Property Directed Abstract Interpretation

Noam Rinetzky Sharon Shoham

Safety Verification

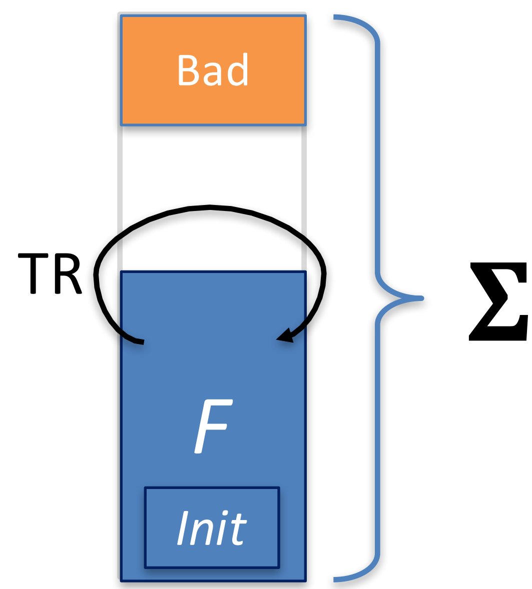
- Verification problem: ($Init, P, Bad$)
  - $Init \subseteq \Sigma$
  - $\mathrm{TR} = \mathrm{TRof}(P) \subseteq \Sigma \times \Sigma$
  - $Bad \subseteq \Sigma$

- Can P reach Bad starting from $Init$?
  - Yes (P is not safe)
  - No (P is safe)

Inductive Invariants

- $F \subseteq \Sigma$ is an inductive invariant if:
  - $Init \subseteq F$
  - $\mathrm{TR}(F) \subseteq F$
  - $F \cap Bad = \emptyset$

- *P* is safe $\Leftrightarrow \exists$ inductive invariant



Property Directed Reachability (PDR)

- IC3 [Bradley, VMCAI'11]
- PDR [Een, Mishchenko & Brayton, FMCAD'11]

- IC3/PDR algorithm
  - Infers inductive invariants
  - SAT-based
  - Iterative

Property Directed Reachability (PDR)

- [Bradley, VMCAI'11]
- [Een, Mishchenko & Brayton, FMCAD'11]

- "Dynamic" abstraction
  - Over-approximates bounded executions
  - Abstraction refined as bound grows
  - No spurious cex s.t. $|cex| \leq$ bound

- Output
  - Inductive invariant
  - cex

PDR-based Algorithms

- [Bradley, VMCAI'11]
- [Een, Mishchenko & Brayton, FMCAD'11]
- [Cimatti & Griggio, CAV'12]
- [Hoder & Bjorner, SAT'12]
- [Bjorner & Gurfinkel, VMCAI'15]
- [Karbyshev, Bjorner, Itzhaky, R & Shoham, CAV'15]
- …

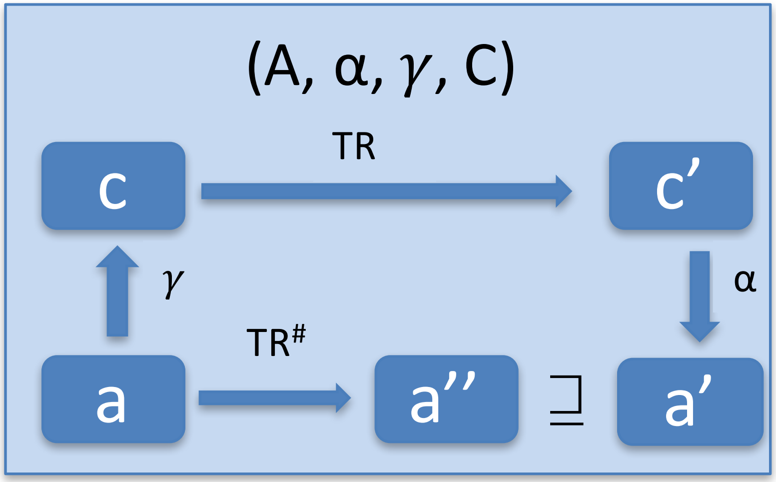
- Successfully applied to software & hardware
- Varied algorithmic details

Our Goal

- Extract essence of PDR
  - Hide algorithmic details
  - Show commonality
  - A unified proof of soundness

Technical Attack

- Abstract Interpretation (AI)
  - Hide algorithmic details
  - Analysis = abstract semantics
    - Over-approximate concrete semantics
  - Sound by construction



- **Our approach**: formulate **PDR** using AI

Main Results

- $[\![P]\!]^B_{\wp(\Omega)}$: Non-standard operational semantics
  - Can compute any inductive invariant
    - and any counterexample
- PDR algorithms interpret P using $[\![P]\!]^B_{\wp(\Omega)}$

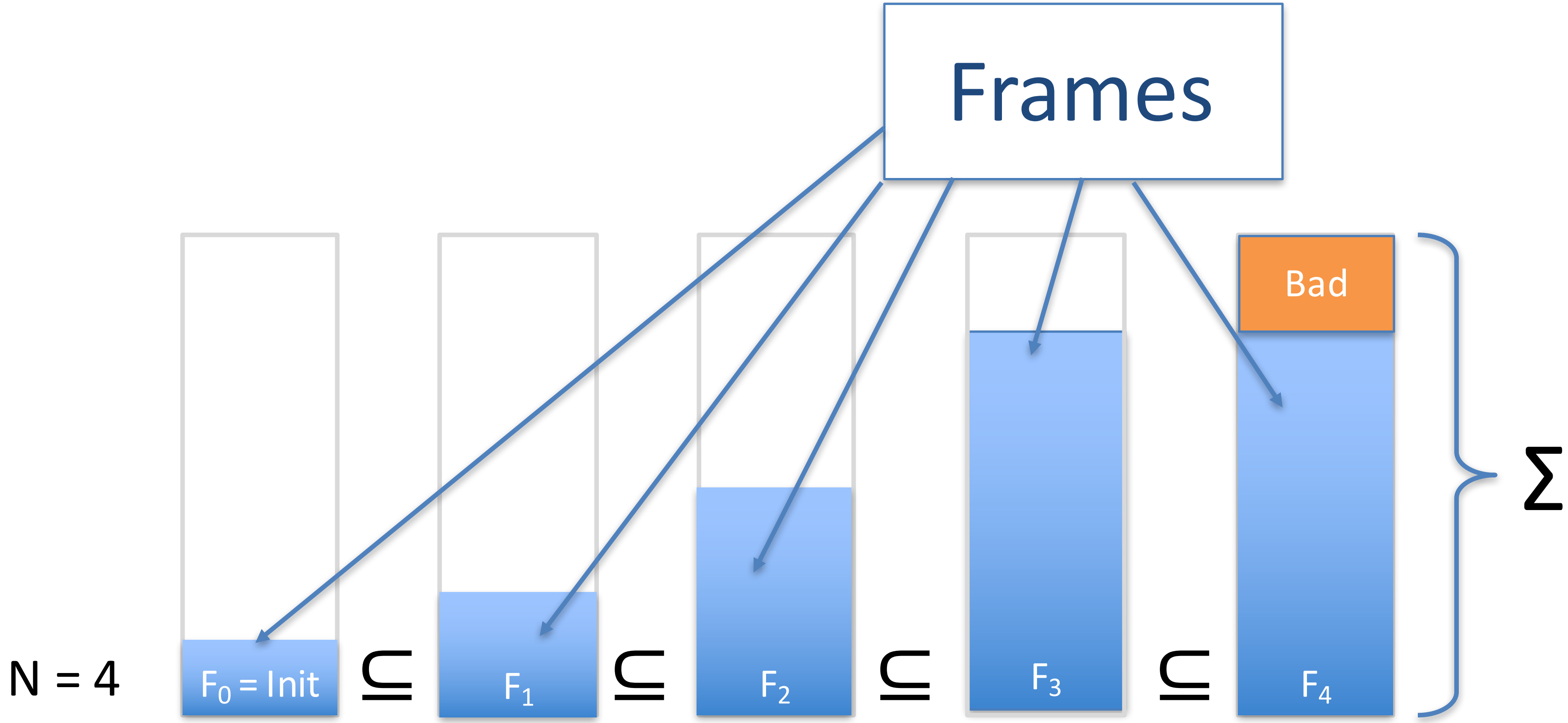
Plan

- PDR
- $[\![P]\!]^{B}_{\wp(\Omega)}$
- PDR as AI using $[\![P]\!]^{B}_{\wp(\Omega)}$

*No SAT in this presentation

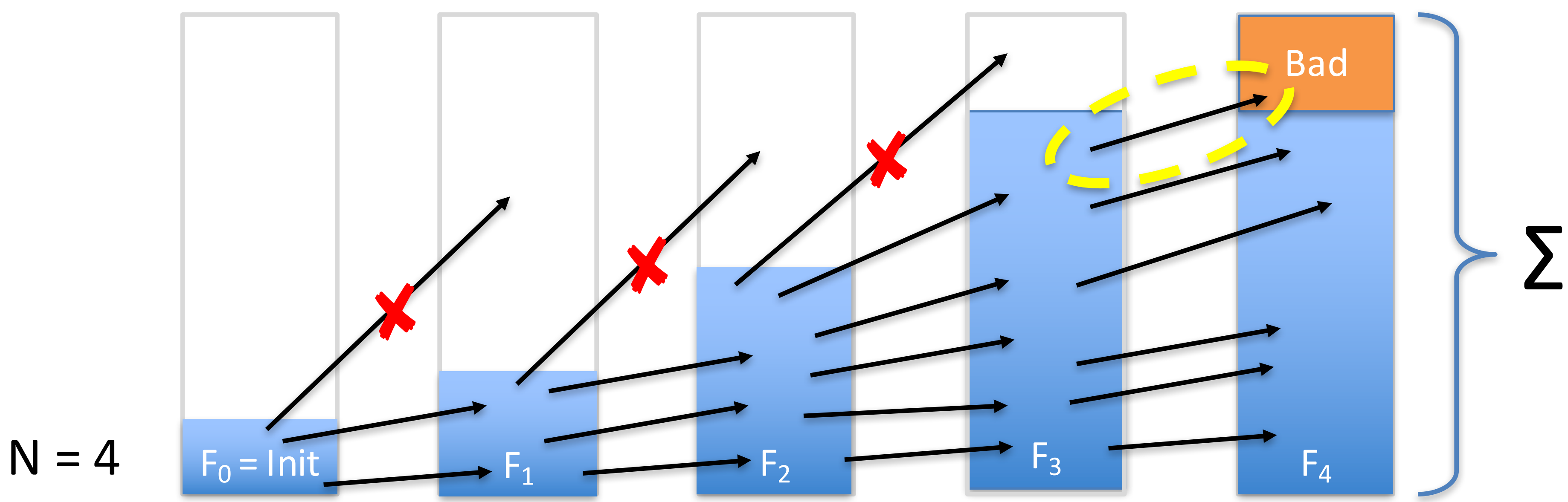
PDR: Intermediate Forward Reachability Sequences

- $\varphi = \langle F_0, \ldots, F_N \rangle$



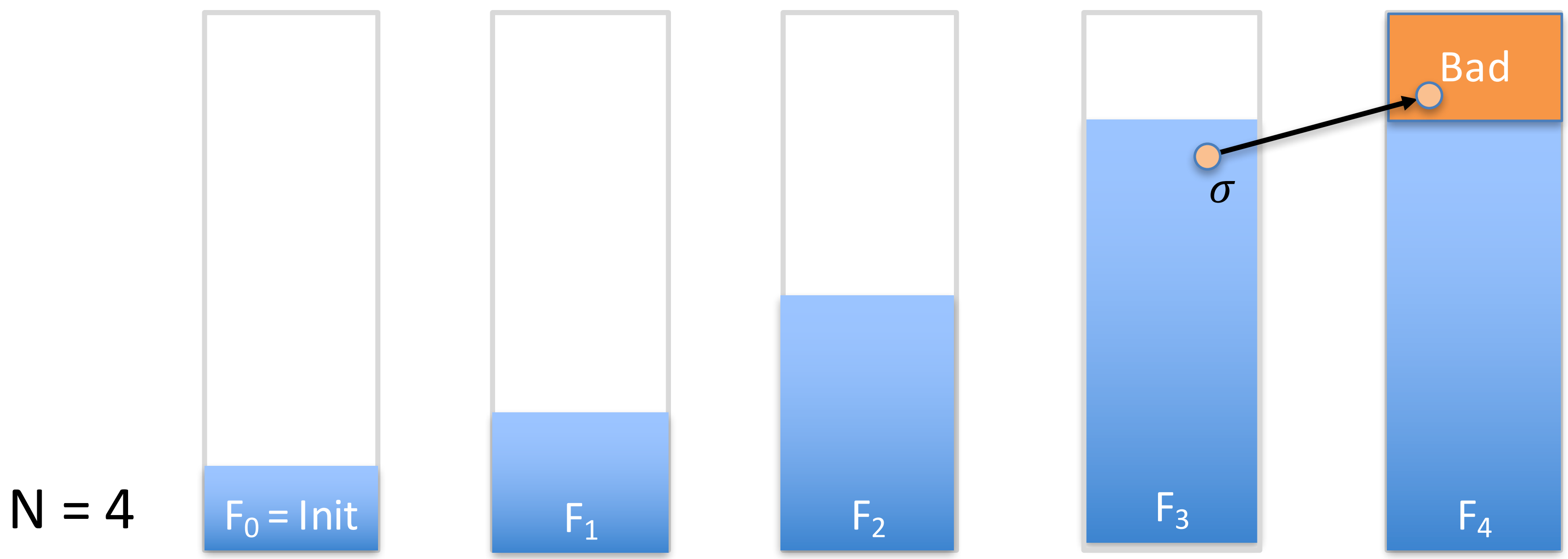
PDR: Intermediate Forward Reachability Sequences

- $\varphi = \langle F_0, \ldots, F_N \rangle$
  - $F_i \cap Bad = \emptyset$
  - $F_0 = Init,\ F_i \subseteq F_{i+1}$
  - $\mathrm{TR}(F_i) \subseteq F_{i+1}$ **(i + 1 < N)**



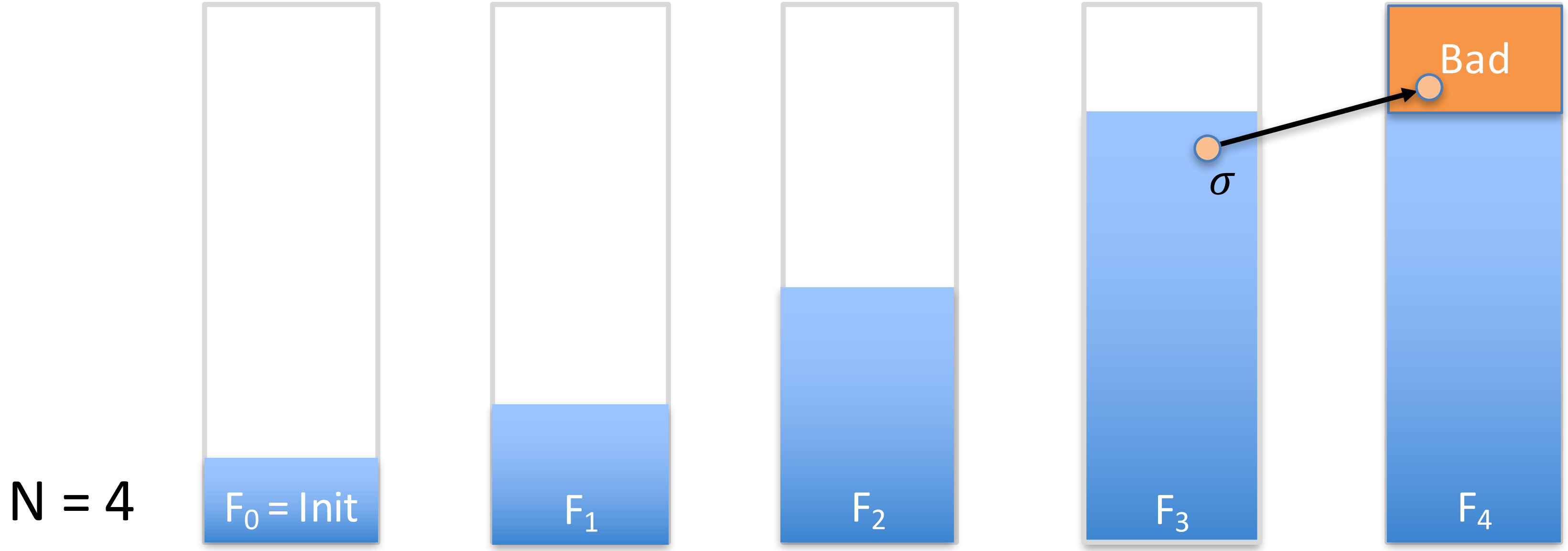
PDR: Obligation Queue

- $\varphi = \langle F_0, \ldots, F_N \rangle$
  - $F_i \cap Bad = \emptyset$
  - $F_0 = Init,\ F_i \subseteq F_{i+1}$
  - $\mathrm{TR}(F_i) \subseteq F_{i+1}$ **(i + 1 < N)**
- Q =[ ]



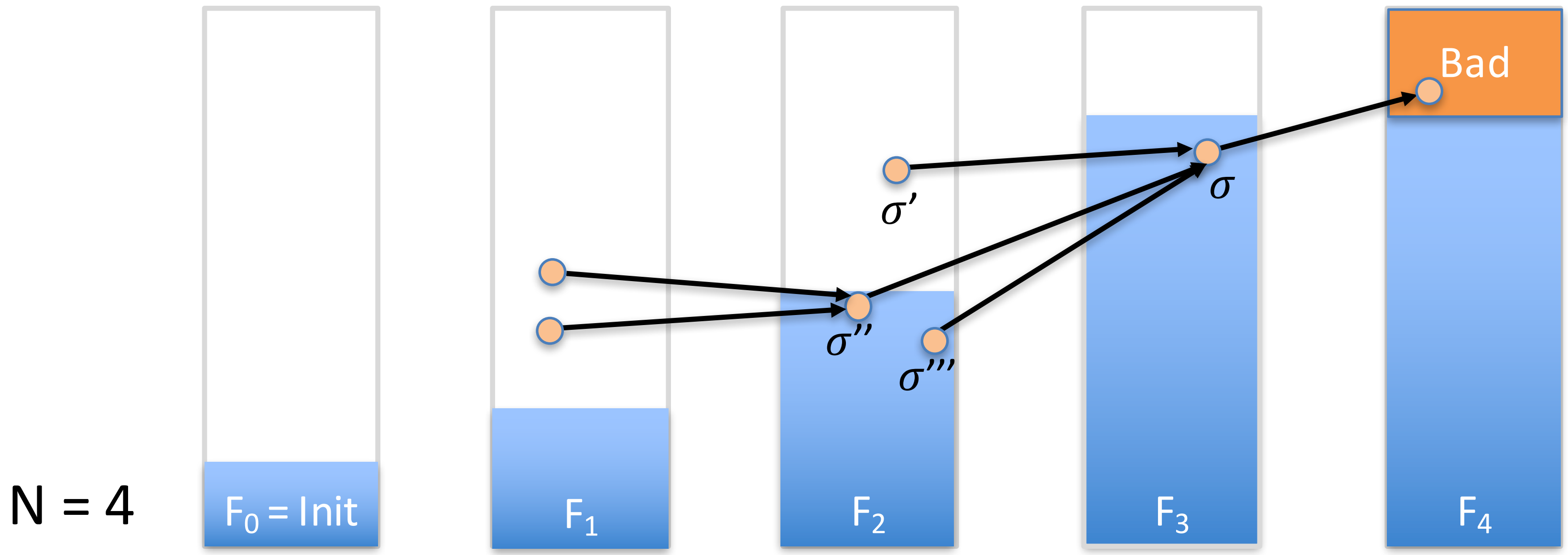
PDR: Queue Initialization

- $\varphi = \langle F_0, \dots, F_N \rangle$
  - $F_i \cap Bad = \emptyset$
  - $F_0 = Init,\ F_i \subseteq F_{i+1}$
  - $\mathrm{TR}(F_i) \subseteq F_{i+1}$ **(i + 1 < N)**
- Q =[$(\sigma,3)$]



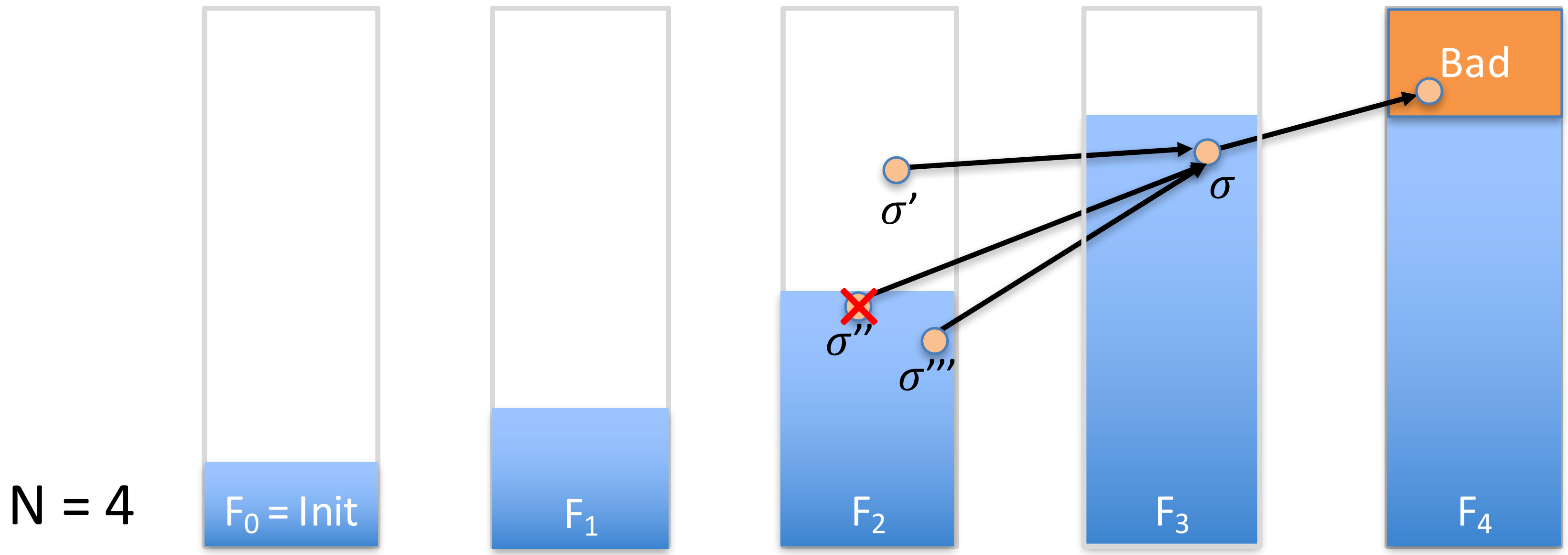
PDR: Backward Step

- $\varphi = \langle F_0, \ldots, F_N \rangle$
  - $F_i \cap Bad = \emptyset$
  - $F_0 = Init,\ F_i \subseteq F_{i+1}$
  - $\mathrm{TR}(F_i) \subseteq F_{i+1}$ **(i + 1 < N)**
- Q =[$(\sigma,3)$]



PDR: Blocking

- $\varphi = \langle F_0, \ldots, F_N \rangle$
  - $F_i \cap Bad = \emptyset$
  - $F_0 = Init,\ F_i \subseteq F_{i+1}$
  - $\mathrm{TR}(F_i) \subseteq F_{i+1}$ **(i + 1 < N)**
- Q =[$(\sigma,3)$]

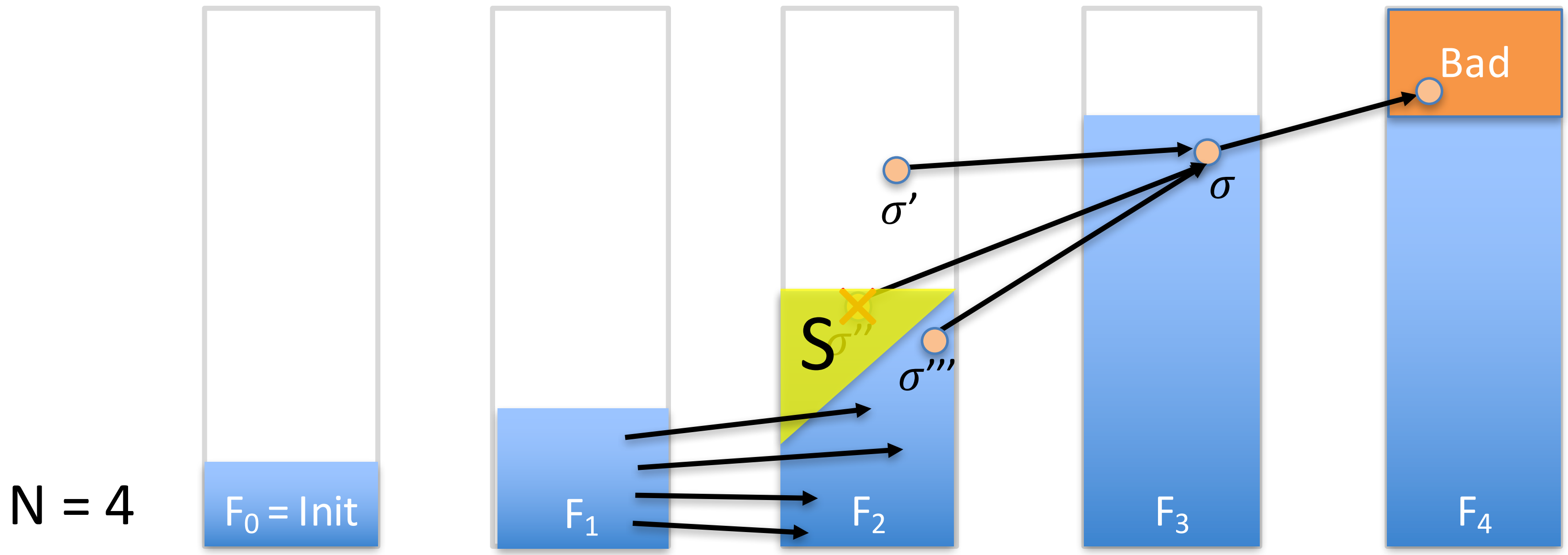


PDR: Generalization

- $\varphi = \langle F_0, \dots, F_N \rangle$
  - $F_i \cap Bad = \emptyset$
  - $F_0 = Init,\ \ F_i \subseteq F_{i+1}$
  - $\mathrm{TR}(F_i) \subseteq F_{i+1}$ **(i + 1 < N)**

S = Gen(F_{i-1},σ):
  S ∩ *Init* = ∅
  S ∩ TR(F_{i-1}) = ∅

- Q =[(σ,3)]

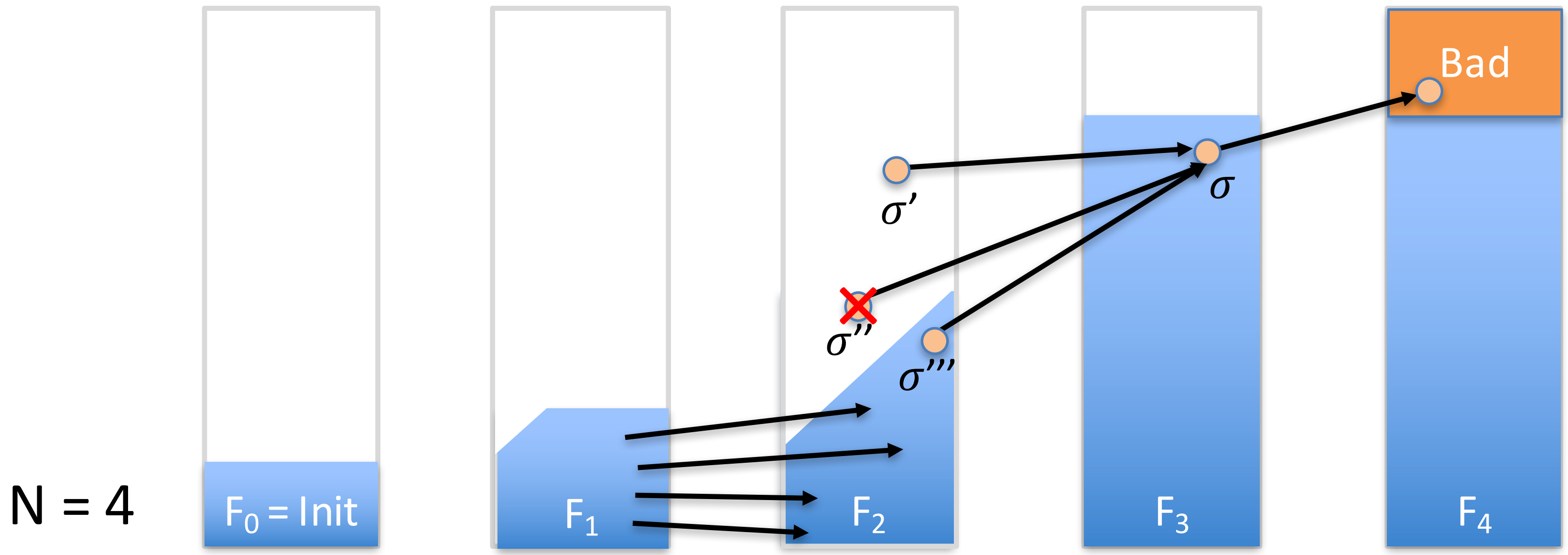


PDR: Generalization

- $\varphi = \langle F_0, \dots, F_N \rangle$
  - $F_i \cap Bad = \emptyset$
  - $F_0 = Init,\ \ F_i \subseteq F_{i+1}$
  - $\mathrm{TR}(F_i) \subseteq F_{i+1}$ **(i + 1 < N)**

S = Gen(F_{i-1},σ):
  S ∩ *Init* = ∅
  S ∩ TR(F_{i-1}) = ∅

- Q =[(σ,3)]

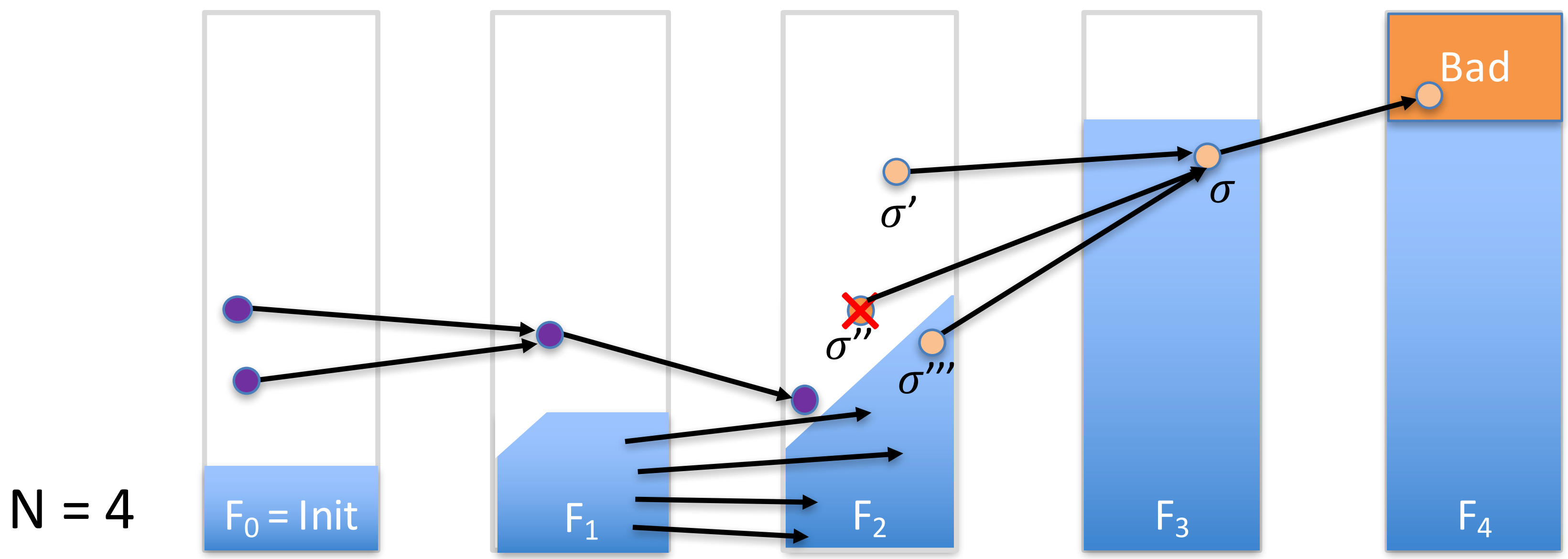


PDR: Generalization

- $\varphi = \langle F_0, \ldots, F_N \rangle$
  - $F_i \cap Bad = \emptyset$
  - $F_0 = Init,\ F_i \subseteq F_{i+1}$
  - $\mathrm{TR}(F_i) \subseteq F_{i+1}$ **(i + 1 < N)**

$S = \mathrm{Gen}(F_{i-1}, \sigma)$:
- $S \cap Init = \emptyset$
- $S \cap \mathrm{TR}(F_{i-1}) = \emptyset$

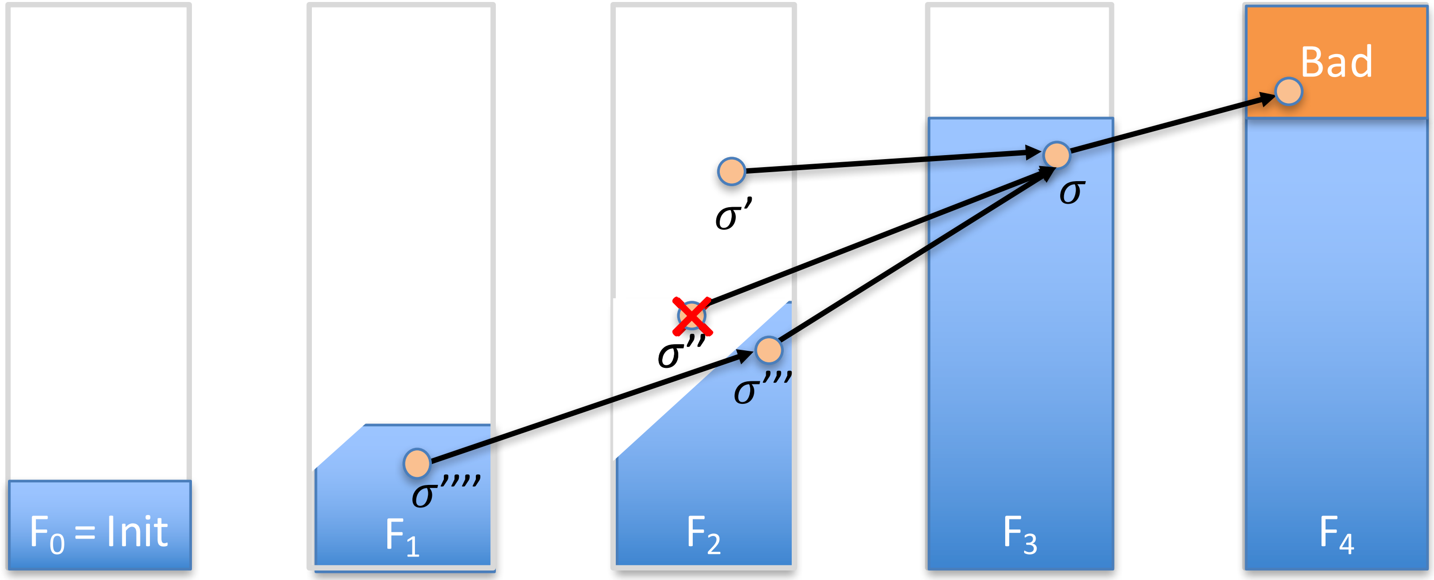
- Q =[$(\sigma,3)$]



PDR: Backward Step

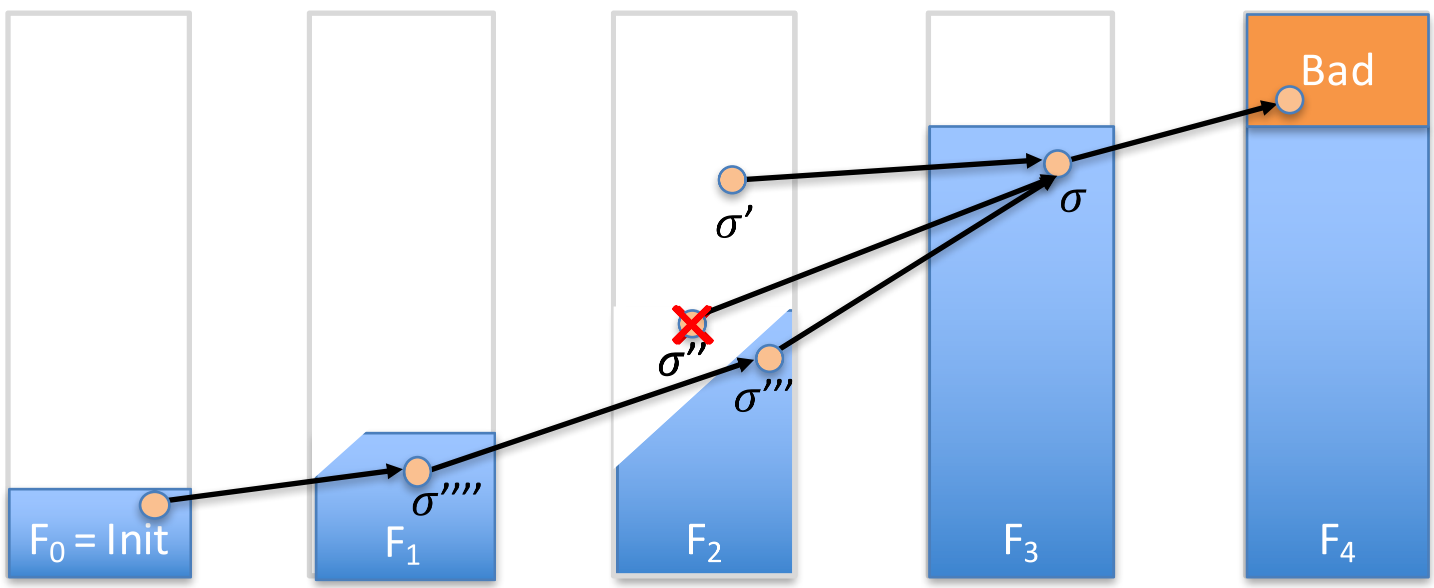
- $\varphi = \langle F_0, \ldots, F_N \rangle$
  - $F_i \cap Bad = \emptyset$
  - $F_0 = Init,\ \ F_i \subseteq F_{i+1}$
  - $\mathrm{TR}(F_i) \subseteq F_{i+1}$ **(i + 1 < N)**
- Q =[$(\sigma'''',1)$ $(\sigma,3)$]

N = 4



PDR: Counterexample

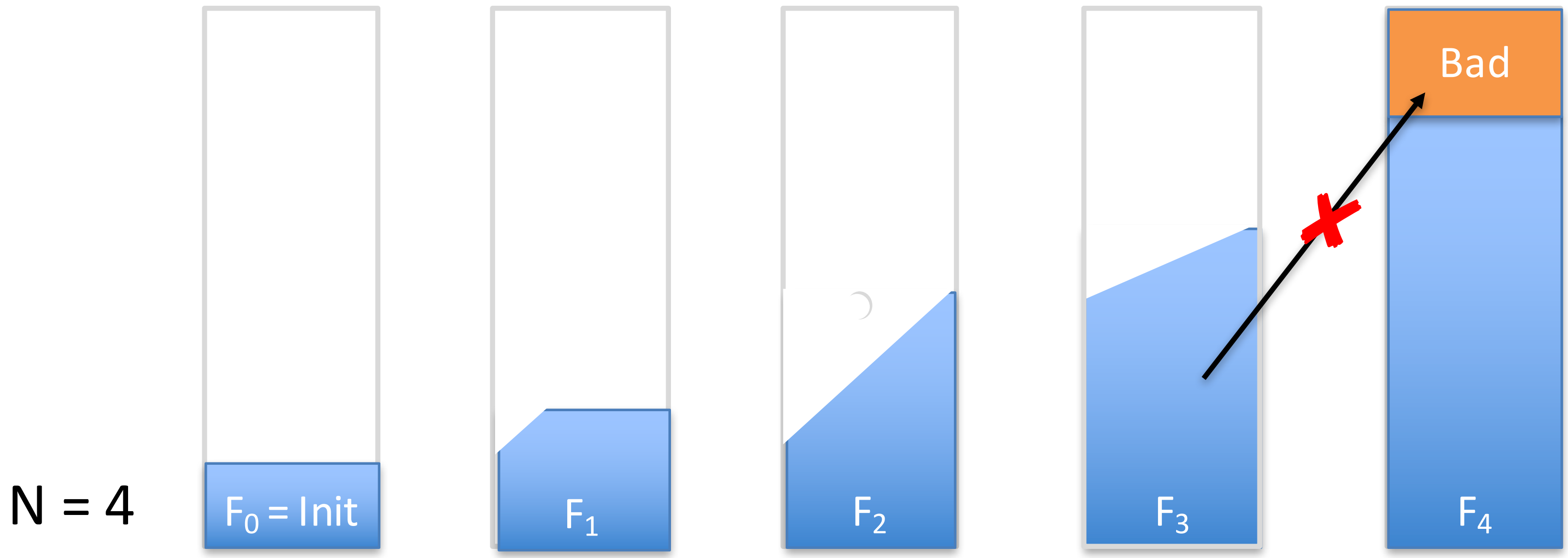
- $\varphi = \langle F_0, \ldots, F_N \rangle$
  - $F_i \cap Bad = \emptyset$
  - $F_0 = Init,\ \ F_i \subseteq F_{i+1}$
  - $\mathrm{TR}(F_i) \subseteq F_{i+1}$ **(i + 1 < N)**
- Q =[$(\sigma'''',1)$ $(\sigma''',2)$ $(\sigma,3)$]



N = 4

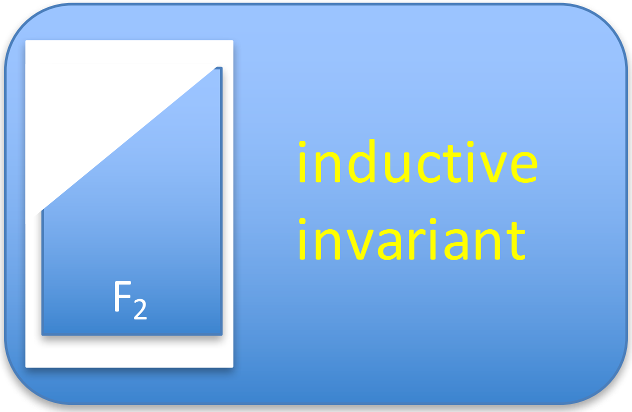
PDR: Forward Reachability Sequence

- $\varphi = \langle F_0, \ldots, F_N \rangle$
  - $F_i \cap Bad = \emptyset$
  - $F_0 = Init,\ \ F_i \subseteq F_{i+1}$
  - $\mathrm{TR}(F_i) \subseteq F_{i+1}$
- Q =[ ]



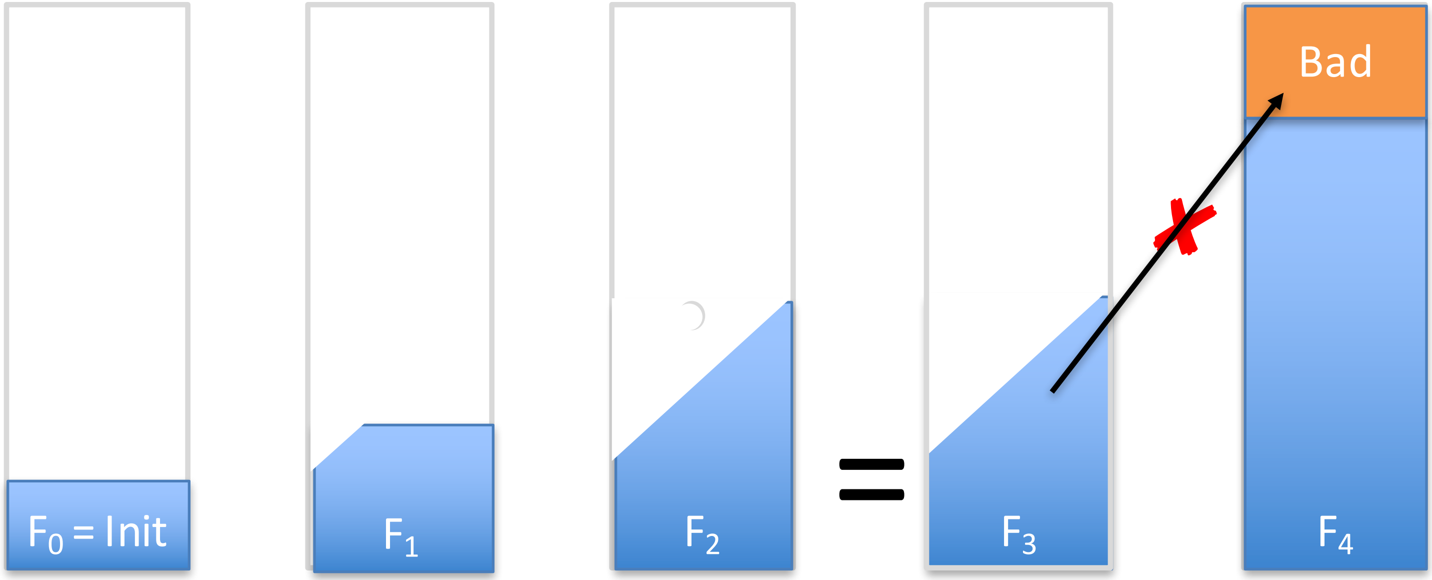
PDR: Fixpoint

- $\varphi = \langle F_0, \ldots, F_N \rangle$
  - $F_i \cap Bad = \emptyset$
  - $F_0 = Init,\ F_i \subseteq F_{i+1}$
  - $\mathrm{TR}(F_i) \subseteq F_{i+1}$



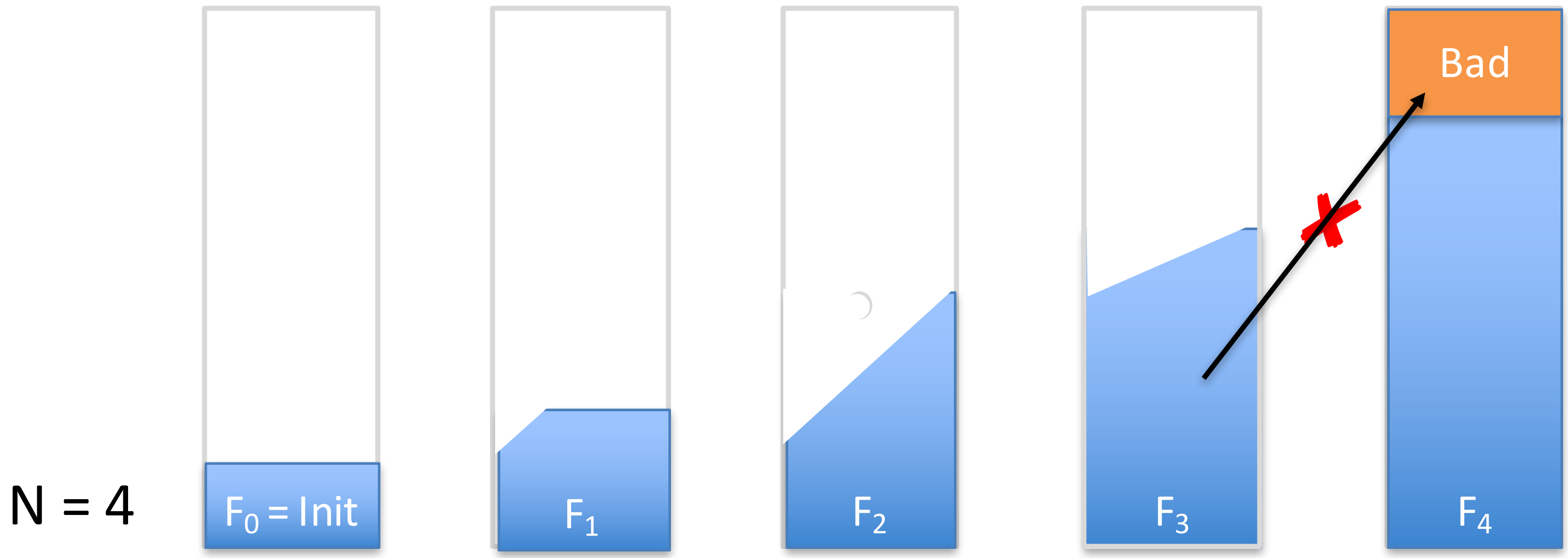
- Q =[ ]

N = 4



PDR: Unfolding

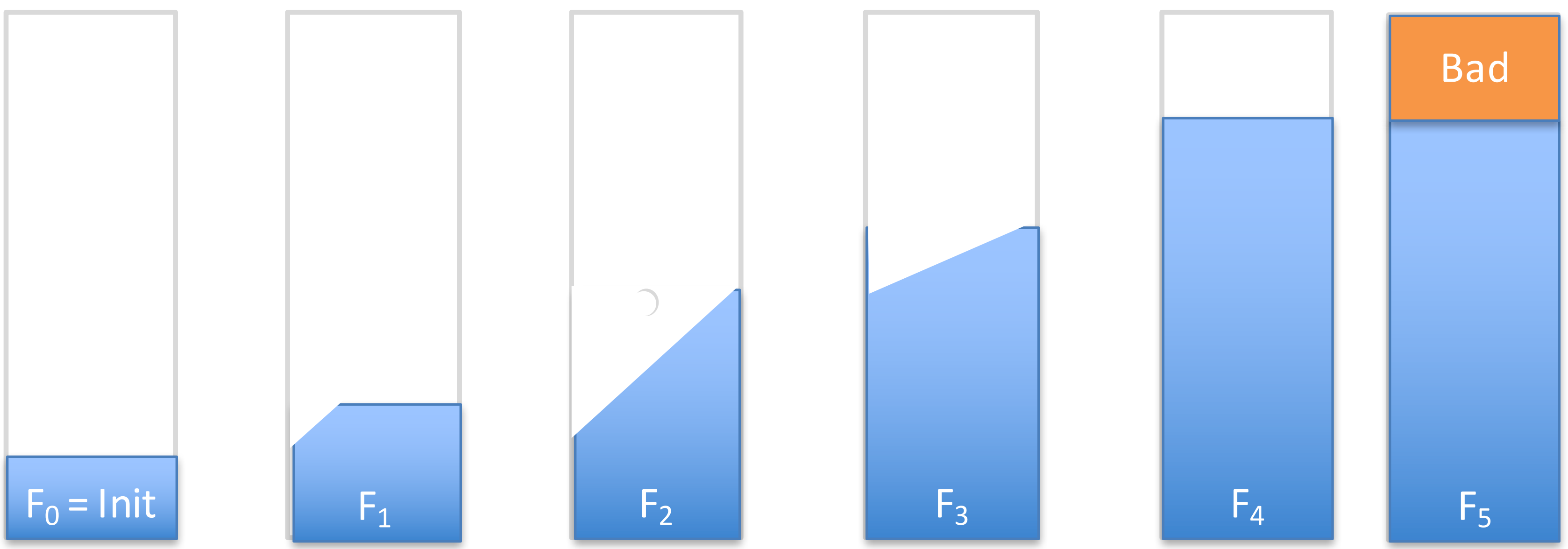
- $\varphi = \langle F_0, \ldots, F_N \rangle$
  - $F_i \cap Bad = \emptyset$
  - $F_0 = Init,\ F_i \subseteq F_{i+1}$
  - $\mathrm{TR}(F_i) \subseteq F_{i+1}$
- Q =[ ]



PDR: Unfolding

- $\varphi = \langle F_0, \ldots, F_N \rangle$
  - $F_i \cap Bad = \emptyset$
  - $F_0 = Init, \; F_i \subseteq F_{i+1}$
  - $\mathrm{TR}(F_i) \subseteq F_{i+1}$ **(i + 1 < N)**
- Q =[ ]

N = 5



PDR Operations

- ✓ Unfolding
- ✓ Queue initialization
- ✓ Backward step
- ✓ Blocking
- ✓ Generalization
- ✓ Termination: Counterexample / Invariant
- Obligation lifting
- Inductive generalization
- Forward propagation
- Pushing obligation forward

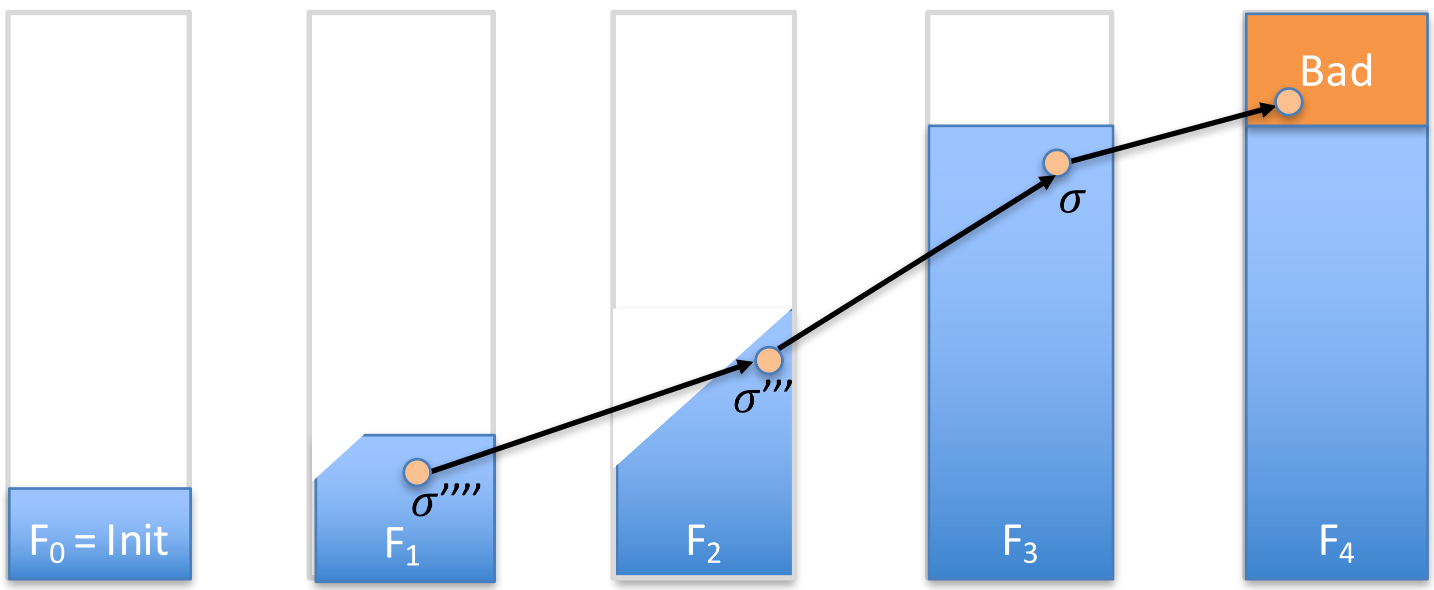
PDR & Abstract Interpretation

- AI & PDR compute inductive invariants
  - AI: Systematic
  - PDR: Specialized
- How can we formulate PDR using AI?
  - Abstractions
  - Transformers

Where to start?

- What concrete semantics does PDR abstract?

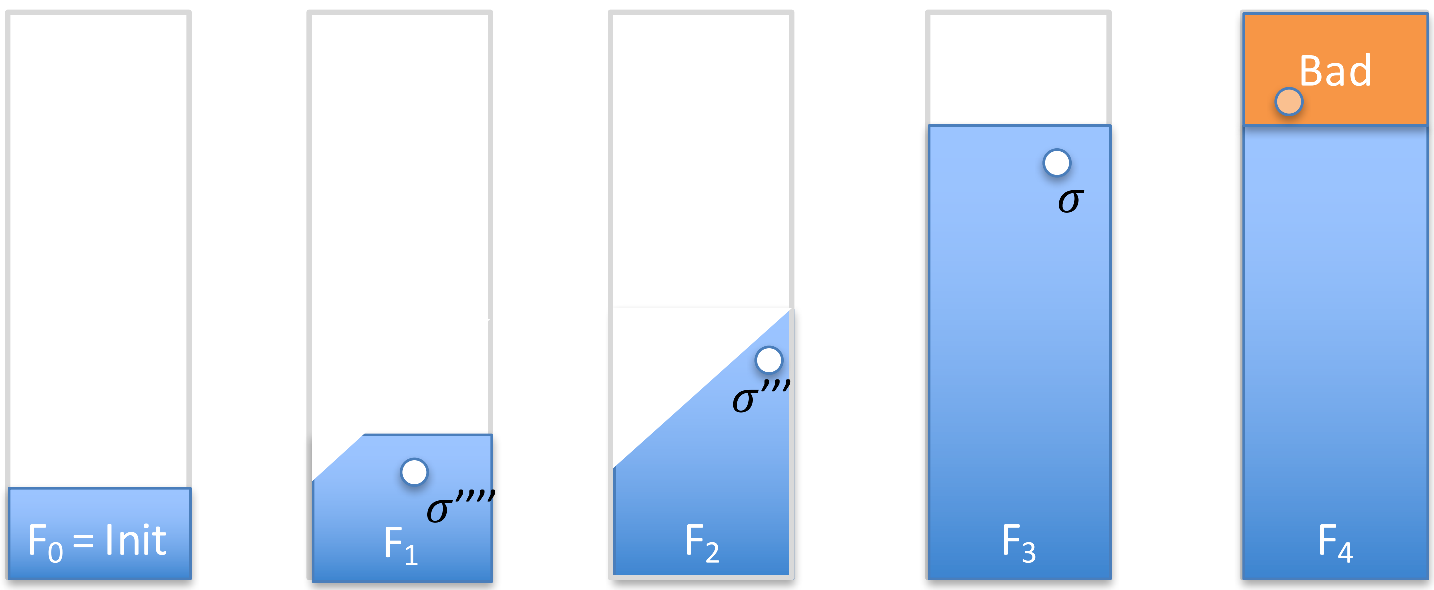
- Q =[$(\sigma'''',1)$ $(\sigma''',2)$ $(\sigma,3)$]



Where to start?

- What concrete semantics does PDR abstracts?

- Q =[ ]



Collecting (Backward) Trace Semantics

- Verification problem: ($Init, P, Bad$)
- $[\![\mathrm{P}]\!]^{B}_{\wp(\Sigma^*)}$ = evil traces $\subseteq \Sigma^*$

$$= \left\{ \pi \in \Sigma^* \;\middle|\; \pi_0 \in Bad \wedge \forall i.\, \pi_i \xrightarrow{\mathrm{TR}^{-1}} \pi_{i+1} \right\}$$

Collecting (Backward) Trace Semantics

- Verification problem: ($Init, P, Bad$)
- $[\![\mathrm{P}]\!]^{B}_{\wp(\Sigma^*)} =$ evil traces $\subseteq \Sigma^*$

$$= \left\{ \pi \in \Sigma^* \;\middle|\; \pi_0 \in Bad \wedge \forall i.\, \pi_i \xrightarrow{\mathrm{TR}^{-1}} \pi_{i+1} \right\}$$

$$\pi = \sigma_0\sigma_1\sigma_2\sigma_3\sigma_4$$

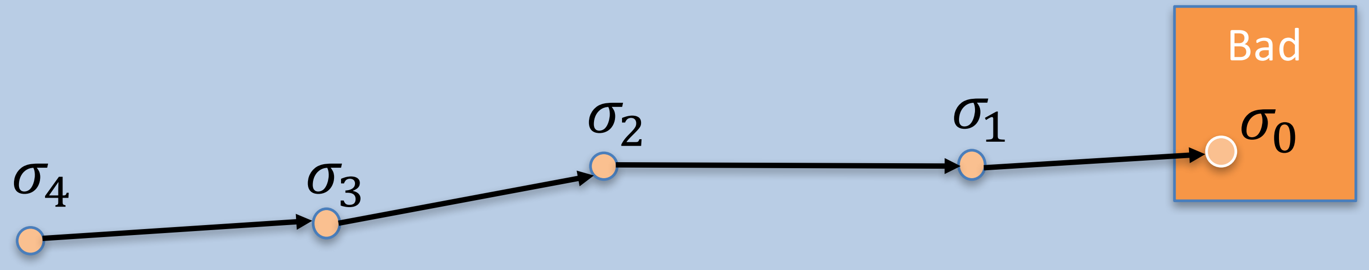
Collecting (Backward) Trace Semantics

- Verification problem: ($Init$, P, Bad)

- $[\![\mathrm{P}]\!]^{B}_{\wp(\Sigma^*)}$ = evil traces $\subseteq \Sigma^*$

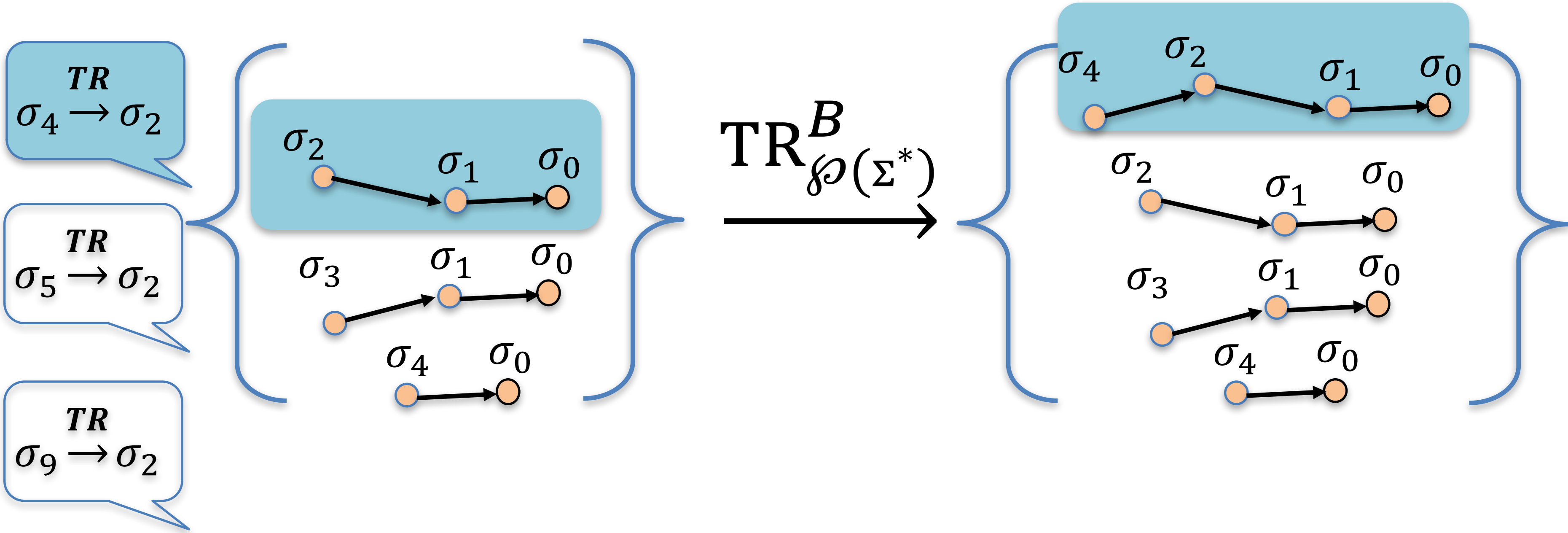
$$= \left\{ \pi \in \Sigma^* \;\middle|\; \pi_0 \in Bad \wedge \forall i.\, \pi_i \xrightarrow{\mathrm{TR}^{-1}} \pi_{i+1} \right\}$$

$$\pi = \sigma_0\sigma_1\sigma_2\sigma_3\sigma_4$$

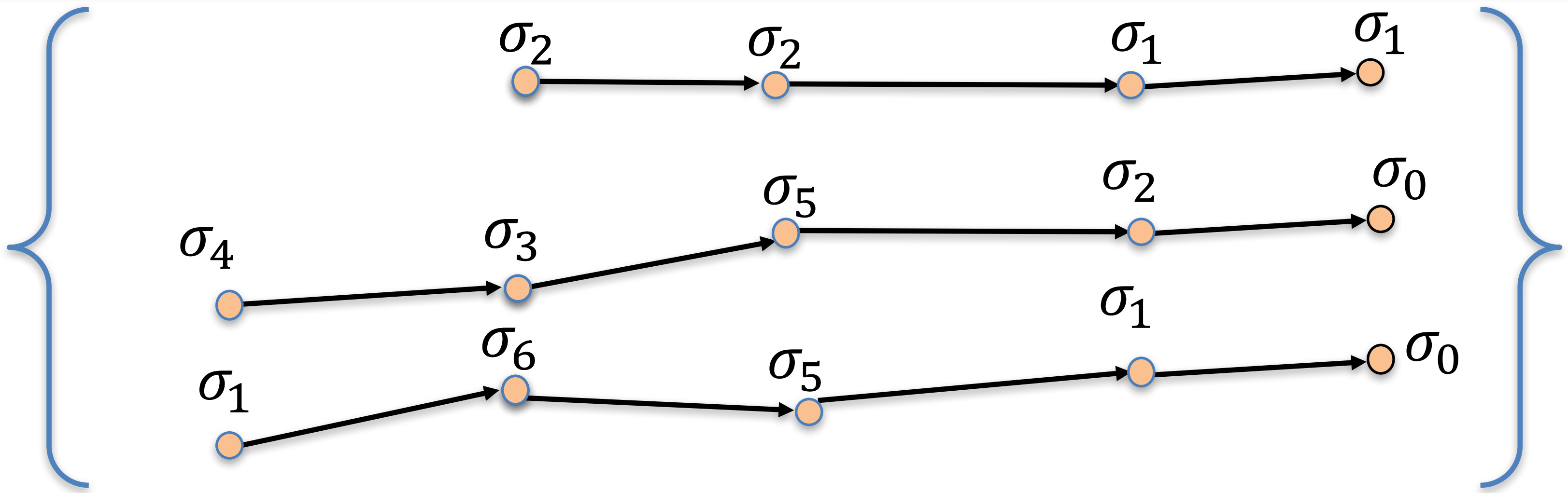


Collecting (Backward) Transitions

- Verification problem: ($Init$, P, Bad)
- $\mathrm{TR}^{B}_{\wp(\Sigma^*)} \subseteq \wp(\Sigma^*) \times \wp(\Sigma^*)$

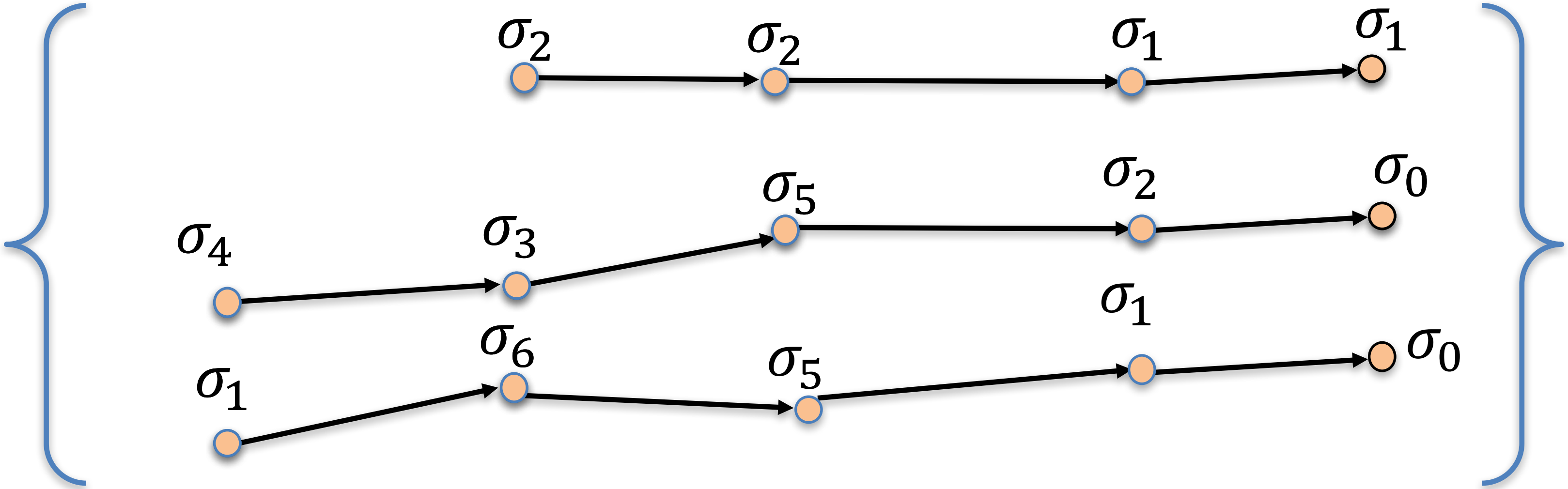


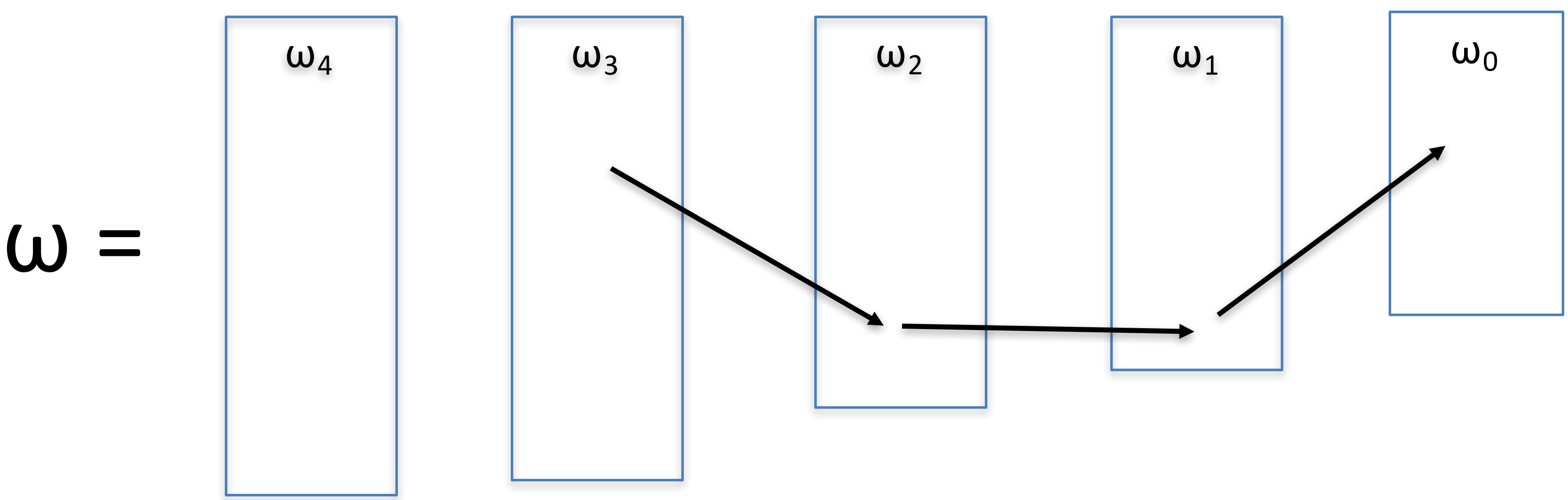
How to abstract?



What would PDR do?

1. Cartesian Abstraction

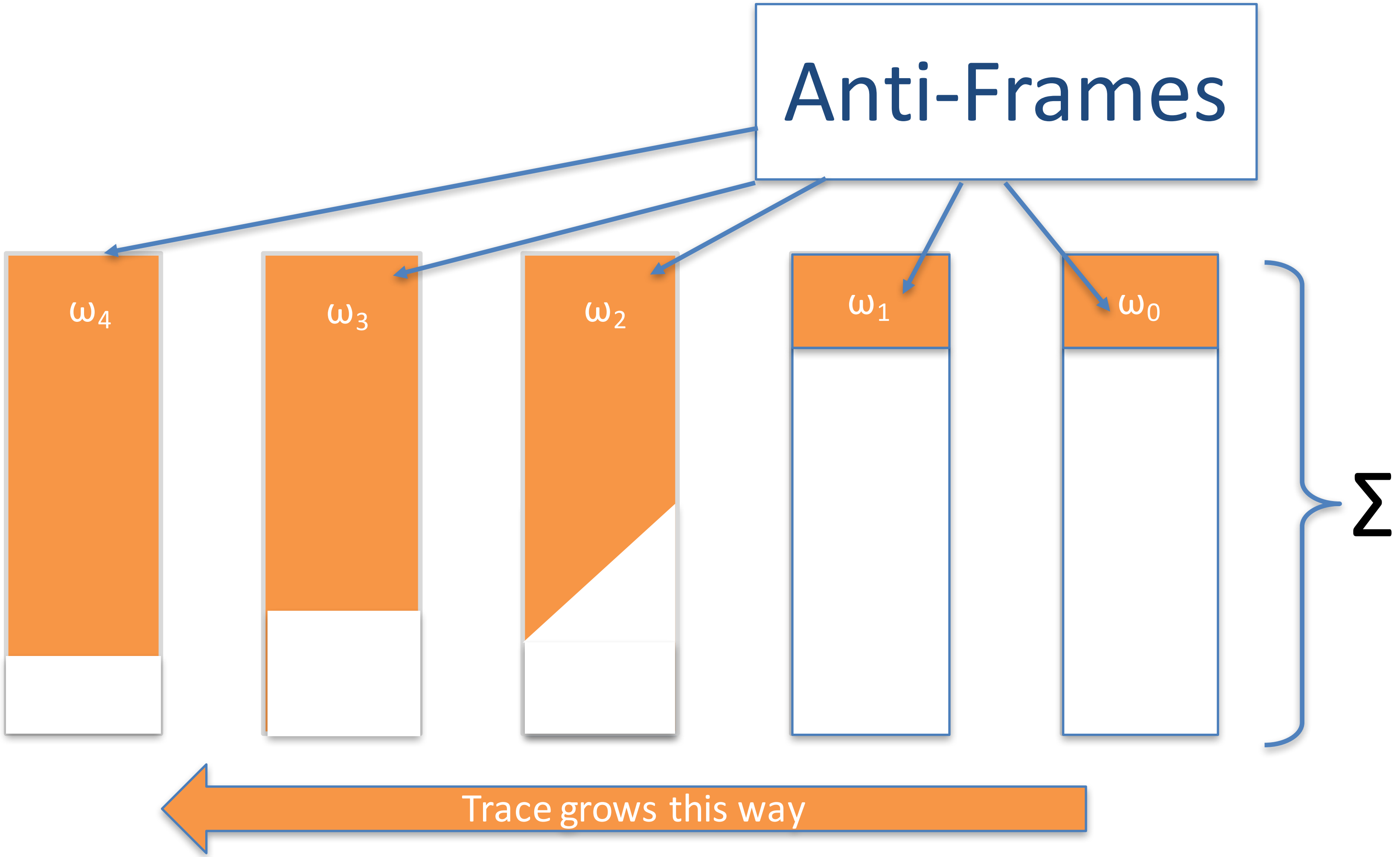




$$\omega = \omega_0\omega_1\omega_2\omega_3\omega_4 \in \Omega = \mathcal{P}(\Sigma)^*$$

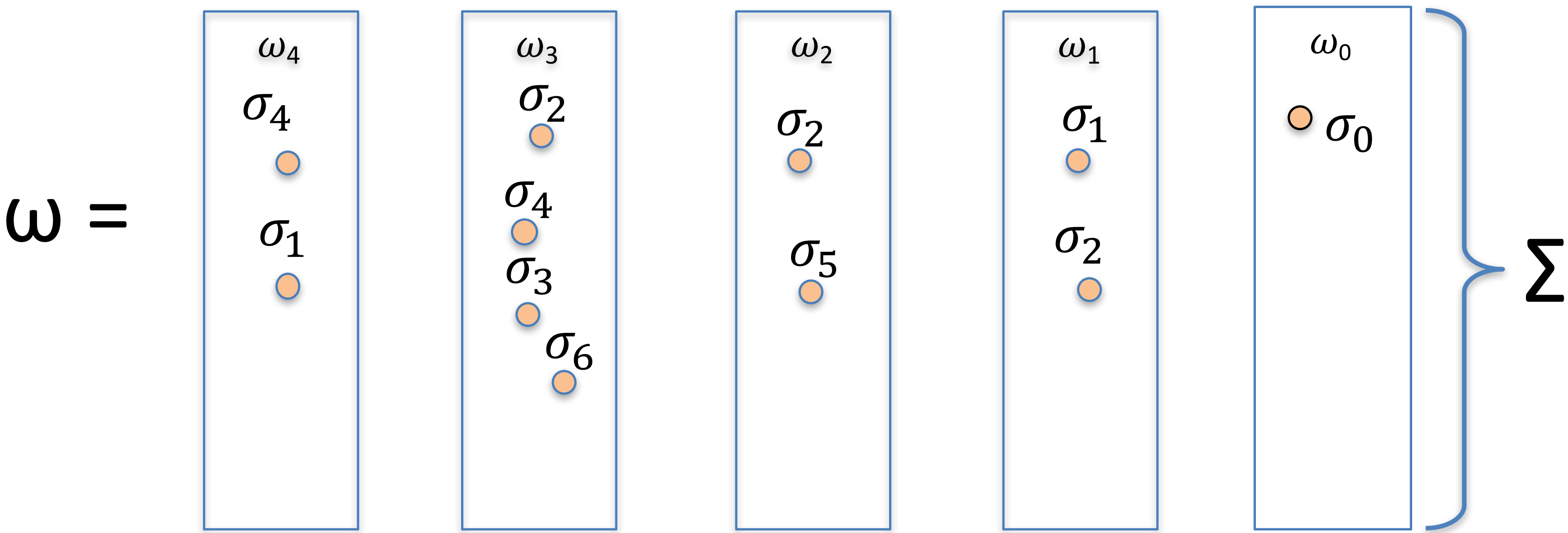
Cartesian Traces

- $\omega = \omega_0\omega_1\omega_2\omega_3\omega_4 \in \Omega = \mathcal{P}(\Sigma)^*$



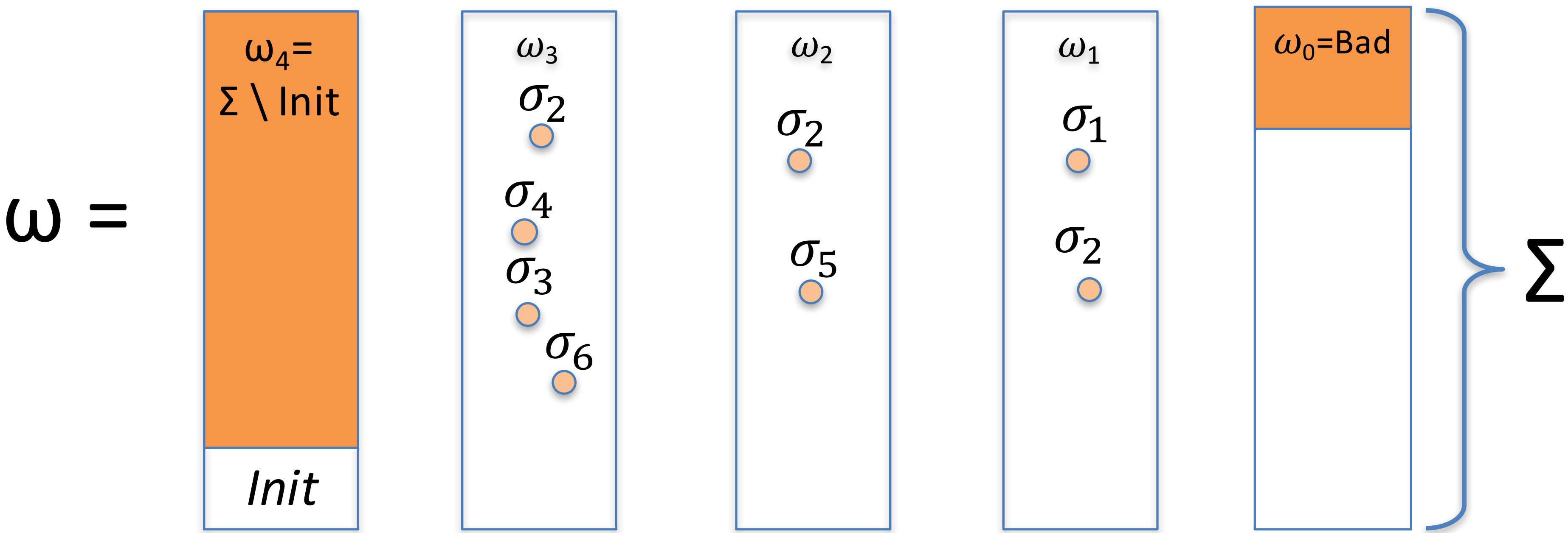
2. Init-Sensitive Abstraction

- Cartesian abstraction allows to capture only states occurring in evil traces



2. Init-Sensitive Abstraction

- $\omega = Bad \_\_\_\_ \boldsymbol{\Sigma} \setminus \boldsymbol{Init}$



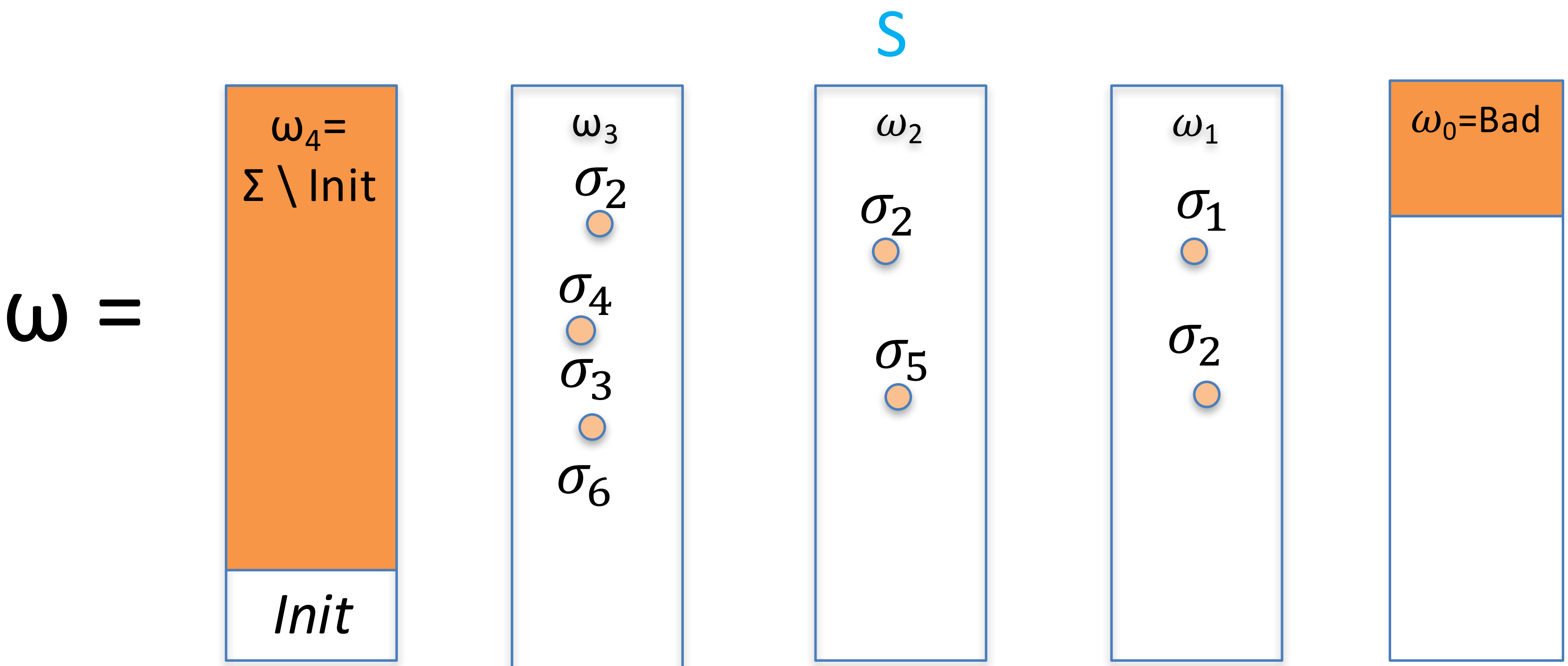
3. Bound-Sensitive Abstraction

- PDR refines the abstraction as the bound grows
- The semantics maintain a set of cartesian traces
- $[\![\boldsymbol{P}]\!]^{\boldsymbol{B}}_{\wp(\boldsymbol{\Omega})} \in \wp(\boldsymbol{\Omega})$

What about Transformers?

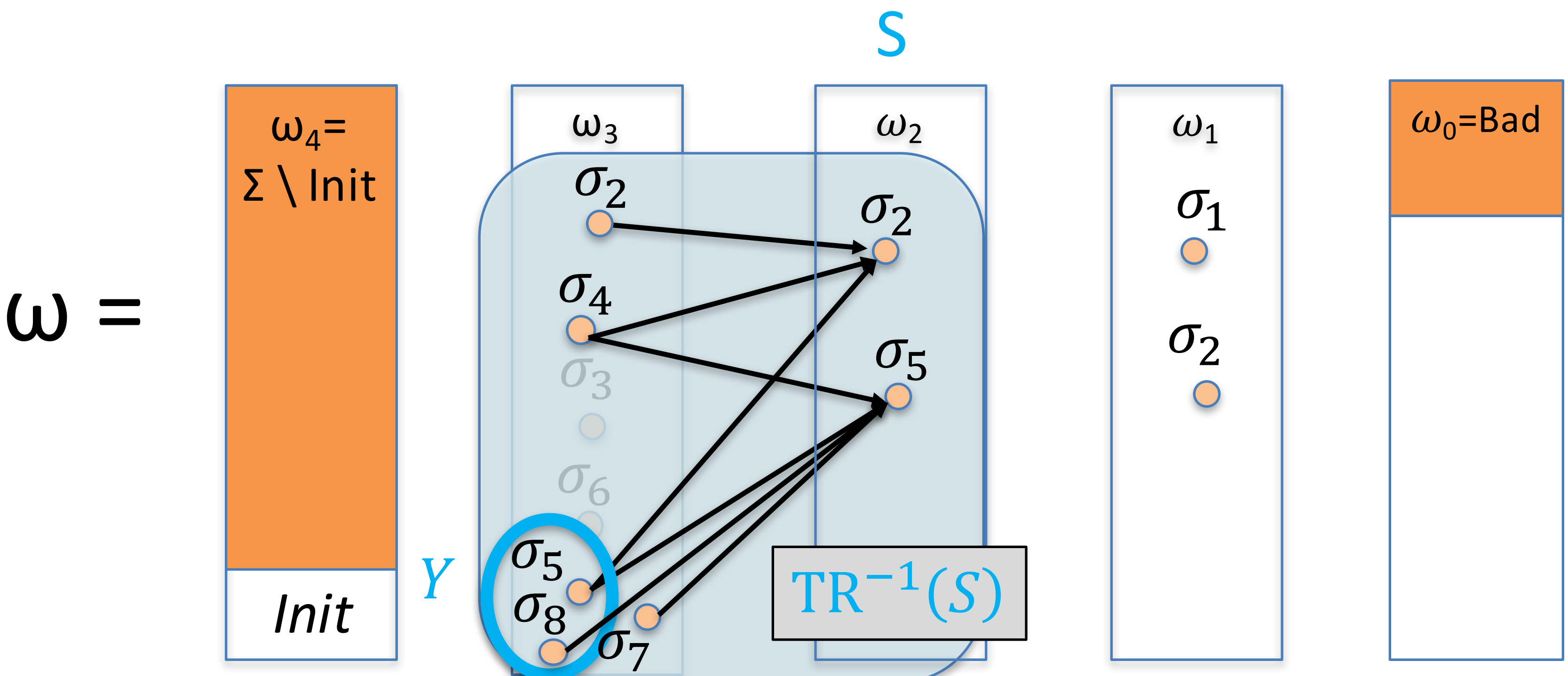
Backward Transitions

- $TR^{B}_{\Omega} \subseteq \Omega \times \Omega$
- $\omega \xrightarrow{TR^{B}_{\Omega}} \omega'$



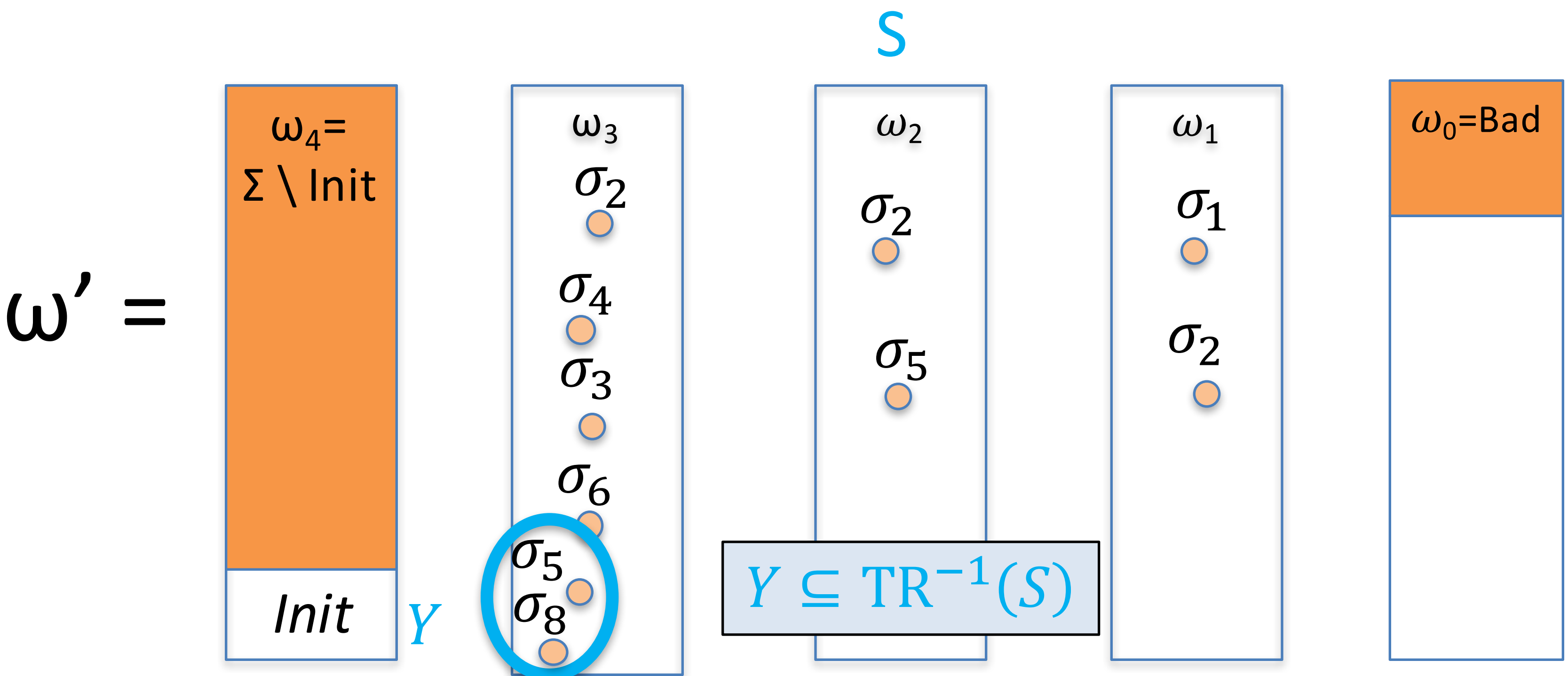
Backward Transitions

- $TR_{\Omega}^{B} \subseteq \Omega \times \Omega$
- $\omega \xrightarrow{TR_{\Omega}^{B}} \omega'$



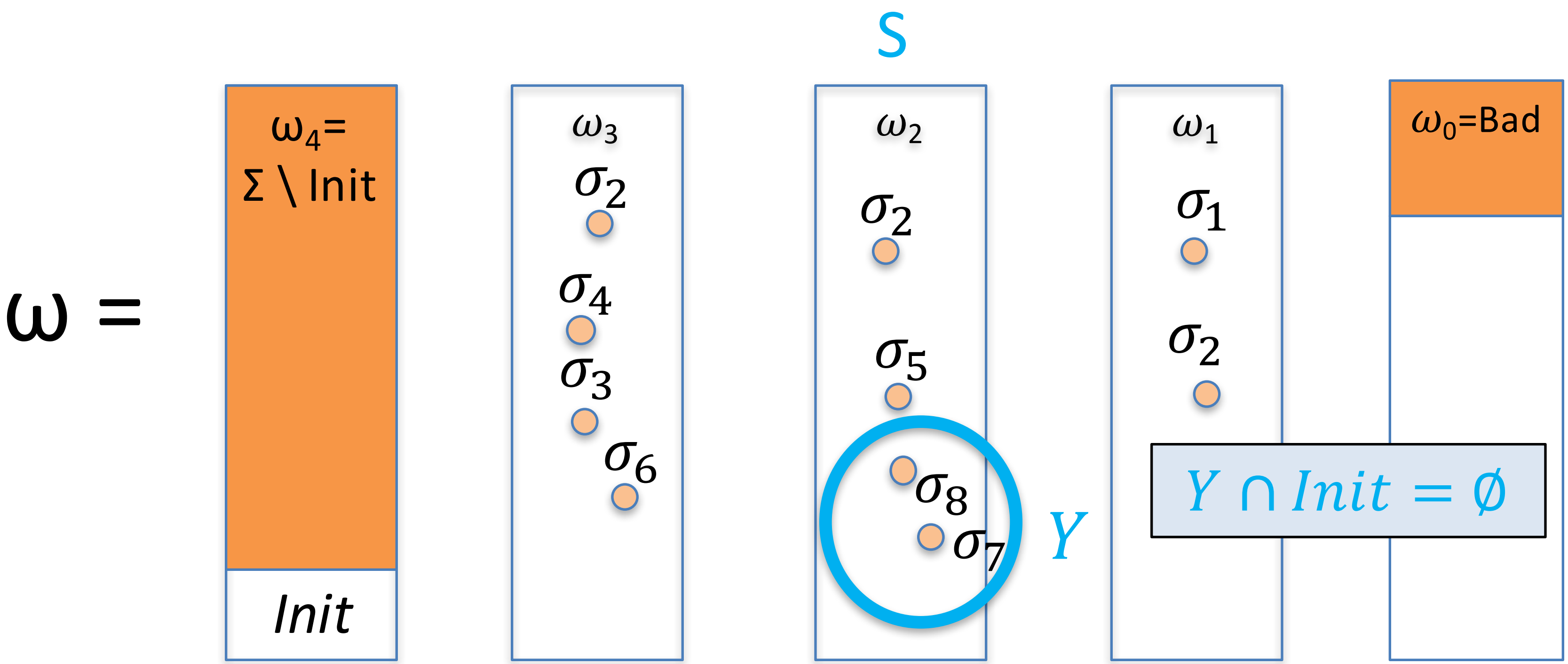
Backward Transitions

- $TR_{\Omega}^{B} \subseteq \Omega \times \Omega$
- $\omega \xrightarrow{TR_{\Omega}^{B}} \omega'$



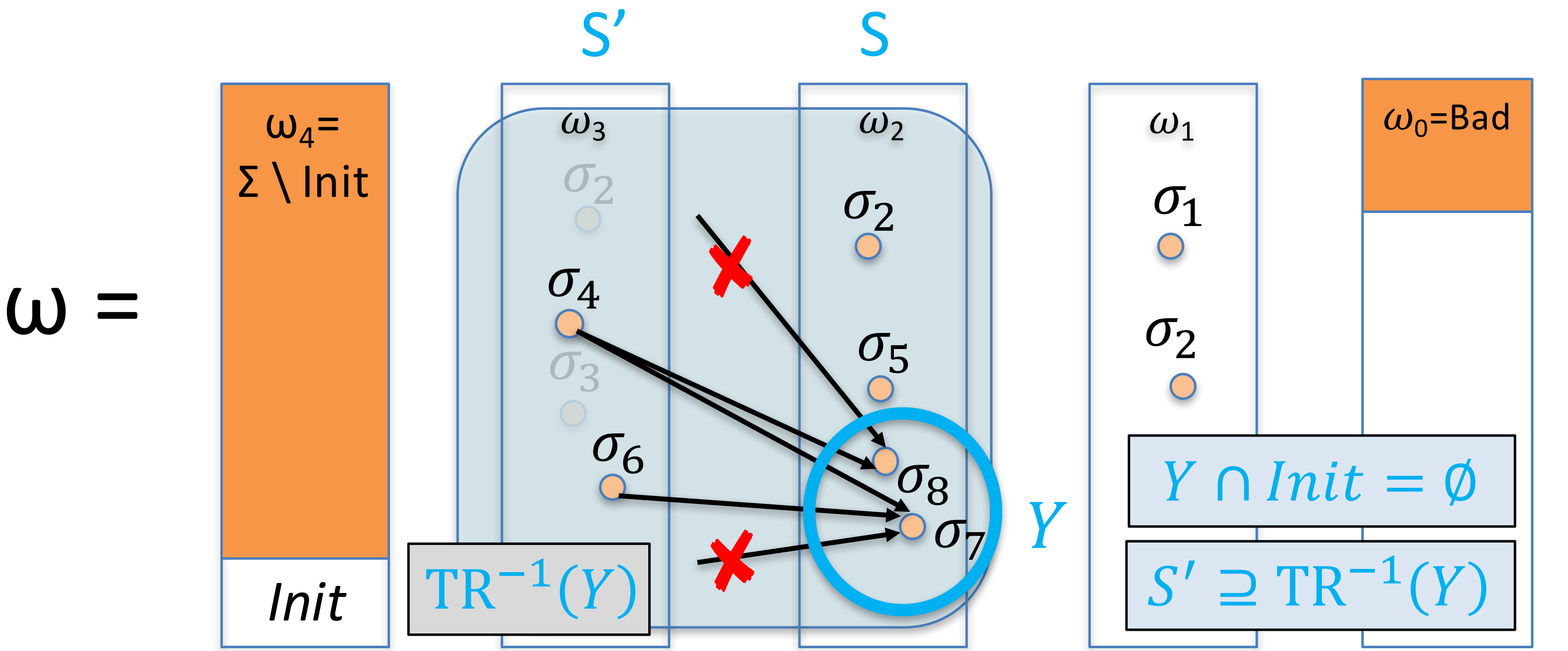
Generalization Transitions

- $TR_{\Omega}^{Gen} \subseteq \Omega \times \Omega$
- $\omega \xrightarrow{TR_{\Omega}^{Gen}} \omega'$



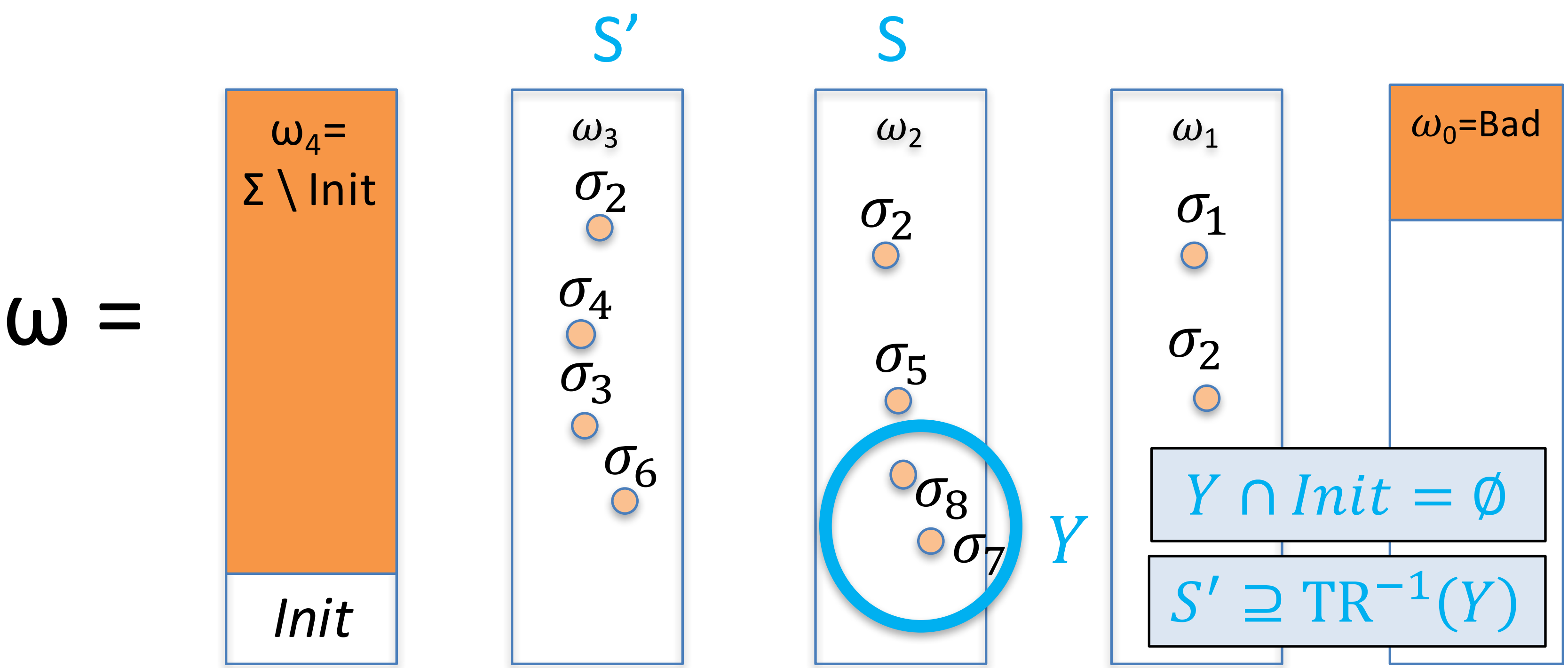
Generalization Transitions

- $TR_{\Omega}^{Gen} \subseteq \Omega \times \Omega$
- $\omega \xrightarrow{TR_{\Omega}^{Gen}} \omega'$



Generalization Transitions

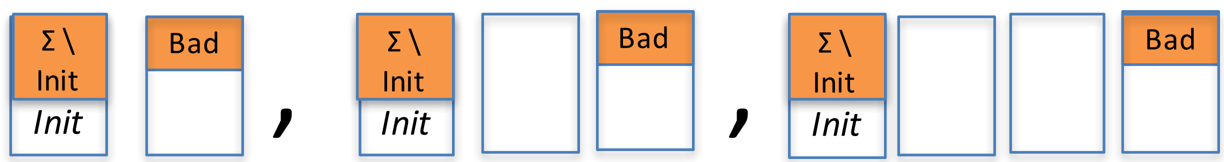
- $TR_{\Omega}^{Gen} \subseteq \Omega \times \Omega$
- $\omega \xrightarrow{TR_{\Omega}^{Gen}} \omega'$



Small step collecting property-guided cartesian trace semantics

- $[\![\boldsymbol{P}]\!]^{\boldsymbol{B}}_{\wp(\boldsymbol{\Omega})} \in \wp(\boldsymbol{\Omega})$

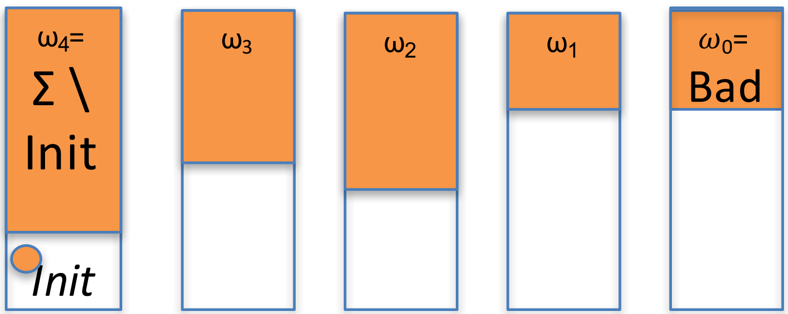
- $\widehat{\Omega} = \{\omega = Bad\ \emptyset \dots \emptyset\ (\Sigma \setminus Init) \mid 2 \leq |\omega|\ \}$

$= \{$ 

$, \dots \}$

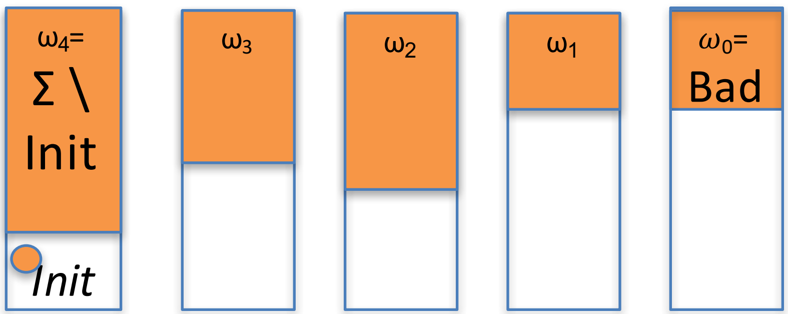
- $[\![\boldsymbol{P}]\!]^{\boldsymbol{B}}_{\wp(\boldsymbol{\Omega})} = \{\ \omega' \mid \omega \in \widehat{\boldsymbol{\Omega}} \wedge \omega \xrightarrow{TR^{B}_{\Omega} \cup TR^{Gen(B)}_{\Omega}}{}^{*}\ \omega' \}$

Safety

- $\boldsymbol{P}$ is safe iff  $\notin [\![\boldsymbol{P}]\!]^{\boldsymbol{B}}_{\wp(\boldsymbol{\Omega})}$

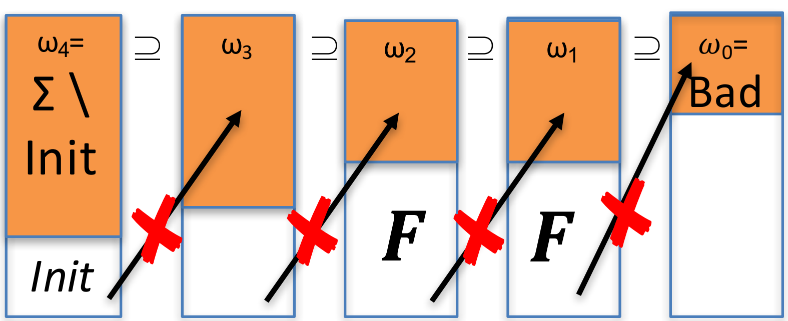
Safety

- $\boldsymbol{P}$ is safe iff



$\notin [\![\boldsymbol{P}]\!]^{\boldsymbol{B}}_{\wp(\boldsymbol{\Omega})}$

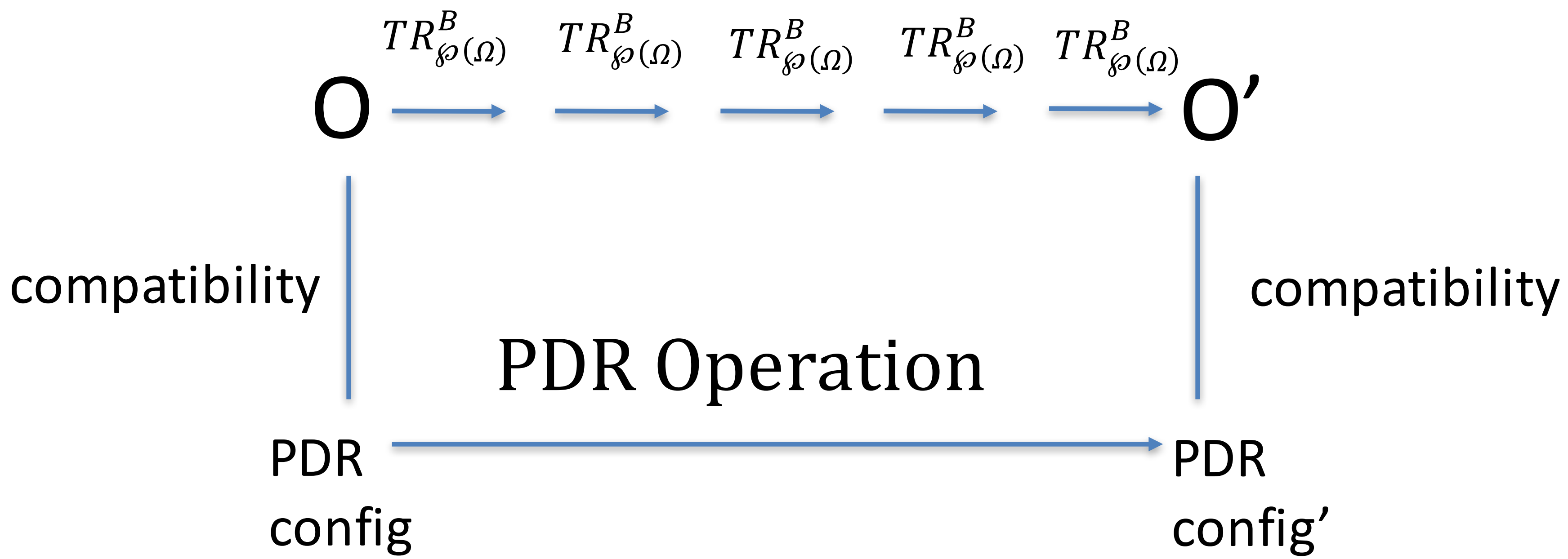
- $\boldsymbol{F}$ is inductive invariant iff



$\in [\![\boldsymbol{P}]\!]^{\boldsymbol{B}}_{\wp(\boldsymbol{\Omega})}$

PDR as Abstract Interpretation

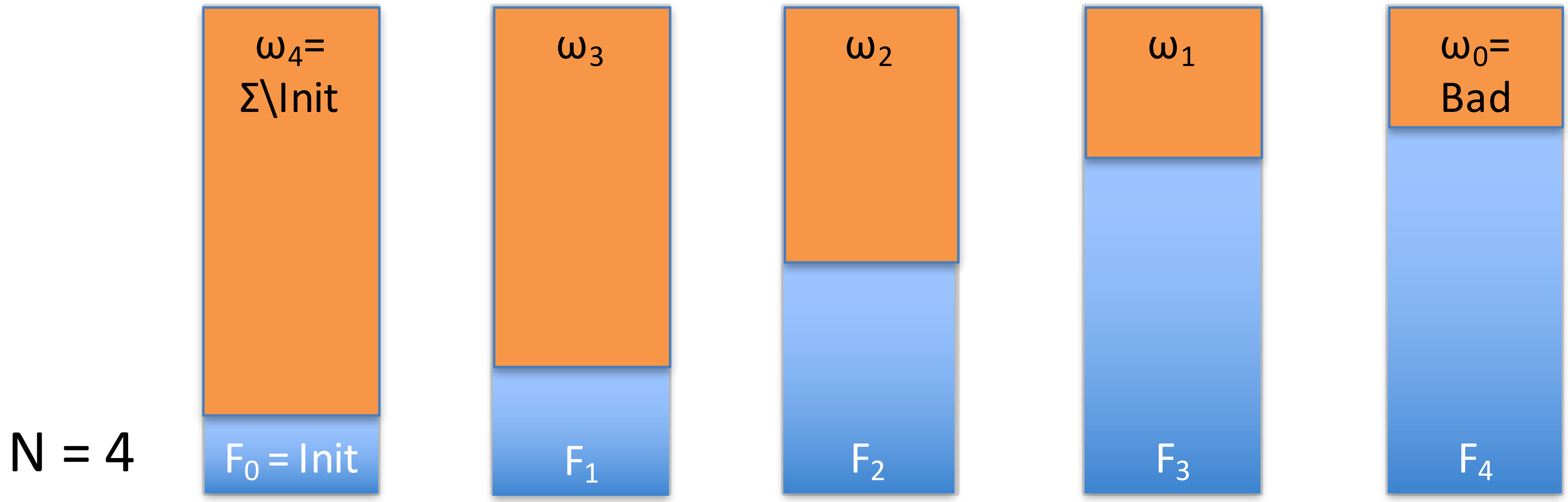
- PDR as interpretation using $\in [\![\boldsymbol{P}]\!]^{\boldsymbol{B}}_{\wp(\boldsymbol{\Omega})}$



Stuttering Simulation

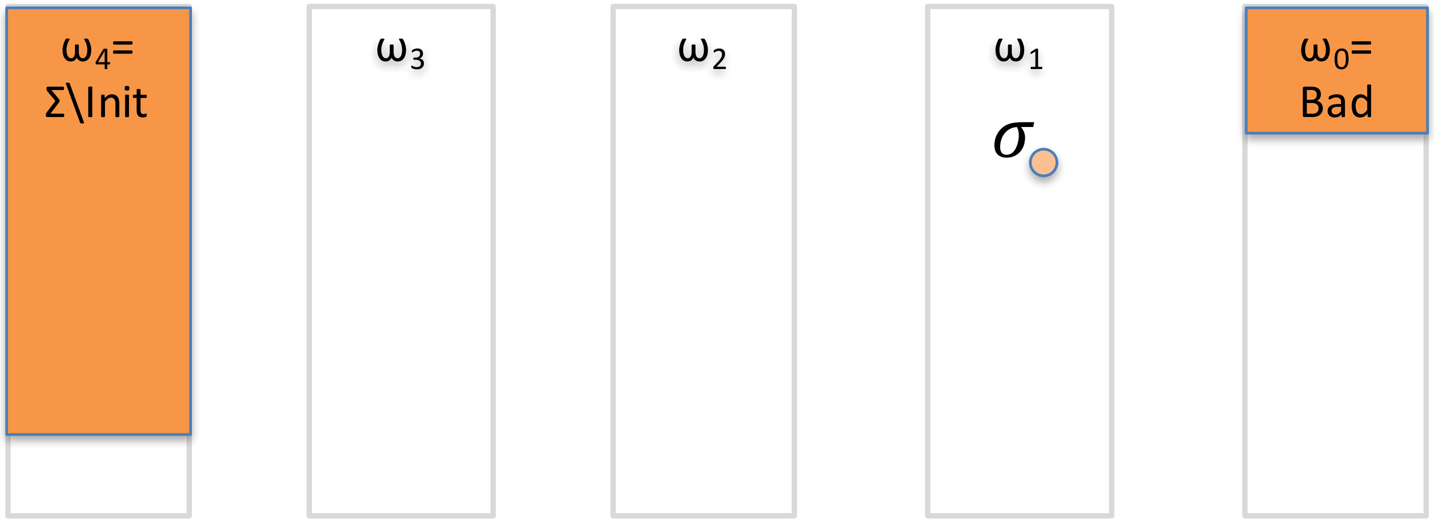
Proof-Compatibility

- An intermediate forward sequence $\langle F_0, .., F_N \rangle$ is **proof-compatible** with $\omega \in \Omega$ if
  - $|\omega| = N + 1$
  - $\forall i. F_i = \Sigma \setminus \omega(N - i)$



CEX-Compatibility

- An obligation (σ, i) is **cex-compatible** with $\omega \in \Omega$ if
  - $|\omega| \geq i + 1$
  - $\sigma \in \omega(|\omega| - i - 1)$
- Q =[(σ,3)]



Compatibility

- A PDR configuration $\langle N, \langle F_0, .., F_N \rangle, q \rangle$ is **compatible** with $O \subseteq \Omega$ if
  - $\exists \omega \in O$ s.t. $\langle F_0, .., F_N \rangle$ is proof compatible with ω
  - $\forall (\sigma, i) \in q$, ∃ a cex-compatible $\omega \in O$

PDR as Abstract Interpretation

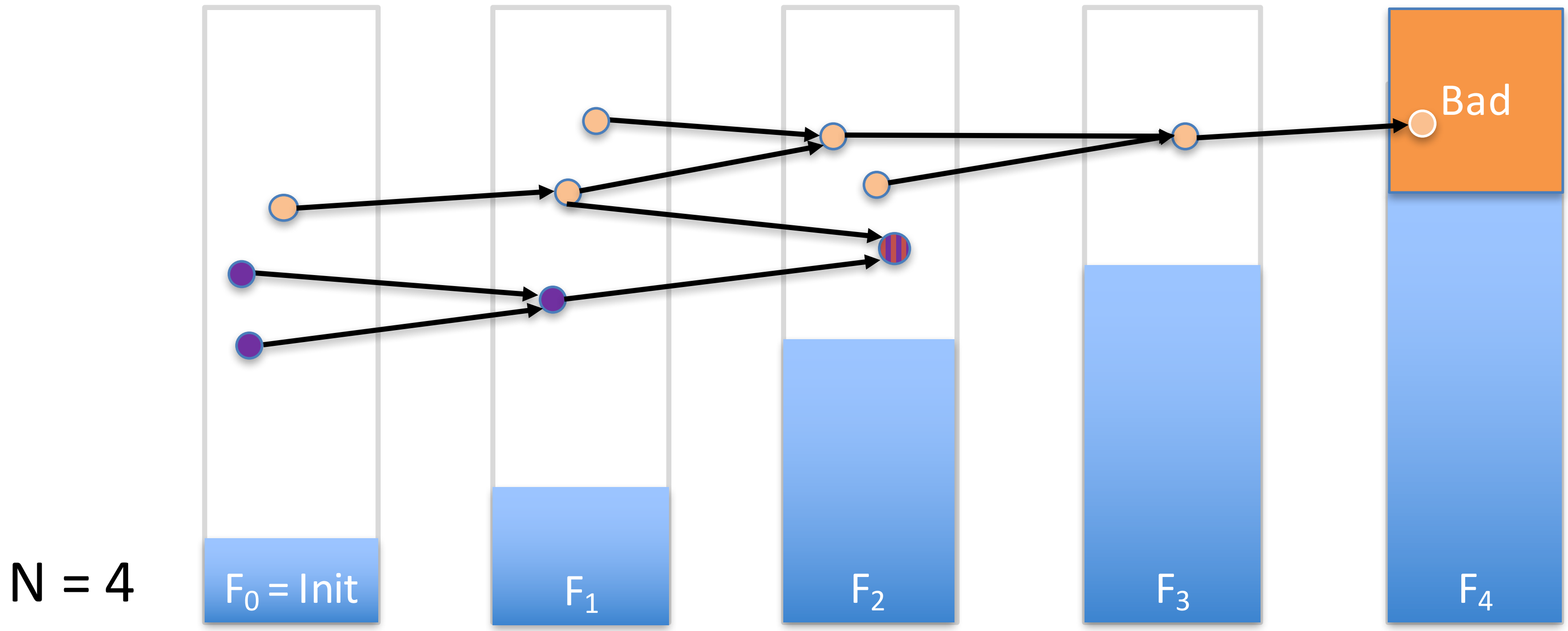
PDR Algorithms interpret the program using $[\![\boldsymbol{P}]\!]^{\boldsymbol{B}}_{\wp(\boldsymbol{\Omega})}$

- Stop at
  - Counterexample
  - Inductive invariant

Conclusions

- PDR algorithms can be explained as abstract interpretation using a non standard semantics
  - 2 key operations
- New proof technique for soundness of future PDR algorithms

F_1 is Where the Action Starts



Thank you!