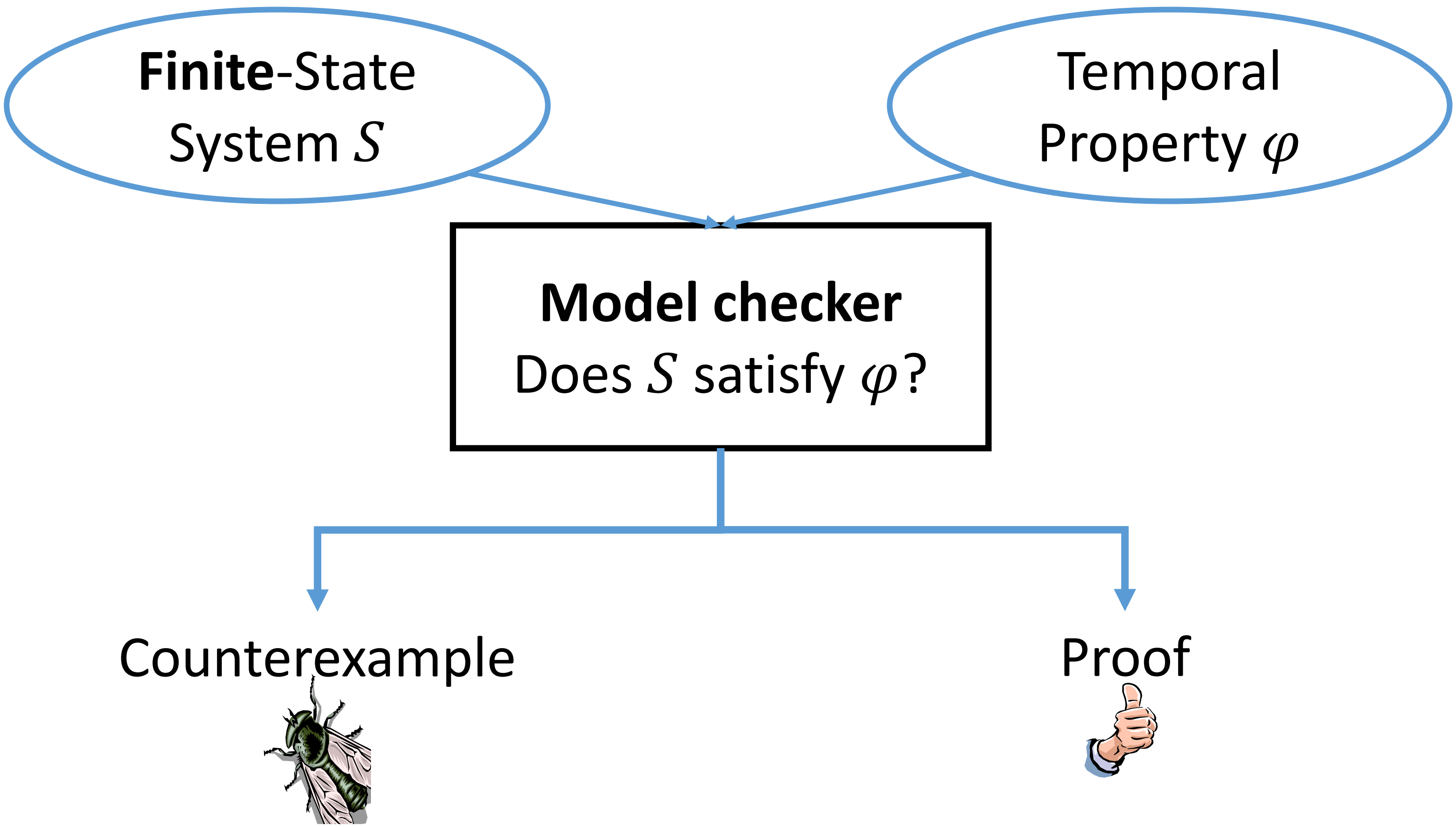
Verification of Infinite-State Systems Using Decidable Logic

Sharon Shoham

Tel Aviv University

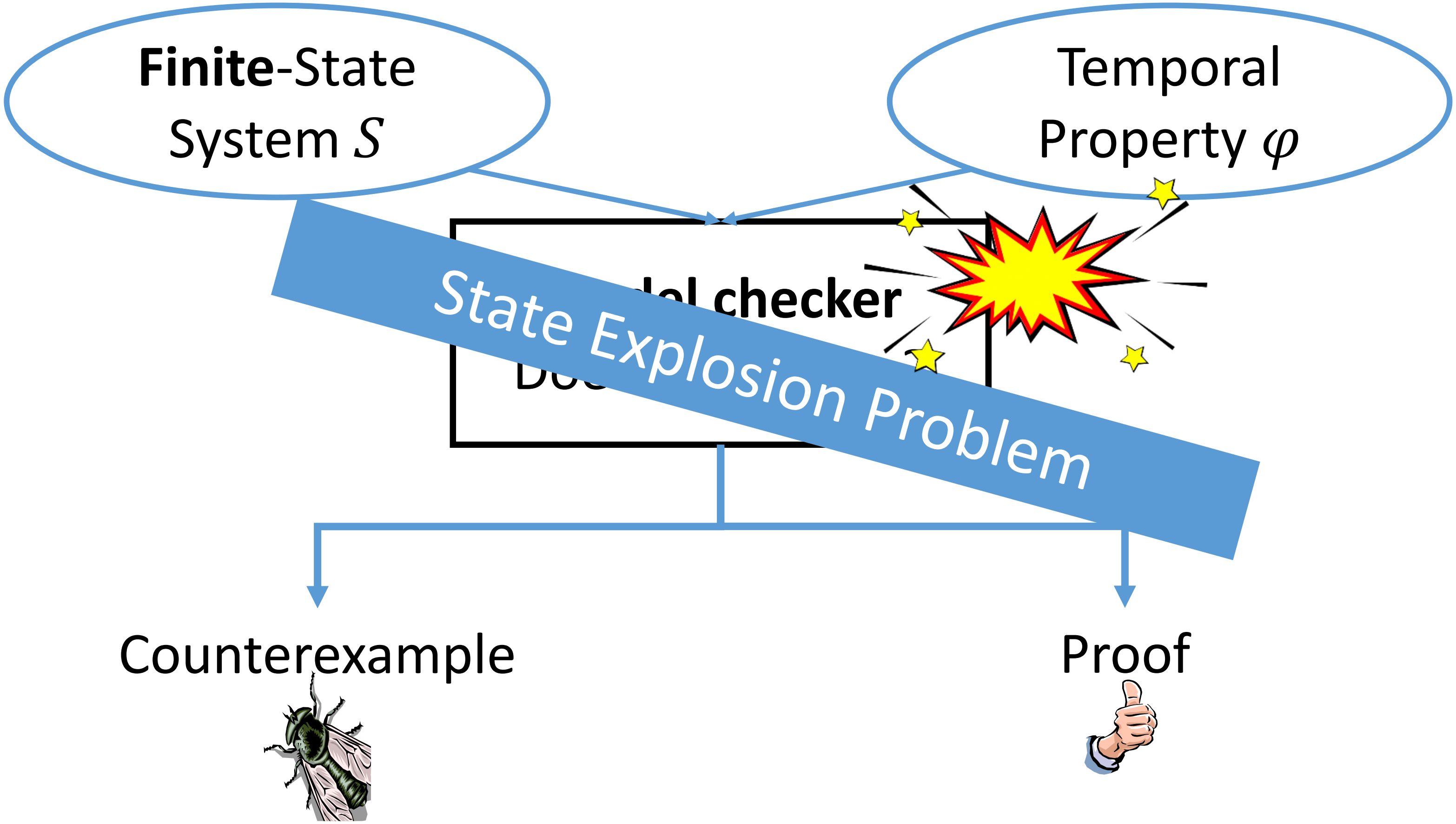
Supervised Verification of Infinite-State Systems

Model Checking [EC81,QS82]



*Clarke, Emerson, and Sifakis won the 2007 Turing award for their contribution to Model Checking

Model Checking [EC81,QS82]



*Clarke, Emerson, and Sifakis won the 2007 Turing award for their contribution to Model Checking

Technion
Israel Institute of Technology

2001-2009

2009-2010 | Stay at home mom

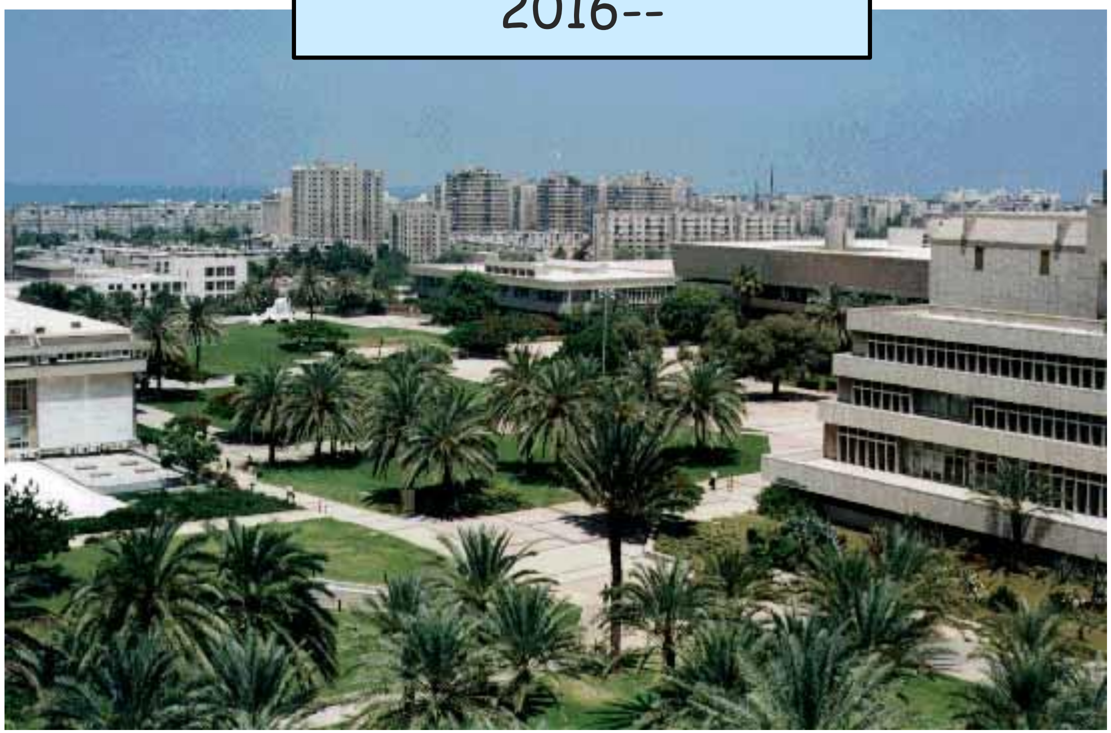
2011 | Postdoc
Technion


2011-2016 | Senior Lecturer
Academic college
of Tel Aviv Yaffo

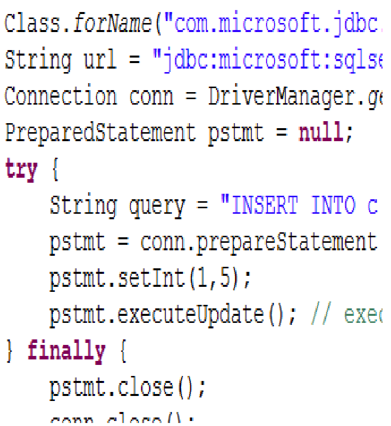
2015-2016 | & Visiting Researcher
Tel Aviv University
(2015-2016)

TEL AVIV UNIVERSITY

2016--



Infinite-State Systems



Programs:

- Dimensions of infinity:
  - unbounded number of dynamically allocated objects
  - unbounded number of threads
  - unbounded domain of variables (naturals, reals…)

Distributed systems:

- Dimensions of infinity:
  - unbounded number of hosts/switches
  - unbounded number of pending messages

Set of states is **infinite**

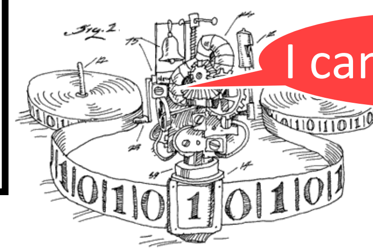
Automatic verification of infinite-state systems

Infinite-State System S

Property φ

Verification
Is there a behavior of S that violates φ?

Rice's Theorem

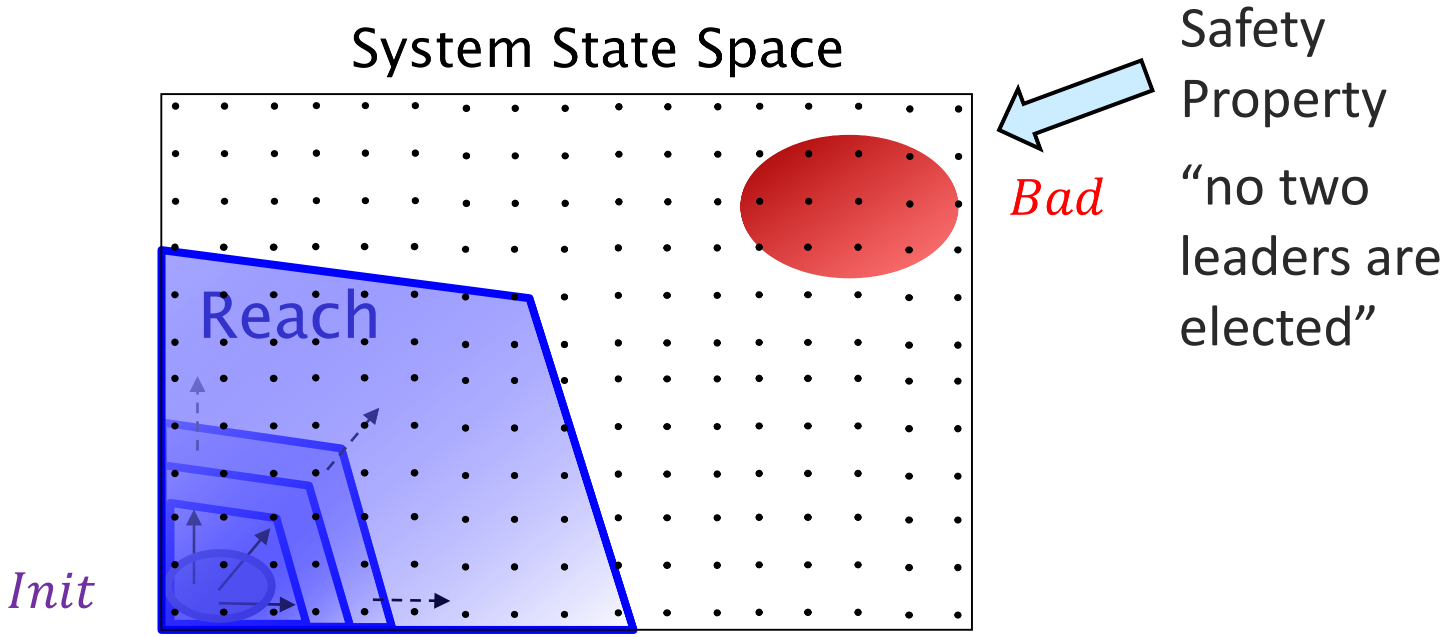


Counterexample

Unknown / Diverge

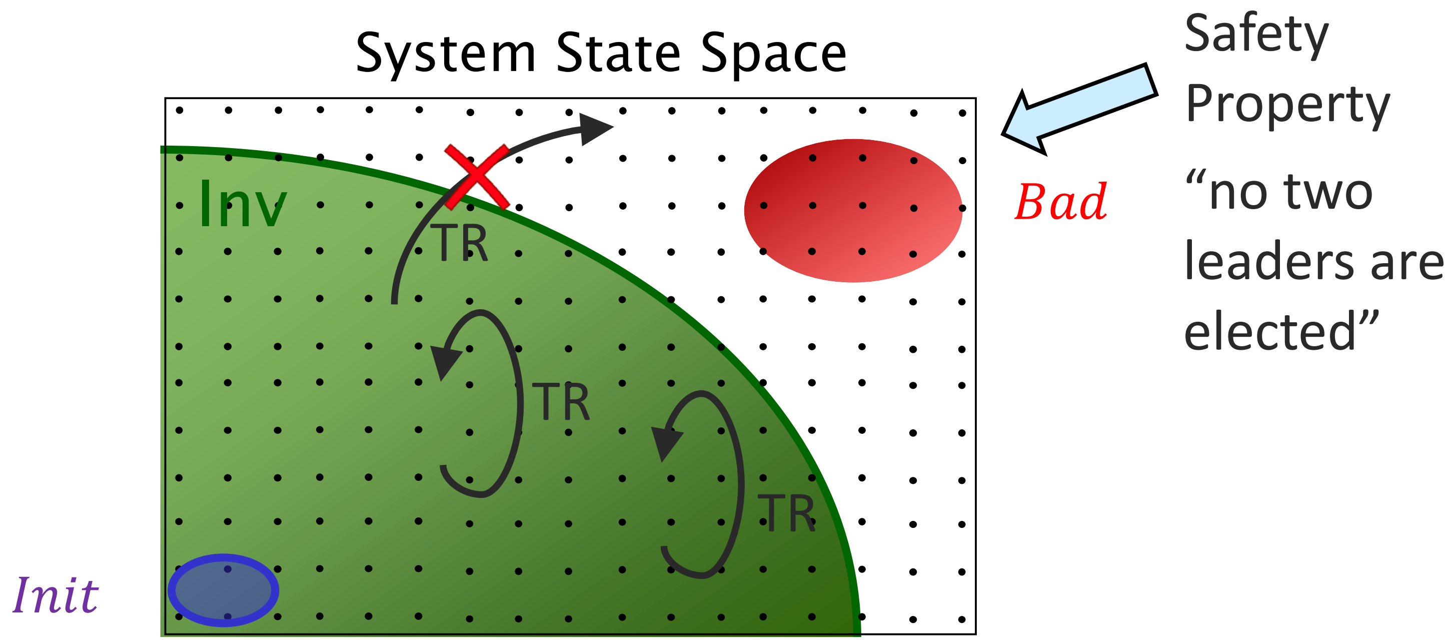
Proof

Safety Verification



System S is **safe** if all the reachable states satisfy the property P = $\neg Bad$

Inductive Invariants



System S is **safe** if all the reachable states satisfy the property P = $\neg Bad$

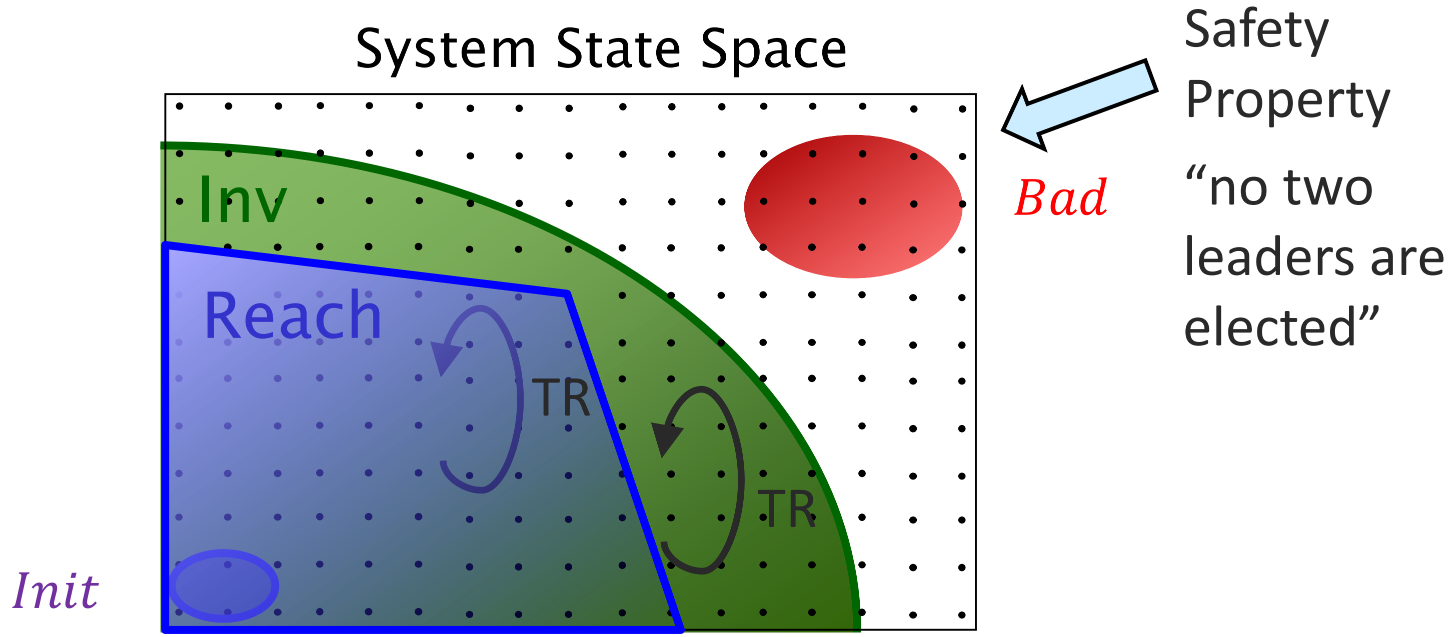
System S is safe iff there exists an **inductive invariant** Inv:

$Init \subseteq Inv$ (**Initiation**)

if $\sigma \in Inv$ and $\sigma \to \sigma'$ then $\sigma' \in Inv$ (**Consecution**)

$Inv \cap Bad = \emptyset$ (**Safety**)

Inductive Invariants



System S is **safe** if all the reachable states satisfy the property P = $\neg Bad$

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

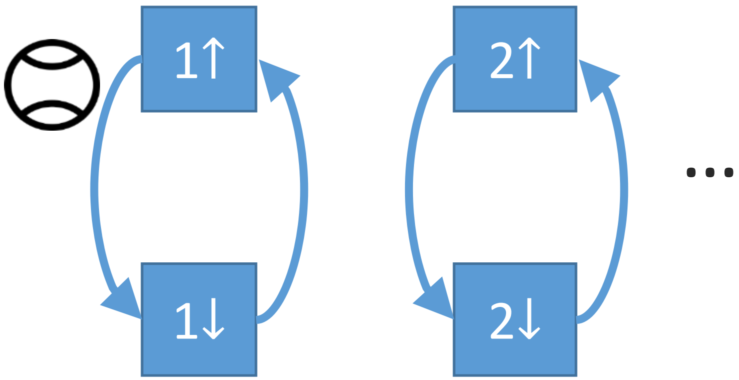
$Init \subseteq Inv$ (**Initiation**)

if $\sigma \in Inv$ and $\sigma \to \sigma'$ then $\sigma' \in Inv$ (**Consecution**)

$Inv \cap Bad = \emptyset$ (**Safety**)

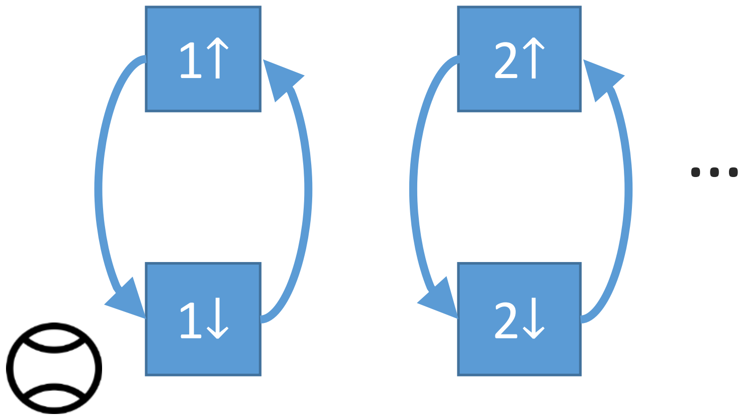
Example

- N pairs of players pass a ball:
  - 1↑ will pass to 1↓
  - 1↓ will pass to 1↑
  - 2↑ will pass to 2↓
  - 2↓ will pass to 2↑ …



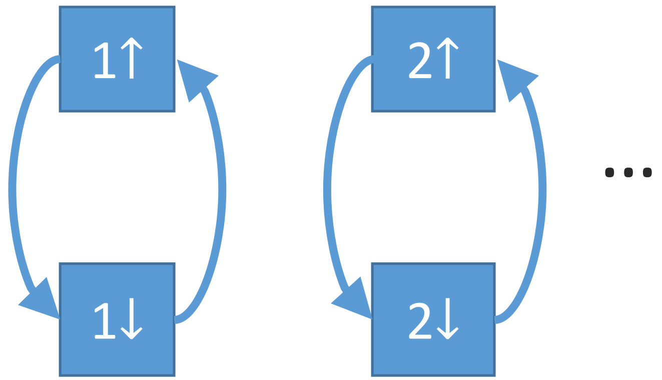
Example

- N pairs of players pass a ball:
  - 1↑ will pass to 1↓
  - 1↓ will pass to 1↑
  - 2↑ will pass to 2↓
  - 2↓ will pass to 2↑ …



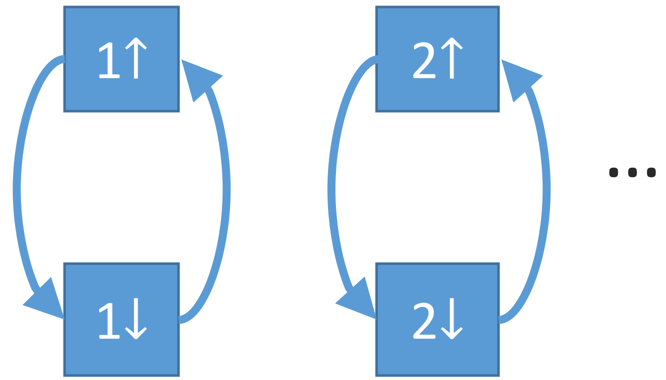
Example

- N pairs of players pass a ball:
  - 1↑ will pass to 1↓
  - 1↓ will pass to 1↑
  - 2↑ will pass to 2↓
  - 2↓ will pass to 2↑ …
- The ball starts at player 1↑
- Can the ball get to 2↓?



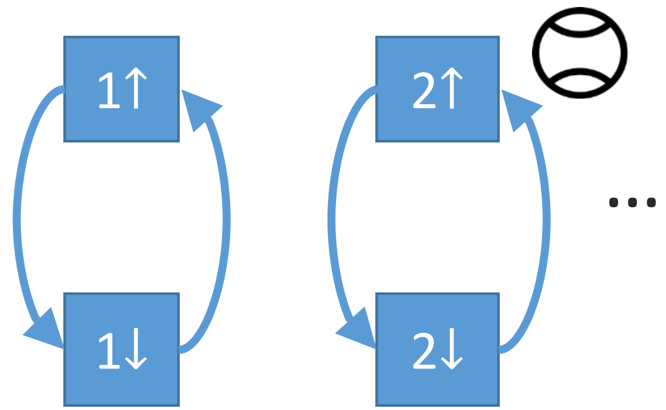
Example

- N pairs of players pass a ball:
  - 1↑ will pass to 1↓
  - 1↓ will pass to 1↑
  - 2↑ will pass to 2↓
  - 2↓ will pass to 2↑ …
- The ball starts at player 1↑
- Can the ball get to 2↓?
- Is "**the ball is not at 2↓**" an inductive invariant?



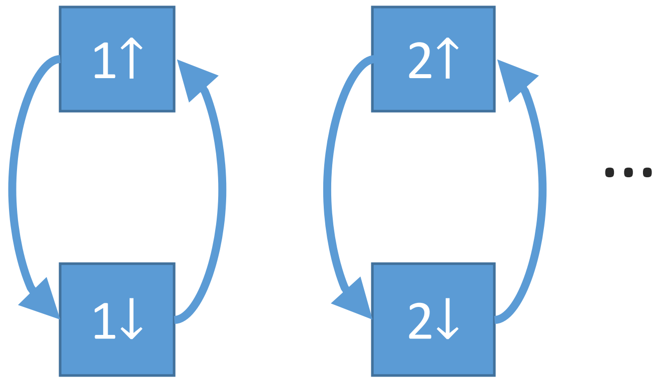
Example

- N pairs of players pass a ball:
  - 1↑ will pass to 1↓
  - 1↓ will pass to 1↑
  - 2↑ will pass to 2↓
  - 2↓ will pass to 2↑ …
- The ball starts at player 1↑
- Can the ball get to 2↓?
- Is "**the ball is not at 2↓**" an inductive invariant? **No!**
  - **Counterexample to induction**



Example

- N pairs of players pass a ball:
  - 1↑ will pass to 1↓
  - 1↓ will pass to 1↑
  - 2↑ will pass to 2↓
  - 2↓ will pass to 2↑ …



- The ball starts at player 1↑
- Can the ball get to 2↓?
- Is "**the ball is not at 2↓**" an inductive invariant? **No!**
  - **Counterexample to induction**
- Inductive invariant: "**the ball is not at 2↑ or 2↓**"

Logic-based verification

- Represent $Init$, Tr, Bad, Inv by logical formulas
  - Formula ⇔ Set of states

- Automated solvers for logical satisfiability made huge progress
  - Propositional logic (SAT) – industrial impact for hardware verification
  - First-order theorem provers
  - Satisfiability modulo theories (SMT) – major trend in software verification
  - Z3, CVC4, iProver, Vampire ….

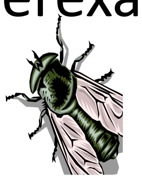
How can we **check** an inductive invariant?

Inv(V) is an inductive invariant if the following verification conditions are valid:

Initiation	$Init(V) \Longrightarrow Inv(V)$	unsat($Init(V) \wedge \neg Inv(V)$)
Cons.	$Inv(V) \wedge TR(V,V') \Longrightarrow Inv(V')$	unsat($Inv(V) \wedge TR(V,V') \wedge \neg Inv(V')$)
Safety	$Inv(V) \Longrightarrow \neg Bad(V)$	unsat($Inv(V) \wedge Bad(V)$)

Are the logical VC's valid ?

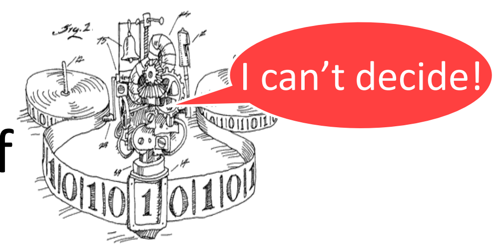
Counterexample



Unknown / Diverge

Proof

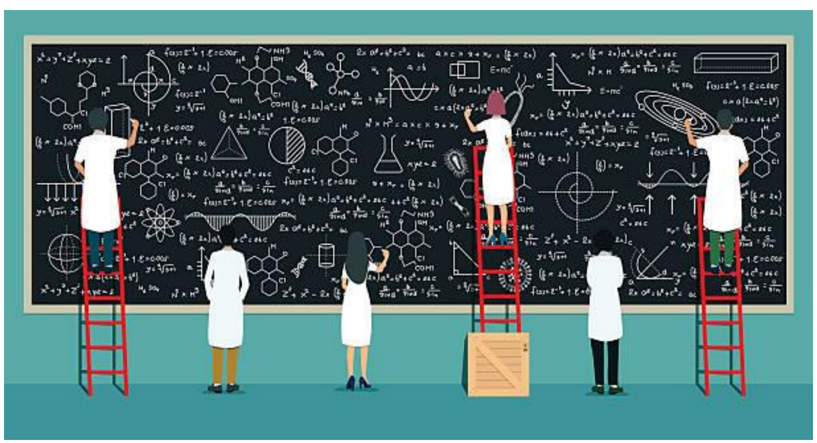
Church's Theorem



What can we do about it?

Interactive theorem provers
(Coq, Isabelle/HOL, LEAN)

- Programmer gives inductive invariant and proves it
- Huge programmer effort (~10-50 lines of proof per line of code)



SMT-based deductive verification
(e.g. Dafny)

- Programmer provides ind. invariant
- VC's generated and discharged automatically
- SMT solver may diverge (matching loops, arithmetic)

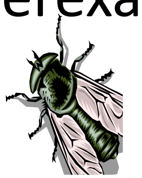
Alternative: Restrict VC's to decidable logic

Inv(V) is an inductive invariant if the following verification conditions are valid:

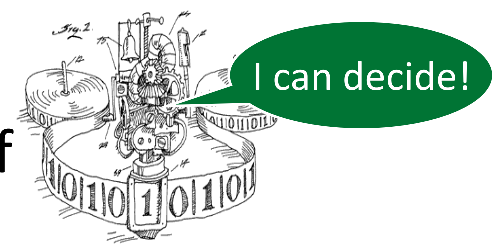
Initiation	$Init(V) \Longrightarrow Inv(V)$	unsat($Init(V) \wedge \neg Inv(V)$)
Cons.	$Inv(V) \wedge TR(V,V') \Longrightarrow Inv(V')$	unsat($Inv(V) \wedge TR(V,V') \wedge \neg Inv(V')$)
Safety	$Inv(V) \Longrightarrow \neg Bad(V)$	unsat($Inv(V) \wedge Bad(V)$)

Are the logical VC's valid ? $\in$ Decidable logic

Counterexample



Proof



Challenges for verification with decidable logic

Formal specification

Modeling in a decidable logic

Deduction

Checking validity of the VC's ✓

Invariant inference

Finding an inductive invariant

This talk

Formal specification

- Use EPR – decidable fragment of first order logic
- Surprisingly expressive

Invariant inference

- Automatic (based on PDR)
  - Semi-algorithm: may diverge
- Interactive
  - Based on graphically displayed counterexamples to induction

Effectively Propositional Logic – EPR

Decidable fragment of first order logic

+ Quantification (∃*∀*) - Theories (e.g., arithmetic)

☺ Allows quantifiers to reason about unbounded sets

- $\forall x,y.\ \textbf{leader}(x) \land \textbf{leader}(y) \rightarrow x = y$

☺ Satisfiability is decidable => Deduction is decidable

☺ Small model property => Finite cex to induction

☺ Turing complete modeling language

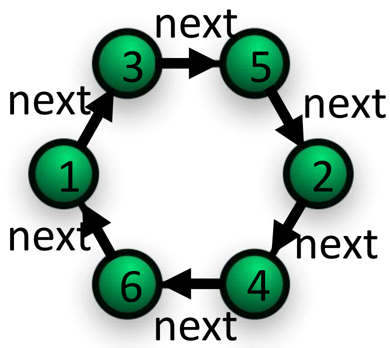
☹ Limited language for safety and inductive invariants

➢ Suffices for many infinite-state systems

Successful verification with EPR

- Shape Analysis
  [Itzhaky et al. CAV’13, POPL’14, CAV’14, CAV’15]
- Software-Defined Networks
  [Ball et al. PLDI’14]
- Distributed protocols
  [Padon et al. PLDI’16, OOPSLA’17, POPL’18, PLDI’18]
- Concurrent Modification Errors in Java programs
  [Frumkin et al. VMCAI’17]

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

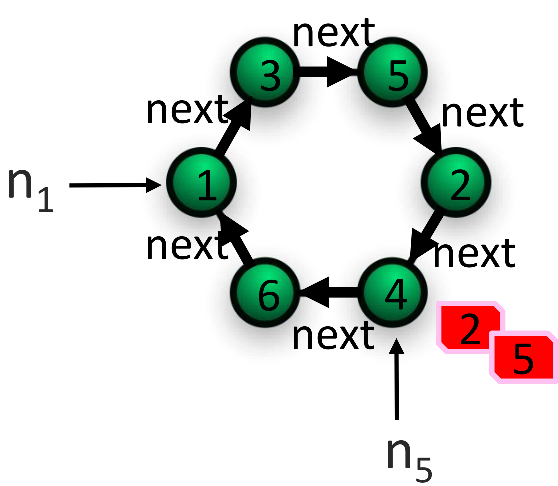
[CACM'79] E. Chang and R. Roberts. *An improved algorithm for decentralized extrema-finding in circular configurations of processes*

Modeling with EPR

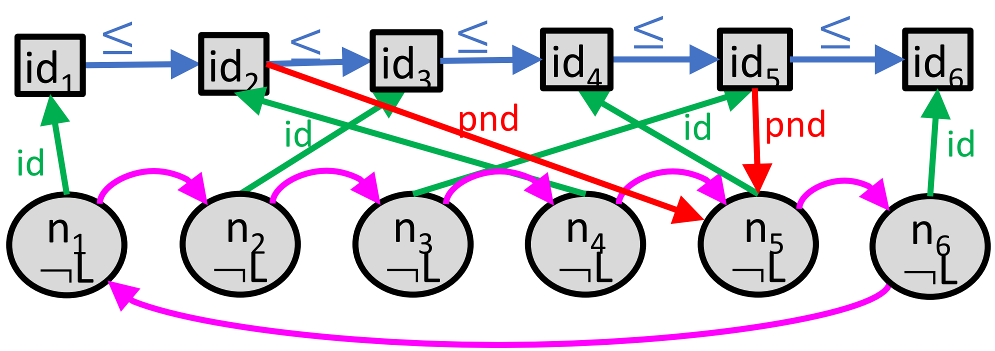
- State: finite first-order structure over vocabulary V
  - $\leq$ (ID, ID) – total order on node id's
  - **btw** (Node, Node, Node) – the ring topology
  - **id**: Node → ID – relate a node to its id
  - **pending**(ID, Node) – pending messages
  - **leader**(Node) – leader(n) means n is the leader

Axiomatized in EPR

protocol state



structure



$\langle n_5, n_1, n_3 \rangle \in I(\textbf{btw})$

Modeling with EPR

- State: finite first-order structure over vocabulary V (+ axioms)

- 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., $\exists\, n_1, n_2.\,\textbf{leader}(n_1) \land \textbf{leader}(n_2) \land n_1 \neq n_2$

- Transition relation: expressed as EPR formula TR(V, V'), e.g.:

  $\exists$n,s. "s = **next**(n)" $\land$ $\forall$x,y. **pending'**(x,y) $\leftrightarrow$ (**pending**(x,y) $\lor$ (x=**id**[n] $\land$ y=s))

  $\lor$ $\exists$n. **pending** (**id**[n],n) $\land$ $\forall$x. **leader'**(x) $\longleftrightarrow$ (**leader**(x) $\lor$ x=n)

  ...

Modeling with EPR

- State: finite first-order structure over vocabulary V (+ axioms)

Propose(n): send(id(n), next(n))

Recv(n,msg): if msg = id(n) then leader(n) := true

if msg > id(n) then send(msg,next(n))

- Transition relation: expressed as EPR formula TR(V, V'), e.g.:

$\exists n,s.$ "$s = \mathbf{next}(n)$" $\land\ \forall x,y.\ \mathbf{pending'}(x,y) \leftrightarrow (\mathbf{pending}(x,y) \lor (x\mathbf{=id}[n] \land y=s))$

$\lor\ \exists n.\ \mathbf{pending}\ (\mathbf{id}[n],n) \land \forall x.\ \mathbf{leader'}(x) \longleftrightarrow (\mathbf{leader}(x) \lor x=n)$

...

Modeling with EPR

- State: finite first-order structure over vocabulary V (+ axioms)

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

- Transition relation: expressed as EPR formula TR(V, V'), e.g.:

  $\exists n,s.$ "$s = \textbf{next}(n)$" $\wedge\ \forall x,y.\ \textbf{pending'}(x,y) \leftrightarrow (\textbf{pending}(x,y) \vee (x{=}\textbf{id}[n] \wedge y{=}s))$

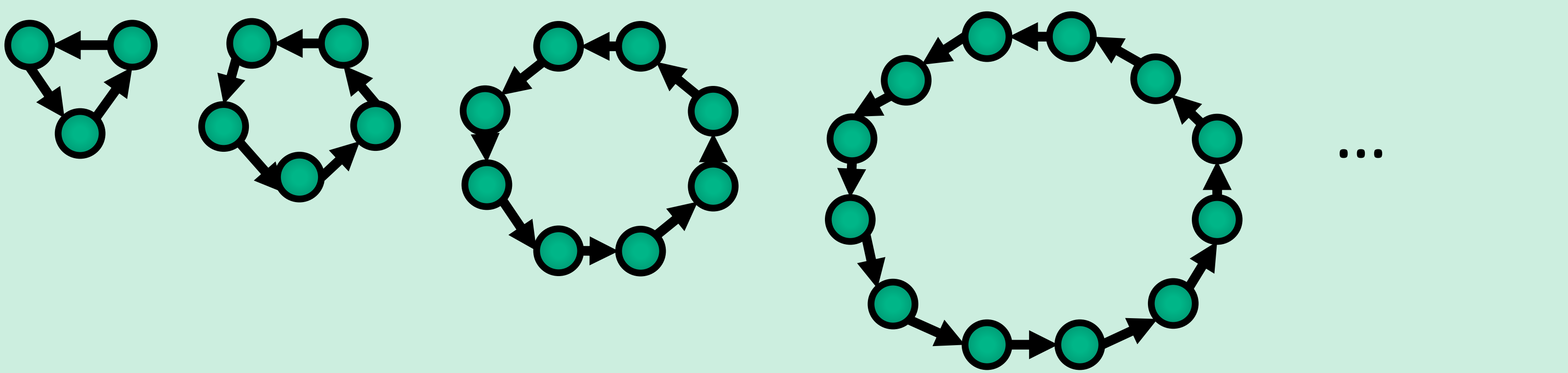
  $\vee\ \exists n.\ \textbf{pending}\ (\textbf{id}[n],n) \wedge \forall x.\ \textbf{leader'}(x) \longleftrightarrow (\textbf{leader}(x) \vee x{=}n)$ ...

Modeling with EPR

- State: finite first-order structure over vocabulary V (+ axioms)

- 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., $\exists\, n_1,n_2.$**leader**$(n_1) \land$ **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 $Init(V), Bad(V), TR(V, V') \in EPR$
- Inductive invariant $Inv(V) \in \forall^*$
- Verification conditions

Initiation	$Init(V) \Longrightarrow Inv(V)$	unsat($Init(V) \wedge \neg Inv(V)$)
Cons.	$Inv(V) \wedge TR(V,V') \Longrightarrow Inv(V')$	unsat($Inv(V) \wedge TR(V,V') \wedge \neg Inv(V')$)
Safety	$Inv(V) \Longrightarrow \neg Bad(V)$	unsat($Inv(V) \wedge Bad(V)$)

Verification conditions ∈ EPR

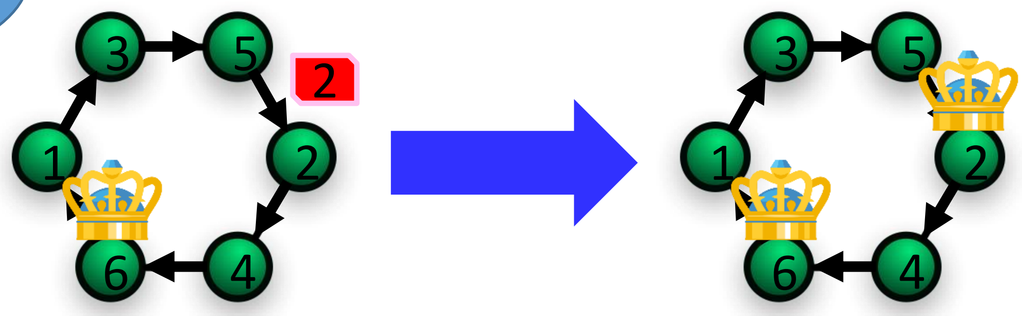
➔ Decidable to check

Inductive Invariant for Leader Election

Safety property:

$$I_0 = \neg\mathrm{Bad} = \forall x,y\colon \mathrm{Node}.\ \mathbf{leader}(x) \land \mathbf{leader}(y) \to x = y$$

Inductive?
No!



- $\leq$ (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

Inductive Invariant for Leader Election

Safety property:

$$I_0 = \neg\mathrm{Bad} = \forall x,y\colon \mathrm{Node}.\ \mathbf{leader}(x) \land \mathbf{leader}(y) \rightarrow x = y$$

Inductive invariant: $\mathrm{Inv} = I0 \land I1 \land I2 \land I3$

$$I_1 = \forall n_1,n_2\colon \mathrm{Node}.\ \mathbf{leader}(n_1) \rightarrow id[n_2] \leq id[n_1]$$

The leader has the highest id

$$I_2 = \forall n_1,n_2\colon \mathrm{Node}.\ pnd(id[n_1], n_1) \rightarrow id[n_2] \leq id[n_1]$$

Only highest id can be self-pnd

$$I_3 = \forall n_1,n_2,n_3\colon \mathrm{Node}.\ btw(n_1,n_2,n_3) \land pnd(id[n_2], n_1) \rightarrow id[n_3] \leq id[n_2]$$

Cannot bypass higher nodes

The reason for using "btw" instead of "next"

- $\leq$ (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

Inductive Invariant for Leader Election

Safety property:

$I_0 = \neg\mathrm{Bad} = \forall x,y\text{: Node. } \mathbf{leader}(x) \land \mathbf{leader}(y) \rightarrow x = y$

Inductive invariant: $\mathrm{Inv} = I0 \land I1 \land I2 \land I3$

$I_1 = \forall n_1,n_2\text{: Node. } \mathbf{leader}(n_1) \rightarrow \mathrm{id}[n_2] \leq \mathrm{id}[n_1]$

The leader has the highest id

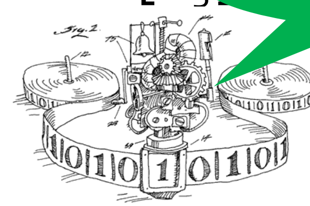
$I_2 = \forall n_1,n_2\text{: Node. } \mathrm{pnd}(\mathrm{id}[n_1], n_1) \rightarrow \mathrm{id}[n_2] \leq \mathrm{id}[n_1]$

Only highest id can be self-pnd

$I_3 = \forall n_1,n_2,n_3\text{: Node. } \mathrm{btw}(n_1,n_2,n_3) \land \mathrm{pnd}(\mathrm{id}[n \ldots], n_1)$
$\rightarrow \mathrm{id}[n_3 \ldots$

Cannot bypass higher nodes

I *can* decide EPR!



$Init(V) \land \neg Inv(V)$
$Inv(V) \land TR(V,V') \land \neg Inv(V')$
$Inv(V) \land Bad(V)$

EPR Solver

Proof

So far

Formal specification

- Use EPR – decidable fragment of first order logic
- Surprisingly expressive
  - Integers: numeric id's expressed with $\leq$
  - Transitive closure: ring topology expressed with btw
  - Network semantics: pending messages
  - Sets and cardinalities (for consensus protocols)

Not in this talk

Next

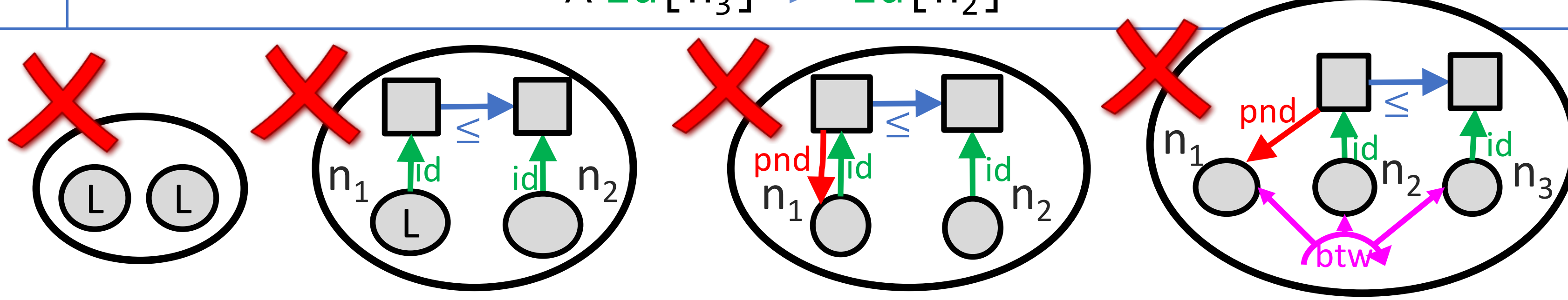
Invariant inference: finding inductive invariants

- Automatically
  - Adapt techniques from finite-state model checking (PDR)
- Interactively
  - Based on graphically displayed counterexamples to induction

How can we find a universally quantified inductive invariant?

Inductive Invariant for Leader Election

I_0 $\neg$Bad	$\forall n_1,n_2 : \text{Node}.\ \textbf{leader}(n_1) \land \textbf{leader}(n_2) \rightarrow n_1 = n_2$ $\neg\exists n_1,n_2 : \text{Node}.\ \textbf{leader}(n_1) \land \textbf{leader}(n_2) \land n_1 \neq n_2$	At most one leader elected
I_1	$\forall n_1,n_2 : \text{Node}.\ \textbf{leader}(n_1) \rightarrow \text{id}[n_2] \leq \text{id}[n_1]$ $\neg\exists n_1,n_2 : \text{Node}.\ \textbf{leader}(n_1) \land \text{id}[n_2] > \text{id}[n_1]$	The leader has the highest id
I_2	$\forall n_1,n_2 : \text{Node}.\ \text{pnd}(\text{id}[n_1],n_1) \rightarrow \text{id}[n_2] \leq \text{id}[n_1]$ $\neg\exists n_1,n_2 : \text{Node}.\ \text{pnd}(\text{id}[n_1],n_1) \land \text{id}[n_2] > \text{id}[n_1]$	Only highest id can be self-pnd
I_3	$\forall n_1,n_2,n_3 : \text{Node}.\ \text{btw}(n_1,n_2,n_3) \land \text{pnd}(\text{id}[n_2],n_1) \rightarrow \text{id}[n_3] \leq \text{id}[n_2]$ $\neg\exists n_1,n_2,n_3 : \text{Node}.\ \text{btw}(n_1,n_2,n_3) \land \text{pnd}(\text{id}[n_2],n_1) \land \text{id}[n_3] > \text{id}[n_2]$	Cannot bypass higher nodes



Construct Inv by excluding "bad" states

1. How to find these states?
2. How to generalize into conjectures?

Generalization using Diagram

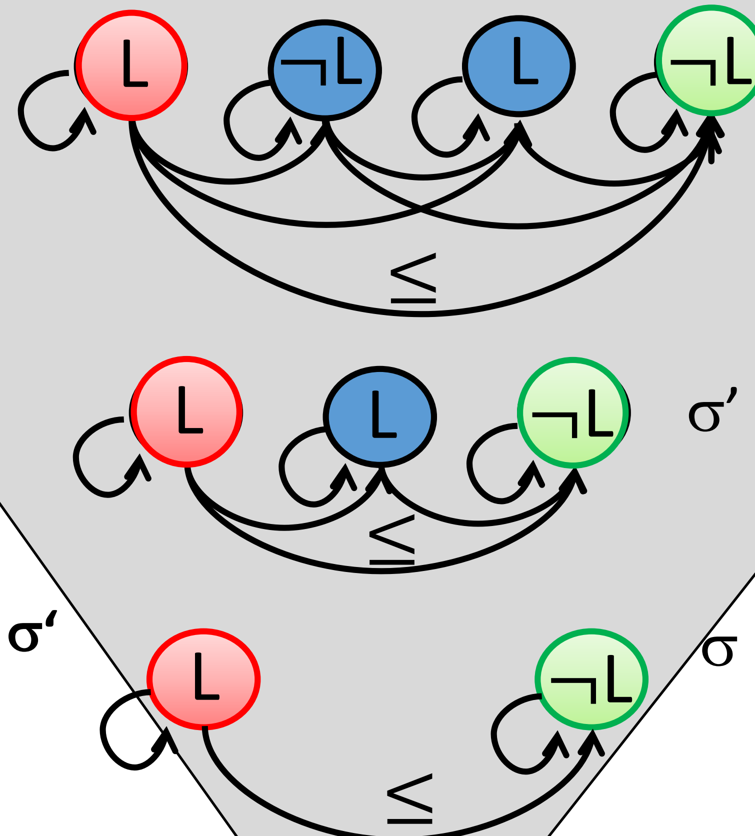
Use **diagrams** as generaliza

- 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



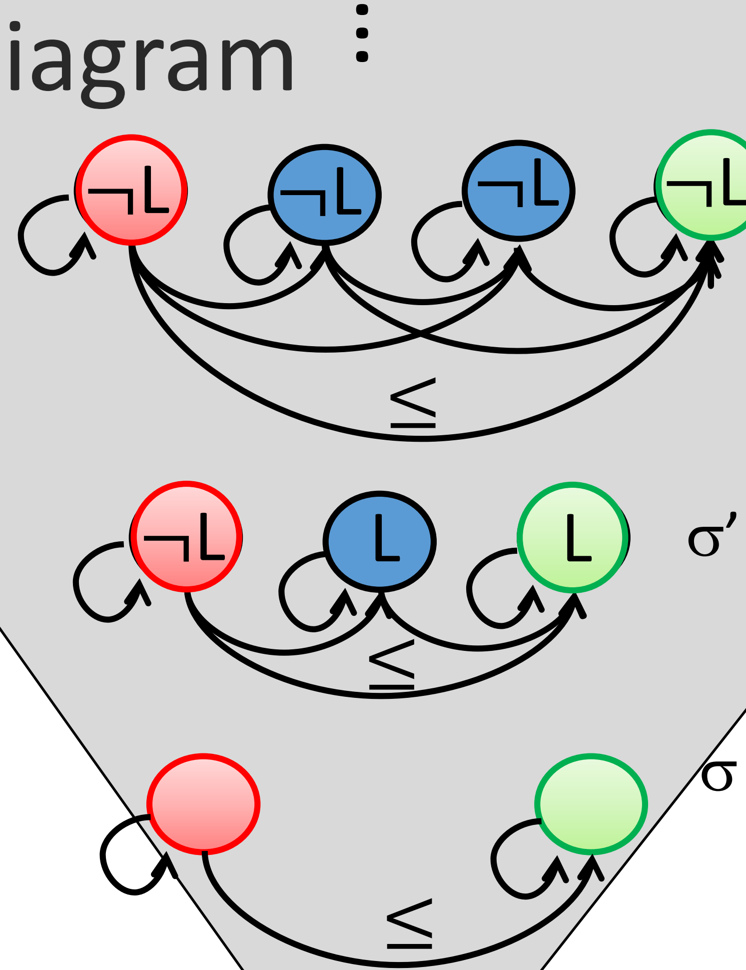
$$\mathrm{exclude}(\sigma) = \neg \mathrm{Diag}(\sigma)$$

[CAV’15, JACM’17] Property-Directed Inference of Universal Invariants or Proving Their Absence, A. Karbyshev, N. Bjorner, S. Itzhaky, N. Rinetzky and S. Shoham.

Generalization using Diagram

Generalize even more if
σ is a partial structure

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

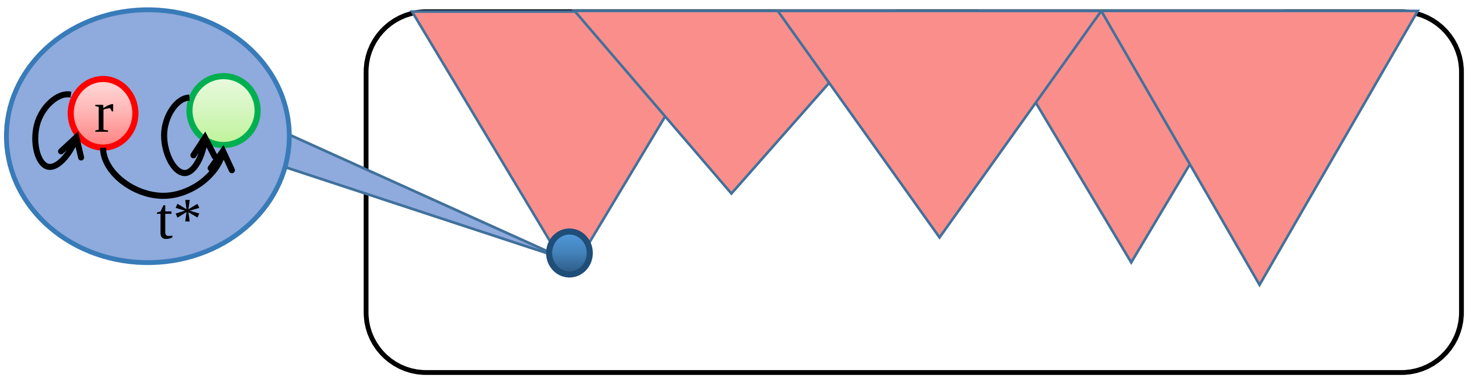


exclude(σ) = ¬Diag(σ)

[CAV'15, JACM'17] Property-Directed Inference of Universal Invariants or Proving Their Absence, A. Karbyshev, N. Bjorner, S. Itzhaky, N. Rinetzky and S. Shoham.

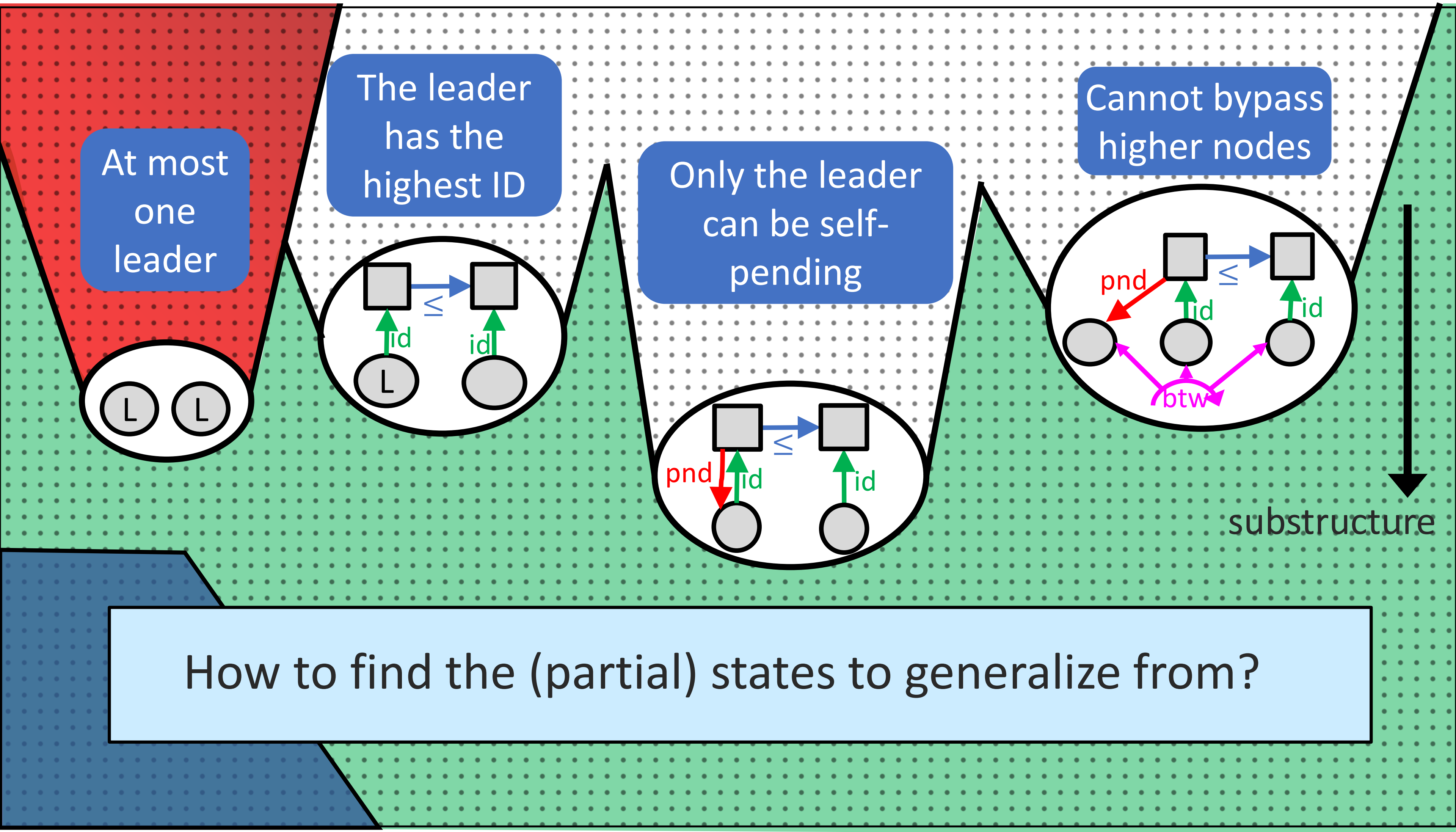
∀* Invariant - excluded substructures

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



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

Leader election example



How to find the (partial) states to generalize from?

(1) UPDR: Automatic inference

- Based on Bradley's IC3/PDR [VMCAI11,FMCAD11]
  - SAT-based verification of finite-state systems
- Abstracts concrete states using their logical diagram
- Backward traversal performed over diagrams
- Blocking of CTI excludes a *generalization* of its diagram ➔ generates universally quantified lemmas

---

- [CAV'15, JACM'17] 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 and S. Shoham.

UPDR: Possible outcomes

- Universal inductive invariant found
  - System is safe

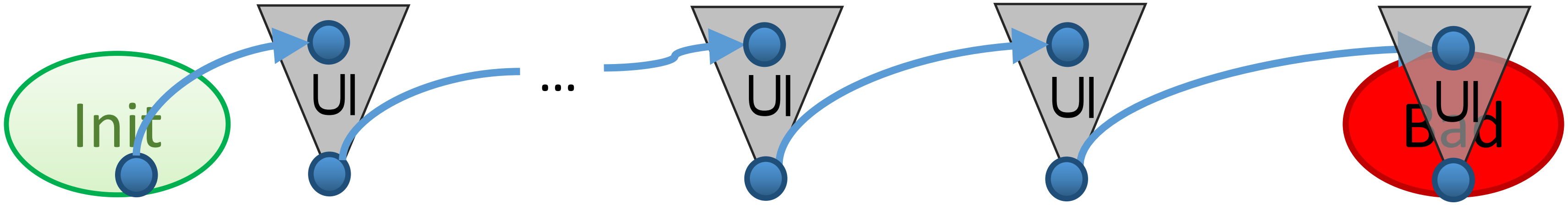
Used to infer inductive invariants / procedure summaries of:

- Heap-manipulating programs, e.g.
  - Singly/Doubly/Nested linked list
  - Iterators in Java - Concurrent modification error (CME)
- Distributed protocols
  - Spanning tree
  - Learning switch
  - …

No need for user-defined predicates/ templates!

UPDR: Possible outcomes

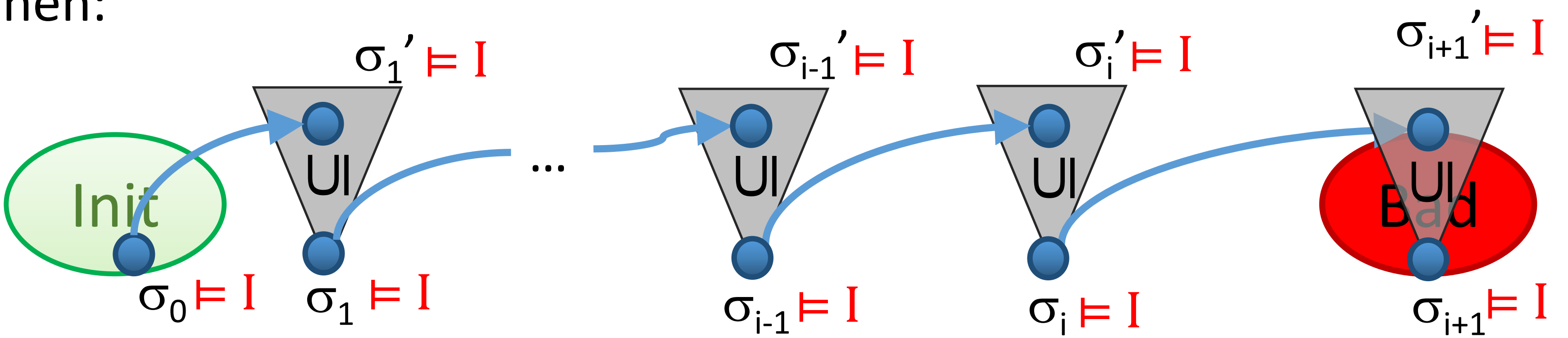
- Universal inductive invariant found
  - System is safe
- Abstract counterexample:
  - Safety not determined*
  - But no universal inductive invariant exists!



* can use Bounded Model Checking to find real counterexamples

Proving the absence of universal invariant

Suppose that a universally quantified inductive invariant I exists.
Then:



I satisfies initiation: $\sigma_0 \vDash \text{Init} \Rightarrow \sigma_0 \vDash \mathrm{I}$

I satisfies consecution: $\sigma_{i-1} \vDash \mathrm{I} \wedge \mathrm{TR}(\sigma_{i-1}, \sigma_i') \Rightarrow \sigma_i' \vDash \mathrm{I}$

I is universal: $\sigma_i' \vDash \text{Diag}(\sigma_i) \Rightarrow \sigma_i \vDash \mathrm{I}$

Contradiction to safety!

$\sigma_{i+1} \vDash \mathrm{I} \Rightarrow$

$\sigma_{i+1} \vDash \neg\text{Bad}$

If there is a universal inductive invariant $\mathrm{I} \in \forall^*$, then any **abstract trace** does not reach Bad

➔ An abstract trace to Bad implies no universal inductive invariant exists

Termination?

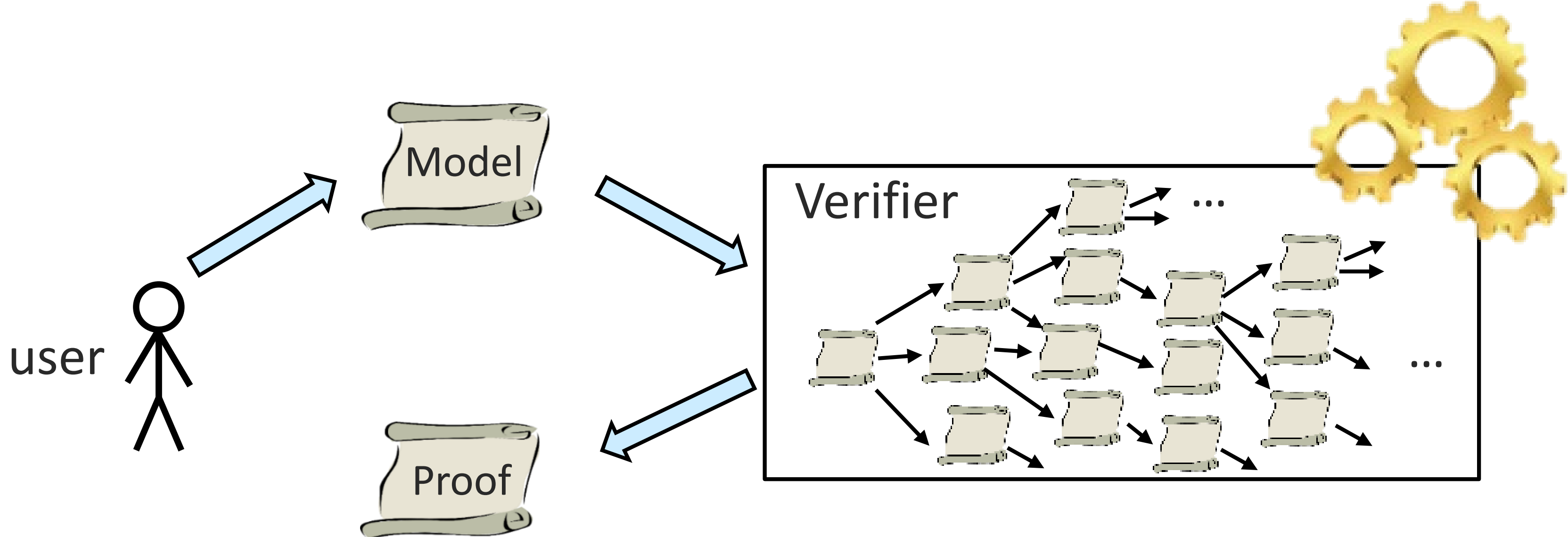
Termination?

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
- More decidable classes

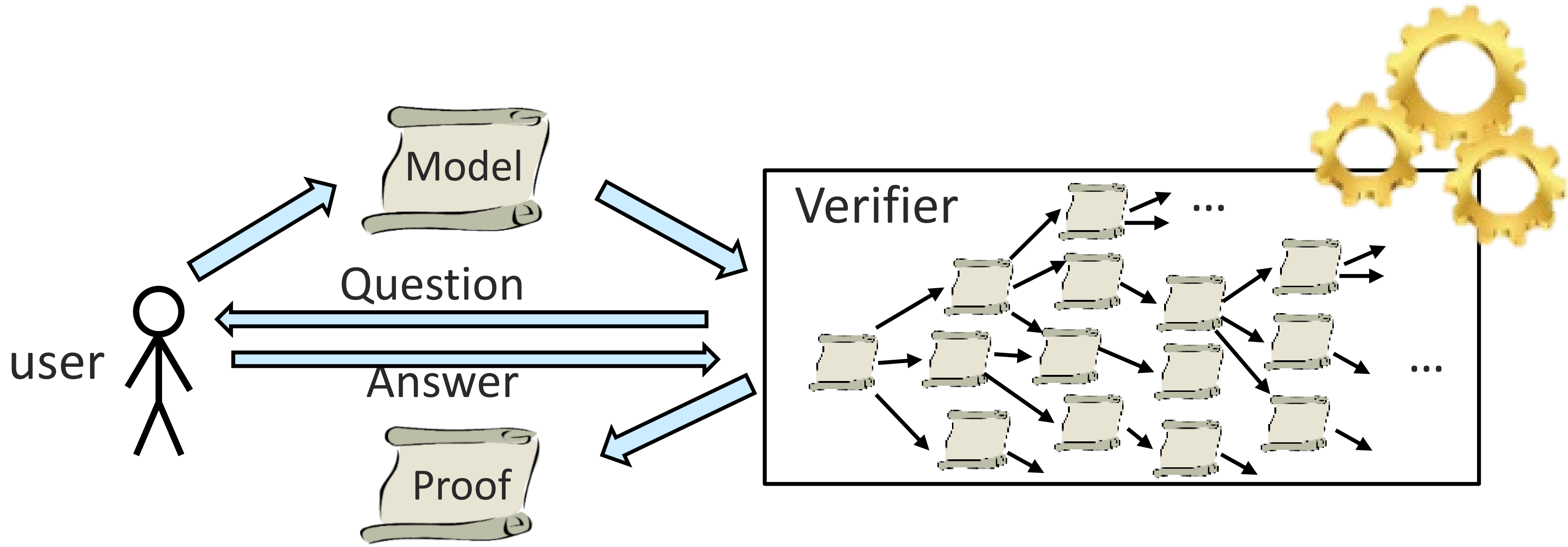
- [POPL'16] Decidability of Inferring Inductive Invariants, O. Padon, N. Immerman, S. Shoham, A. Karbyshev, and M. Sagiv.

Automatic Verification (e.g., UPDR)



Ultimately limited by undecidability

Interactive Verification



- Divide the problem between the human and the machine
- Find a suitable way to conduct the interaction

Supervised Verification of Infinite-State Systems

Interactive Verification

Proof Assistants

Ultimately limited by human

proof/code:
Verdi: ~10
IronFleet: ~4

Ultimately limited by undecidability

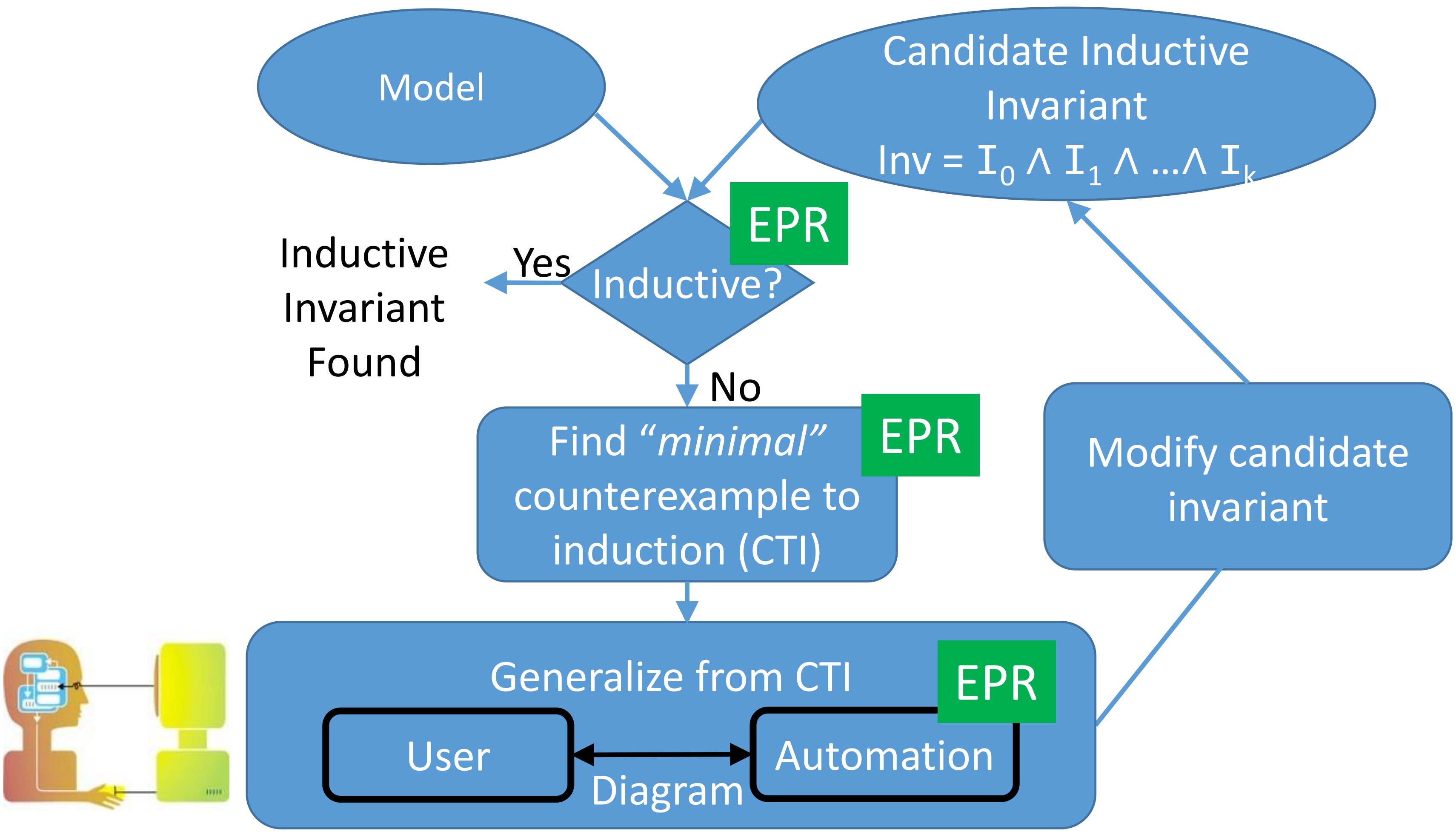
Model Checking
Static Analysis

Expressiveness

Automation

Supervised Verification of Infinite-State Systems

(2) Ivy: Interactive inference



[PLDI'16] IVy: Safety Verification by Interactive Generalization. O. Padon, K. McMillan, A. Panda, M. Sagiv, S. Shoham https://github.com/Microsoft/ivy

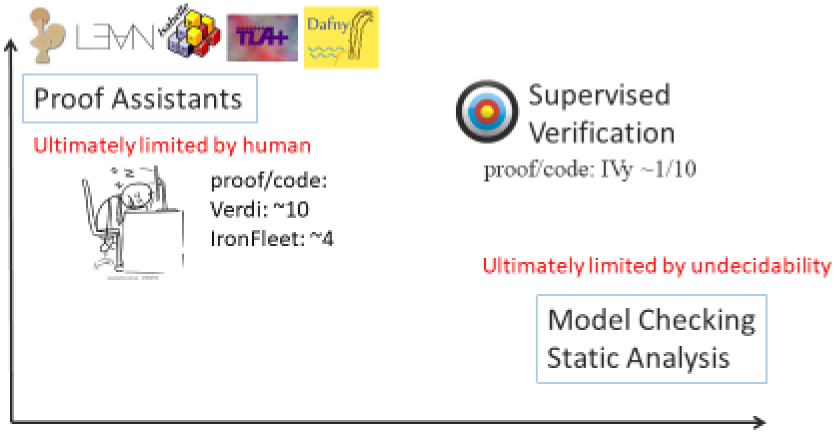
Summary 1

Verification with decidable logic

- Domain knowledge and axioms
- Derived relations
- Modularity
- Prophecy

- EPR - decidable fragment of FOL
  - Deduction is decidable
  - Finite counterexamples
- Can be made surprisingly powerful
  - Transitive closure: linked lists, ring topology [PLDI'16]
  - Paxos, Multi-Paxos, [OOPSLA'17]
  - Liveness and Temporal Properties [POPL'18]
  - Developing verified implementations [PLDI'18]

Summary 2



Invariant Inference

- Automatic inference: UPDR
- Interactive inference: Ivy

- Use logical diagram to infer Inv ∈ ∀*
- Can also prove absence of Inv ∈ ∀*

Decidable logic is useful!

- Other logics
- Theoretical understanding of limitations and tradeoffs
- Interactive verification
  - Dividing the problem between human and machine
  - Inference schemes
  - Forms of interaction

Seeking postdocs and students

erc Supervised Verification of Infinite-State Systems


צועדים עם רון בלב
GOOD VIBES