Hardware Verification 2IMF20

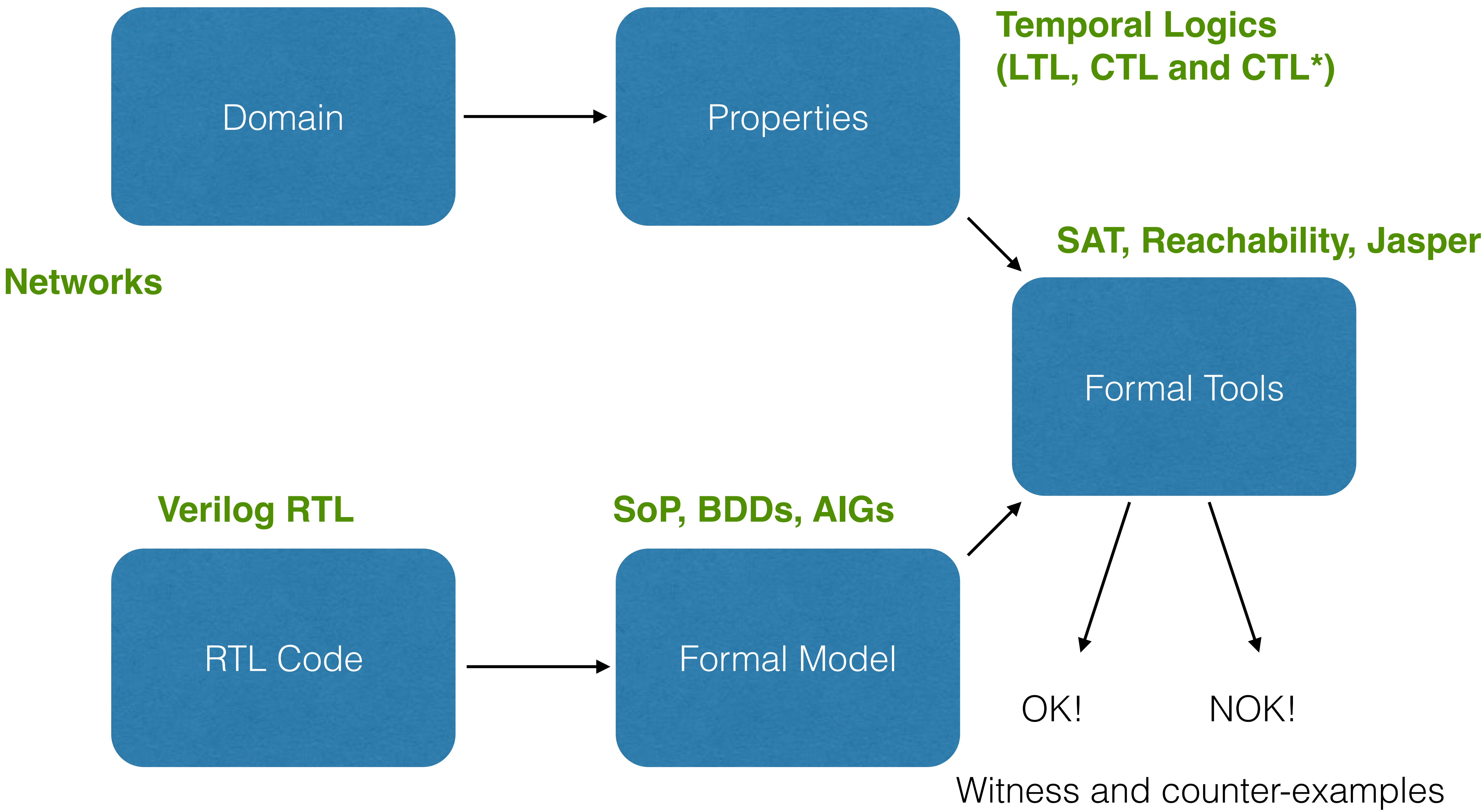
Julien Schmaltz

Lecture 05:
Symbolic CTL model checking
Bounded model checking with SAT

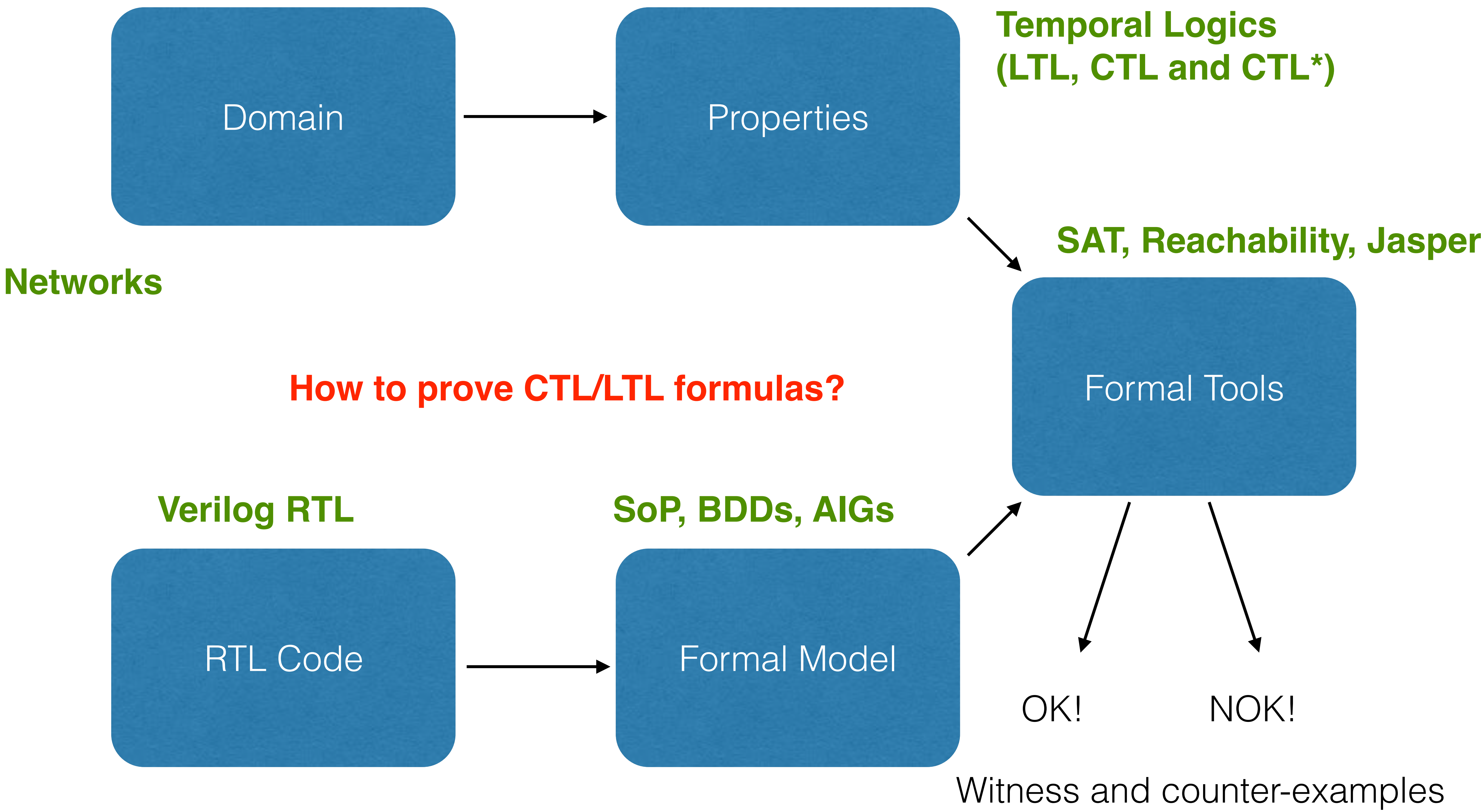
Technische Universiteit **Eindhoven** University of Technology

Where innovation starts

Course content - So far



Course content - Today



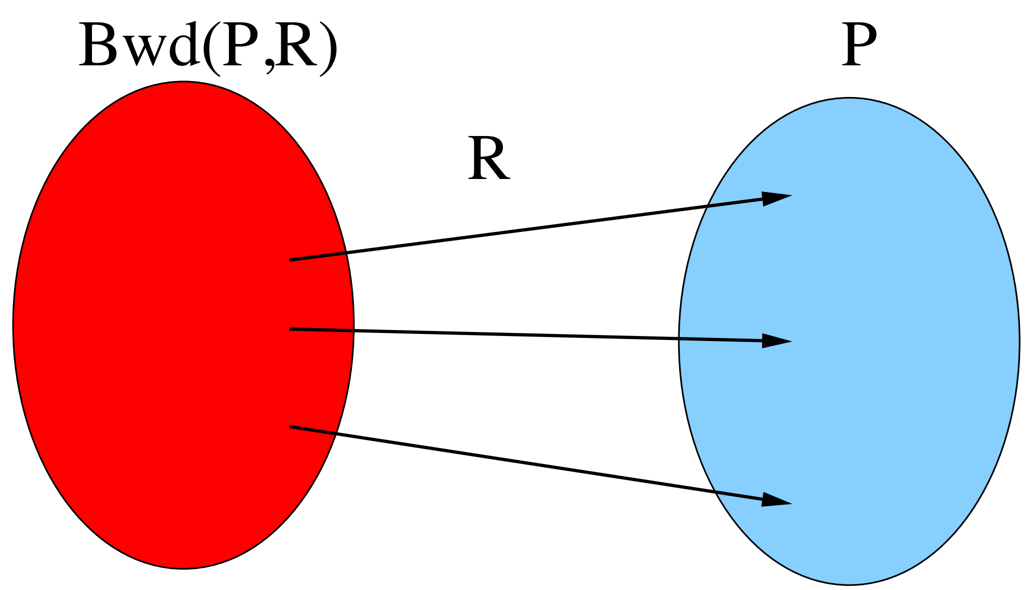
Two techniques

» Symbolic model checking with BDDs

» Bounded model checking with SAT

Symbolic model checking with BDD

Backward image



$$\text{Bwd}(P, R) = \{ \overline{v} | \exists \overline{v}' : \overline{v}' \in P \wedge (\overline{v}, \overline{v}') \in R \}$$

can be also written as Boolean function:

$$f(\overline{v}) = \exists \overline{v}' : (P(\overline{v}') \wedge T(\overline{v}, \overline{v}')$$

This is **EX**P.

Fixpoints

- Assume a finite set S
  - The least fixpoint $\mu Y.\tau(Y)$ is the limit L of

$$\mathit{false} \subseteq \tau(\mathit{false}) \subseteq \tau(\tau(\mathit{false})) \subseteq L$$

  - The greatest fixpoint $\nu Y.\tau(Y)$ is the limit L of

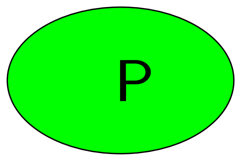
$$\mathit{true} \supseteq \tau(\mathit{true}) \supseteq \tau(\tau(\mathit{true})) \supseteq L$$

Note: S is finite, so convergence is finite

EFp as fixpoint

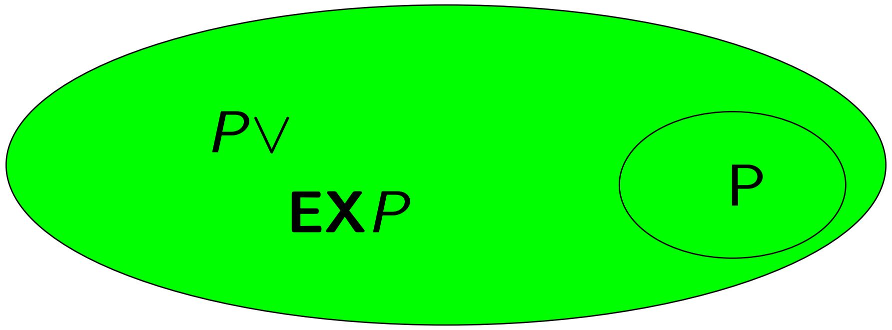
EFp is true if p is true now or if **EF**p holds in the previous step

It is the limit of the increasing series:



EFp as fixpoint

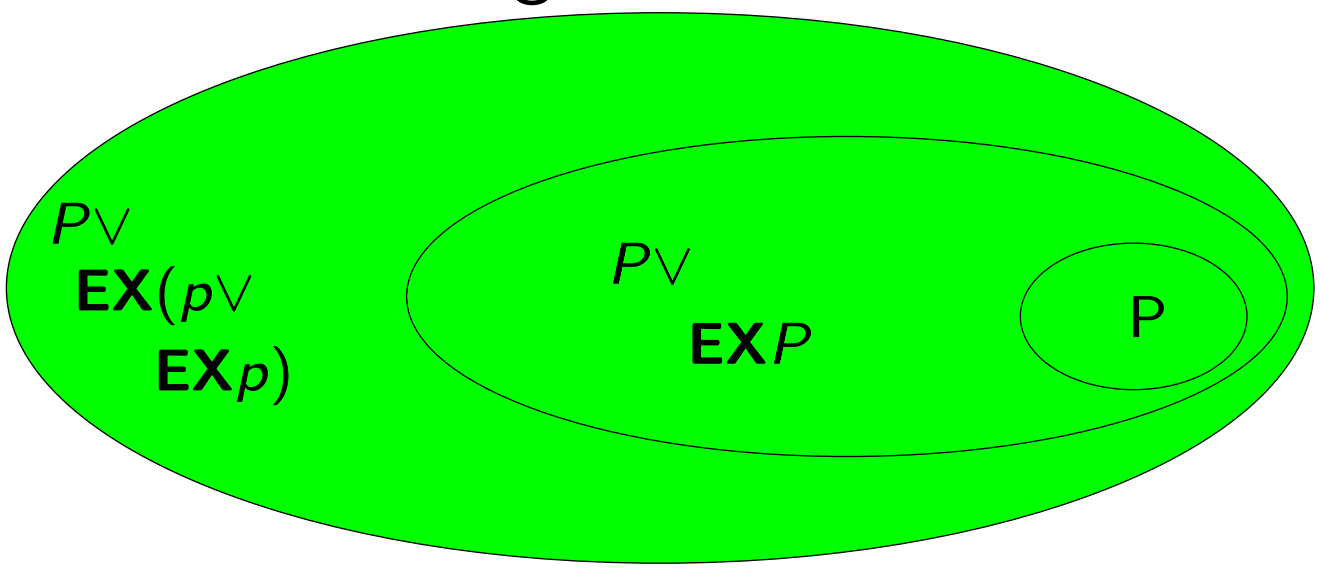
EFp is true if p is true now or if **EF**p holds in the previous step
It is the limit of the increasing series:



EFp as fixpoint

EFp is true if p is true now or if **EF**p holds in the previous step

It is the limit of the increasing series:



EU

Key properties of $\mathbf{EF}p$ (here seen as $\mathbf{E}(\top \mathbf{U} p)$)

- Either p now or somewhere in the future $\mathbf{EF}p$:
  $\mathbf{EF}p = p \vee \mathbf{EXEF}p$

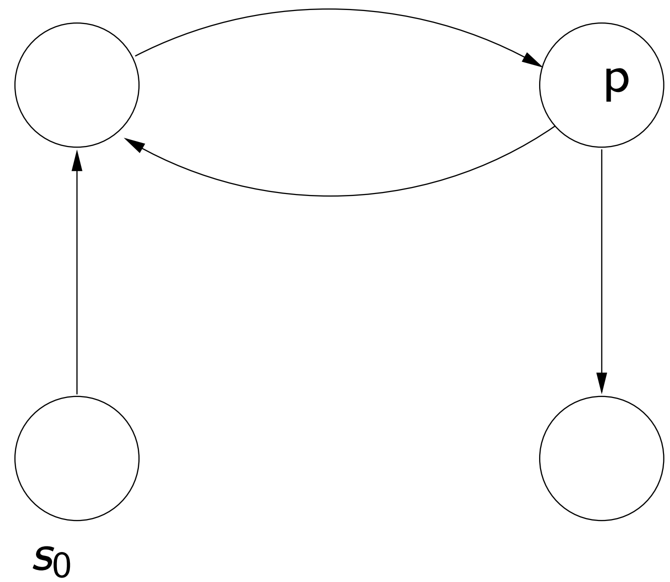
We write $\mathbf{EF}p = \mathbf{Lfp}S.p \vee \mathbf{EX}S$

How to compute $\mathbf{EF}p$:

$$\begin{array}{rcl} U_0 & = & \mathit{false} \\ U_1 & = & p \vee \mathbf{EX}U_0 \\ U_2 & = & p \vee \mathbf{EX}U_1 \\ U_3 & = & p \vee \mathbf{EX}U_2 \\ \ldots & & \end{array}$$

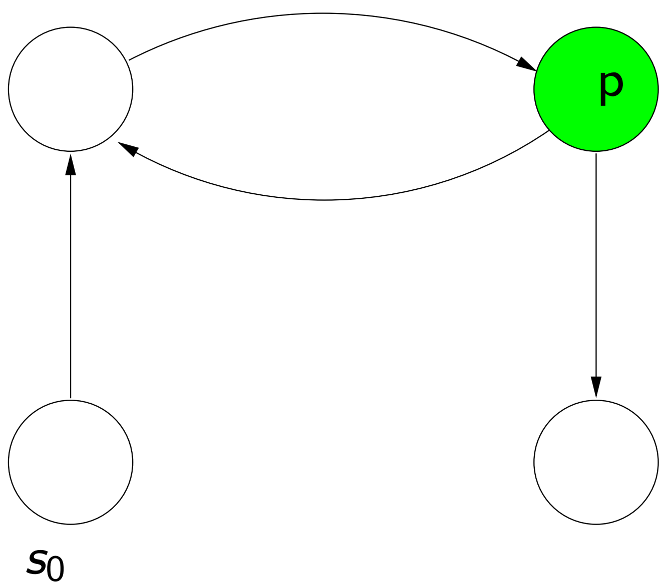
Stop when $U_{i+1} = U_i$

Check **EF**p example



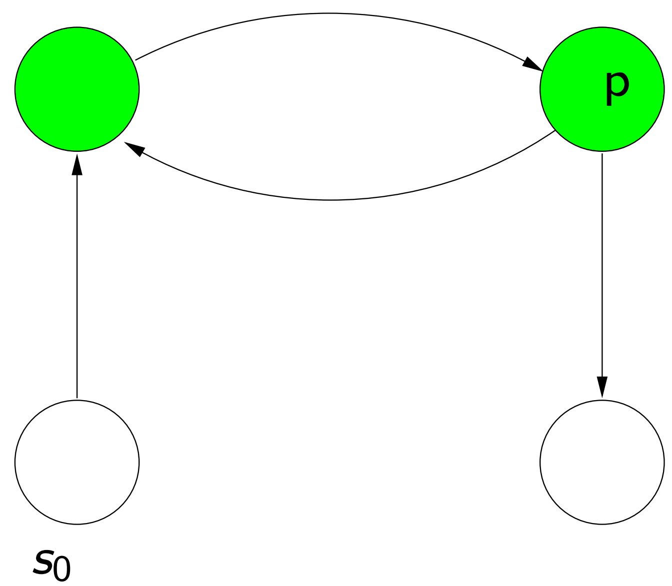
$$U_0 = \emptyset$$

Check **EF**p example



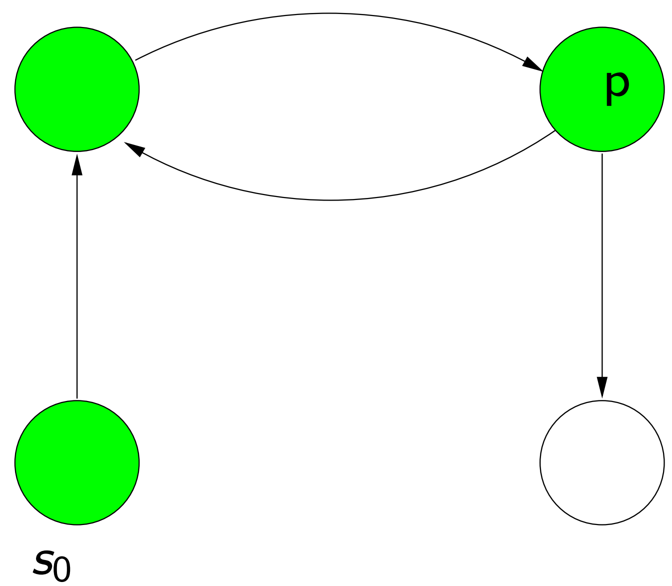
$$U_1 = p \vee \mathbf{EX}\, U_0$$

Check **EF**p example



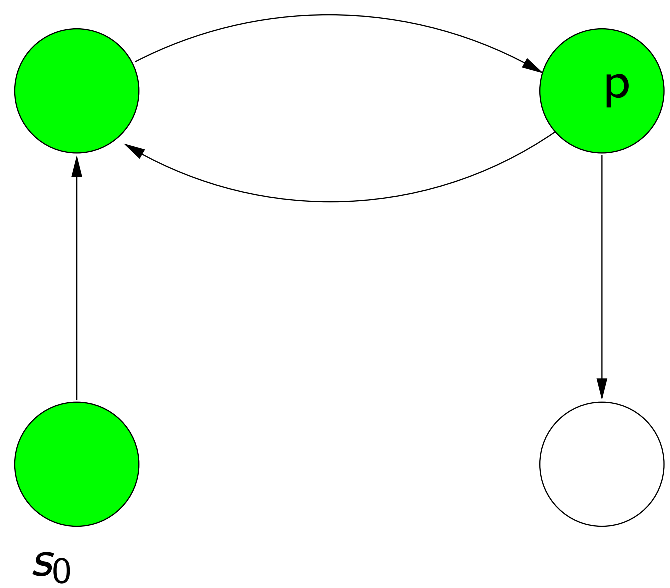
$$U_2 = p \lor \mathbf{EX}\, U_1$$

Check **EF**p example



$$U_3 = p \lor \mathbf{EX}\, U_2$$

Check **EF**p example

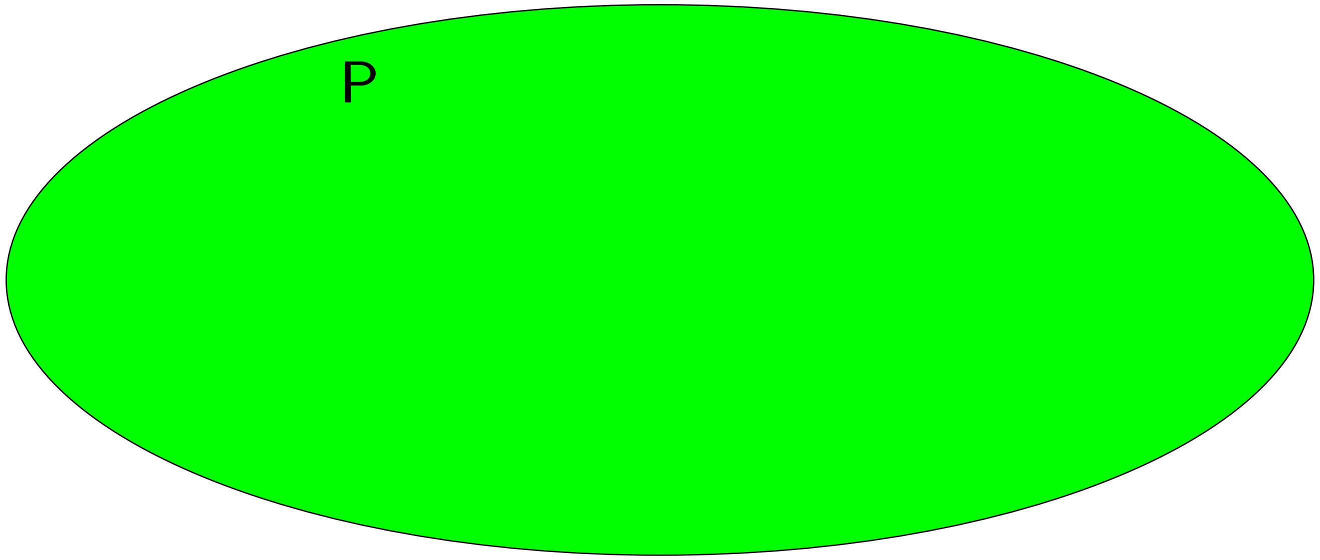


$$U_4 = U_3$$

EGp as fixpoint

EGp is true if p is true now and if **EG**p holds in the previous step

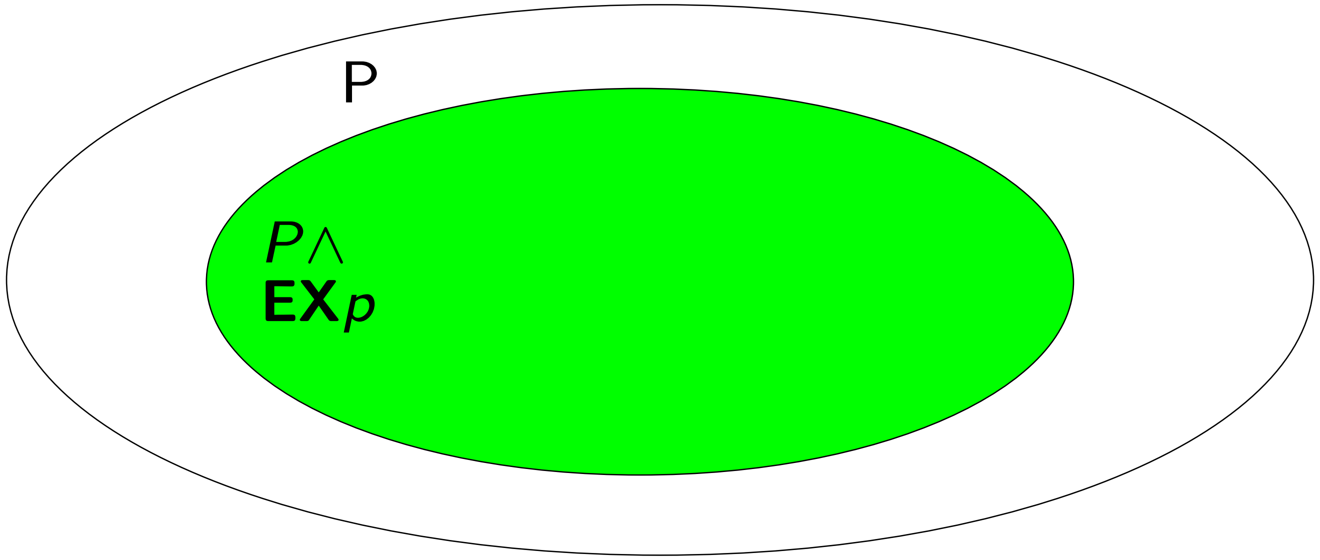
It is the limit of the decreasing series:



EGp as fixpoint

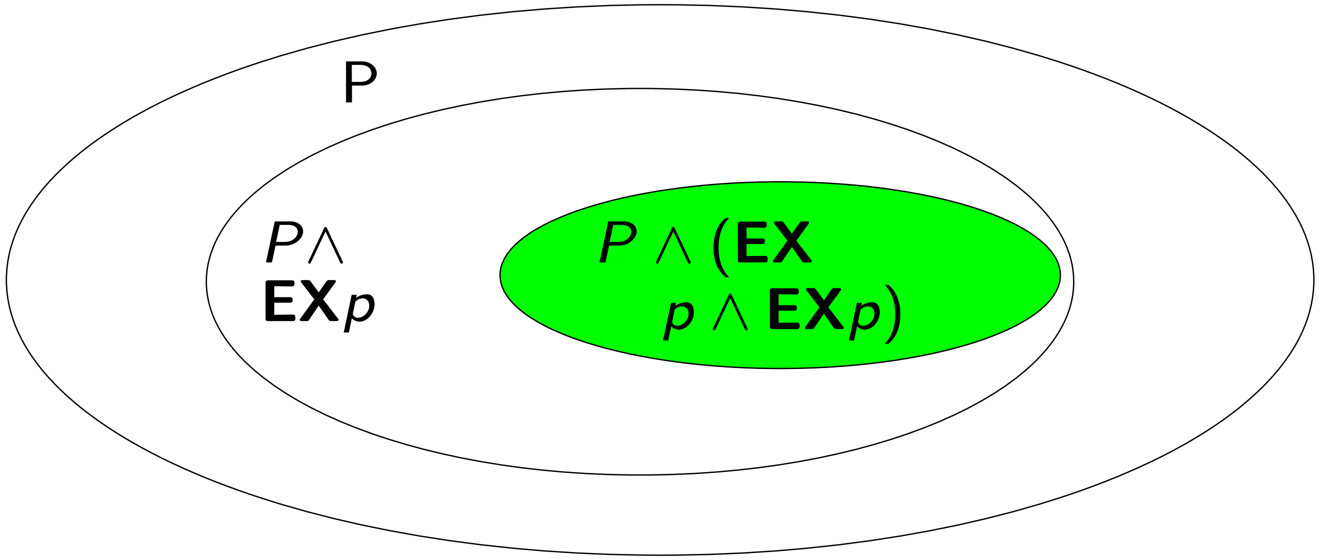
EGp is true if p is true now and if **EG**p holds in the previous step

It is the limit of the decreasing series:



EGp as fixpoint

EGp is true if p is true now and if **EG**p holds in the previous step
It is the limit of the decreasing series:



EG as Greatest Fixpoint

Key properties of $\mathbf{EG}p$

- $\mathbf{EG}p$ if p holds now and somewhere in the future:
  $\mathbf{EG}p = p \wedge \mathbf{EXEG}p$

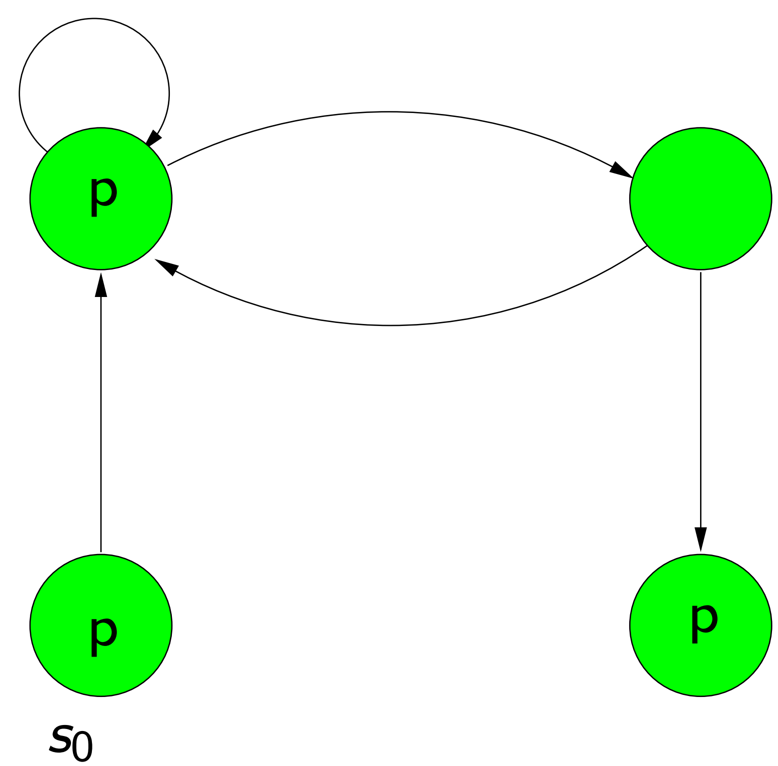
We write $\mathbf{EG}p = \mathbf{Gfp}S.p \wedge \mathbf{EX}S$

How to compute $\mathbf{EG}p$:

$$
\begin{array}{lcl}
U_0 & = & true \\
U_1 & = & p \wedge \mathbf{EX}U_0 \\
U_2 & = & p \wedge \mathbf{EX}U_1 \\
U_3 & = & p \wedge \mathbf{EX}U_2 \\
\ldots & &
\end{array}
$$

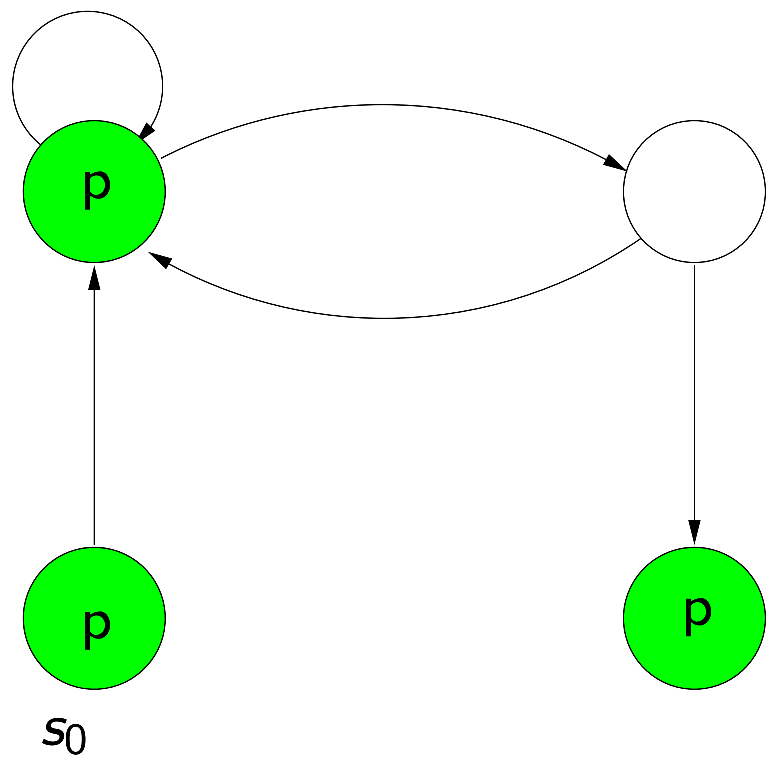
Stop when $U_{i+1} = U_i$

Check **EG**p example



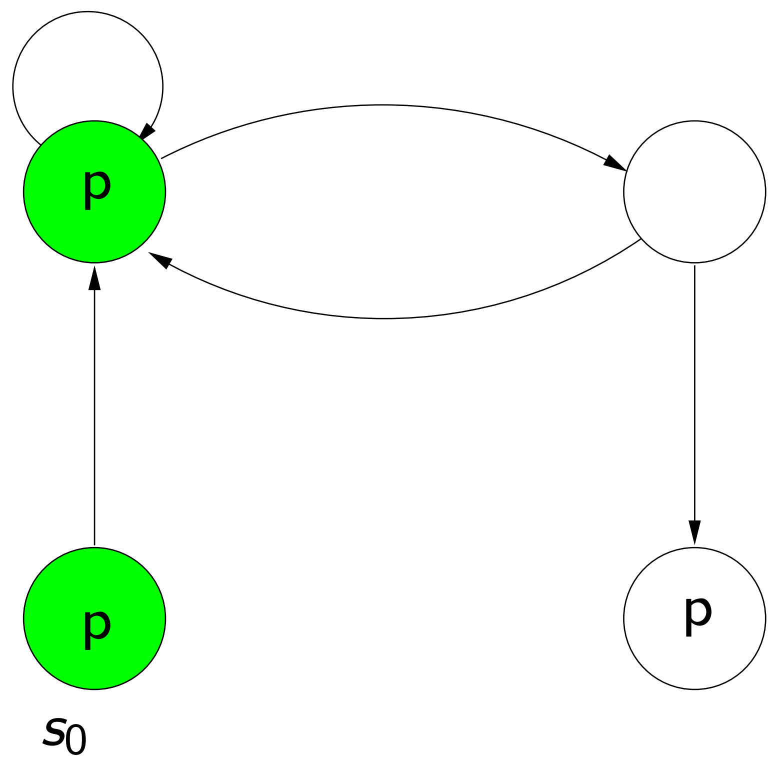
$$U_0 = S$$

Check **EG**p example



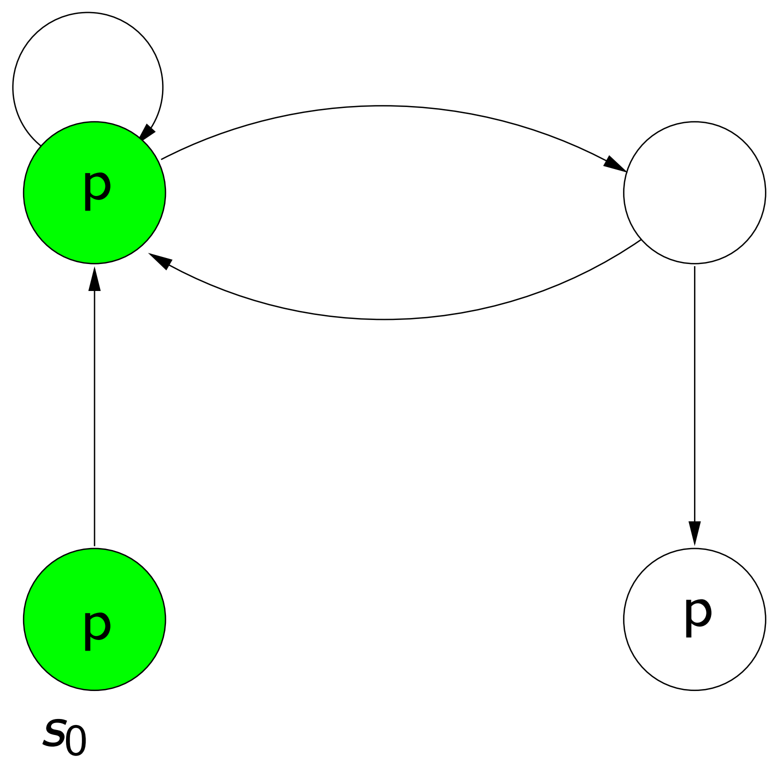
$$U_1 = p \wedge \mathbf{EX}\, U_0$$

Check **EG**p example



$$U_2 = p \wedge \mathbf{EX}\, U_1$$

Check **EG**p example



$$U_3 = U_2$$

Computing Fixpoints with BDDs

How to evaluate fixpoints using ROBDD's:

$$\mathbf{EF}p = \mathbf{Lfp}S.p \vee \mathbf{EX}S$$

Introduce state variables:

$$\mathbf{EF}p = \mathbf{Lfp}S.p(\overline{v}) \vee \exists \overline{v}'[T(\overline{v}, \overline{v}') \wedge S(\overline{v}')]$$

Now compute the sequence

$$U_0(\overline{v})\ U_1(\overline{v})\ U_2(\overline{v})\ \ldots$$

until convergence.
Convergence can be detected since the sets $U_i(\overline{v})$ are represented as ROBDD's

Other fixpoints

- $\mathbf{AF}p = \mu U.(p \vee \mathbf{AX}U)$
- $\mathbf{AG}p = \nu U.(p \wedge \mathbf{AX}U)$
- $\mathbf{E}(p\mathbf{U}q) = \mu U.(q \vee (p \wedge \mathbf{EX}U))$
- $\mathbf{A}(p\mathbf{U}q) = \mu U.(q \vee (q \wedge \mathbf{AX}U))$

SAT-based model checking

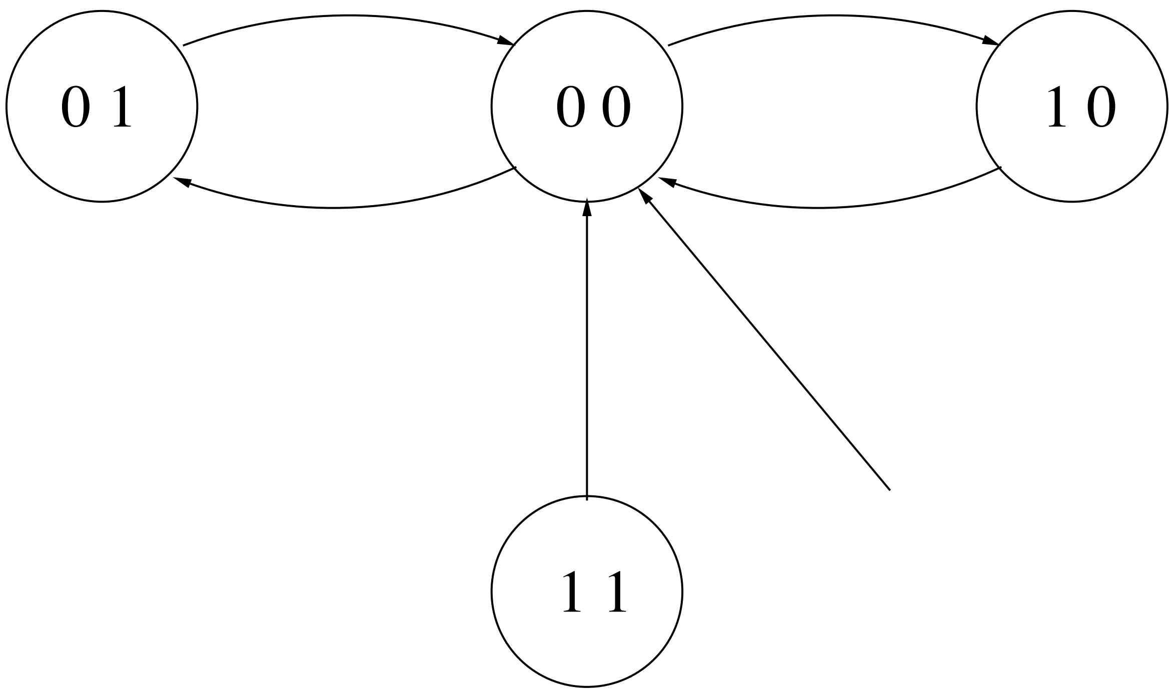
A simple mutual exclusion (SMUTE)

Consider 2 processes competing for a shared resource

```
process A
     forever
     A.pc = 0
     wait for B.pc = 0
     A.pc = 1
     access resource
     end forever
end process
```

```
process B
     forever
     B.pc = 0
     wait for A.pc = 0
     B.pc = 1
     access resource
     end forever
end process
```

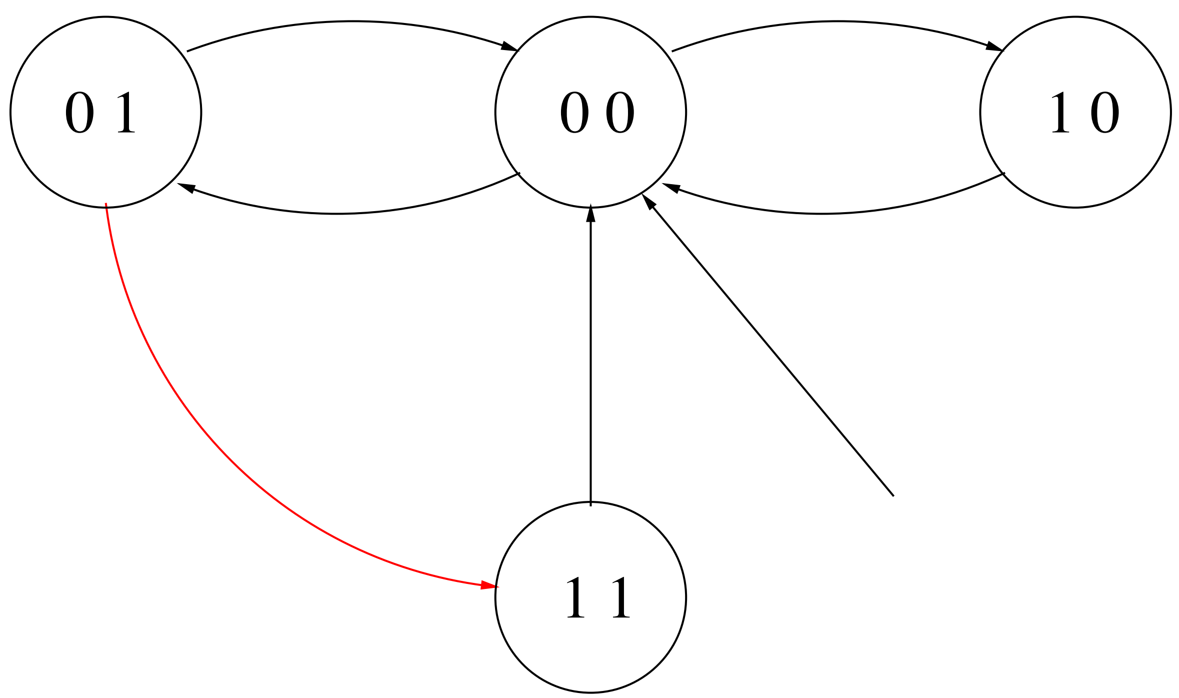
Kripke Structure for SMUTE



State space: $S = \{0, 1\}^2$
State vector: $s \in S = \{0, 1\}^2$
Transition relation $T \subseteq S^2$
An (initialized) path: 00, 01, 00, 10, 00, 01, ...

SMUTE is safe: never 2 processes access the resource simultaneously ($\mathbf{G}\neg(A.pc = 1 \land B.pc = 1)$)

Kripke Structure for unsafe SMUTE



Path: 00, 01, 11 is a counter-example to safety

$\mathbf{G}\neg(A.pc = 1 \wedge B.pc = 1)$ is false

$\mathbf{F}(A.pc = 1 \wedge B.pc = 1)$ is true

Kripke Structures

- A Kripke structure M is a quadruple $M = \langle S, I, T, L \rangle$
  - S is a set of states, and I a set of initial states
  - T is the transition relation
  - L is the labeling function, $L(s) =$ atomic propositions true in s
- A path π is an infinite sequence of states $s_0, s_1, s_2, \ldots$
- $\pi_i = (s_i, s_{i+1}, \ldots)$ denotes suffix starting at position i
- $M \models f$ means that M satisfies f (later restricted to LTL)

Limitations of Model Checking: why bounded ?

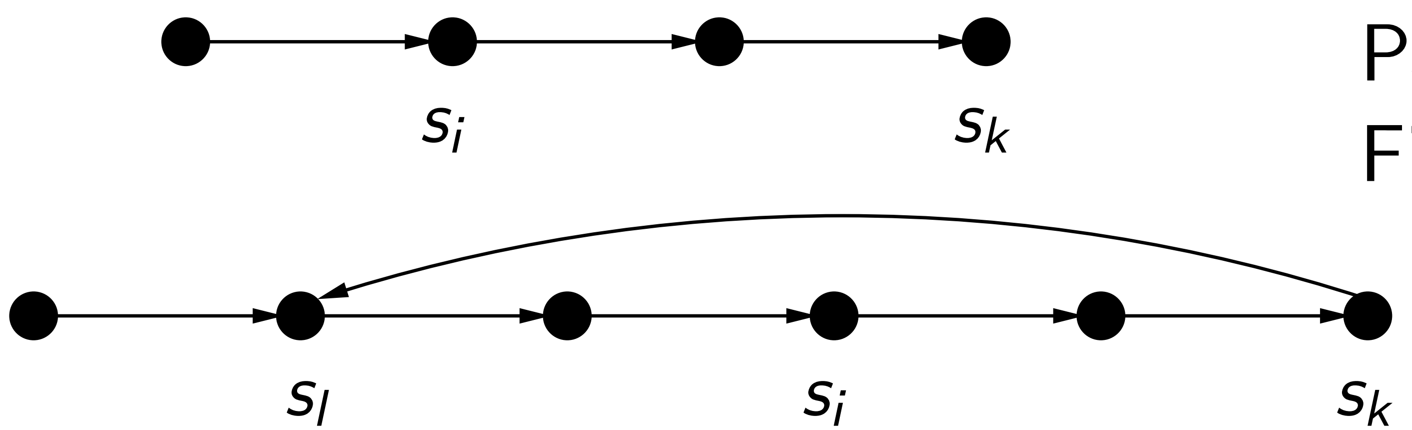
- Model checking suffers from state-space explosion
- Initial motivation for BMC: leverage advances in SAT solving
- Idea: restrict search to counter-examples with some length k

Basic Idea

- LTL formulas defined over all paths
  - finding counter-examples = exists a contradicting trace
  - for instance, a *counter-example* to $\mathbf{G}p$? = *witness* for $\mathbf{F}\neg p$?
  - for commodity we use path-quantifiers
  - $M \models \mathbf{A}f \equiv M \models \neg(\mathbf{E}\neg f)$
  - for now on, we only look at the existential problem ($M \models \mathbf{E}f$)
- Finite paths may represent infinite behaviors
  - paths with loops

Two cases for a bounded path

Idea: *finite paths may say something about infinite behaviors*



Path without a back loop
Finite behavior up to s_k

Path with a back loop
Infinite behavior

Definition ((k,l)-loop)

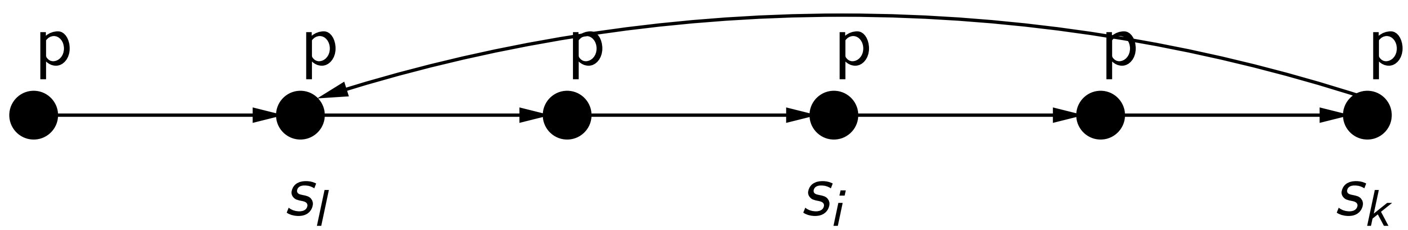
For $l \leq k$ we call a path π a *(k,l)-loop* if $T(\pi(k), \pi(l))$ and $\pi = u \cdot v^\omega$ with $u = (\pi(0), \ldots, \pi(l-1))$ and $v = (\pi(l), \ldots, \pi(k))$. We call π a *k-loop* if there exists $k \geq l \geq 0$ for wich π is a *(k,l)-loop*.

Bounded path and witnesses

- Can a path with no loop be a witness for $\mathbf{G}p$?
  - Justify.
- Can a path with no loop be a witness of $\mathbf{F}p$?
  - Justify.

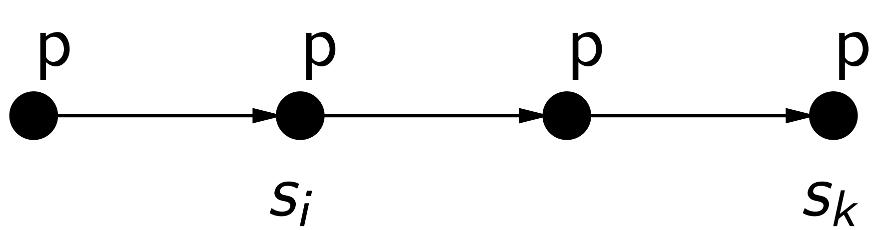
Witnesses for $\mathbf{G}p$: example

Let us consider the following *k-loop*:



- Thus, it is a witness for $\mathbf{G}p$

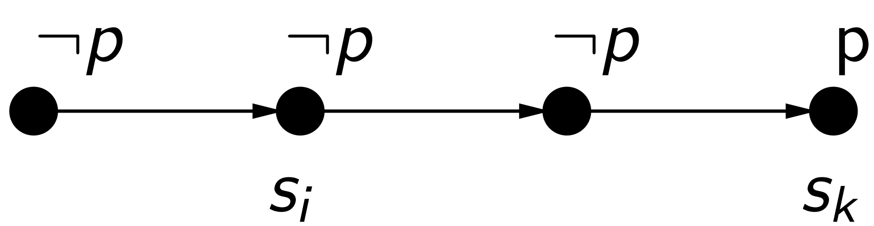
Let us consider the following path:



- This cannot be a witness for $\mathbf{G}p$, as there might be states after s_k that does not satisfy p.

Witnesses for $\mathbf{F}p$: example

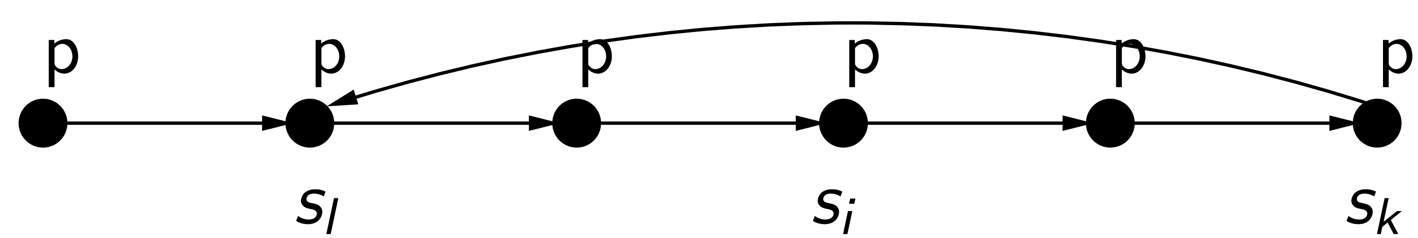
Let us consider the following path:



- Thus, it is a witness for $\mathbf{F}p$

Bounded semantics for a loop

- Prefix with loops, thus infinite behaviors preserved
  - All the information about the infinite behavior is contained in the bounded prefix



- Maintain the original LTL semantics

Definition

Let $k \geq 0$ and π be a *k-loop*. Then an LTL formula f is valid along the path π with bound k (in symbols $\pi \models_k f$) iff $\pi \models f$.

Bounded semantics without a loop: $\mathbf{F}f$

- Infinite behaviors unkown
- *Unbounded* semantics: f holds on some suffix of π

$$\pi \models \mathbf{F}f \qquad \text{iff} \qquad \pi_i \models f \text{ for some } i \geq 0$$

- Bounded semantics: f holds before position k

$$\pi \models_k^i \mathbf{F}f \qquad \text{iff} \qquad \exists j, i \leq j \leq k . \pi \models_k^j f$$

where $\pi \models_k^i f$ reads "f holds in state s_i of path π of length k"

Bounded semantics without a loop: $\mathbf{G}f$

- Infinite behaviors unkown
- *Unbounded* semantics: f holds in all suffixes

$$\pi \models \mathbf{G}f \qquad \text{iff} \qquad \pi_i \models f \text{ for all } i \geq 0$$

- Bounded semantics: $\mathbf{G}f$ is always false !
    - f might not hold after k

Consequence: the duality between $\mathbf{G}$ and $\mathbf{F}$ ($\neg\mathbf{F}f \equiv \mathbf{G}\neg f$) no longer holds in BMC !

Bounded semantics without a loop

Let $k \geq 0$ and π be a path that is *not* a k-loop. Then, an LTL formula f is valid along π with bound k (in symbols $\pi \models_k f$) iff $\pi \models_k^0 f$ where:

$$
\begin{array}{lll}
\pi \models_k^i p & \text{iff} & p \in L(\pi(i)) \\
\pi \models_k^i \neg p & \text{iff} & p \notin L(\pi(i)) \\
\pi \models_k^i f \wedge g & \text{iff} & \pi \models_k^i f \text{ and } \pi \models_k^i g \\
\pi \models_k^i f \vee g & \text{iff} & \pi \models_k^i f \text{ or } \pi \models_k^i g \\
\pi \models_k^i \mathbf{G} f & & \text{is always false} \\
\pi \models_k^i \mathbf{F} f & \text{iff} & \exists j, i \leq j \leq k . \pi_i \models_k^j f \\
\pi \models_k^i \mathbf{X} f & \text{iff} & i < k \text{ and } \pi \models_k^{i+1} f \\
\pi \models_k^i f \mathbf{U} g & \text{iff} & \exists j, i \leq j \leq k . \pi \models_k^j g \text{ and } \forall n, i \leq n < j \pi \models_k^n f \\
\pi \models_k^i f \mathbf{R} g & \text{iff} & \exists j, i \leq j \leq k . \pi \models_k^j f \text{ and } \forall n, i \leq n < j \pi \models_k^n g
\end{array}
$$

Bounded and unbounded semantics are equivalent

- Consider the existential model checking problem ($M \models \mathbf{E}f$)
- It can be reduced to an equivalent bounded model checking problem

Theorem

Let f be an LTL formula and M be a Kripke structure. Then:

$$M \models \mathbf{E}f \qquad \textit{iff} \qquad \exists k \geq 0 \textit{ with } M \models_k \mathbf{E}f$$

Bounded and unbounded semantics: Lemma 1

- If f is valid on a bounded path, then it is valid in the unbounded path
- Proof: easy – only based on the definition of bounded semantics (exercise)

Theorem

Let f be an LTL formula and π a path, then $\pi \models_k f \Rightarrow \pi \models f$

Bounded and unbounded semantics: Lemma 2

- If $\mathbf{E}f$ holds in the *unbounded* semantics, then there is a bounded path such that f is valid
- Proof: easy – Note: $M \models \mathbf{E}f \equiv \exists\pi : \pi \models f$ (exercise)

Theorem

Let f be an LTL formula and M be a Kripke structure. Then,

$$M \models \mathbf{E}f \qquad \Rightarrow \qquad \exists k \geq 0 \textit{ with } M \models_k \mathbf{E}f$$

Bounded semantics: Summary

- We have given a bounded semantics for the existential model checking problem
- We have shown that this semantics is equivalent to the unbounded one for a sufficiently large bound
- Coming next: translation from BMC to SAT

Introduction

- Reducing BMC to SAT enables the use of efficient SAT solvers to perform model checking
- The goal of the game:
  - Take an LTL formula f, a Kripke structure M and a bound k
  - Build a propositional formula $[[M, f]]_k$ "equivalent" to f
  - *Formula* $[[M, f]]_k$ *is SAT iff* M *has a path along which* f *is valid*
- This formula has two parts:
  - Unfolding the transition relation up to depth k (all valid paths of length k)
  - Constraints paths to be witnesses for formula f
  - The latter considers the loop and the no loop cases

Unfolding the transition relation

- Formula $[[M]]_k$ represents the k-unfolding of the transition relation T
- It represents all valid paths of length k
- A valid path:
  - First state is initial
  - All successors are obtained using T

Definition

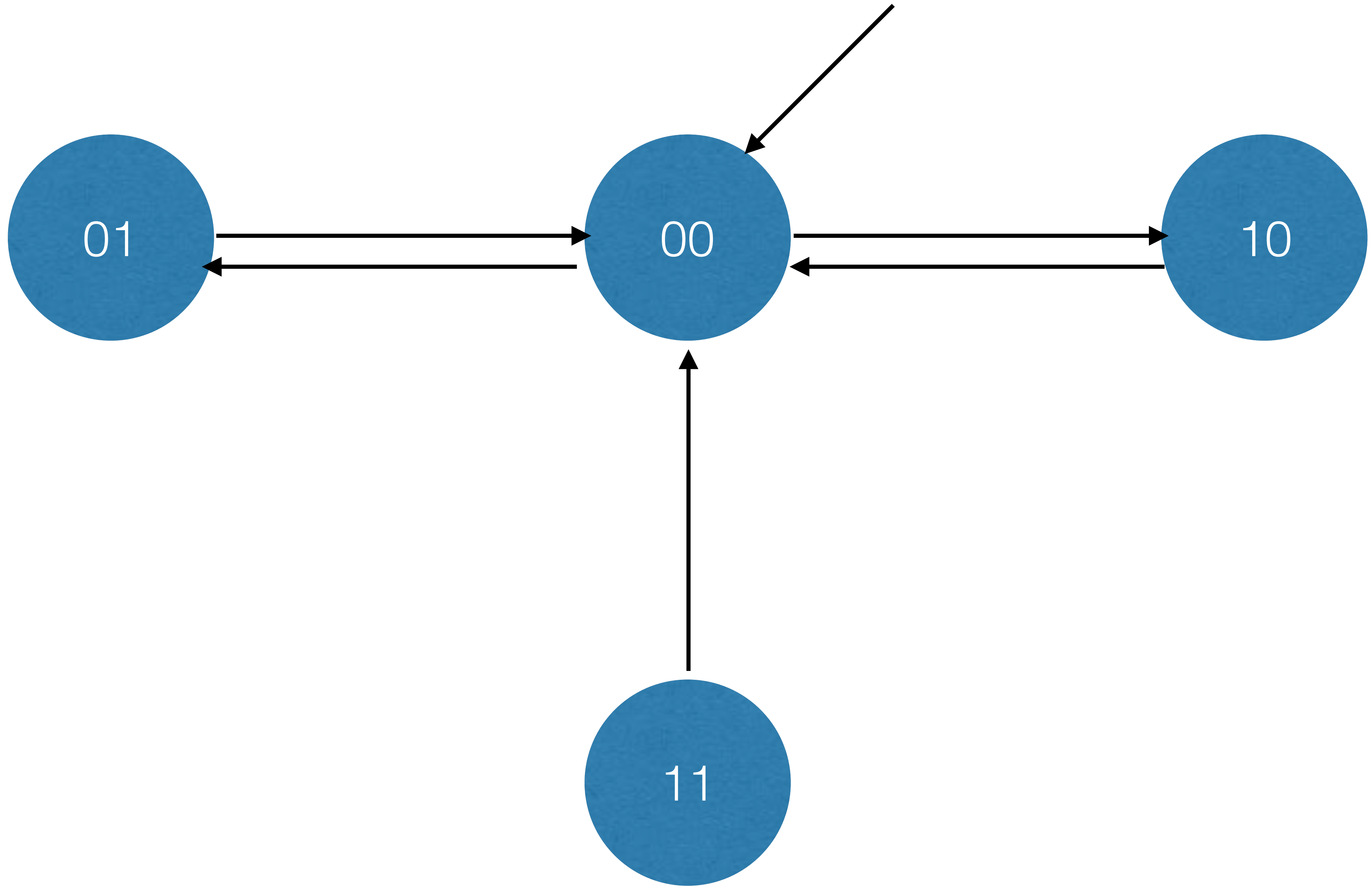
For a Kripke structure M and $k \geq 0$

$$[[M]]_k := I(s_0) \wedge \bigwedge_{i=0}^{k-1} T(s_i, s_{i+1})$$

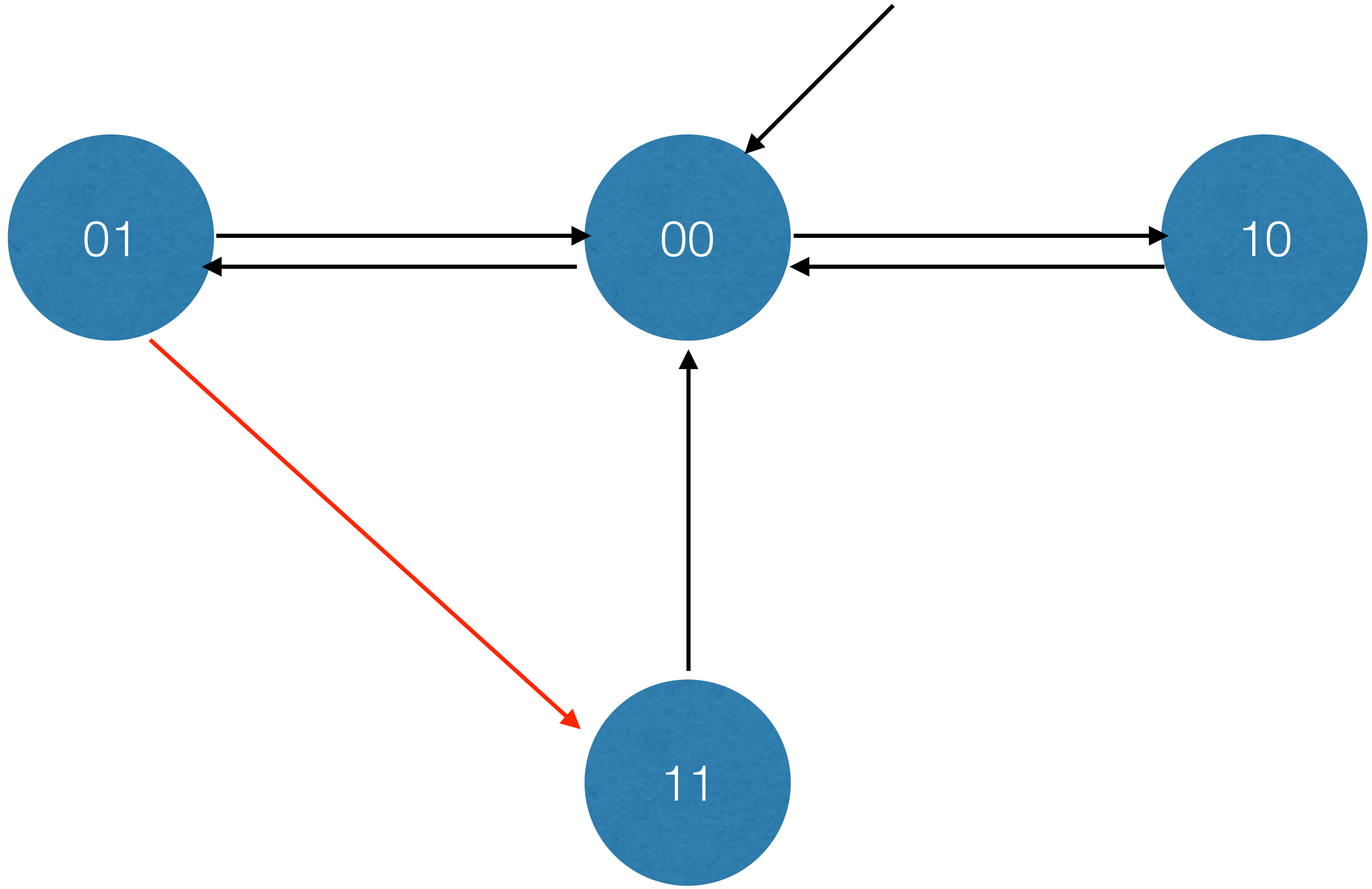
Unfolding the transition relation: Example

Example

show the unfolding for the mutual exclusion example



$$
T(a, b, a', b') = \begin{array}{rc} & \neg a \wedge \neg b \wedge \neg a' \wedge b' \\ \vee & \neg a \wedge \neg b \wedge a' \wedge \neg b' \\ \vee & \neg a \wedge b \wedge \neg a' \wedge \neg b' \\ \vee & a \wedge \neg b \wedge \neg a' \wedge \neg b' \\ \vee & a \wedge b \wedge \neg a' \wedge \neg b' \end{array}
$$



$$T(a,b,a',b') = \begin{array}{rc} & \neg a \wedge \neg b \wedge \neg a' \wedge b' \\ \vee & \neg a \wedge \neg b \wedge a' \wedge \neg b' \\ \vee & \neg a \wedge b \wedge \neg a' \wedge \neg b' \\ \vee & \neg a \wedge b \wedge a' \wedge b' \\ \vee & a \wedge \neg b \wedge \neg a' \wedge \neg b' \\ \vee & a \wedge b \wedge \neg a' \wedge \neg b' \end{array}$$

Our goal is to prove : $\mathbf{G}\,\neg(a \wedge b)$

With BMC, the idea is to prove that there is no witness path satisfying its negation. That is, we consider the following property:

$$\mathbf{F}\;(a \wedge b)$$

The objective is to create a SAT instance that is UNSAT iff no path exists that satisfies this property.

We will then be able to conclude that the initial property is true.

Let us unfold the transition relation.

Let assume k = 2 (2 iterations from the initial state.)
The initial state is defined as follows:

$$I(a,b) = \neg a \wedge \neg b$$

Initial state. Let a_0 and b_0 be the initial variables. $I(a_0, b_0) = \neg a_0 \wedge \neg b_0$

Iteration 1. Let a_1 and b_1 be the variables after 1 iteration.

$$T(a_0, b_0, a_1, b_1) = \begin{array}{cl} & \neg a_0 \wedge \neg b_0 \wedge \neg a_1 \wedge b_1 \\ \vee & \neg a_0 \wedge \neg b_0 \wedge a_1 \wedge \neg b_1 \\ \vee & \neg a_0 \wedge b_0 \wedge \neg a_1 \wedge \neg b_1 \\ \vee & a_0 \wedge \neg b_0 \wedge \neg a_1 \wedge \neg b_1 \\ \vee & a_0 \wedge b_0 \wedge \neg a_1 \wedge \neg b_1 \end{array}$$

Iteration 2. Let a_2 and b_2 be the variables after 2 iterations.

$$T(a_1, b_1, a_2, b_2) = \begin{array}{ll} & \neg a_1 \wedge \neg b_1 \wedge \neg a_2 \wedge b_2 \\ \vee & \neg a_1 \wedge \neg b_1 \wedge a_2 \wedge \neg b_2 \\ \vee & \neg a_1 \wedge b_1 \wedge \neg a_2 \wedge \neg b_2 \\ \vee & a_1 \wedge \neg b_1 \wedge \neg a_2 \wedge \neg b_2 \\ \vee & a_1 \wedge b_1 \wedge \neg a_2 \wedge \neg b_2 \end{array}$$

Complete unfolding produces:

$$I(a_0, b_0) \wedge T(a_0, b_0, a_1, b_1) \wedge T(a_1, b_1, a_2, b_2)$$

$$\begin{array}{ll} & \neg a_0 \wedge \neg b_0 \\ \wedge & ((\neg a_1 \wedge b_1) \vee (a_1 \wedge \neg b_1)) \\ \wedge & ((\neg a_1 \wedge b_1) \vee (a_1 \wedge \neg b_1)) \wedge (\neg a_2 \wedge \neg b_2) \end{array}$$

Translation: second component

- The second component restricts the paths to be witnesses for formula f
- The translation of an LTL formula f depends on the shape of paths
  - Paths with no loop
  - Paths with a loop

Loop condition and successor in a loop

- Loop condition: simply check if there is a transition from state s_k to a previous state
- Loop successor: increment by 1 except for the last state

Definition (Loop condition)

The loop condition L_k is true iff there exists a back loop from state s_k to a previous state or to itself: $L_k := \bigvee_{l=0}^{k} T(s_k, s_l)$

Definition (Successor in a loop)

Let k, l, i be such that $0 \leq i, l \leq k/$ Define $succ(i)$ of i in a (k,l)-loop as $succ(i) := i + 1$ for $i < k$ and $succ(i) = l$ for $i = k$.

Loop condition: Example

Example

Put here loop conditions for SMUTE

We need to compute the loop condition for k = 2 that is L_2

$$L_2 = T(a_2, b_2, a_0, b_0) \vee T(a_2, b_2, a_1, b_1) \vee T(a_2, b_2, a_2, b_2)$$

$$T(a_2, b_2, a_2, b_2) = \text{false}$$

$$T(a_2, b_2, a_0, b_0) = a_2 \wedge b_2 \wedge \neg a_0 \wedge \neg b_0$$

$$T(a_2, b_2, a_1, b_1) = \neg a_2 \wedge \neg b_2 \wedge (\neg a_1 \wedge b_1 \vee a_1 \wedge \neg b_1)$$

Translation for a loop

- Given: LTL formula f and (k,l)-loops paths π
- Recursive translation over the subterms of f and states in π
- Introduce intermediate formulae of the form ${}_l[[\cdot]]_k^i$
  - l start of the loop
  - k is the bound
  - i current position
- Translation rule for $\mathbf{G}f$:

$$ {}_l[[\mathbf{G}f]]_k^i := {}_l[[f]]_k^i \wedge {}_l[[\mathbf{G}f]]_k^{succ(i)} $$

- Translation rule for $\mathbf{F}f$:

$$ {}_l[[\mathbf{F}f]]_k^i := {}_l[[f]]_k^i \vee {}_l[[\mathbf{F}f]]_k^{succ(i)} $$

Translation for a loop: Example

Example

show translation for SMUTE and a loop

Translation for $\mathbf{F}\ (a \wedge b)$

For this transition relation, only one loop is possible, it is 2,1-loop.

So, we get:

$$a_2 \wedge b_2 \vee a_1 \wedge b_1$$

Translation without a loop

- Special case of the translation for a loop
- Extend paths and consider all properties beyond k false (base case)
- Inductive case: $\forall i \leq k$
- Translation for $\mathbf{G}f$:

$$[[\mathbf{G}f]]_k^i \equiv false$$

- Translation rule for $\mathbf{F}f$:

$$[[\mathbf{F}f]]_k^i := [[f]]_k^i \vee [[\mathbf{F}f]]_k^{i+1}$$

Translation to SAT: Example

Example

Show the translation for SMUTE and Gp

Translation for $\mathbf{F}\ (a \wedge b)$

Path has length 2, so starts with a0 and b0 and ends with a2 and b2.

$$a_0 \wedge b_0 \vee a_1 \wedge b_1 \vee a_2 \wedge b_2$$

Translation to SAT: Soundness and completeness

- General translation rule

$$[[M, f]]_k := [[M]]_k \wedge \left(\left(\neg L_k \wedge [[f]]_k^0\right) \vee \bigvee_{l=0}^{k} \left(T(s_k, s_l) \wedge {}_l[[f]]_k^0 \right) \right)$$

- Translation scheme is sound and complete w.r.t. the bounded semantics

Theorem

$[[M, f]]_k$ is satisfiable iff $M \models_k \mathbf{E} f$.

Translation to SAT: Final example

Example

show counter-example generation for Gp on the faulty SMUTE (recall the previous parts of the example)

The unfolding with k = 2 gives us

$$
\begin{aligned}
& \neg a_0 \wedge \neg b_0 \\
\wedge \quad & ((\neg a_1 \wedge b_1) \vee (a_1 \wedge \neg b_1)) \\
\wedge \quad & ((\neg a_1 \wedge b_1) \vee (a_1 \wedge \neg b_1)) \wedge (\neg a_2 \wedge \neg b_2)
\end{aligned}
$$

The only possible loop is a 2-1-loop. So, the part for path without loops gives us:

$$\neg T(a_2, b_2, a_1, b_1) \wedge (a_0 \wedge b_0 \vee a_1 \wedge b_1 \vee a_2 \wedge b_2))$$

The encoding for paths with a loop gives us:

$$T(a_2, b_2, a_1, b_1) \wedge (a_2 \wedge b_2 \vee a_1 \wedge b_1)$$

The complete SAT instance is on the next slide.

$$
\begin{array}{ll}
& \neg a_0 \wedge \neg b_0 \\
\wedge & ((\neg a_1 \wedge b_1) \vee (a_1 \wedge \neg b_1)) \\
\wedge & ((\neg a_1 \wedge b_1) \vee (a_1 \wedge \neg b_1)) \wedge (\neg a_2 \wedge \neg b_2)
\end{array}
$$

$$
\bigwedge \left(\begin{array}{c}
\neg T(a_2, b_2, a_1, b_1) \wedge (a_0 \wedge b_0 \vee a_1 \wedge b_1 \vee a_2 \wedge b_2) \\
\bigvee \\
T(a_2, b_2, a_1, b_1) \wedge (a_2 \wedge b_2 \vee a_1 \wedge b_1)
\end{array} \right)
$$

This is UNSAT.

Write the CNF for it? Nothing to write?

As an exercise. Try to do it for proving F (a&b).

Completeness

- Typical application of BMC = increment k until counter-example found
- If a witness exists ($M \models_k \mathbf{E}f$) this procedure terminates
- But, if there is no witness ($M \not\models_k \mathbf{E}f$) then does not terminates
- Incomplete BMC good for "bug hunting"
- Complete BMC needed to prove "bug-less"
  - Compositional proofs
  - Proof broken for one flawed proof (one missed counter-example)
- In the next slides: one technique for completeness
  - Completeness threshold

Completeness threshold

- For every finite system M, property p, there exists a number $\mathcal{CT}$ such that the absence of errors up to $\mathcal{CT}$ proves p for M.
  - for instance: the longest "shortest" path from an initial state to any reachable state
- We call $\mathcal{CT}$, the completeness threshold of M w.r.t. p (and the translation scheme)

Completeness threshold: Reachability diameter (1)

- Consider **G**p formulae
- Completeness threshold = minimal number of steps to reach all states
- We call it "reachability diameter" of M
- Left part of the implication: every state that can be reached in n steps ...
- Right part: ... can also be reached in t steps

Definition

$$rd(M) := min\{i | \forall s_0, \ldots, s_n . \exists s'_0, \ldots, s'_t, t \leq i .$$

$$I(s_0) \wedge \bigwedge_{j=0}^{i} T(s_j, s_{j+1}) \Rightarrow \left(I(s'_0) \wedge \bigwedge_{j=0}^{t-1} T(s'_j, s'_{j+1}) \wedge s'_t = s_n \right) \}$$

Completeness threshold: Reachability diameter (2)

- How big should n be ?
- Simple answer (worst case): size of state space, $n = 2^{|V|}$, where V is set of variables defining M
- Better option:
  - Take $n = i + 1$
  - Check wether every state can be reached in $i + 1$ steps

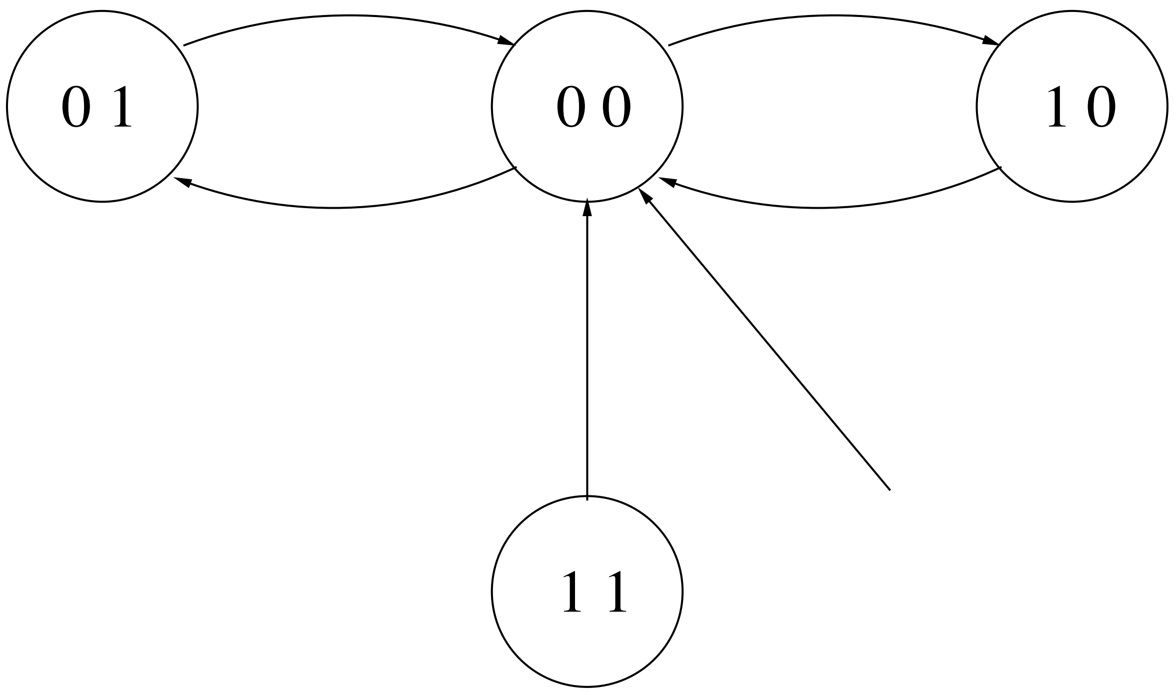
$$rd(M) := min\{i | \forall s_0, \dots, s_{i+1}. \exists s'_0, \dots, s'_i.$$

$$I(s_0) \wedge \bigwedge_{j=0}^{i} T(s_j, s_{j+1}) \Rightarrow \left(I(s'_0) \wedge \bigwedge_{j=0}^{i-1} T(s'_j, s'_{j+1}) \wedge \bigvee_{j=0}^{i} s'_j = s_{i+1} \right) \}$$

Example

show rd for Gp of SMUTE

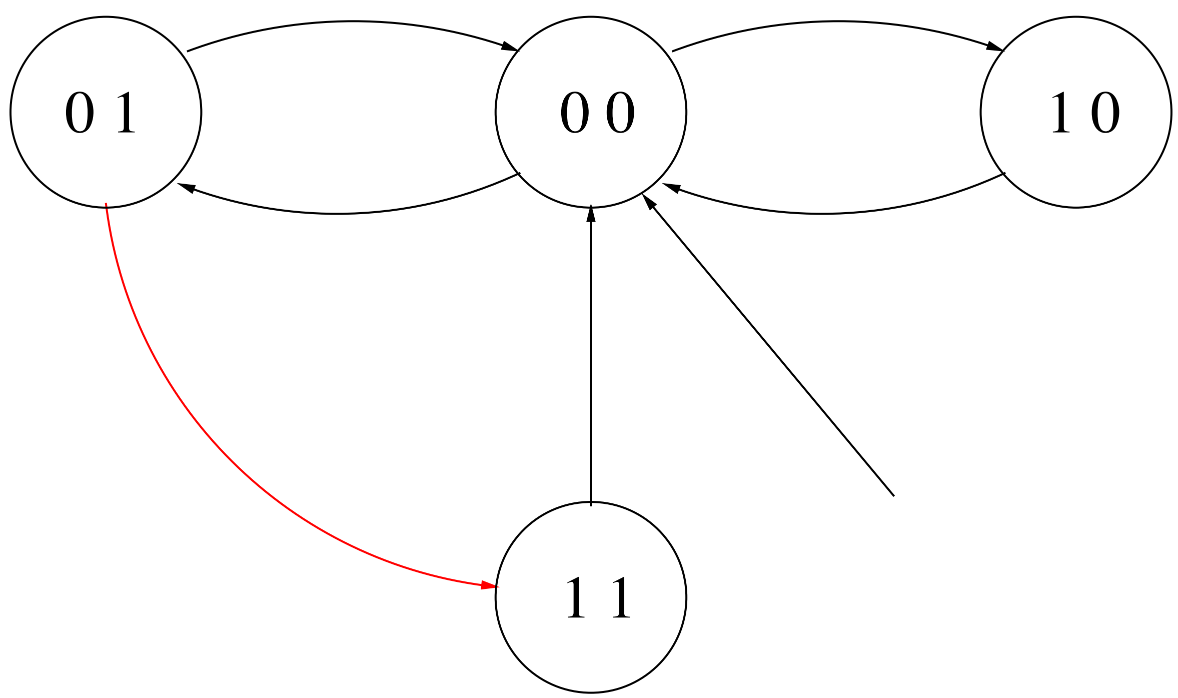
Kripke Structure for SMUTE



State space: $S = \{0, 1\}^2$
State vector: $s \in S = \{0, 1\}^2$
Transition relation $T \subseteq S^2$
An (initialized) path: 00, 01, 00, 10, 00, 01, ...

SMUTE is safe: never 2 processes access the resource simultaneously ($\mathbf{G}\neg(A.pc = 1 \wedge B.pc = 1)$)

Kripke Structure for unsafe SMUTE



Path: 00, 01, 11 is a counter-example to safety

$\mathbf{G}\neg(A.pc = 1 \wedge B.pc = 1)$ is false

$\mathbf{F}(A.pc = 1 \wedge B.pc = 1)$ is true

Completeness threshold: Recurrence diameter for reachability

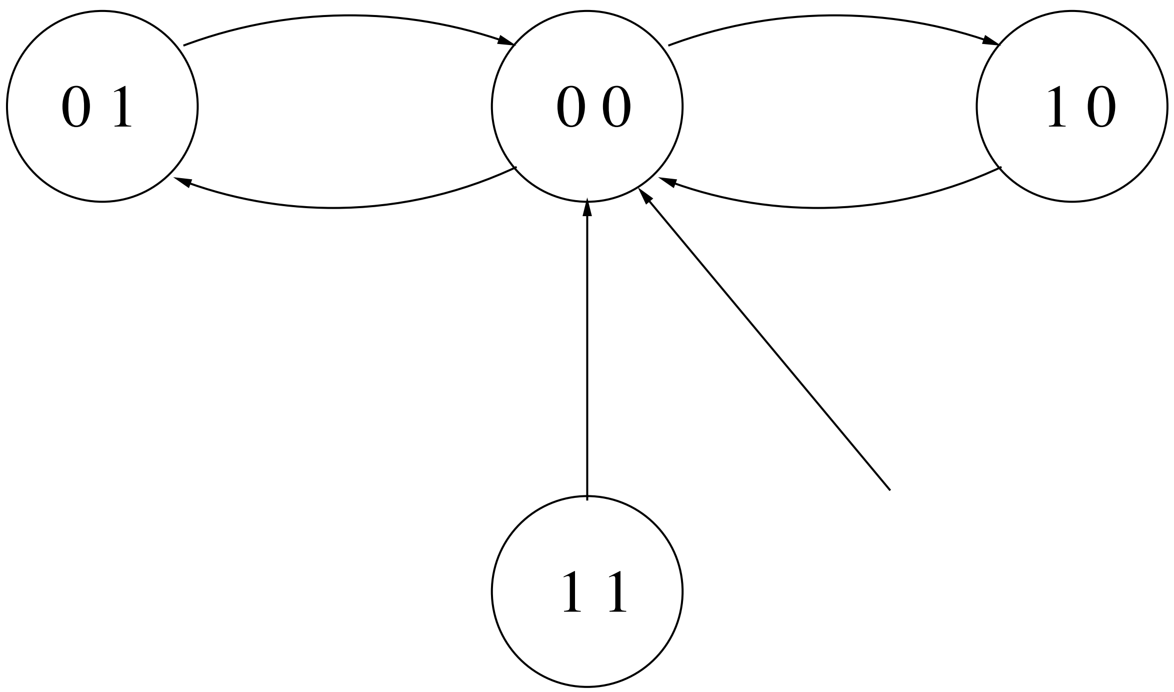
- Previous formula involved alternation of quantifiers ($\forall x.\exists y$)
- Hard to compute on realistic problems
- Overapproximation of $rd(M)$
  - noted $rdr(M)$
  - compute the longest loop-free path
  - overapproximation because every shortest path is also a loop-free path

$$rdr(M) := \max\{i \mid \exists s_0, \ldots, s_i . I(s_0) \wedge \bigwedge_{j=0}^{i} T(s_j, s_{j+1}) \wedge \bigwedge_{j=0}^{i-1} \bigwedge_{k=j+1}^{i} s_j \neq s_k\}$$

Example

show rdr for Gp of SMUTE

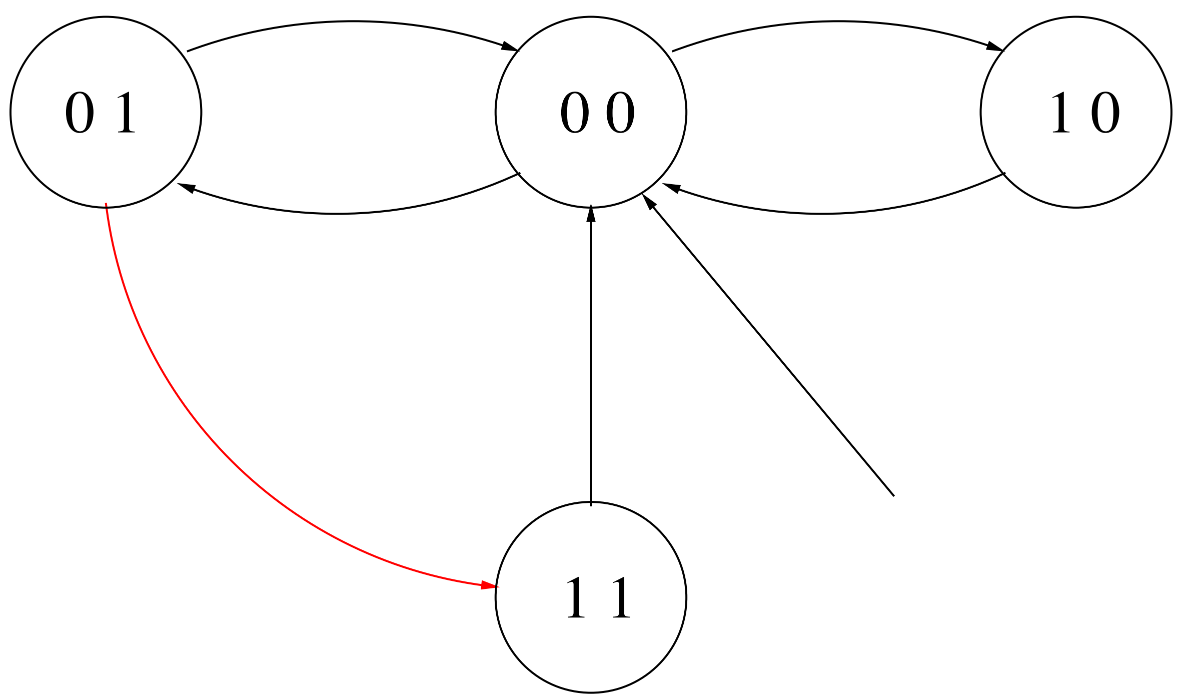
Kripke Structure for SMUTE



State space: $S = \{0, 1\}^2$
State vector: $s \in S = \{0, 1\}^2$
Transition relation $T \subseteq S^2$
An (initialized) path: 00, 01, 00, 10, 00, 01, ...

SMUTE is safe: never 2 processes access the resource simultaneously ($\mathbf{G}\neg(A.pc = 1 \wedge B.pc = 1)$)

Kripke Structure for unsafe SMUTE



Path: 00, 01, 11 is a counter-example to safety

$\mathbf{G}\neg(A.pc = 1 \wedge B.pc = 1)$ is false

$\mathbf{F}(A.pc = 1 \wedge B.pc = 1)$ is true