SLAB: A Certifying Model Checker for Infinite-State Concurrent Systems

Klaus Dräger[1], Andrey Kupriyanov[1], Bernd Finkbeiner[1]
and Heike Wehrheim[2]

[1]Saarland University, [2]Paderborn University,
Germany

March 23, 2010

SLAB

SLAB

SLAB (for *SL*icing *AB*stractions) is a Model Checker for:

SLAB

SLAB (for *SL*icing *AB*stractions) is a Model Checker for:

- **infinite state** (real time, infinite datatypes, ...)

SLAB

SLAB (for *SL*icing *AB*stractions) is a Model Checker for:

- **infinite state** (real time, infinite datatypes, ...)
- **concurrent systems** (communication protocols, control systems,...)

SLAB

SLAB (for *SL*icing *AB*stractions) is a Model Checker for:

- **infinite state** (real time, infinite datatypes, ...)
- **concurrent systems** (communication protocols, control systems,...)
- which can produce **certificates** if the system is correct

SLAB

SLAB (for *SL*icing *AB*stractions) is a Model Checker for:

- **infinite state** (real time, infinite datatypes, ...)
- **concurrent systems** (communication protocols, control systems,...)
- which can produce **certificates** if the system is correct

System Description:

SLAB

SLAB (for *SL*icing *AB*stractions) is a Model Checker for:

- **infinite state** (real time, infinite datatypes, ...)
- **concurrent systems** (communication protocols, control systems,...)
- which can produce **certificates** if the system is correct

System Description:

- transition systems described by linear constraints

SLAB

SLAB (for *SL*icing *AB*stractions) is a Model Checker for:

- **infinite state** (real time, infinite datatypes, ...)
- **concurrent systems** (communication protocols, control systems,...)
- which can produce **certificates** if the system is correct

System Description:

- transition systems described by linear constraints
- safety properties

Certification

Certification

Standard practice for model checkers

- Error found $\Longrightarrow$ Counterexample
- Error unreachable $\Longrightarrow$ "Correct"

Certification

Standard practice for model checkers

- Error found $\Longrightarrow$ Counterexample
- Error unreachable $\Longrightarrow$ "Correct"

But model checker is just another piece of software...

- ... it may contain bugs!

Certification

Standard practice for model checkers

- Error found $\Longrightarrow$ Counterexample
- Error unreachable $\Longrightarrow$ "Correct"

But model checker is just another piece of software...

- ... it may contain bugs!

We need a **certificate as a proof** of correctness

- Should be independently and automatically checkable
- Provides higher degree of confidence in the verification results
- Helps understanding the reason of system correctness

SLAB: A Certifying Model Checker

SLAB: A Certifying Model Checker

SLAB

- Error found $\Longrightarrow$ Counterexample
- Error unreachable $\Longrightarrow$ Certificate of correctness

SLAB: A Certifying Model Checker

SLAB

- Error found $\Longrightarrow$ Counterexample
- Error unreachable $\Longrightarrow$ Certificate of correctness

Certificates

- In graphical format
- In SMT-LIB format
- Independently checkable by standard SMT solvers

SLAB: A Certifying Model Checker

SLAB

- Error found $\Longrightarrow$ Counterexample
- Error unreachable $\Longrightarrow$ Certificate of correctness

Certificates

- In graphical format
- In SMT-LIB format
- Independently checkable by standard SMT solvers

Coming soon...

- Integration into theorem proving process
- Interfacing with Coq

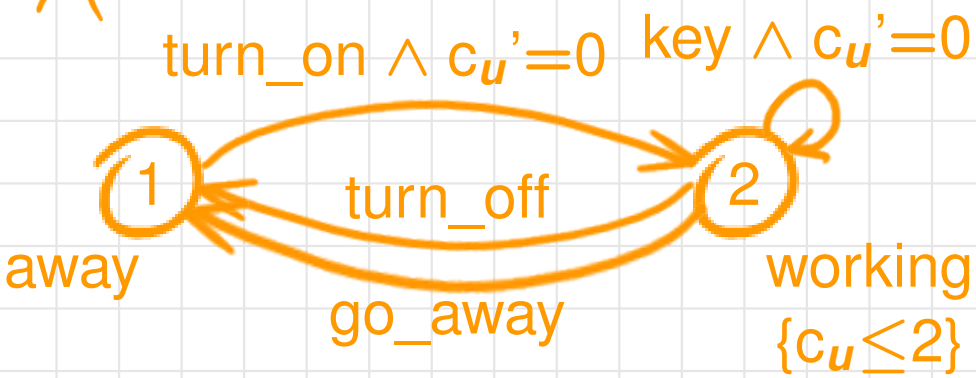
SLAB Certificate Example

SLAB Certificate Example

SLAB Certificate Example

c_u, c_L: clock

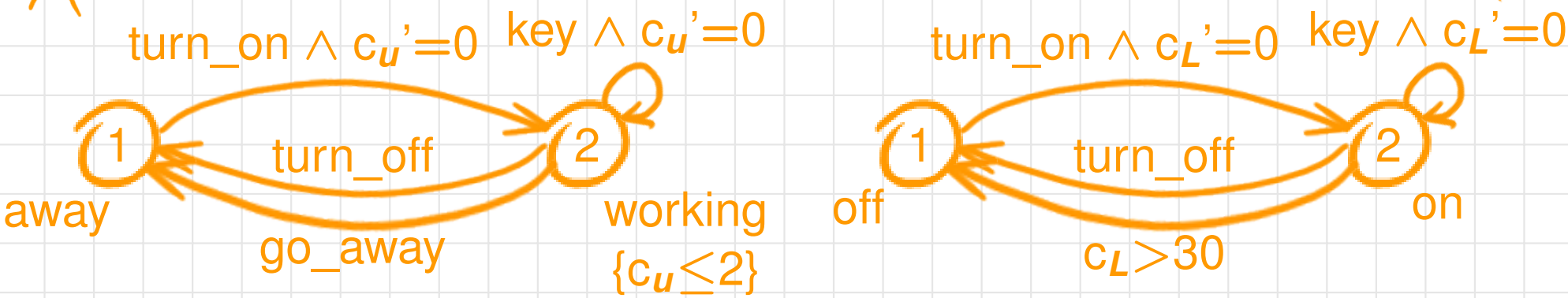
turn_on, key, ... : event



SLAB Certificate Example

c_u, c_L: clock

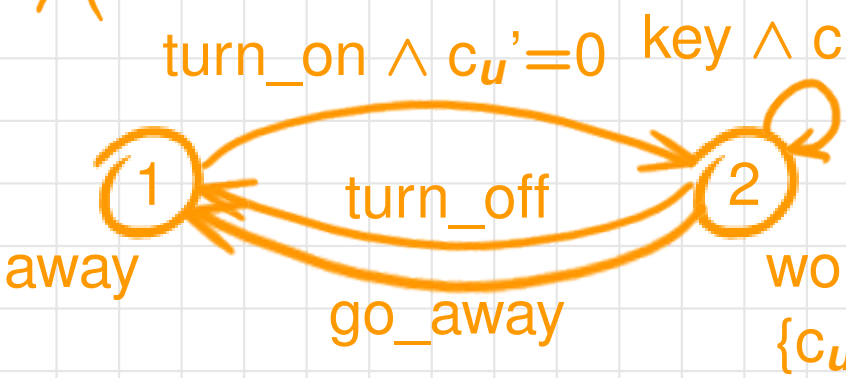
turn_on, key, ... : event

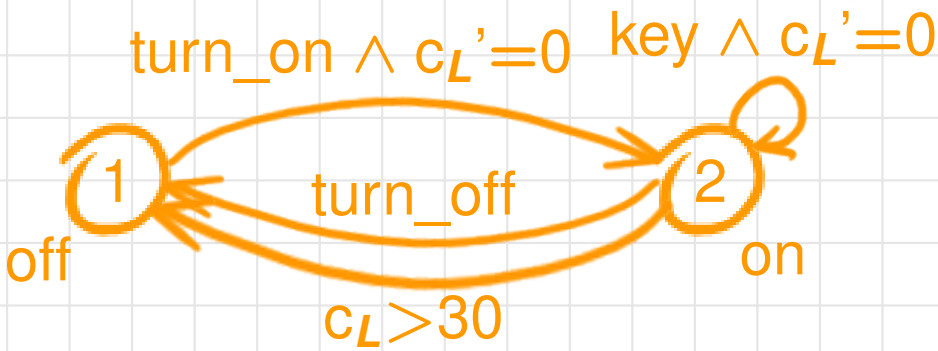


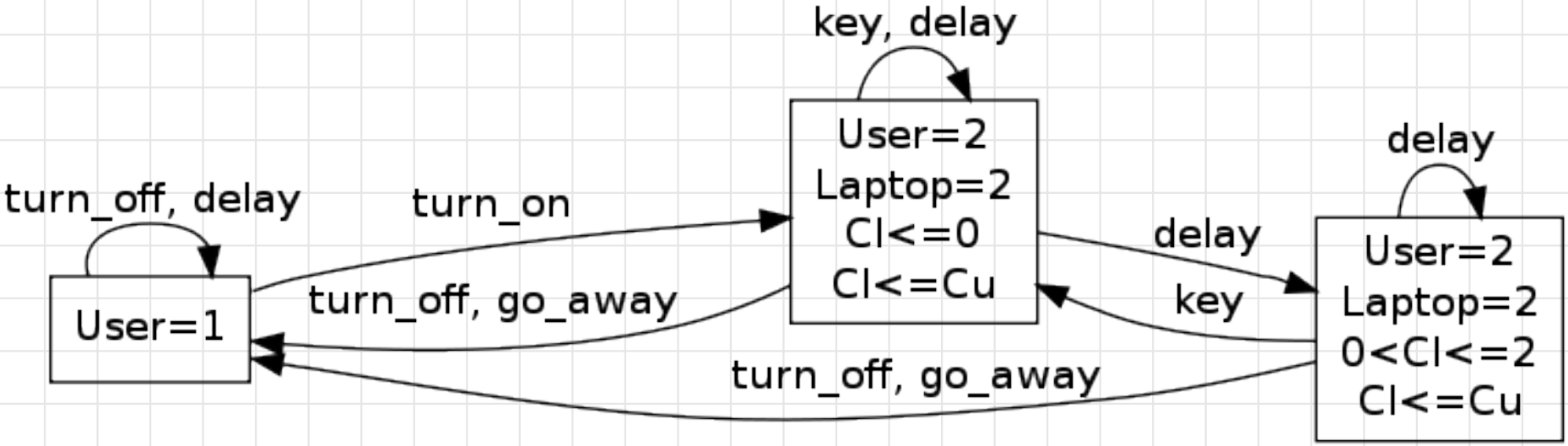
SLAB Certificate Example

c_u, c_L: clock

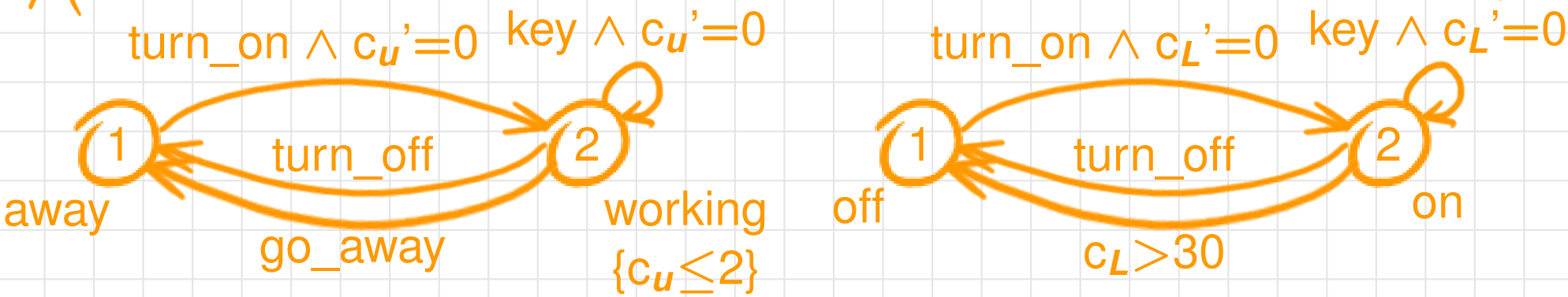
turn_on, key, ... : event



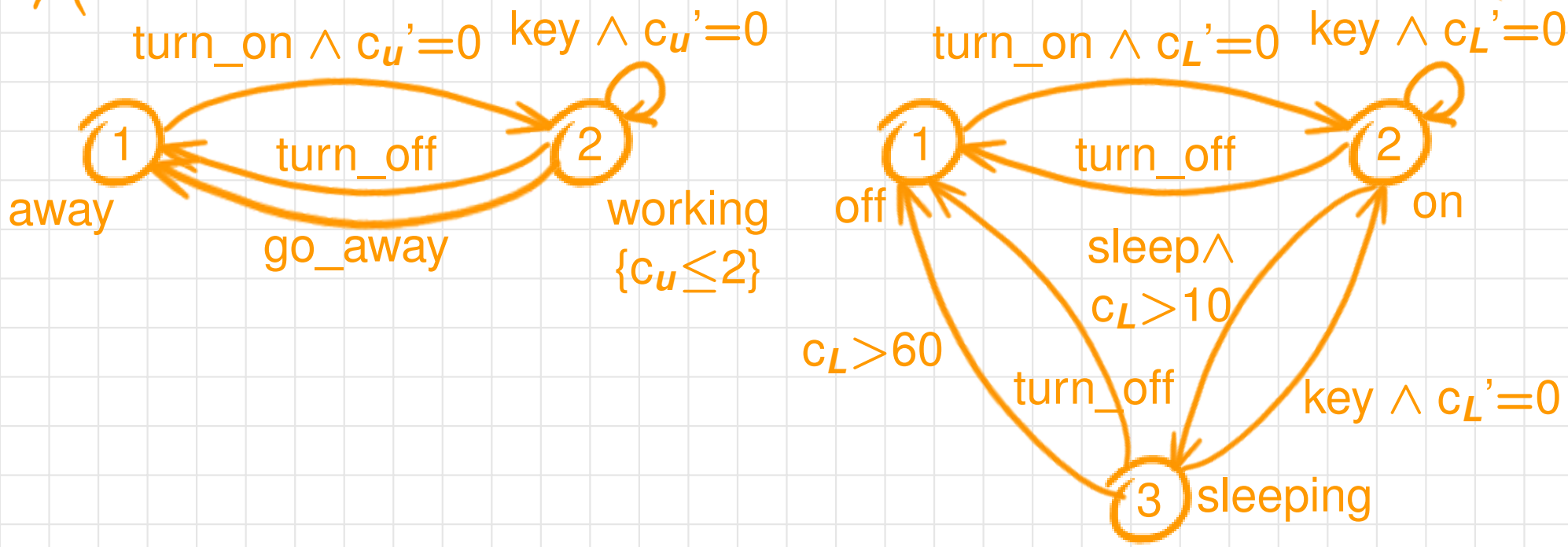




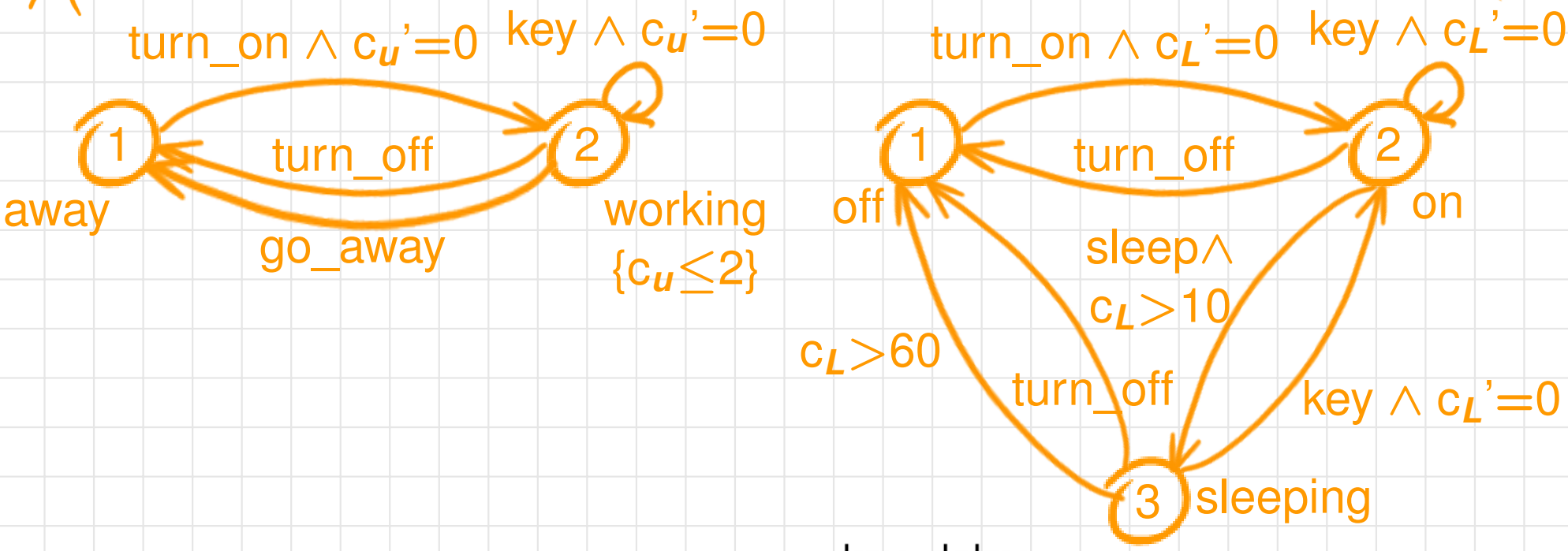
SLAB Certificate Example (2)

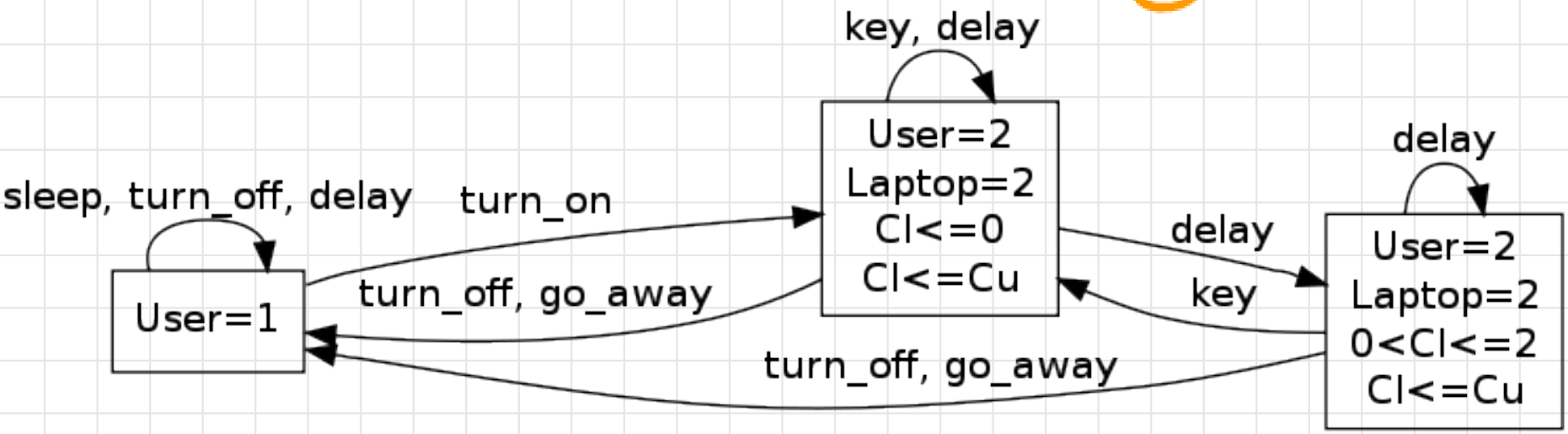


SLAB Certificate Example (2)



SLAB Certificate Example (2)





System and Certificate

Transition System $\mathcal{S} = (\mathit{init}, \mathcal{T}, \mathit{error})$

- Initial condition $\mathit{init}(V)$
- Set of system transitions $\mathcal{T}$: $\tau_i(V, V')$
- Error condition $\mathit{error}(V)$
- init, error, τ_i are linear arithmetic constraints

Certificate $\mathcal{C} = (N, E, \varphi, \eta)$

- Finite graph with nodes N and edges E
- Node labeling: $\varphi : N \rightarrow \mathit{Asrt}(V)$
- Edge labeling: $\eta : E \rightarrow 2^{\mathcal{T}}$

System and Certificate

$x, y : \mathbb{Z}$

$init : x = 0$

Transition System $\mathcal{S} = (init, \mathcal{T}, error)$

- Initial condition $init(V)$
- Set of system transitions $\mathcal{T}$: $\tau_i(V, V')$
- Error condition $error(V)$
- $init$, $error$, τ_i are linear arithmetic constraints

Certificate $\mathcal{C} = (N, E, \varphi, \eta)$

- Finite graph with nodes N and edges E
- Node labeling: $\varphi : N \rightarrow Asrt(V)$
- Edge labeling: $\eta : E \rightarrow 2^{\mathcal{T}}$

System and Certificate

$x, y : \mathbb{Z}$

$init : x = 0$

$\tau_1 : x' = 1 \wedge y' = x + 1$

$\tau_2 : y' = x - 1$

$\tau_3 : x \geq 1 \wedge x' = y$

Transition System $\mathcal{S} = (init, \mathcal{T}, error)$

- Initial condition $init(V)$
- Set of system transitions $\mathcal{T}$: $\tau_i(V, V')$
- Error condition $error(V)$
- $init$, $error$, τ_i are linear arithmetic constraints

Certificate $\mathcal{C} = (N, E, \varphi, \eta)$

- Finite graph with nodes N and edges E
- Node labeling: $\varphi : N \rightarrow Asrt(V)$
- Edge labeling: $\eta : E \rightarrow 2^{\mathcal{T}}$

System and Certificate

Transition System $\mathcal{S} = (init, \mathcal{T}, error)$

- Initial condition $init(V)$
- Set of system transitions $\mathcal{T}$: $\tau_i(V, V')$
- Error condition $error(V)$
- $init$, $error$, τ_i are linear arithmetic constraints

$$x, y : \mathbb{Z}$$

$$init : x = 0$$

$$\tau_1 : x' = 1 \wedge y' = x + 1$$

$$\tau_2 : y' = x - 1$$

$$\tau_3 : x \geq 1 \wedge x' = y$$

$$error : x < 0$$

Certificate $\mathcal{C} = (N, E, \varphi, \eta)$

- Finite graph with nodes N and edges E
- Node labeling: $\varphi : N \rightarrow Asrt(V)$
- Edge labeling: $\eta : E \rightarrow 2^{\mathcal{T}}$

System and Certificate

Transition System $\mathcal{S} = (init, \mathcal{T}, error)$

- Initial condition $init(V)$
- Set of system transitions $\mathcal{T}$: $\tau_i(V, V')$
- Error condition $error(V)$
- $init$, $error$, τ_i are linear arithmetic constraints

$x, y : \mathbb{Z}$

$init : x = 0$

$\tau_1 : x' = 1 \wedge y' = x + 1$

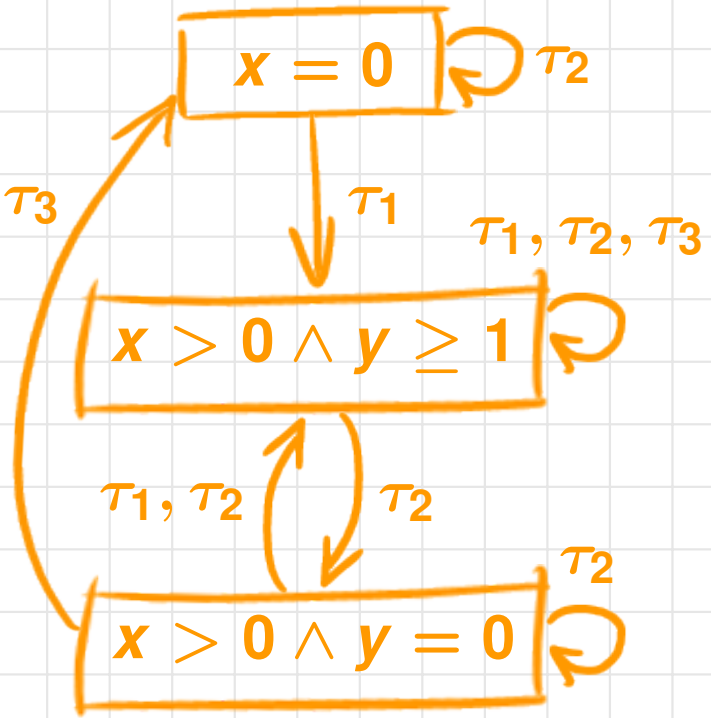
$\tau_2 : y' = x - 1$

$\tau_3 : x \geq 1 \wedge x' = y$

$error : x < 0$

Certificate $\mathcal{C} = (N, E, \varphi, \eta)$

- Finite graph with nodes N and edges E
- Node labeling: $\varphi : N \rightarrow Asrt(V)$
- Edge labeling: $\eta : E \rightarrow 2^{\mathcal{T}}$



Certificate Correctness

$$x, y : \mathbb{Z}$$

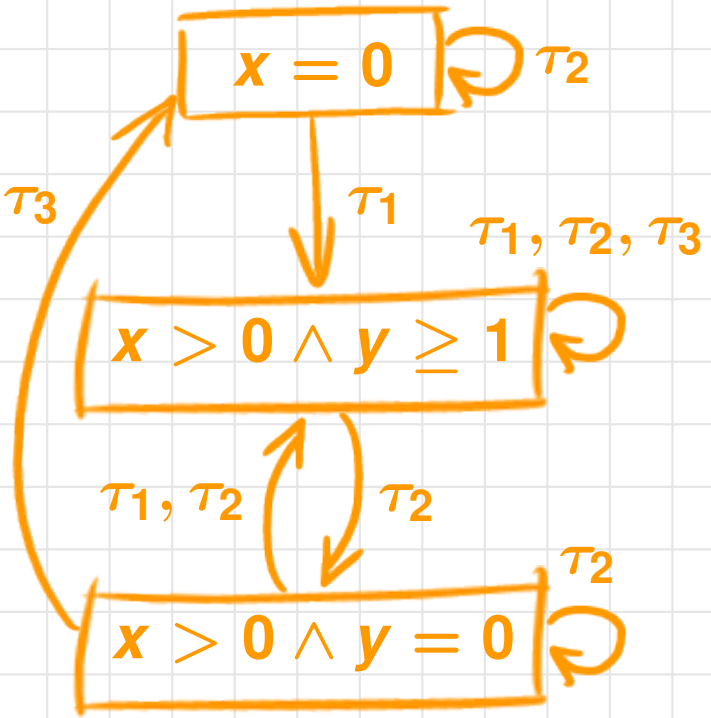
$$\mathit{init} : x = 0$$

$$\tau_1 : x' = 1 \wedge y' = x + 1$$

$$\tau_2 : y' = x - 1$$

$$\tau_3 : x \geq 1 \wedge x' = y$$

$$\mathit{error} : x < 0$$



Certificate Correctness

- $\Phi = \bigvee_{n \in N} \varphi_n$

$x, y : \mathbb{Z}$

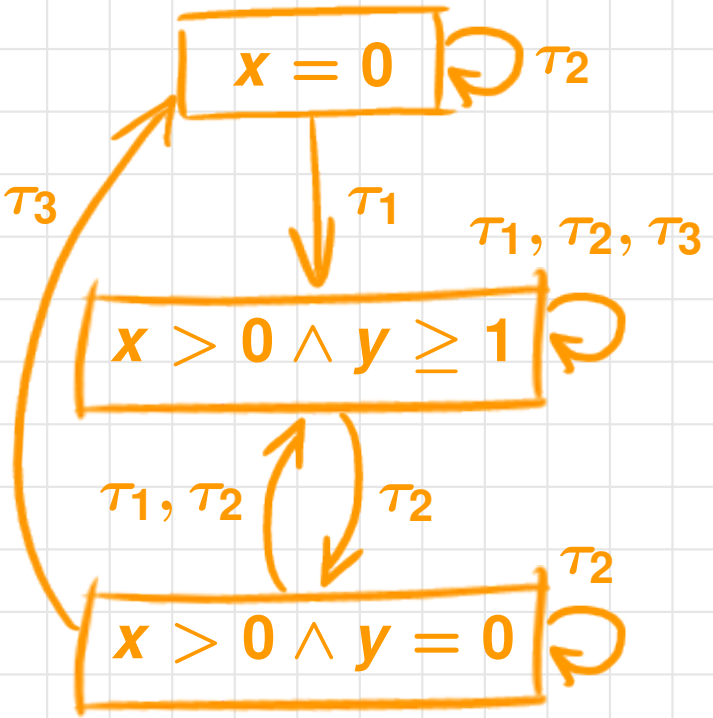
$init : x = 0$

$\tau_1 : x' = 1 \wedge y' = x + 1$

$\tau_2 : y' = x - 1$

$\tau_3 : x \geq 1 \wedge x' = y$

$error : x < 0$



Certificate Correctness

- $\Phi = \bigvee_{n \in N} \varphi_n$

Verification Conditions

$x, y : \mathbb{Z}$

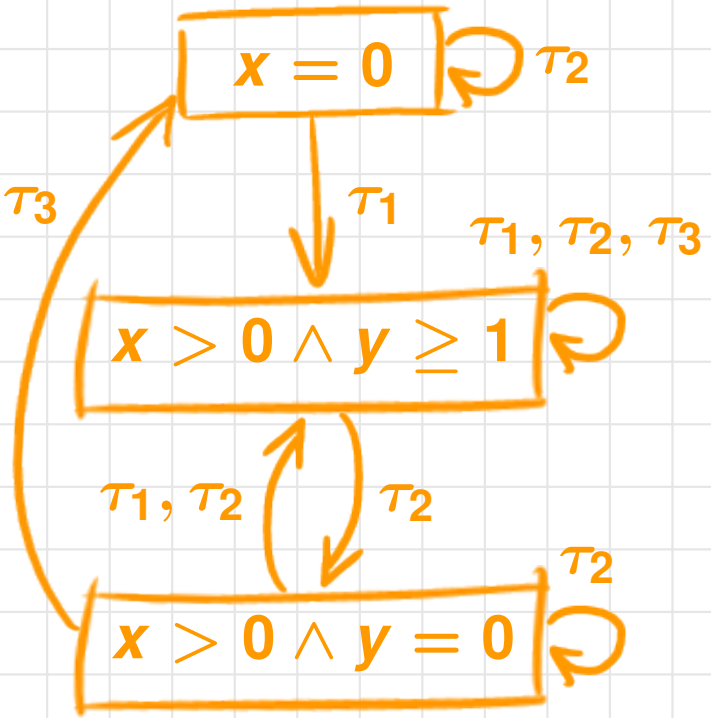
$init : x = 0$

$\tau_1 : x' = 1 \wedge y' = x + 1$

$\tau_2 : y' = x - 1$

$\tau_3 : x \geq 1 \wedge x' = y$

$error : x < 0$



Certificate Correctness

- $\Phi = \bigvee_{n \in N} \varphi_n$

Verification Conditions

- Initiation:
  $init \Rightarrow \Phi$

$x, y : \mathbb{Z}$

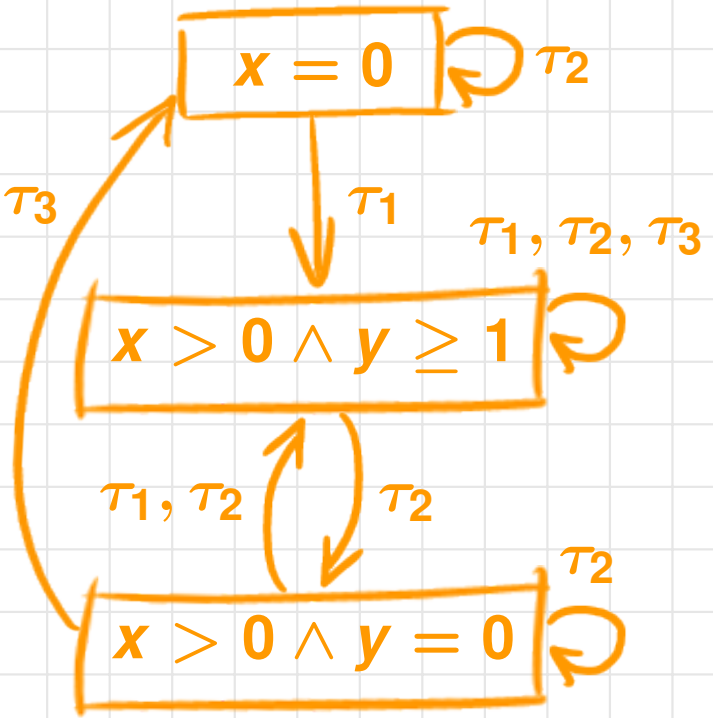
$init : x = 0$

$\tau_1 : x' = 1 \wedge y' = x + 1$

$\tau_2 : y' = x - 1$

$\tau_3 : x \geq 1 \wedge x' = y$

$error : x < 0$



Certificate Correctness

$x, y : \mathbb{Z}$

$init : x = 0$

$\tau_1 : x' = 1 \wedge y' = x + 1$

$\tau_2 : y' = x - 1$

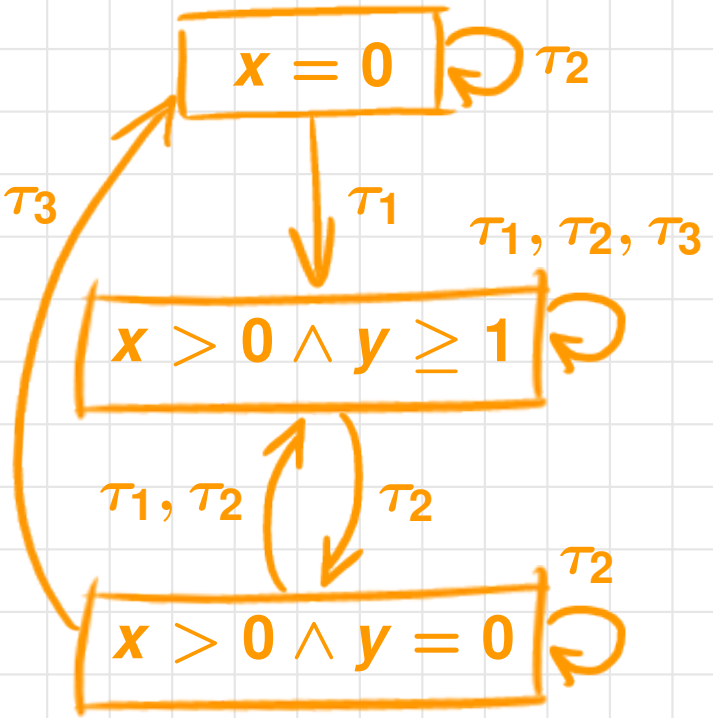
$\tau_3 : x \geq 1 \wedge x' = y$

$error : x < 0$

- $\Phi = \bigvee_{n \in N} \varphi_n$

Verification Conditions

- Initiation:
  $init \Rightarrow \Phi$
- Consecution:
  $\forall n \in N, \tau \in \mathcal{T} : \{\varphi_n\}\ \tau\ \{\bigvee_{n \xrightarrow{\tau} m} \varphi_m\}$



Certificate Correctness

- $\Phi = \bigvee_{n \in N} \varphi_n$

Verification Conditions

- Initiation:
  $init \Rightarrow \Phi$
- Consecution:
  $\forall n \in N, \tau \in \mathcal{T} : \{\varphi_n\}\ \tau\ \{\bigvee_{n \xrightarrow{\tau} m} \varphi_m\}$
  $\Rightarrow \forall \tau \in \mathcal{T} : \{\Phi\}\ \tau\ \{\Phi\}$

$x, y : \mathbb{Z}$

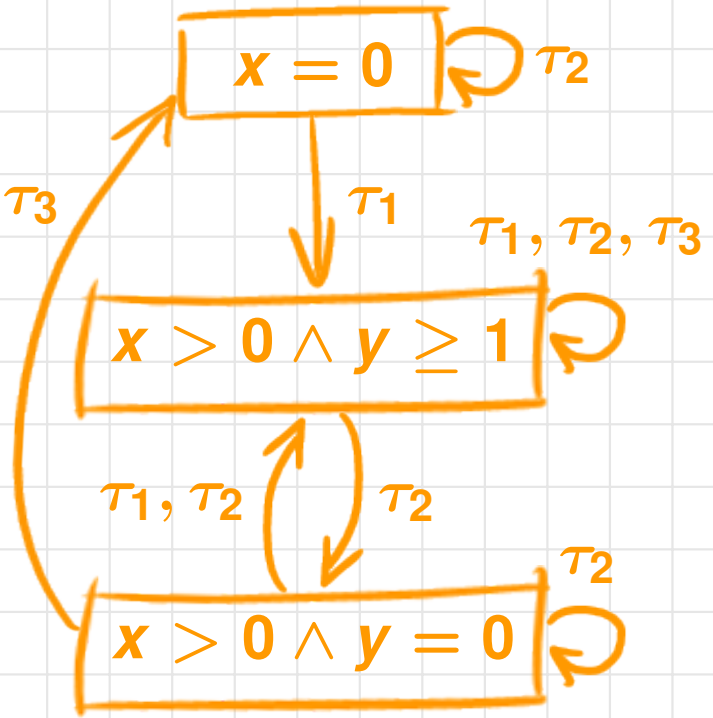
$init : x = 0$

$\tau_1 : x' = 1 \wedge y' = x + 1$

$\tau_2 : y' = x - 1$

$\tau_3 : x \geq 1 \wedge x' = y$

$error : x < 0$



Certificate Correctness

$x, y : \mathbb{Z}$

$init : x = 0$

$\tau_1 : x' = 1 \wedge y' = x + 1$

$\tau_2 : y' = x - 1$

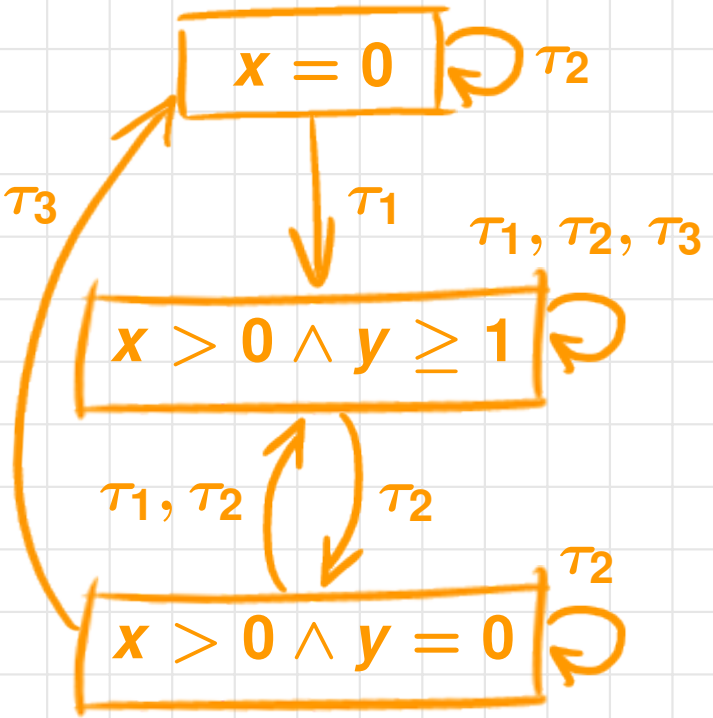
$\tau_3 : x \geq 1 \wedge x' = y$

$error : x < 0$

- $\Phi = \bigvee_{n \in N} \varphi_n$

Verification Conditions

- Initiation:
  $init \Rightarrow \Phi$
- Consecution:
  $\forall n \in N, \tau \in \mathcal{T} : \{\varphi_n\}\ \tau\ \{\bigvee_{n \xrightarrow{\tau} m} \varphi_m\}$
  $\Rightarrow \forall \tau \in \mathcal{T} : \{\Phi\}\ \tau\ \{\Phi\}$
- Error exclusion:
  $\Phi \Rightarrow \neg error$



Certificate Correctness

Inductive Invariant

- $\Phi = \bigvee_{n \in N} \varphi_n$

Verification Conditions

- Initiation:
  $init \Rightarrow \Phi$
- Consecution:
  $\forall n \in N, \tau \in \mathcal{T} : \{\varphi_n\}\ \tau\ \{\bigvee_{n \xrightarrow{\tau} m} \varphi_m\}$
  $\Rightarrow \forall \tau \in \mathcal{T} : \{\Phi\}\ \tau\ \{\Phi\}$
- Error exclusion:
  $\Phi \Rightarrow \neg error$

$x, y : \mathbb{Z}$

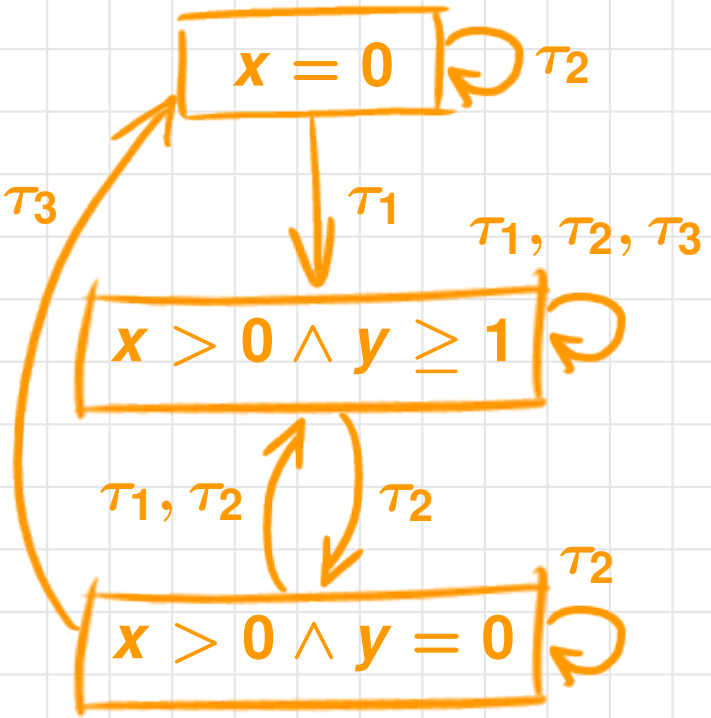
$init : x = 0$

$\tau_1 : x' = 1 \wedge y' = x + 1$

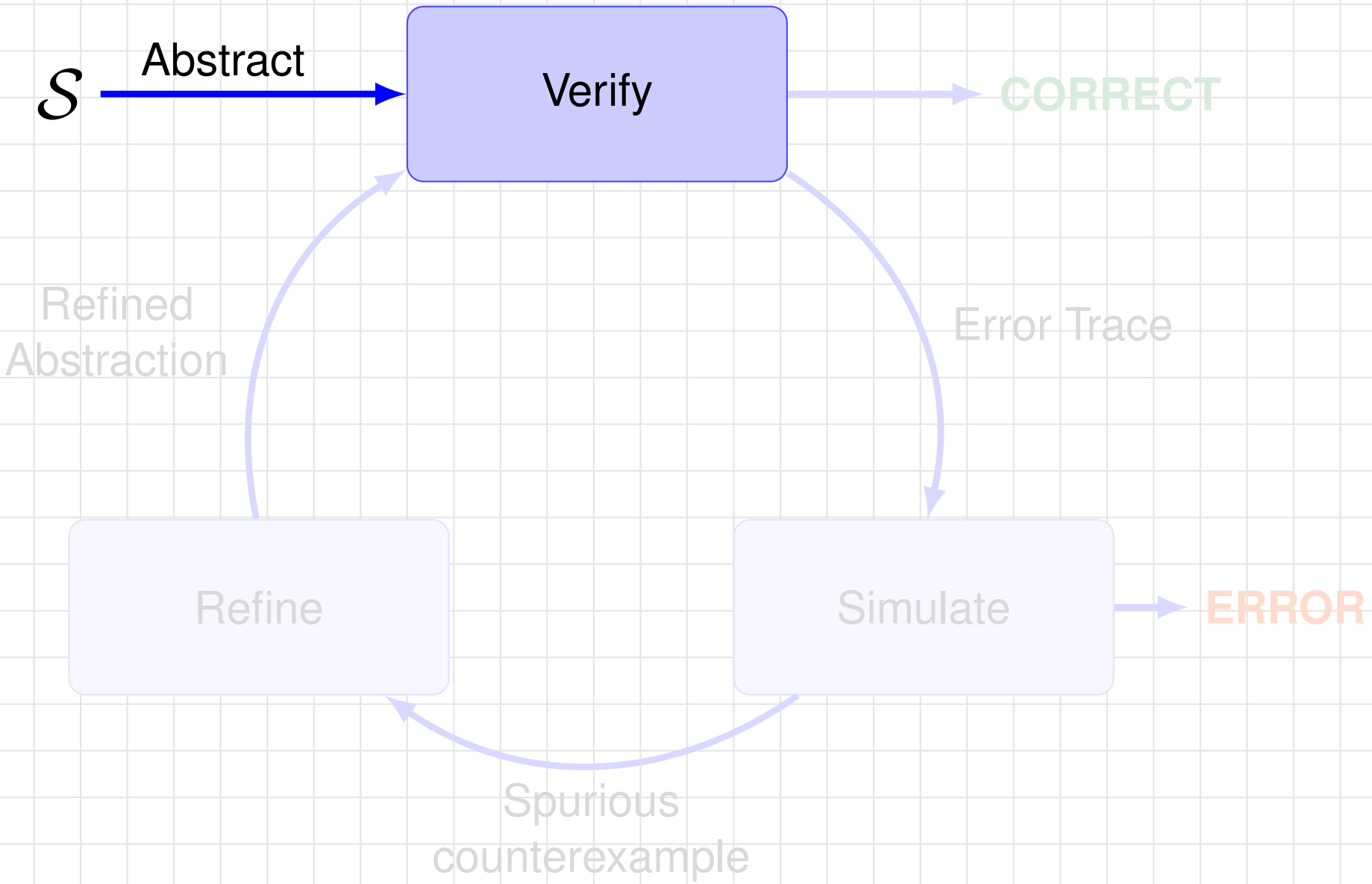
$\tau_2 : y' = x - 1$

$\tau_3 : x \geq 1 \wedge x' = y$

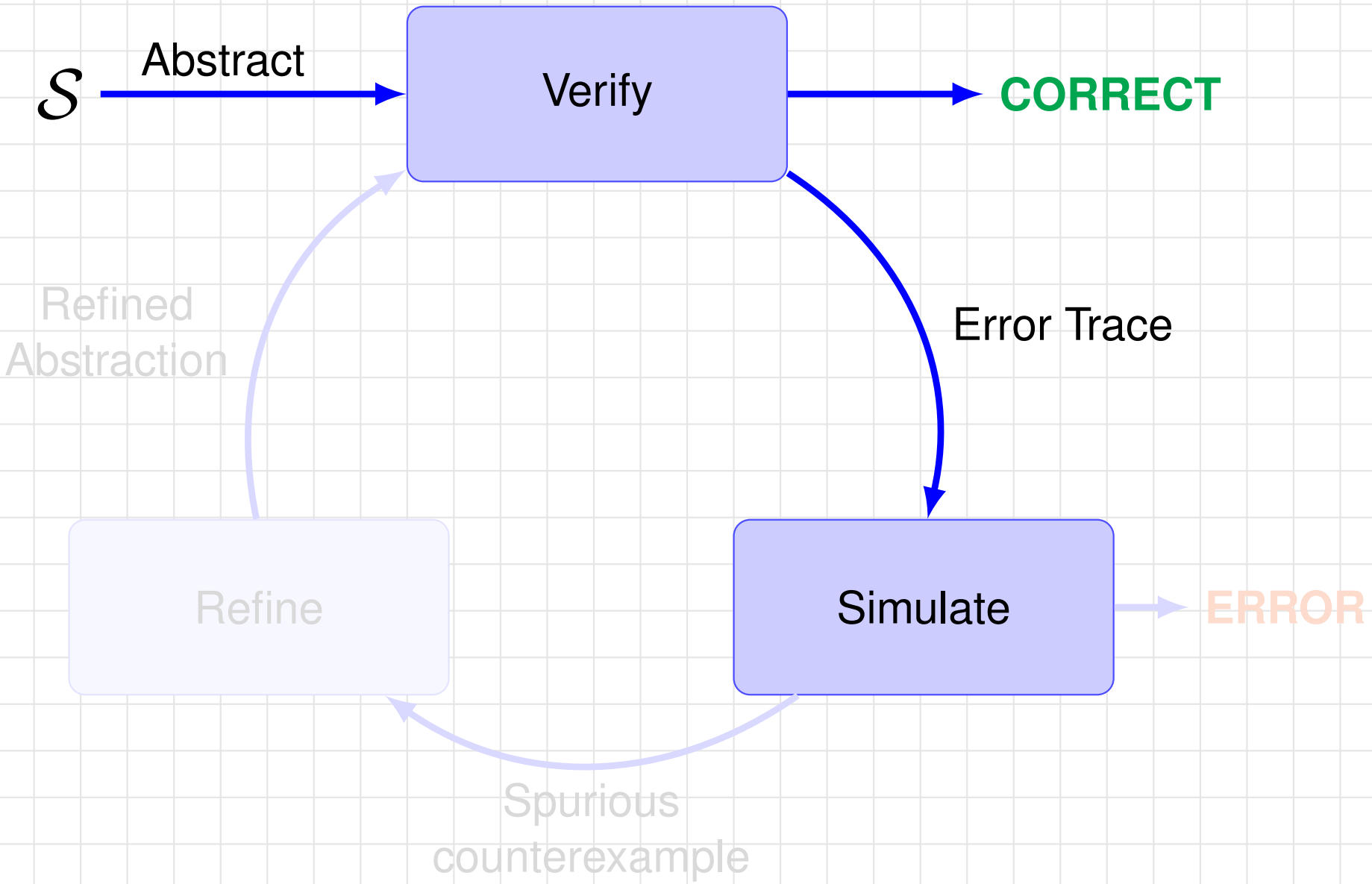
$error : x < 0$



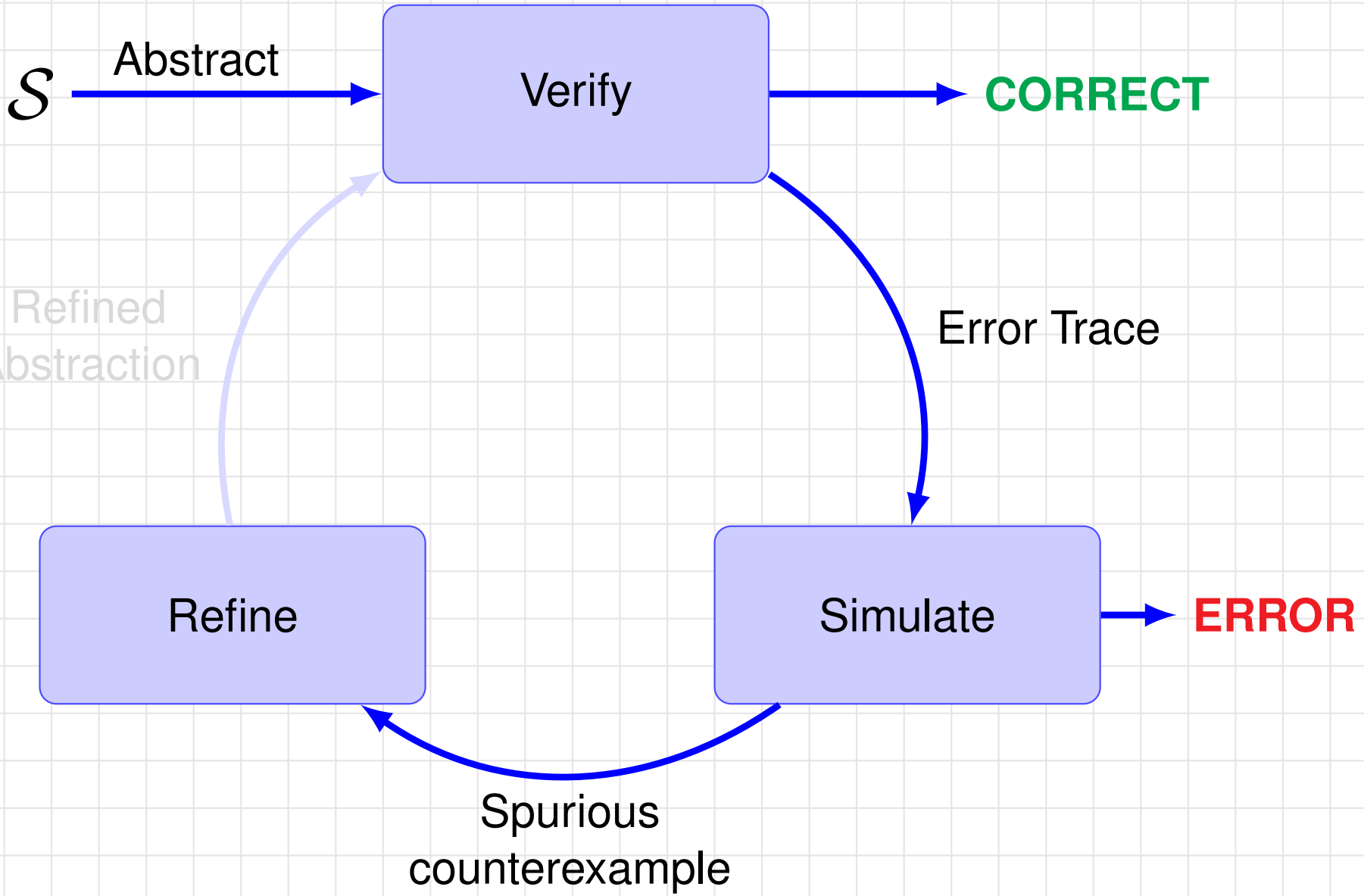
Counterexample-Guided Abstraction Refinement



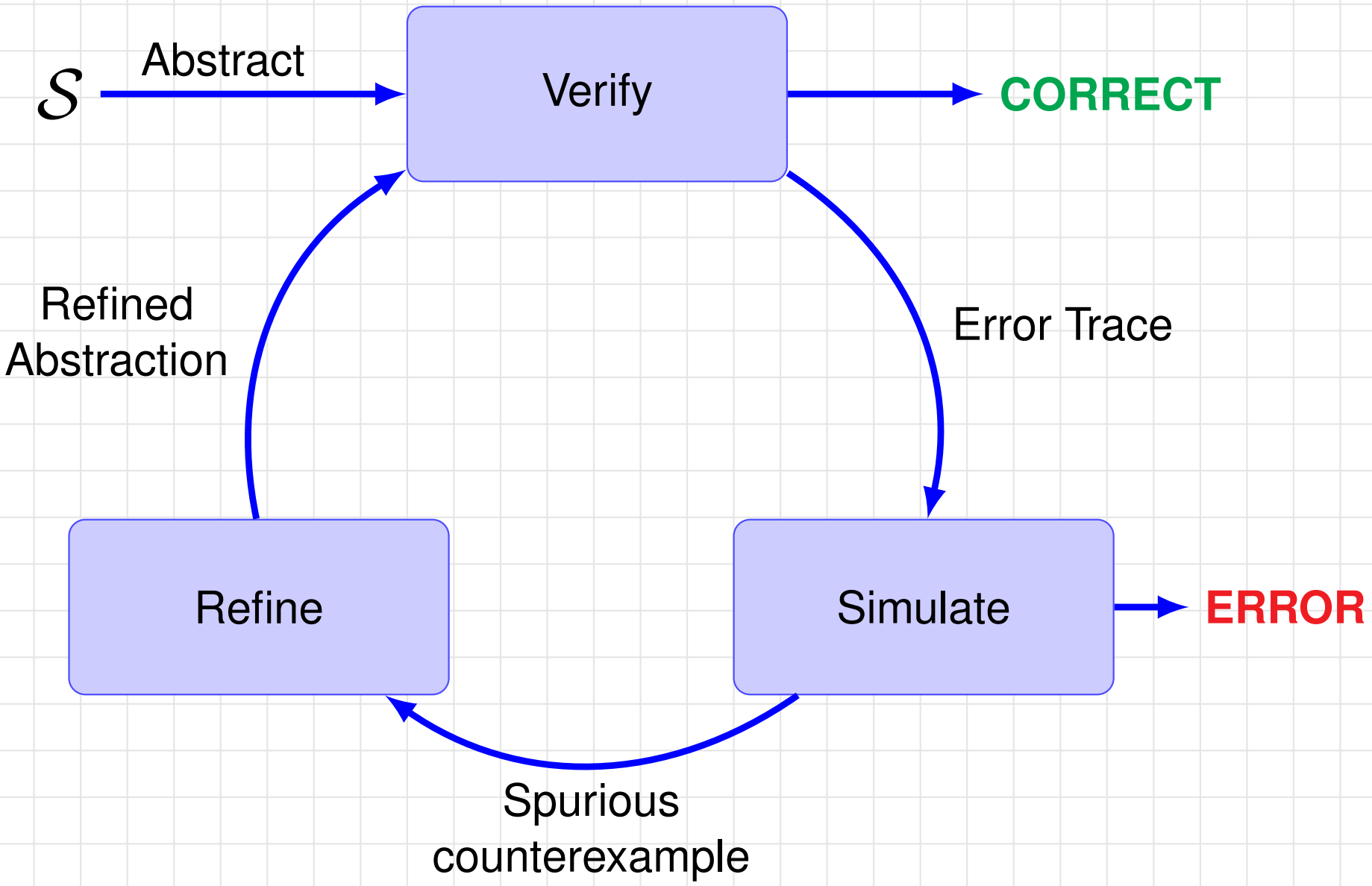
Counterexample-Guided Abstraction Refinement



Counterexample-Guided Abstraction Refinement

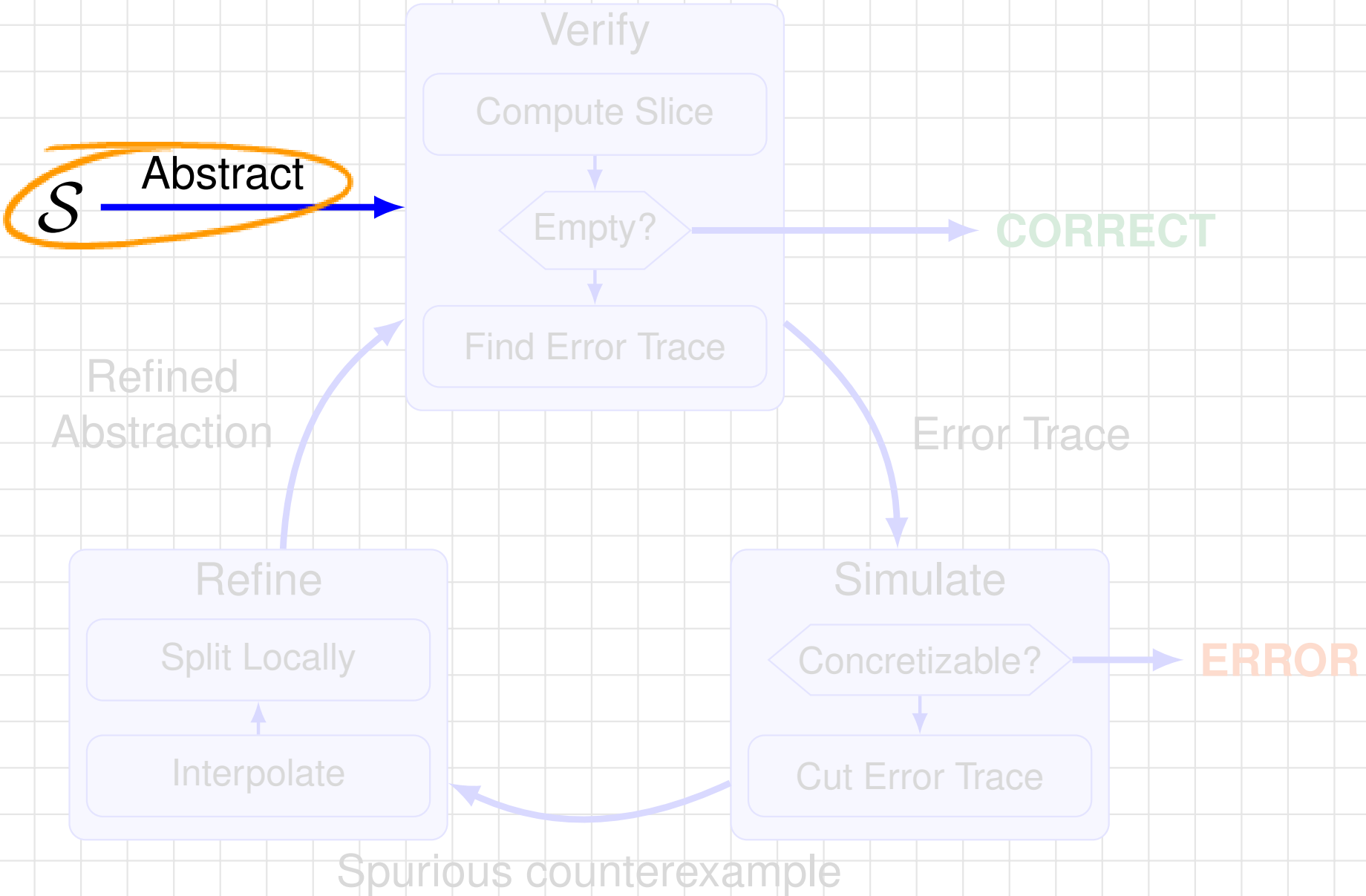


Counterexample-Guided Abstraction Refinement



Slicing Abstraction Refinement

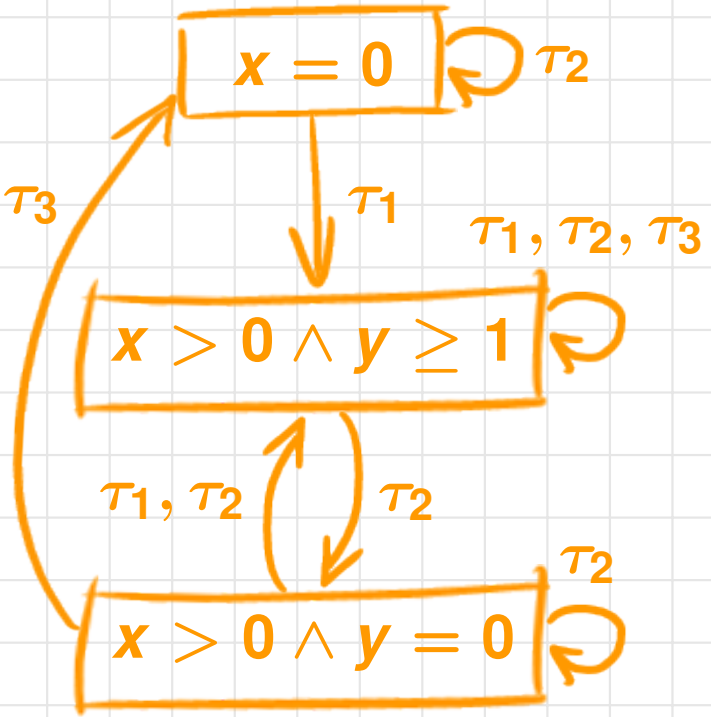
Slicing Abstraction Refinement



System and Certificate

Transition System $\mathcal{S} = (\mathit{init}, \mathcal{T}, \mathit{error})$

- Initial condition $\mathit{init}(V)$
- Set of system transitions $\mathcal{T}$: $\tau_i(V, V')$
- Error condition $\mathit{error}(V)$



Certificate $\mathcal{C} = (N, E, \varphi, \eta)$

- Finite graph with nodes N and edges E
- Node labeling: $\varphi : N \rightarrow \mathit{Asrt}(V)$
- Edge labeling: $\eta : E \rightarrow 2^{\mathcal{T}}$

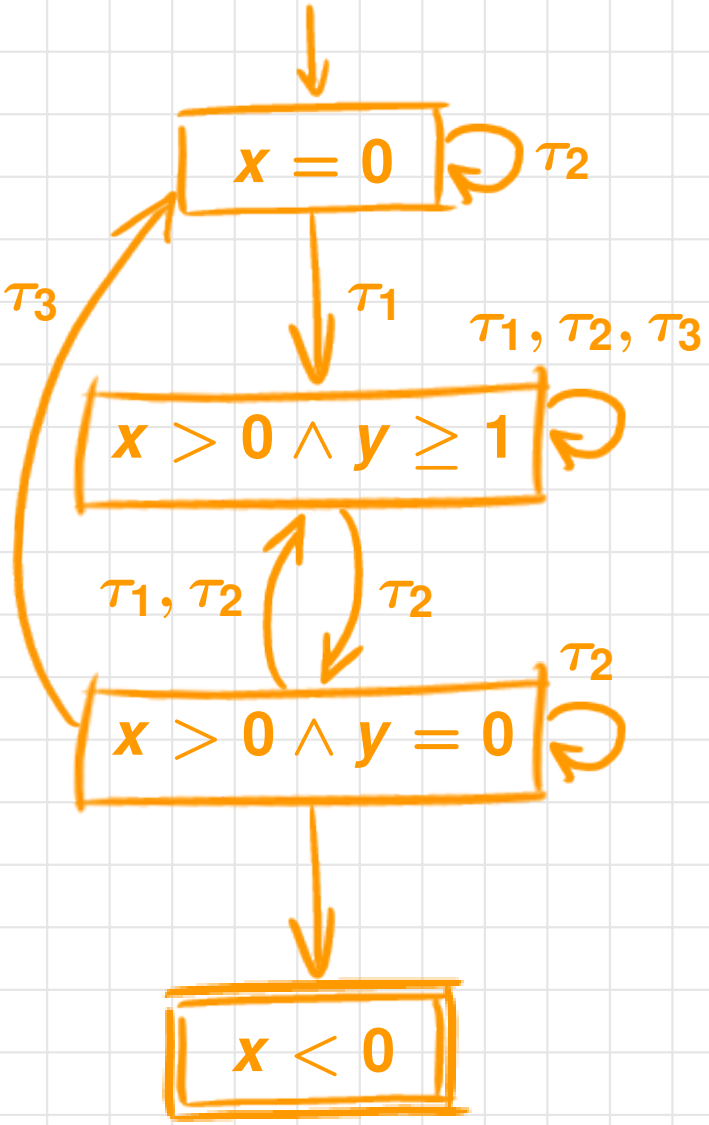
System and Abstraction

Transition System $\mathcal{S} = (init, \mathcal{T}, error)$

- Initial condition $init(V)$
- Set of system transitions $\mathcal{T}$: $\tau_i(V, V')$
- Error condition $error(V)$

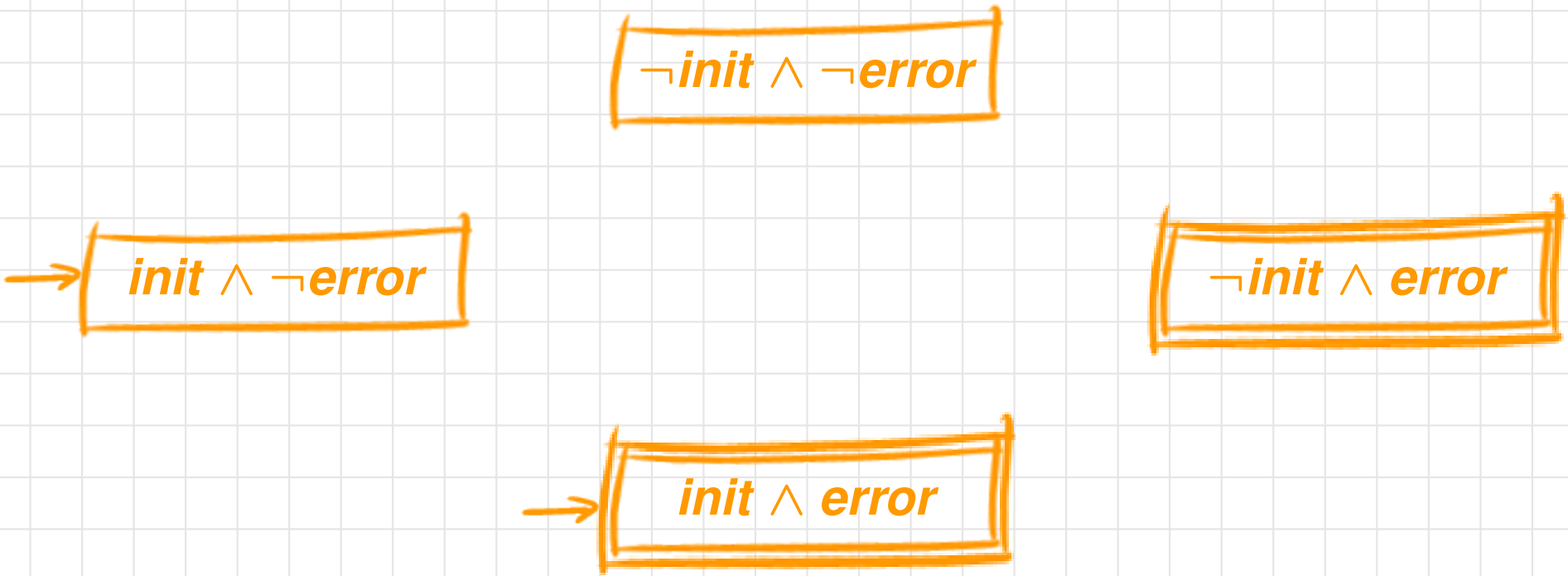
Abstraction $\mathcal{A} = (N, E, \varphi, \eta, I, F)$

- Finite graph with nodes N and edges E
- Node labeling: $\varphi : N \rightarrow Asrt(V)$
- Edge labeling: $\eta : E \rightarrow 2^{\mathcal{T}}$
- Initial nodes $I \subseteq N$, final nodes $F \subseteq N$



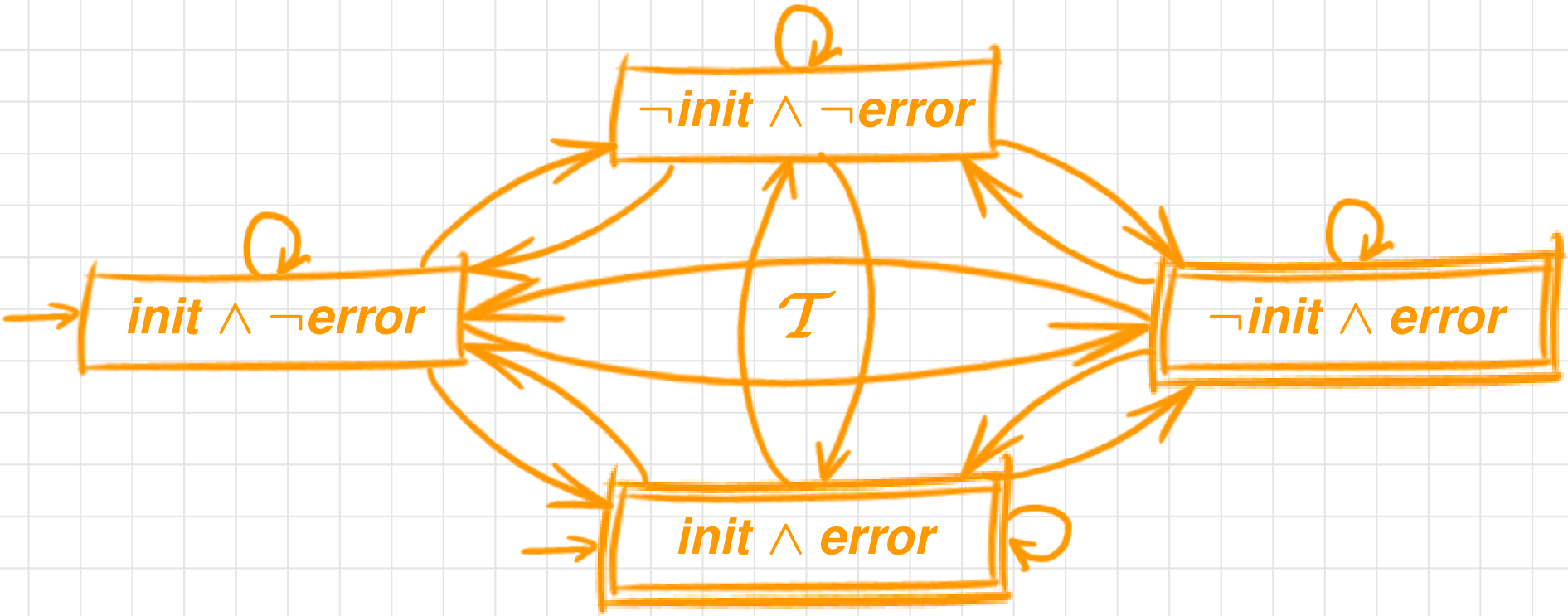
Initial Abstraction

- Initial abstraction over *init*, *error*



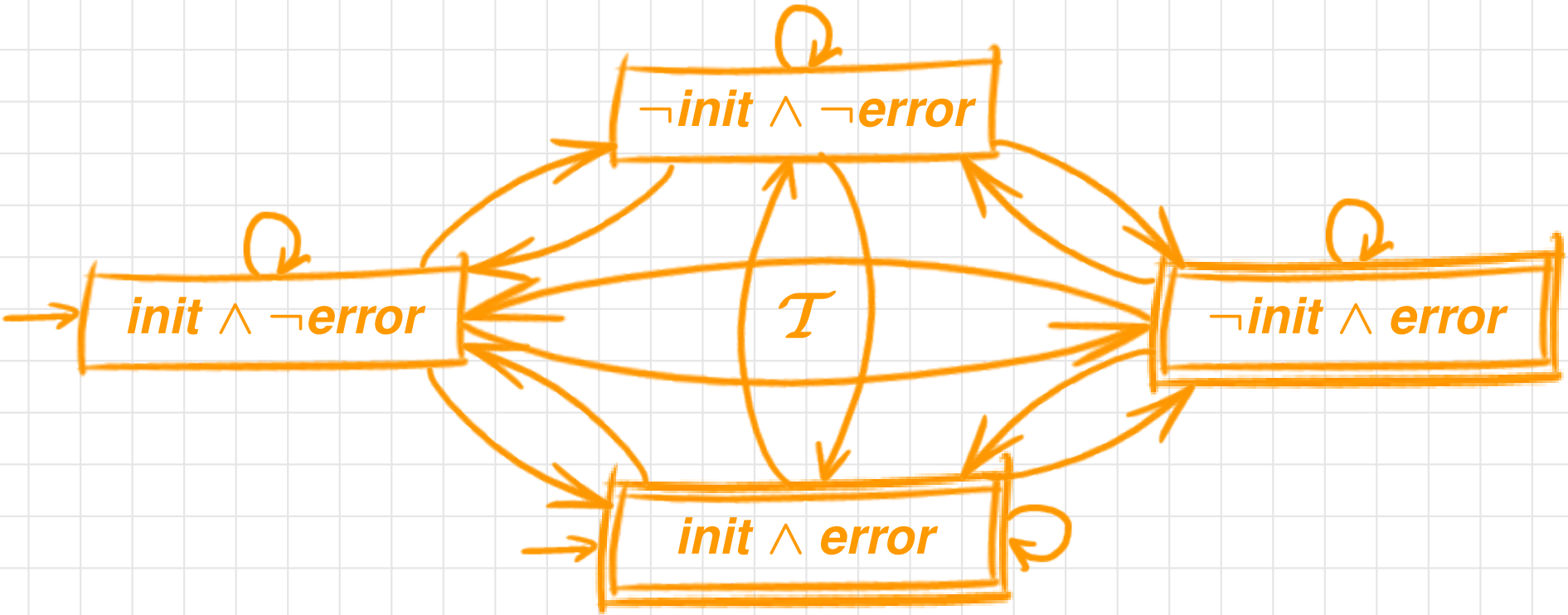
Initial Abstraction

- Initial abstraction over *init*, *error*



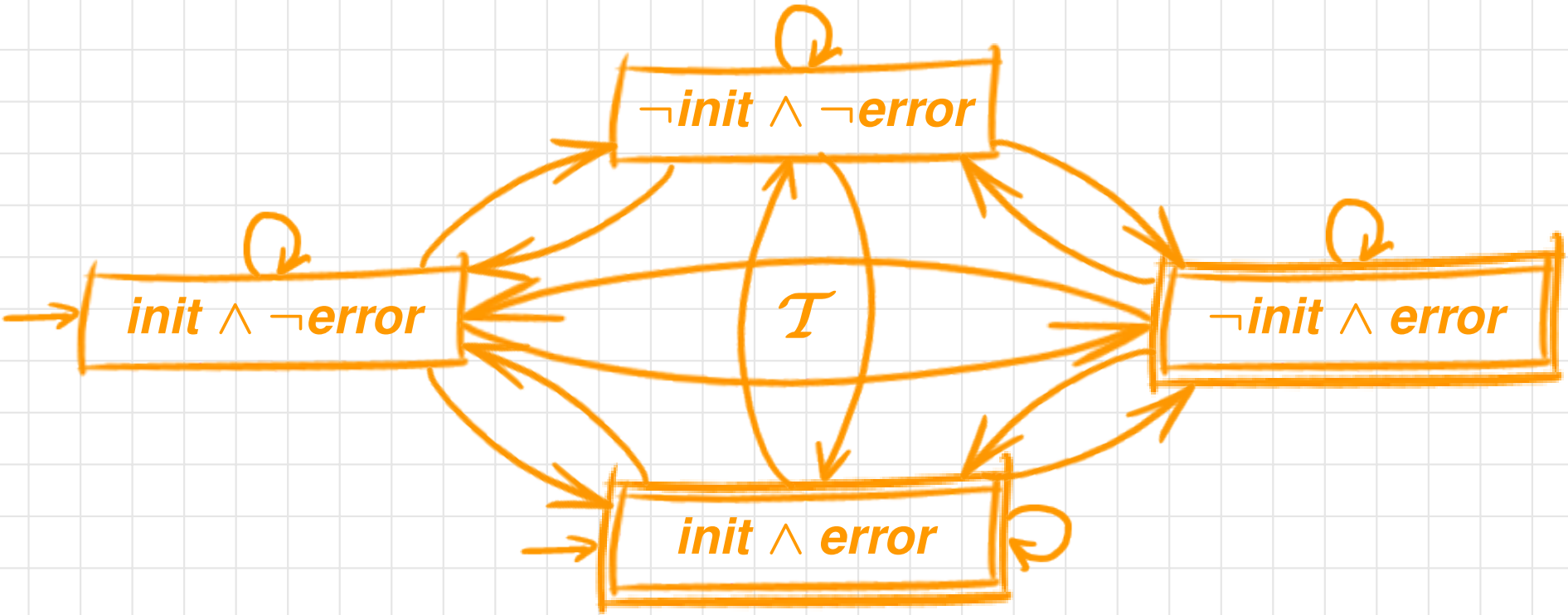
Initial Abstraction

- Initial abstraction over *init*, *error*
- We need only *irreducible* error paths



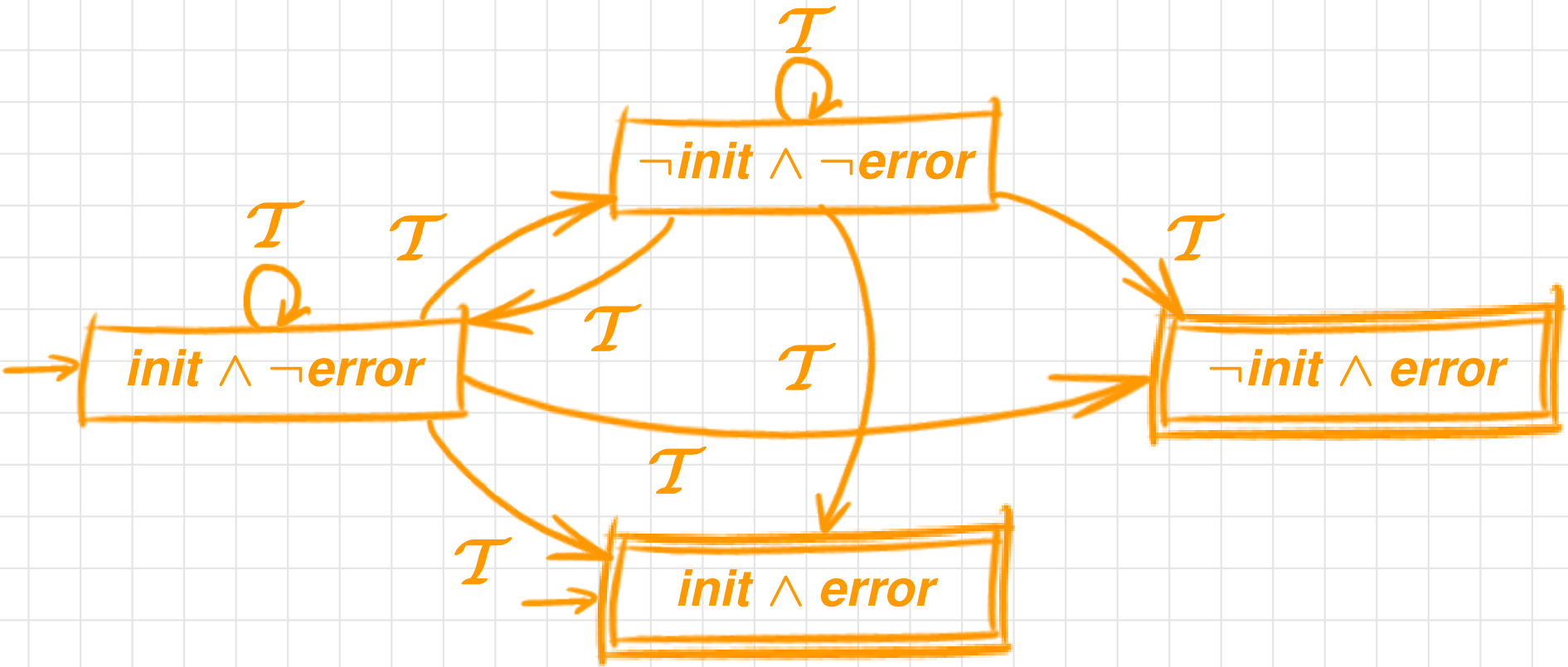
Initial Abstraction

- Initial abstraction over *init*, *error*
- We need only *irreducible* error paths



Initial Abstraction

- Initial abstraction over *init*, *error*
- We need only *irreducible* error paths

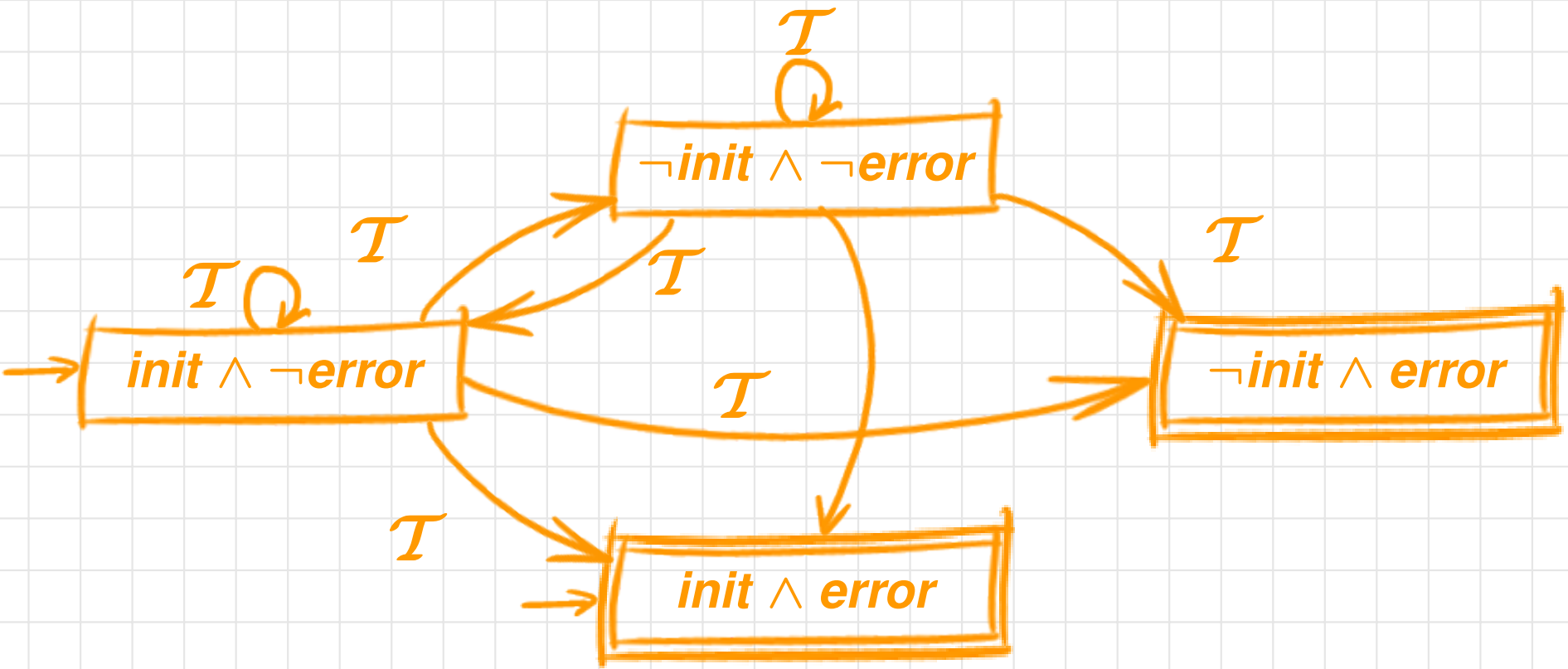


Example: Initial Abstraction

$\tau_1 : x' = 1 \wedge y' = x + 1$

$init : x = 0$ $\quad \tau_2 : y' = x - 1 \wedge x' = x$ $\quad error : x < 0$

$\tau_3 : x \geq 1 \wedge x' = y \wedge y' = y$



Example: Initial Abstraction

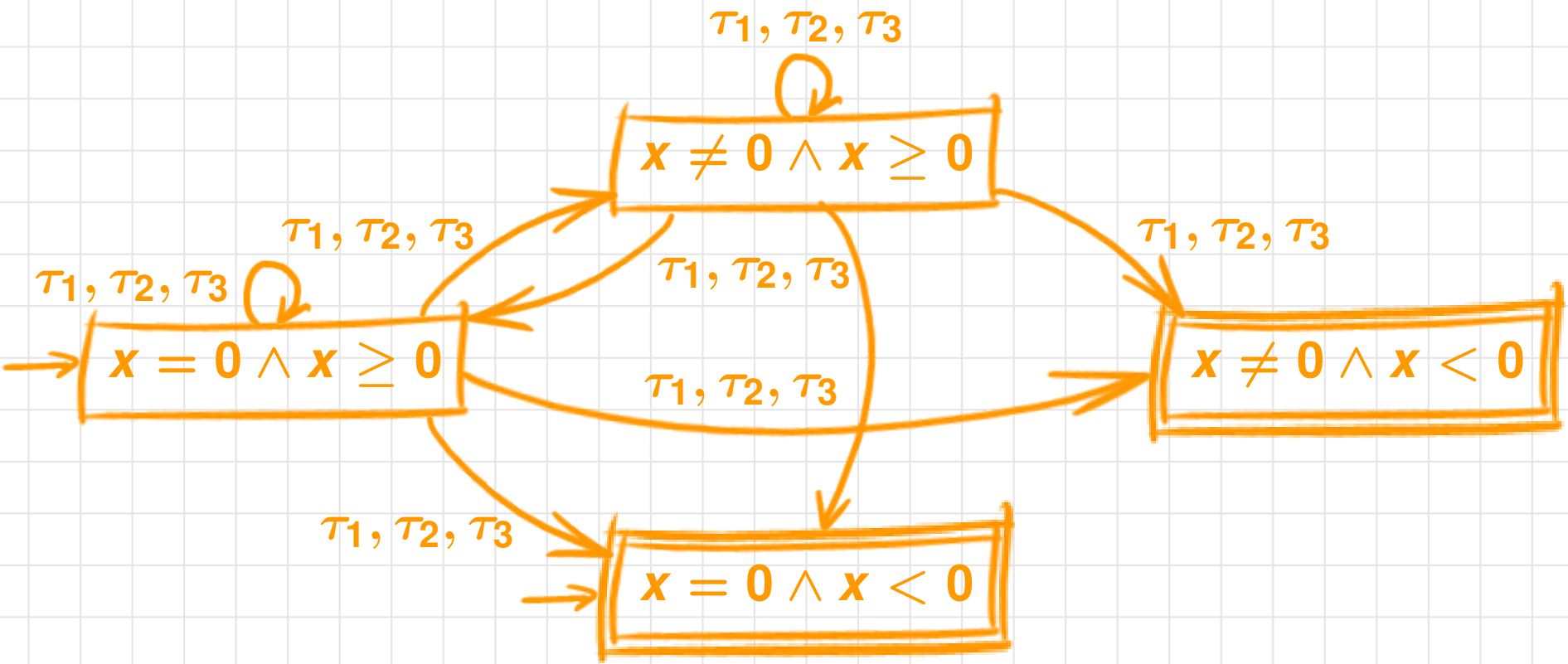
$\tau_1 : x' = 1 \wedge y' = x + 1$

$\tau_2 : y' = x - 1 \wedge x' = x$

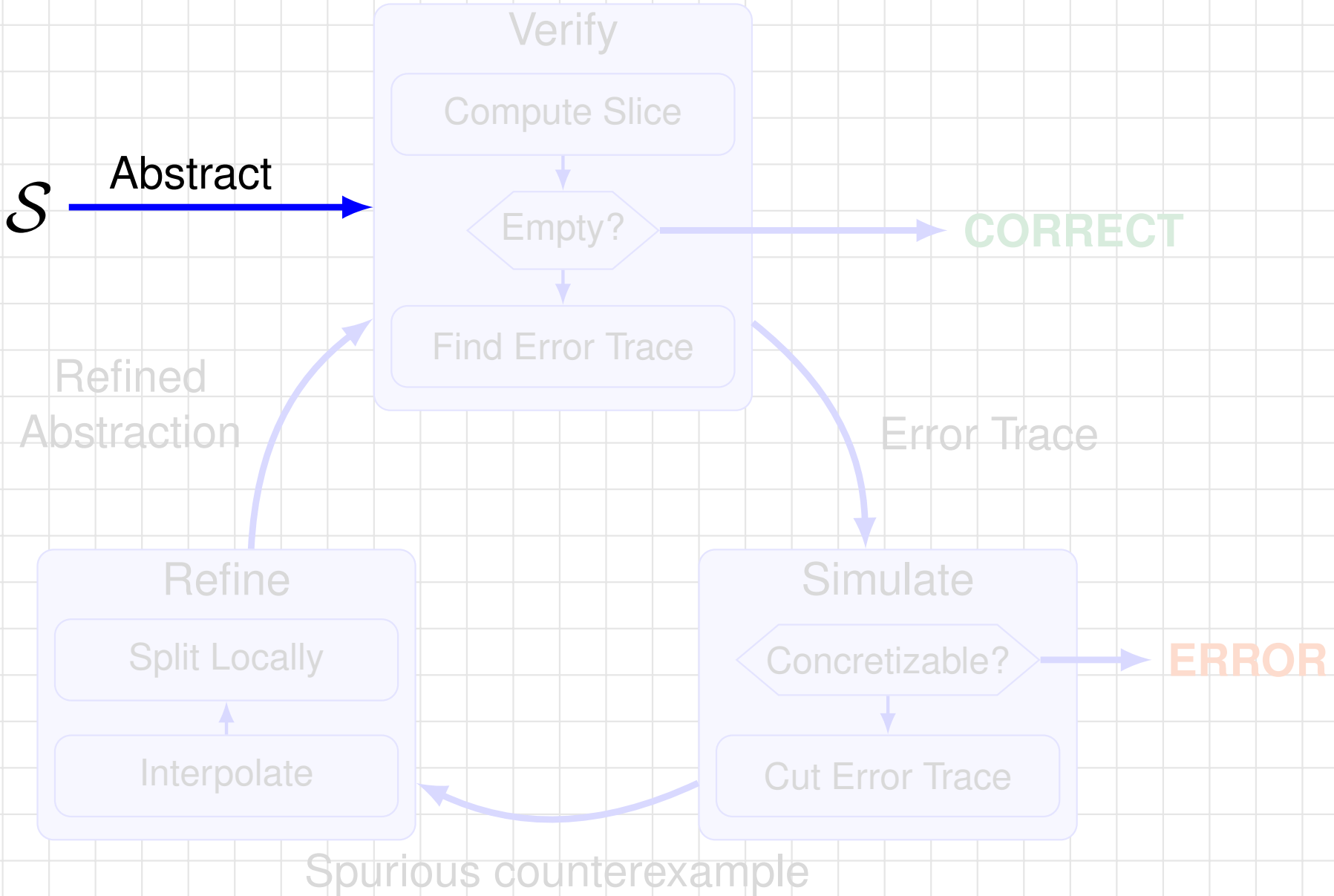
$\tau_3 : x \geq 1 \wedge x' = y \wedge y' = y$

$init : x = 0$

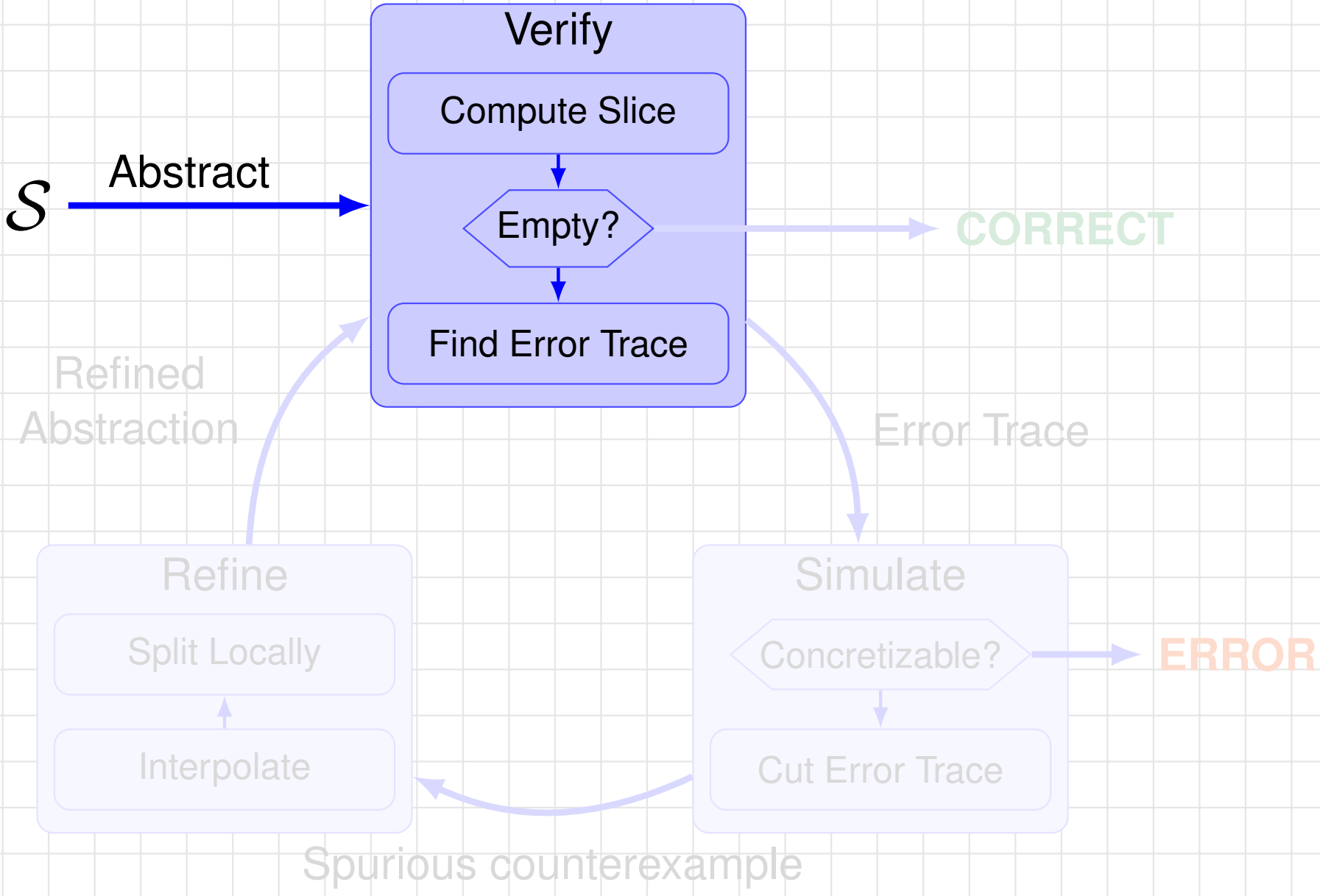
$error : x < 0$



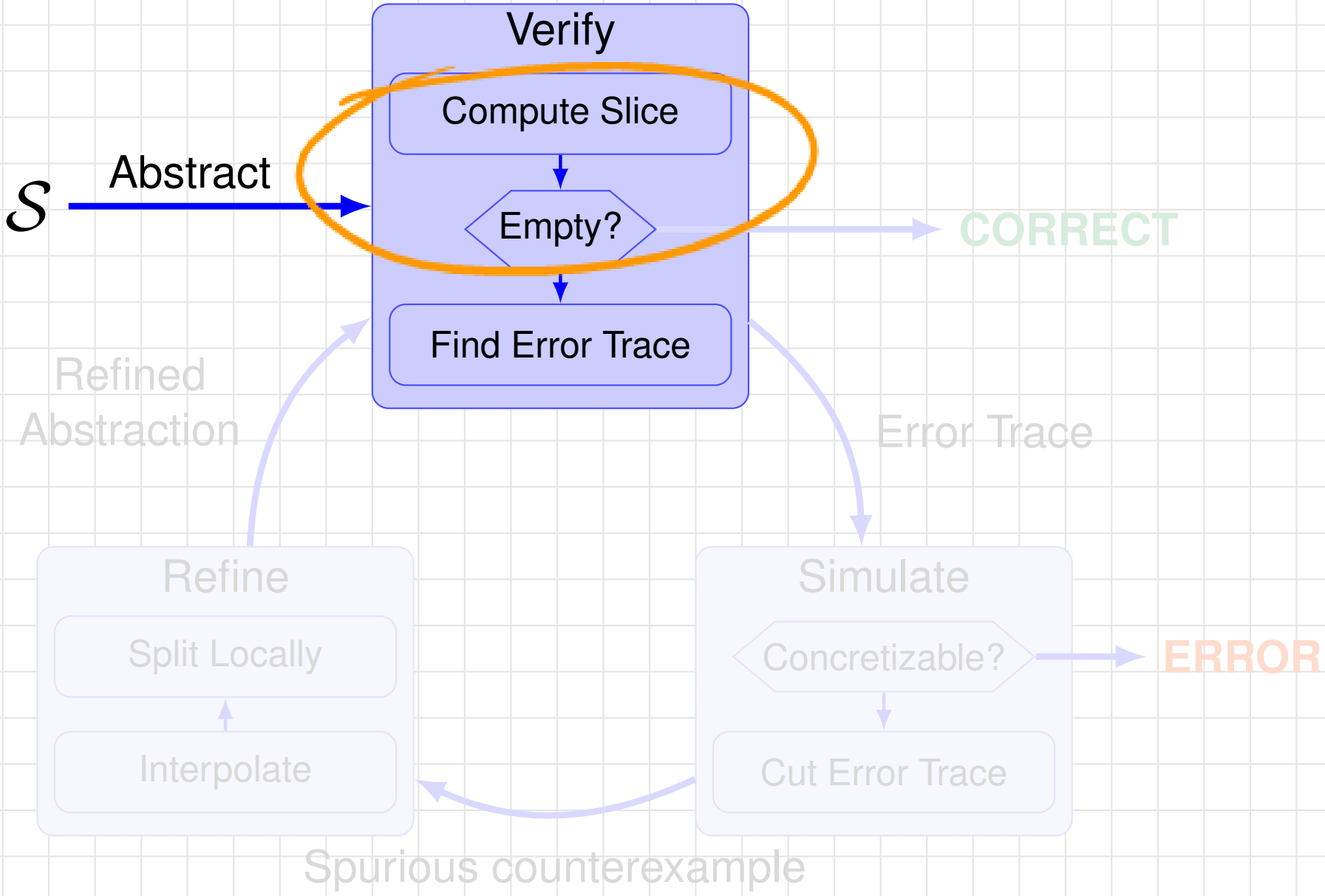
Slicing Abstraction Refinement



Slicing Abstraction Refinement



Slicing Abstraction Refinement



Slicing

Slicing

- Inconsistent nodes

Slicing

- Inconsistent nodes

Slicing

- Inconsistent nodes

- Unreachable nodes

Slicing

- Inconsistent nodes

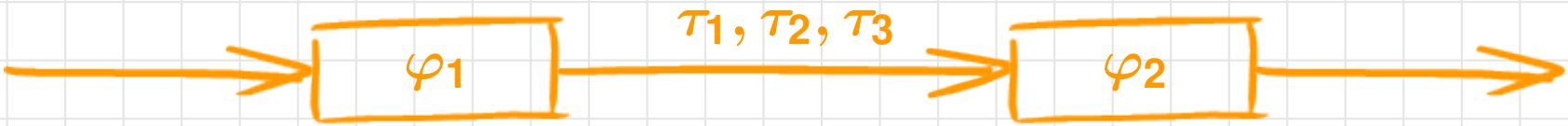
- Unreachable nodes

Slicing

- Inconsistent nodes

- Unreachable nodes

- Inconsistent transitions

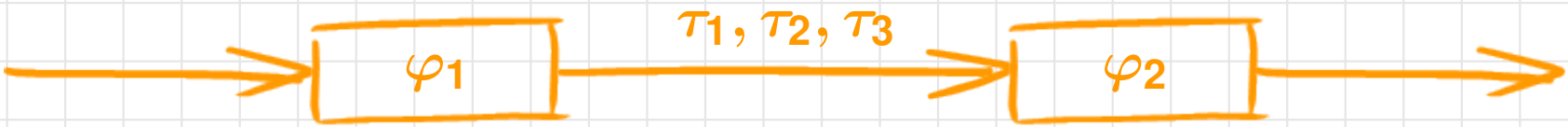


Slicing

- Inconsistent nodes

- Unreachable nodes

- Inconsistent transitions



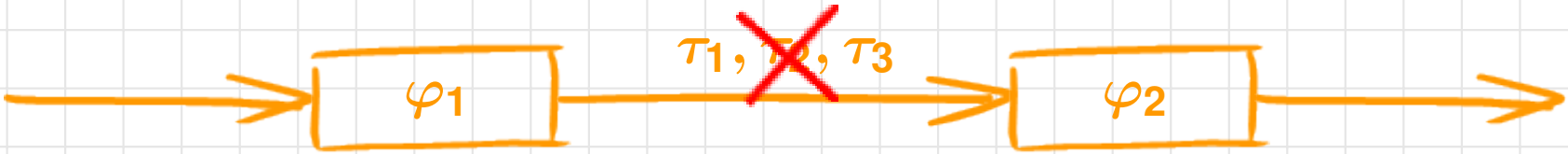
$\varphi_1 \wedge \tau_2 \wedge \varphi_2$ unsatisfiable

Slicing

- Inconsistent nodes

- Unreachable nodes

- Inconsistent transitions



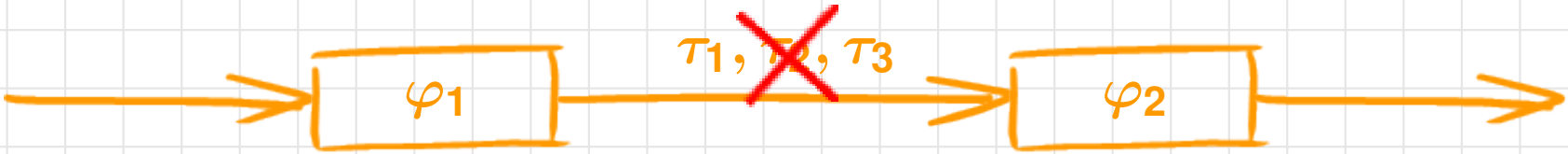
$\varphi_1 \wedge \tau_2 \wedge \varphi_2$ unsatisfiable

Slicing

- Inconsistent nodes

- Unreachable nodes

- Inconsistent transitions



$\varphi_1 \wedge \tau_2 \wedge \varphi_2$ unsatisfiable

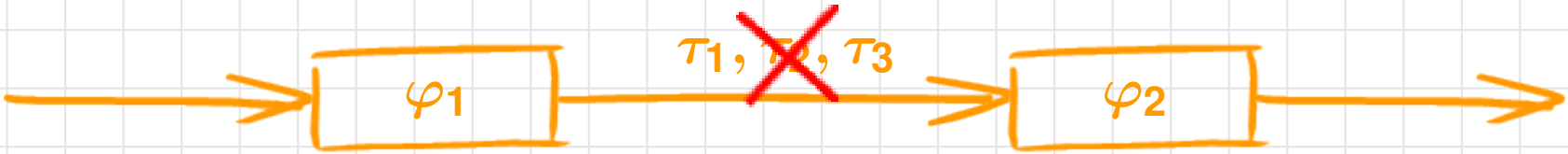
- Empty edges

Slicing

- Inconsistent nodes

- Unreachable nodes

- Inconsistent transitions



$\varphi_1 \wedge \tau_2 \wedge \varphi_2$ unsatisfiable

- Empty edges

Example: Slicing

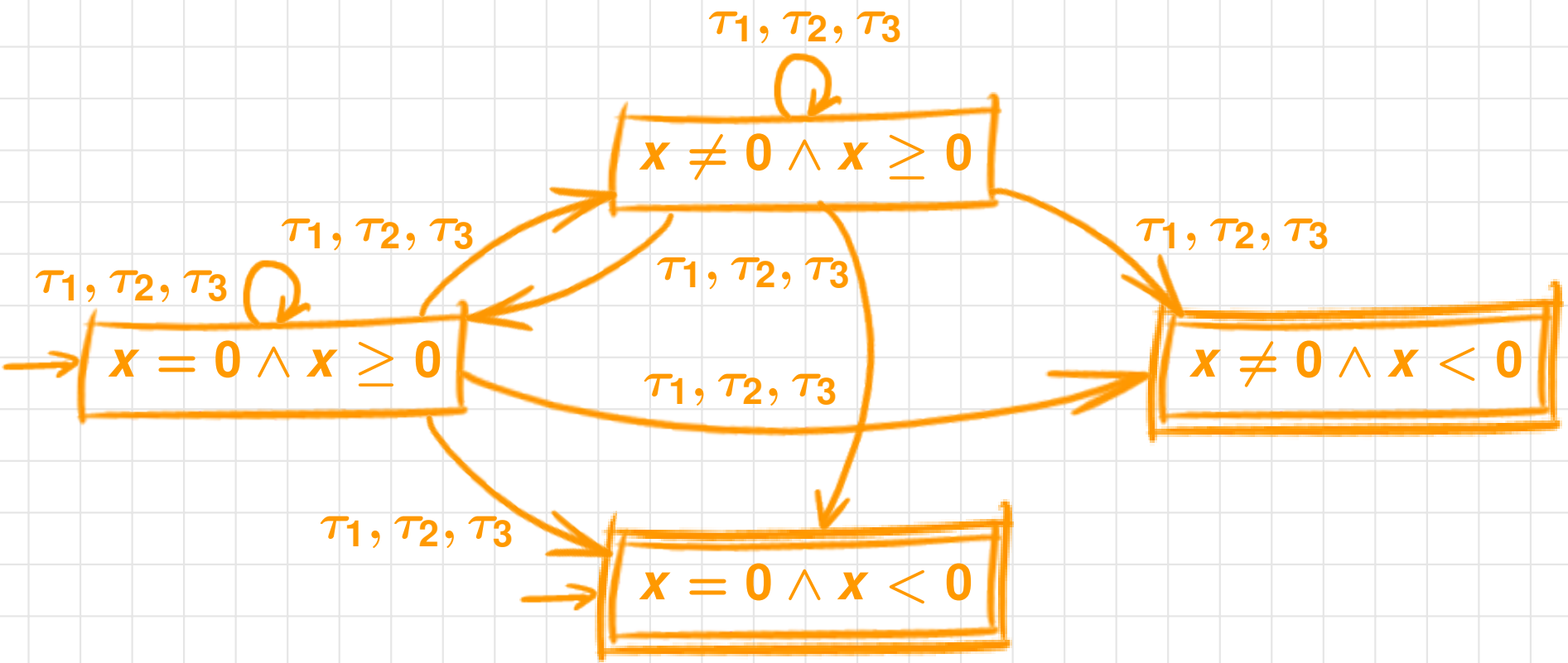
$\tau_1 : x' = 1 \wedge y' = x + 1$

$\tau_2 : y' = x - 1 \wedge x' = x$

$\tau_3 : x \geq 1 \wedge x' = y \wedge y' = y$

$init : x = 0$

$error : x < 0$



Example: Slicing

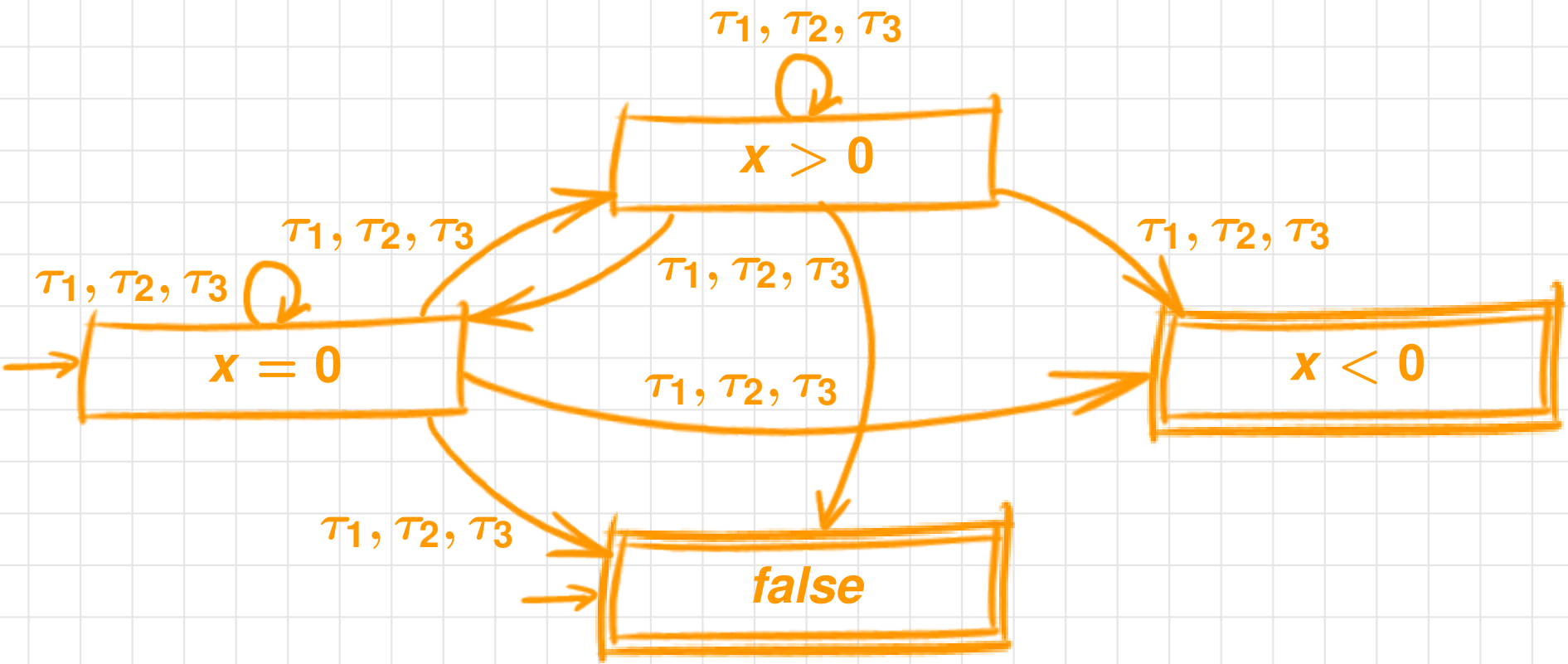
$\tau_1 : x' = 1 \wedge y' = x + 1$

$\tau_2 : y' = x - 1 \wedge x' = x$

$\tau_3 : x \geq 1 \wedge x' = y \wedge y' = y$

$init : x = 0$

$error : x < 0$



Example: Slicing

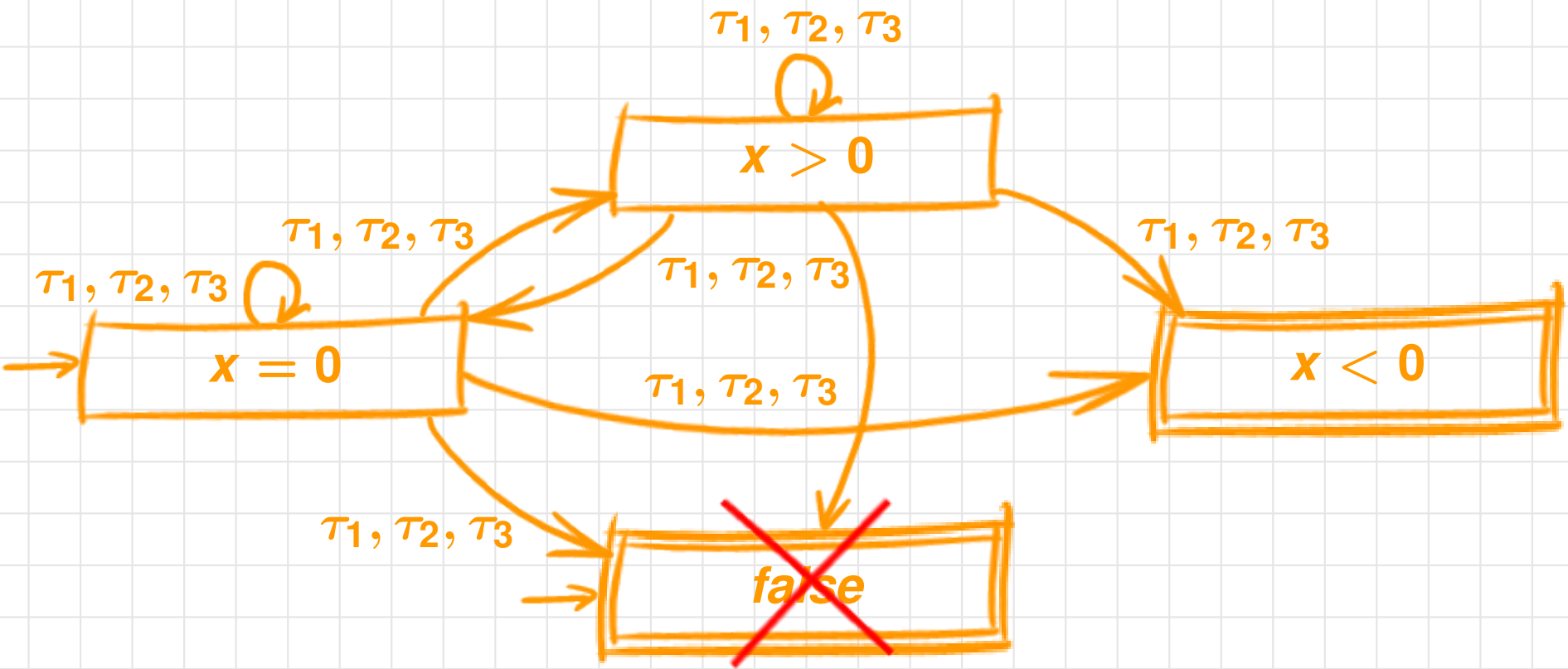
$\tau_1 : x' = 1 \wedge y' = x + 1$

$\tau_2 : y' = x - 1 \wedge x' = x$

$\tau_3 : x \geq 1 \wedge x' = y \wedge y' = y$

$init : x = 0$

$error : x < 0$



Example: Slicing

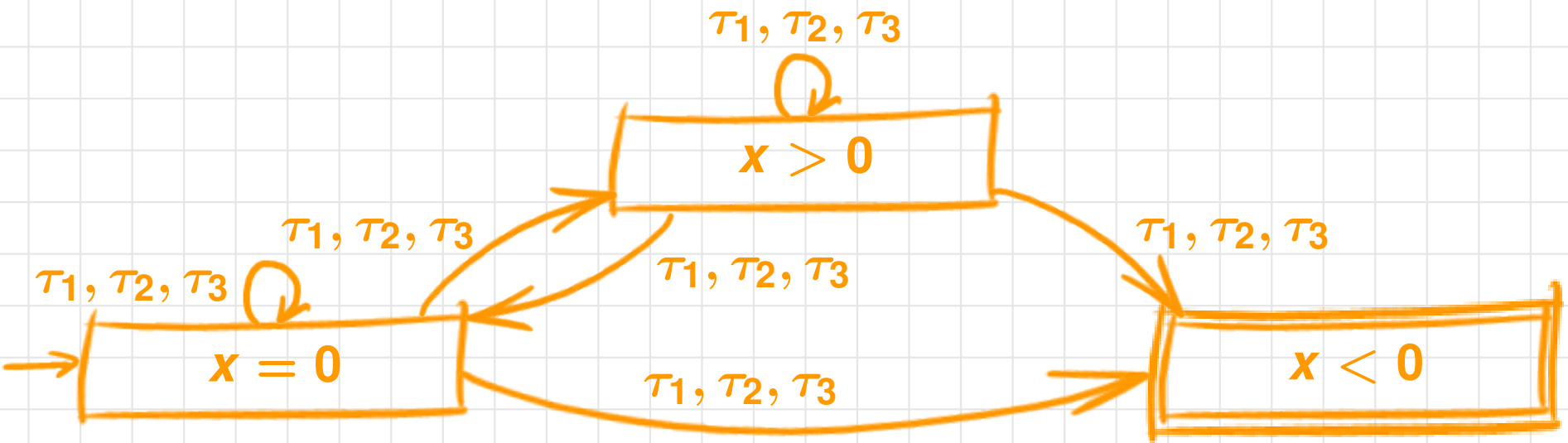
$\tau_1 : x' = 1 \wedge y' = x + 1$

$\tau_2 : y' = x - 1 \wedge x' = x$

$\tau_3 : x \geq 1 \wedge x' = y \wedge y' = y$

$init : x = 0$

$error : x < 0$



Example: Slicing

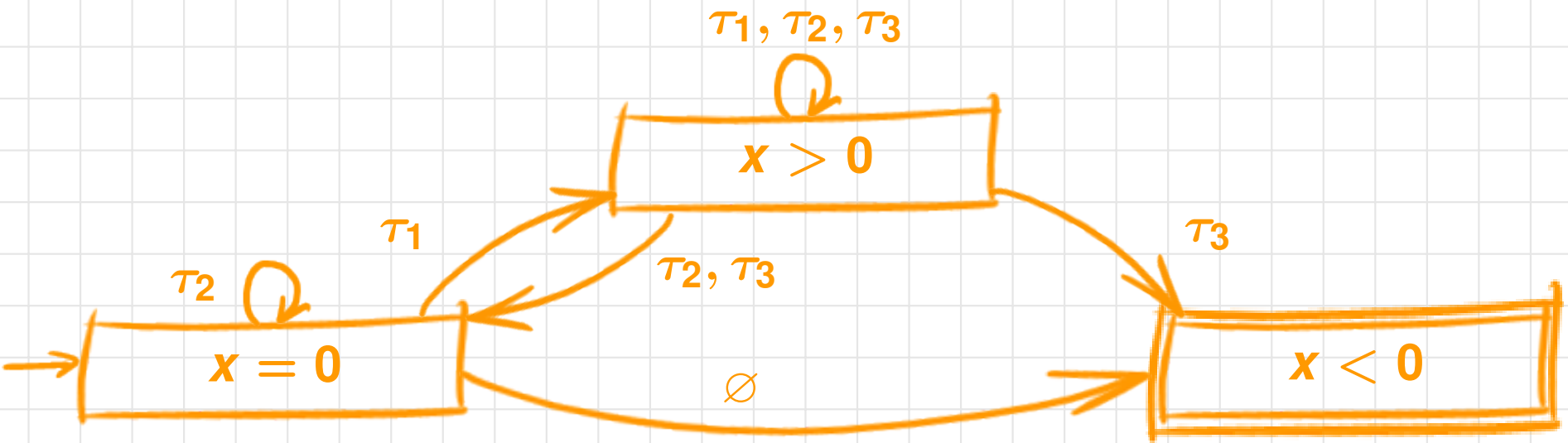
$\tau_1 : x' = 1 \wedge y' = x + 1$

$\tau_2 : y' = x - 1 \wedge x' = x$

$\tau_3 : x \geq 1 \wedge x' = y \wedge y' = y$

$init : x = 0$

$error : x < 0$



Example: Slicing

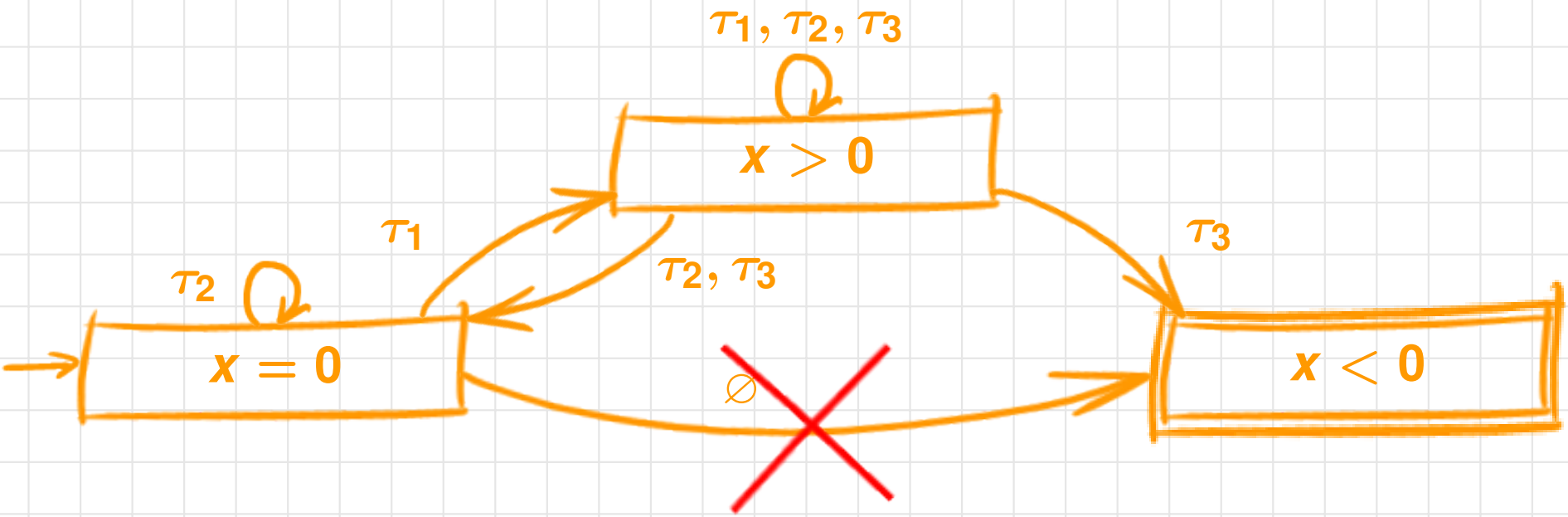
$\tau_1 : x' = 1 \wedge y' = x + 1$

$\tau_2 : y' = x - 1 \wedge x' = x$

$\tau_3 : x \geq 1 \wedge x' = y \wedge y' = y$

$init : x = 0$

$error : x < 0$



Example: Slicing

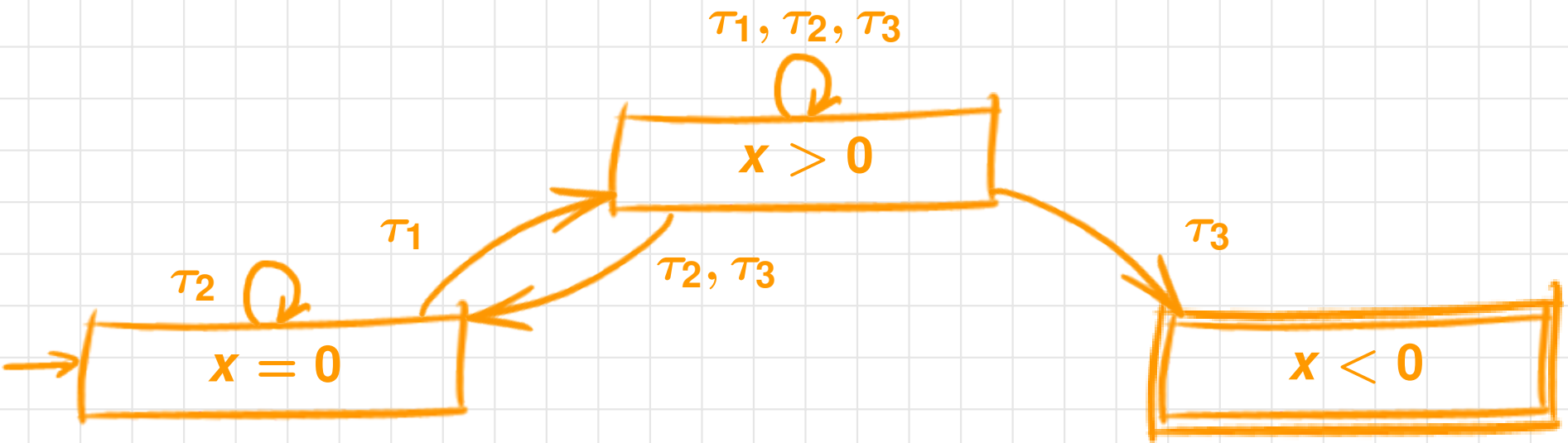
$\tau_1 : x' = 1 \wedge y' = x + 1$

$\tau_2 : y' = x - 1 \wedge x' = x$

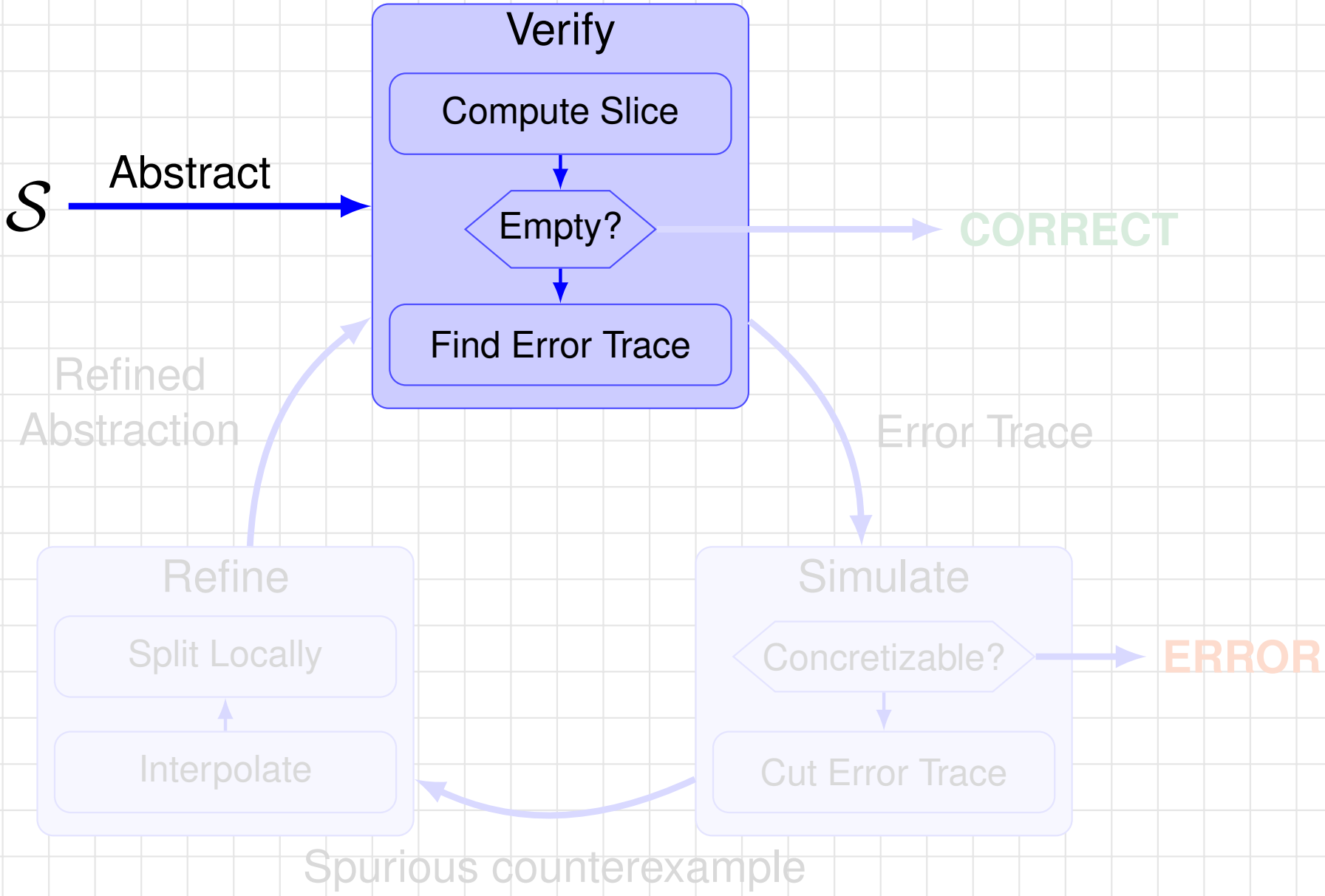
$\tau_3 : x \geq 1 \wedge x' = y \wedge y' = y$

$init : x = 0$

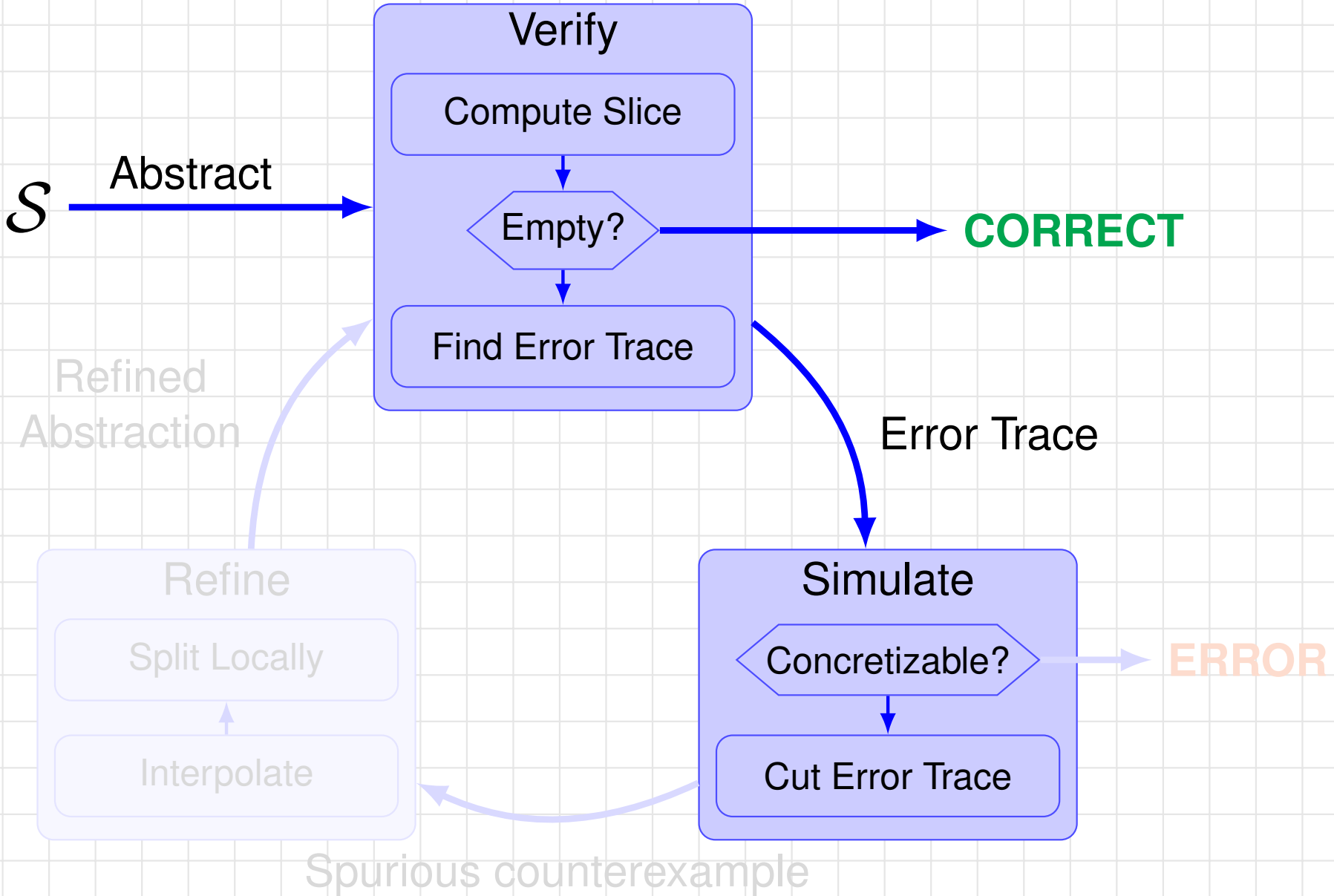
$error : x < 0$



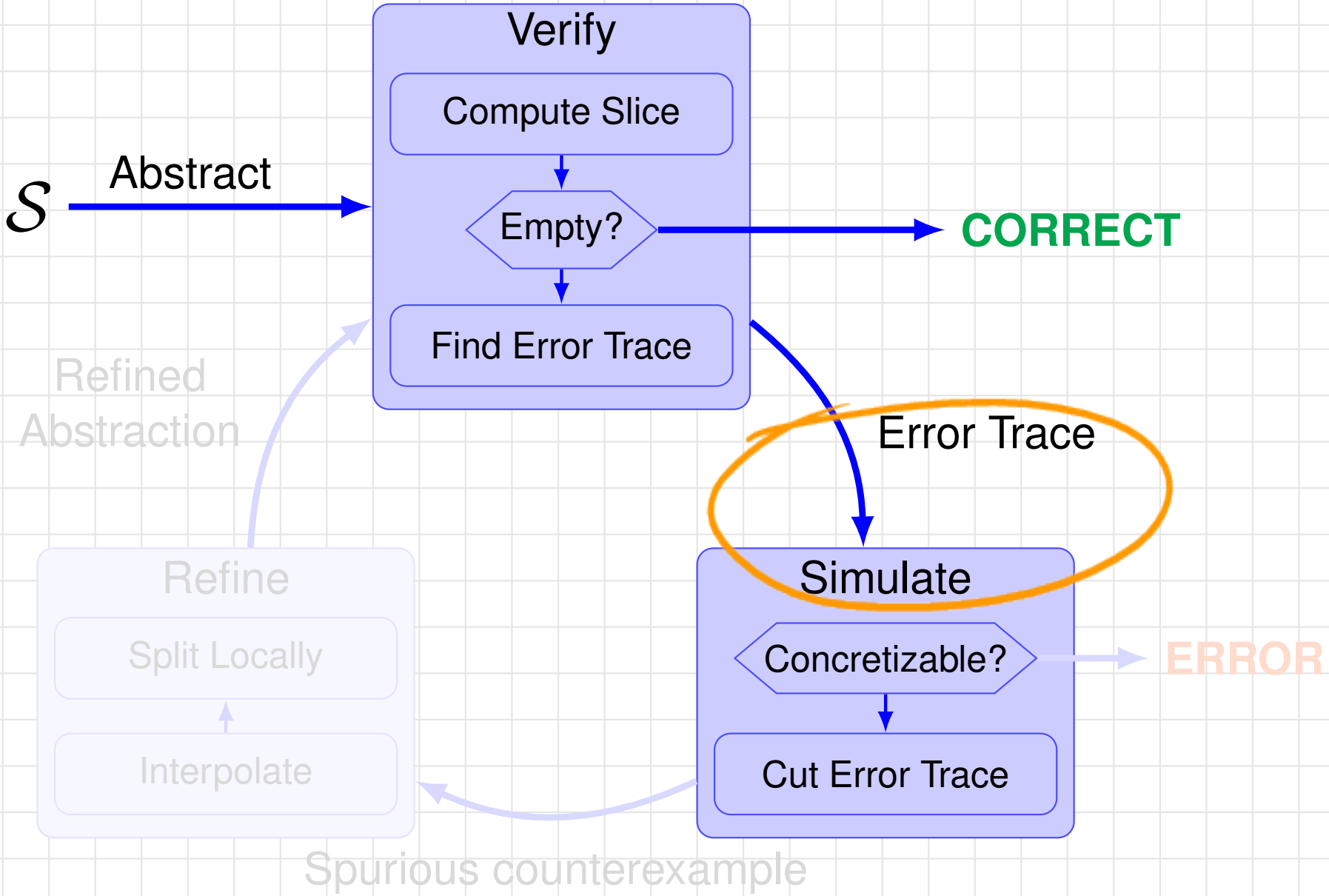
Slicing Abstraction Refinement



Slicing Abstraction Refinement



Slicing Abstraction Refinement



Example: Error Path Analysis

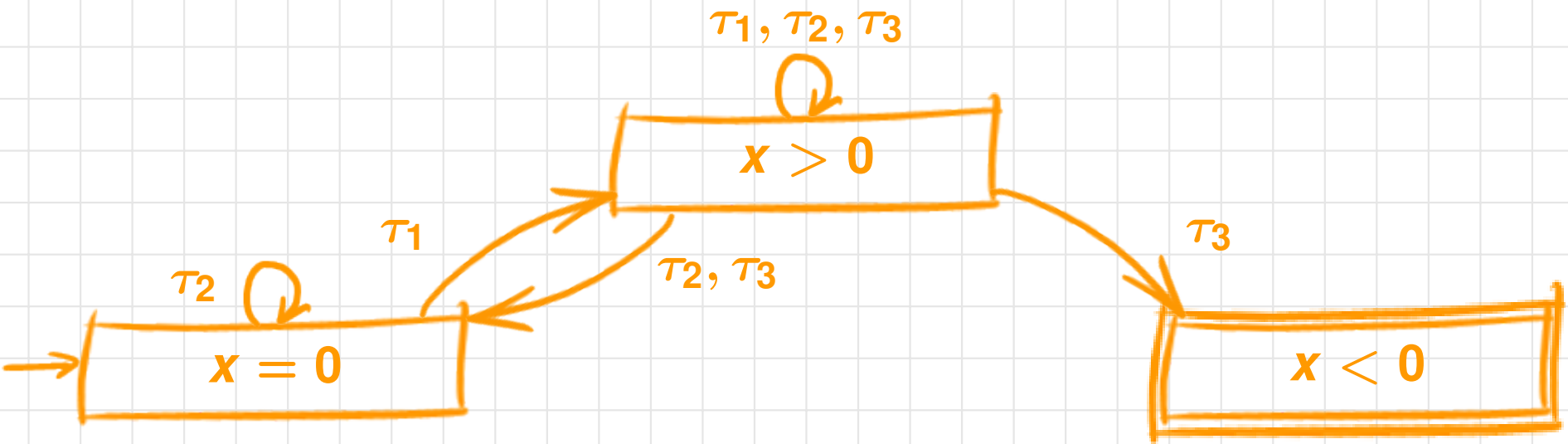
$\tau_1 : x' = 1 \wedge y' = x + 1$

$\tau_2 : y' = x - 1 \wedge x' = x$

$\tau_3 : x \geq 1 \wedge x' = y \wedge y' = y$

$init : x = 0$

$error : x < 0$



Example: Error Path Analysis

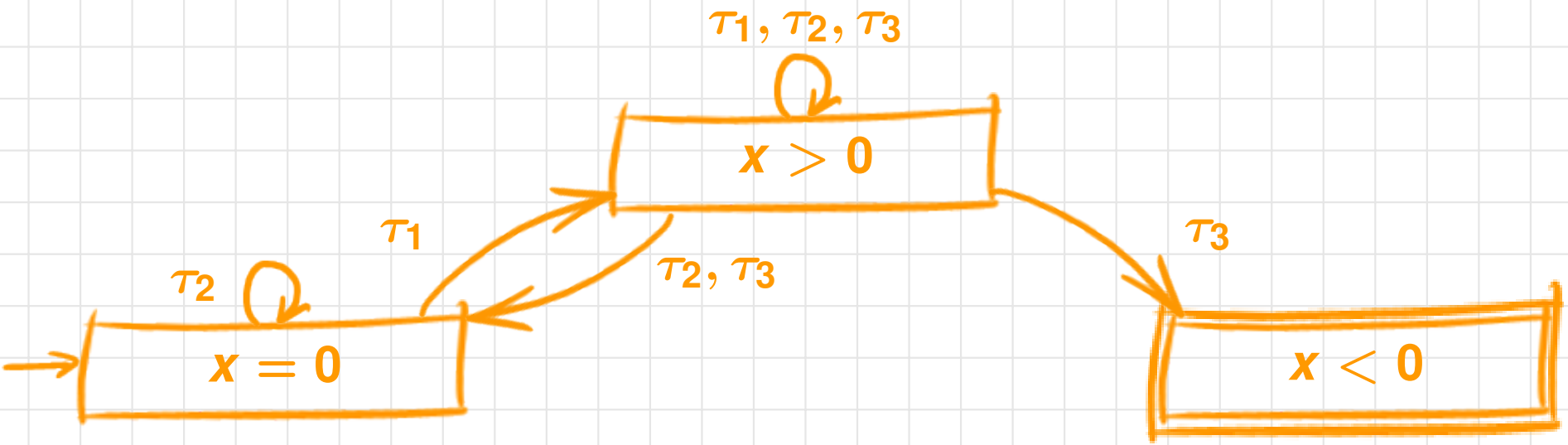
$\tau_1 : x' = 1 \wedge y' = x + 1$

$\tau_2 : y' = x - 1 \wedge x' = x$

$\tau_3 : x \geq 1 \wedge x' = y \wedge y' = y$

$init : x = 0$

$error : x < 0$



$x^0 = 0$

$\Phi =$

Example: Error Path Analysis

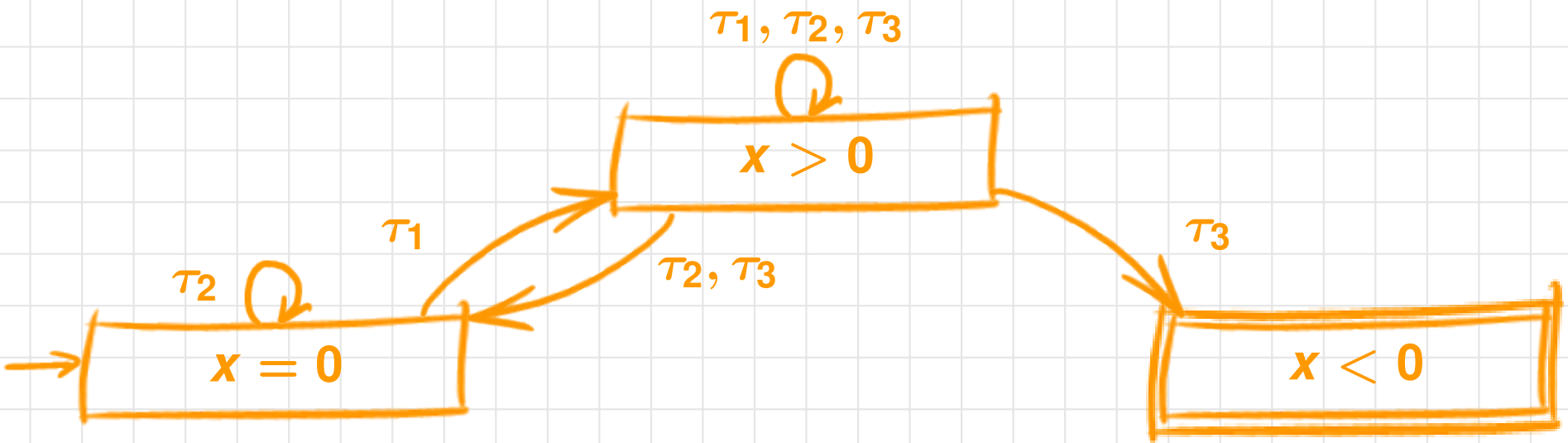
$\tau_1 : x' = 1 \wedge y' = x + 1$

$\tau_2 : y' = x - 1 \wedge x' = x$

$\tau_3 : x \geq 1 \wedge x' = y \wedge y' = y$

$init : x = 0$

$error : x < 0$



$$\Phi = x^0 = 0 \wedge x^1 = 1 \wedge y^1 = x^0 + 1$$

Example: Error Path Analysis

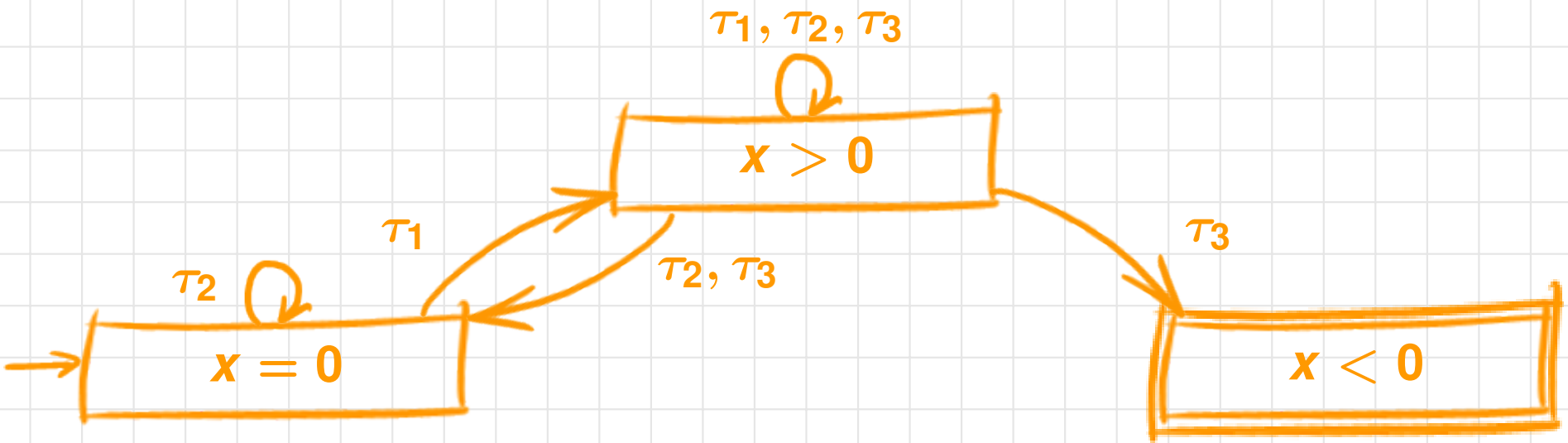
$\tau_1 : x' = 1 \wedge y' = x + 1$

$\tau_2 : y' = x - 1 \wedge x' = x$

$\tau_3 : x \geq 1 \wedge x' = y \wedge y' = y$

$init : x = 0$

$error : x < 0$



$$\Phi = x^0 = 0 \wedge x^1 > 0 \wedge x^1 = 1 \wedge y^1 = x^0 + 1$$

Example: Error Path Analysis

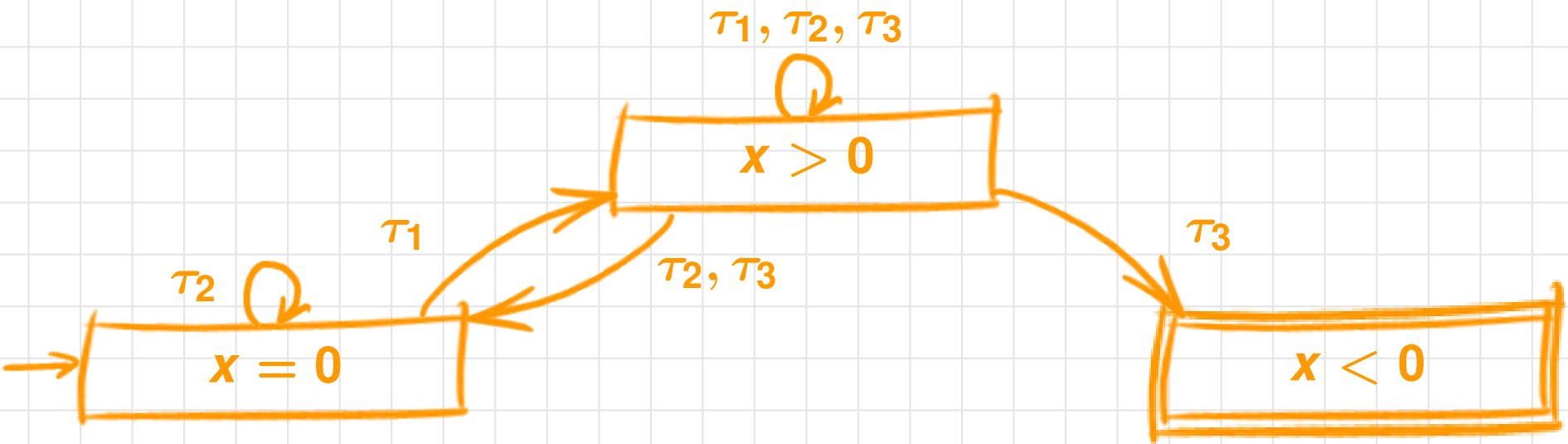
$\tau_1 : x' = 1 \wedge y' = x + 1$

$\tau_2 : y' = x - 1 \wedge x' = x$

$\tau_3 : x \geq 1 \wedge x' = y \wedge y' = y$

$init : x = 0$

$error : x < 0$



$$\Phi = \underbrace{x^0 = 0}_{} \wedge \underbrace{x^1 > 0}_{} \wedge \underbrace{x^1 = 1 \wedge y^1 = x^0 + 1}_{} \wedge \underbrace{x^1 \geq 1 \wedge x^2 = y^1 \wedge y^2 = y^1}_{}$$

Example: Error Path Analysis

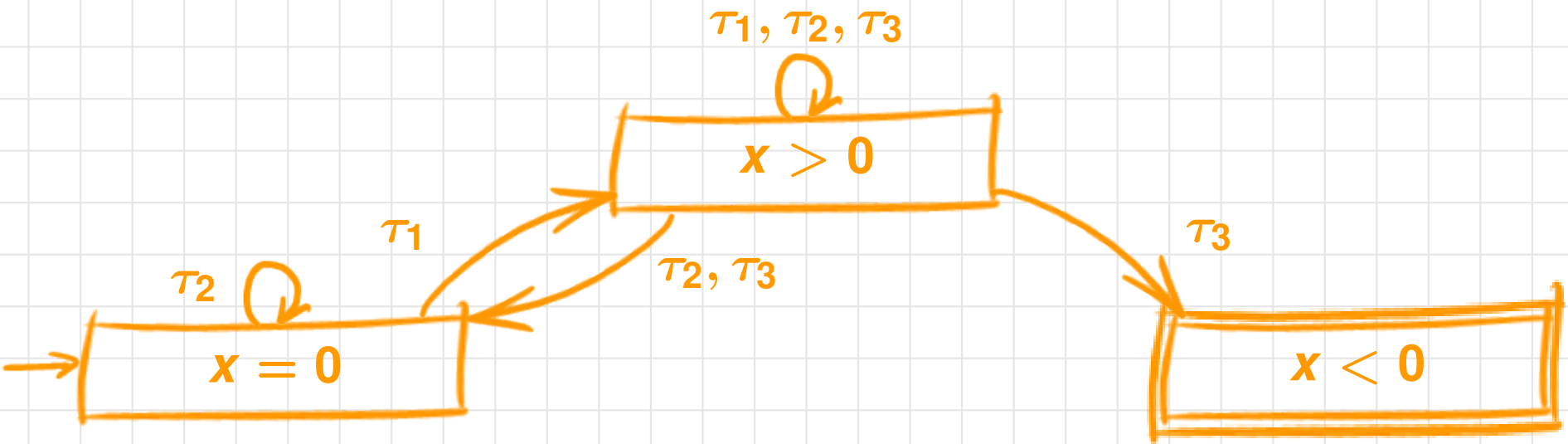
$\tau_1 : x' = 1 \wedge y' = x + 1$

$\tau_2 : y' = x - 1 \wedge x' = x$

$\tau_3 : x \geq 1 \wedge x' = y \wedge y' = y$

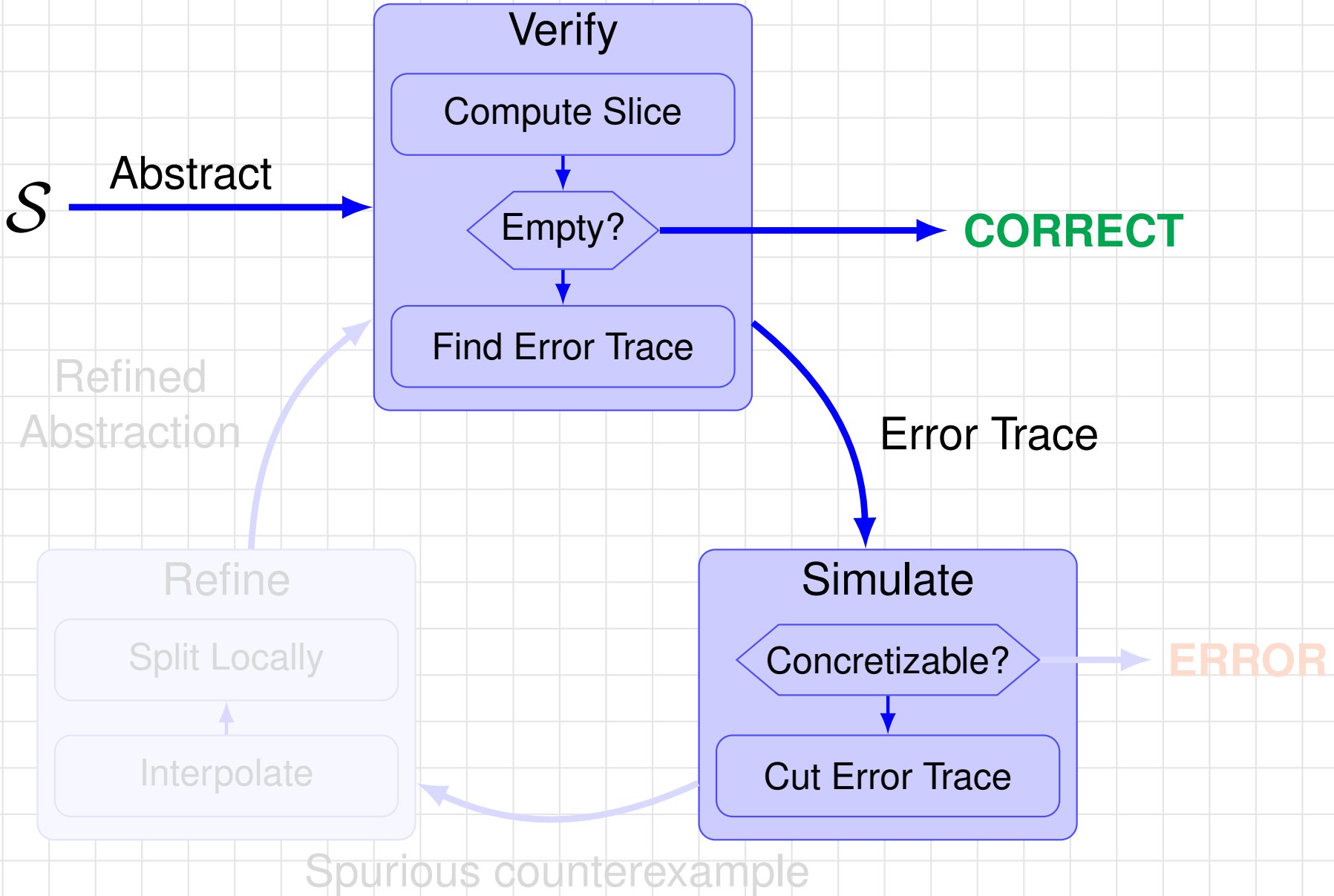
$init : x = 0$

$error : x < 0$

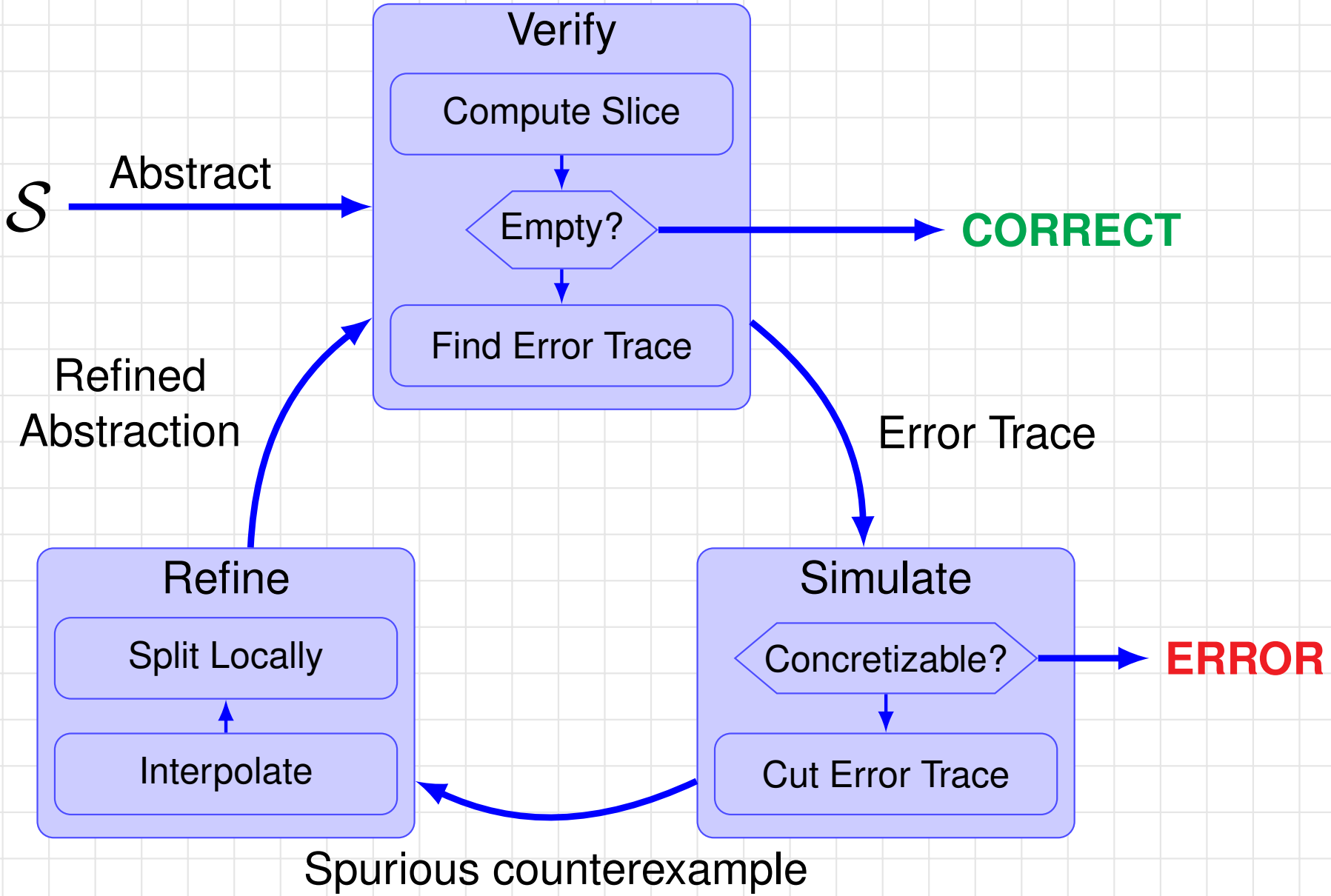


$$\Phi = \underbrace{x^0 = 0}_{} \wedge \underbrace{x^1 > 0}_{} \wedge \underbrace{x^2 < 0}_{} \wedge \underbrace{x^1 = 1 \wedge y^1 = x^0 + 1}_{} \wedge \underbrace{x^1 \geq 1 \wedge x^2 = y^1 \wedge y^2 = y^1}_{}$$

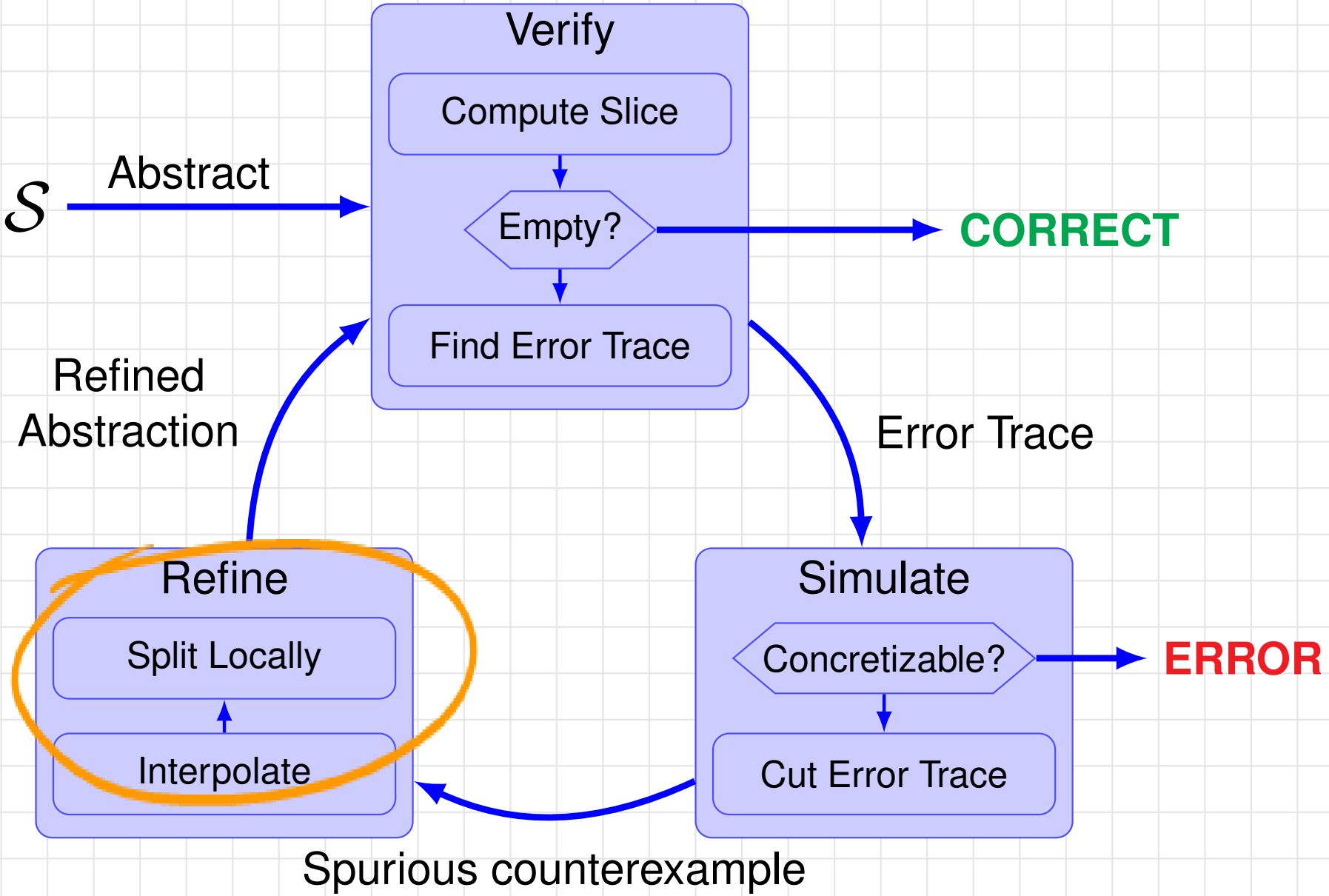
Slicing Abstraction Refinement



Slicing Abstraction Refinement



Slicing Abstraction Refinement



Abstraction Refinement

Abstraction Refinement

Spurious counterexample

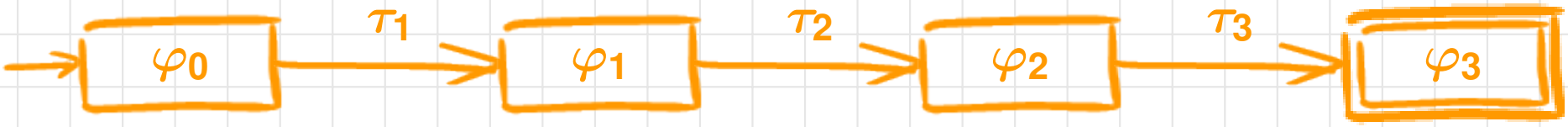
Abstraction Refinement

Spurious counterexample

$$\Phi = \varphi_0(V^0) \wedge \tau_1(V^0, V^1) \wedge \varphi_1(V^1) \wedge \tau_2(V^1, V^2) \wedge \varphi_2(V^2) \wedge \tau_3(V^2, V^3) \wedge \varphi_3(V^3)$$

Abstraction Refinement

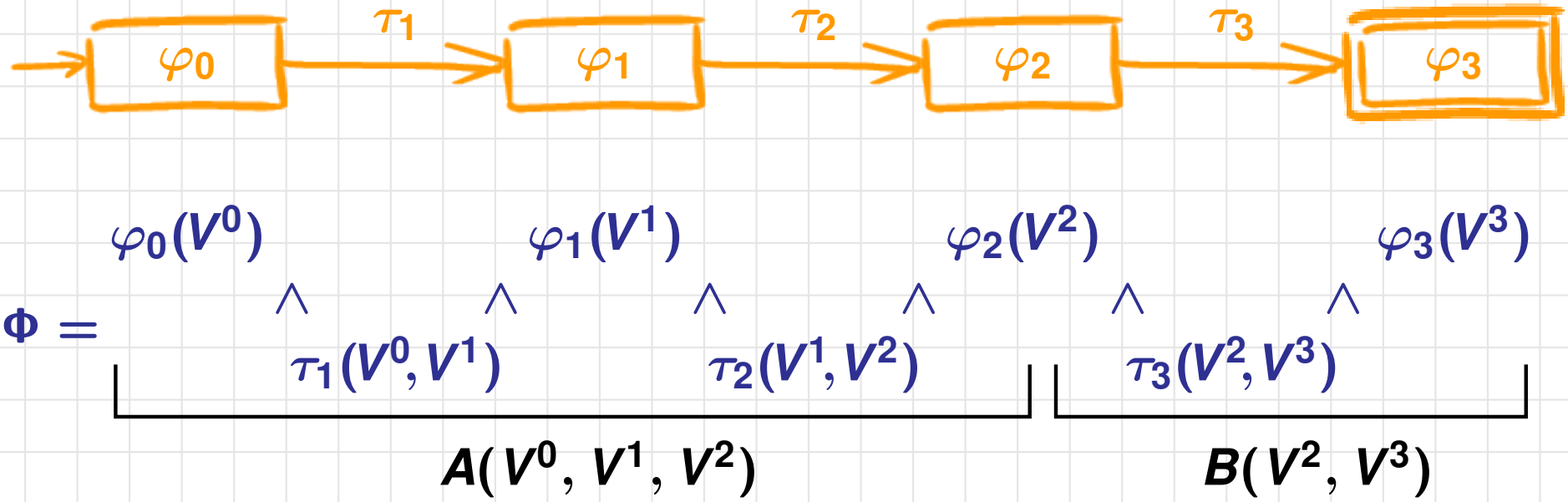
Spurious counterexample



$$\Phi = \underbrace{\varphi_0(V^0) \wedge \tau_1(V^0, V^1) \wedge \varphi_1(V^1) \wedge \tau_2(V^1, V^2) \wedge \varphi_2(V^2)}_{A(V^0, V^1, V^2)} \wedge \underbrace{\tau_3(V^2, V^3) \wedge \varphi_3(V^3)}_{B(V^2, V^3)}$$

Abstraction Refinement

Spurious counterexample

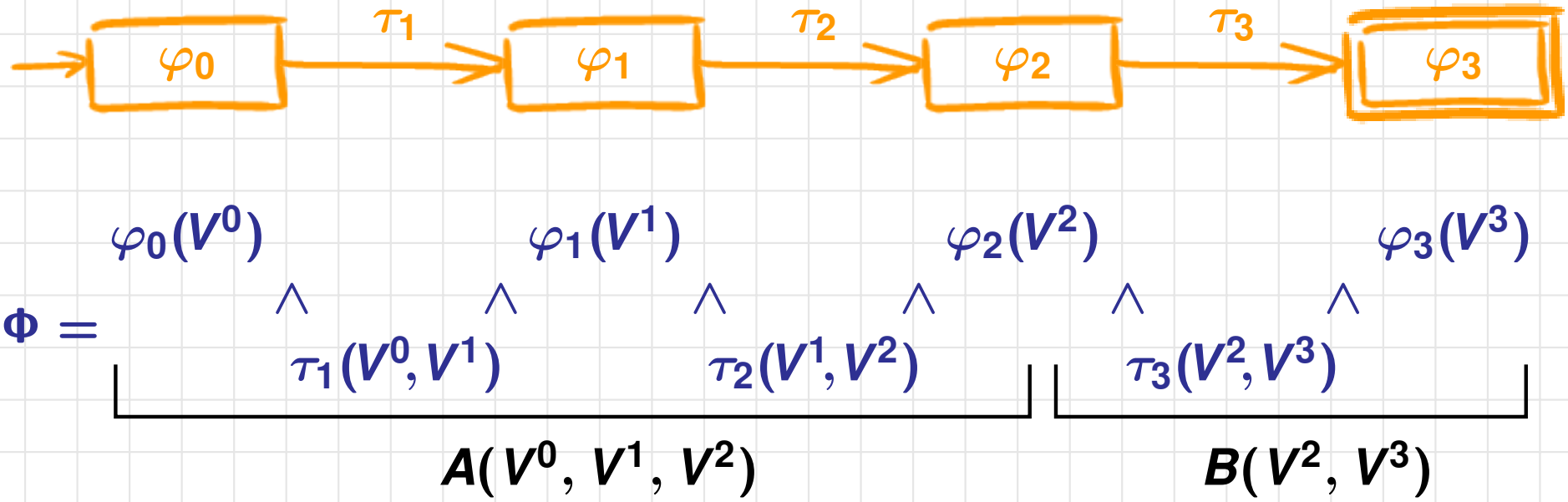


$A \wedge B$ unsat, but A, B sat $\rightsquigarrow$ Craig interpolant η:

- $A \models \eta$
- $B \models \neg\eta$
- $Var(\eta) \subseteq Var(A) \cap Var(B)$

Abstraction Refinement

Spurious counterexample



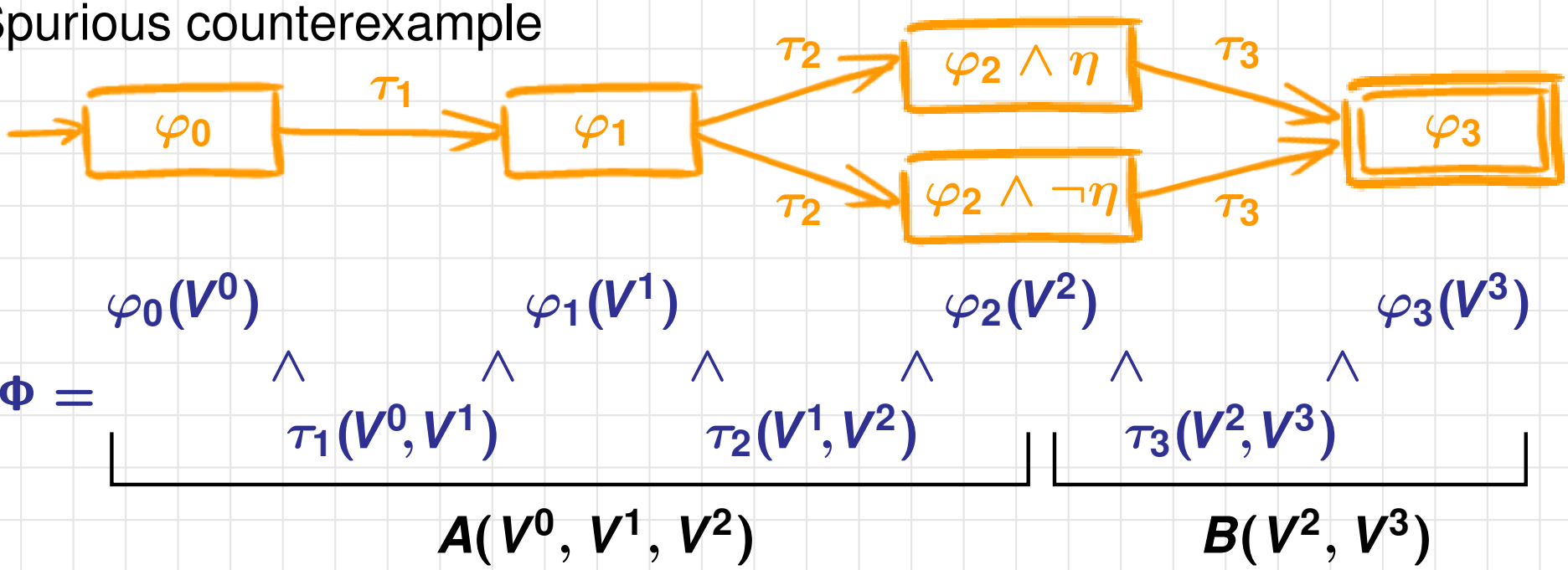
$A \wedge B$ unsat, but A, B sat $\rightsquigarrow$ Craig interpolant η:

- $A \models \eta$
- $B \models \neg\eta$
- $Var(\eta) \subseteq Var(A) \cap Var(B)$

$\rightsquigarrow$ split locally with $\eta, \neg\eta$

Abstraction Refinement

Spurious counterexample



$A \wedge B$ unsat, but A, B sat $\rightsquigarrow$ Craig interpolant η:

- $A \models \eta$
- $B \models \neg\eta$
- $Var(\eta) \subseteq Var(A) \cap Var(B)$

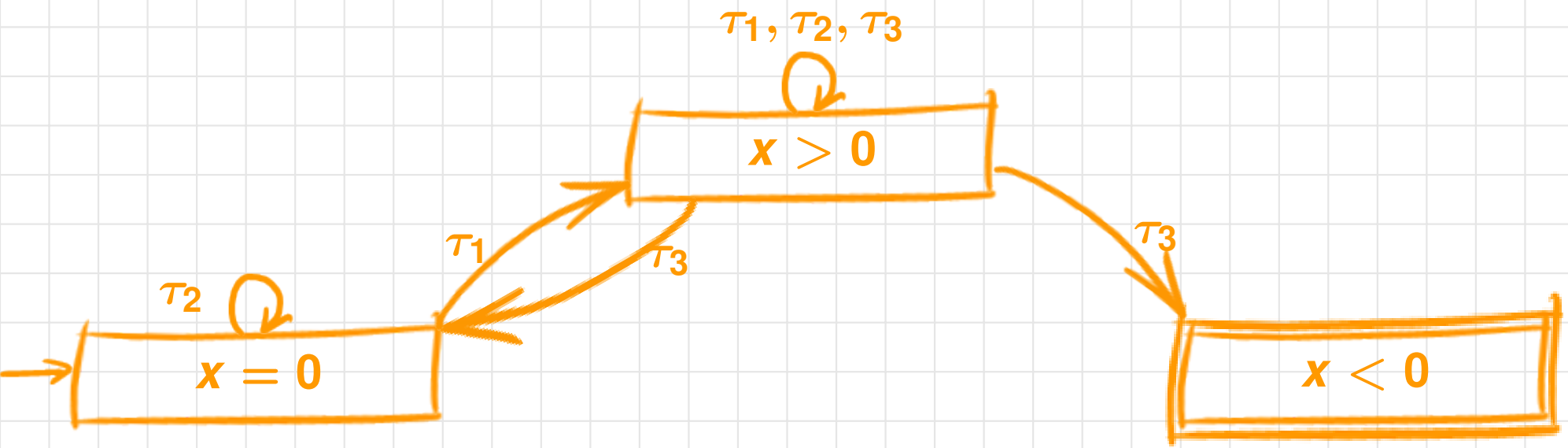
$\rightsquigarrow$ split locally with $\eta, \neg\eta$

Example: Abstraction Refinement

$\tau_1 : x' = 1 \wedge y' = x + 1$

$init : x = 0$ $\quad \tau_2 : y' = x - 1 \wedge x' = x$ $\quad error : x < 0$

$\tau_3 : x \geq 1 \wedge x' = y \wedge y' = y$



$\Phi = x^0 = 0 \wedge x^1 > 0 \wedge x^2 < 0 \wedge x^1 = 1 \wedge y^1 = x^0 + 1 \wedge x^1 \geq 1 \wedge x^2 = y^1 \wedge y^2 = y^1$

Example: Abstraction Refinement

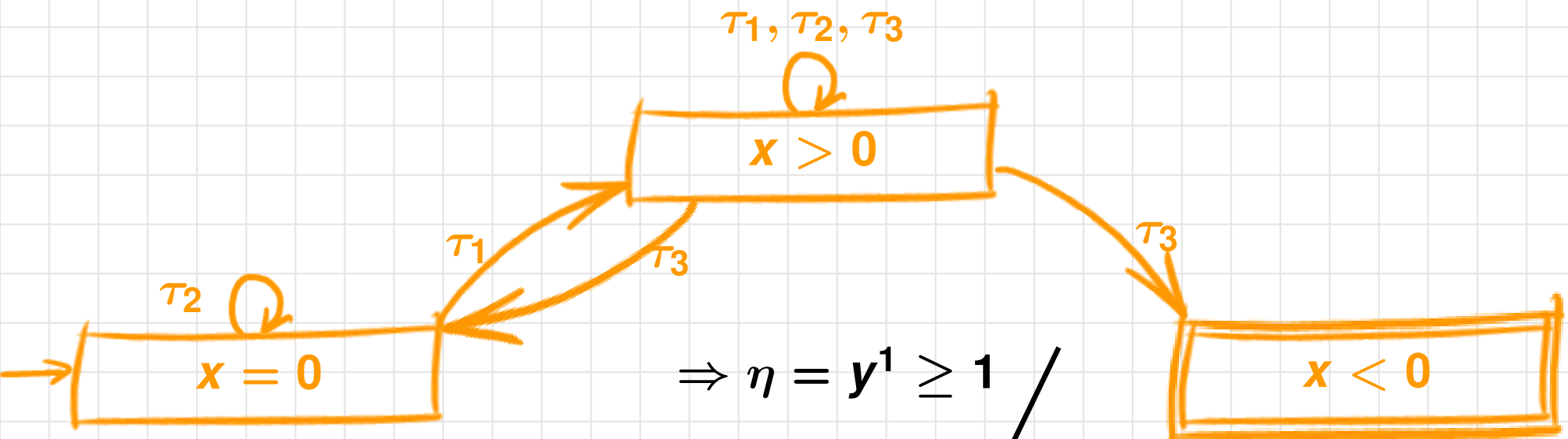
$\tau_1 : x' = 1 \wedge y' = x + 1$

$\tau_2 : y' = x - 1 \wedge x' = x$

$\tau_3 : x \geq 1 \wedge x' = y \wedge y' = y$

$init : x = 0$

$error : x < 0$



$\Rightarrow \eta = y^1 \geq 1$

$x^0 = 0 \quad x^1 > 0 \quad x^2 < 0$

$\Phi = \quad \wedge \quad \wedge \quad \wedge \quad \wedge$

$x^1 = 1 \wedge y^1 = x^0 + 1 \qquad x^1 \geq 1 \wedge x^2 = y^1 \wedge y^2 = y^1$

Example: Abstraction Refinement

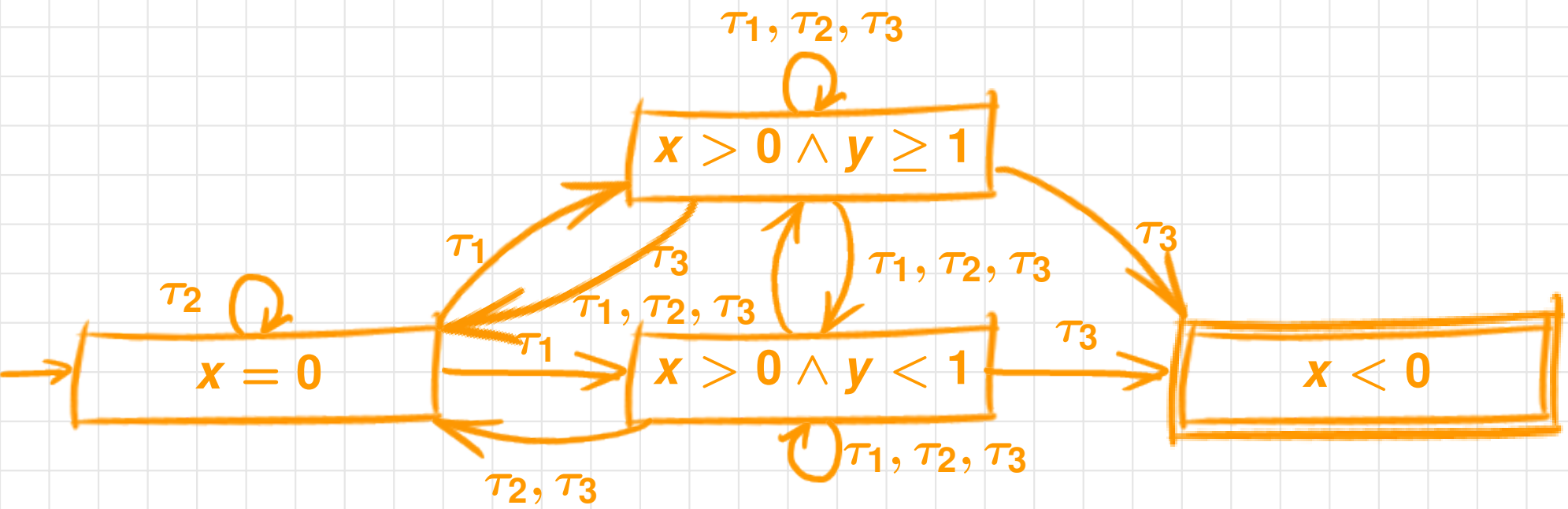
$\tau_1 : x' = 1 \wedge y' = x + 1$

$\tau_2 : y' = x - 1 \wedge x' = x$

$\tau_3 : x \geq 1 \wedge x' = y \wedge y' = y$

$init : x = 0$

$error : x < 0$



Example: Slicing (2)

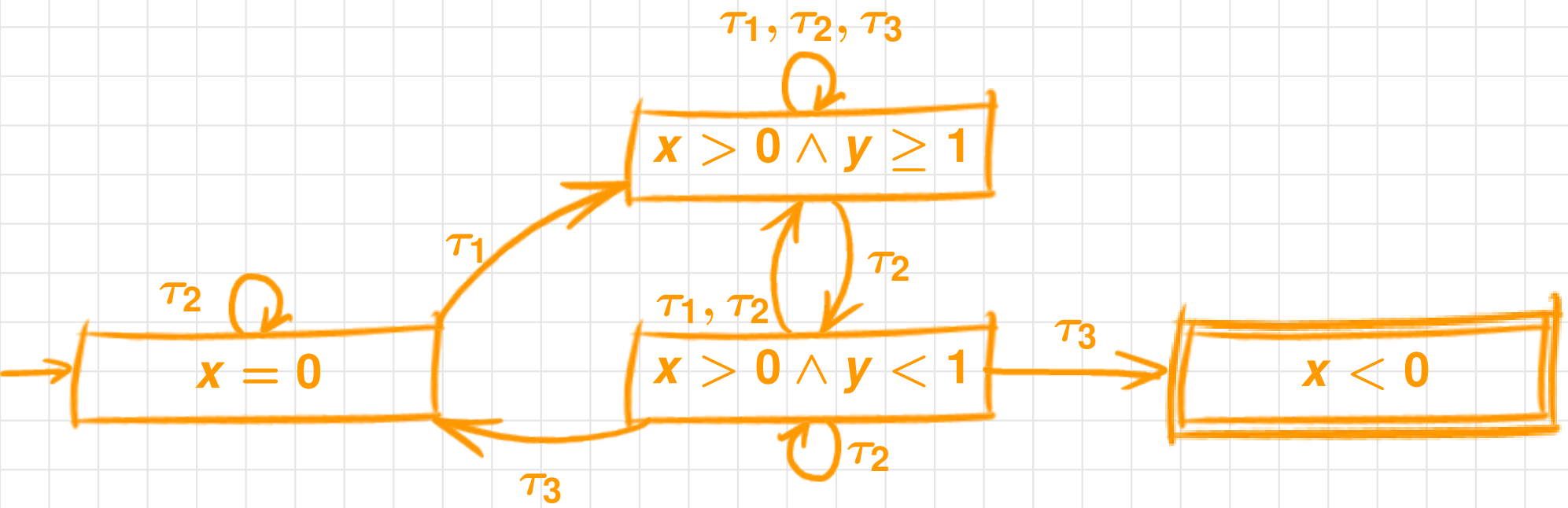
$\tau_1 : x' = 1 \wedge y' = x + 1$

$\tau_2 : y' = x - 1 \wedge x' = x$

$\tau_3 : x \geq 1 \wedge x' = y \wedge y' = y$

$init : x = 0$

$error : x < 0$



Example: Abstraction Refinement (2)

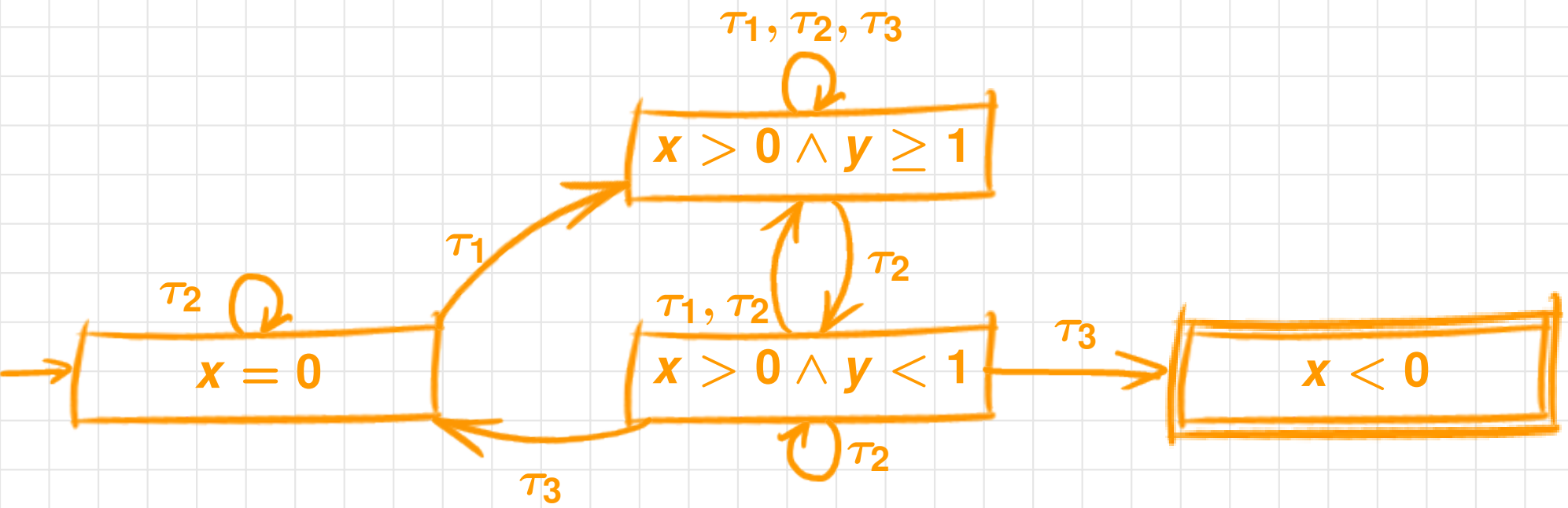
$\tau_1 : x' = 1 \wedge y' = x + 1$

$\tau_2 : y' = x - 1 \wedge x' = x$

$\tau_3 : x \geq 1 \wedge x' = y \wedge y' = y$

$init : x = 0$

$error : x < 0$



Example: Abstraction Refinement (2)

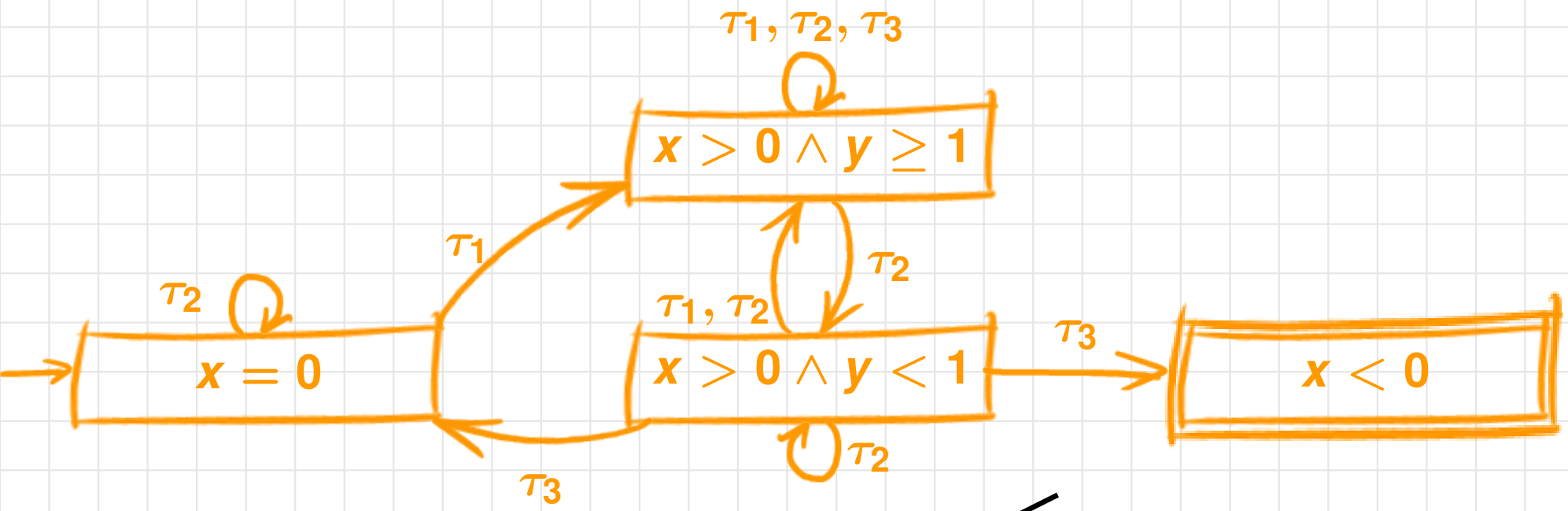
$\tau_1 : x' = 1 \wedge y' = x + 1$

$\tau_2 : y' = x - 1 \wedge x' = x$

$\tau_3 : x \geq 1 \wedge x' = y \wedge y' = y$

$init : x = 0$

$error : x < 0$



$\Rightarrow \eta = y^2 \geq 0$

Example: Abstraction Refinement (2)

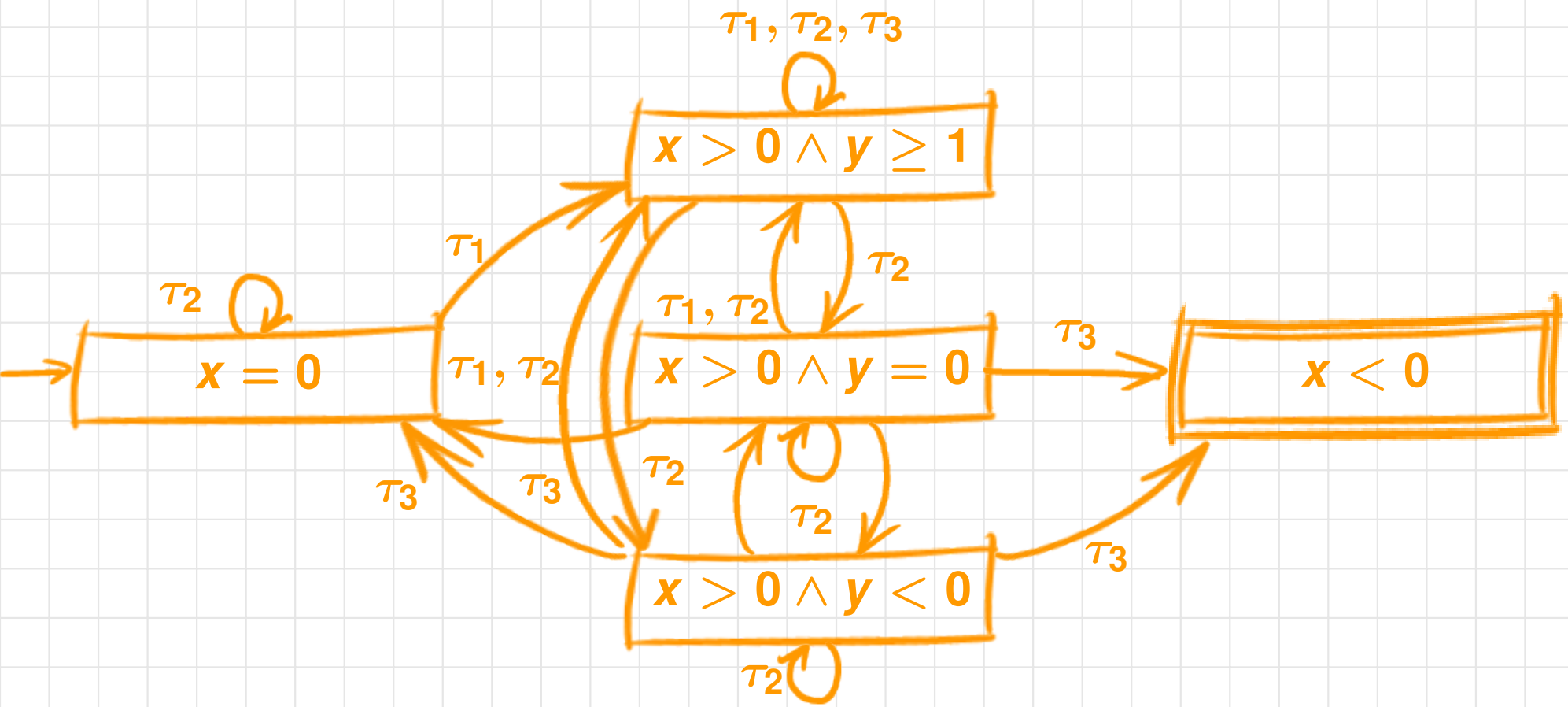
$\tau_1 : x' = 1 \wedge y' = x + 1$

$\tau_2 : y' = x - 1 \wedge x' = x$

$\tau_3 : x \geq 1 \wedge x' = y \wedge y' = y$

$init : x = 0$

$error : x < 0$



Example: Slicing (3)

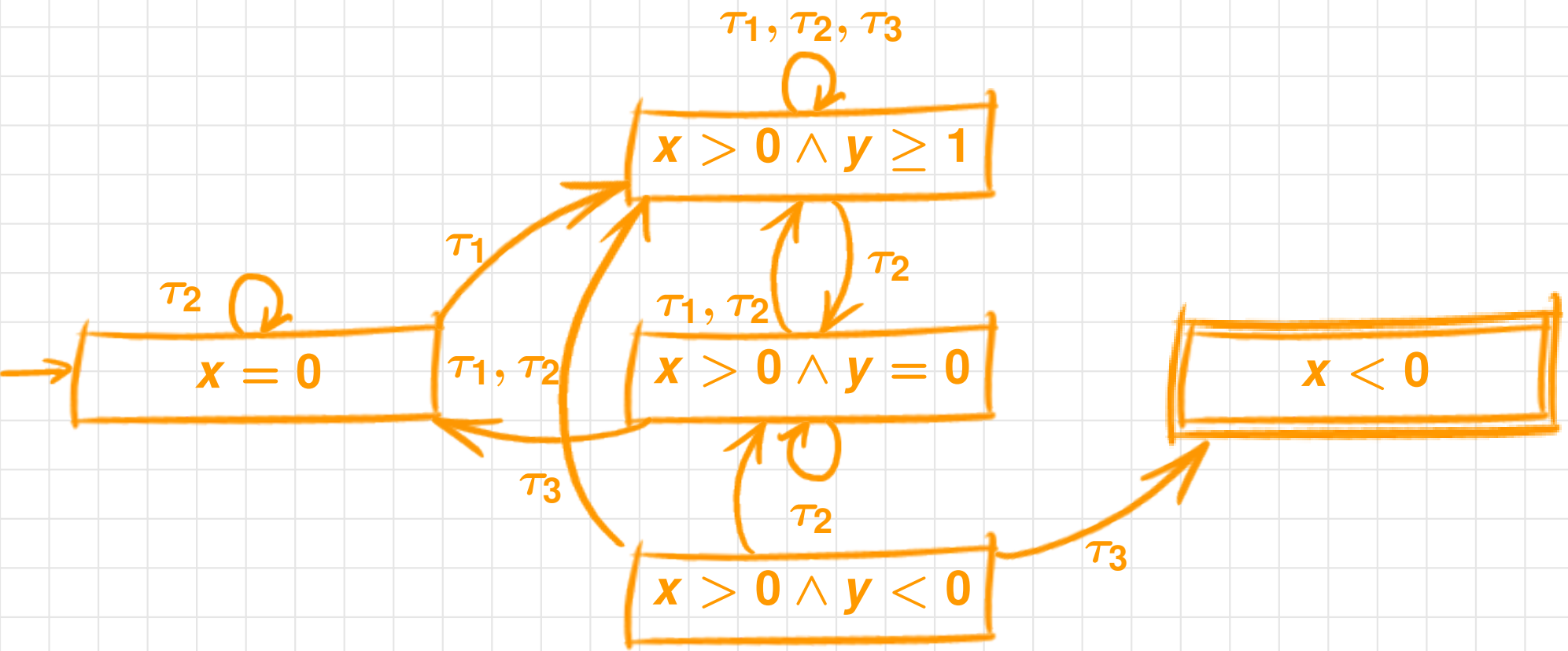
$\tau_1 : x' = 1 \wedge y' = x + 1$

$\tau_2 : y' = x - 1 \wedge x' = x$

$\tau_3 : x \geq 1 \wedge x' = y \wedge y' = y$

$init : x = 0$

$error : x < 0$



Example: Slicing (3)

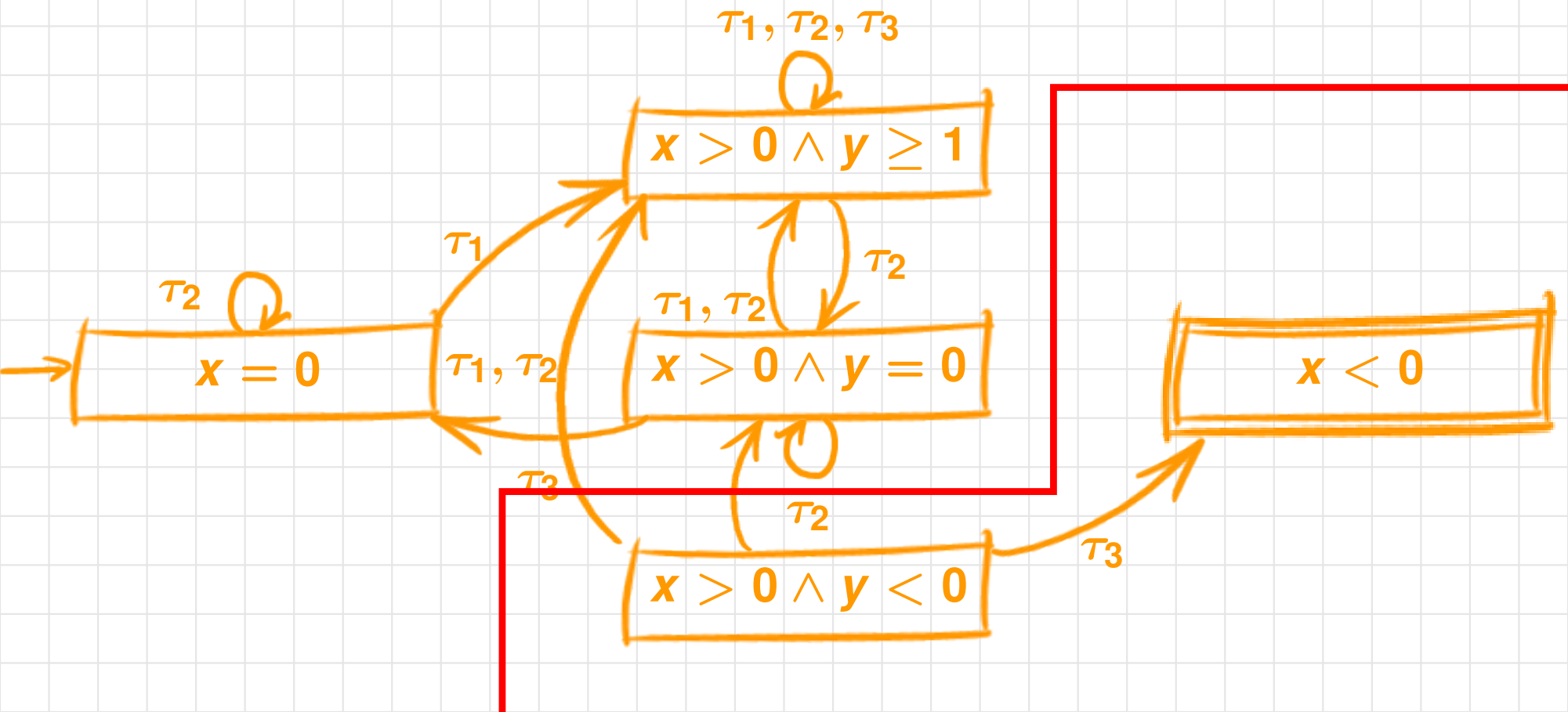
$\tau_1 : x' = 1 \wedge y' = x + 1$

$\tau_2 : y' = x - 1 \wedge x' = x$

$\tau_3 : x \geq 1 \wedge x' = y \wedge y' = y$

$init : x = 0$

$error : x < 0$



Example: Certificate

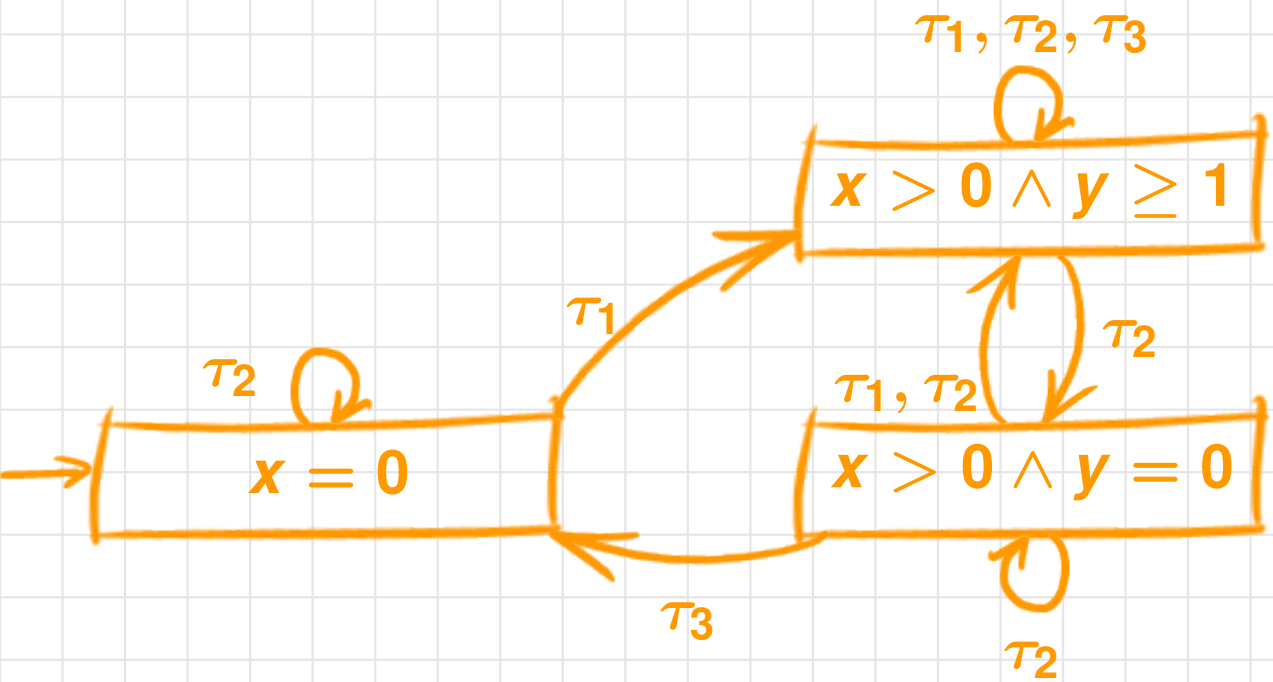
$\tau_1 : x' = 1 \wedge y' = x + 1$

$\tau_2 : y' = x - 1 \wedge x' = x$

$\tau_3 : x \geq 1 \wedge x' = y \wedge y' = y$

$init : x = 0$

$error : x < 0$



Demo

Summary

Summary

SLAB is a Model Checker for:

- **infinite state** (real time, infinite datatypes, ...)
- **concurrent systems** (communication protocols, control systems,...)
- which can produce **certificates** if the system is correct

Summary

SLAB is a Model Checker for:

- **infinite state** (real time, infinite datatypes, ...)
- **concurrent systems** (communication protocols, control systems,...)
- which can produce **certificates** if the system is correct

Certificates

- In graphical format
- In SMT-LIB format
- Independently checkable by standard SMT solvers
- Coming soon: interface with Coq theorem prover

Summary

SLAB is a Model Checker for:

- **infinite state** (real time, infinite datatypes, ...)
- **concurrent systems** (communication protocols, control systems,...)
- which can produce **certificates** if the system is correct

Certificates

- In graphical format
- In SMT-LIB format
- Independently checkable by standard SMT solvers
- Coming soon: interface with Coq theorem prover

Thank you for attention!