Combining Predicate Abstraction with Fixpoint Approximations

Tuba Yavuz

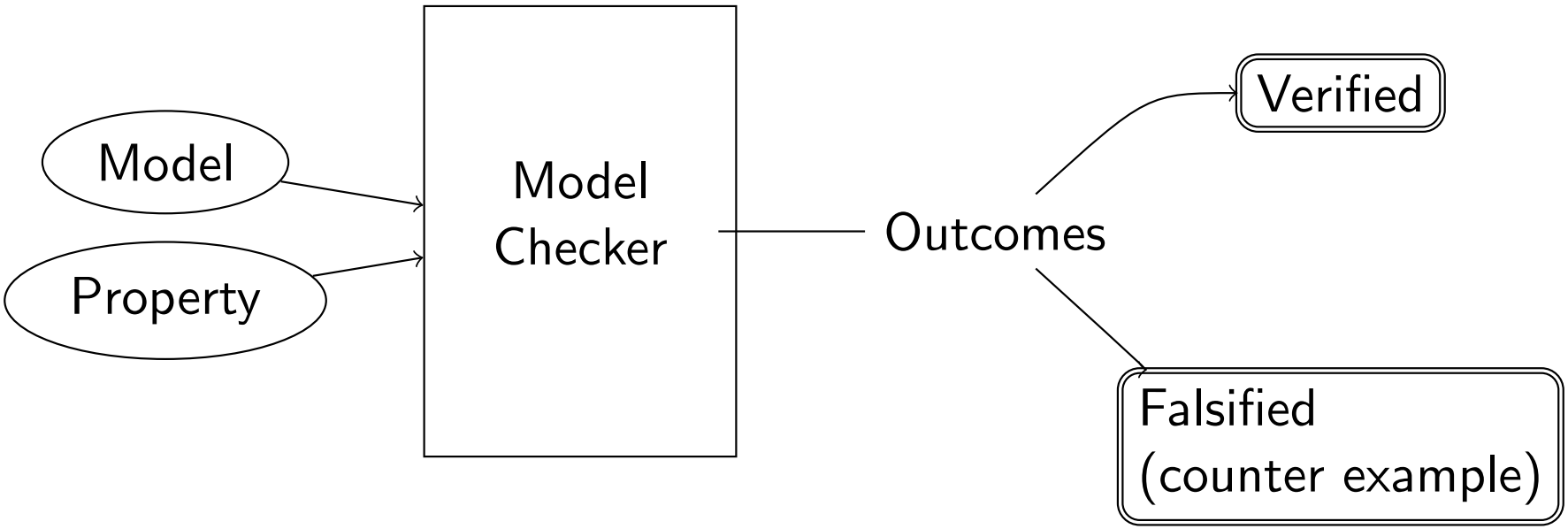
University of Florida

tuba@ece.ufl.edu

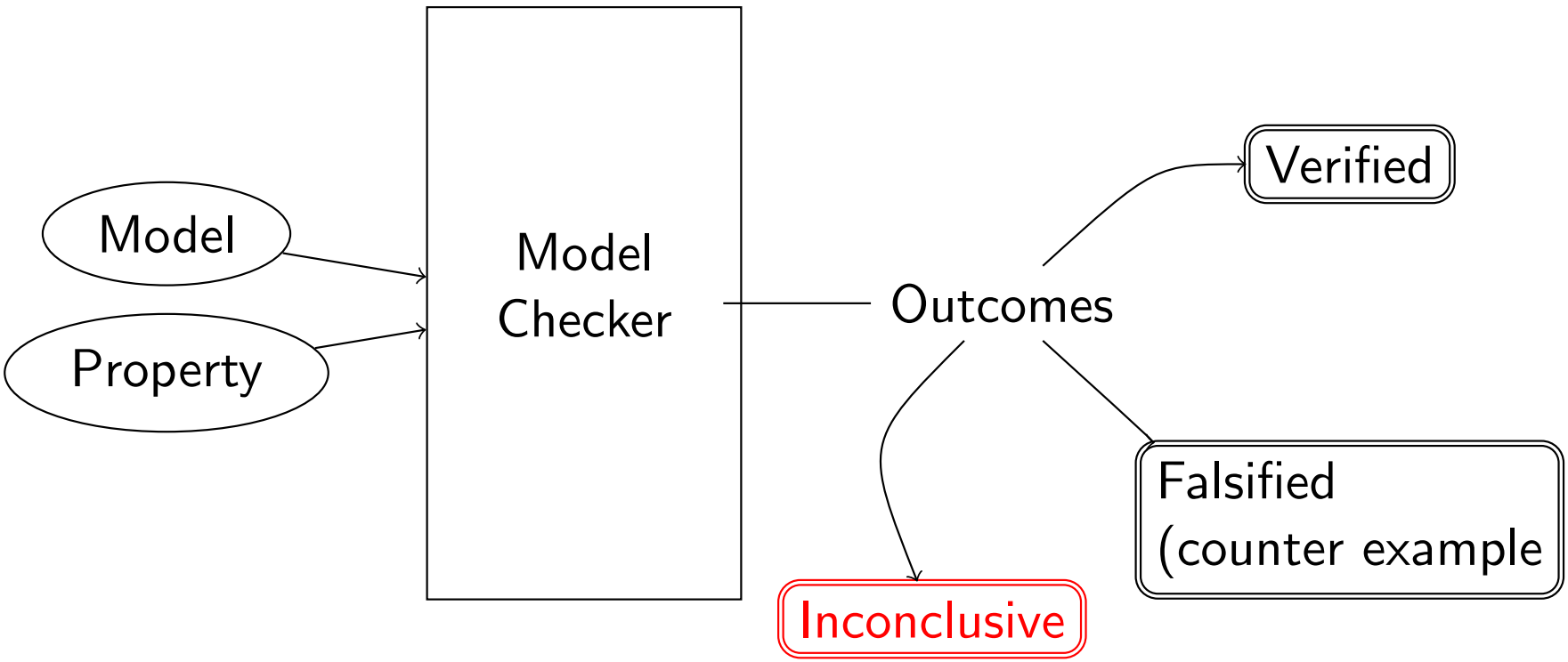
SEFM 2016, Vienna, Austria

Overview

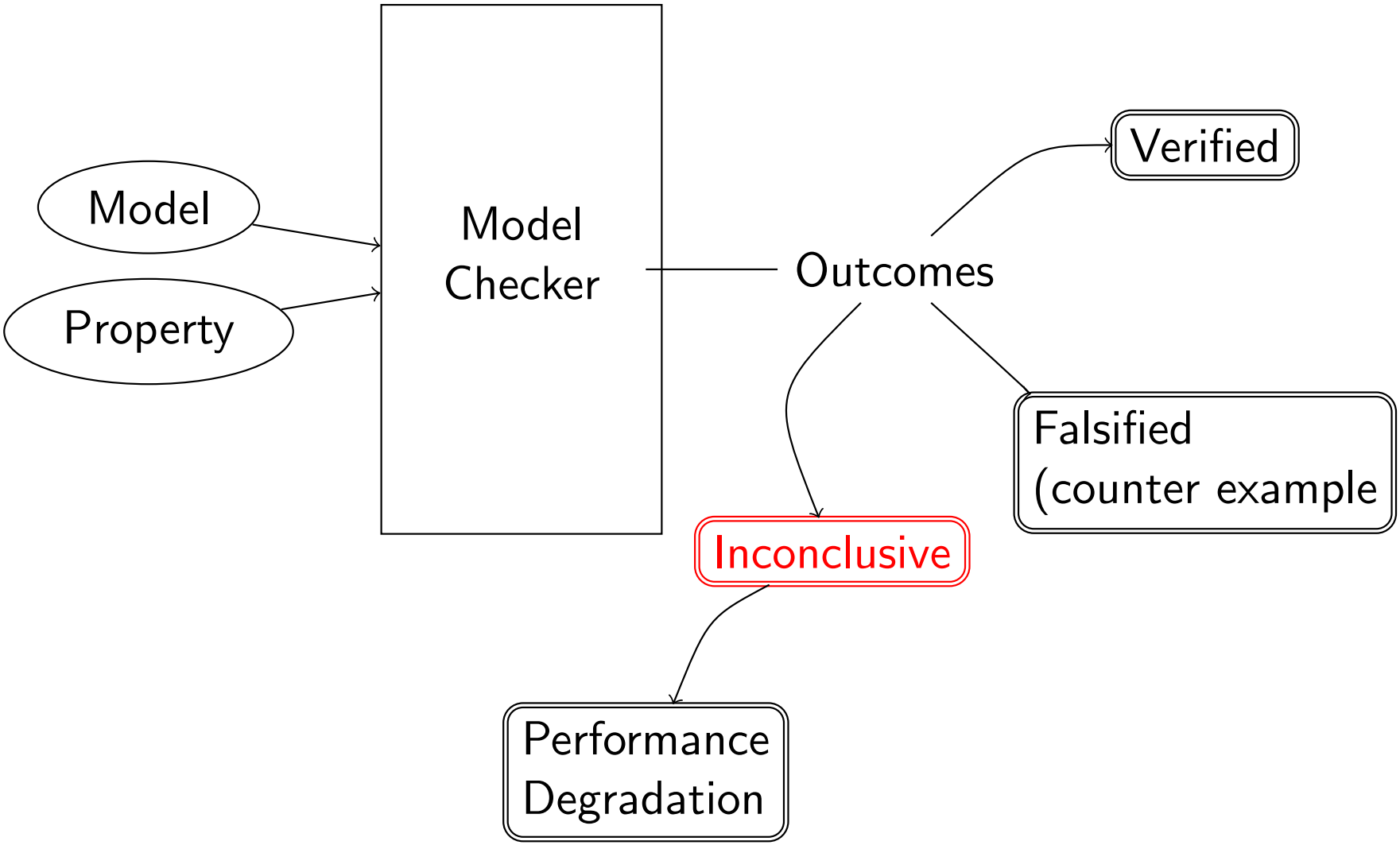
A simplified illustration on model checking.



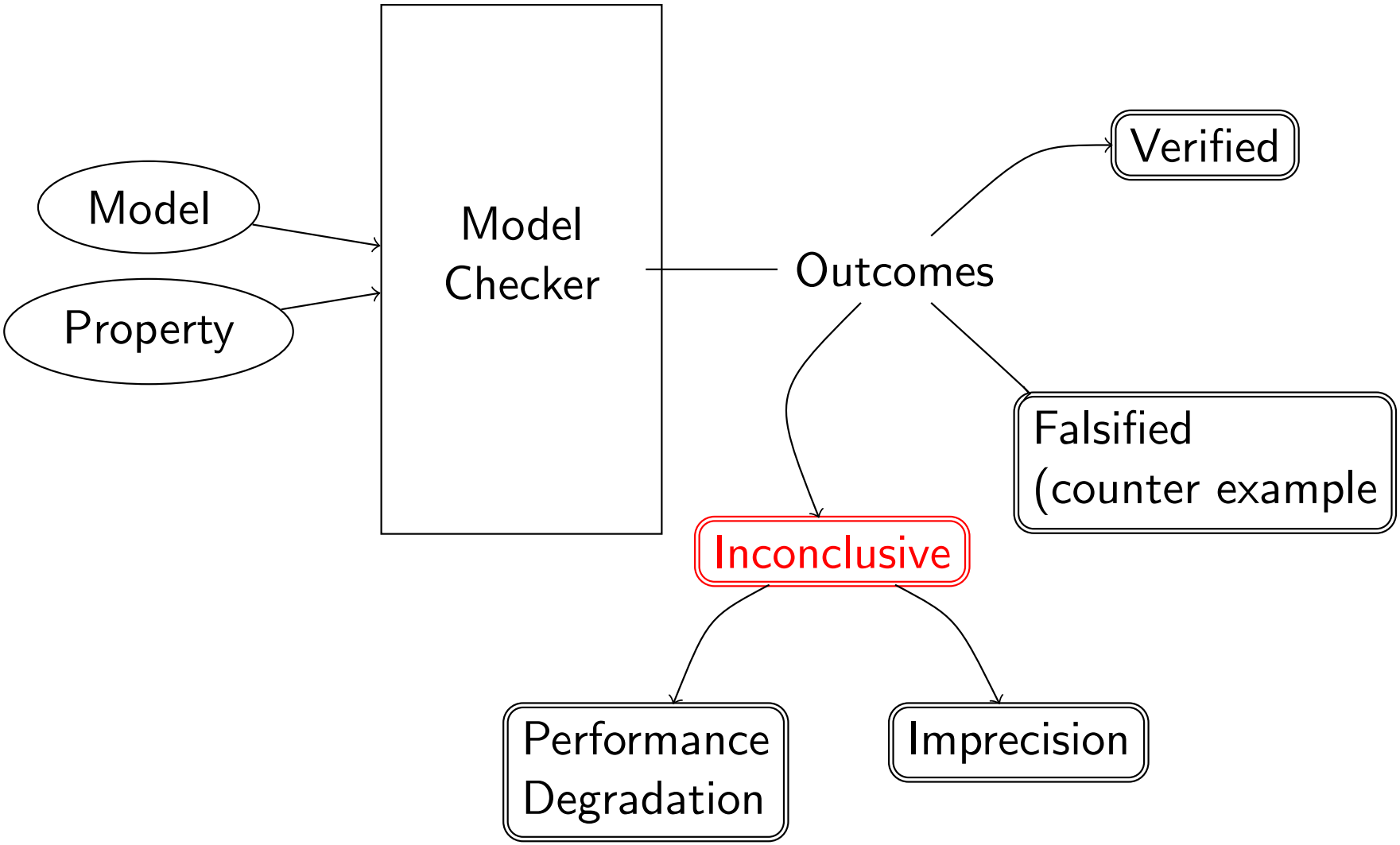
A more realistic illustration on model checking.



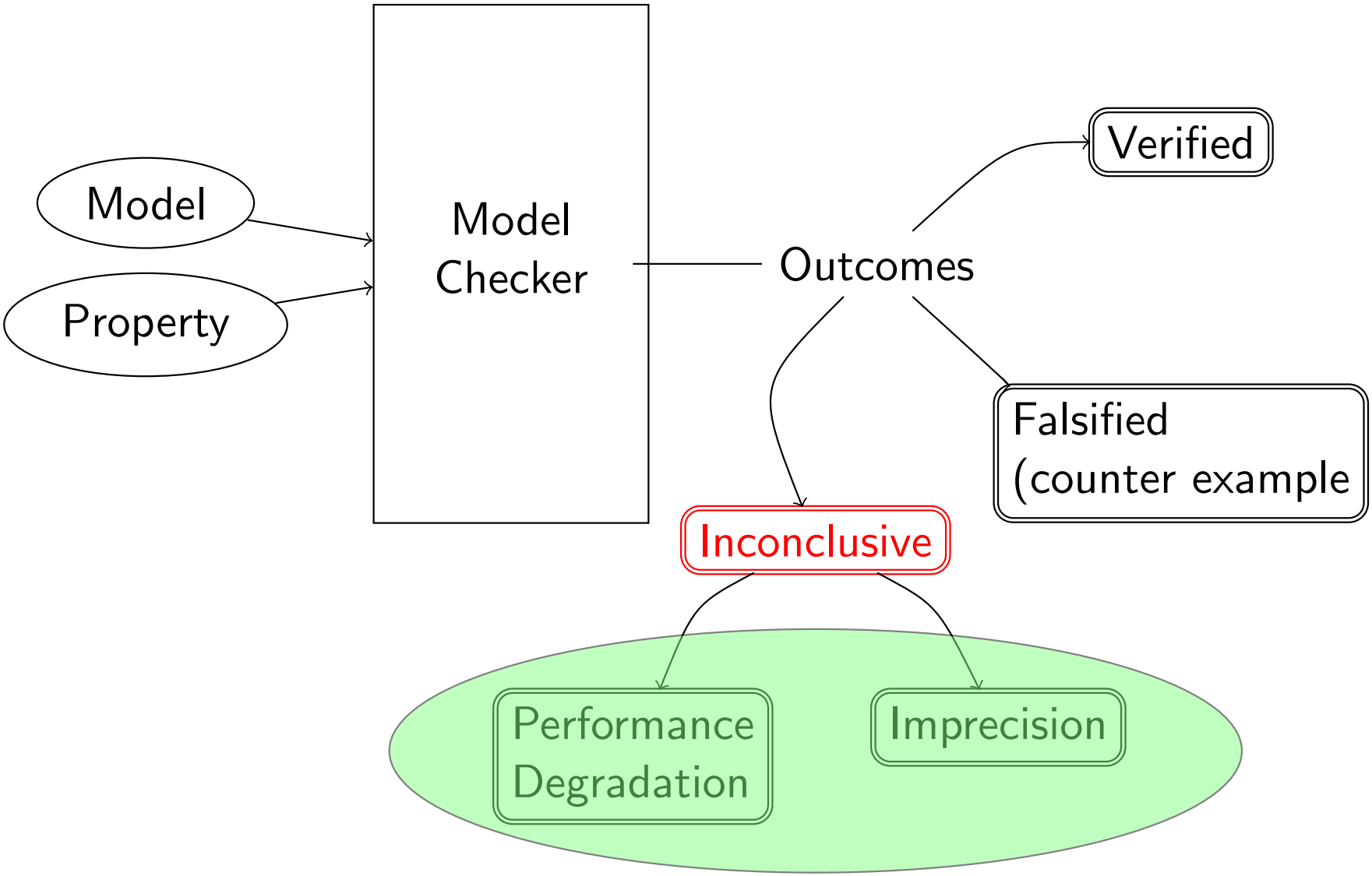
A more realistic illustration on model checking.



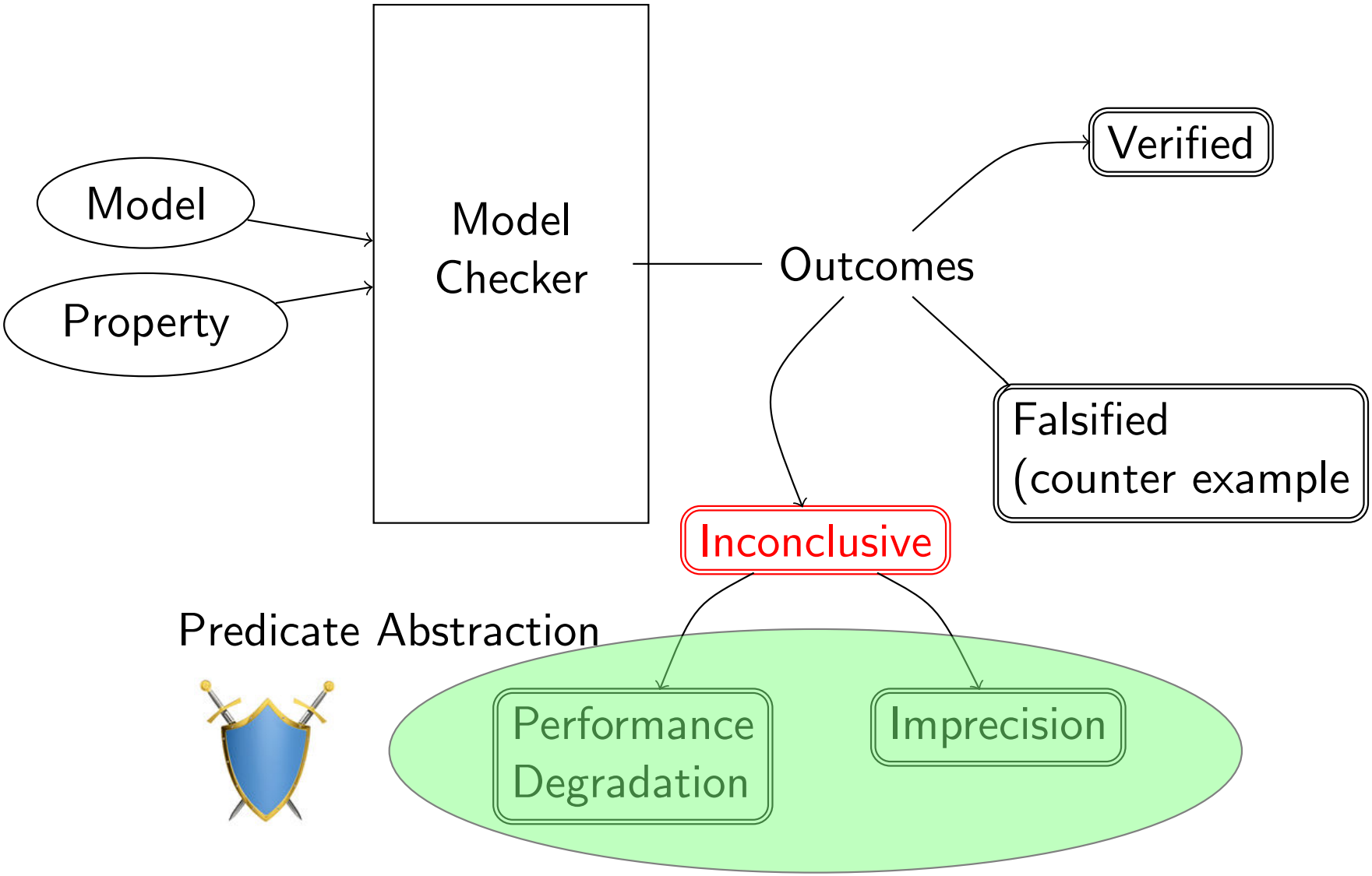
A more realistic illustration on model checking.



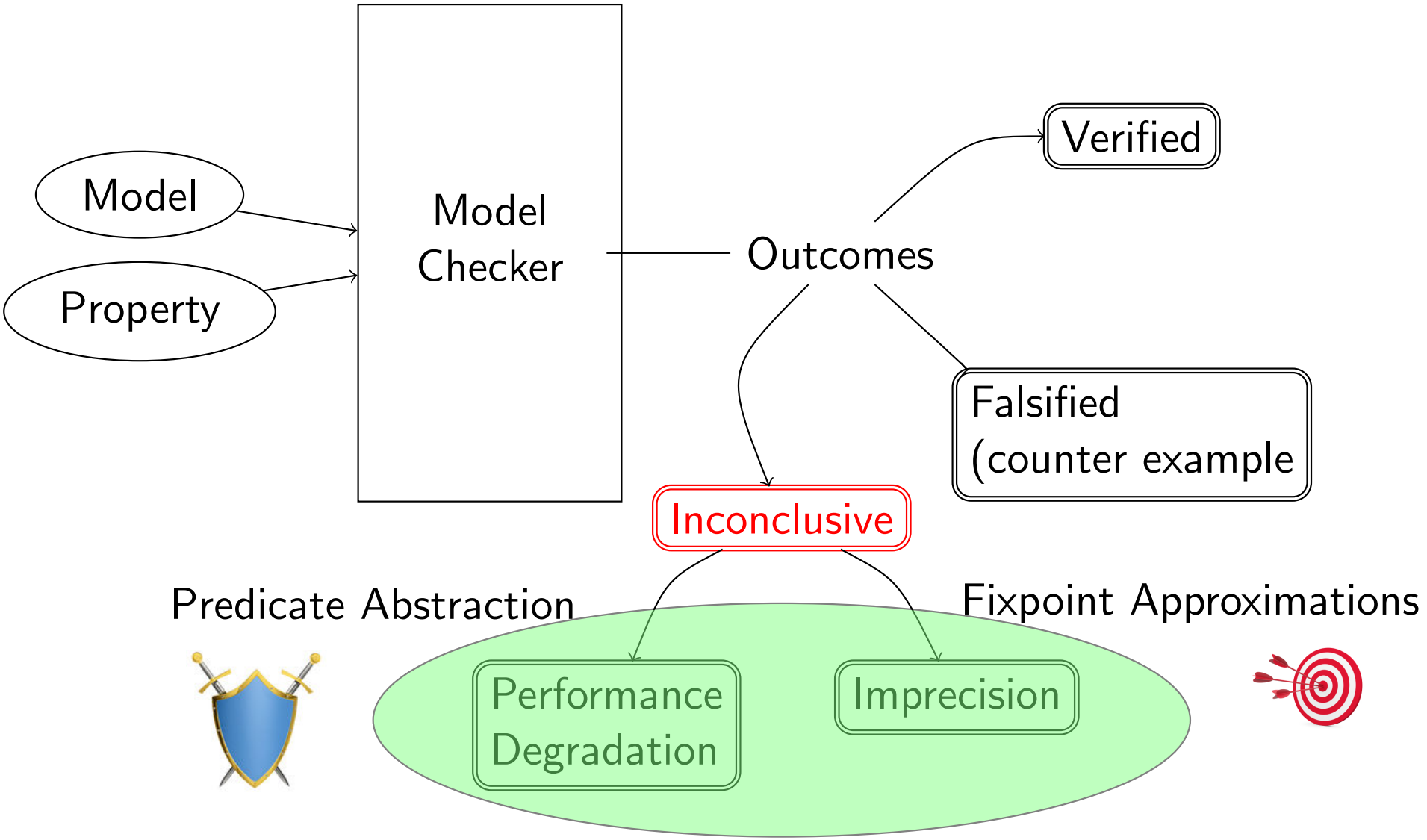
A more realistic illustration on model checking.



A more realistic illustration on model checking.

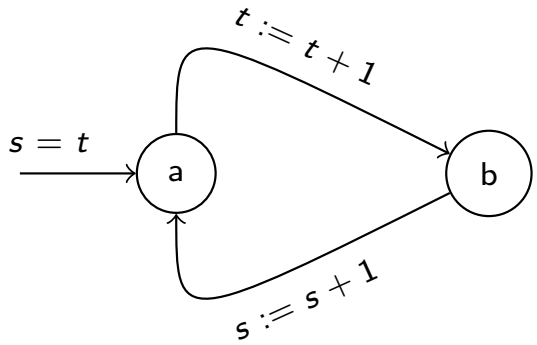


A more realistic illustration on model checking.



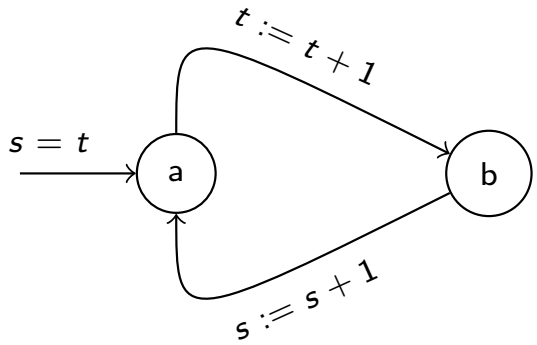
Overview

Predicate Abstraction

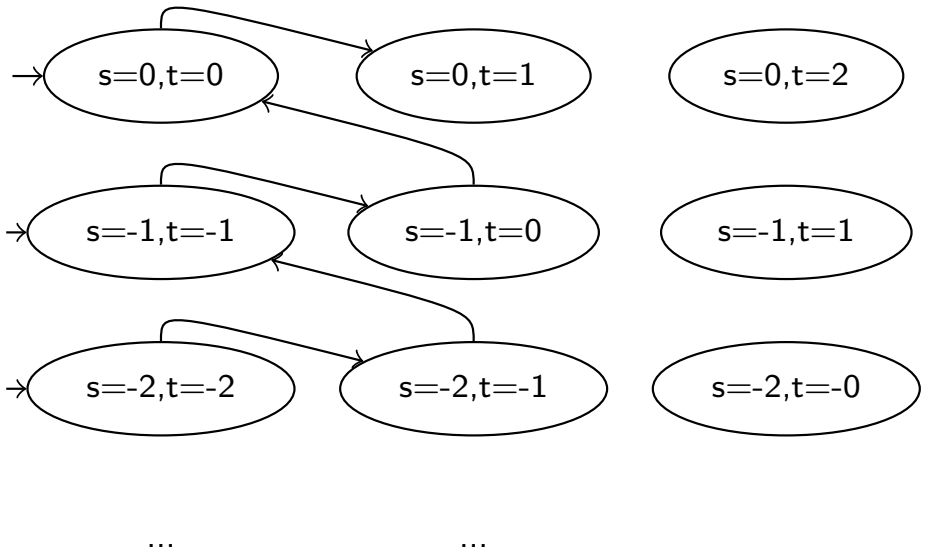


Predicate Set $= \{s = t\}$

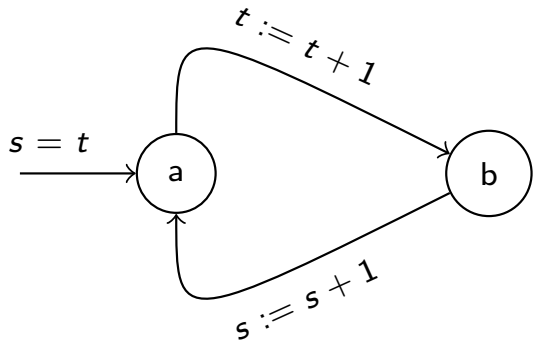
Predicate Abstraction



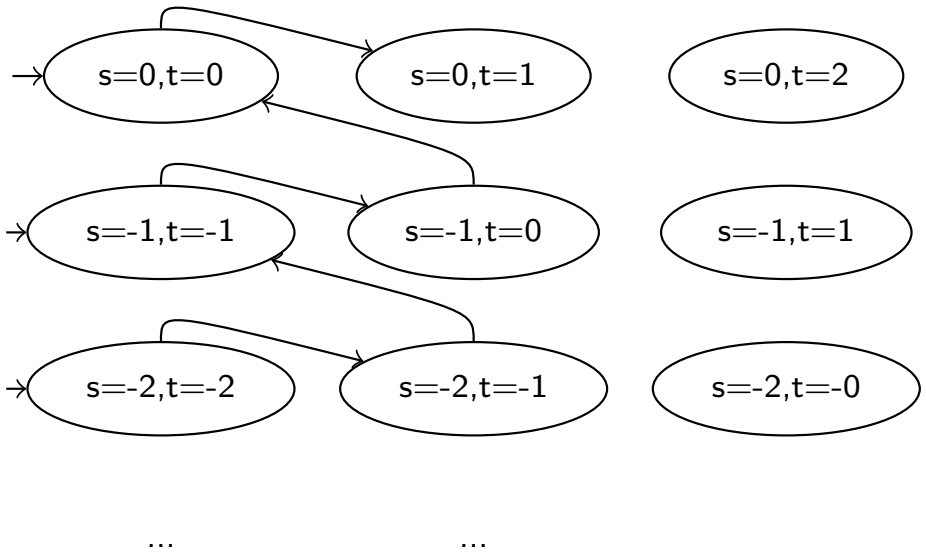
Predicate Set $= \{s = t\}$

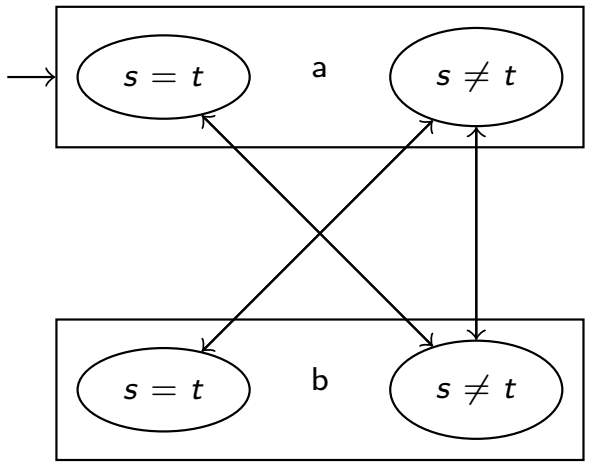


Predicate Abstraction

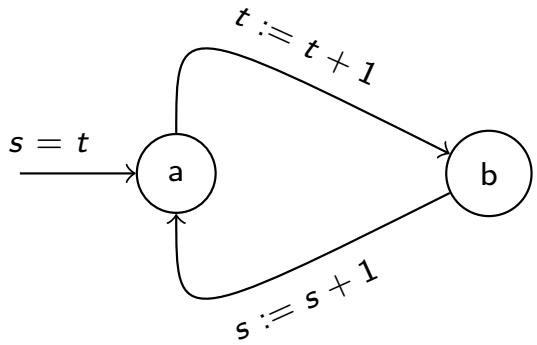


Predicate Set $= \{s = t\}$



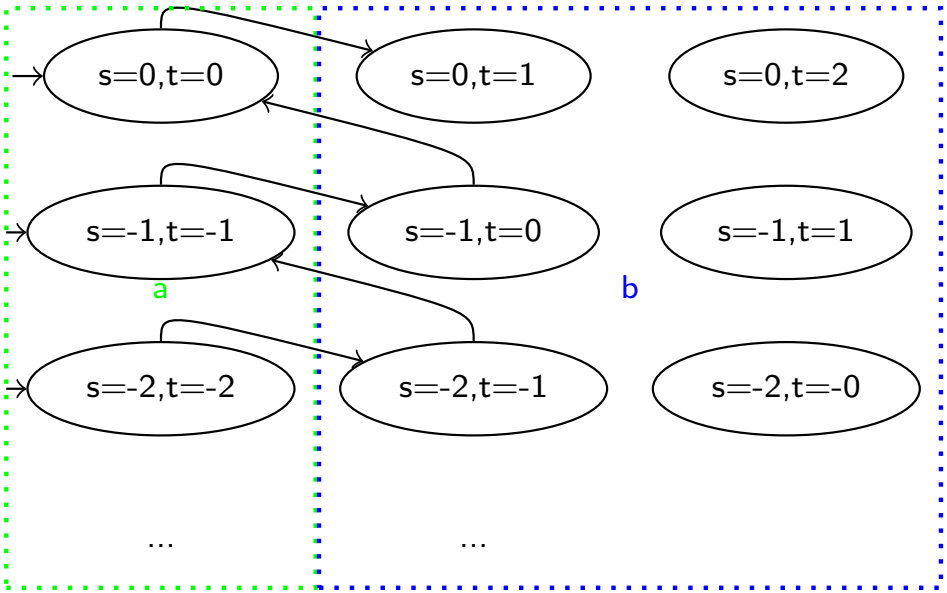


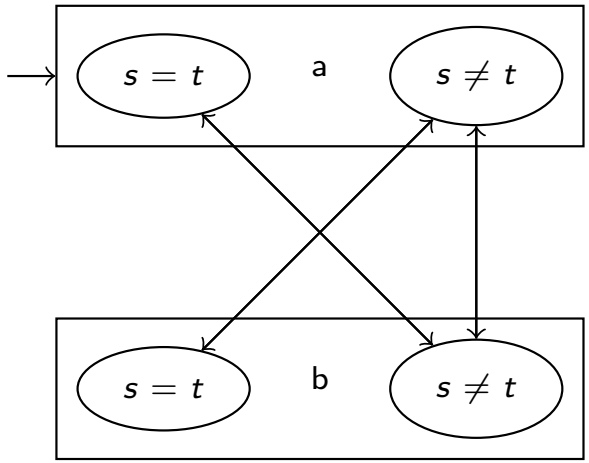
Predicate Abstraction



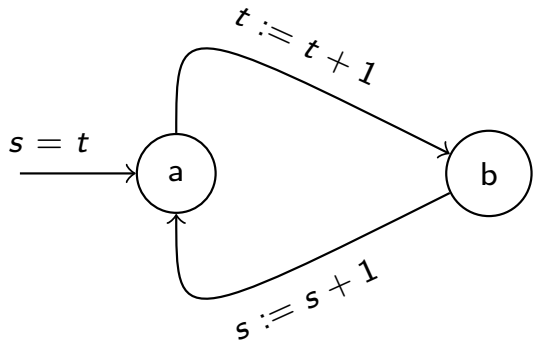
Predicate Set $= \{s = t\}$

$\gamma(s = t)$ $\gamma(s \neq t)$



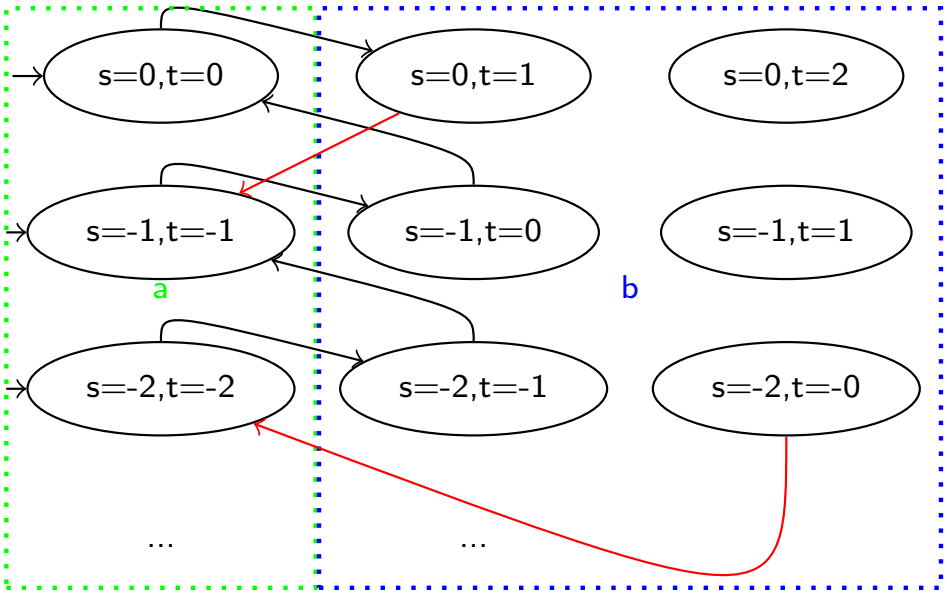


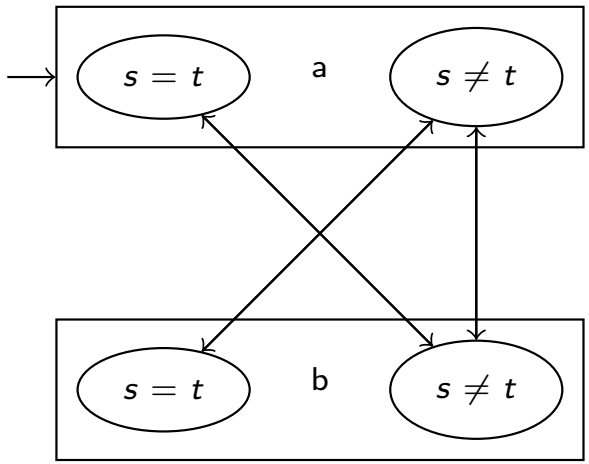
Predicate Abstraction



Predicate Set $= \{s = t\}$

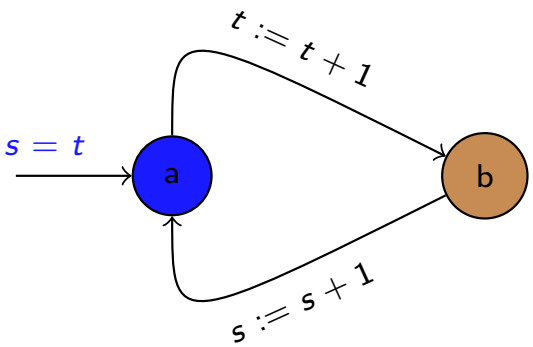
$\gamma(s = t)$ $\gamma(s \neq t)$





Overview

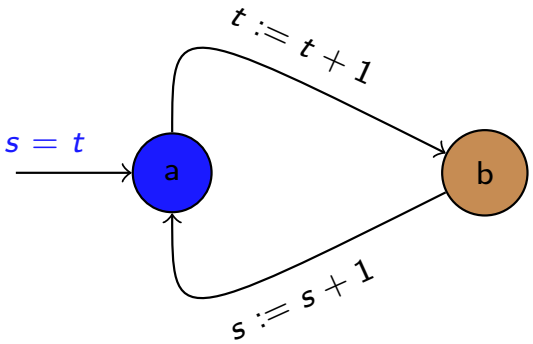
Fixpoint Approximation

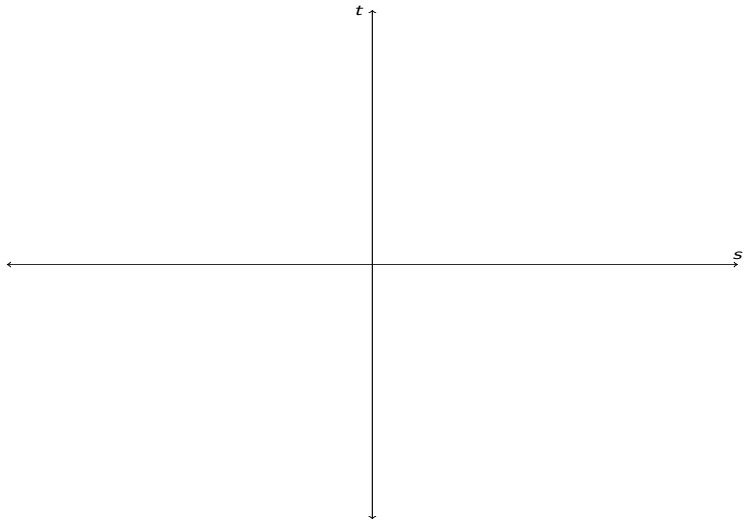


$f = AG(s > 0 \rightarrow t > 0)$?

$or\ \neg EF(s > 0 \wedge t \leq 0)$?

Fixpoint Approximation

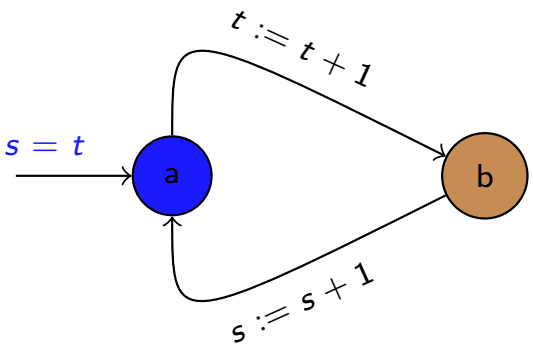


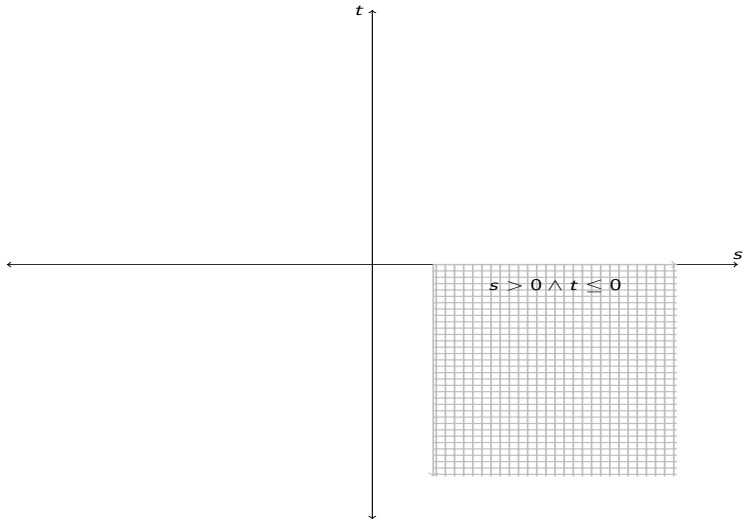


$f = AG(s > 0 \rightarrow t > 0)?$

$or\ \neg EF(s > 0 \wedge t \leq 0)?$

Fixpoint Approximation

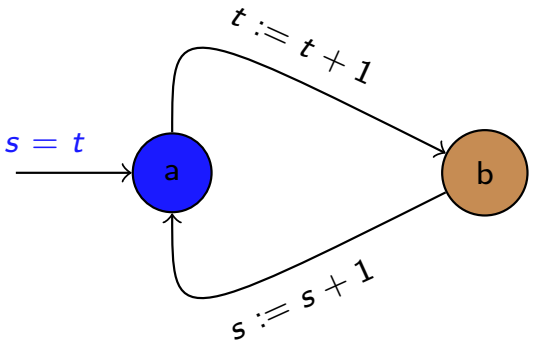


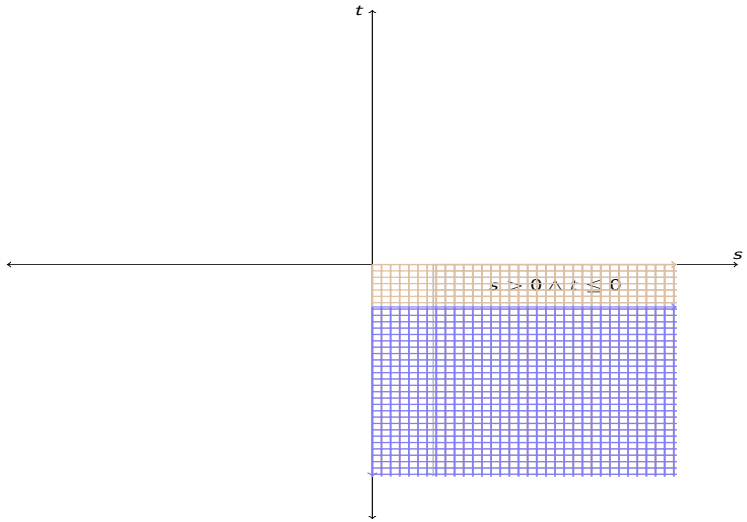


$f = AG(s > 0 \rightarrow t > 0)$?

$or\ \neg EF(s > 0 \wedge t \leq 0)$?

Fixpoint Approximation

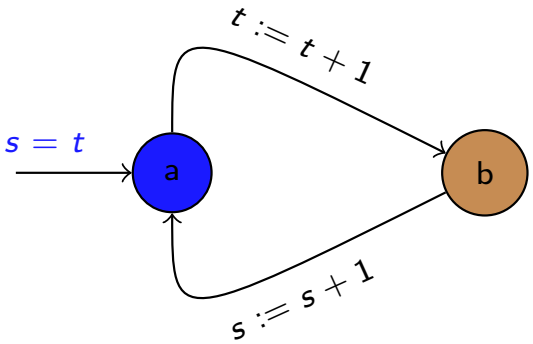


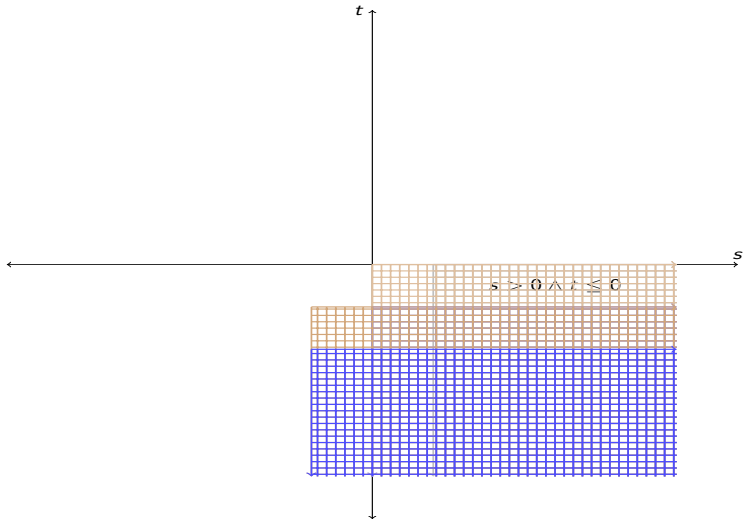


$f = AG(s > 0 \rightarrow t > 0)$?

$or\ \neg EF(s > 0 \wedge t \leq 0)$?

Fixpoint Approximation

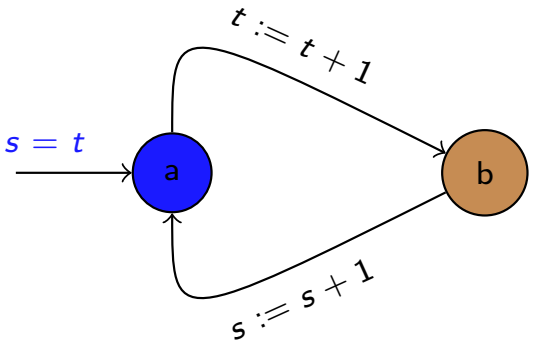


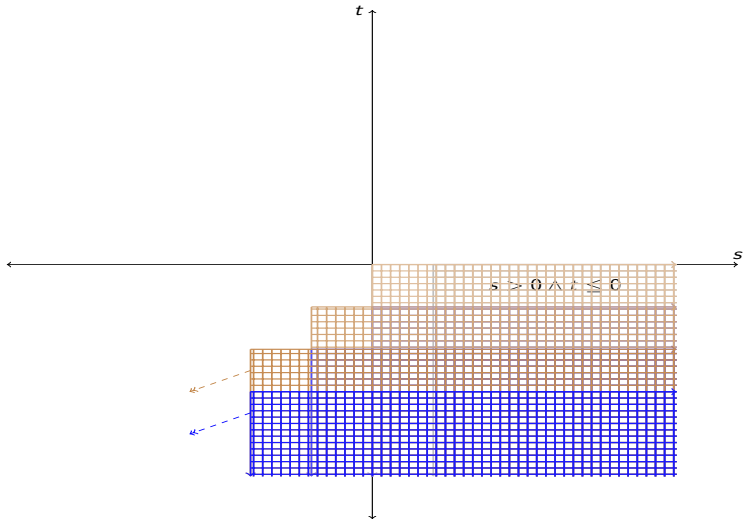


$f = AG(s > 0 \rightarrow t > 0)$?

$or\ \neg EF(s > 0 \wedge t \leq 0)$?

Fixpoint Approximation

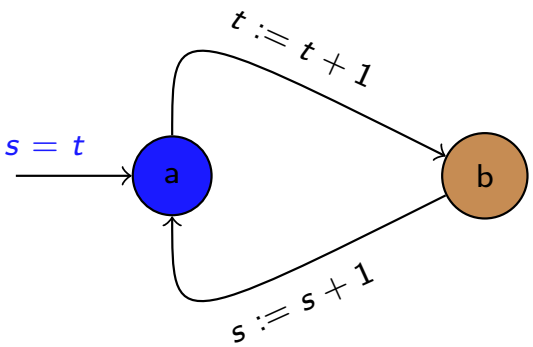


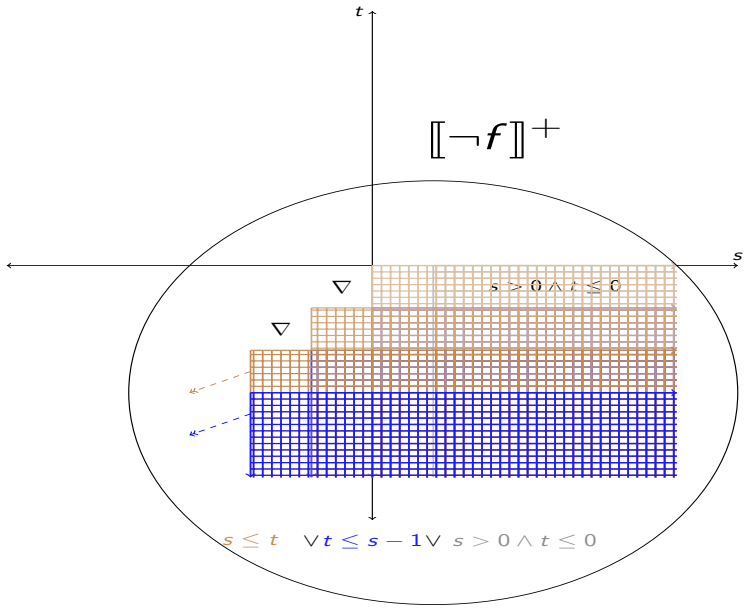


$f = AG(s > 0 \rightarrow t > 0)?$

$or\ \neg EF(s > 0 \wedge t \leq 0)?$

Fixpoint Approximation

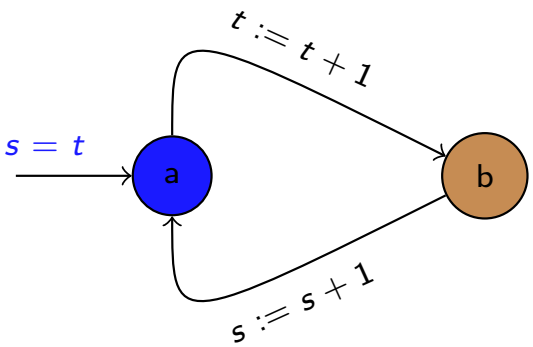


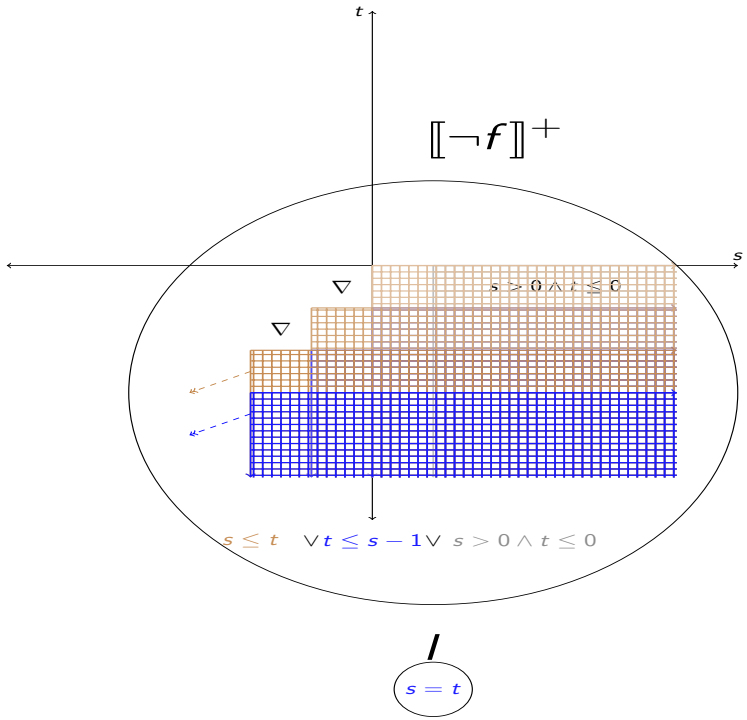


$f = AG(s > 0 \rightarrow t > 0)$?

$or\ \neg EF(s > 0 \wedge t \leq 0)$?

Fixpoint Approximation





$f = AG(s > 0 \rightarrow t > 0)$?

or $\neg EF(s > 0 \wedge t \leq 0)$?

Predicate Abstraction vs Fixpoint Approximation

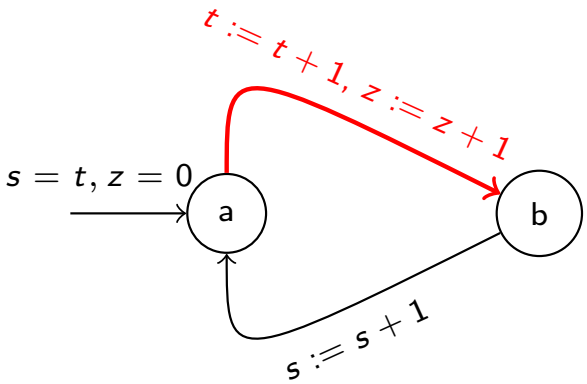
	Predicate Abstraction	**Fixpoint Approximation** *(no abstraction)*
Properties	verified ACTL only	full CTL
Counter-examples	may be spurious	real
Cost	construction time & memory (exponential in the number of predicates)	verification time (e.g., quantifier elimination for Presburger arithmetic $\mathcal{O}(m^{2^n})$, m: # of constraints, n: # of variables)

Overview

Proposed Approach: Partial Predicate Abstraction

Given an infinite-state system T^C defined over a set of boolean variables, V_{bool}, and integer variables, V_{int}, partition the integer variables into two sets, i.e., $V_{int} = V_{conc} \cup V_{abst}$ via a set of predicates φ defined over V_{abst} to obtain a *partially predicate abstracted* system T^{C+A} that is defined over the boolean variables, $V_{bool} \cup V_{\varphi}$, and integer variables, V_{conc}, where each $b_i \in V_{\varphi}$ represents a predicate φ_i.

Partial Predicate Abstraction - An Example



Predicate Set $= \{z > 0\}$

T^C

$V_{bool} = \{pc\}, V_{int} = \{s, t, z\}$

T^{C+A}

$V_{bool} = \{pc, b_{z>0}\}, V_{int} = \{s, t\}$

$$\tau \equiv pc = true \wedge t' = t + 1 \wedge s' = s \wedge z' = z + 1 \wedge pc' = false \xrightarrow{\alpha^\tau} pc = true \wedge t' = t + 1 \wedge s' = s \wedge pc' = false \wedge ((b_{z>0} = true \wedge b'_{z>0} = true) \vee (b_{z>0} = false \wedge b'_{z>0} = *))$$

Abstracting States and Transitions

Abstracting a state s:

$$\alpha(s) = \exists V(\varphi).s \wedge \bigwedge_{i=1}^{|\varphi|} \varphi_i \iff b_i$$

Abstracting a transition τ:

$$\alpha^{\tau}(\tau) = \exists V(\varphi).\exists V(\phi)'.\tau \wedge CS \wedge \bigwedge_{i=1}^{|\varphi|} \varphi_i \iff b_i \wedge \bigwedge_{i=1}^{|\varphi|} \varphi_i' \iff b_i'$$

where CS is the consistency constraint that precisely maps states in which predicate variables do not change:

$$CS = \bigwedge_{\varphi_i \in \varphi} ((\bigwedge_{v \in V(\varphi_i)} v' = v) \implies b_i' \iff b_i)$$

Partially Predicate Abstracted Transition System

Given a concrete infinite-state transition system $T^C = (S^C, I^C, R^C, V^C)$ and a set of predicates φ, where $V(\varphi) \subseteq V^C_{int}$, the partially predicate abstracted transition system $T^{C+A} = (S^{C+A}, I^{C+A}, R^{C+A}, V^{C+A})$ is defined as follows:

- $S^{C+A} \subseteq \mathcal{B}^{|V^C_{bool}|+|\varphi|} \times \mathcal{Z}^{|V^C_{int} \setminus V(\varphi)|}$
- $S^{C+A} = \bigcup_{s^C \in S^C} \alpha(s^C)$.
- $I^{C+A} = \bigcup_{is^C \in I^C} \alpha(is^C)$.
- $R^{C+A} = \bigcup_{r^C \in R^C} \alpha^\tau(r^C)$.

Overview

Does T^{C+A} over-approximate T^C?

We need to show:

1. **Initial states are over-approximated:**
   $\exists s_C.(\alpha(s_C) = s_{C+A} \ \wedge \ s_C \in I^C)$ implies $s_{C+A} \in I^{C+A}$.

2. **Transitions are over-approximated:**
   $\exists s_C, s'_C.(\alpha(s_C) = s_{C+A} \ \wedge \ \alpha(s'_C) = s'_{C+A} \ \wedge \ (s_C, s'_C) \in R^C)$ implies $(s_{C+A}, s'_{C+A}) \in R^{C+A}$.

Does T^{C+A} over-approximate T^{C}?

We need to show:

1. **Initial states are over-approximated:**
   $\exists s_C.(\alpha(s_C) = s_{C+A} \ \wedge \ s_C \in I^C)$ implies $s_{C+A} \in I^{C+A}$.
   Follows from the construction that each concrete initial state is mapped to some abstract state:

$$I^{C+A} = \bigcup_{is^C \in I^C} \alpha(is^C)$$

2. **Transitions are over-approximated:**
   $\exists s_C, s'_C.(\alpha(s_C) = s_{C+A} \ \wedge \ \alpha(s'_C) = s'_{C+A} \ \wedge \ (s_C, s'_C) \in R^C)$ implies $(s_{C+A}, s'_{C+A}) \in R^{C+A}$.

Safe approximation of transitions.

Transitions are over-approximated:
$\exists s_C, s'_C.(\alpha(s_C) = s_{C+A} \ \wedge \ \alpha(s'_C) = s'_{C+A} \ \wedge \ (s_C, s'_C) \in R^C)$ implies $(s_{C+A}, s'_{C+A}) \in R^{C+A}$.

Safe approximation of transitions.

Transitions are over-approximated:
$\exists s_C, s'_C.(\alpha(s_C) = s_{C+A} \ \wedge \ \alpha(s'_C) = s'_{C+A} \ \wedge \ (s_C, s'_C) \in R^C)$ implies $(s_{C+A}, s'_{C+A}) \in R^{C+A}$.
or :

$$post[\tau^C](\gamma(s^{C+A})) \ \subseteq \ \gamma(post[\alpha^\tau(\tau^C)](s^{C+A}))$$

Safe approximation of transitions.

Transitions are over-approximated:
$\exists s_C, s'_C.(\alpha(s_C) = s_{C+A} \ \wedge \ \alpha(s'_C) = s'_{C+A} \ \wedge \ (s_C, s'_C) \in R^C)$ implies $(s_{C+A}, s'_{C+A}) \in R^{C+A}$.
or :

$$post[\tau^C](\gamma(s^{C+A})) \ \subseteq \ \gamma(post[\alpha^\tau(\tau^C)](s^{C+A}))$$

Due to safe approximation of transitions, i.e., existence of a Galois connection $(\alpha^\tau, \gamma^\tau)$:

$$\tau_C \sqsubseteq (\exists V(\varphi).\exists V(\phi)'.\tau_C \wedge CS \wedge \bigwedge_{i=1}^{|\varphi|} \varphi_i \iff b_i \wedge \bigwedge_{i=1}^{|\varphi|} \varphi'_i \iff b_i)[\bar{\varphi}, \bar{\varphi'}/\bar{b}, \bar{b'}]$$

Fixpoint Approximations on Partially Predicate Abstracted System

Theorem

Given a partially predicate abstracted system T^{C+A} obtained from T^C and an ACTL property f^A, if T^{C+A} satisfies f^A then T^C satisfies $\gamma(f^A)$.

- γ is the concretization function that preserves the structure of the ACTL formula while replacing each atomic formula a with $a[\bar{\varphi}/\bar{b}]$.
- Restriction to ACTL follows from $T^C \sqsubseteq_\alpha T^{C+A}$.
- Fixpoint approximation in the partially predicate abstracted system may involve
  - Under-approximation of the correctness property, $[\![f^A]\!]^-_{C+A}$, or over-approximation of the negated property, $[\![\neg f^A]\!]^+_{C+A}$.
  - Over-approximation of the reachable states, i.e., $S^{C+A} = (\mu Z.I^{C+A} \vee Post[R^{C+A}](Z))^+$.

Overview

The Ticket Algorithm

- A mutual exclusion algorithm that requires each thread wishing to enter the critical section to obtain a unique ticket number, where ticket numbers increase by one.
- The thread with the smallest unused ticket number has the right to enter the critical section.
- A thread that leaves the critical section tries to enter it again.
- 2 global integer variables (s, t) and a local ticket number (a_i) for each thread i.
- Verification of *mutual exclusion* and *liveness* properties requires **fixpoint approximations**.

Fixpoint Approximations vs Predicate Abstraction

Model	ALV (Fix. Appr.)		NuXmv (CEGAR, k-induction)	
	Memory	**Time** (sec)	**Memory**	**Time** (sec)
ticket2	1.73M	0.06	79.43 M	2.01 (bound=24, verified)
ticket3	3.71 M	0.88	-	>2700 (bound >115, unable to prove)
ticket4	6.64 M	3.49	-	>2700 (bound >58, unable to prove)
ticket5	280.77 M	1170.36	-	>2700 (bound >24, unable to prove)

Table: Comparison of ALV's fixpoint approximation based approach with NuXmv's CEGAR loop using predicate abstraction and k-induction.

Overview

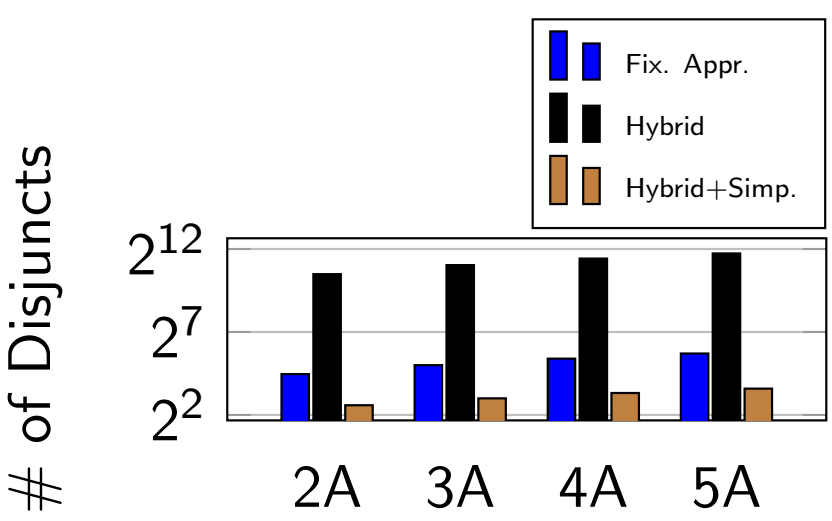
Airport Ground Traffic Control

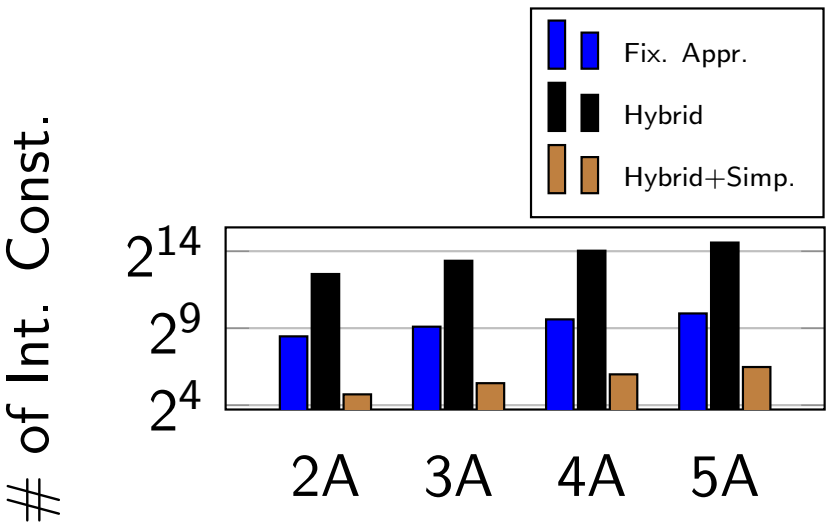
- A resource sharing model: airplanes access taxiways and runways.
- An arriving (departing) airplane goes through flying (parking), landing (taxiing), taxiing (taking off), and parking (flying) states.
- The layout of the ground network imposes constraints on the atomic actions, e.g., moving from one taxiway that intersects with a runway to another taxiway.
- Integer variables (rc_i) are used to count the number of airplanes using each resource.

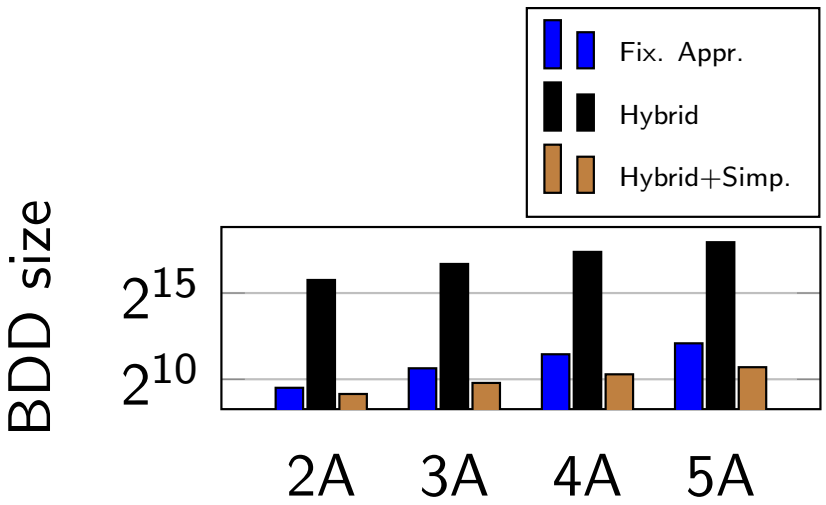
Airport Ground Traffic Control

- A resource sharing model: airplanes access taxiways and runways.
- An arriving (departing) airplane goes through flying (parking), landing (taxiing), taxiing (taking off), and parking (flying) states.
- The layout of the ground network imposes constraints on the atomic actions, e.g., moving from one taxiway that intersects with a runway to another taxiway.
- Integer variables (rc_i) are used to count the number of airplanes using each resource.
- Modified to
  - embed the ticket algorithm for mutual exclusion use of one of the taxiways.
  - faithfully represent continuous attempts for entering the critical section by having parked arriving airplanes go back to the flying mode.

Safety Verification - Transition System Sizes

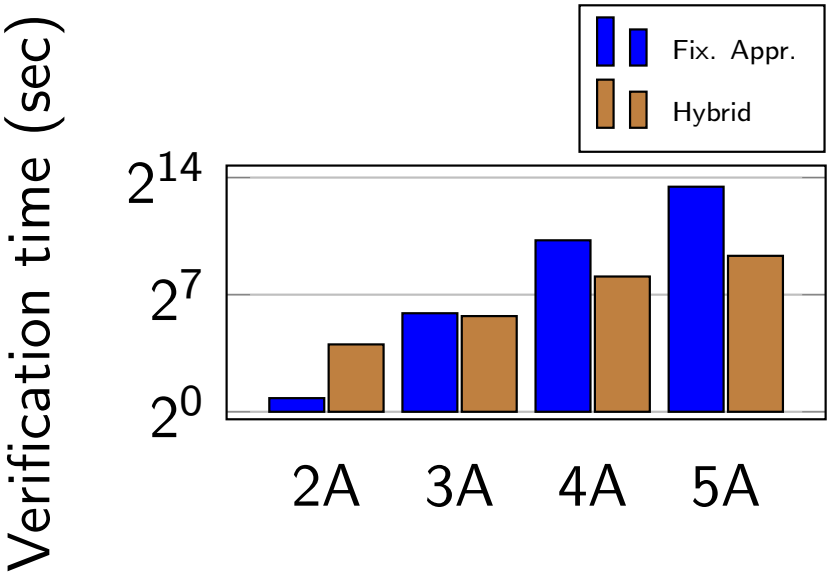


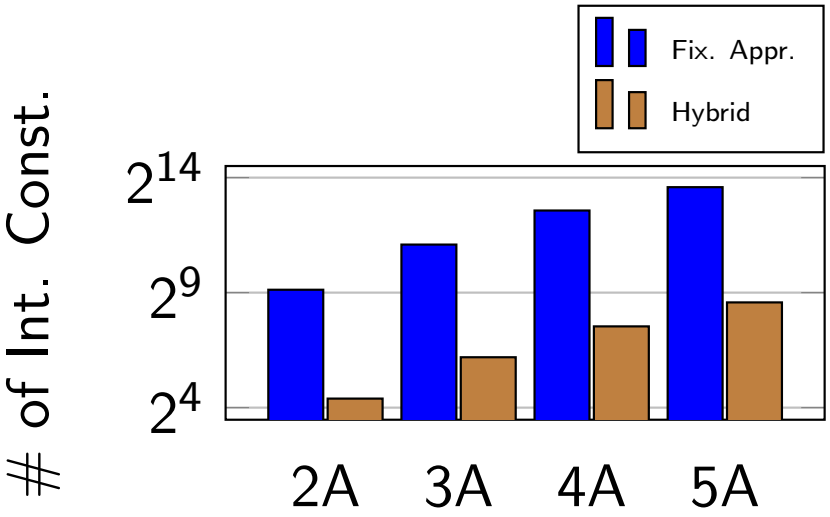


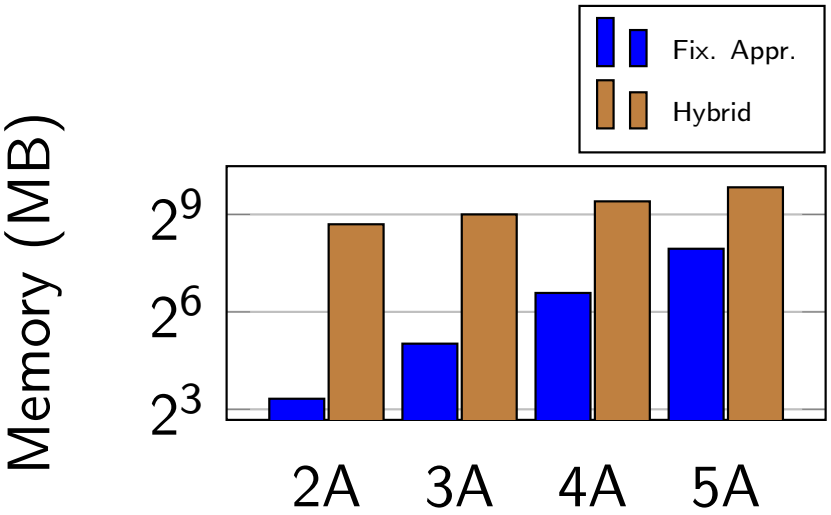


- Predicates: $rc_i < 1$
- # of predicates: 6
- # of abstracted variables: 6

Safety Verification - Performance

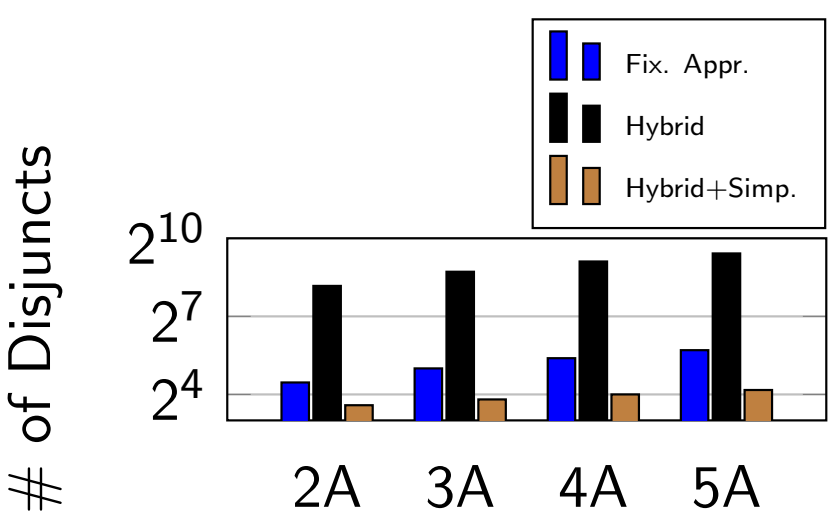


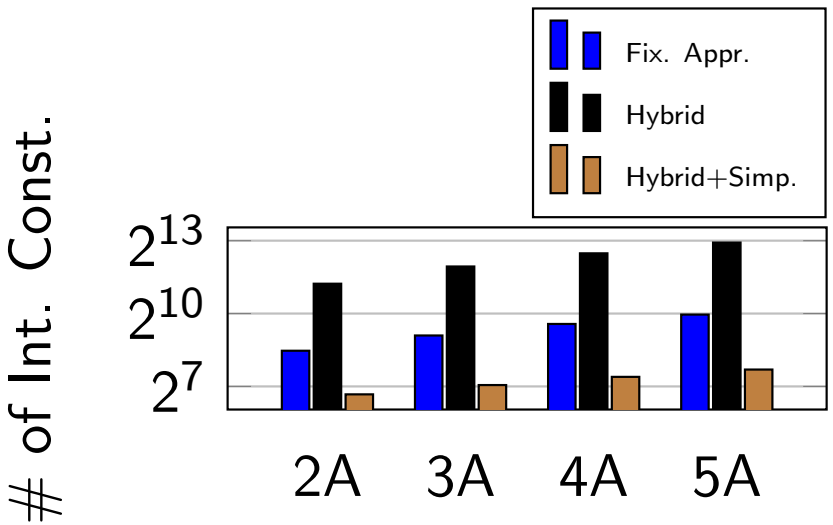


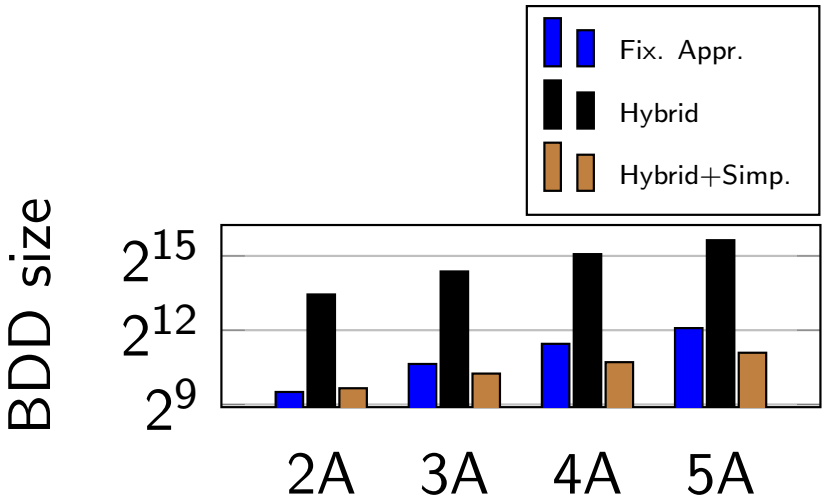


- Predicates: $rc_i < 1$
- # of predicates: 6
- # of abstracted variables: 6

Liveness Verification - Transition System Sizes

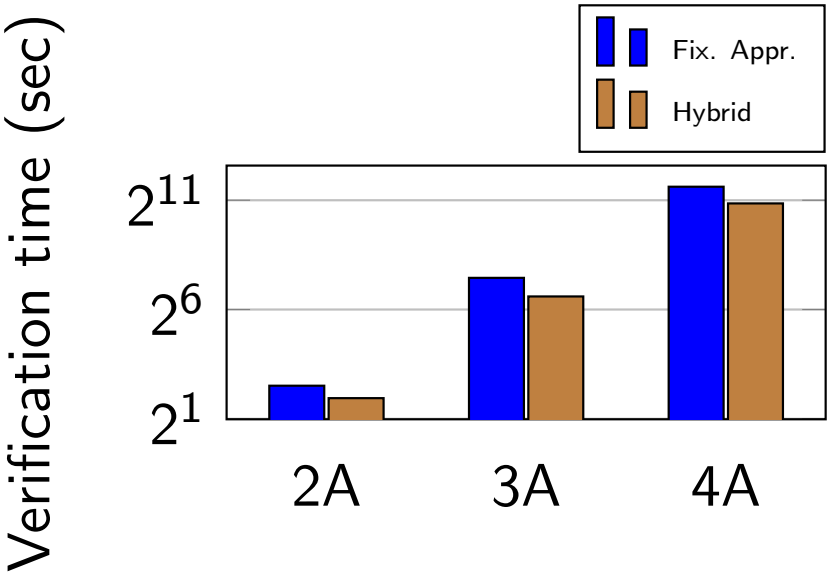


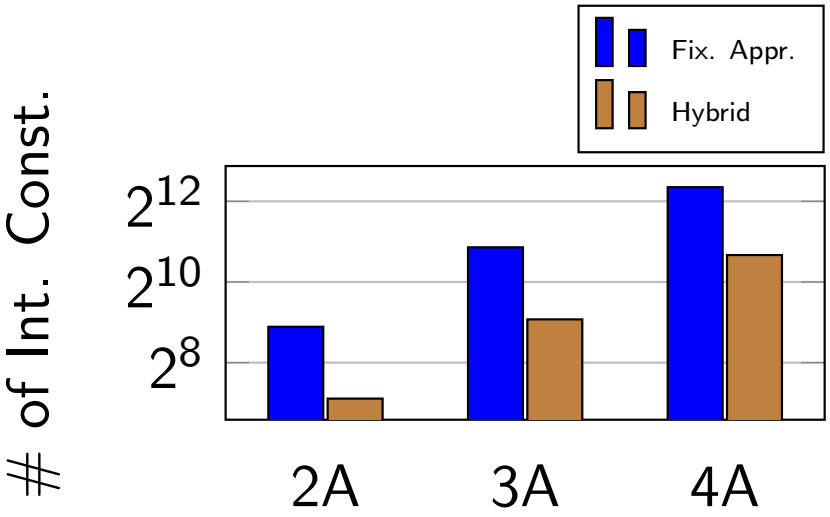


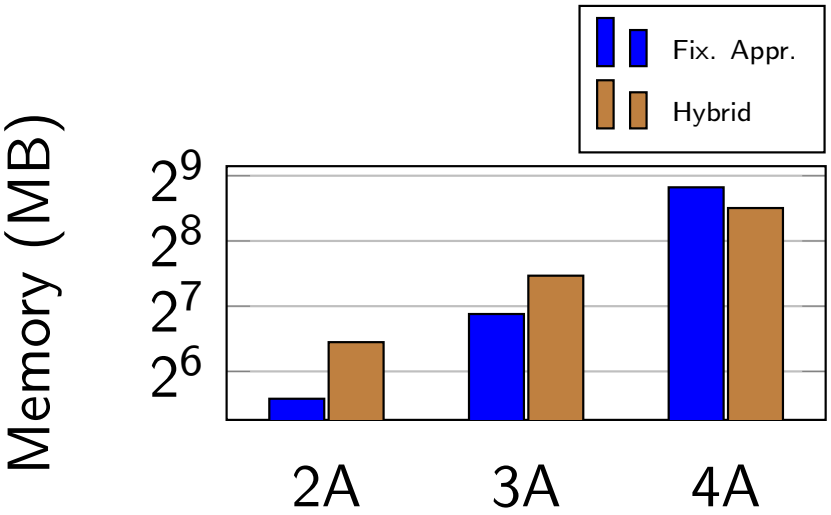


- Predicates:
  $rc_i = 0, rc_i = 1, rc_i < 1$
- # of predicates: 6
- # of abstracted variables: 2

Liveness Verification - Performance







- Predicates:
  $rc_i = 0, rc_i = 1, rc_i < 1$
- # of predicates: 6
- # of abstracted variables: 2

Overview

Related Work

- Gurfinkel & Chaki (STTT, 2010) combines predicate abstraction and numerical abstraction to improve precision. Its application in the context of software model checking shows similar improvements in performance over pure numerical representation.
- Jhala & McMillan (TACAS, 2006) deal with the inadequacy of CEGAR for generating integer-based predicates by proposing a technique that considers constants from a bounded range and works if the system satisfies the property and involves a bounded number of iterations. So it may not be effective for models like the ticket algorithm.
- Podelski & Rybalchenko (TOPLAS, 2007) present a transition-based predicate abstraction and use primed versions of the variables in the predicates. Our approach is not able to relate primed and unprimed variables but it can relate concrete and abstracted variables in a liveness property.

Overview

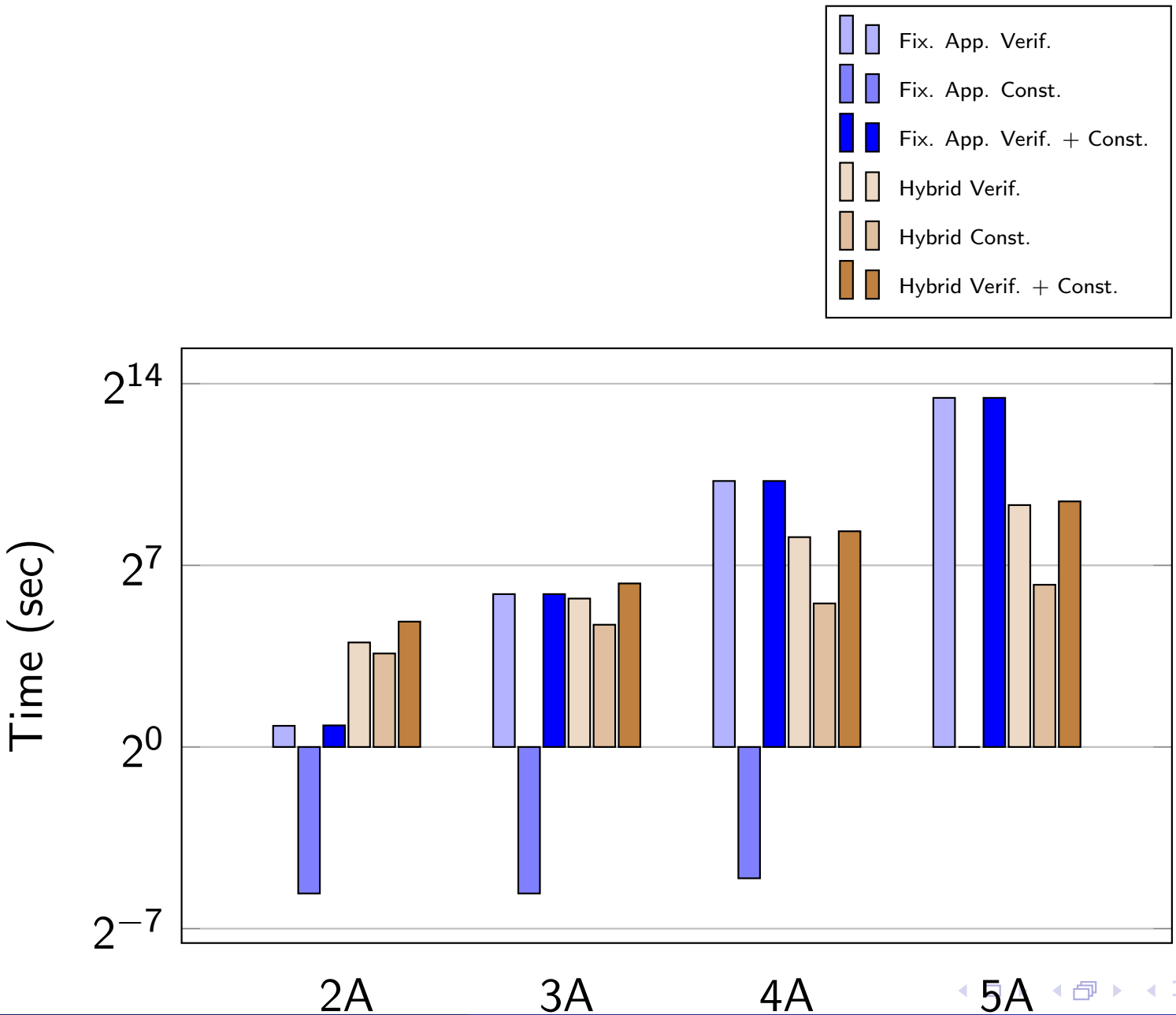
Conclusion & Future Work

- Combining predicate abstraction and fixpoint approximations is an effective technique for balancing the precision and performance.
- Empirical results show improvement in verification time for both safety and liveness verification. For liveness verification, memory usage is also improved.
- Future work will employ the hybrid approach in a counter-example guided abstraction and approximation refinement loop.

Thank you

Questions?

Safety Verification + Construction



Liveness Verification + Construction