Essence

[Hasuo et al., CAV'22]



Extend $F_0 \subseteq \ldots \subseteq F_k$ and $B_0 \subseteq \ldots \subseteq B_k$

Idea:

$$F_{k+1} := \overline{B_k}$$
$$B_{k+1} := B_k$$

Problem:

$$\mathrm{post}(F_k) \nsubseteq F_{k+1}$$

iff

$$\mathrm{post}(F_k) \cap B_k \neq \emptyset$$

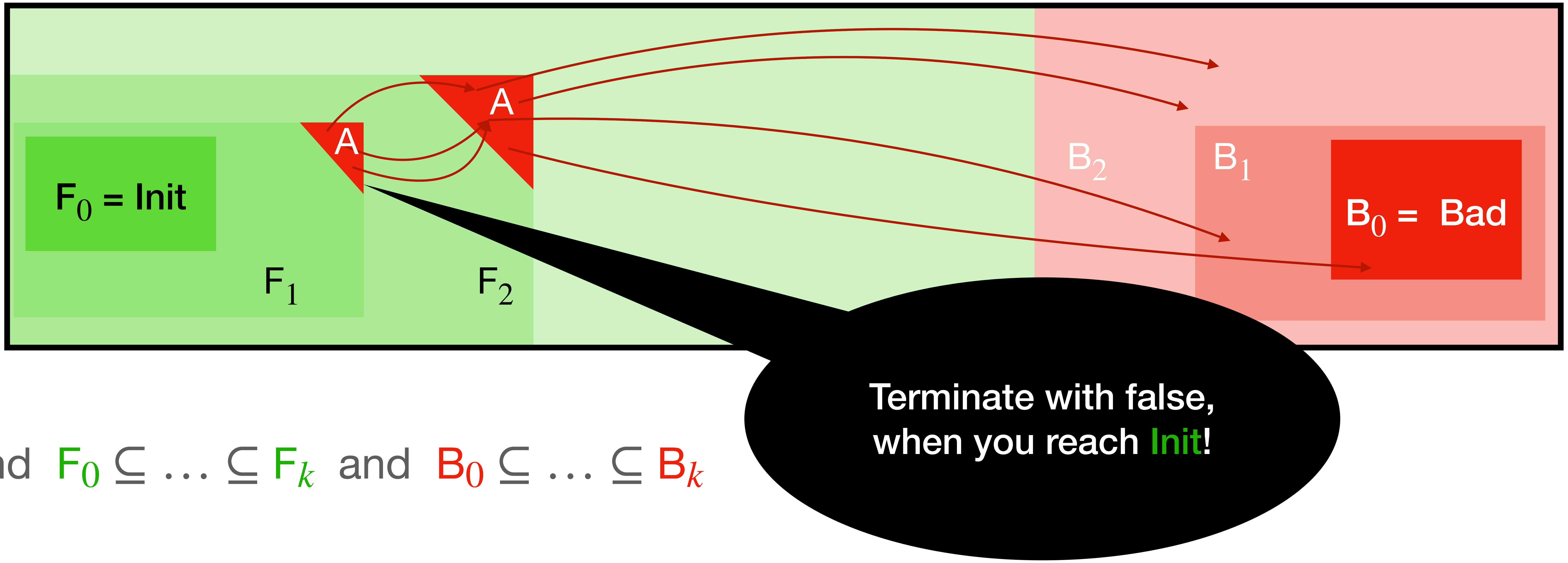
Solution:

$$A := \mathrm{pre}^{F_0 \ldots F_k}(B_k)$$

1. $\mathrm{Init} = F_0$
   $B_0 = \mathrm{Bad}$
2. $F_i \cup \mathrm{post}(F_i) \subseteq F_{i+1}$
   $B_{i+1} \subseteq B_i \cup \mathrm{pre}(B_i)$
3. $F_k \cap B_k = \emptyset$

IC3 in Essence

[Hasuo et al., CAV'22]



Extend $F_0 \subseteq \ldots \subseteq F_k$ and $B_0 \subseteq \ldots \subseteq B_k$

Idea:

$F_{k+1} := \overline{B_k}$

$B_{k+1} := B_k$

Problem:

$\mathrm{post}(F_k) \nsubseteq F_{k+1}$

iff

$\mathrm{post}(F_k) \cap B_k \neq \emptyset$

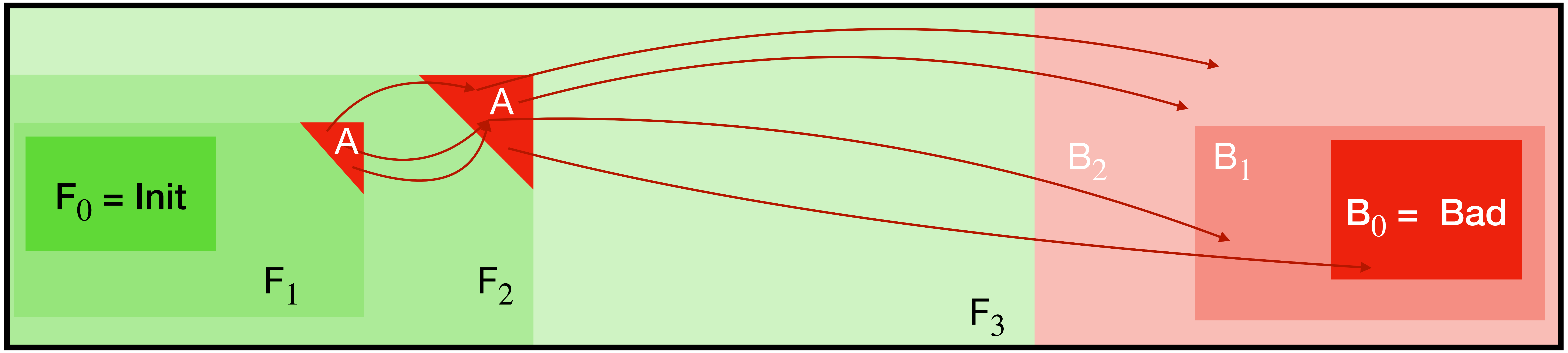
Solution:

$A := \mathrm{pre}^{F_0 \ldots F_k}(B_k)$

1. $\mathrm{Init} = F_0$
   $B_0 = \mathrm{Bad}$
2. $F_i \cup \mathrm{post}(F_i) \subseteq F_{i+1}$
   $B_{i+1} \subseteq B_i \cup \mathrm{pre}(B_i)$
3. $F_k \cap B_k = \emptyset$

IC3 in Essence

[Hasuo et al., CAV'22]



Extend $F_0 \subseteq \ldots \subseteq F_k$ and $B_0 \subseteq \ldots \subseteq B_k$

Idea:

$F_{k+1} := \overline{B_k}$

$B_{k+1} := B_k$

Problem:

$\mathrm{post}(F_k) \not\subseteq F_{k+1}$

iff

$\mathrm{post}(F_k) \cap B_k \neq \varnothing$

Solution:

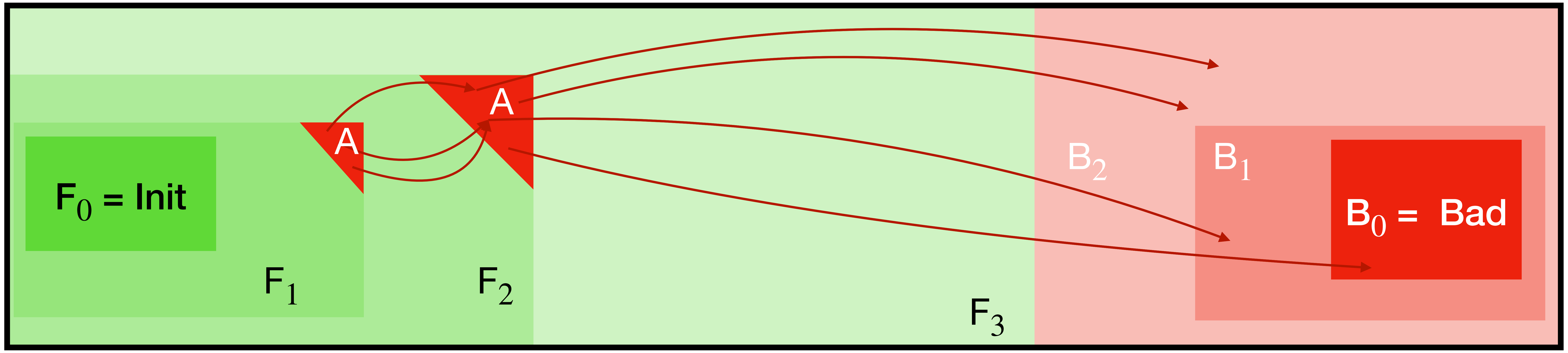
$A := \mathrm{pre}^{F_0 \ldots F_k}(B_k)$

$B_{k+1} := B_k \cup A$

1. $\mathrm{Init} = F_0$
   $B_0 = \mathrm{Bad}$
2. $F_i \cup \mathrm{post}(F_i) \subseteq F_{i+1}$
   $B_{i+1} \subseteq B_i \cup \mathrm{pre}(B_i)$
3. $F_k \cap B_k = \varnothing$

IC3 in Essence

[Hasuo et al., CAV'22]



Extend $F_0 \subseteq \ldots \subseteq F_k$ and $B_0 \subseteq \ldots \subseteq B_k$

Idea:

$$F_{k+1} := \overline{B_k}$$
$$B_{k+1} := B_k$$

Problem:

$$\mathrm{post}(F_k) \nsubseteq F_{k+1}$$

iff

$$\mathrm{post}(F_k) \cap B_k \neq \emptyset$$

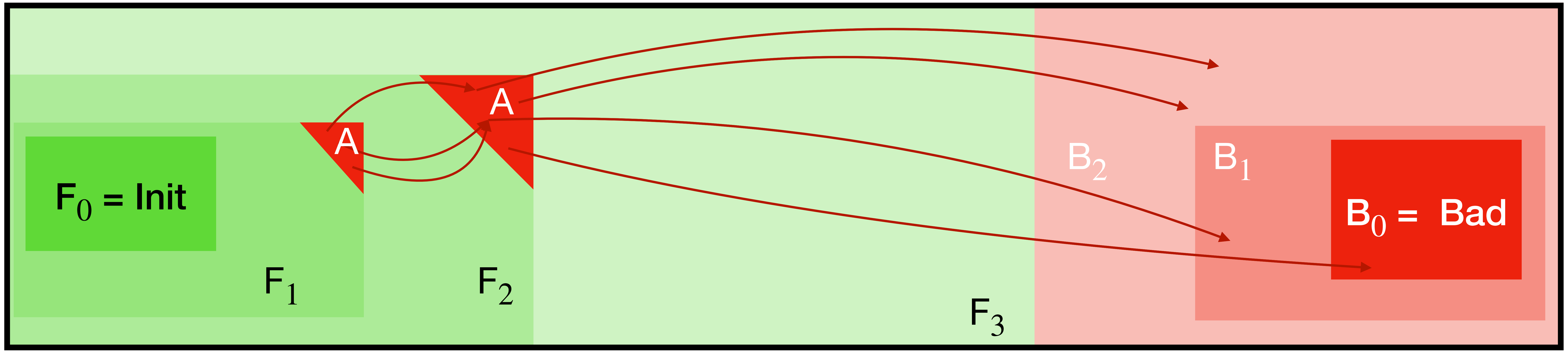
Solution:

$$A := \mathrm{pre}^{F_0 \ldots F_k}(B_k)$$
$$(F_0 \cap \overline{A}) \subseteq \ldots \subseteq (F_k \cap \overline{A}) \subseteq F_{k+1}$$
$$B_{k+1} := B_k \cup A$$

1. $\mathrm{Init} = F_0$
   $B_0 = \mathrm{Bad}$
2. $F_i \cup \mathrm{post}(F_i) \subseteq F_{i+1}$
   $B_{i+1} \subseteq B_i \cup \mathrm{pre}(B_i)$
3. $F_k \cap B_k = \emptyset$

IC3 in Essence

[Hasuo et al., CAV'22]



Extend $F_0 \subseteq \ldots \subseteq F_k$ and $B_0 \subseteq \ldots \subseteq B_k$

This is an operation on sequences [Hasuo et al.'22].

Idea:

$$F_{k+1} := \overline{B_k}$$
$$B_{k+1} := B_k$$

Problem:

$$\mathrm{post}(F_k) \not\subseteq F_{k+1}$$

iff

$$\mathrm{post}(F_k) \cap B_k \neq \varnothing$$

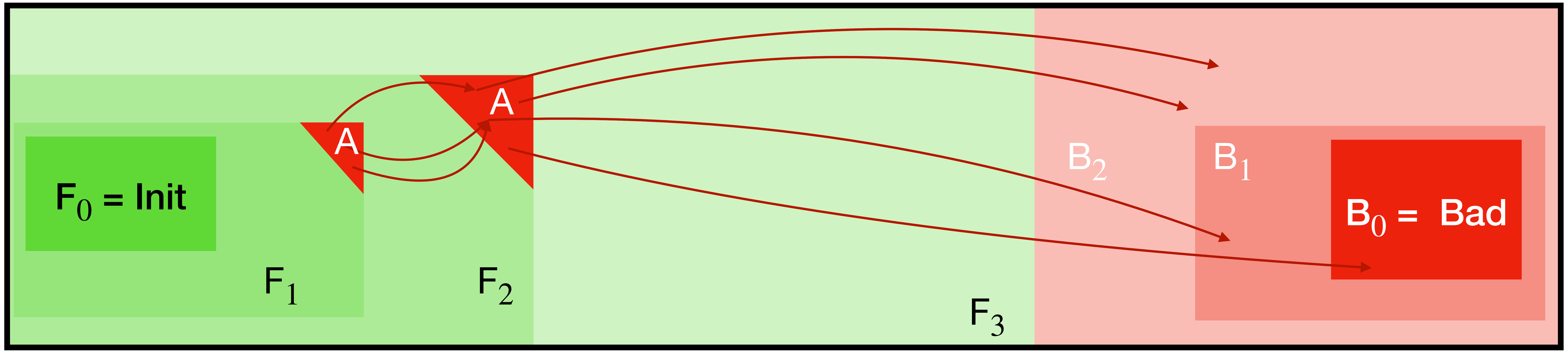
Solution:

$$A := \mathrm{pre}^{F_0 \ldots F_k}(B_k)$$
$$(F_0 \cap \overline{A}) \subseteq \ldots \subseteq (F_k \cap \overline{A}) \subseteq F_{k+1}$$
$$B_{k+1} := B_k \cup A$$

1. $\mathrm{Init} = F_0$
   $B_0 = \mathrm{Bad}$
2. $F_i \cup \mathrm{post}(F_i) \subseteq F_{i+1}$
   $B_{i+1} \subseteq B_i \cup \mathrm{pre}(B_i)$
3. $F_k \cap B_k = \varnothing$

IC3 in Essence

[Hasuo et al., CAV'22]



Extend $F_0 \subseteq \ldots \subseteq F_k$ and $B_0 \subseteq \ldots \subseteq B_k$

Terminate with true, when $(F_k, B_k) = (F_{k+1}, B_{k+1})$.

Idea:

$$F_{k+1} := \overline{B_k}$$

$$B_{k+1} := B_k$$

Problem:

$$\mathrm{post}(F_k) \not\subseteq F_{k+1}$$

iff

$$\mathrm{post}(F_k) \cap B_k \neq \emptyset$$

Solution:

$$A := \mathrm{pre}^{F_0 \ldots F_k}(B_k)$$

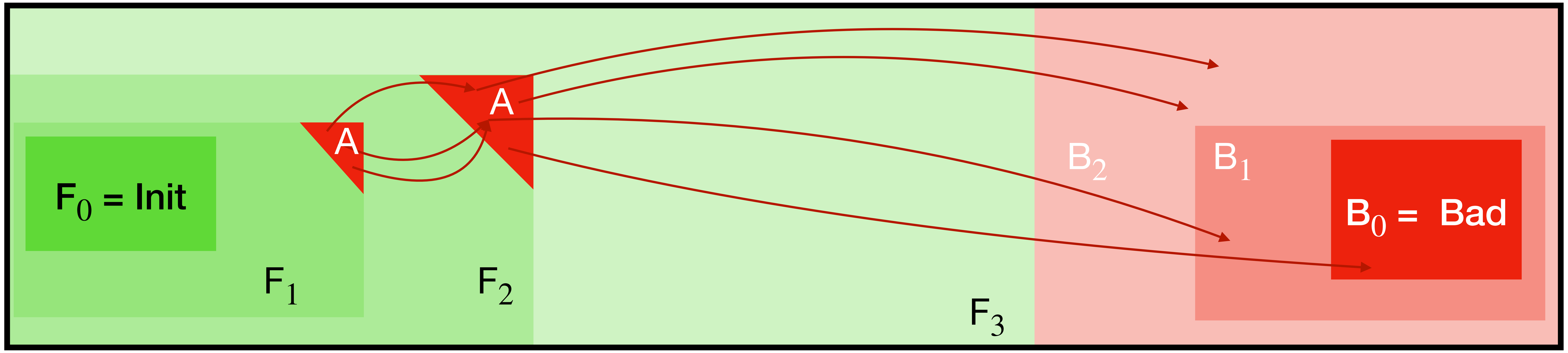
$$(F_0 \cap \overline{A}) \subseteq \ldots \subseteq (F_k \cap \overline{A}) \subseteq F_{k+1}$$

$$B_{k+1} := B_k \cup A$$

1. $\mathrm{Init} = F_0$
   $B_0 = \mathrm{Bad}$
2. $F_i \cup \mathrm{post}(F_i) \subseteq F_{i+1}$
   $B_{i+1} \subseteq B_i \cup \mathrm{pre}(B_i)$
3. $F_k \cap B_k = \emptyset$

IC3 in Essence

[Hasuo et al., CAV'22]



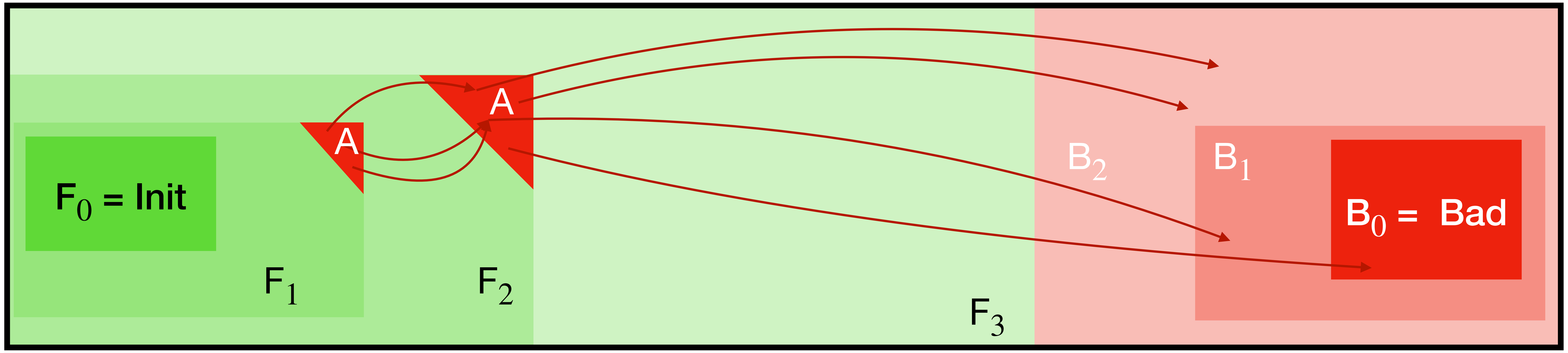
$\mathrm{pre}^{F_0 \ldots F_k}(B_k)$:

$$\mathrm{pre}^{F}(X) = F \cap \mathrm{pre}(X)$$
$$\mathrm{pre}^{\mathscr{F}.F}(X) = \mathrm{pre}^{F}(X) \cup \mathrm{pre}^{\mathscr{F}}(\mathrm{pre}^{F}(X))$$

1. $\mathrm{Init} = F_0$
   $B_0 = \mathrm{Bad}$
2. $F_i \cup \mathrm{post}(F_i) \subseteq F_{i+1}$
   $B_{i+1} \subseteq B_i \cup \mathrm{pre}(B_i)$
3. $F_k \cap B_k = \emptyset$

IC3 in Essence

[Hasuo et al., CAV'22]



$\mathrm{pre}^{F_0 \dots F_k}(B_k)$:

$$\mathrm{pre}^{F}(X) = F \cap \mathrm{pre}(X)$$
$$\mathrm{pre}^{\mathscr{F}.F}(X) = \mathrm{pre}^{F}(X) \cup \mathrm{pre}^{\mathscr{F}}(\mathrm{pre}^{F}(X))$$

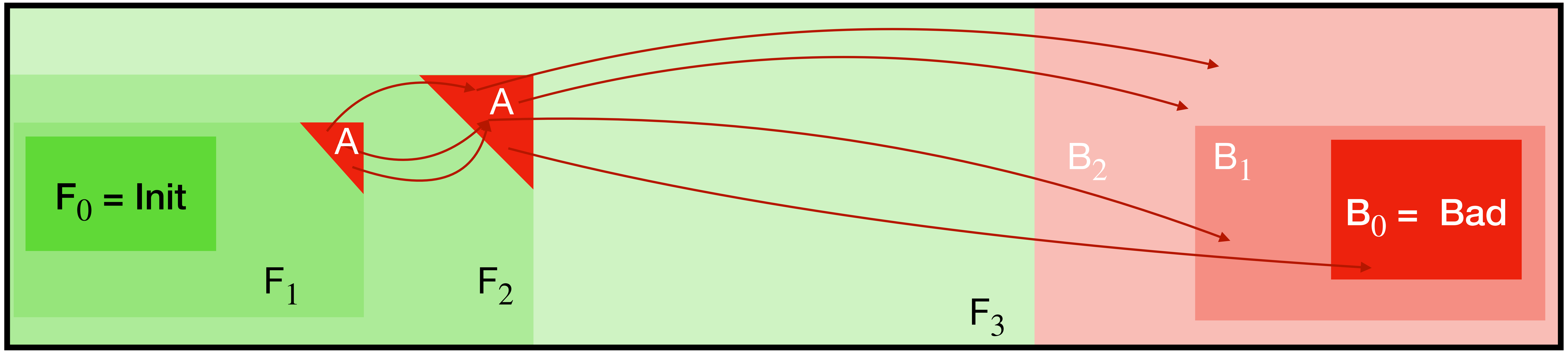
Theorem(S+C+Termination):

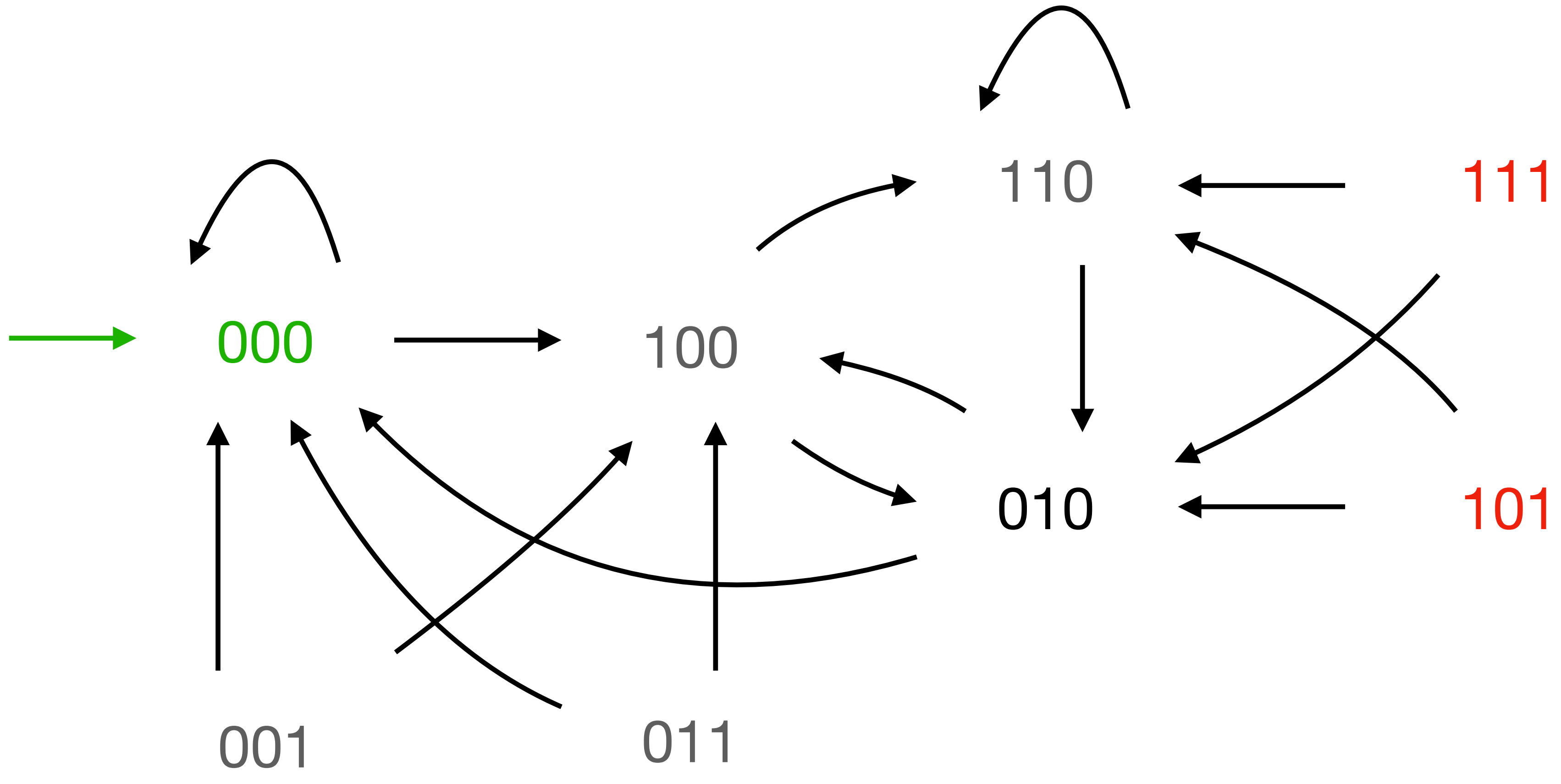
IC3 returns true if and only if $\mathrm{post}^*(\mathrm{Init}) \cap \mathrm{Bad} = \emptyset$.
It is guaranteed to terminate even on WSTS [Majumdar et al., CAV'13].

1. $\mathrm{Init} = F_0$
   $B_0 = \mathrm{Bad}$
2. $F_i \cup \mathrm{post}(F_i) \subseteq F_{i+1}$
   $B_{i+1} \subseteq B_i \cup \mathrm{pre}(B_i)$
3. $F_k \cap B_k = \emptyset$

IC3 on the Example

[Hasuo et al., CAV'22]

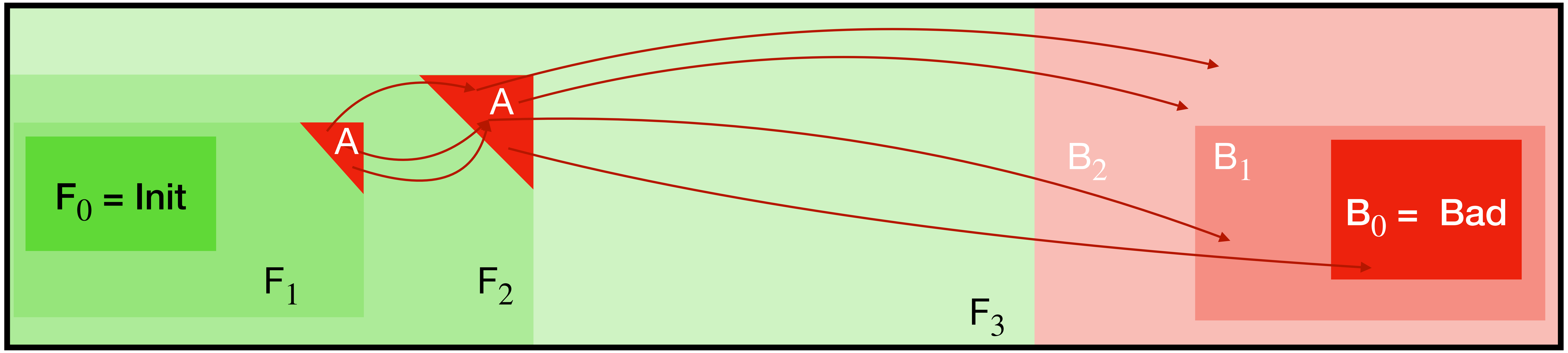


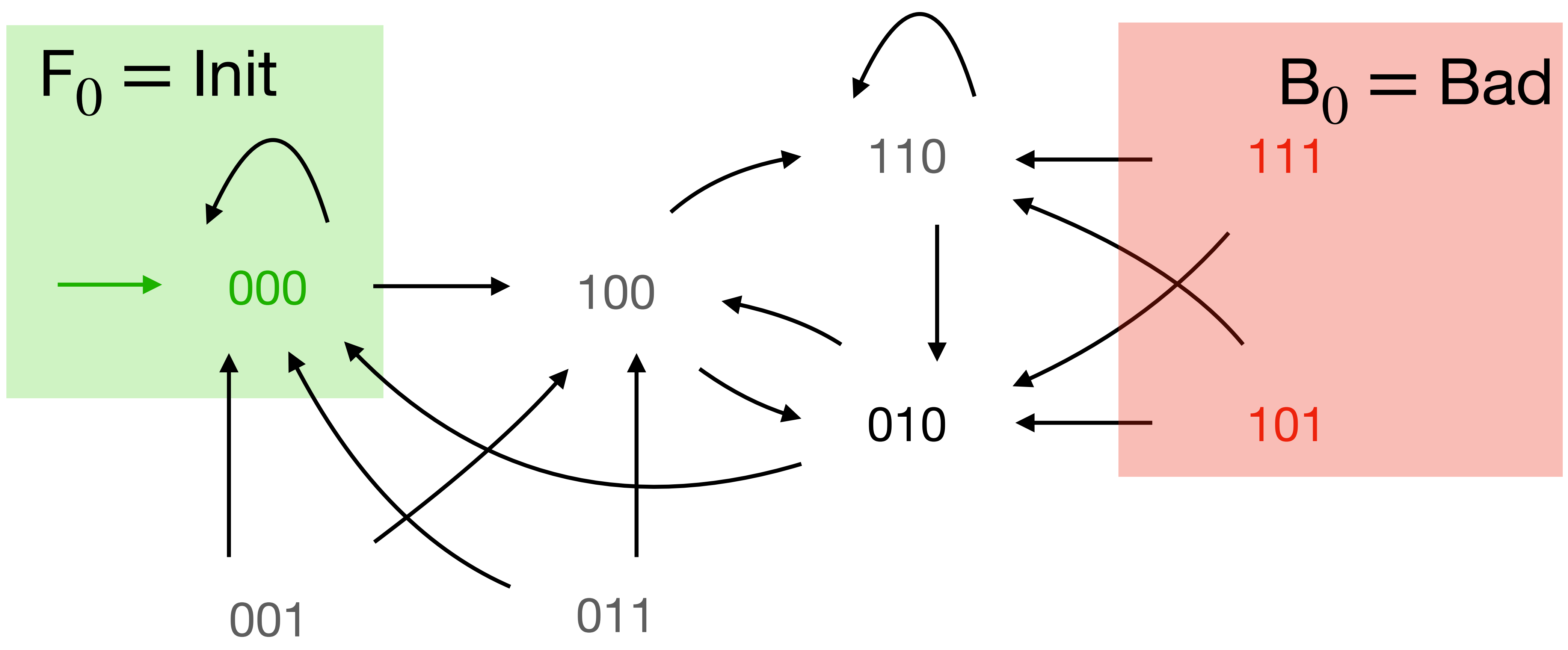


1. Init $= F_0$
   $B_0 =$ Bad
2. $F_i \cup \mathrm{post}(F_i) \subseteq F_{i+1}$
   $B_{i+1} \subseteq B_i \cup \mathrm{pre}(B_i)$
3. $F_k \cap B_k = \varnothing$

IC3 on the Example

[Hasuo et al., CAV'22]

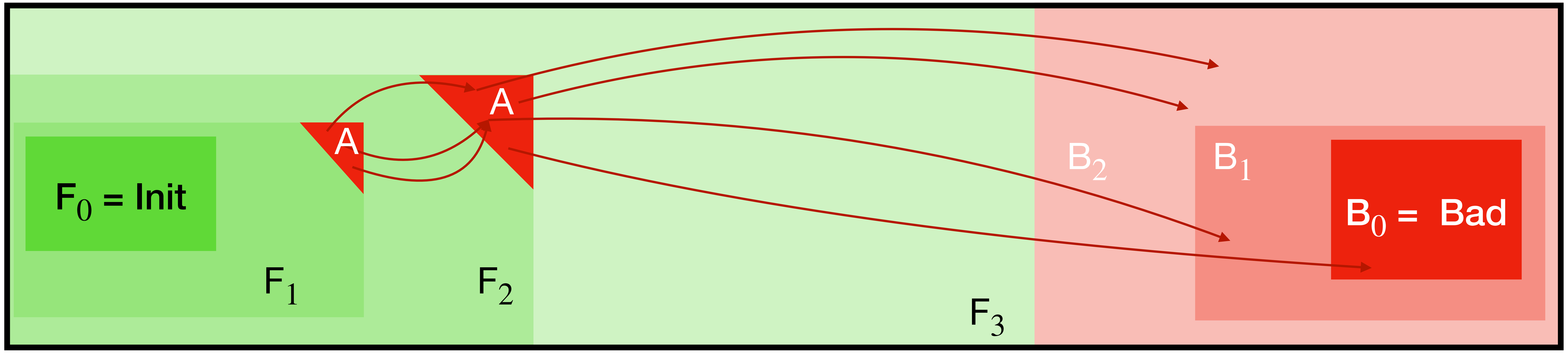


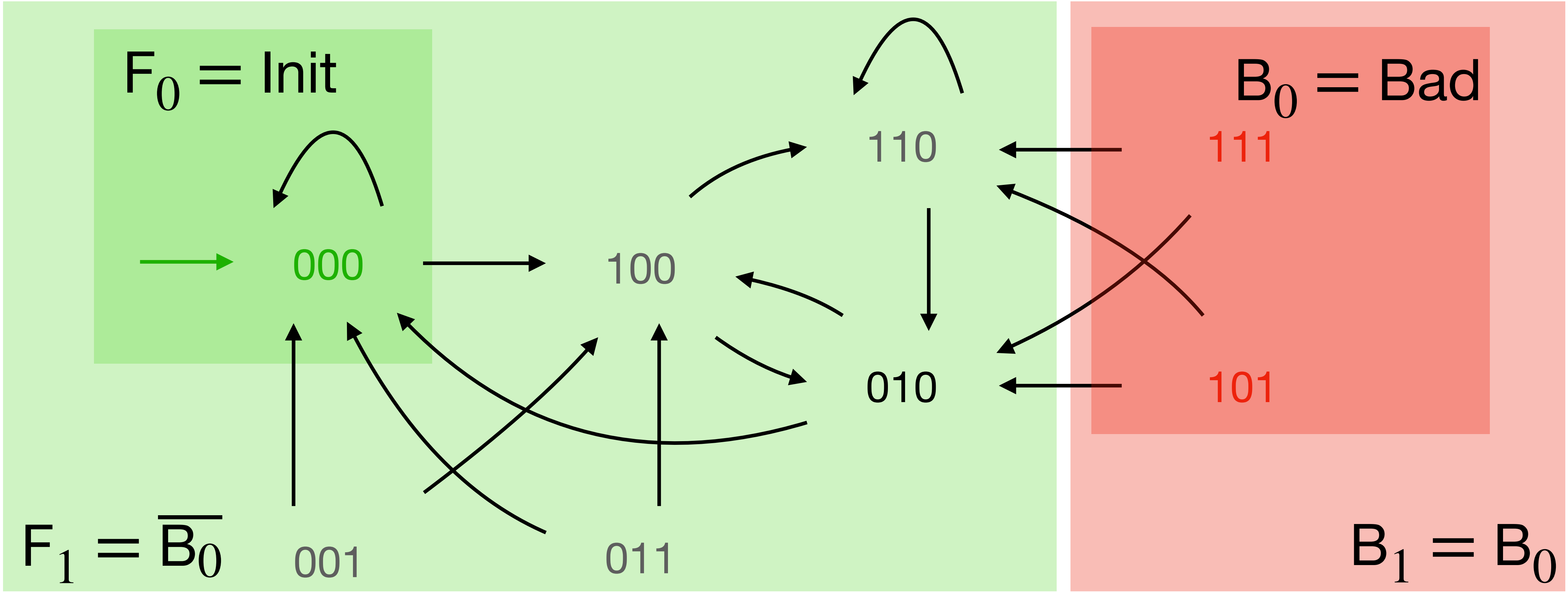


1. $\mathrm{Init} = F_0$
   $B_0 = \mathrm{Bad}$
2. $F_i \cup \mathrm{post}(F_i) \subseteq F_{i+1}$
   $B_{i+1} \subseteq B_i \cup \mathrm{pre}(B_i)$
3. $F_k \cap B_k = \varnothing$

IC3 on the Example

[Hasuo et al., CAV'22]

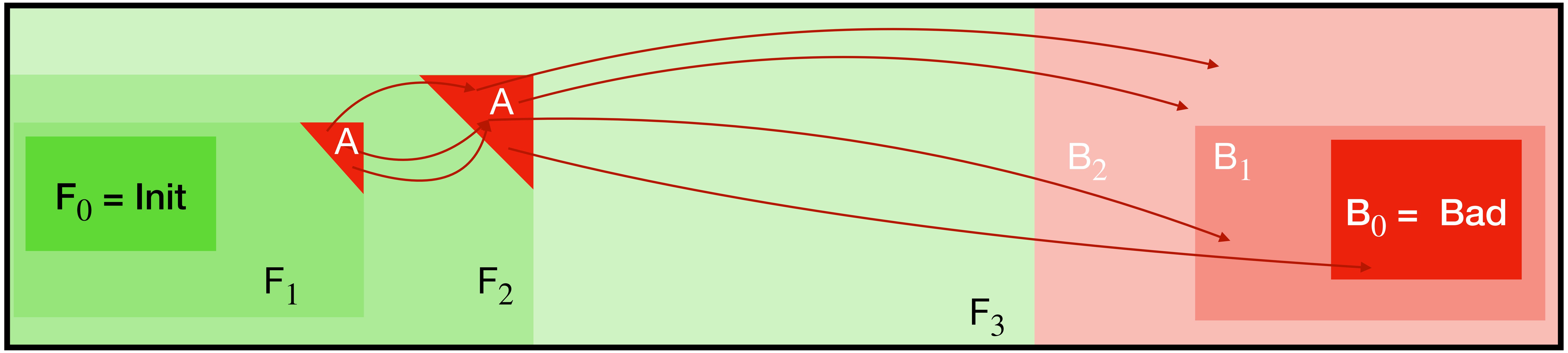


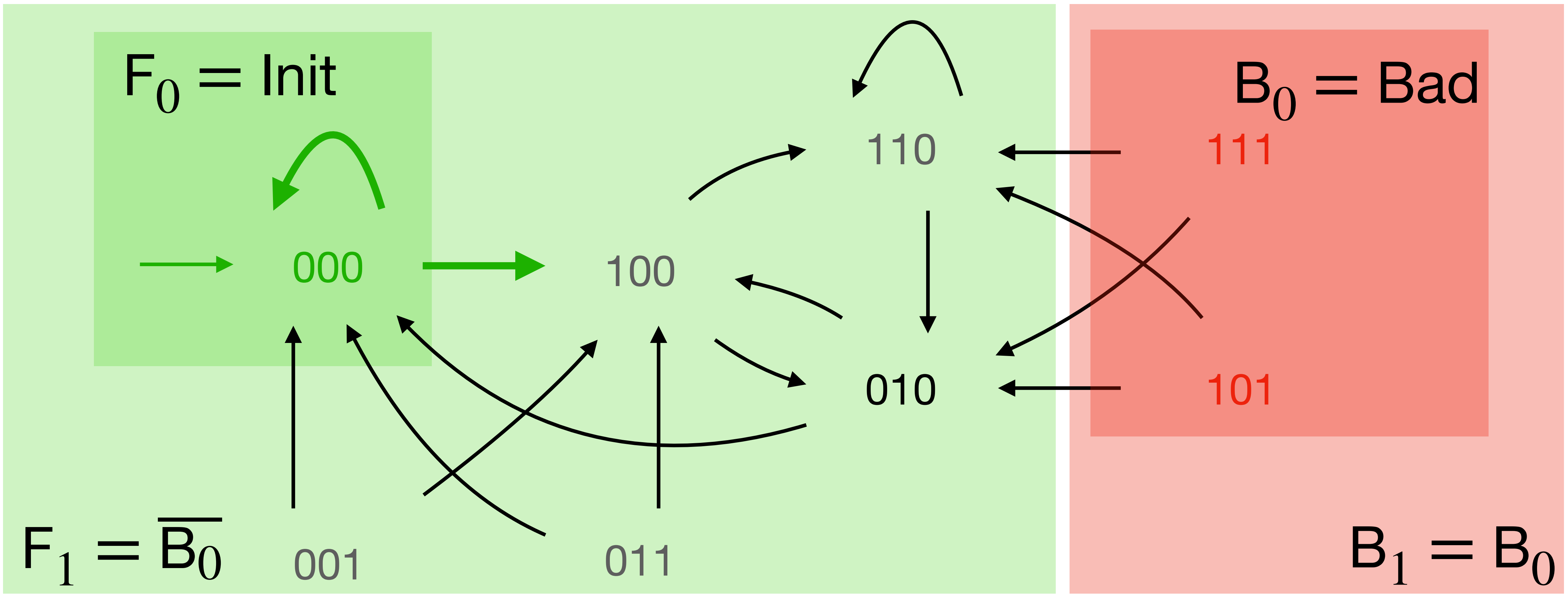


1. $\text{Init} = F_0$
   $B_0 = \text{Bad}$
2. $F_i \cup \text{post}(F_i) \subseteq F_{i+1}$
   $B_{i+1} \subseteq B_i \cup \text{pre}(B_i)$
3. $F_k \cap B_k = \emptyset$

IC3 on the Example

[Hasuo et al., CAV'22]



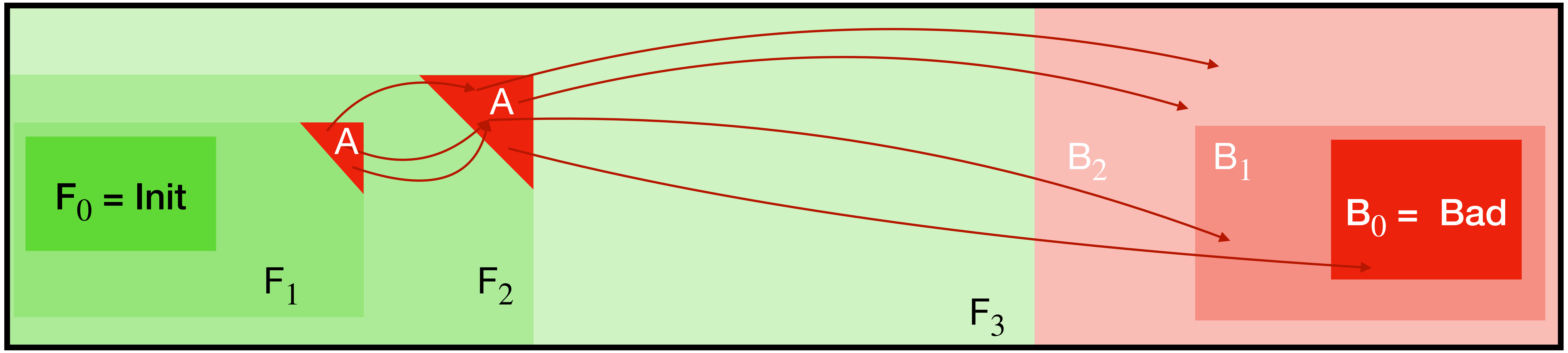


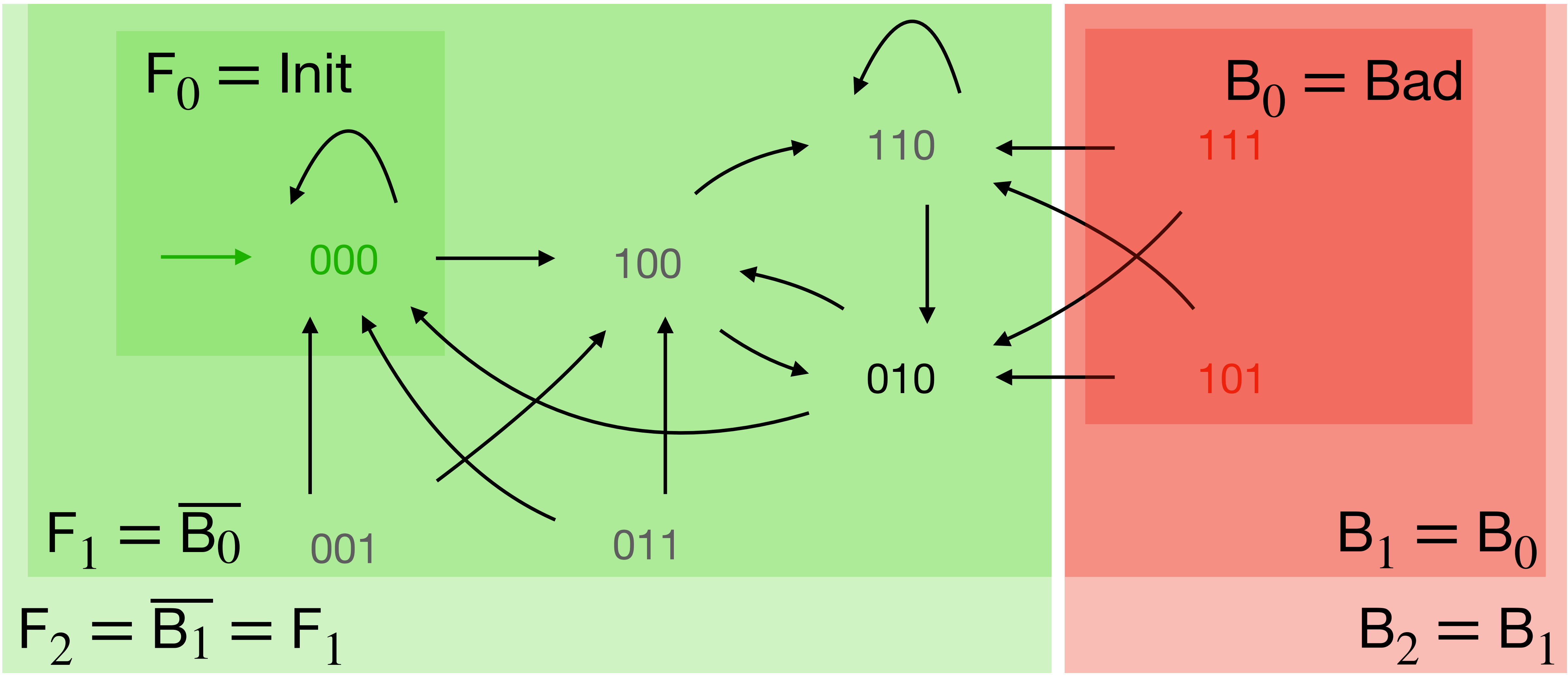
$\mathrm{post}(F_0) \subseteq F_1$ so $A_1 = \varnothing$

1. $\mathrm{Init} = F_0$
   $B_0 = \mathrm{Bad}$
2. $F_i \cup \mathrm{post}(F_i) \subseteq F_{i+1}$
   $B_{i+1} \subseteq B_i \cup \mathrm{pre}(B_i)$
3. $F_k \cap B_k = \varnothing$

IC3 on the Example

[Hasuo et al., CAV’22]



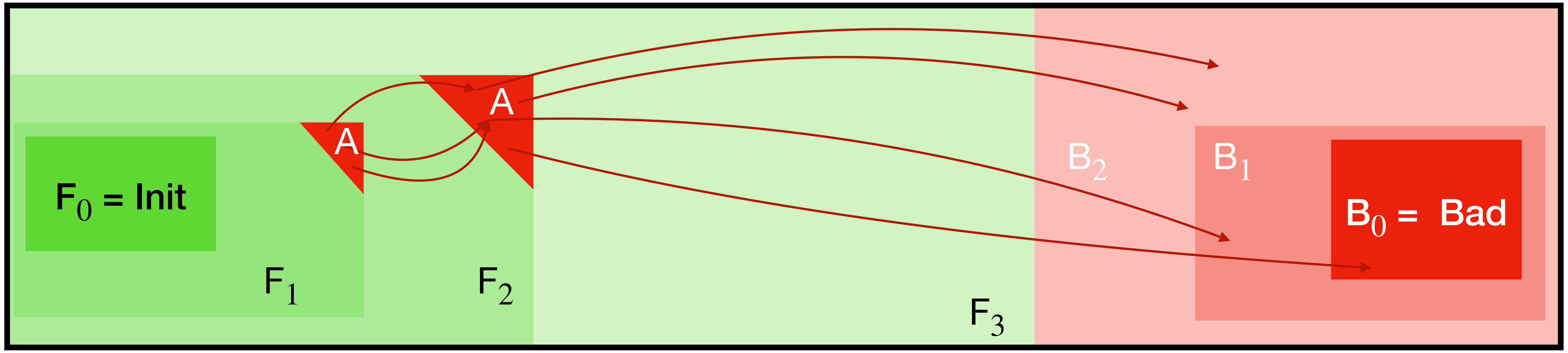


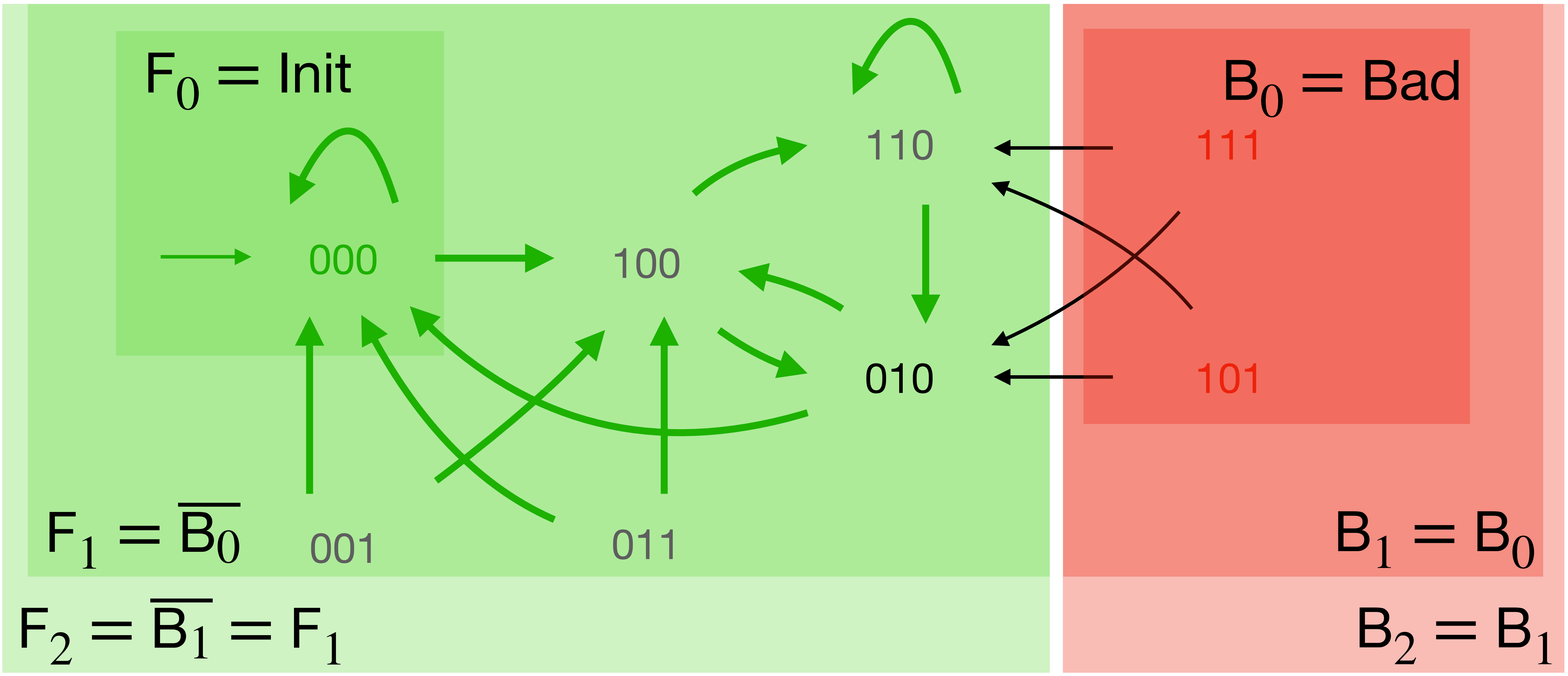
$\text{post}(F_0) \subseteq F_1$ so $A_1 = \varnothing$

1. $\text{Init} = F_0$
   $B_0 = \text{Bad}$
2. $F_i \cup \text{post}(F_i) \subseteq F_{i+1}$
   $B_{i+1} \subseteq B_i \cup \text{pre}(B_i)$
3. $F_k \cap B_k = \varnothing$

IC3 on the Example

[Hasuo et al., CAV'22]





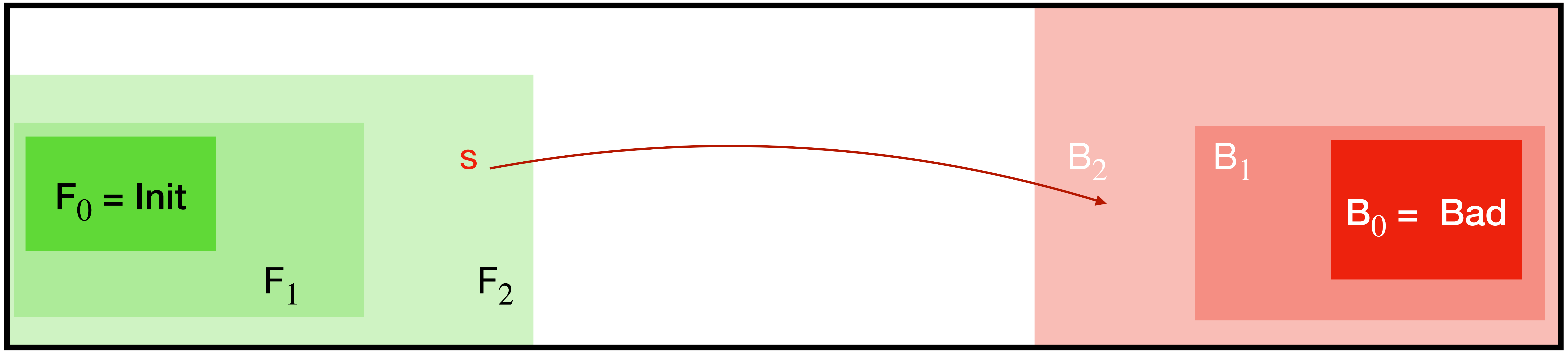
$\mathrm{post}(F_0) \subseteq F_1$ so $A_1 = \varnothing$

$\mathrm{post}(F_1) \subseteq F_2$ so $A_2 = \varnothing$

1. $\mathrm{Init} = F_0$
   $B_0 = \mathrm{Bad}$
2. $F_i \cup \mathrm{post}(F_i) \subseteq F_{i+1}$
   $B_{i+1} \subseteq B_i \cup \mathrm{pre}(B_i)$
3. $F_k \cap B_k = \varnothing$

IC3 in Detail

[Hasuo et al., CAV'22]

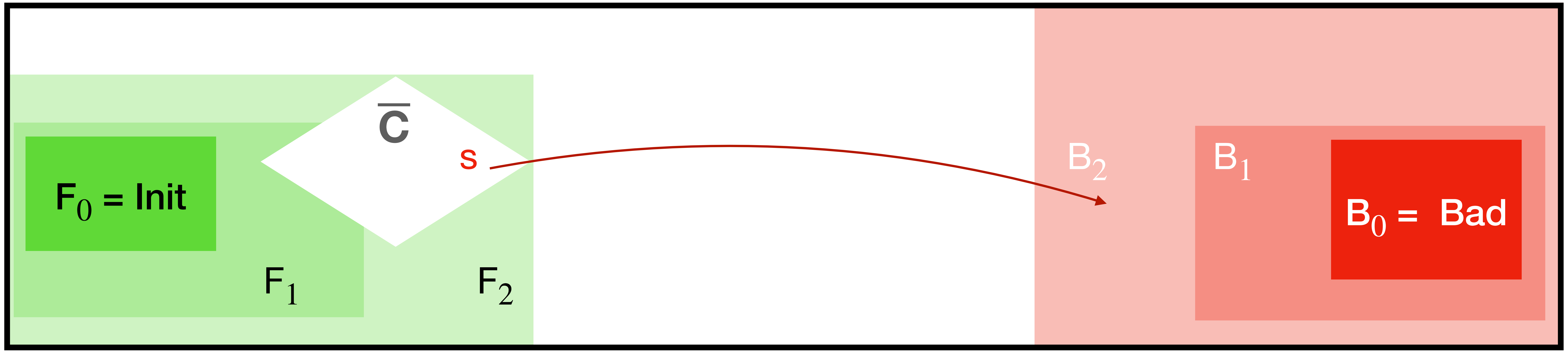


Removing CTIs:

Let $s \in F_k \cap \mathrm{pre}(B_k)$.

IC3 in Detail

[Hasuo et al., CAV'22]



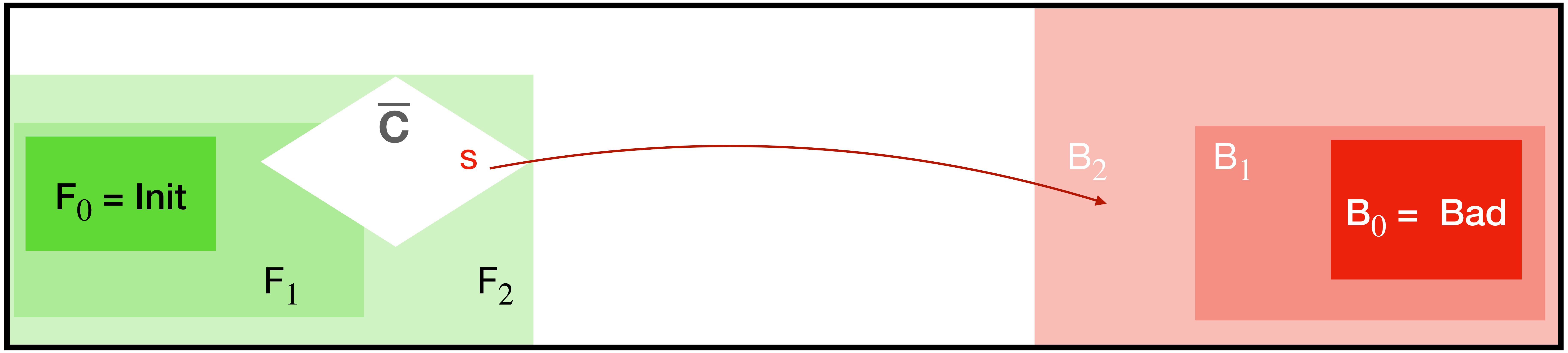
Removing CTIs:

Let $s \in F_k \cap \mathrm{pre}(B_k)$.

Let $C \subseteq \overline{\{s\}}$.

IC3 in Detail

[Hasuo et al., CAV'22]



Removing CTIs:

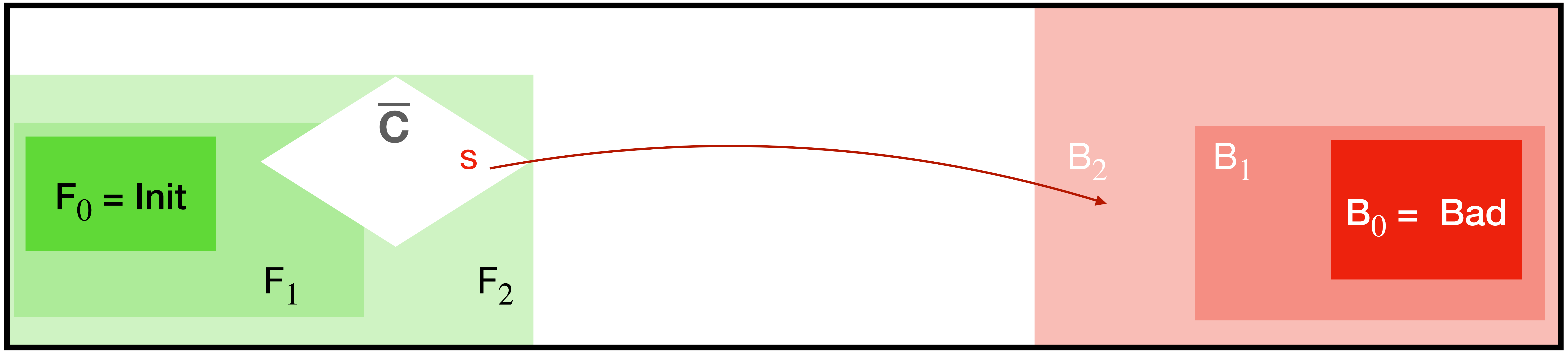
Let $s \in F_k \cap \mathrm{pre}(B_k)$.

Let $C \subseteq \overline{\{s\}}$.

Inductive Clauses

IC3 in Detail

[Hasuo et al., CAV'22]



Removing CTIs:

Let $s \in F_k \cap \mathrm{pre}(B_k)$.

Let $C \subseteq \overline{\{s\}}$.

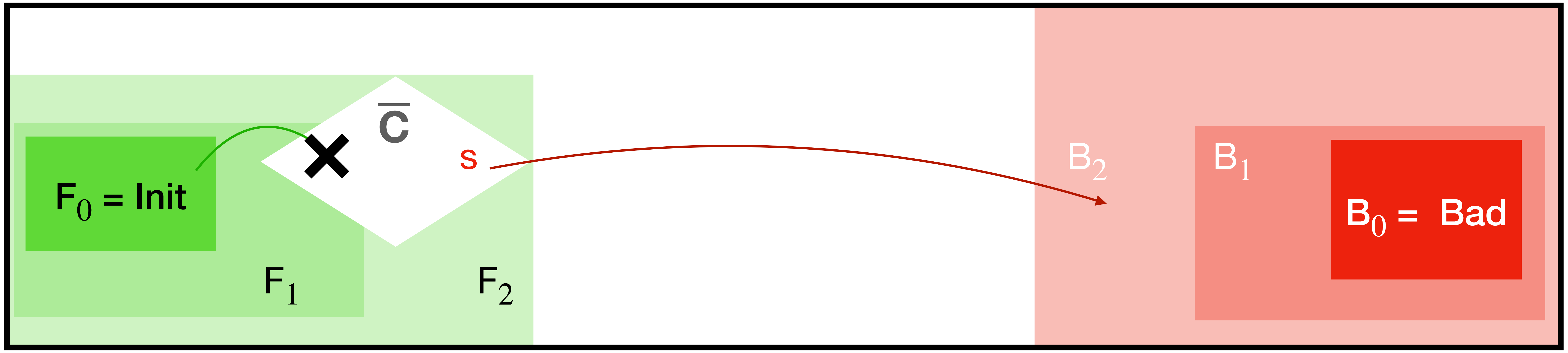
Lemma (Relative Inductiveness):

$$\mathrm{Init} \subseteq C$$

$$\mathrm{post}(F_{k-1} \cap C) \subseteq C$$

IC3 in Detail

[Hasuo et al., CAV'22]



Removing CTIs:

Let $s \in F_k \cap \mathrm{pre}(B_k)$.

Let $C \subseteq \overline{\{s\}}$.

Lemma (Relative Inductiveness):

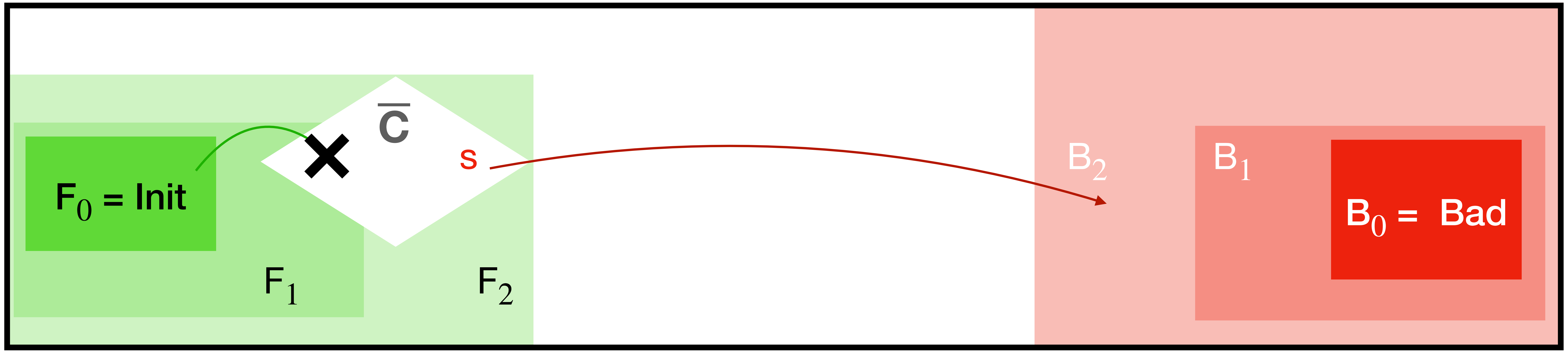
$\mathrm{Init} \subseteq C$

$\mathrm{post}(F_{k-1} \cap C) \subseteq C$

imply $\mathrm{post}(F_i \cap C) \subseteq (F_{i+1} \cap C)$ f.a. i .

IC3 in Detail

[Hasuo et al., CAV'22]



Removing CTIs:

Let $s \in F_k \cap \mathrm{pre}(B_k)$.

Let $C \subseteq \overline{\{s\}}$.

Lemma (Relative Inductiveness):

$$\mathrm{Init} \subseteq C$$

$$\mathrm{post}(F_{k-1} \cap C) \subseteq C$$

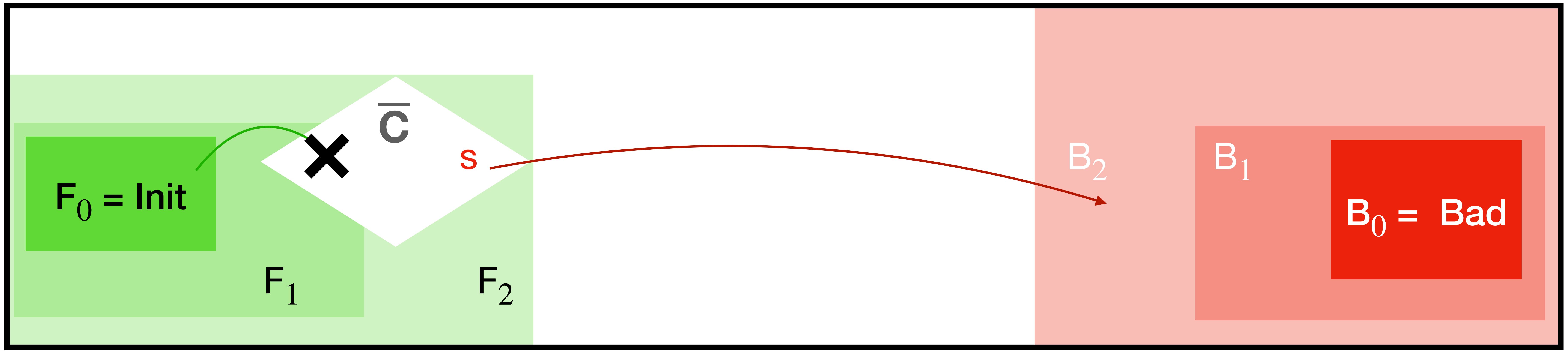
imply $\mathrm{post}(F_i \cap C) \subseteq (F_{i+1} \cap C)$ f.a. i .

$\mathrm{post}(F_i) \subseteq F_{i+1}$ by 2.

$\mathrm{post}(F_i \cap C) \subseteq C$ by monotonicity of post.

IC3 in Detail

[Hasuo et al., CAV'22]



Removing CTIs:

Let $s \in F_k \cap \mathrm{pre}(B_k)$.

Let $C \subseteq \overline{\{s\}}$.

Lemma (Relative Inductiveness):

$\mathrm{Init} \subseteq C$

$\mathrm{post}(F_{k-1} \cap C) \subseteq C$

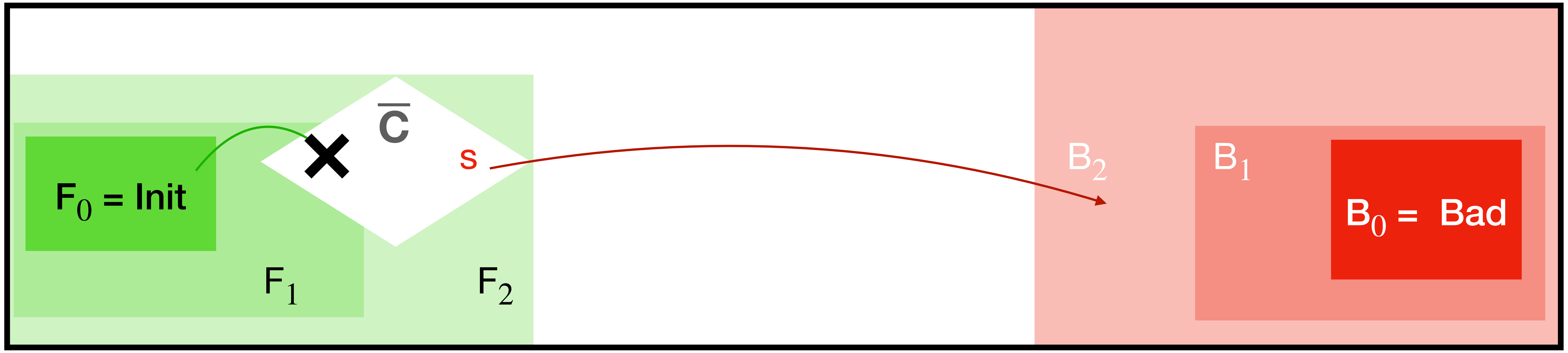
imply $\mathrm{post}(F_i \cap C) \subseteq (F_{i+1} \cap C)$ f.a. i .

With this terminology

F_{i+1} is inductive relative to F_i , by 2.

IC3 in Detail

[Hasuo et al., CAV'22]



Removing CTIs:

Let $s \in F_k \cap \mathrm{pre}(B_k)$.

Let $C \subseteq \overline{\{s\}}$.

Lemma (Relative Inductiveness):

$\mathrm{Init} \subseteq C$

$\mathrm{post}(F_{k-1} \cap C) \subseteq C$

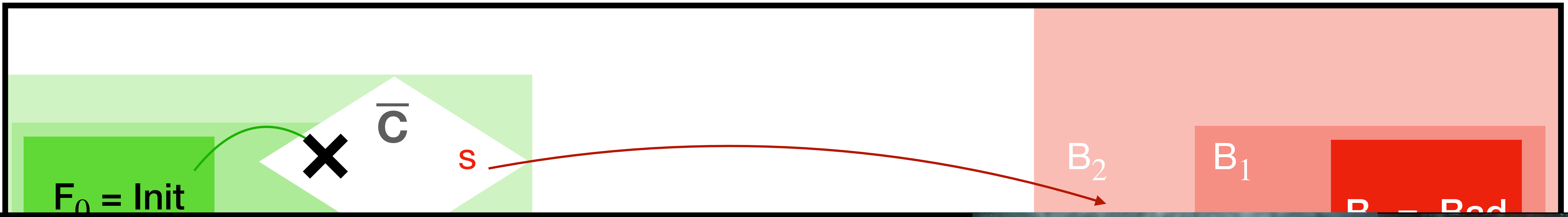
imply $\mathrm{post}(F_i \cap C) \subseteq (F_{i+1} \cap C)$ f.a. i .

Strengthen the sequence of candidate invariants:

$$(F_0 \cap C) \subseteq \ldots \subseteq (F_k \cap C)$$

IC3 in Detail

[Hasuo et al., CAV'22]



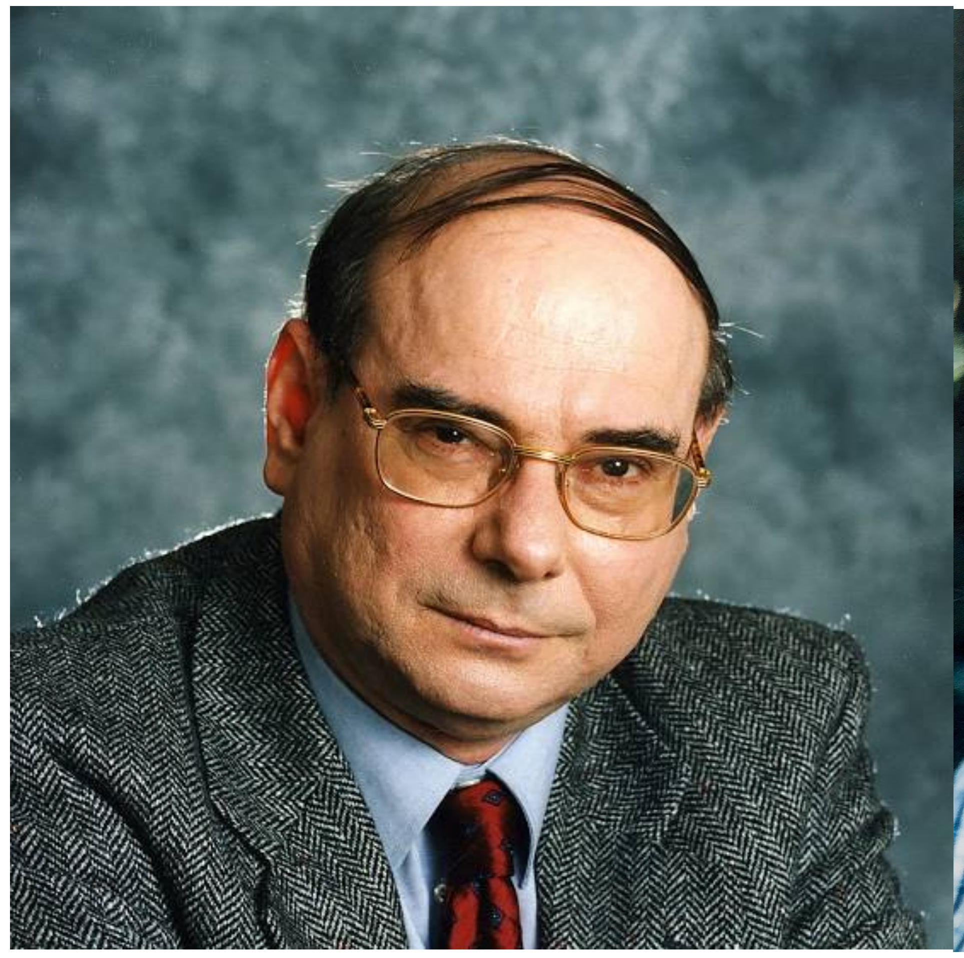
This is an incremental construction of an inductive invariant.
The choice of F_{k+1} is a monolithic step.

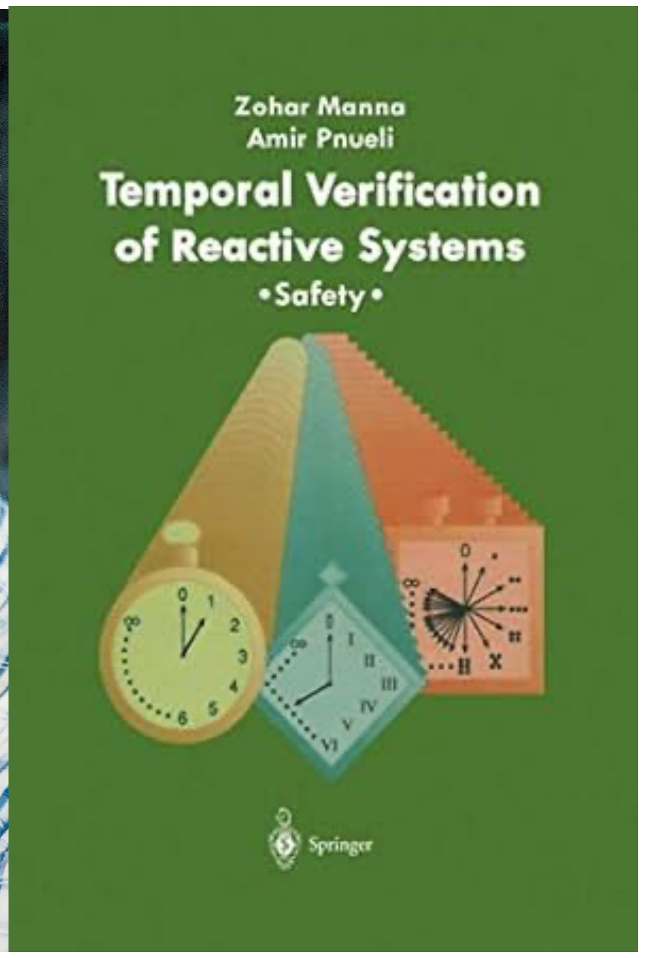
C) f.a. i .

Strengthen the sequence of candidate invariants:

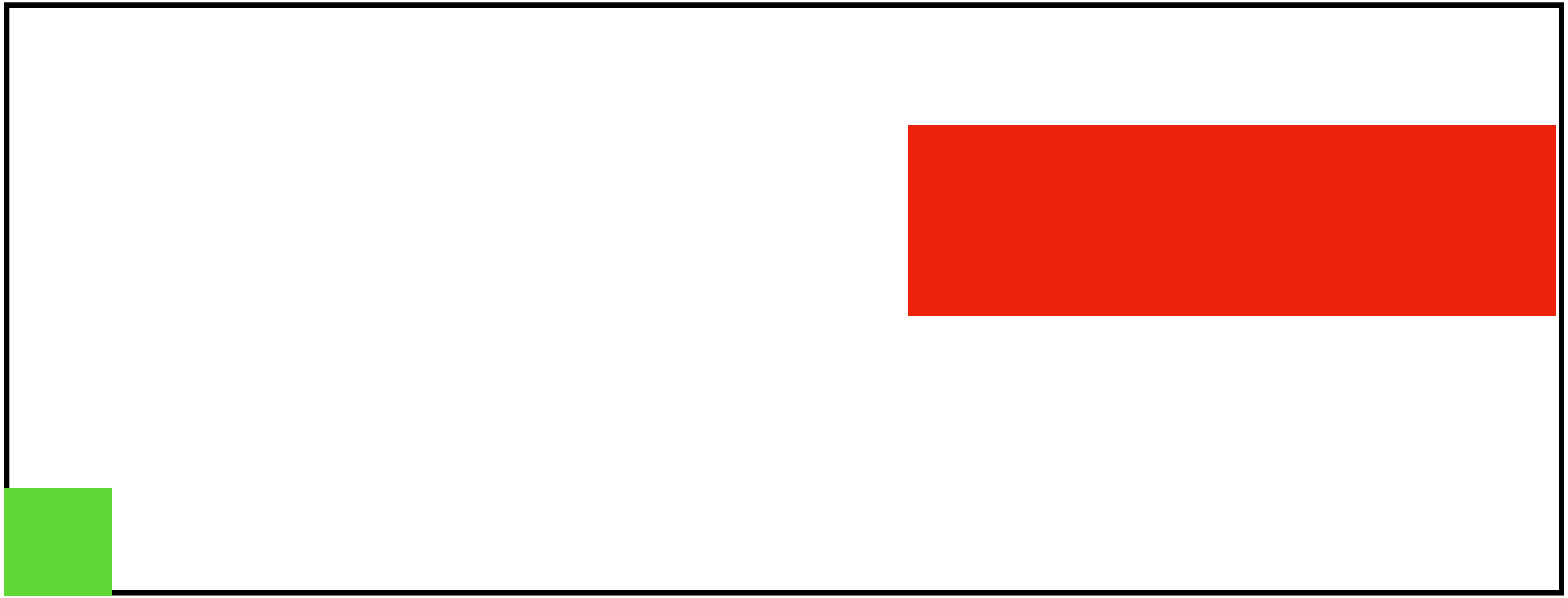
$$(F_0 \cap C) \subseteq \ldots \subseteq (F_k \cap C)$$

Relative Inductiveness



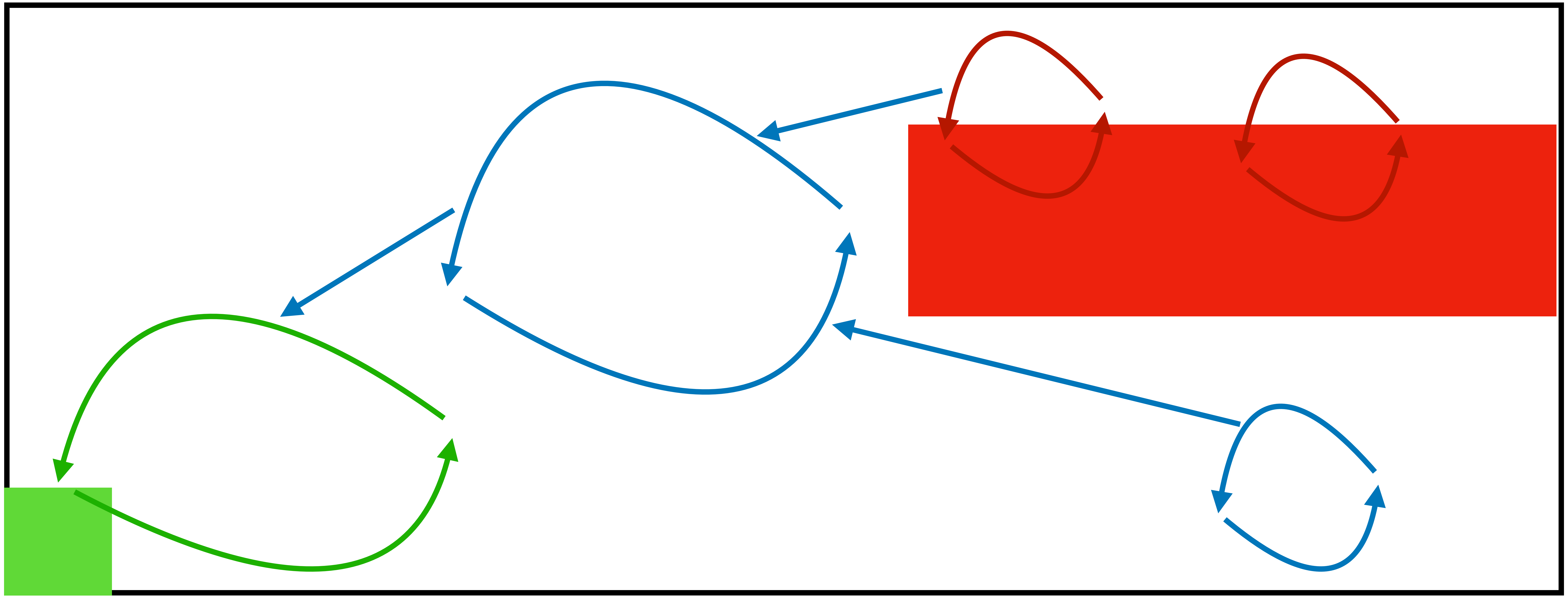


Determine strongly connected components.



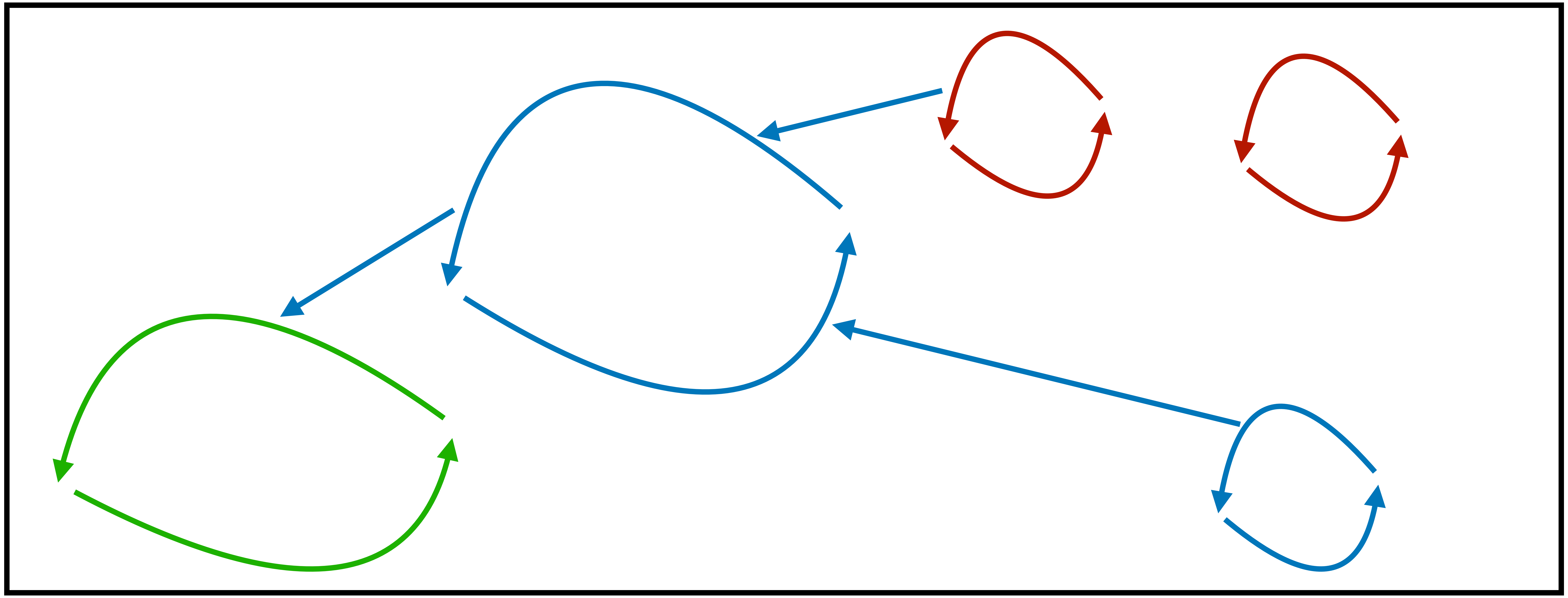
Relative Inductiveness

Determine strongly connected components.



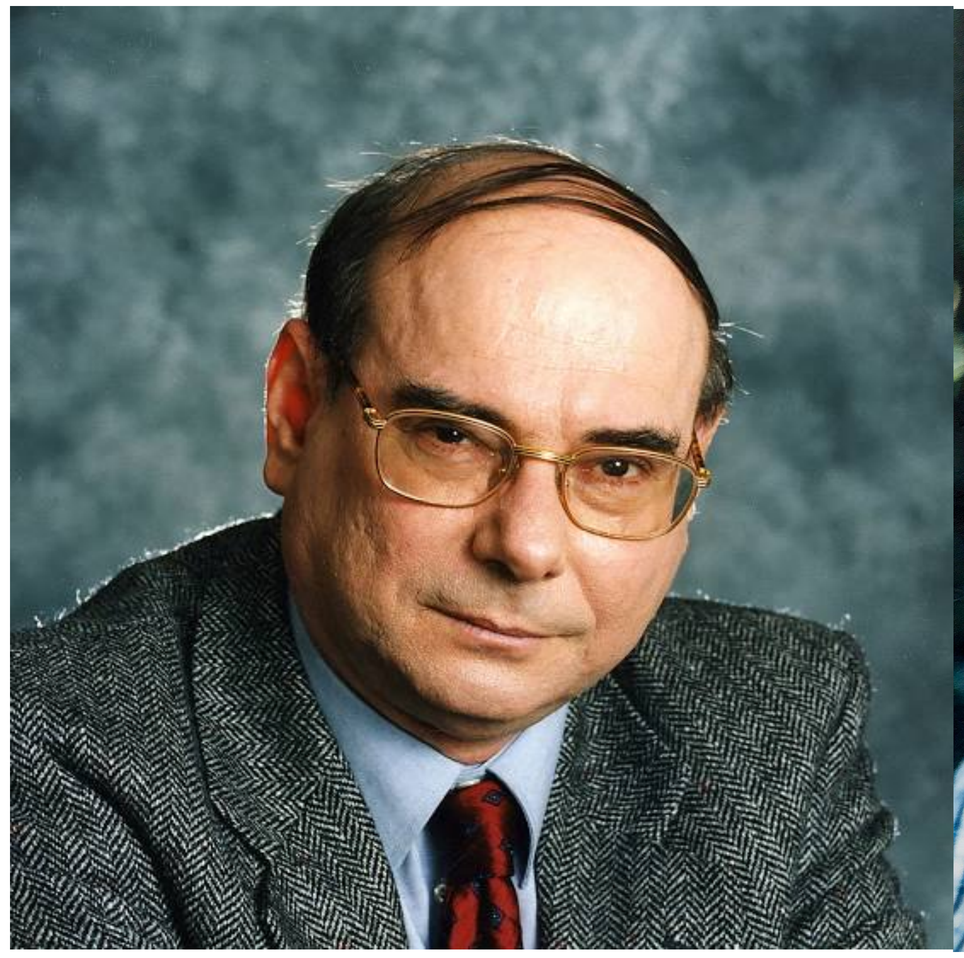
Relative Inductiveness

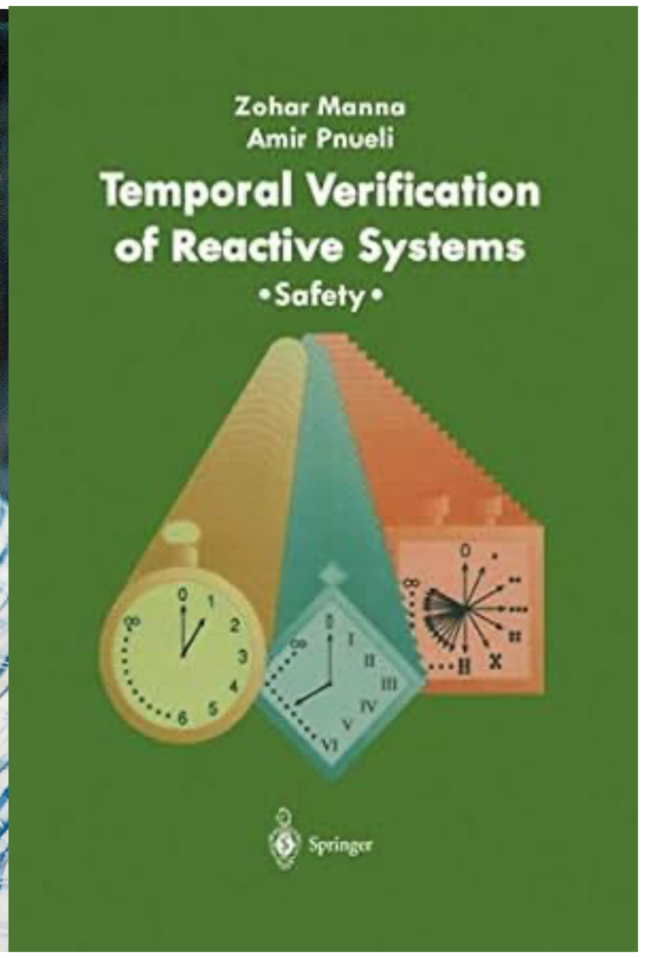
Determine strongly connected components.

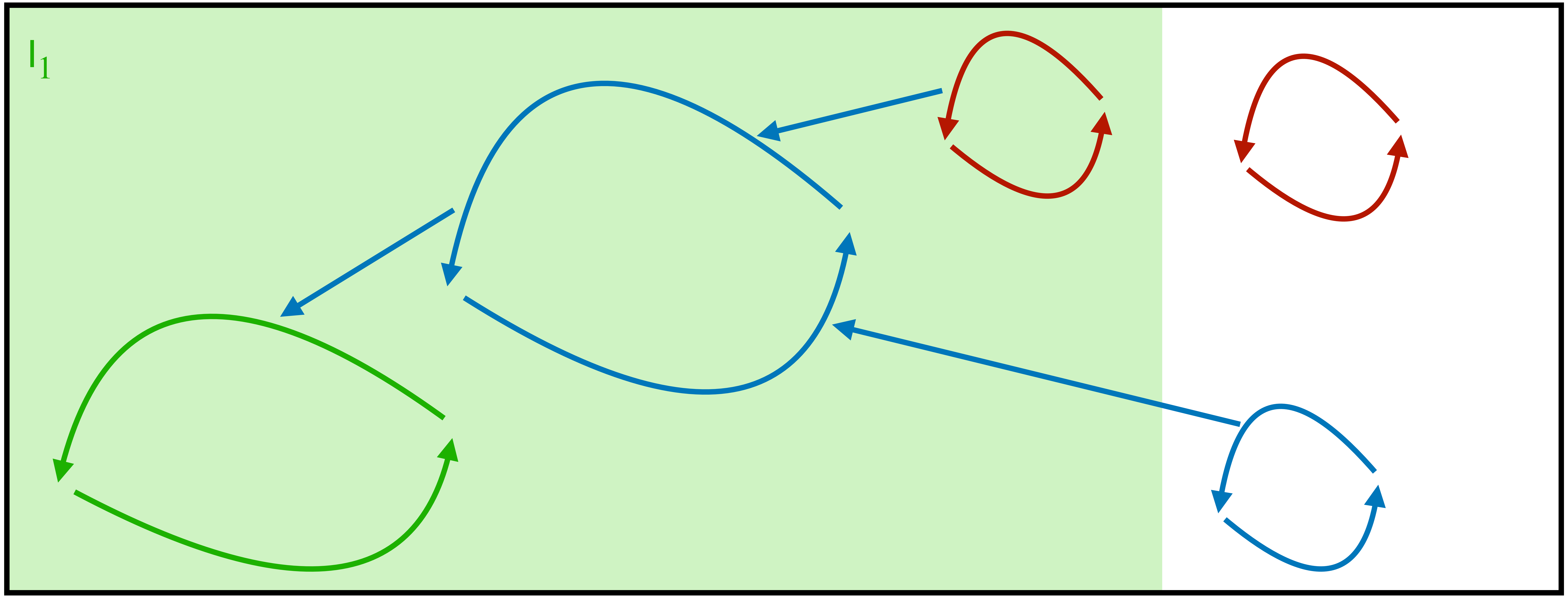


Relative Inductiveness

I_1 is inductive.



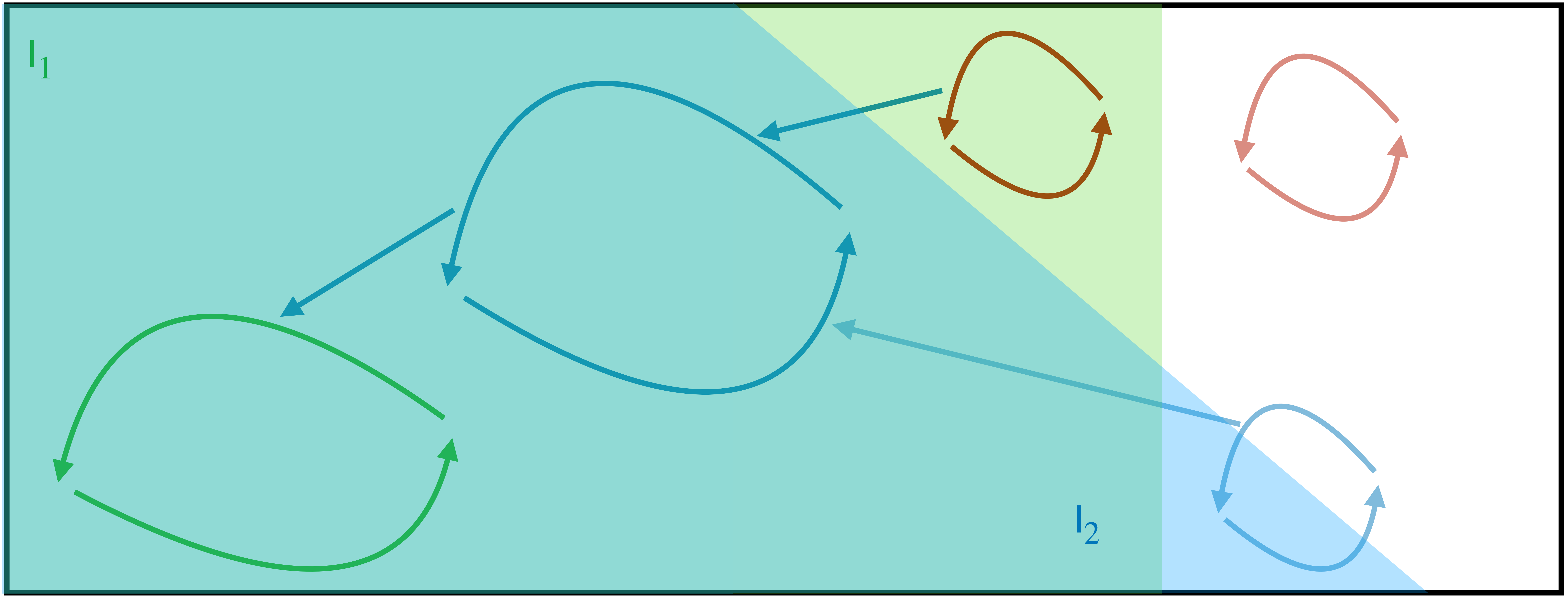




Relative Inductiveness

I_1 is inductive.

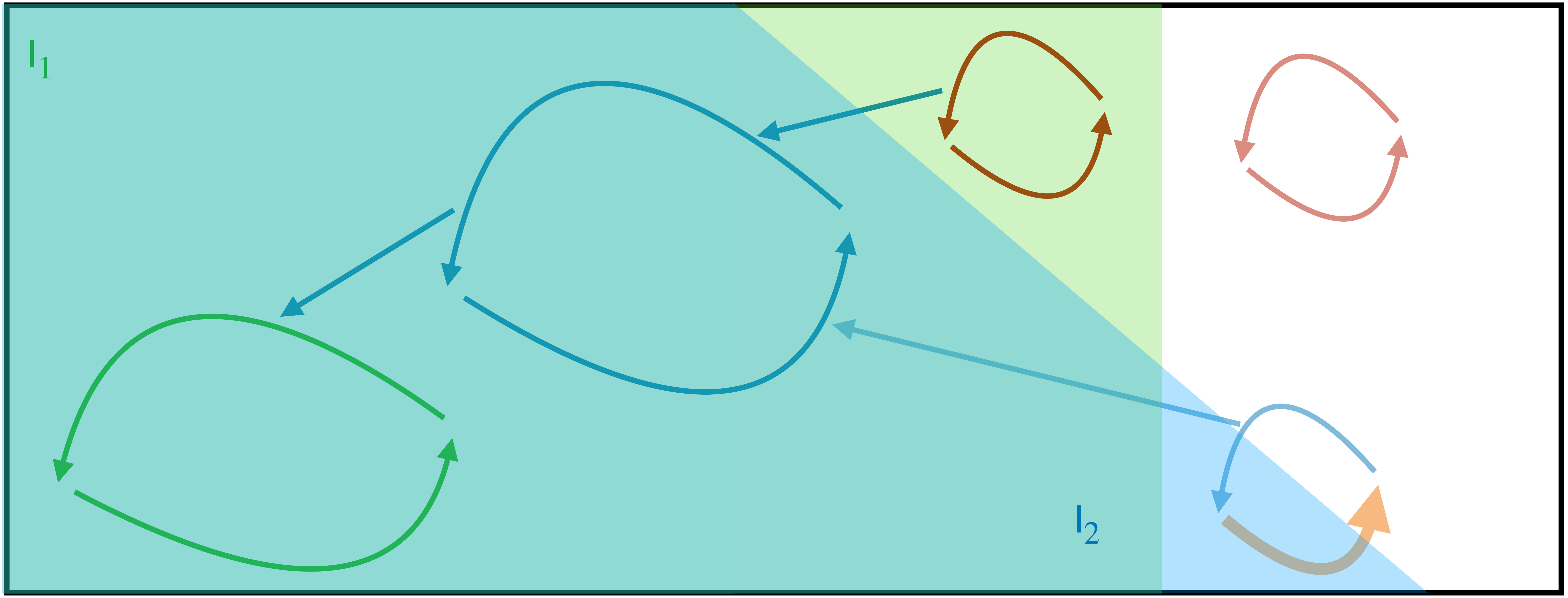
I_2 is not inductive



Relative Inductiveness

I_1 is inductive.

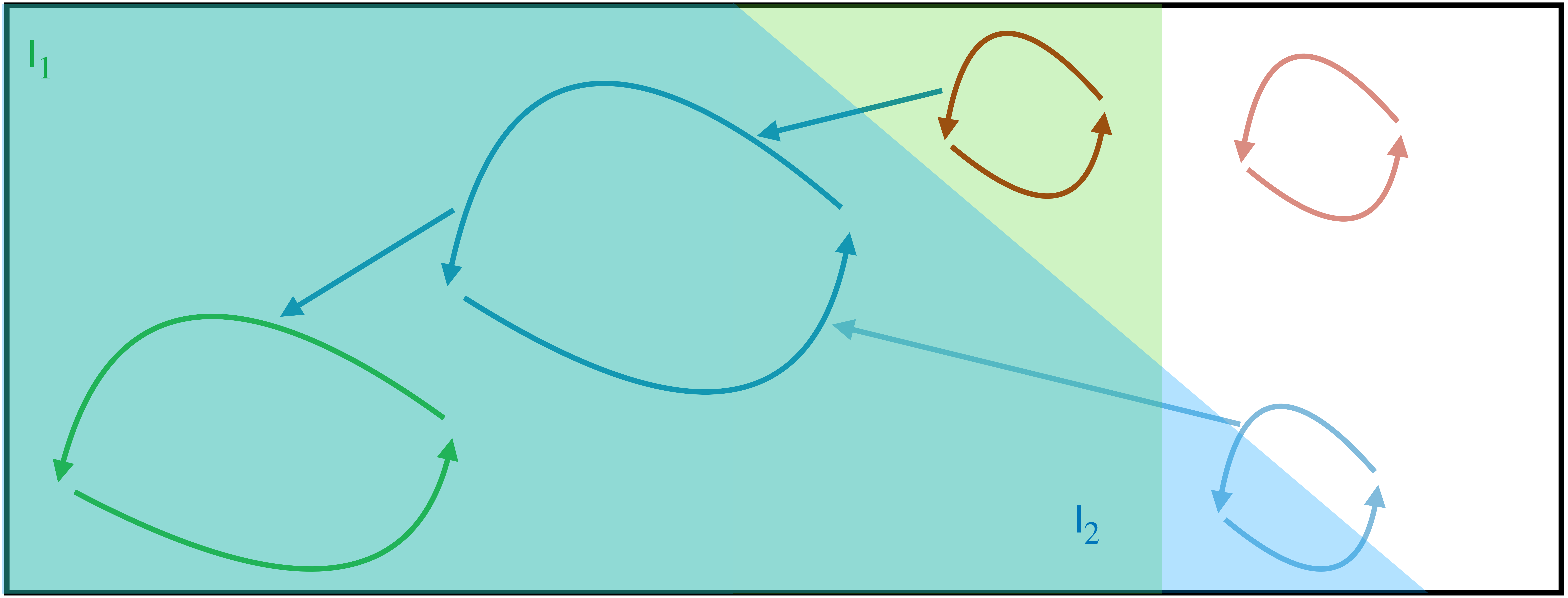
I_2 is not inductive



Relative Inductiveness

I_1 is inductive.

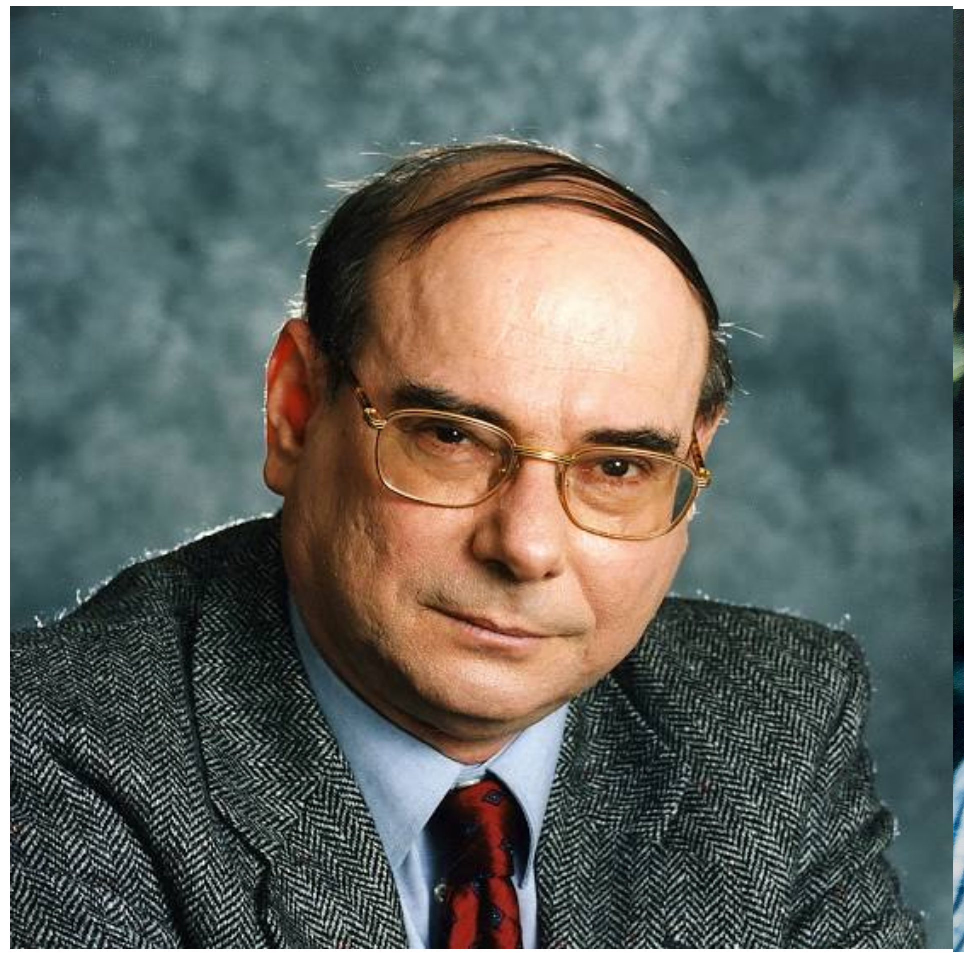
I_2 is not inductive, but inductive relative to I_1.

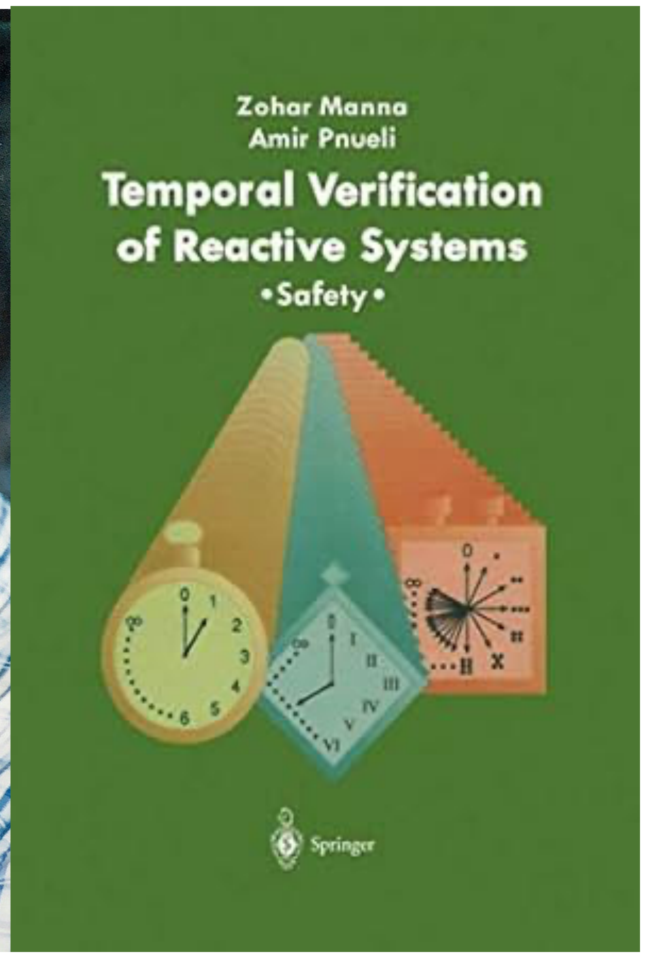


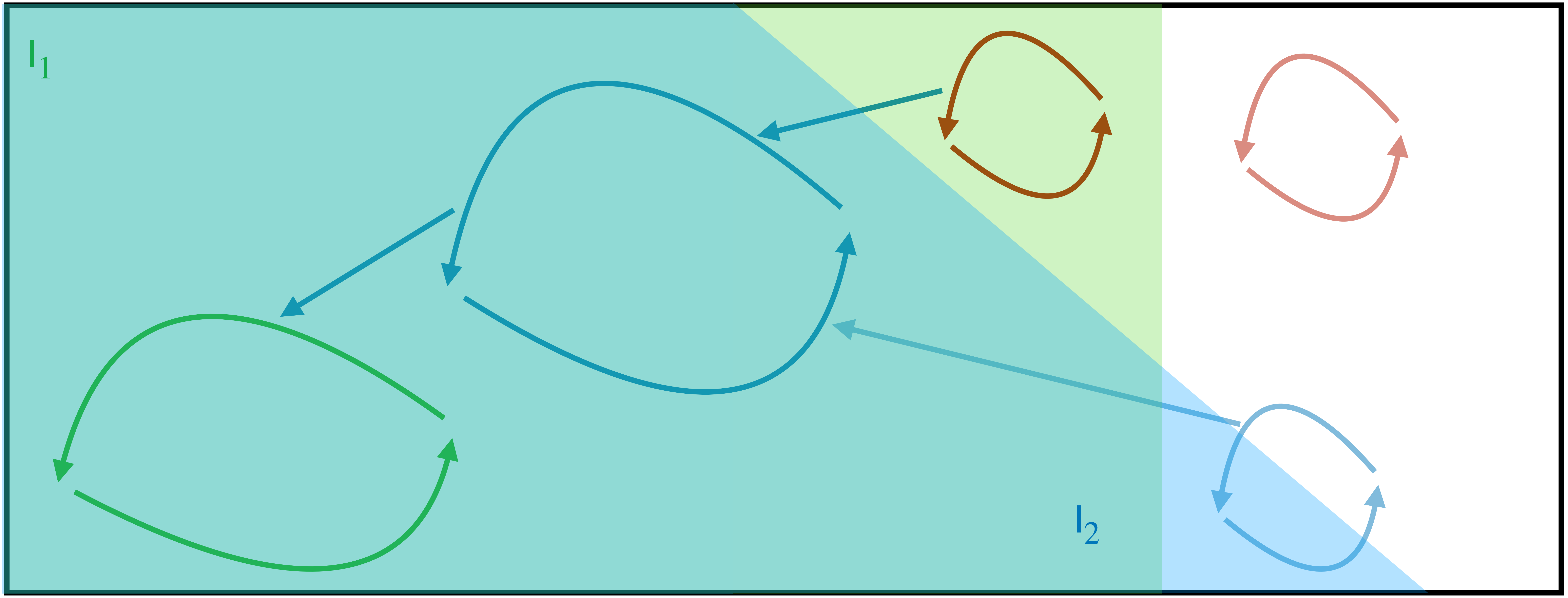
Relative Inductiveness

I_1 is inductive.

I_2 is not inductive, but inductive relative to I_1.







Relative inductiveness (and IC3) iteratively removes unreachable parts of the state space.

IC3 Q&A

Q1. What does all this have to do with circuits?
A1. This is the semantic view and generalizes to other program classes.
Circuits come in when you want to implement (answer relative inductiveness queries with SAT).

Q2. Why is it called property directed?
A2. You take into account initial and bad states.

Q3. Can you do this for nested fixed points?
A3. With a reduction from nested fixed points to safety.

Q4. What is the most recent development?
A4. You learn an invariant with an Angluin's active teacher model.
The ICE algorithm emphasizes this point of view.
Recent papers study the information gain of queries.

Q5. Why does IC3 not work well for parallel programs?
A5. Think about the interference pre.

IC3 Q&A

Q6. How do you pick C?

A6. If post(F_{k-1}) cap $\{s\} = \varnothing$, read C off of an unsat core.

Q7. Do you have to take $F_{k+1} = \overline{B_k}$?

A7. No, you can use any predicate that has a chance to over-approximate post(F_k).