Advanced model checking for verification and safety assessment

Alessandro Cimatti

Fondazione Bruno Kessler (FBK)

Invited Lectures on Advanced Verification

Part 1

Lectures prepared in collaboration with

Stefano Tonetta and Marco Gario

Slides on IC3 borrowed from Alberto Griggio (VTSA’15)

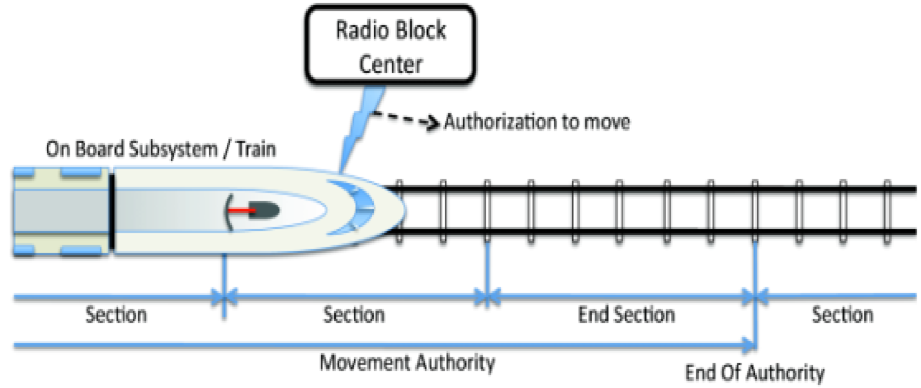
Outline

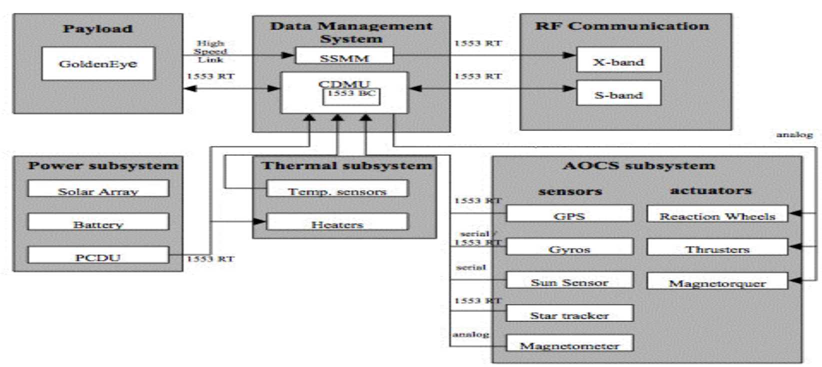
- Motivation
- Finite-State Model Checking
  - Invariant Checking
    - IC3
  - LTL Checking
- Infinite-State Model Checking
- Wrap-up

Motivation

Embedded Safety-Critical Systems

- Embedded with software to deliver intelligent:
  - Transportation
  - Communication
  - Automation
- Across domains:
  - Railways
  - Avionics
  - Automotive
  - Space
  - Health
- Key properties and challenges:
  - Interaction of components
  - Decomposition of services
  - Safety requirements





Model-based system engineering

- Models used for system requirements, architectural design, analysis, validation and verification
- Different system-level analysis (safety, reliability, performance, …)
- Formal methods as back-end
  - Formal specification to assign models a rigorous mathematical semantics
  - Formal verification to prove the properties on the models.
- Design models translated into input for verification engine
- Requirements formalized into properties
- Model checking appealing because integrated as push-button

AIR6110 Wheel Braking System

- Joint scientific study with Boeing
- Aerospace Information Report 6110:
  - Traditional Aircraft/System Development Process Example
- Wheel Brake System of a fictional dual-engine aircraft
- Objectives:
  - Analyze the system safety through formal techniques
  - Demonstrate the usefulness and suitability of formal techniques for improving the overall traditional development and supporting aircraft certification

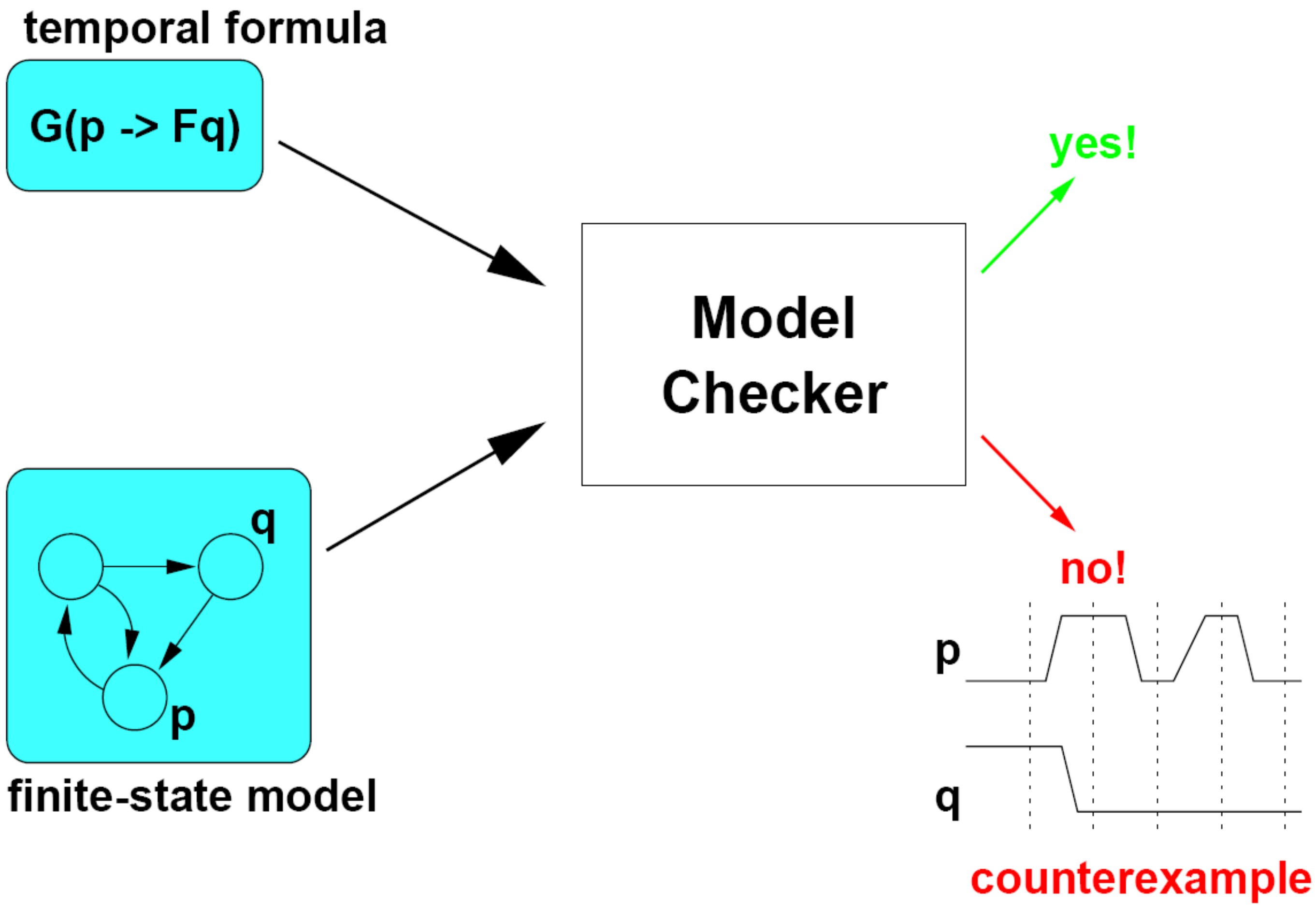
NASA NextGen Air Traffic Control

- Joint project with NASA Ames and Langley
- Allocation of tasks between Aircraft and Ground
  - Model and Study a design space with more than 1600 configurations
- Objectives:
  - Apply Formal Methods to study the quality and Safety of many design proposals
  - Highlight Implicit assumptions

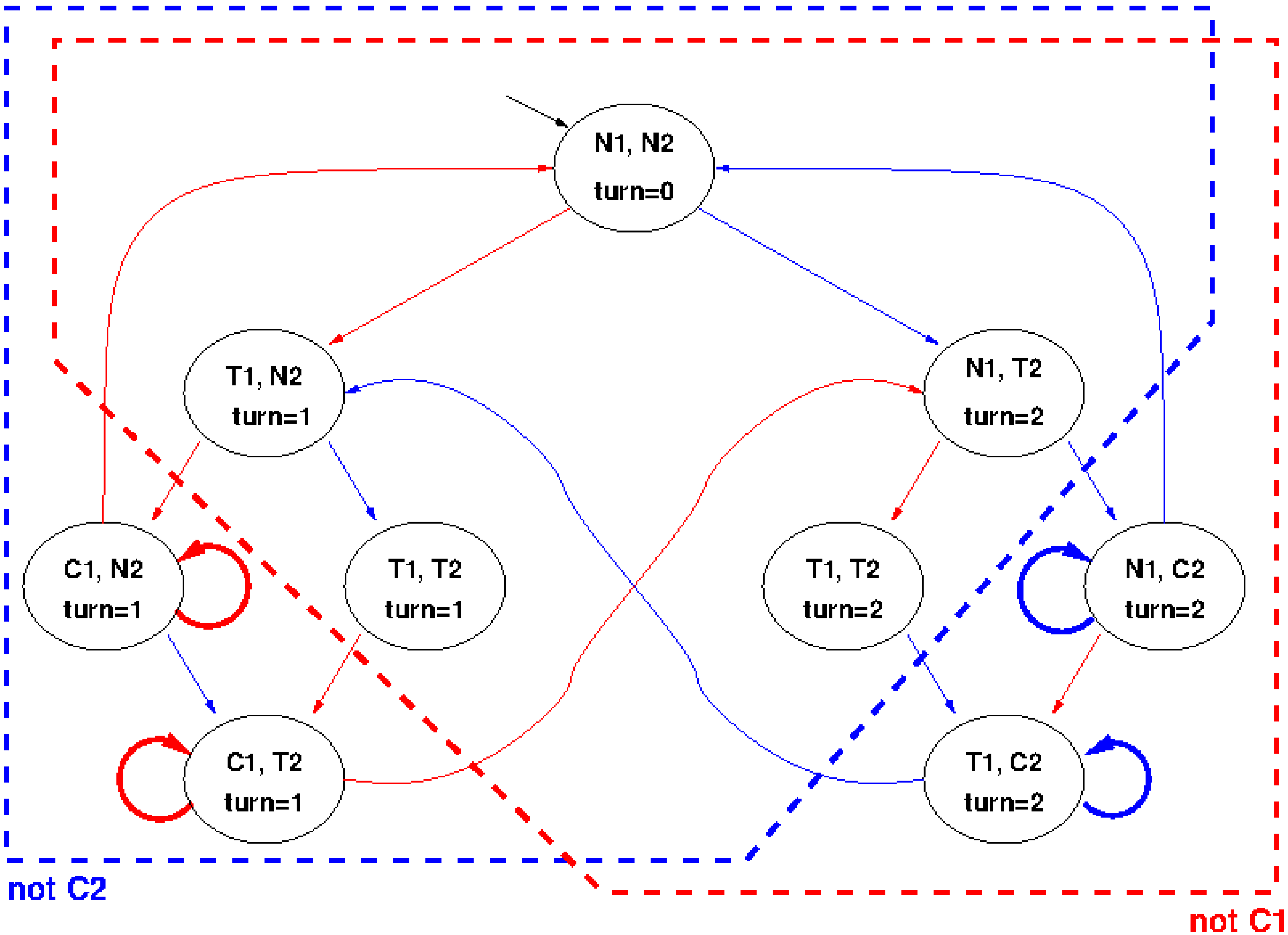
Finite-State Model Checking

Invariant Checking

Model checking



Mutual exclusion example



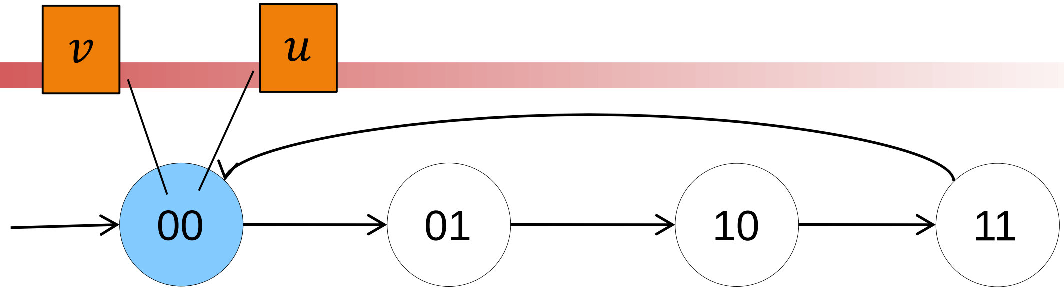
N: non-critical
T: trying
C: critical
User1
User2

Property:
always
not C1 or not C2
i.e.
(C1 and C2)
is not reachable

Symbolic representation

- Symbolic Boolean variables $V = \{v_1, \ldots, v_n\}$ to represent the state space
- A state is an assignment to the variables
- Symbolic formulas used to represent:
  - Set of states: $\phi(V) \equiv \{ s \mid s \models \phi \}$
    - Abuse of notation $s \in \phi$ iff $s \models \phi$
  - Set of transitions: $\phi(V, V') \equiv \{ \langle s, s' \rangle \mid \langle s, s' \rangle \models \phi \}$
    - Where the variables $V' = \{v'_1, \ldots, v'_n\}$ represent next state variables
- A transition system is a tuple $\langle V, I, T \rangle$ where:
  - V is the set of variables
  - The set of initial states represented by the formula $I(V)$
  - The transition relation represented by the formula $T(V, V')$

Example



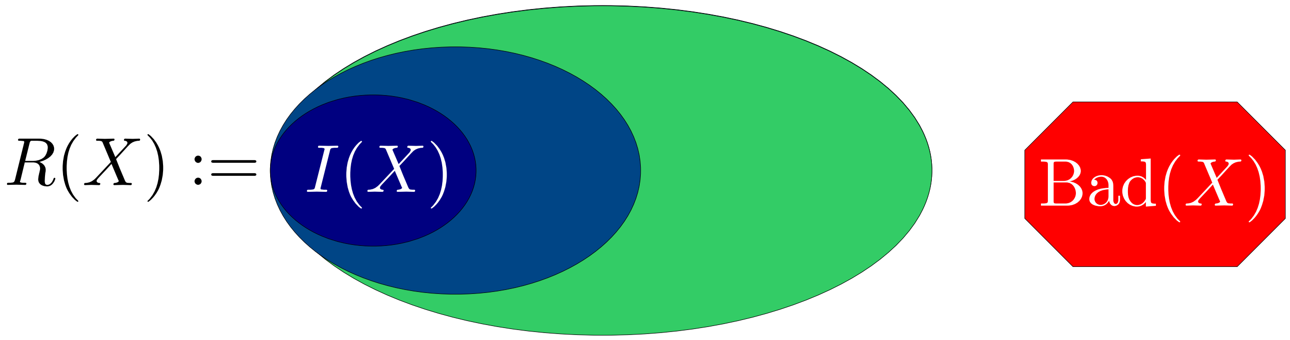
- $V := \{u, v\}$
- $I := \neg u \wedge \neg v$
- $T := u' \leftrightarrow \neg u \wedge v' \leftrightarrow (u\ xor\ v)$

Invariant properties

- A path of the system S is a sequence $s_0, s_1, \ldots, s_k$ of states such that $s_0 \vDash I$ and for all $i, 0 \leq i < k$, $s_i, s_{i+1} \vDash T$
- A state s is reachable iff there exists a path $s_0, s_1, \ldots, s_k$ such that $s = s_k$
- A formula $P(V)$ is an invariant iff for all paths $s_0, s_1, \ldots, s_k$, for all $i, s_i \vDash P$
- Equivalent to say that no state in $\neg P$ is reachable

Forward reachability checking

- Forward image computation:
  - Compute all states reachable from Q in one transition:
    $$FwdImg(Q) \coloneqq \exists V(Q(V) \wedge T(V,V'))[V/V']$$
- Prove that a set of states Bad is not reachable:
  - Start from initial states: $R \coloneqq I$
  - Apply $FwdImg$ iteratively: $oldR \coloneqq R; R \coloneqq FwdImg(R) \cup R$
  - until fixpoint $oldR = R$

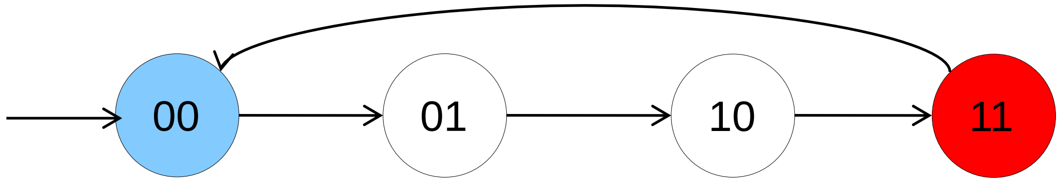


Bounded Model Checking

- Reachability encoded into a satisfiability problem
  $I(V_0) \wedge T(V_0, V_1) \wedge T(V_1, V_2) \wedge \cdots \wedge T(V_{k-1}, V_k) \wedge Bad(V_k)$
- The formula is sat iff there exists a path of length k that reaches Bad
- Checked for increasing values of k
- Exploited incrementality of SAT solvers
- Finite-state space $\Rightarrow$ a completeness threshold K exists
  - If unsat for all $k \leq K$ then Bad is not reachable
  - K is typically very large $\Rightarrow$ unfeasible to reach in practice

Example

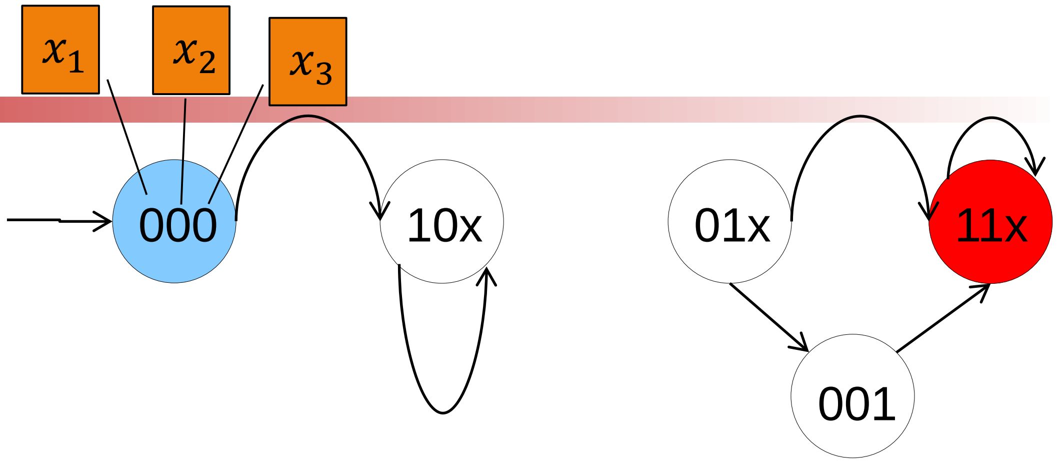
- $V \coloneqq \{u, v\}$
- $I \coloneqq \neg u \wedge \neg v$
- $T \coloneqq u' \leftrightarrow u \wedge v' \leftrightarrow (u\ xor\ v)$
- $Bad \coloneqq u \wedge v$
- BMC:
  - $(\neg u_0 \wedge \neg v_0) \wedge (u_0 \wedge v_0)$ UNSAT
  - $(\neg u_0 \wedge \neg v_0) \wedge \big(u_1 \leftrightarrow u_0 \wedge v_1 \leftrightarrow (u_0\ xor\ v_0)\big) \wedge (u_1 \wedge v_1)$ UNSAT
  - …
  - $(\neg u_0 \wedge \neg v_0) \wedge \big(u_1 \leftrightarrow u_0 \wedge v_1 \leftrightarrow (u_0\ xor\ v_0)\big)$
    $\big(u_2 \leftrightarrow u_1 \wedge v_2 \leftrightarrow (u_1\ xor\ v_1)\big) \wedge$
    $\big(u_3 \leftrightarrow u_2 \wedge v_3 \leftrightarrow (u_2\ xor\ v_2)\big) \wedge (u_3 \wedge v_3)$
    SAT



Induction and K-induction

- Induction
  - Base case: check if the initial state satisfies P (invariant)
  - Inductive case: check if the transitions preserve the invariant
    $$P(V) \wedge T(V, V') \vDash P(V')$$
  - We say P is inductive invariant
- K-induction
  - Base case: check if all initial path satisfies P (invariant) up to k steps
  - Inductive case: check if every path of $k+1$ steps preserve the invariant
    $$P(V_0) \wedge T(V_0, V_1\ ) \wedge P(V_1) \wedge T(V_1, V_2) \wedge \cdots \wedge P(V_{k-1}) \wedge T(V_{k-1}, V_k) \vDash P(V_k)$$
  - Strengthened with simple path condition to avoid repeating states
  - We say P is k-inductive invariant
- Typically however P is not (k-)inductive
  - ⇨ find Inv such that Inv is inductive invariant and $Inv \vDash P$

Example



- $V := \{x_1, x_{2,}, x_3\}$
- $I := \neg x_1 \wedge \neg x_2 \wedge \neg x_3$
- $Bad := x_1 \wedge x_2$
- $P := \neg x_1 \vee \neg x_2$
- Inductive?
  - No
- k-inductive?
  - Yes for k=3
- Inductive invariant?

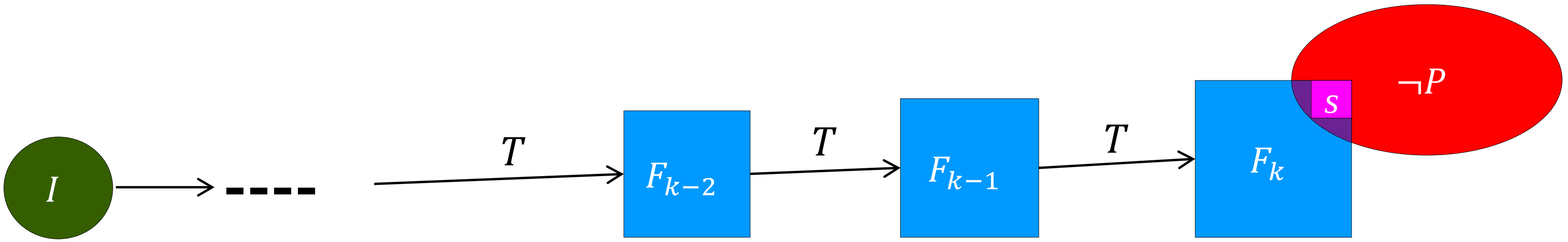
Finite State Model-Checking

IC3

IC3

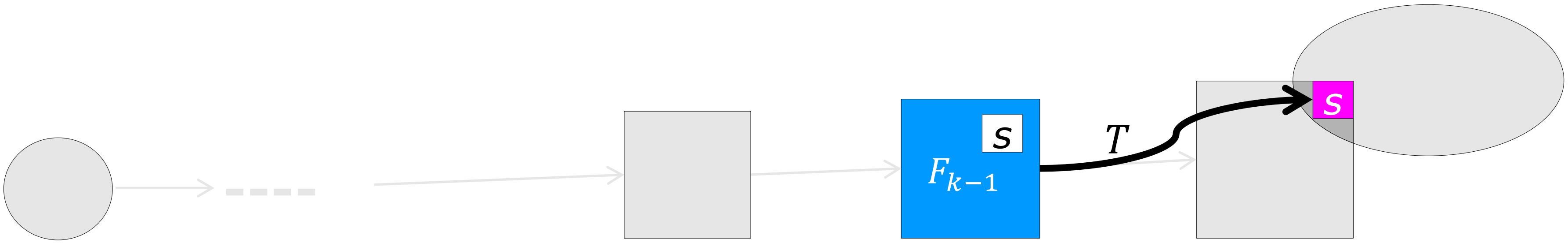
- Very successful SAT-based model checking algorithm
- Based on induction
  - Given a symbolic transition system and invariant property P, build an inductive invariant F s.t. $F \vDash P$
- Inductive invariant built incrementally
  - Trace of formulas $F_0 \equiv I, F_1, \ldots, F_k$ s.t:
    - for $i > 0$, F_i is a set of clauses, overapproximation of states reachable in up to i steps
    - $F_{i+1} \subseteq F_i$ (so $F_i \vDash F_{i+1}$)
    - $F_i \wedge T \vDash F'_{i+1}$
    - For all $\mathrm{i} < \mathrm{k}, F_i \vDash P$
- Strengthen formulas until $F_k = F_{k+1}$
- Exploiting efficient SAT solvers

A (very) high level view of IC3



- Blocking phase: incrementally strengthen trace until $F_k \vDash P$
  - Get bad cube s

A (very) high level view of IC3

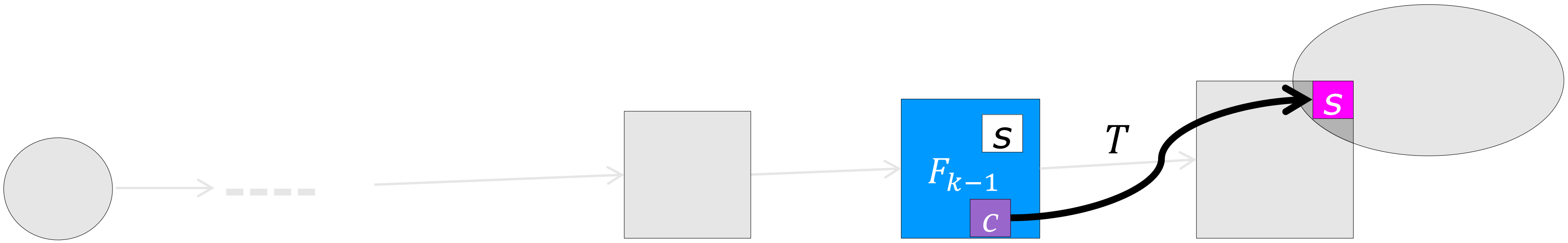


- Blocking phase: incrementally strengthen trace until $F_k \vDash P$
  - Get bad cube s
  - Call SAT solver on $F_{k-1} \wedge \neg s \wedge T \wedge s'$

    (i.e., check if $F_{k-1} \wedge \neg s \wedge T \vDash \neg s'$)

Check if $\neg s$ is inductive relative to F_{k-1}

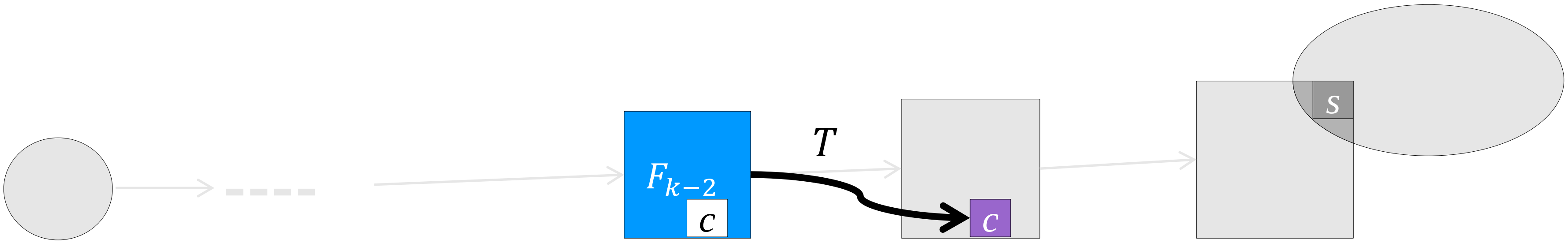
A (very) high level view of IC3



- Blocking phase: incrementally strengthen trace until $F_k \vDash P$
  - Get bad cube s
  - Call SAT solver on $F_{k-1} \wedge \neg s \wedge T \wedge s'$
    - SAT: s is reachable from F_{k-1} in 1 step
    - Get a cube c in the preimage of s and try (recursively) to prove it unreachable from $F_{k-2}, \ldots$
      - c is a counterexample to induction (CTI)

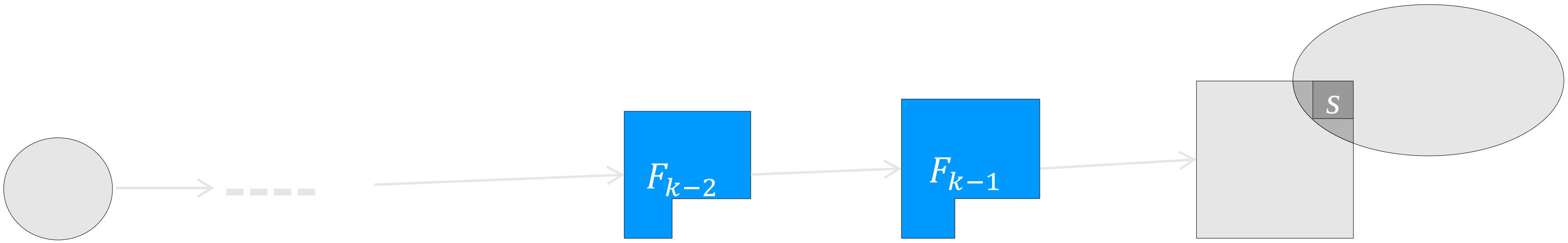
If I is reached, a counterexample to P is found

A (very) high level view of IC3



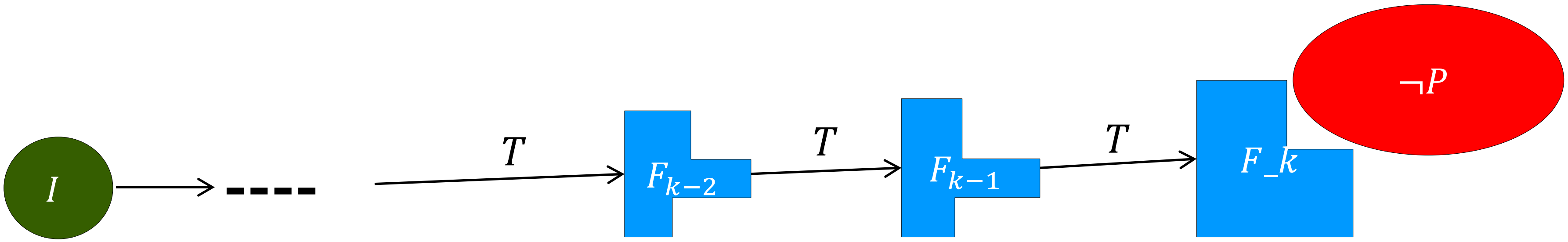
- Blocking phase: incrementally strengthen trace until $F_k \models P$
  - Get bad cube s
  - Call SAT solver on $F_{k-1} \wedge \neg s \wedge T \wedge s'$
    - UNSAT: $\neg s$ is inductive relative to F_{k-2}
    - Generalize c to g and block by adding $\neg g$ to $F_{i-1}, F_{i-2}, \ldots, F_1$

A (very) high level view of IC3



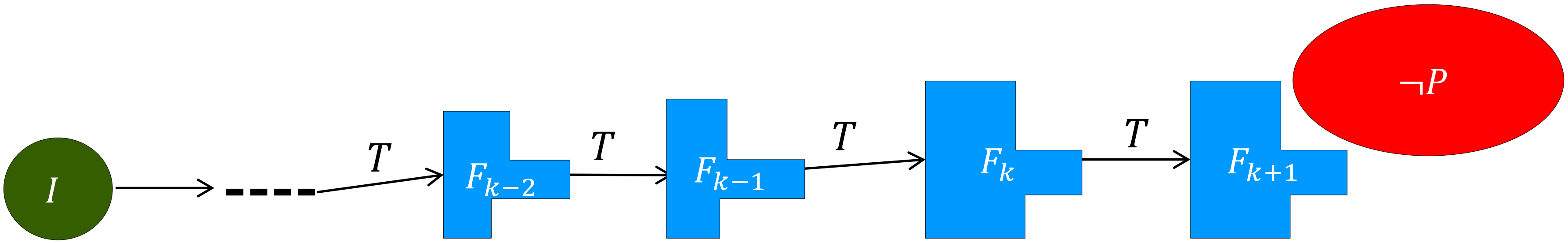
- Blocking phase: incrementally strengthen trace until $F_k \vDash P$
  - Get bad cube s
  - Call SAT solver on $F_{k-1} \wedge \neg s \wedge T \wedge s'$
    - UNSAT: $\neg s$ is inductive relative to F_{k-2}
    - Generalize c to g and block by adding $\neg g$ to $F_{i-1}, F_{i-2}, \ldots, F_1$

A (very) high level view of IC3



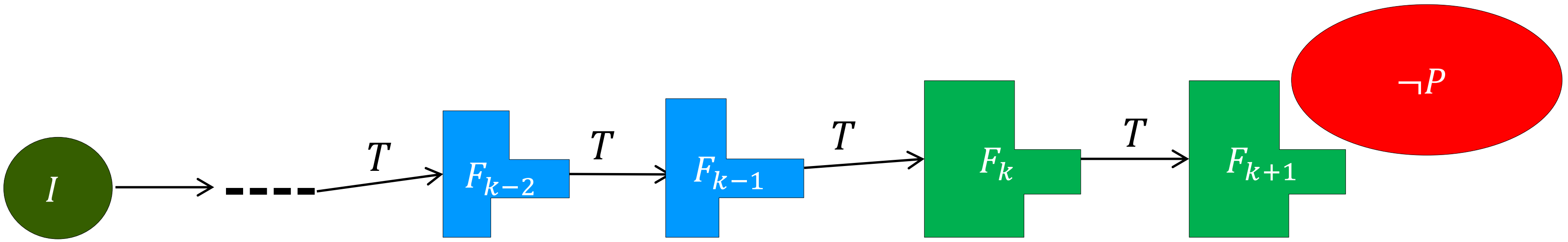
- Propagation: extend trace to F_{k+1} and push forward clauses
  - For each *i* and each clause $c \in F_i$:
    - Call SAT solver on $F_i \wedge T \wedge \neg c'$
      - If UNSAT, add c to F_{i+1}

A (very) high level view of IC3



- Propagation: extend trace to F_{k+1} and push forward clauses
  - For each i and each clause $c \in F_i$:
    - Call SAT solver on $F_i \wedge T \wedge \neg c'$
      - If UNSAT, add c to F_{i+1}

A (very) high level view of IC3



- Propagation: extend trace to F_{k+1} and push forward clauses
  - For each i and each clause $c \in F_i$:
    - Call SAT solver on $F_i \wedge T \wedge \neg c'$
      - If UNSAT, add c to F_{i+1}
- If $F_i \equiv F_{i+1}$, P is proved,
  - otherwise start another round of blocking and propagation

Inductive Clause Generalization

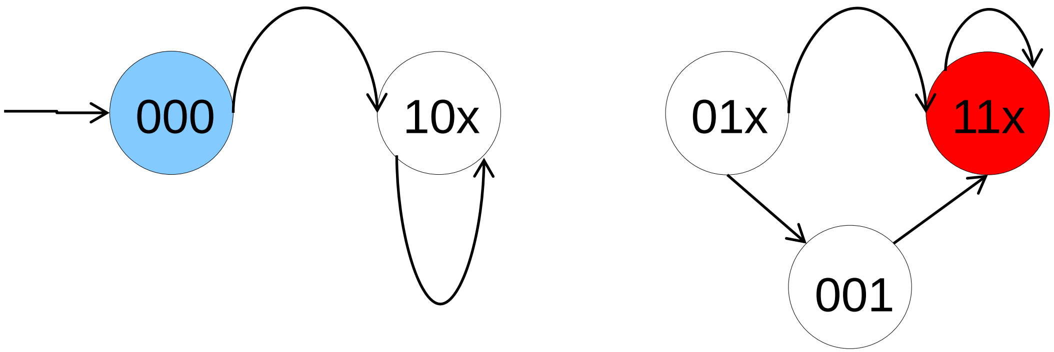
- Crucial step of IC3
- Given a relatively inductive clause $c \stackrel{\text{def}}{=} \{l_1, \ldots, l_n\}$
- compute a generalization $g \subseteq c$ that is still inductive

$$F_{i-1} \land T \land g \models g' \qquad (1)$$

- Drop literals from c and check that (1) still holds
  - Accelerate with unsat cores returned by the SAT solver
    - Using SAT under assumptions
- However, make sure the base case still holds
  - If $I \not\models c \setminus \{l_j\}$, then l_j cannot be dropped

Example

No counterexamples of length 0



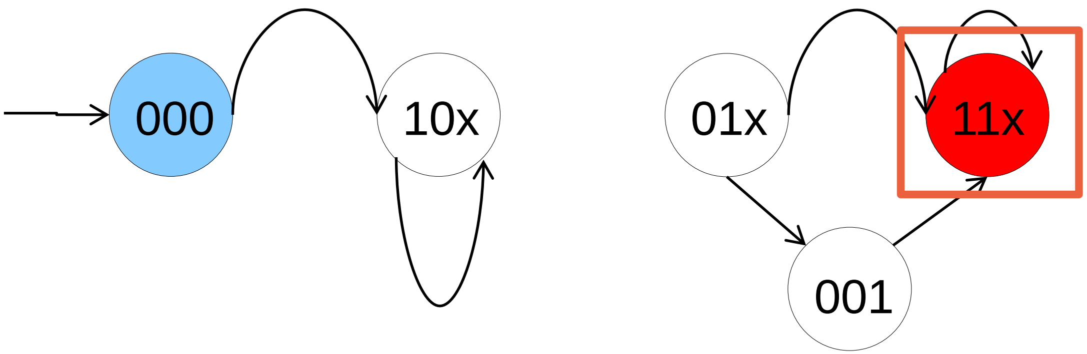
$$I = \neg x_1 \wedge \neg x_2 \wedge \neg x_3$$
$$P = \neg x_1 \vee \neg x_2$$

$$F_0 = I$$
$$F_1 = \top$$

[borrowed and adapted from F. Somenzi]

Example

Get bad cube $c = x_1 \wedge x_2$ in $F_1 \wedge \neg P$

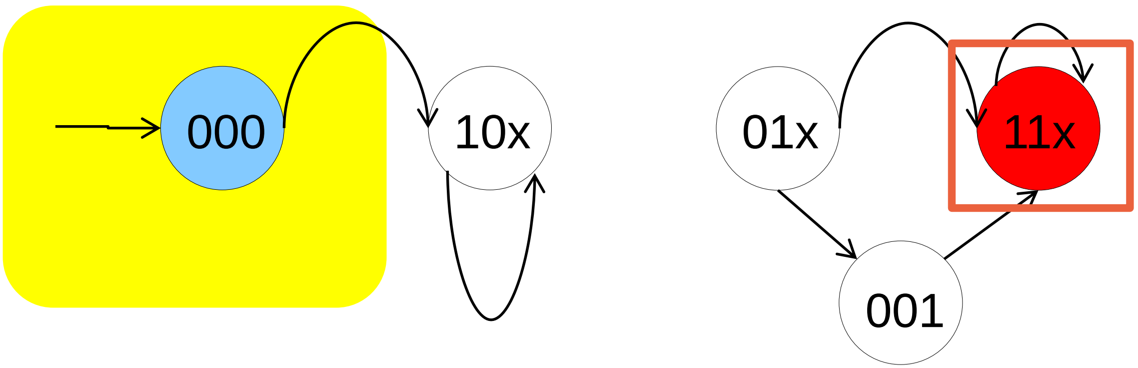


$$I = \neg x_1 \wedge \neg x_2 \wedge \neg x_3$$
$$P = \neg x_1 \vee \neg x_2$$

$$F_0 = I$$
$$F_1 = \top$$

Example

Is $\neg c$ inductive relative to F_0? $\quad F_0 \wedge T \wedge \neg c \models \neg c'$

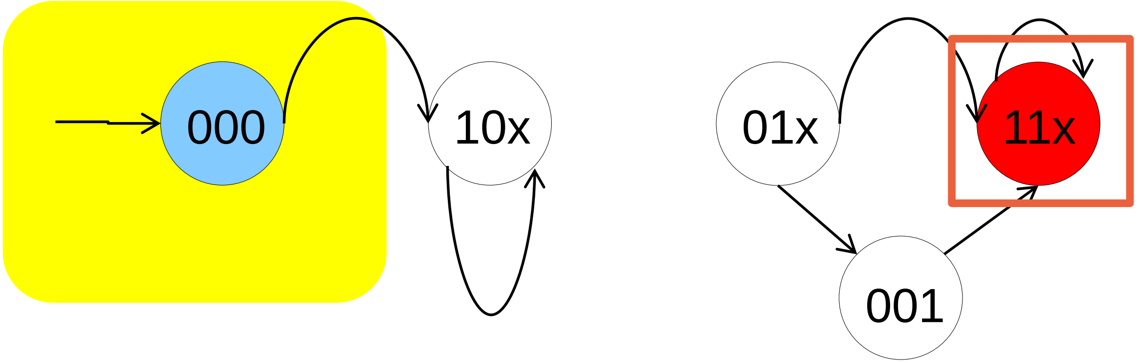


$$I = \neg x_1 \wedge \neg x_2 \wedge \neg x_3$$
$$P = \neg x_1 \vee \neg x_2$$

$$F_0 = I$$
$$F_1 = \top$$

Example

Yes, generalize $\neg c = \neg x_1 \vee \neg x_2$

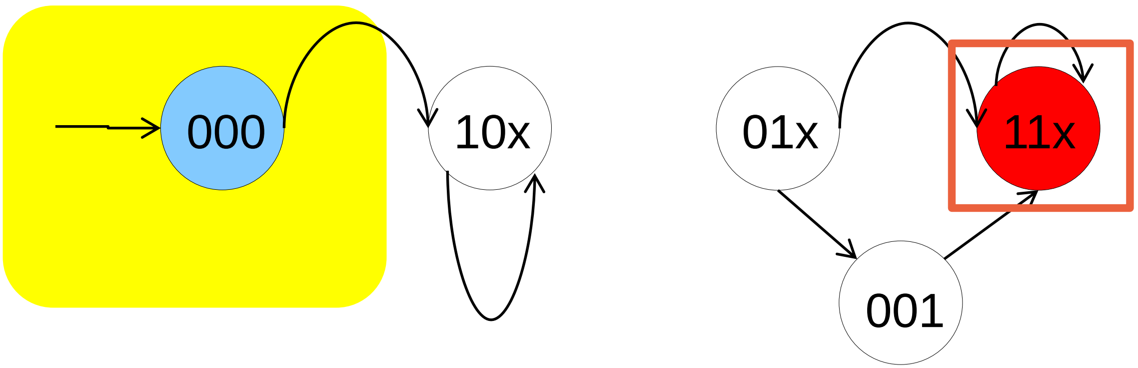


$$I = \neg x_1 \wedge \neg x_2 \wedge \neg x_3$$
$$P = \neg x_1 \vee \neg x_2$$

$$F_0 = I$$
$$F_1 = \top$$

Example

Yes, generalize $\neg c = \neg x_1 \vee \neg x_2$



$$I = \neg x_1 \wedge \neg x_2 \wedge \neg x_3$$
$$P = \neg x_1 \vee \neg x_2$$

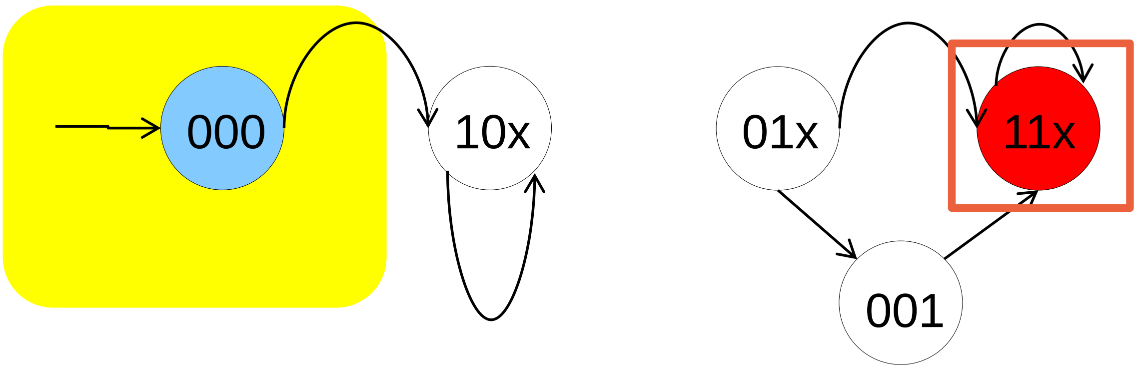
$$F_0 = I$$
$$F_1 = \top$$

Try dropping $\neg x_2$

$$F_0 \wedge T \wedge \neg x_1 \not\models \neg x_1'$$

Example

Yes, generalize $\neg c = \neg x_1 \vee \neg x_2$



$$I = \neg x_1 \wedge \neg x_2 \wedge \neg x_3$$
$$P = \neg x_1 \vee \neg x_2$$

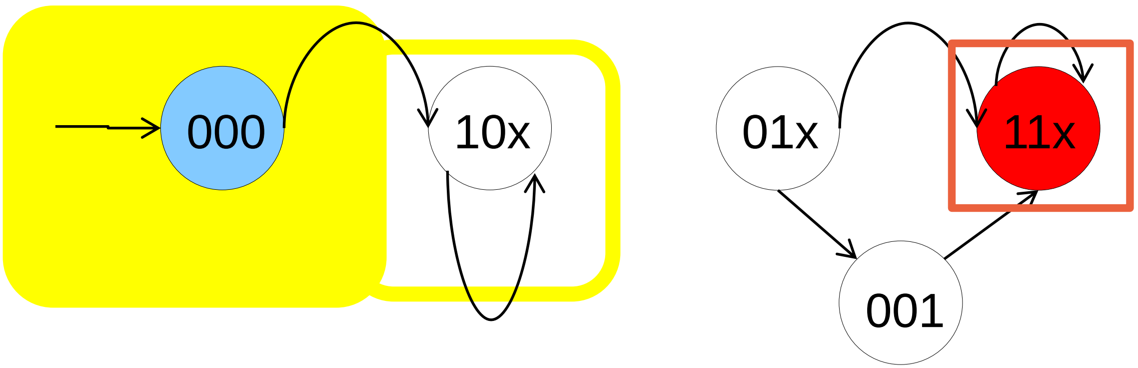
$$F_0 = I$$
$$F_1 = \top$$

Try dropping $\neg x_1$

$$F_0 \wedge T \wedge \neg x_2 \models \neg x_2'$$

Example

Yes, generalize $\neg c = \neg x_1 \vee \neg x_2$



$$I = \neg x_1 \wedge \neg x_2 \wedge \neg x_3$$
$$P = \neg x_1 \vee \neg x_2$$

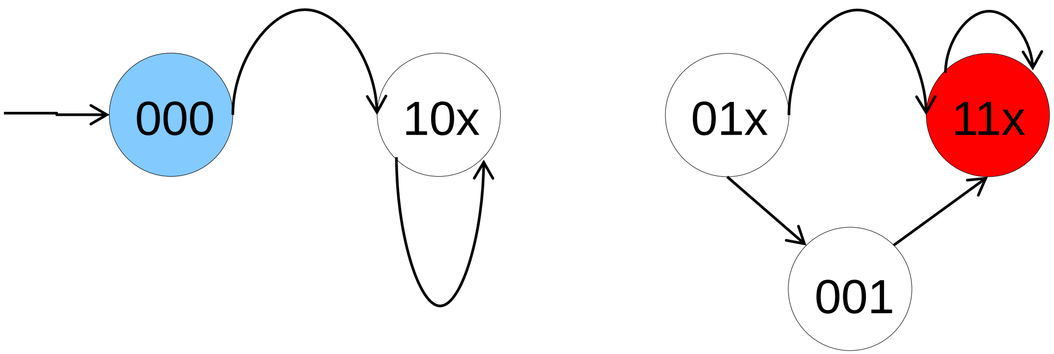
$$F_0 = I$$
$$F_1 = \top$$

Try dropping $\neg x_1$

$$F_0 \wedge T \wedge \neg x_2 \models \neg x_2'$$

Example

Update F_1

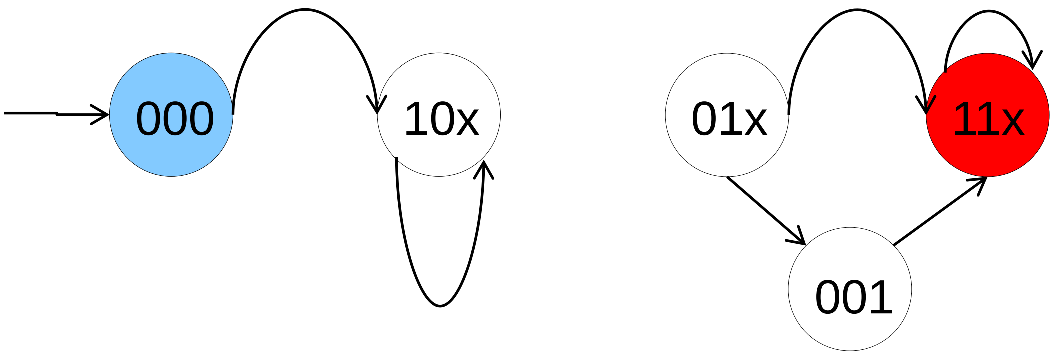


$$I = \neg x_1 \land \neg x_2 \land \neg x_3$$
$$P = \neg x_1 \lor \neg x_2$$

$$F_0 = I$$
$$F_1 = \neg x_2$$

Example

Blocking done for F_1. Add F_2 and propagate forward

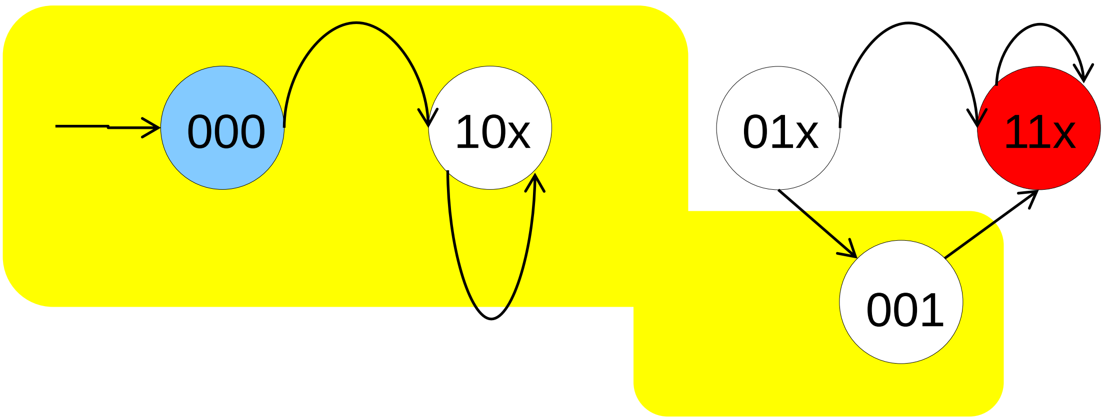


$$I = \neg x_1 \wedge \neg x_2 \wedge \neg x_3$$
$$P = \neg x_1 \vee \neg x_2$$

$$F_0 = I$$
$$F_1 = \neg x_2$$
$$F_2 = \top$$

Example

No clause propagates from F_1 to F_2

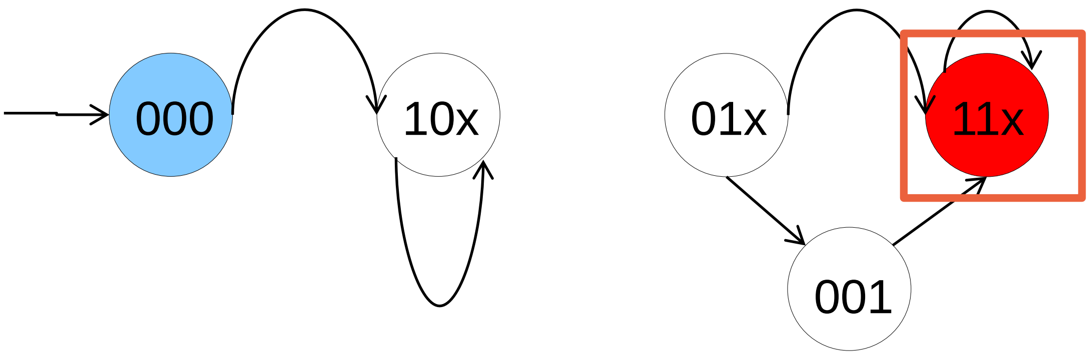


$$I = \neg x_1 \wedge \neg x_2 \wedge \neg x_3$$
$$P = \neg x_1 \vee \neg x_2$$

$$F_0 = I$$
$$F_1 = \neg x_2$$
$$F_2 = \top$$

Example

Get bad cube $c = x_1 \wedge x_2$ in $F_2 \wedge \neg P$

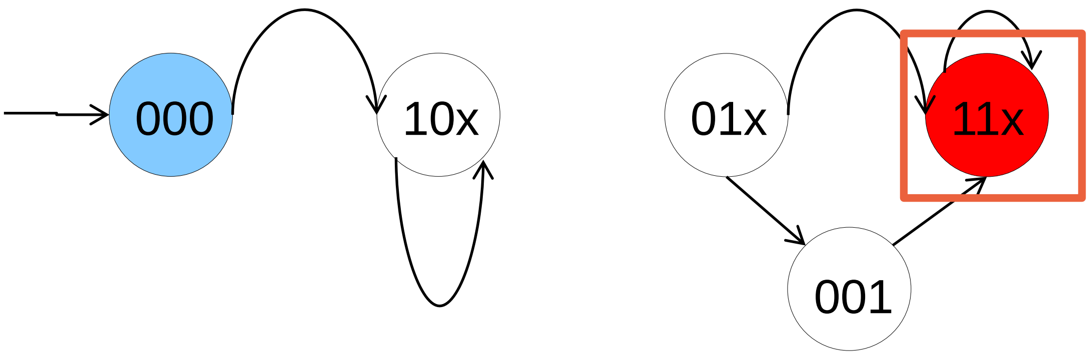


$$I = \neg x_1 \wedge \neg x_2 \wedge \neg x_3$$
$$P = \neg x_1 \vee \neg x_2$$

$$F_0 = I$$
$$F_1 = \neg x_2$$
$$F_2 = \top$$

Example

Is $\neg c$ inductive relative to F_1 ? $F_1 \wedge T \wedge \neg c \models \neg c'$

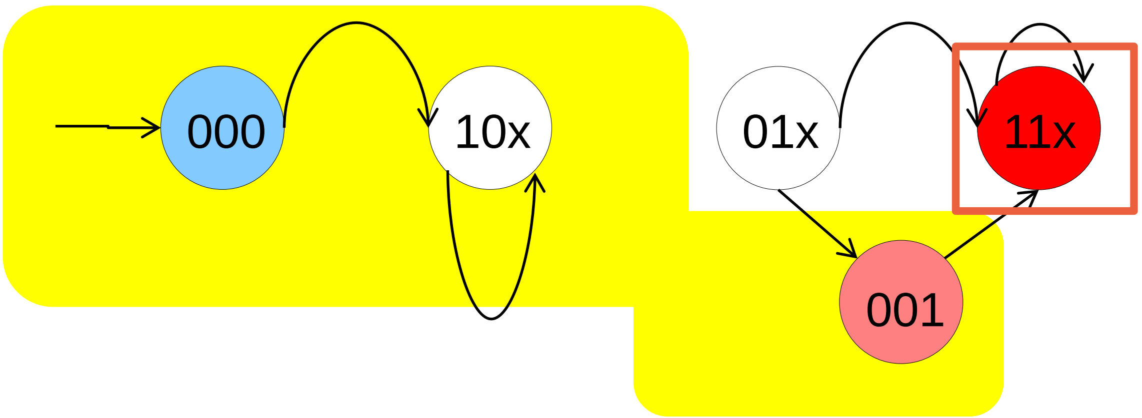


$$I = \neg x_1 \wedge \neg x_2 \wedge \neg x_3$$
$$P = \neg x_1 \vee \neg x_2$$

$$F_0 = I$$
$$F_1 = \neg x_2$$
$$F_2 = \top$$

Example

No, found CTI $s = \neg x_1 \wedge \neg x_2 \wedge x_3$

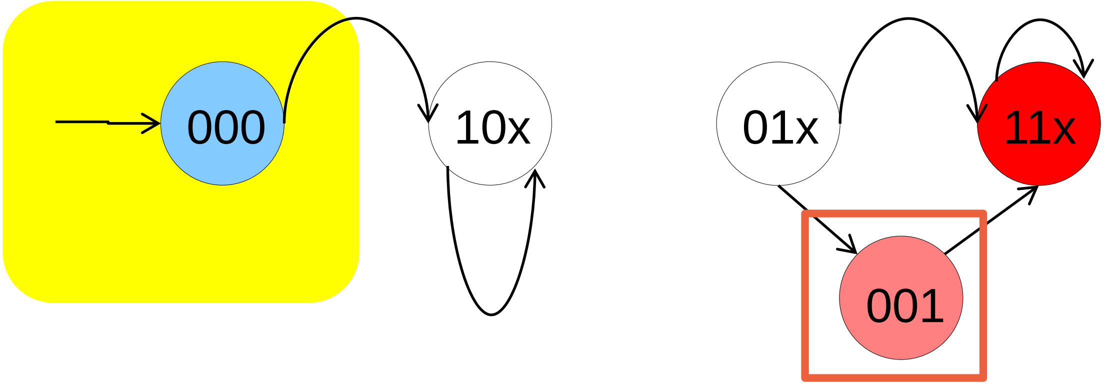


$$I = \neg x_1 \wedge \neg x_2 \wedge \neg x_3$$
$$P = \neg x_1 \vee \neg x_2$$

$$F_0 = I$$
$$F_1 = \neg x_2$$
$$F_2 = \top$$

Example

Try blocking $\neg s$ at level 0: $F_0 \wedge T \wedge \neg s \models \neg s'$

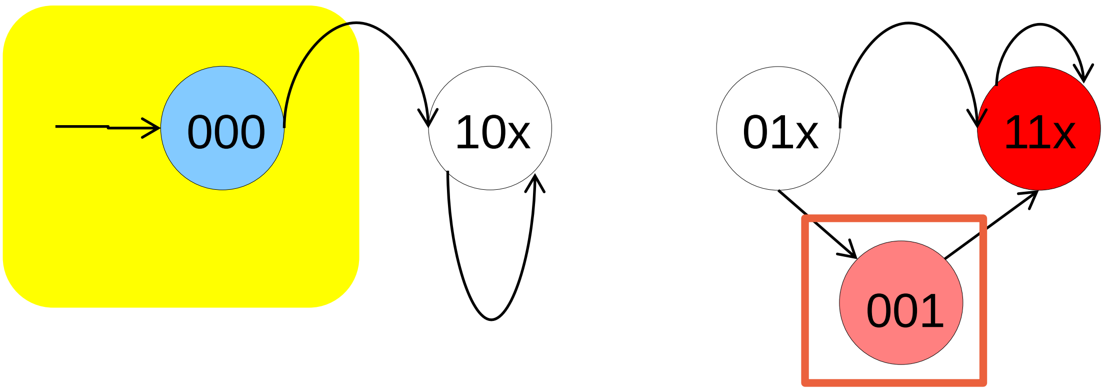


$$I = \neg x_1 \wedge \neg x_2 \wedge \neg x_3$$
$$P = \neg x_1 \vee \neg x_2$$

$$F_0 = I$$
$$F_1 = \neg x_2$$
$$F_2 = \top$$

Example

Yes, generalize $\neg s = x_1 \vee x_2 \vee \neg x_3$



$$I = \neg x_1 \wedge \neg x_2 \wedge \neg x_3$$
$$P = \neg x_1 \vee \neg x_2$$

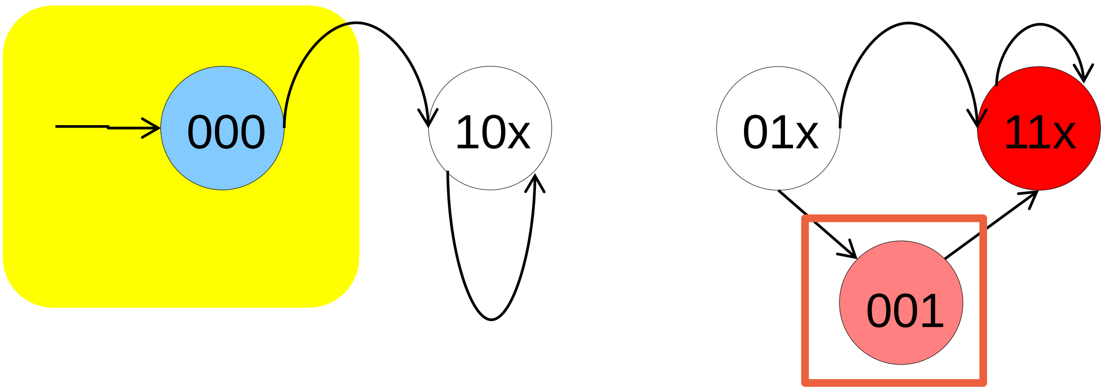
$$F_0 = I$$
$$F_1 = \neg x_2$$
$$F_2 = \top$$

Try dropping x_1

$$F_0 \wedge T \wedge x_2 \vee \neg x_3 \not\models x_2' \vee \neg x_3'$$

Example

Yes, generalize $\neg s = x_1 \vee x_2 \vee \neg x_3$



$$I = \neg x_1 \wedge \neg x_2 \wedge \neg x_3$$
$$P = \neg x_1 \vee \neg x_2$$

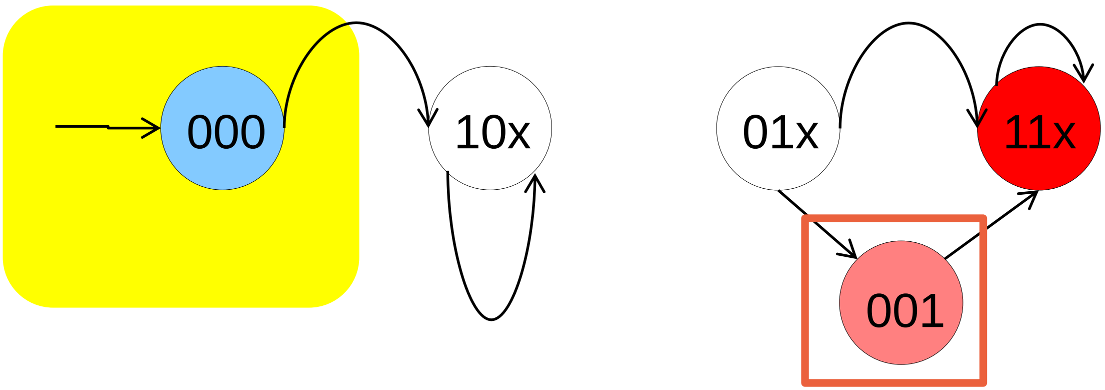
$$F_0 = I$$
$$F_1 = \neg x_2$$
$$F_2 = \top$$

Try dropping x_2

$$F_0 \wedge T \wedge x_1 \vee \neg x_3 \models x_1' \vee \neg x_3'$$

Example

Yes, generalize $\neg s = x_1 \vee x_2 \vee \neg x_3$



$$I = \neg x_1 \wedge \neg x_2 \wedge \neg x_3$$
$$P = \neg x_1 \vee \neg x_2$$

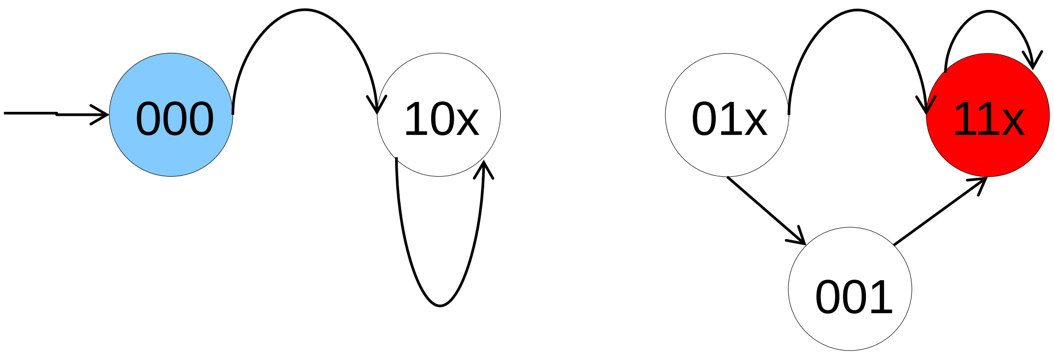
$$F_0 = I$$
$$F_1 = \neg x_2$$
$$F_2 = \top$$

Try dropping x_3

$$I \not\models x_1$$

Example

Update F_1

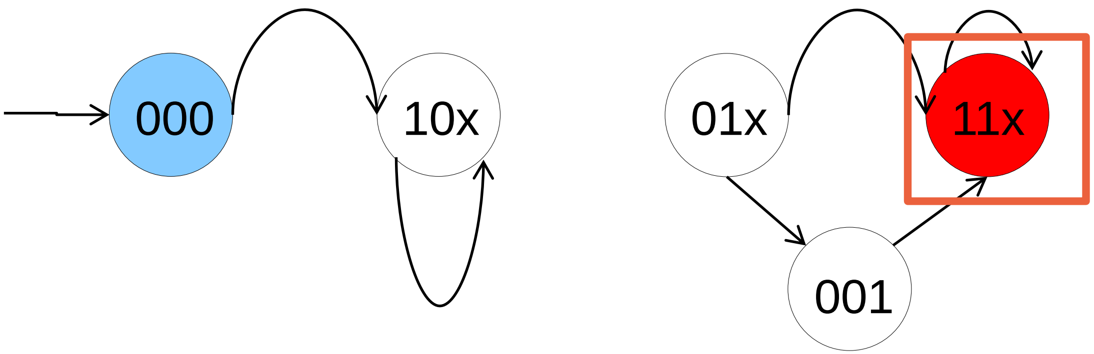


$$I = \neg x_1 \wedge \neg x_2 \wedge \neg x_3$$
$$P = \neg x_1 \vee \neg x_2$$

$$F_0 = I$$
$$F_1 = \neg x_2 \wedge (x_1 \vee \neg x_3)$$
$$F_2 = \top$$

Example

Return to the original bad cube c

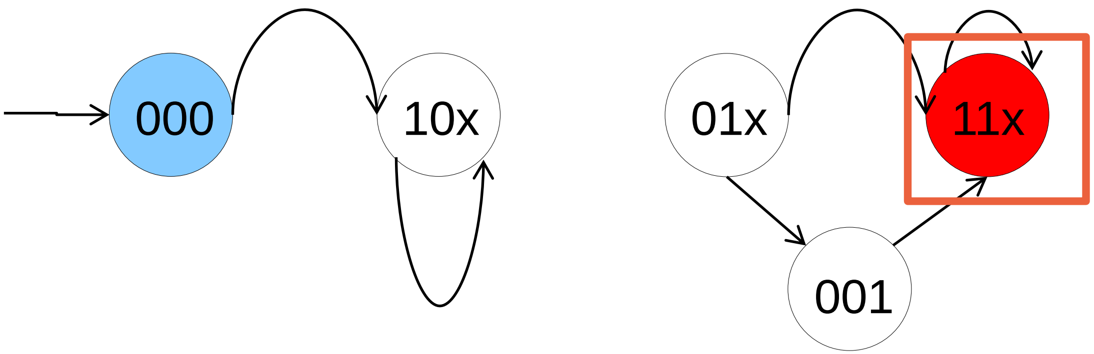


$$I = \neg x_1 \wedge \neg x_2 \wedge \neg x_3$$
$$P = \neg x_1 \vee \neg x_2$$

$$F_0 = I$$
$$F_1 = \neg x_2 \wedge (x_1 \vee \neg x_3)$$
$$F_2 = \top$$

Example

Is $\neg c$ inductive relative to F_1? $F_1 \wedge T \wedge \neg c \models \neg c'$

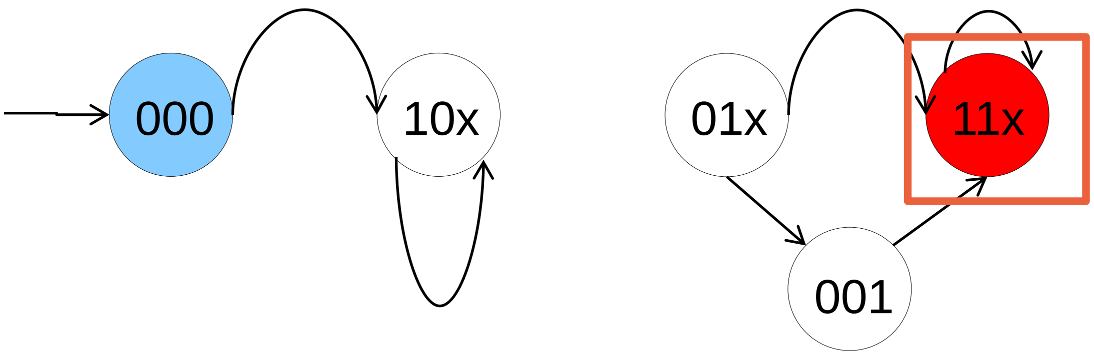


$$I = \neg x_1 \wedge \neg x_2 \wedge \neg x_3$$
$$P = \neg x_1 \vee \neg x_2$$

$$F_0 = I$$
$$F_1 = \neg x_2 \wedge (x_1 \vee \neg x_3)$$
$$F_2 = \top$$

Example

Yes, generalize $\neg c = \neg x_1 \vee \neg x_2$



$$I = \neg x_1 \wedge \neg x_2 \wedge \neg x_3$$
$$P = \neg x_1 \vee \neg x_2$$

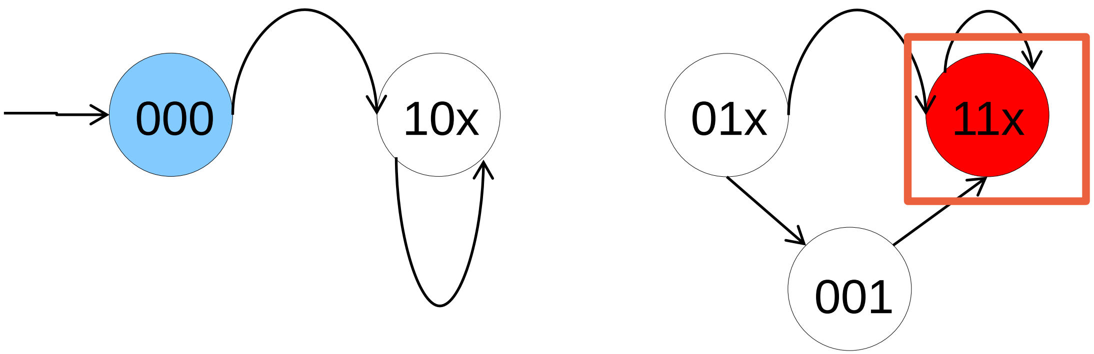
$$F_0 = I$$
$$F_1 = \neg x_2 \wedge (x_1 \vee \neg x_3)$$
$$F_2 = \top$$

Try dropping $\neg x_1$

$$F_1 \wedge T \wedge \neg x_2 \models \neg x_2'$$

Example

Update F_2 and add new frame F_3

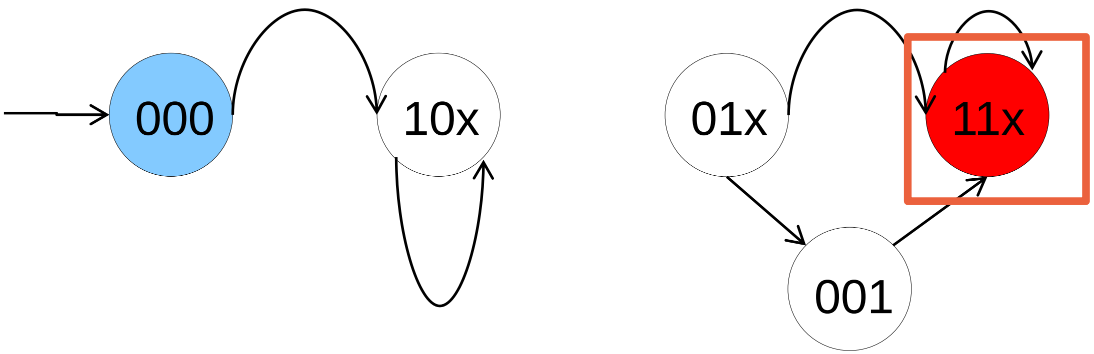


$$I = \neg x_1 \wedge \neg x_2 \wedge \neg x_3$$
$$P = \neg x_1 \vee \neg x_2$$

$$F_0 = I$$
$$F_1 = \neg x_2 \wedge (x_1 \vee \neg x_3)$$
$$F_2 = \neg x_2$$
$$F_3 = \top$$

Example

Perform forward propagation



$$I = \neg x_1 \wedge \neg x_2 \wedge \neg x_3$$
$$P = \neg x_1 \vee \neg x_2$$

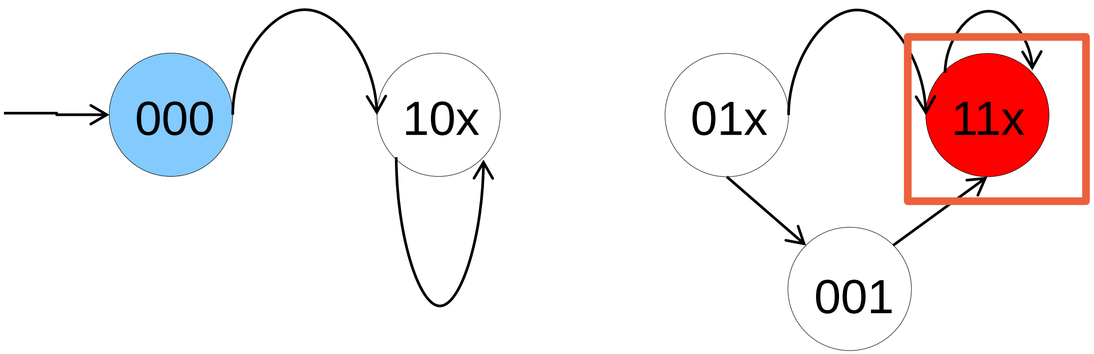
$$F_0 = I$$
$$F_1 = \neg x_2 \wedge (x_1 \vee \neg x_3)$$
$$F_2 = \neg x_2$$
$$F_3 = \top$$

From F_1 to F_2 :

$F_1 \wedge T \models (x_1' \vee \neg x_3')$

Example

Perform forward propagation



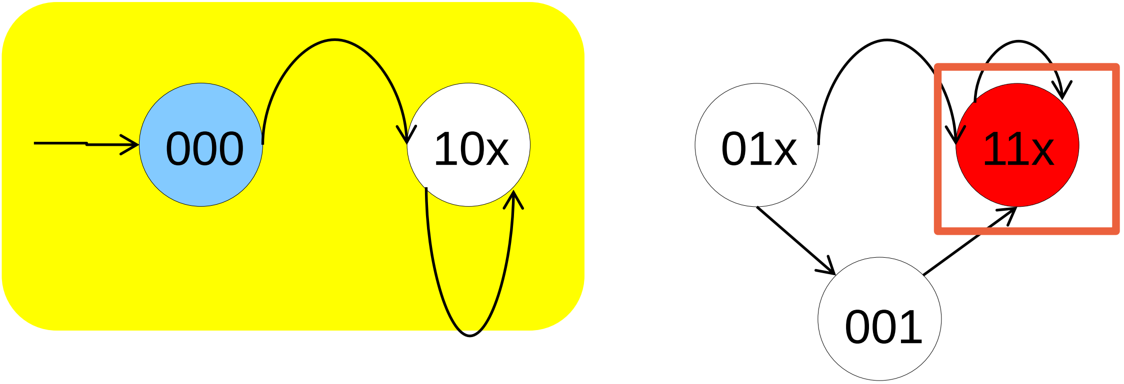
$$I = \neg x_1 \wedge \neg x_2 \wedge \neg x_3$$
$$P = \neg x_1 \vee \neg x_2$$

Found fixpoint!

$$F_0 = I$$
$$F_1 = \neg x_2 \wedge (x_1 \vee \neg x_3)$$
$$F_2 = \neg x_2 \wedge (x_1 \vee \neg x_3)$$
$$F_3 = \top$$

Example

Perform forward propagation



$$I = \neg x_1 \wedge \neg x_2 \wedge \neg x_3$$
$$P = \neg x_1 \vee \neg x_2$$

$$F_0 = I$$
$$F_1 = \neg x_2 \wedge (x_1 \vee \neg x_3)$$
$$F_2 = \neg x_2 \wedge (x_1 \vee \neg x_3)$$
$$F_3 = \top$$

Inductive invariant:

$$F_1 \equiv F_2 \equiv \neg x_2 \wedge (x_1 \vee \neg x_3)$$

Finite State Model-Checking

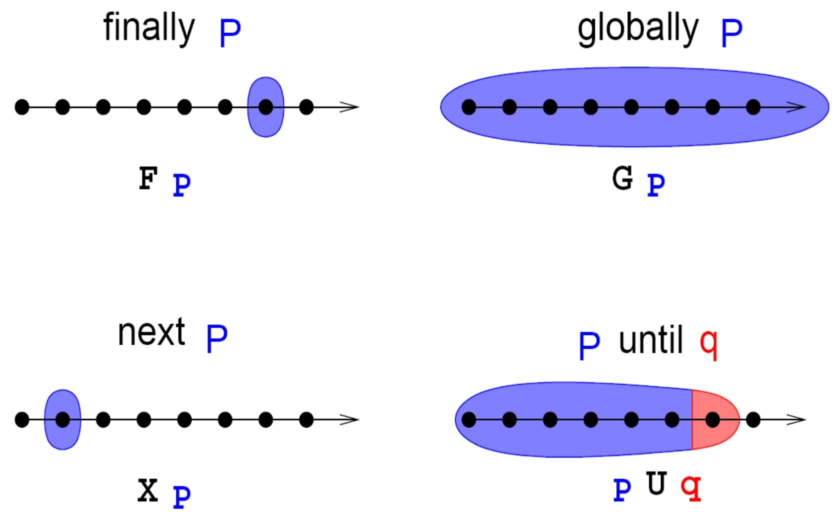
Liveness Checking

Linear Temporal Logic

- Linear models: state sequences (traces)
- Built over set of atomic propositions AP
- LTL is the smallest set of formulas such that:
  - any atomic proposition $p \in AP$ is an LTL formula
  - if ϕ_1 and ϕ_2 are LTL formulas, then $\neg\phi_1, \phi_1 \wedge \phi_2$ and $\phi_1 \vee \phi_2$ are LTL formulas
  - if ϕ_1 and ϕ_2 are LTL formulas, then $X\phi_1, F\phi_1, G\phi_1$ and $\phi_1 U \phi_2$ are LTL formulas

LTL semantics

- Semantics defined for every trace, for every $i \in \mathbb{N}$.
- Given an infinite trace $\pi = s_0, s_1, ..$
  - $\pi, i \models p$ iff $s_i \models p$
  - Standard definition for $\neg$, $\wedge$, $\vee$
  - $\pi, i \models X\phi$ iff $s_{i+1}, s_{i+2}, \ldots \models \phi$
  - $\pi, i \models \phi_1 U\, \phi_2$ iff there exists $j \geq i$, $\pi, j \models \phi_2$ and for all $k, i \leq k < j$, $\pi, k \models \phi_1$
  - $\pi, i \models F\phi$ iff there exists $j \geq i$, $\pi, j \models \phi$
  - $\pi, i \models G\phi$ iff for all $j \geq i$, $\pi, j \models \phi$
- $M \models \phi$ iff $M, \pi, 0 \models \phi$ for every trace π of M.



LTL examples

- Gp "always p" – like invariant (if we assume deadlock freedom)
- $G(p \rightarrow Fq)$ "p is always followed by q" - reaction
- $G(p \rightarrow Xq)$ "whenever p holds, q is set to true" – immediate reaction
- GFp "infinitely many times p" – fairness
- FGp "eventually permanently p"
- $G(speed\_above\_limit \rightarrow (brake\ U\ \neg speed\_above\_limit))$

LTL verification

- Given an LTL property ϕ, build a transition system $M_{\neg\phi}$ with a fairness condition $f_{\neg\phi}$, such that

$$M \times M_{\neg\phi} \vDash FG\neg f_{\neg\phi}$$

- FG requires a doubly-nested fixpoint
- SAT-based approaches typically reduce the problem to safety

Liveness2safety

- Based on the existence of a lasso-shaped counterexample, with $f_{\neg\phi}$ at least once in the loop
- liveness to safety transformation: absence of lasso-shaped counterexamples as an invariant property
  - Duplicate the state variables $V_{copy} \coloneqq \{v_c \mid v \in V\}$
  - Non-deterministically save the current state
  - Remember when $f_{\neg\phi}$ in extra state var $triggered$
  - Invariant: $G\neg(V = V_{copy} \wedge triggered)$

K-liveness

- Simple but effective technique for LTL verification of finite-state systems
- Key insight: $M \times M_{\neg\phi} \models FG\neg f_{\neg\phi}$ iff there exists *k* such that $f_{\neg\phi}$ is visited at most *k* times
  - Again, a safety property
- K-liveness: increase k incrementally
  - Liveness checking as a sequence of safety checks
- Using IC3 as safety checker
  - Exploits the highly incremental nature of IC3

Wrapping up…

- Motivations
- Finite-State Model Checking
  - From BDD-based to SAT-based
- Invariant Checking
  - IC3
- LTL Checking
  - BMC: traces as models, found with SAT checks
  - Liveness to safety
  - Proving limit for violations to fairness

Infinite State Model-Checking

Infinite State Transition System

- Same definition as before: $\langle V, I, T\rangle$
- First-order instead of propositional formulas:
  - Signature: set Σ of constant, functional, and relational symbols
  - Structure: a domain D and interpretation $\mathcal{I}$ of the symbols in the signature
  - Theory: set $\mathcal{T}$ of axioms (a model of $\mathcal{T}$ is a structure that satisfy $\mathcal{T}$)
- Some constant symbols are used as the variables of the transition system
  - They have a flexible interpretation that varies along time
  - The other symbols are rigid
- In the following $\models$ implicitly means $\models_{\mathcal{T}}$, i.e. is restricted to the models of a given theory

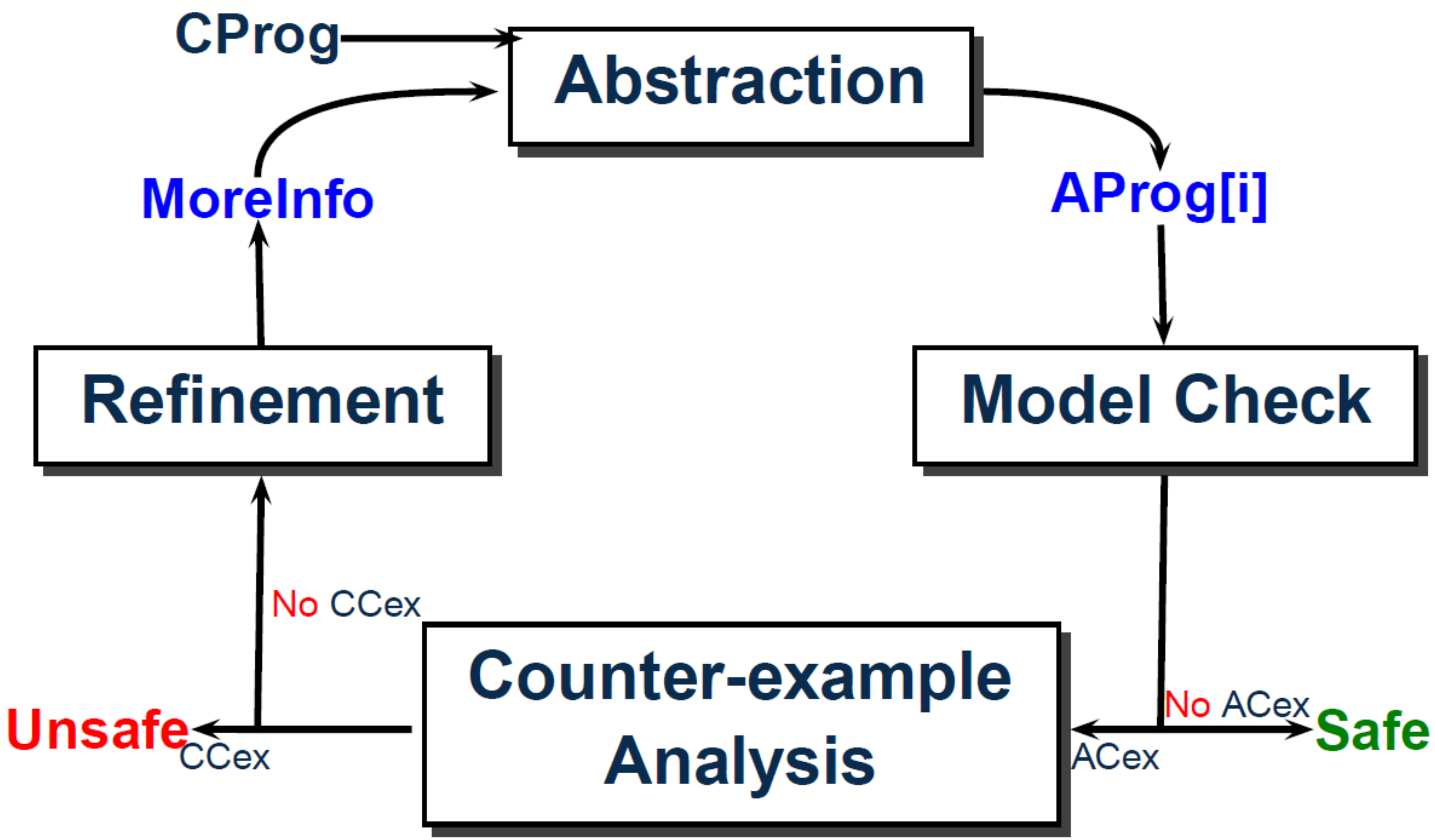
Example

- $V := \{x, y\}$
- $I := y \leq x$
- $T := (x' = x + 1) \wedge (y' \leq y)$
- $\Sigma := \{x, y, 0, 1, +, \leq, \dots\}$
- $\mathcal{T}$:= theory of reals
- $y \leq x \wedge T \vDash_{\mathcal{T}} y' \leq x'$

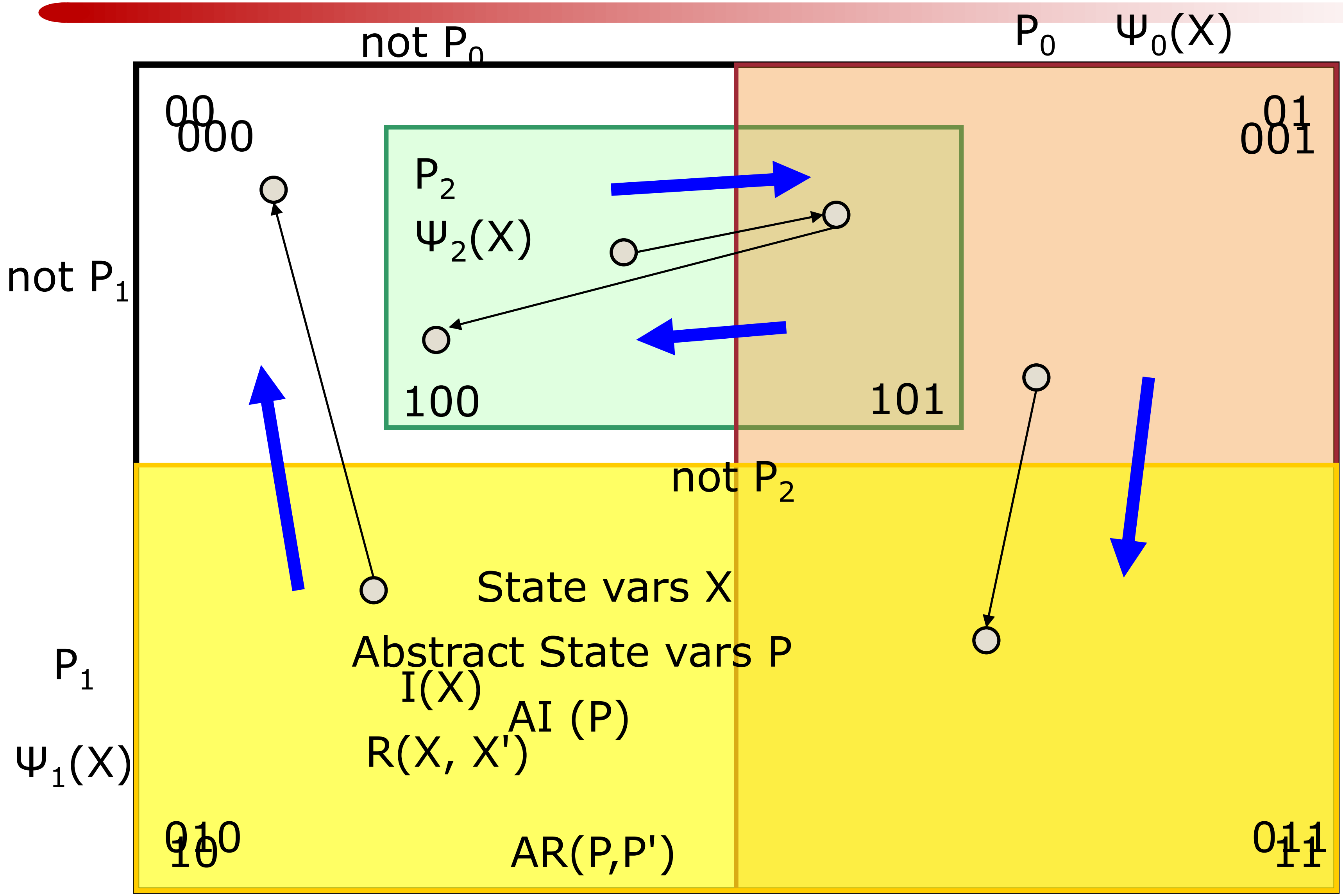
From SAT to SMT

- Previous algorithms assume to have a solver for the satisfiability of formulas
- First developed for finite-state systems with the support of SAT solvers
- SAT solvers substituted by Satisfiability Modulo Theory (SMT) solvers:
  - Satisfiability for decidable fragments of first-order logic
  - SAT solver used to enumerate Boolean models
  - Integrated with decision procedure for specific theories, e.g., theory of real linear arithmetic
- Search algorithms applied to infinite-state systems (although in general undecidable)
- Lift to SMT straightforward for BMC and k-induction
- Not for IC3:
  - Requires alternative effective generalization

Counter-Example Guided Abstraction-Refinement (CEGAR)

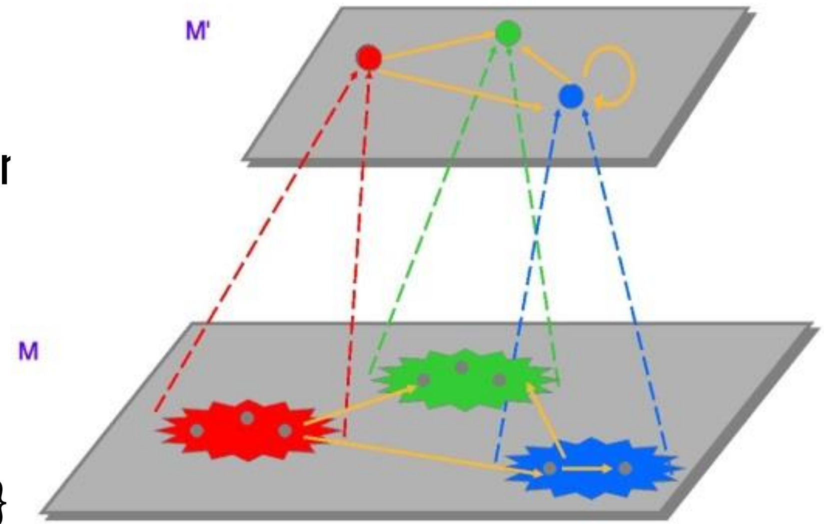


Predicate abstraction



Predicate Abstraction

- Reduction to finite-state MC
- Predicates $\mathbb{P}$ over concrete variables to define the abstractio
- Abstract state space given by Boolean variables, one for each predicate $\hat{V} = \{v_p \mid p \in \mathbb{P}\}$
- Abstract state $\alpha(s) = \{v_p \mid s(p) = \top\}$



- Abstract transition iff there exists a concrete transition between two corresponding concrete states

$$\hat{T} = \{\langle \hat{s}, \hat{s}' \rangle \mid \exists s, s', \alpha(s) = \hat{s}, \alpha(s') = \hat{s}', T(s, s')\}$$

- Transitions computed with ALLSMT:

$$\hat{T}(\hat{V}, \hat{V}') = \exists V, V' (T(V, V') \wedge \bigwedge_{p \in \mathbb{P}} v_p \leftrightarrow p(V) \wedge \bigwedge_{p \in \mathbb{P}} v'_p \leftrightarrow p(V'))$$

Interpolation

An interpolant for an unsatisfiable formula

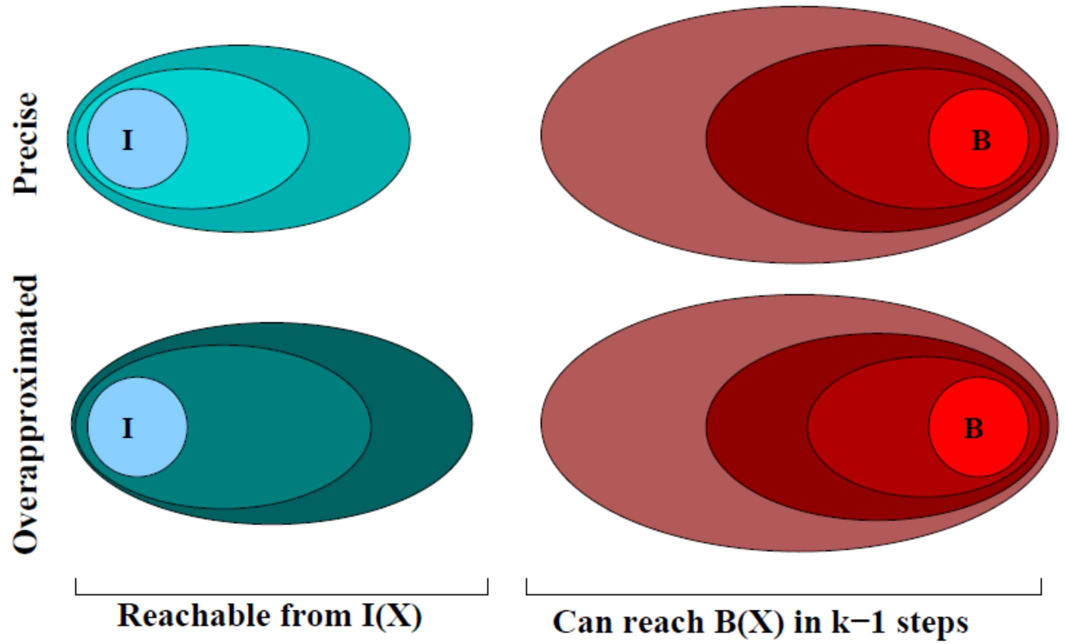
$$\Phi_1(X, Y) \wedge \Phi_2(Y, Z)$$

is a formula $Itp(Y)$ such that:

- $\Phi_1(X, Y) \rightarrow Itp(Y)$
- $Itp(Y) \wedge \Phi_2(Y, Z)$ is unsatisfiable

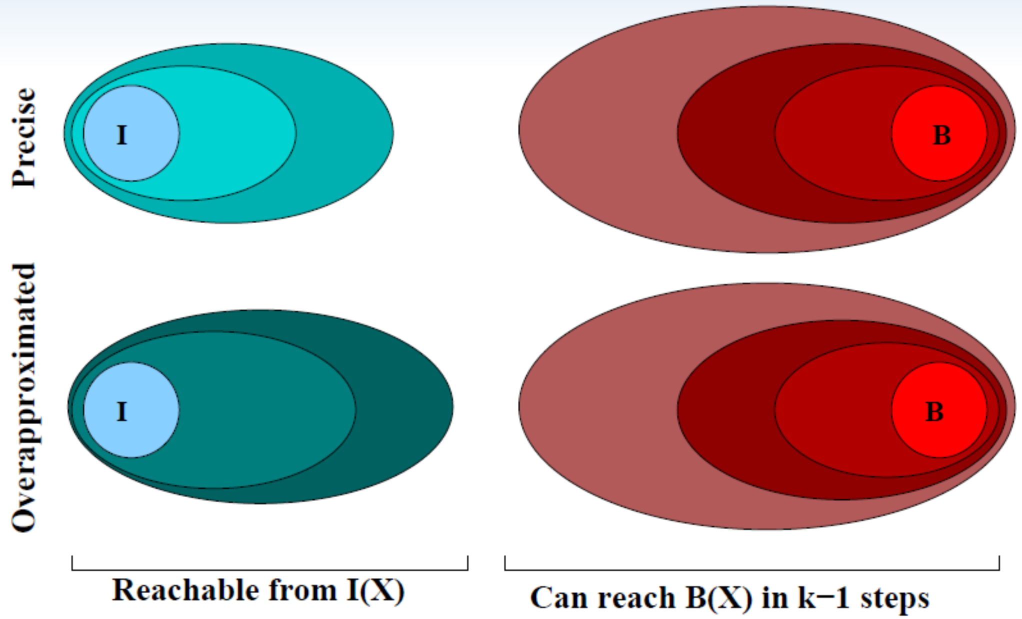
Interpolation-based model checking

$$\overbrace{I(X_0) \wedge R(X_0, X_1)}^{\Phi_1(X_0,X_1)} \underbrace{\wedge}_{Itp(X_1)} \overbrace{R(X_1, X_2) \ldots \wedge R(X_{k-1}, X_k) \wedge B(X_k)}^{\Phi_2(X_1,\ldots,X_k)}$$



$$Itp(X_1) = Itp(R, I(X_0), k)$$

Interpolation-based model checking



- Precise reachability
  - $\mathcal{R}_0 = I$
  - $\mathcal{R}_i = Img(R, \mathcal{R}_{i-1}) \cup \mathcal{R}_{i-1}$
- Interpolation based reachability
  - $Itp_0 = I(X_1)$
  - $Itp_i = Itp(R, Itp_{i-1}, k) \cup Itp_{i-1}$

Abstraction Refinement

- Abstract traces are overapproximations
  - Spurious counterexamples can be generated
- Standard abstraction refinement techniques based on interpolation
  - Sequence of abstract states $\hat{s}_0, \hat{s}_1, \ldots, \hat{s}_k$
  - SMT check on

$$\hat{s}_0(V_0) \wedge T(V_0, V_1) \wedge \hat{s}_1(V_1) \wedge T(V_1, V_2) \wedge \cdots \wedge T(V_{k-1}, V_k) \wedge \hat{s}_k(V_k)$$

  - If unsat, compute sequence of interpolants for

$$[\hat{s}_0(V_0) \wedge T(V_0, V_1) \wedge \cdots \wedge T(V_{i-1}, V_i)]$$

$$[\hat{s}_i(V_i) \wedge T(V_0, V_1) \wedge \cdots \wedge T(V_{k-1}, V_k) \wedge \hat{s}_k(V_k)]$$

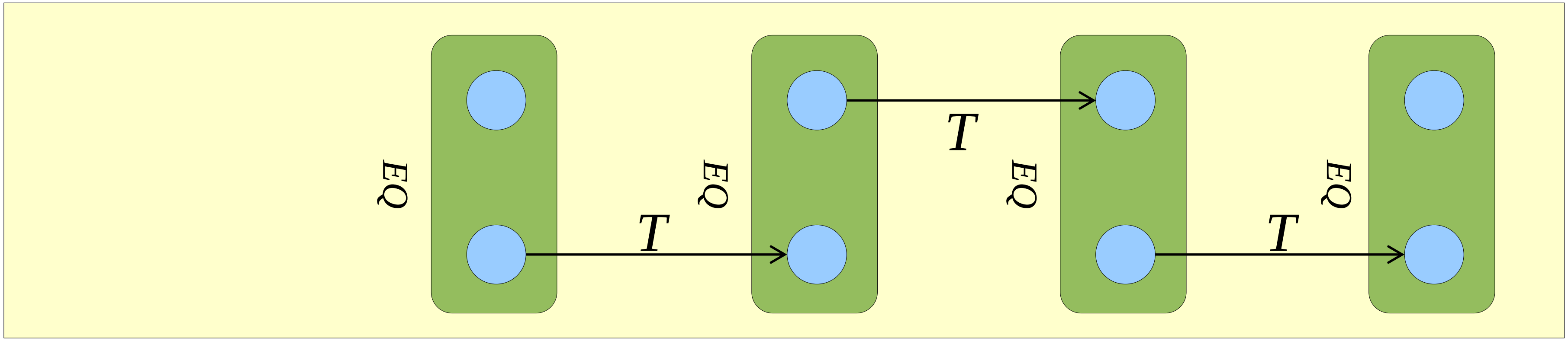
using the same UNSAT proof (called sequence interpolants)

  - Add all the predicates in the interpolants to $\mathbb{P}$

Implicit Predicate Abstraction

- Abstract version of BMC and k-induction, avoiding explicit computation of the abstract transition relation
  - By embedding the abstraction in the SMT encoding
  - $EQ(V_1, V_2) \coloneqq \bigwedge_{p \in \mathbb{P}} p(V_1) \leftrightarrow p(V_2)$
- The abstract unrolling is

$$T(V_0, \overline{V}_1) \wedge EQ(\overline{V}_1, V_1) \wedge T(V_1, \overline{V}_2) \wedge EQ(\overline{V}_2, V_2) \wedge T(V_2, V_3) \wedge \cdots$$



Infinite State Model-Checking

IC3 with Implicit Abstraction

IC3 with Implicit Abstraction

- Integrate the idea of Implicit Abstraction within IC3
- Use abstract transition relation
- Learn clauses only over predicates
- Use abstract relative induction check:

$$AbsRelInd(F, T, \mathrm{c}, \mathbb{P})$$
$$:= F(V) \wedge \mathrm{c}(V) \wedge T(V, \overline{V}) \wedge \bigwedge_{p \in \mathbb{P}} \left(p(V') \leftrightarrow p(\overline{V})\right) \wedge \neg c(V')$$

- If UNSAT ⇨ inductive strengthening as in the Boolean case
- No theory-specific technique needed

IC3 with Implicit Abstraction

- Integrate the idea of Implicit Abstraction within IC3
- Use abstract transition relation
- Learn clauses only over predicates
- Use abstract relative induction check:

$$AbsRelInd(F, T, \mathrm{c}, \mathbb{P})$$
$$:= F(V) \wedge \mathrm{c}(V) \wedge T(V, \overline{V}) \wedge \bigwedge_{p \in \mathbb{P}} \left(p(V') \leftrightarrow p(\overline{V})\right) \wedge \neg c(V')$$

- If SAT ⇨ abstract predecessor from the SMT model
- No preimage needed

Example

- $T \coloneqq (2x_1' - 3x_1 \leq 4x_2' + 2x_2 + 3) \wedge (3x_1 - 2x_2' = 0)$
- $\mathbb{P} \coloneqq \{(x_1 - x_2 \geq 4), (x_1 < 3)\}$
- $s \coloneqq \neg(x_1 - x_2 \geq 4) \wedge (x_1 < 3)$
- $AbsRelInd(\emptyset, T, \neg s, \mathbb{P}) = T\left(V, \overline{V}'\right) \wedge \neg s(V) \wedge s(V') \wedge (x_1 - x_2 \geq 4) \leftrightarrow (\overline{x}_1 - \overline{x}_2 \geq 4) \wedge (x_1 < 3) \leftrightarrow (\overline{x}_1 < 3)$
- $AbsRelInd(\emptyset, T, s, \mathbb{P})$ is SAT
- Compute a predecessor from SMT model:

$$\mu \stackrel{\text{def}}{=} \{x_1 \mapsto 0, x_2 \mapsto 1\}$$

$$\neg(x_1 - x_2 \geq 4) \wedge (x_1 < 3)$$

Abstraction refinement

- Abstract counterexample check can use incremental SMT
- Abstraction refinement is *fully incremental*
- No restart from scratch
- Can keep all the clauses of $F_1, \ldots, F_k$
- Refinements monotonically strengthen T

$$T_{new} \coloneqq T_{old} \wedge \bigwedge_{p \in new\mathbb{P}} \left(p(V) \leftrightarrow p(\overline{V})\right) \wedge \left(p(V') \leftrightarrow p\left(\overline{V}'\right)\right)$$

- All IC3 invariants on $F_1, \ldots, F_k$ are preserved
  - $F_{i+1} \subseteq F_i$ (so $F_i \vDash F_{i+1}$)
  - $F_i \wedge T \vDash F'_{i+1}$
  - For all $\mathrm{i} < \mathrm{k}, F_i \vDash P$

Example

- System with 2 state vars *c* and *d*
  - Init: $(d = 1) \wedge (c \geq d)$
  - Trans: $(c' = c + d) \wedge (d' = d + 1)$
  - Property: $(d > 2) \rightarrow (c > d)$
- Check base case: $\mathrm{Init} \models \mathrm{Property}$

- Predicates $\mathbb{P}$

$(d = 1), (c \geq d),$

$(d > 2), (c > d)$

Example

- System with 2 state vars *c* and *d*
  - Init: $(d = 1) \wedge (c \geq d)$
  - Trans: $(c' = c + d) \wedge (d' = d + 1)$
  - Property: $(d > 2) \rightarrow (c > d)$
- Get bad cube
  - SMT check $F_1 \wedge \neg P$
  - SAT with model $\mu \coloneqq \{c = 0, d = 3\}$
  - Evaluate predicates wrt. μ
    - Return

  $s \coloneqq \{\neg(d = 1), \neg(c \geq d), (d > 2), \neg(c > d)\}$

- Predicates $\mathbb{P}$

  $(d = 1), (c \geq d),$

  $(d > 2), (c > d)$
- Trace
  - $F_0 \coloneqq Init$
  - $F_1 \coloneqq \top$

Example

- System with 2 state vars *c* and *d*
  - Init: $(d = 1) \wedge (c \geq d)$
  - Trans: $(c' = c + d) \wedge (d' = d + 1)$
  - Property: $(d > 2) \rightarrow (c > d)$
- Rec. block s
  - Check

$AbsRelInd(F_0, T, \neg s, \mathbb{P})$
$:= Init \wedge (\overline{c} = c + d) \wedge \left(\overline{d} = d + 1\right)$
$\wedge \left(d' = 1 \leftrightarrow \overline{d} = 1\right) \wedge \left(c' \geq d' \leftrightarrow \overline{c} \geq \overline{d}\right)$
$\wedge \left(d' > 2 \leftrightarrow \overline{d} > 2\right) \wedge \left(c' > d' \leftrightarrow \overline{c} > \overline{d}\right) \wedge \neg s \wedge s'$

- Predicates $\mathbb{P}$

$(d = 1), (c \geq d),$
$(d > 2), (c > d)$

- Trace
  - $F_0 := Init$
  - $F_1 := \top$

Example

- System with 2 state vars *c* and *d*
  - Init: $(d = 1) \wedge (c \geq d)$
  - Trans: $(c' = c + d) \wedge (d' = d + 1)$
  - Property: $(d > 2) \rightarrow (c > d)$
- Rec. block s
  - Check $AbsRelInd(F_0, T, \neg s, \mathbb{P})$: UNSAT
  - Generalize: $\{\neg(d > 2)\}$
  - Update $F_1 \coloneqq F_1 \wedge \neg(d > 2)$

- Predicates $\mathbb{P}$

  $(d = 1), (c \geq d),$

  $(d > 2), (c > d)$
- Trace
  - $F_0 \coloneqq Init$
  - $F_1 \coloneqq \top$

Example

- System with 2 state vars *c* and *d*
  - Init: $(d = 1) \wedge (c \geq d)$
  - Trans: $(c' = c + d) \wedge (d' = d + 1)$
  - Property: $(d > 2) \rightarrow (c > d)$

- Forward propagation

- Predicates $\mathbb{P}$

  $(d = 1), (c \geq d),$

  $(d > 2), (c > d)$

- Trace
  - $F_0 \coloneqq Init$
  - $F_1 \coloneqq \neg(d > 2)$
  - $F_2 \coloneqq \top$

Example

- System with 2 state vars *c* and *d*
  - Init: $(d = 1) \wedge (c \geq d)$
  - Trans: $(c' = c + d) \wedge (d' = d + 1)$
  - Property: $(d > 2) \rightarrow (c > d)$
- Get bad cube at 2
  - $s \coloneqq \{\neg(d = 1), \neg(c \geq d),$
    $(d > 2), \neg(c > d)\}$

- Predicates $\mathbb{P}$

  $(d = 1), (c \geq d),$

  $(d > 2), (c > d)$
- Trace
  - $F_0 \coloneqq Init$
  - $F_1 \coloneqq \neg(d > 2)$
  - $F_2 \coloneqq \top$

Example

- System with 2 state vars *c* and *d*
  - Init: $(d = 1) \wedge (c \geq d)$
  - Trans: $(c' = c + d) \wedge (d' = d + 1)$
  - Property: $(d > 2) \rightarrow (c > d)$

- Recursively block s
  - …
  - Update $F_1 \coloneqq F_1 \wedge (c \geq d)$
  - …
  - Update $F_2 \coloneqq F_2 \wedge (c \geq d) \vee \neg(d > 2)$

- Predicates $\mathbb{P}$

  $(d = 1), (c \geq d)$,

  $(d > 2), (c > d)$

- Trace
  - $F_0 \coloneqq Init$
  - $F_1 \coloneqq \neg(d > 2)$
  - $F_2 \coloneqq \top$

Example

- System with 2 state vars *c* and *d*
  - Init: $(d = 1) \wedge (c \geq d)$
  - Trans: $(c' = c + d) \wedge (d' = d + 1)$
  - Property: $(d > 2) \rightarrow (c > d)$
- Forward propagation

- Predicates $\mathbb{P}$

  $(d = 1), (c \geq d),$

  $(d > 2), (c > d)$
- Trace
  - $F_0 \coloneqq Init$
  - $F_1 \coloneqq \neg(d > 2) \wedge (c \geq d) \wedge F_2$
  - $F_2 \coloneqq (\mathrm{c} > \mathrm{d}) \vee \neg(d > 2)$
  - $F_3 \coloneqq \top$

Example

- System with 2 state vars *c* and *d*
  - Init: $(d = 1) \wedge (c \geq d)$
  - Trans: $(c' = c + d) \wedge (d' = d + 1)$
  - Property: $(d > 2) \rightarrow (c > d)$
- Get cube at 3
  - $s \coloneqq \{\neg(d = 1), \neg(c \geq d),$
    $(d > 2), \neg(c > d)\}$

- Predicates $\mathbb{P}$

  $(d = 1), (c \geq d),$

  $(d > 2), (c > d)$
- Trace
  - $F_0 \coloneqq Init$
  - $F_1 \coloneqq \neg(d > 2) \wedge (c \geq d) \wedge F_2$
  - $F_2 \coloneqq (\mathrm{c} > \mathrm{d}) \vee \neg(d > 2)$
  - $F_3 \coloneqq \top$

Example

- System with 2 state vars *c* and *d*
  - Init: $(d = 1) \wedge (c \geq d)$
  - Trans: $(c' = c + d) \wedge (d' = d + 1)$
  - Property: $(d > 2) \rightarrow (c > d)$
- Recursively block s
  - $AbsRelInd$ is sat
  - SMT model:

$\mu \coloneqq \{c = 0, d = 2, c' = 0, d = 3, \overline{c} = 2, \overline{d} = 3\}$

  - Abstract predecessor:

$\{\neg(d > 2), \neg(c > d), \neg(d = 1), \neg(c \geq d)\}$

- Predicates $\mathbb{P}$

$(d = 1), (c \geq d),$

$(d > 2), (c > d)$

- Trace
  - $F_0 \coloneqq Init$
  - $F_1 \coloneqq \neg(d > 2) \wedge (c \geq d) \wedge F_2$
  - $F_2 \coloneqq (\mathrm{c} > \mathrm{d}) \vee \neg(d > 2)$
  - $F_3 \coloneqq \top$

Example

- System with 2 state vars *c* and *d*
  - Init: $(d = 1) \wedge (c \geq d)$
  - Trans: $(c' = c + d) \wedge (d' = d + 1)$
  - Property: $(d > 2) \rightarrow (c > d)$
- Recursively block c
  - …
  - Reached level 0, abstract cex:

$s_0 := \neg(d > 2), \neg(c > d), (d = 1), (c \geq d)$
$s_1 := \neg(d > 2), \neg(c > d), \neg(d = 1), (c \geq d)$
$s_2 := \neg(d > 2), \neg(c > d), \neg(d = 1), \neg(c \geq d)$
$s := \neg(d = 1), \neg(c \geq d), (d > 2), \neg(c > d)$

- Predicates $\mathbb{P}$

  $(d = 1), (c \geq d),$

  $(d > 2), (c > d)$
- Trace
  - $F_0 := Init$
  - $F_1 := \neg(d > 2) \wedge (c \geq d) \wedge F_2$
  - $F_2 := (\mathrm{c} > \mathrm{d}) \vee \neg(d > 2)$
  - $F_3 := \top$

Example

- System with 2 state vars *c* and *d*
  - Init: $(d = 1) \wedge (c \geq d)$
  - Trans: $(c' = c + d) \wedge (d' = d + 1)$
  - Property: $(d > 2) \rightarrow (c > d)$

- Check abstract counterexample
  $s_0(V_0) \wedge T(V_0, V_1) \wedge s_1(V_1) \wedge T(V_1, V_2) \wedge s_2(V_2)$
  $\wedge\, T(V_2, V_3) \wedge s(V_3)$

UNSAT

- Predicates $\mathbb{P}$

  $(d = 1), (c \geq d),$

  $(d > 2), (c > d)$

- Trace
  - $F_0 \coloneqq Init$
  - $F_1 \coloneqq \neg(d > 2) \wedge (c \geq d) \wedge F_2$
  - $F_2 \coloneqq (\mathrm{c} > \mathrm{d}) \vee \neg(d > 2)$
  - $F_3 \coloneqq \top$

Example

- System with 2 state vars *c* and *d*
  - Init: $(d = 1) \wedge (c \geq d)$
  - Trans: $(c' = c + d) \wedge (d' = d + 1)$
  - Property: $(d > 2) \rightarrow (c > d)$
- Check abstract counterexample
- Extract new predicates from sequence interpolants:
$$d \geq 2, d \geq 3$$
- Update $\mathbb{P}$

- Predicates $\mathbb{P}$

  $(d = 1), (c \geq d),$

  $(d > 2), (c > d),$

  $(d \geq 2), (d \geq 3)$
- Trace
  - $F_0 := Init$
  - $F_1 := \neg(d > 2) \wedge (c \geq d) \wedge F_2$
  - $F_2 := (\mathrm{c} > \mathrm{d}) \vee \neg(d > 2)$
  - $F_3 := \top$

Example

- System with 2 state vars *c* and *d*
  - Init: $(d = 1) \wedge (c \geq d)$
  - Trans: $(c' = c + d) \wedge (d' = d + 1)$
  - Property: $(d > 2) \rightarrow (c > d)$
- Update abstract Trans
- Resume IC3 from level 3

- Predicates $\mathbb{P}$

  $(d = 1), (c \geq d),$

  $(d > 2), (c > d),$

  $(d \geq 2), (d \geq 3)$

- Trace
  - $F_0 := Init$
  - $F_1 := \neg(d > 2) \wedge (c \geq d) \wedge F_2$
  - $F_2 := (\mathrm{c} > \mathrm{d}) \vee \neg(d > 2) \wedge F_3$
  - $F_3 := (d = 1) \vee (d \geq 2) \wedge \neg(c \geq d) \wedge F_4$
  - $F_4 := (c > d) \vee \neg(d > 2)$

Example

- System with 2 state vars *c* and *d*
  - Init: $(d = 1) \wedge (c \geq d)$
  - Trans: $(c' = c + d) \wedge (d' = d + 1)$
  - Property: $(d > 2) \rightarrow (c > d)$
- Update abstract Trans
- Resume IC3 from level 3
- …
- Forward propagation

$$F_2 \wedge \hat{T}_{\mathbb{P}} \vDash (c' \geq d') \vee \neg(d' \geq 2)$$

- Predicates $\mathbb{P}$

$$(d = 1), (c \geq d),$$
$$(d > 2), (c > d),$$
$$(d \geq 2), (d \geq 3)$$

- Trace
  - $F_0 \coloneqq Init$
  - $F_1 \coloneqq \neg(d > 2) \wedge (c \geq d) \wedge F_2$
  - $F_2 \coloneqq (\mathrm{c} > \mathrm{d}) \vee \neg(d > 2) \wedge F_3$
  - $F_3 \coloneqq (d = 1) \vee (d \geq 2) \wedge \neg(c \geq d) \wedge F_4$
  - $F_4 \coloneqq (c > d) \vee \neg(d > 2)$

Example

- System with 2 state vars *c* and *d*
  - Init: $(d = 1) \wedge (c \geq d)$
  - Trans: $(c' = c + d) \wedge (d' = d + 1)$
  - Property: $(d > 2) \rightarrow (c > d)$
- Update abstract Trans
- Resume IC3 from level 3
- …
- Forward propagation
  $$F_2 \wedge \hat{T}_{\mathbb{P}} \vDash (c' \geq d') \vee \neg(d' \geq 2)$$
- Fixpoint ⇒ Property is true

- Predicates $\mathbb{P}$
  $$(d = 1), (c \geq d),$$
  $$(d > 2), (c > d),$$
  $$(d \geq 2), (d \geq 3)$$
- Trace
  - $F_0 \coloneqq Init$
  - $F_1 \coloneqq \neg(d > 2) \wedge (c \geq d) \wedge F_2$
  - $F_2 \coloneqq F_3 \coloneqq (c \geq d) \vee \neg(d \geq 2) \wedge (d = 1) \vee (d \geq 2) \wedge \neg(c \geq d) \wedge F_4$
  - $F_4 \coloneqq (c > d) \vee \neg(d > 2)$

Infinite State Model-Checking

Liveness Checking

LTL from Finite to Infinite

- Use first-order predicates instead of propositions:
  - $G(x \geq a \wedge x \leq b)$
  - $GF(x = a) \wedge GF(x = b)$
- Predicates interpreted according to specific theory
- “next” variables to express changes/transitions:
  - $G(x' = x + 1)$
  - $G(a' - a \leq b)$
- BMC
  - Add encoding of lasso-shape and fairness
  - Sound for finding traces, but not complete
  - The only counterexaple may be not lasso-shape
- K-liveness
  - No change
  - Sound to prove properties, but not complete
  - Property may hold, but fairness can be visited an unbounded number of times

Liveness to Safety for Infinite States

- Unsound for infinite-state systems
  - Not all counterexamples are lasso-shaped

$$I(S) \stackrel{\text{def}}{=} (x = 0) \qquad T(S) \stackrel{\text{def}}{=} (x' = x + 1) \qquad \varphi \stackrel{\text{def}}{=} \mathbf{FG}(x < 5)$$

- Liveness to safety with Implicit Abstraction
  - Apply the l2s transformation to the abstract system
    - Save the values of the predicates instead of the concrete state
  - Do it on-the-fly, tightly integrating l2s with IC3
  - Sound but incomplete
    - When abstract loop found, simulate in the concrete and refine
    - Might still diverge during refinement
      - Intrinsic limitation of state predicate abstraction

Wrap-up

Lecture Summary

- Overview of SAT-based model checking techniques
- Details on IC3, as currently the prominent algorithm
- Liveness reduced to safety
- Lifting SAT-based MC to SMT
  - For invariant checking
    - Easy for BMC and k-induction
    - Predicate abstraction to reduce to finite-state MC
    - Implicit abstraction to avoid explicit computation of abstract state space
    - Implicit abstraction to lift IC3 to SMT
  - For liveness
    - BMC and K-liveness sound but not complete
    - Liveness2safety on abstract state space

Not covered

- Other MC approaches: BDD-Based, Interpolation, …
- Other Properties: CTL, PSL, termination, epistemic, …
- Other kind of systems
  - Continuous-time/hybrid systems
  - Probabilistic Systems
  - Software (control-flow graphs)
  - …

Next lecture

L1 Functional Verification → Model-Checking

L2

- Safety Assessment → Model-Based Safety-Assessment
- Hierarchical Decomposition → Contract-Based Design

→ Contract-Based Safety-Assessment

Readings

A list of suggested readings on the topics of the course. The list is not meant to be complete.

- Model checking:
  - Edmund M. Clarke, Orna Grumberg, Doron A. Peled: Model Checking. The MIT Press, 1999
  - Kenneth L. McMillan: Symbolic Model Checking. Kluwer, 1993
  - Christel Baier, Joost-Pieter Katoen: Principles of Model Checking. The MIT Press, 2008
- Bounded Model Checking:
  - Armin Biere, Alessandro Cimatti, Edmund M. Clarke, Ofer Strichman, Yunshan Zhu: Bounded model checking. Advances in Computers 58: 117-148 (2003)

Readings

- K-induction:
  - Mary Sheeran, Satnam Singh, Gunnar Stålmarck: Checking Safety Properties Using Induction and a SAT-Solver. FMCAD 2000: 108-125
  - Niklas Eén, Niklas Sörensson: Temporal induction by incremental SAT solving. Electr. Notes Theor. Comput. Sci. 89(4): 543-560 (2003)
- IC3 for Finite-State Transition Systems:
  - Aaron R. Bradley: SAT-Based Model Checking without Unrolling. VMCAI 2011: 70-87
  - Fabio Somenzi, Aaron R. Bradley: IC3: where monolithic and incremental meet. FMCAD 2011: 3-8
  - Aaron R. Bradley: Understanding IC3. SAT 2012: 1-14
  - Krystof Hoder, Nikolaj Bjørner: Generalized Property Directed Reachability. SAT 2012: 157-171

Readings

- LTL Model Checking:
  - Amir Pnueli: The Temporal Logic of Programs. FOCS 1977: 46-57
  - Moshe Y. Vardi: An Automata-Theoretic Approach to Linear Temporal Logic. Banff Higher Order Workshop 1995: 238-266
  - Edmund M. Clarke, Orna Grumberg, Kiyoharu Hamaguchi: Another Look at LTL Model Checking. Formal Methods in System Design 10(1): 47-71 (1997)
- Liveness to safety:
  - Armin Biere, Cyrille Artho, Viktor Schuppan: Liveness Checking as Safety Checking. Electr. Notes Theor. Comput. Sci. 66(2): 160-177 (2002)
  - Yi Fang, Kenneth L. McMillan, Amir Pnueli, Lenore D. Zuck: Liveness by Invisible Invariants. FORTE 2006: 356-371
  - Koen Claessen, Niklas Sörensson: A liveness checking algorithm that counts. FMCAD 2012: 52-59

Readings

- K-Induction for Infinite-State Systems:
  - Leonardo Mendonça de Moura, Harald Rueß, Maria Sorea: Bounded Model Checking and Induction: From Refutation to Verification (Extended Abstract, Category A). CAV 2003: 14-26
  - Temesghen Kahsai, Cesare Tinelli: PKind: A parallel k-induction based model checker. PDMC 2011: 55-62
  - Alessandro Cimatti, Sergio Mover, Alessandro Cimatti: SMT-based scenario verification for hybrid systems. Formal Methods in System Design 42(1): 46-66 (2013)
  - Jonathan Laurent, Alwyn Goodloe, Lee Pike: Assuring the Guardians. RV 2015: 87-101

Readings

- Interpolation-based Model Checking:
  - Kenneth L. McMillan: Interpolation and SAT-Based Model Checking. CAV 2003: 1-13
  - Kenneth L. McMillan: Applications of Craig Interpolants in Model Checking. TACAS 2005: 1-12
  - Kenneth L. McMillan: Lazy Abstraction with Interpolants. CAV 2006: 123-136
- Liveness to Safety for Infinite-State Systems:
  - Viktor Schuppan, Armin Biere: Liveness Checking as Safety Checking for Infinite State Spaces. Electr. Notes Theor. Comput. Sci. 149(1): 79-96 (2006)
  - Andreas Podelski, Andrey Rybalchenko: Transition predicate abstraction and fair termination. ACM Trans. Program. Lang. Syst. 29(3) (2007)
  - Alessandro Cimatti, Alberto Griggio, Sergio Mover, Alessandro Cimatti: Verifying LTL Properties of Hybrid Systems with K-Liveness. CAV 2014: 424-440

Readings

- Implicit Abstraction:
  - Stefano Tonetta: Abstract Model Checking without Computing the Abstraction. FM 2009: 89-105
- IC3 for Infinite-State Systems:
  - Alessandro Cimatti, Alberto Griggio: Software Model Checking via IC3. CAV 2012: 277-293
  - Alessandro Cimatti, Alberto Griggio, Sergio Mover, Alessandro Cimatti: IC3 Modulo Theories via Implicit Predicate Abstraction. TACAS 2014: 46-61
  - Johannes Birgmeier, Aaron R. Bradley, Georg Weissenbacher: Counterexample to Induction-Guided Abstraction-Refinement (CTIGAR). CAV 2014: 831-848
  - Yakir Vizel, Arie Gurfinkel: Interpolating Property Directed Reachability. CAV 2014: 260-276
  - Nikolaj Bjørner, Arie Gurfinkel: Property Directed Polyhedral Abstraction. VMCAI 2015: 263-281