Synthesizing Universally-Quantified Inductive Invariants

Sharon Shoham

The Blavatnik School of Computer Science
The Raymond and Beverly Sackler Faculty of Exact Sciences
Tel Aviv University

Tel Aviv University

Synthesizing Universally-Quantified Inductive Invariants

Aleksandr Karbyshev

SOLIDUM PETIT IN PROFUNDIS · UNIVERSITAS ARHUSIENSIS

Nikolaj Bjorner

Microsoft Research

Shachar Itzhaky

Technion Israel Institute of Technology

Noam Rinetzky

TEL AVIV UNIVERSITY אוניברסיטת תל אביב

Oded Padon

TEL AVIV UNIVERSITY אוניברסיטת תל אביב

Kenneth McMillan

Microsoft Research

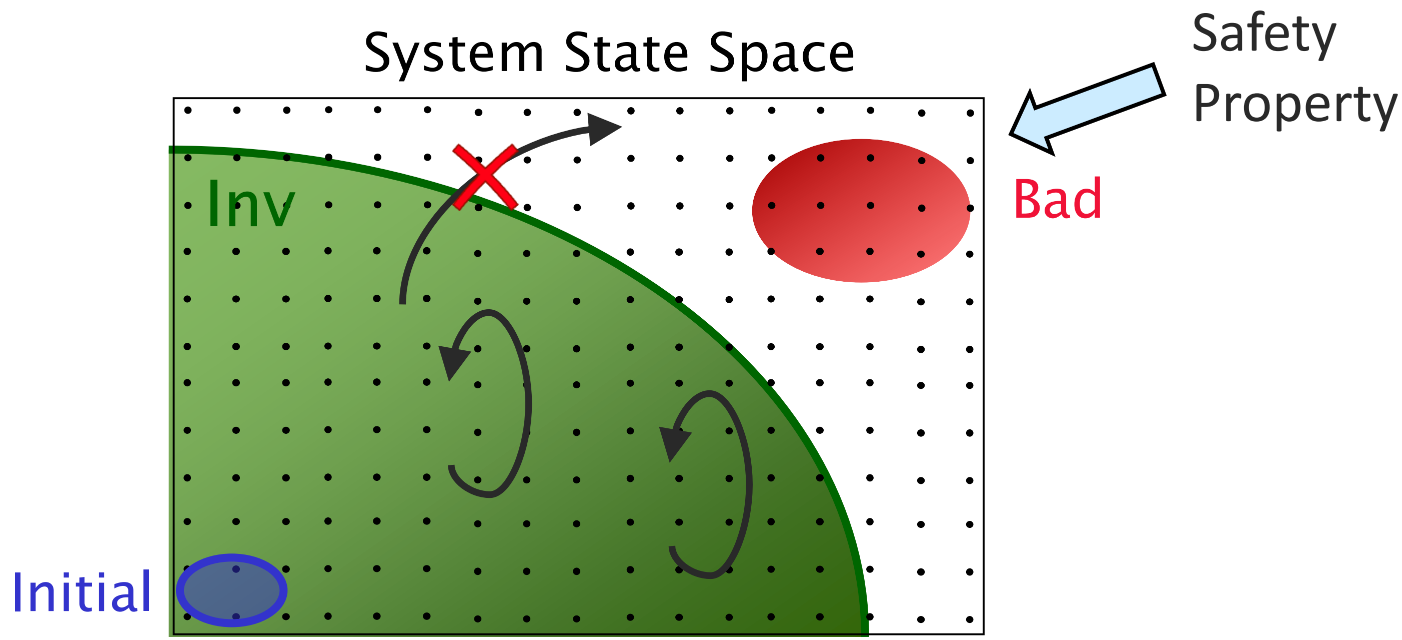
Aurojit Panda

Berkeley UNIVERSITY OF CALIFORNIA

Mooly Sagiv

TEL AVIV UNIVERSITY אוניברסיטת תל אביב

Safety Verification

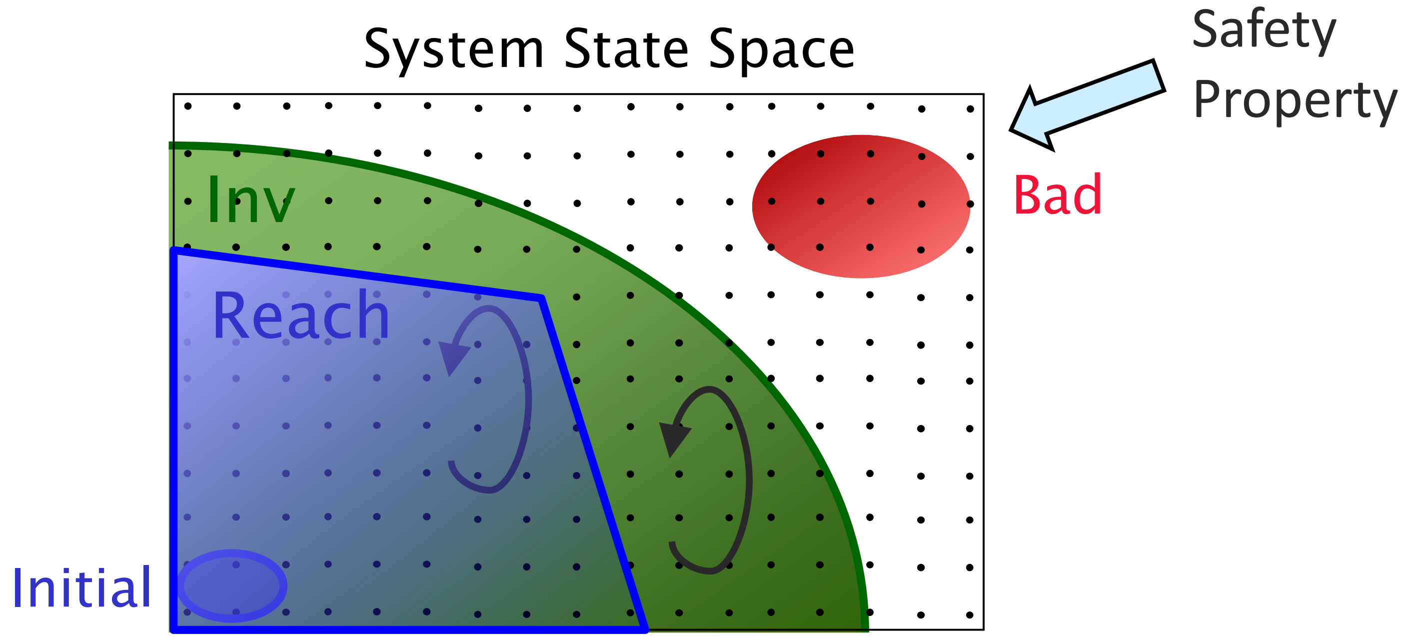


System S is **safe** if all the reachable states satisfy the property P = ¬Bad

System S is safe iff there exists an **inductive invariant** Inv:

Inv $\Rightarrow$ P=¬Bad	(Safety)
Init $\Rightarrow$ Inv	(Initiation)
if $\sigma \models$ Inv and T(σ, σ') then $\sigma' \models$ Inv	(Consecution)

Safety Verification



System S is **safe** if all the reachable states satisfy the property P = ¬Bad

System S is safe iff there exists an **inductive invariant** Inv:

Inv ⇒ P=¬Bad	(Safety)
Init ⇒ Inv	(Initiation)
if σ ⊨ Inv and T(σ, σ’) then σ’ ⊨ Inv	(Consecution)

Challenges

Infer inductive invariants for safety verification

But also

- Specification: reasoning about infinite-state systems
  - Unbounded number of objects, threads, messages,…
  - Quantification is useful

- Deduction: reasoning about inductive invariants
  - Undecidability of implication checking

This talk

Specify systems and properties in decidable fragment of first-order logic

- Allows quantifiers to reason about unbounded sets
- Decidable to check inductiveness

Synthesize quantified inductive invariants

- Automatically by universal property directed reachability
- Interactively by providing graphical UI for gradually strengthening the inductive invariant

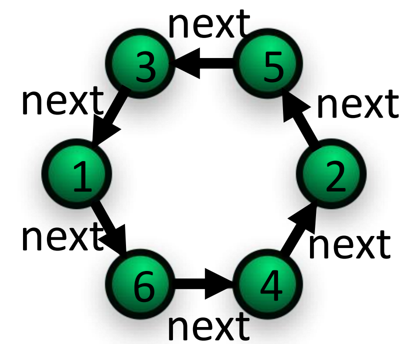
Effectively Propositional Logic – EPR
a.k.a. Bernays-Schönfinkel-Ramsey class

- Limited fragment of first-order logic
  - Restricted quantifier prefix: $\exists^* \forall^* \phi_{Q.F.}$
    - No $\forall^* \exists^*$
    - No recursive function symbols
    - No arithmetic

- Finite model property
  - A formula is satisfiable iff it is holds on models proportional to the number of existential variables

- Satisfiability is decidable
- Support from Z3, Iprover, Vampire

Example: Leader Election in a Ring

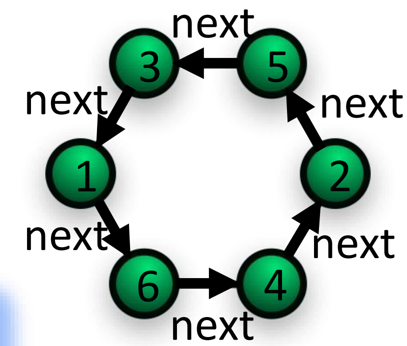


- Nodes are organized in a unidirectional ring
- Each node has a unique numeric id
- Protocol:
  - Each node sends its id to the next
  - A node that receives a message passes it (to the next) if the id in the message is higher than the node's own id
  - A node that receives its own id becomes the leader
- Theorem:
  - The protocol selects at most one leader

[CACM'79] E. Chang and R. Roberts. *An improved algorithm for decentralized extrema-finding in circular configurations of processes*

Example: Leader Election in a Ring

- Nodes are organized in a unidirectional ring
- Each node has a unique numeric id
- Protocol:
  - E
  - A … next) if the id in t…
  - A … ler
- The…
  - The protocol selects at most one leader



Proposition: This algorithm detects one and only one highest number.

Argument: By the circular nature of the configuration and the consistent direction of messages, any message must meet all other processes before it comes back to its initiator. Only one message, that with the highest number, will not encounter a higher number on its way around. Thus, the only process getting its own message back is the one with the highest number.

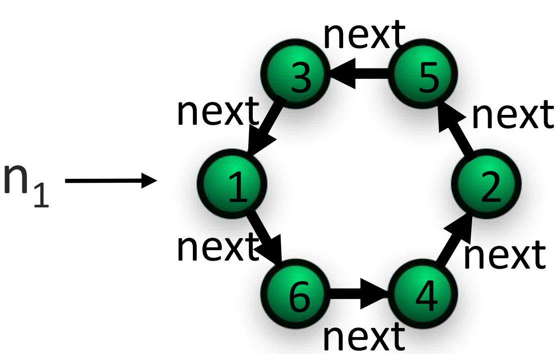
[CACM'79] E. Chang and R. Roberts. *An improved algorithm for decentralized extrema-finding in circular configurations of processes*

Modeling with EPR

- State: finite first-order structure over vocabulary V
  - $\leq$ (ID, ID) – total order on node id's
  - **btw** (Node, Node, Node) – the ring topology
  - **id**: Node → ID – relate a node to its id
  - **pending**(ID, Node) – pending messages
  - **leader**(Node) – leader(n) means n is the leader

Axiomatized in first-order logic

protocol state



structure

$$\sigma = (\{n_1, \ldots, n_6, id_1, \ldots, id_6\}, I)$$

$I(\leq) = \{\langle id_1, id_1\rangle, \langle id_1, id_2\rangle, \langle id_1, id_3\rangle, \langle id_1, id_4\rangle \ldots\}$

$I(\textbf{btw}) = \{\langle n_1, n_3, n_5\rangle, \langle n_1, n_3, n_2\rangle, \langle n_1, n_3, n_4\rangle \ldots \}$

$I(\textbf{id}) = \{n_1 \mapsto id_1, n_2 \mapsto id_6, n_3 \mapsto id_4, \ldots \}$

$I(\textbf{pending}) = \{\}$

$I(\textbf{leader}) = \{\}$

Modeling with EPR

- State: finite first-order structure over vocabulary V
- Initial states and safety property: EPR formulas over V
  - Init(V) – initial states, e.g., $\forall\, id, n.\ \neg$**pending**(id, n)
  - Bad(V) – bad states, e.g., $\exists\, n_1, n_2.\,\mathbf{leader}(n_1) \wedge \mathbf{leader}(n_2) \wedge n_1 \neq n_2$
- Transition relation:

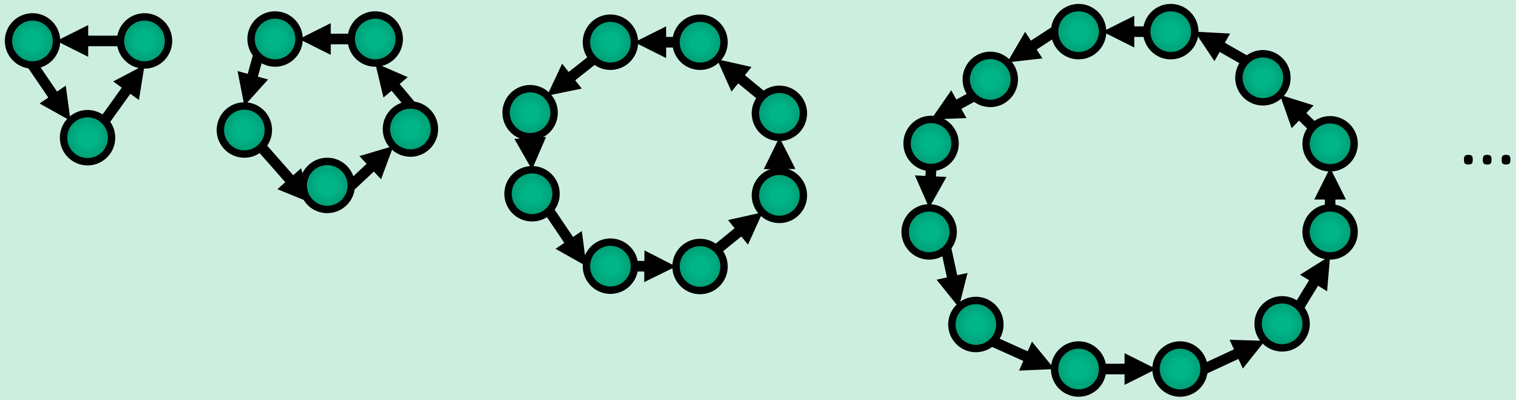
  EPR formula TR(V, V')

  V' is a copy of V describing the next state
  - e.g. $\forall\, n.\ \mathbf{leader'}(n) \leftrightarrow (\mathbf{leader}(n) \vee \mathbf{pending}\ (\mathbf{id}[n], n))$

Modeling with EPR

- State: finite first-order structure over vocabulary V
- Initial states and safety property: EPR formulas over V
  - Init(V) – initial states, e.g., $\forall\, \mathrm{id}, \mathrm{n}.\ \neg\textbf{pending}(\mathrm{id}, \mathrm{n})$
  - Bad(V) – bad states, e.g., $\exists\, n_1, n_2.\ \textbf{leader}(n_1) \land \textbf{leader}(n_2) \land n_1 \neq n_2$

Specify and verify the protocol for **any** number of nodes in the ring



Using EPR for Verification

- System Model in EPR

  Init(V), Bad(V), TR(V, V')

- Inv(V) is an inductive invariant if:
  - Initiation $\quad Init \wedge \neg Inv \quad$ unsat
  - Consecution $\quad Inv \wedge TR \wedge \neg Inv' \quad$ unsat
  - Safety $\quad Inv \wedge Bad \quad$ unsat

Decidable to check for Inv $\in \forall^*$

Useful for: linked lists, network routing, distributed protocols,...

Using EPR for Verification

- System Model in EPR

  Init(V), Bad(V), TR(V, V')

- Inv(V) is an inductive invariant if:

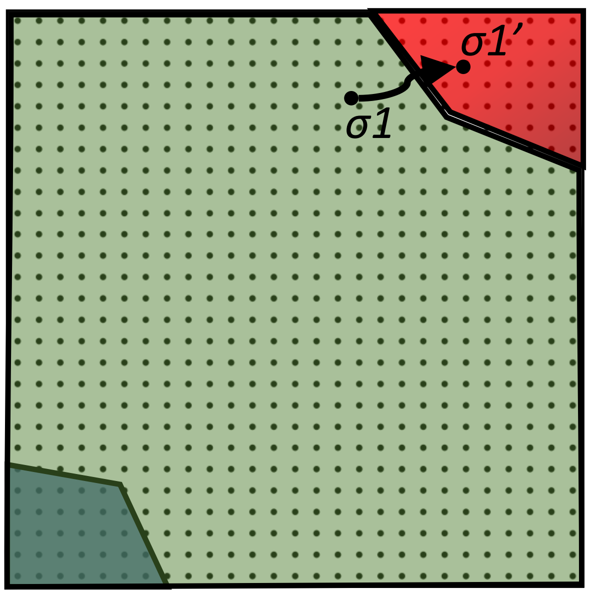
Challenge: find Inv $\in \forall^*$

Decidable to check for Inv $\in \forall^*$

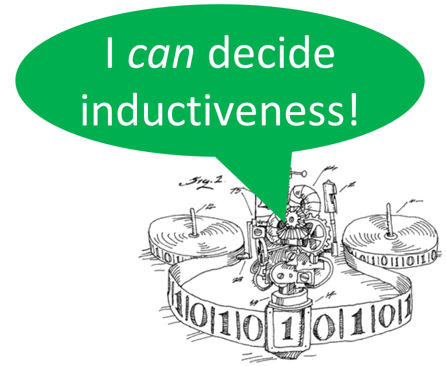
Useful for: linked lists, network routing, distributed protocols,...

Naïve algorithm

Iterative strengthening



Inv = ¬Bad

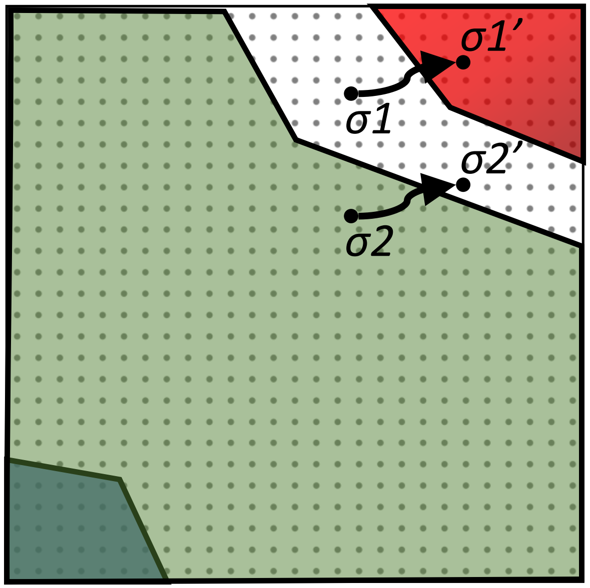


Check Inductiveness

Counterexample To Induction (CTI)

Naïve algorithm

Iterative strengthening



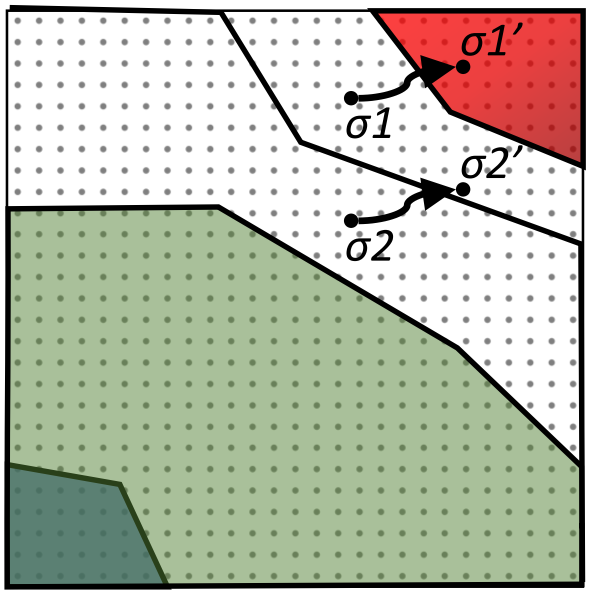
Inv = ¬Bad ∧ “*Avoid*(σ1)”

Check Inductiveness

Counterexample To Induction (CTI)

Naïve algorithm

Iterative strengthening



Inv = ¬Bad ∧ "*Avoid*(σ1)" ∧ "*Avoid*(σ2)"

Key challenge for invariant inference:

generalization

Generalization using Diagram

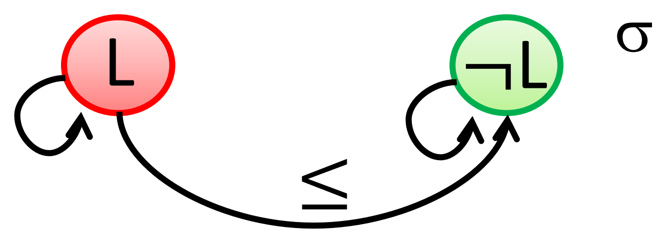
Use **diagrams** as abstract representation of states

- state σ is a finite first-order structure

$$\begin{aligned}\mathrm{Diag}(\sigma) = \exists\, x\, y.\ & x \neq y \wedge L(x) \wedge \neg L(y) \\ & \wedge \leq(x, y) \wedge \neg\leq(y, x) \\ & \wedge \leq(x, x) \wedge \leq(y, y)\end{aligned}$$

$\sigma' \models \mathrm{Diag}(\sigma)$ iff σ is a substructure of σ'

σ is obtained from σ' by removing elements and projecting relations on remaining elements



[CAV'15, JACM] Property-Directed Inference of Universal Invariants or Proving Their Absence, A. Karbyshev, N. Bjorner, S. Itzhaky, N. Rinetzky and S. Shoham.

Generalization using Diagram

Use **diagrams** as abstract re

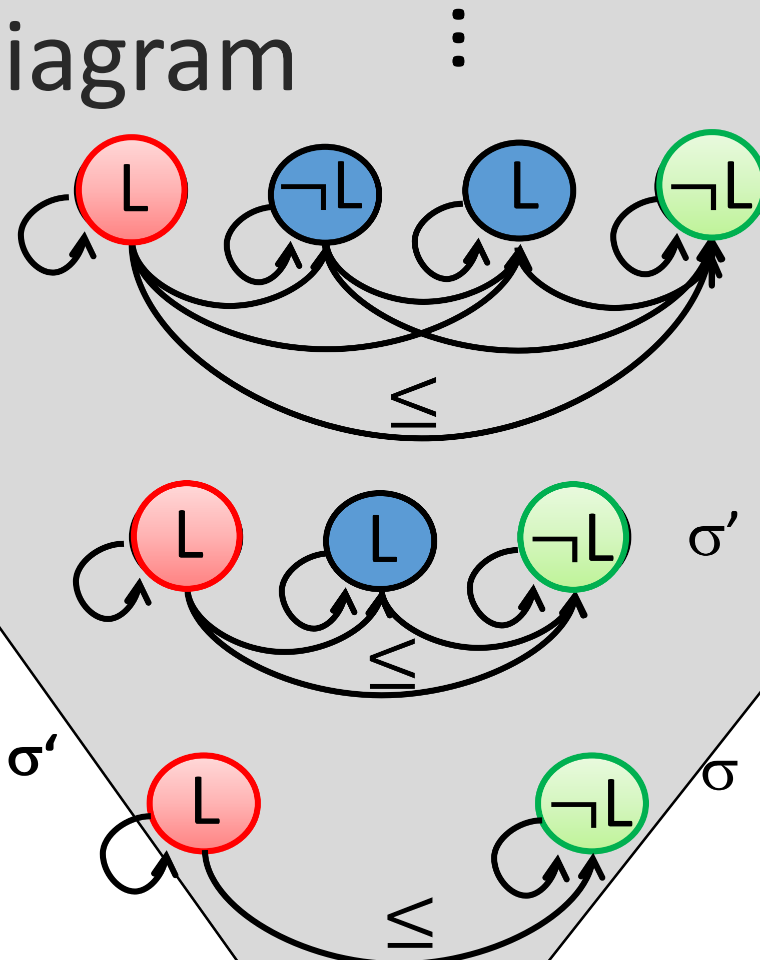
- state σ is a finite first-or

$$\mathrm{Diag}(\sigma) = \exists\, x\, y.\ x \neq y \wedge L(x) \wedge \neg L(y) \wedge \leq(x, y) \wedge \neg\leq(y, x) \wedge \leq(x, x) \wedge \leq(y, y)$$

σ' ⊨ Diag(σ) iff σ is a substructure of σ'

σ is obtained from σ' by removing elements and projecting relations on remaining elements

$$\mathrm{Avoid}(\sigma) = \neg\mathrm{Diag}(\sigma)$$



[CAV'15, JACM] Property-Directed Inference of Universal Invariants or Proving Their Absence, A. Karbyshev, N. Bjorner, S. Itzhaky, N. Rinetzky and S. Shoham.

Generalization using Diagram

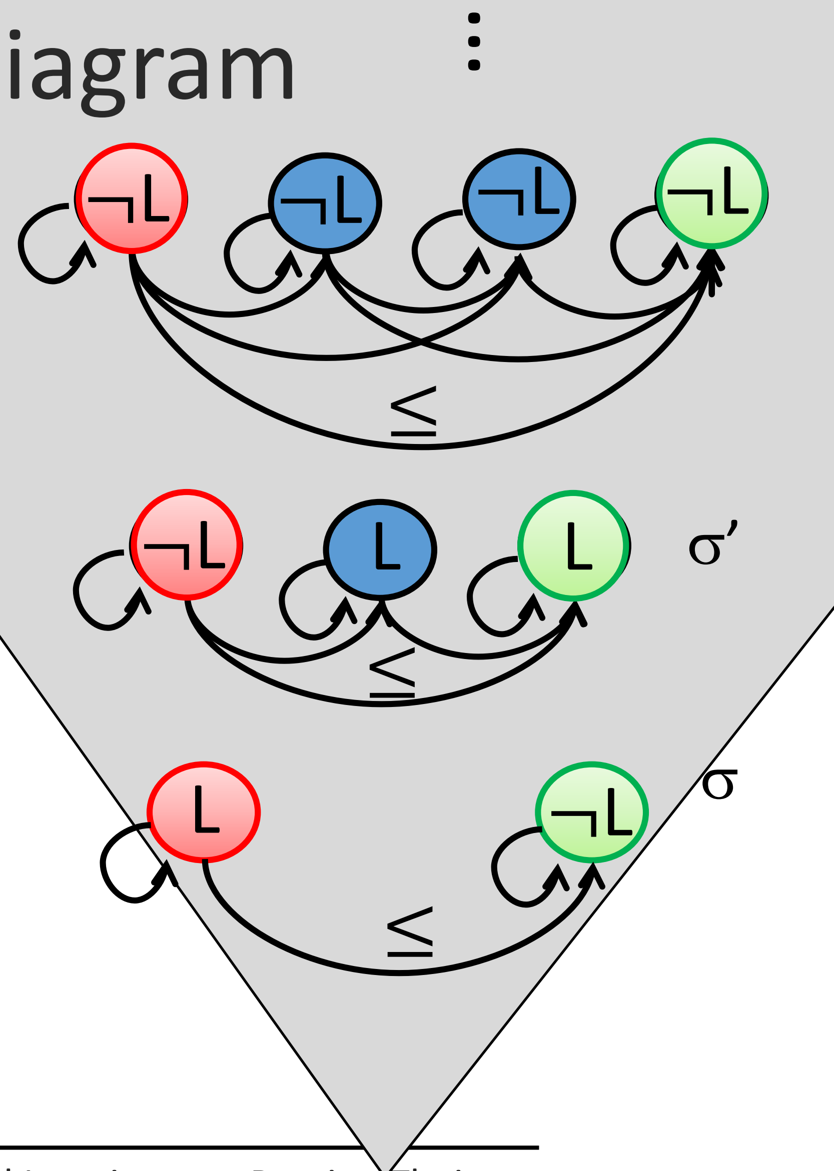
Can generalize more

- remove facts/conjuncts

$$\mathrm{Diag}(\sigma) = \exists\, x\, y.\ x \neq y \wedge L(x) \wedge \neg L(y) \wedge \leq(x, y) \wedge \neg\leq(y, x) \wedge \leq(x, x) \wedge \leq(y, y)$$

$$\mathrm{gen}(\mathrm{Diag}(\sigma)) = \exists\, x\, y.\ x \neq y \wedge \leq(x, y) \wedge \neg\leq(y, x) \wedge \leq(x, x) \wedge \leq(y, y)$$

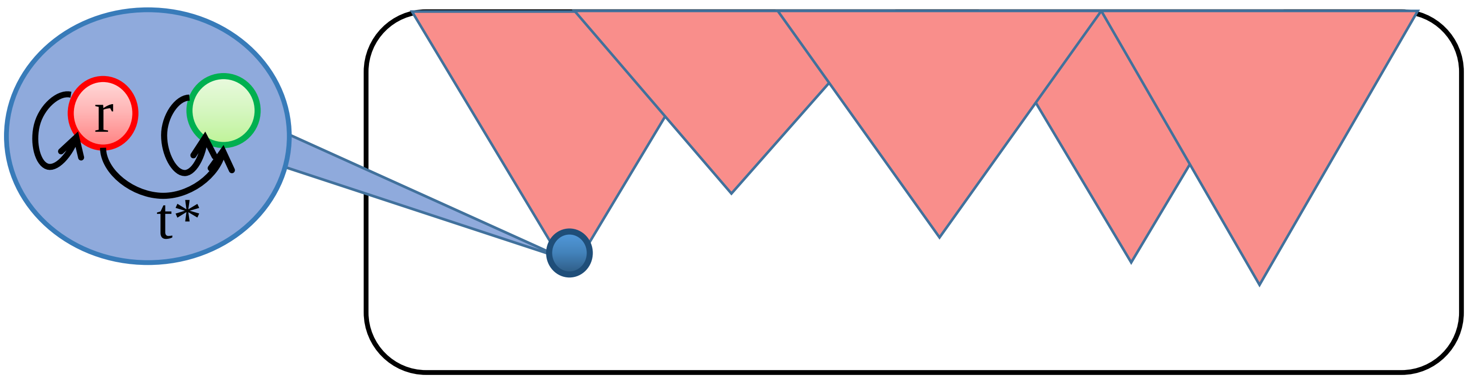
$$\mathrm{Avoid}(\sigma) = \neg\mathrm{gen}(\mathrm{Diag}(\sigma))$$



[CAV'15, JACM] Property-Directed Inference of Universal Invariants or Proving Their Absence, A. Karbyshev, N. Bjorner, S. Itzhaky, N. Rinetzky and S. Shoham.

Universally-Quantified Invariant

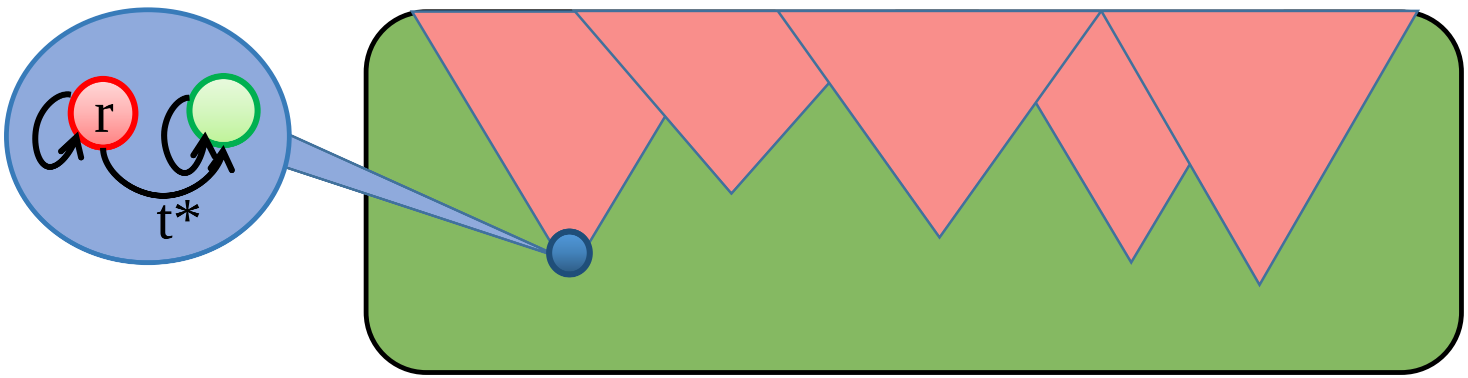
$$\mathrm{Inv} \equiv \underbrace{\forall \bar{x}.\ (l_{1,1}(\bar{x}) \lor \ldots \lor l_{1,m}(\bar{x}))}_{\text{clause / conjecture}} \land \ldots \land \forall \bar{x}.\ (l_{n,1}(\bar{x}) \lor \ldots \lor l_{n,m}(\bar{x}))$$



$$\mathrm{Inv} \equiv \neg \underbrace{\exists \bar{x}.\ (\neg l_{1,1}(\bar{x}) \land \ldots \land \neg l_{1,m}(\bar{x}))}_{\text{cube}} \land \ldots \land \neg \exists \bar{x}.\ (\neg l_{n,1}(\bar{x}) \land \ldots \land \neg l_{n,m}(\bar{x}))$$

Universally-Quantified Invariant

$$\text{Inv} \equiv \underbrace{\forall \bar{x}.\ (l_{1,1}(\bar{x}) \vee \ldots \vee l_{1,m}(\bar{x}))}_{\text{clause / conjecture}} \wedge \ldots \wedge \forall \bar{x}.\ (l_{n,1}(\bar{x}) \vee \ldots \vee l_{n,m}(\bar{x}))$$



Questions:

- How to find the states to generalize from?
- How to select which facts to remove in the generalization?

Next

- UPDR: **Semi-algorithm** for inference of **universal** inductive invariants

- IVy: **Interactive** approach for inferring **universal** inductive invariants

Automatic Synthesis of Universal Invariants

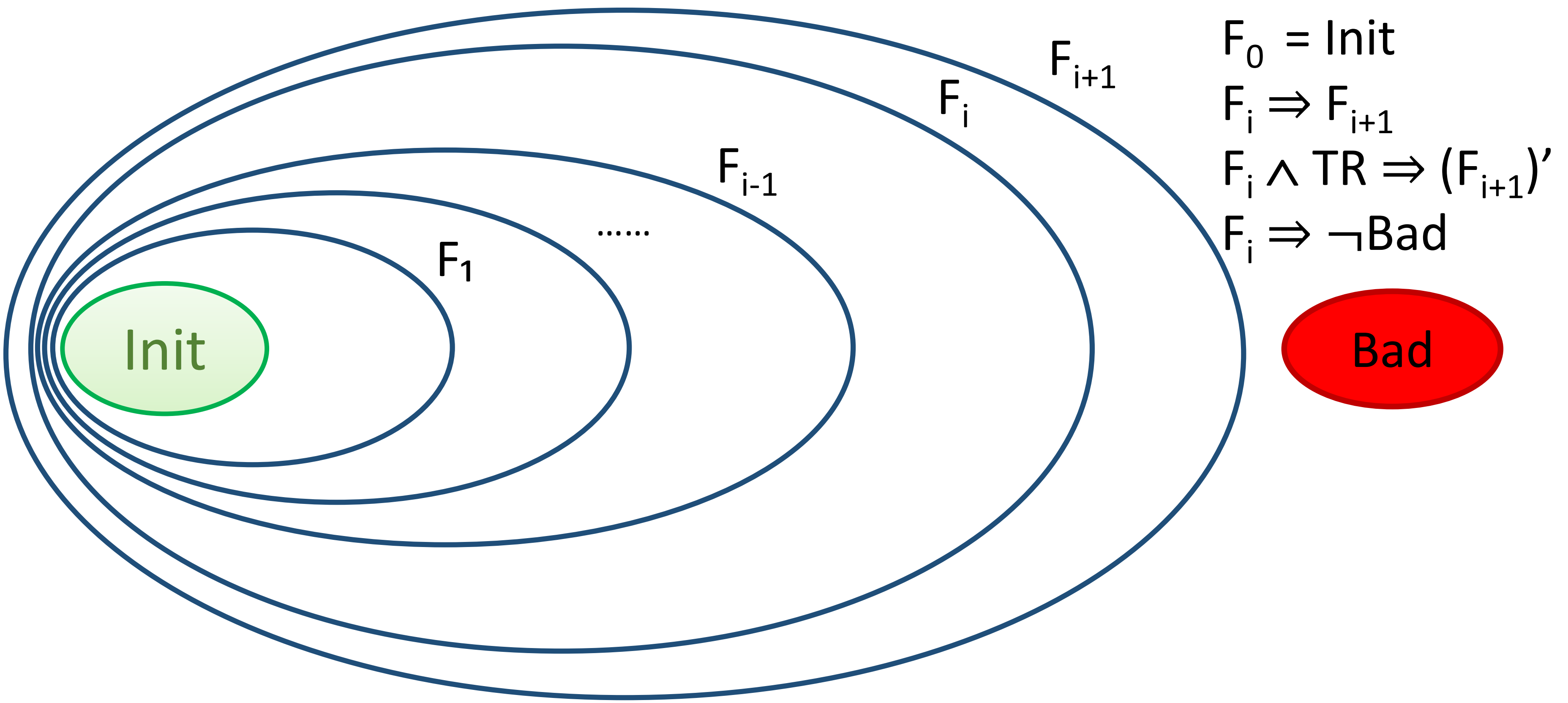
- [CAV’15, JACM] Property-Directed Inference of Universal Invariants or Proving Their Absence, A. Karbyshev, N. Bjorner, S. Itzhaky, N. Rinetzky and S. Shoham.
- [POPL’16] Decidability of Inferring Inductive Invariants, O. Padon, N. Immerman, S. Shoham, A. Karbyshev and M. Sagiv.
- [VMCAI’17] Property Directed Reachability for Proving Absence of Concurrent Modification Errors, A. Frumkin, Y. Feldman, O. Lhoták, O. Padon, M. Sagiv and S. Shoham.

Universal Property Directed Reachability (UPDR)

- Performs automatic generalization
- Based on Bradley's IC3/PDR [VMCAI11,FMCAD11]

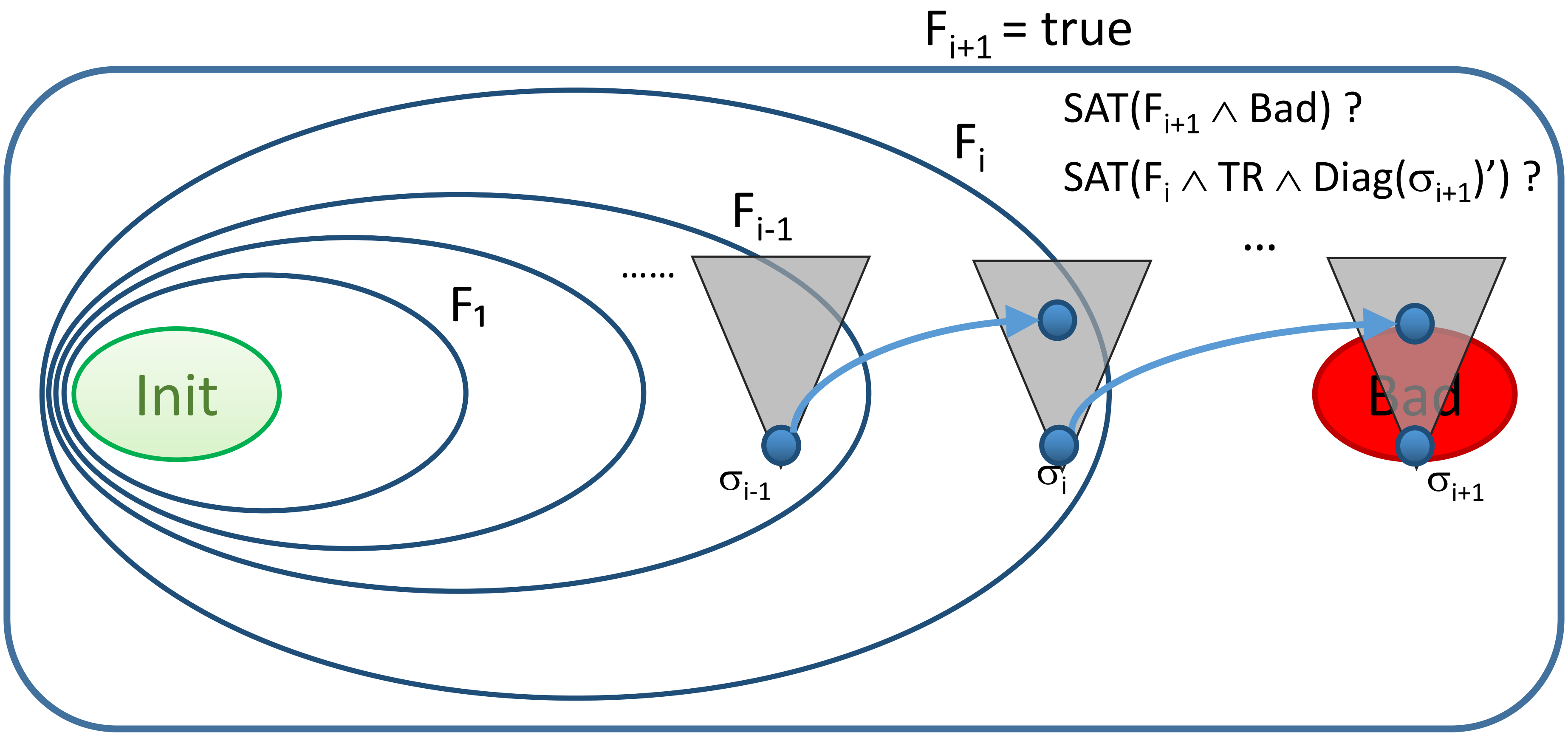
- [CAV'15, JACM] Property-Directed Inference of Universal Invariants or Proving Their Absence, A. Karbyshev, N. Bjorner, S. Itzhaky, N. Rinetzky and S. Shoham.

Property Directed Reachability



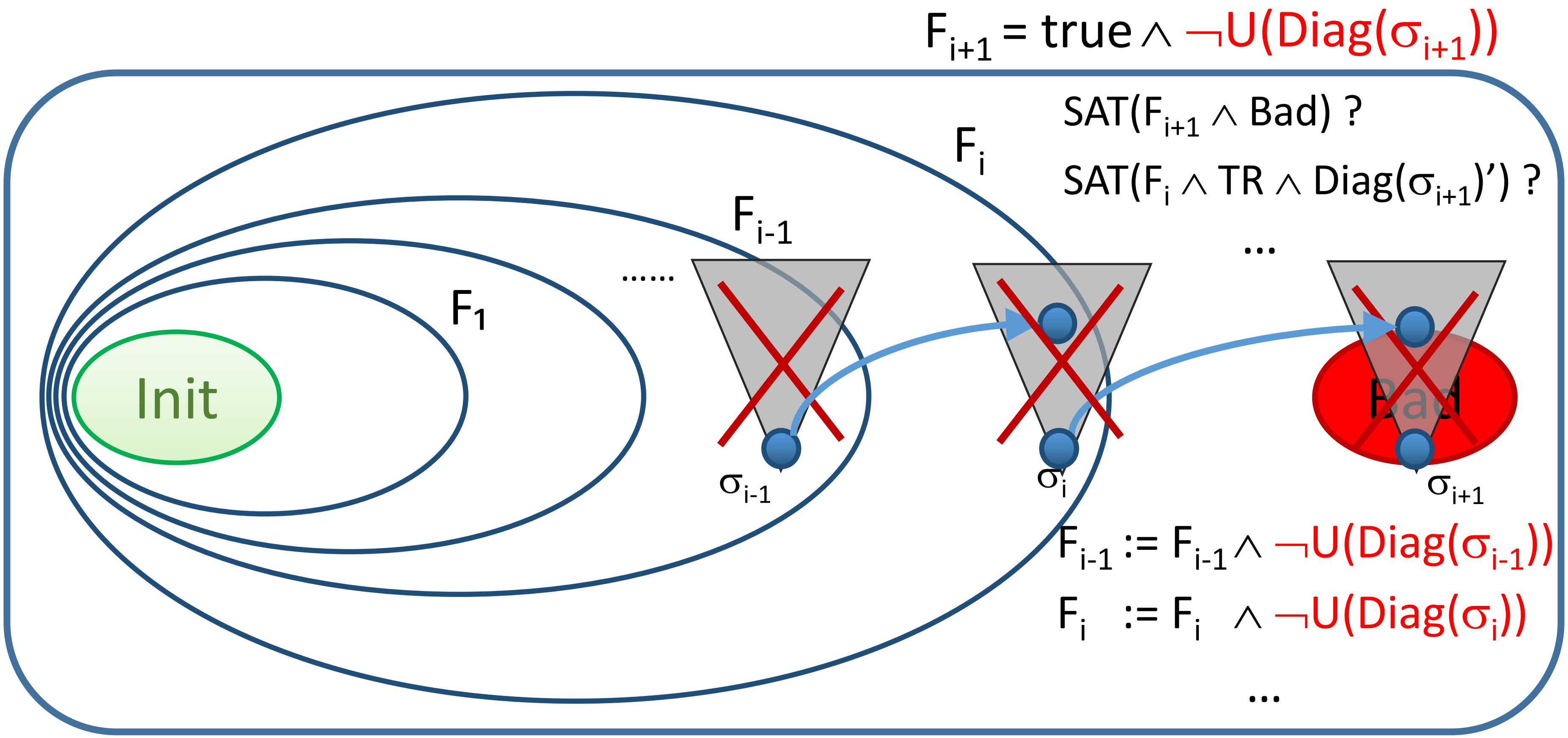
- F_i over-approximates the states that are reachable in at most i steps
- If $F_{k+1} \equiv F_k$ then F_k is an inductive invariant
- Computation of F_i is guided by the property P=$\neg$Bad

How is F_{i+1} computed in (U)PDR?



If Diag(σ_{i+1}) is reachable from F_i: continue backwards until Init

How is F_{i+1} computed in (U)PDR?

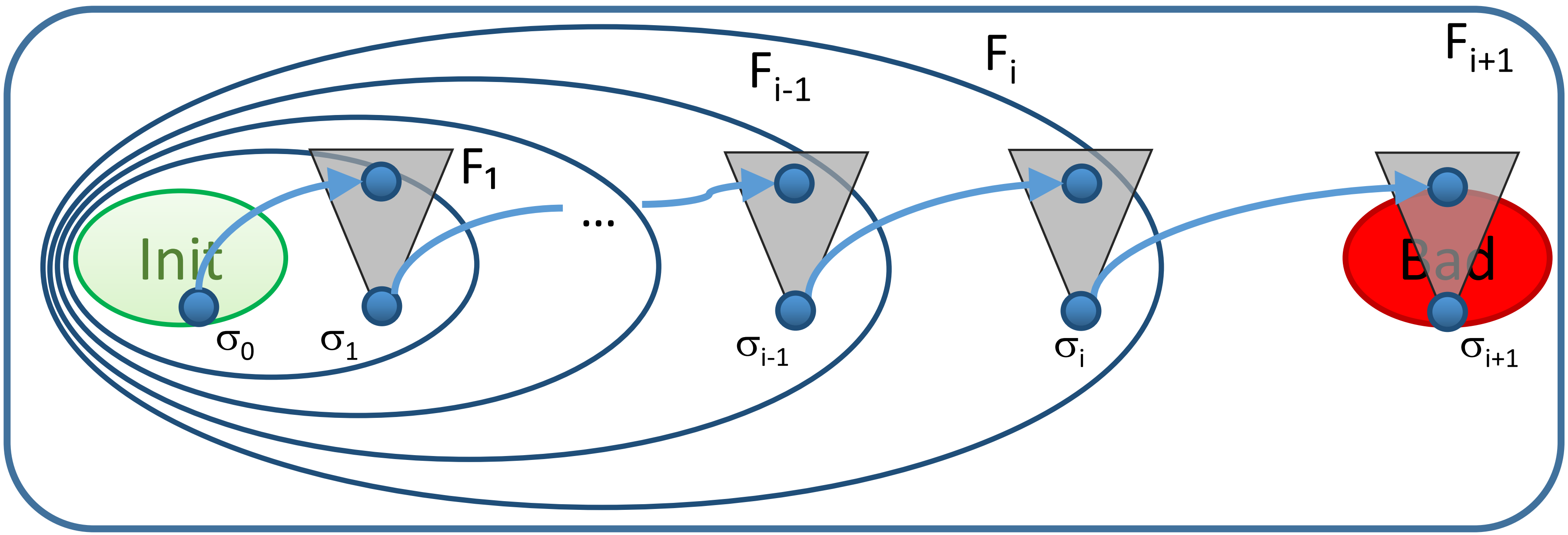


If Diag(σ_{i+1}) is reachable from F_i: continue backwards until Init

If Diag(σ_j) is unreachable from F_{j-1} : strengthen F_j to exclude UnsatCore(Diag(σ_j))

UPDR: Possible Outcomes

- Fixpoint: universal inductive invariant found
  - System is safe
- Abstract counterexample:



UPDR: Possible Outcomes

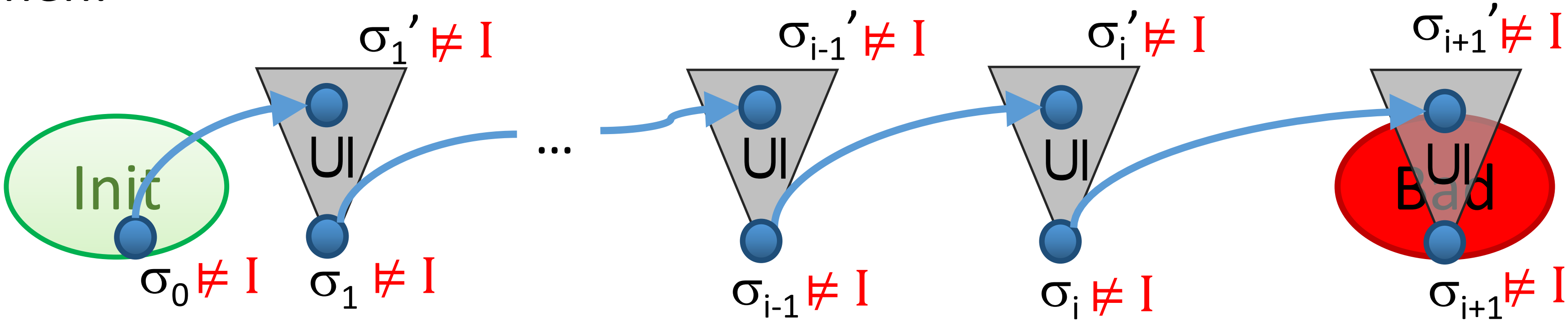
- Fixpoint: universal inductive invariant found
  - System is safe
- Abstract counterexample:
  - Safety not determined*
  - But no universal inductive invariant exists!

* can use Bounded Model Checking to find real counterexamples

Proving the absence of universal Invariant

Suppose that a safety universal invariant I exists.

Then:



I satisfies safety: $\sigma_{i+1} \models$ Bad $\Rightarrow \sigma_{i+1} \not\models \mathrm{I}$

I is universal: $\sigma_{i+1}' \models \mathrm{Diag}(\sigma_{i+1}) \Rightarrow \sigma_{i+1}' \not\models \mathrm{I}$

I satisfies consecution: $\sigma_{i+1}' \not\models \mathrm{I} \wedge \mathrm{TR}(\sigma_i, \sigma_{i+1}') \Rightarrow \sigma_i \not\models \mathrm{I}$

I satisfies initiation: $\sigma_0 \not\models \mathrm{I} \Rightarrow \sigma_0 \not\models$ Init

If there is $\mathrm{I} \in \forall^*$, then any **relaxed trace** does not reach Init

➔ A relaxed trace from Init to Bad implies no $\mathrm{I} \in \forall^*$ exists

Experiments

Used to infer inductive invariants / procedure summaries of:

- Heap-manipulating programs, e.g.
  - Singly-linked list
  - Doubly-linked list
  - Nested lists
  - Iterators in Java - Concurrent modification error (

No need for user-defined predicates/ templates!

- Distributed protocols
  - Spanning tree
  - Learning switch

- [CAV'15, JACM] Property-Directed Inference of Universal Invariants or Proving Their Absence, A. Karbyshev, N. Bjorner, S. Itzhaky, N. Rinetzky and S. Shoham.
- [VMCAI'17] Property Directed Reachability for Proving Absence of Concurrent Modification Errors, A. Frumkin, Y. Feldman, O. Lhoták, O. Padon, M. Sagiv, S. Shoham.

Termination?

Is it decidable to **infer universal** inductive invariants? [POPL’16]

- No, in the general case
  - if the vocabulary contains at least one binary relation which is unrestricted
- Yes, for linked lists
  - if the vocabulary contains only one "transitive closure" binary relation, but as many constants and unary predicates as desired
  - UPDR will also terminate
  - proof uses well-quasi-order and Kruskal’s tree theorem

- [POPL’16] Decidability of Inferring Inductive Invariants, O. Padon, N. Immerman, S. Shoham, A. Karbyshev, and M. Sagiv.

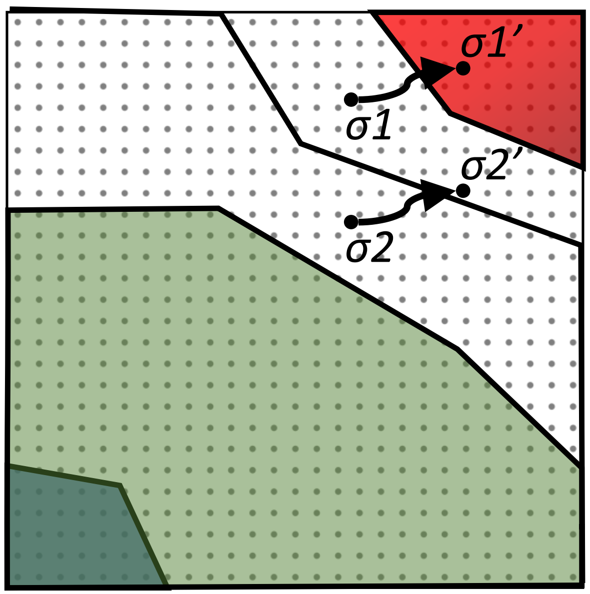
Interactive Synthesis of Universal Invariants

https://github.com/Microsoft/ivy

- [PLDI’16] Ivy: Interactive Verification of Parameterized Systems via Effectively Propositional Reasoning, O.Padon, K. L. McMillan, A. Panda, M. Sagiv and S. Shoham.
- [OOPSLA’17] Paxos Made EPR — Decidable Reasoning about Distributed Protocols. O. Padon, G. Losa, M. Sagiv and S. Shoham.

Invariant Inference in IVy

Iterative strengthening



Inv = $\neg$Bad $\wedge$ "*Avoid*(σ1)" $\wedge$ "*Avoid*(σ2)"…

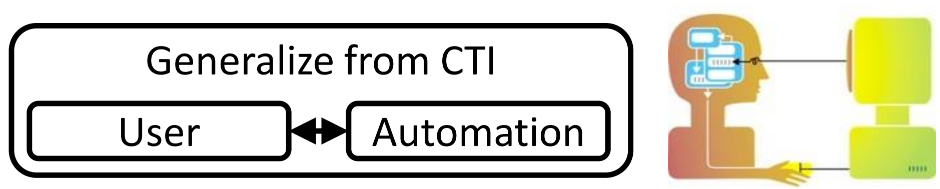
Key challenge for invariant inference:

generalization

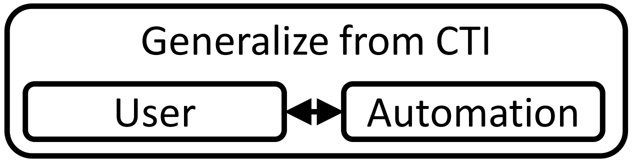
UPDR: diagram + unsat core

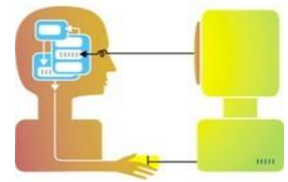
IVy's approach: put the user in the loop

interactive generalization



Interactive Generalization from CTI





1. Generalize by removing "irrelevant" facts to form a conjecture
   - User graphically selects which facts to remove
2. Check if the conjecture is true up to K: BMC(K)
   - User determines the right K to use
   - IVy uses a SAT solver
3. Automatically remove more facts: Interpolate(K)
   - IVy uses the SAT solver to discover more facts to remove
   - User examines the result – it could be wrong

Summary

- Decidable deduction using EPR
  - EPR transition system
  - Inductive invariant Inv ∈ ∀*
- Synthesis of Inv ∈ ∀* by generalization
  - Automatically: UPDR
  - Interactively: IVy
- Key idea: use **diagram** to generalize from counterexamples to induction
  - Can sometimes prove absence of Inv ∈ ∀*