nuXmv Seminar 4: nuXmv Model Checking Algorithms

K-induction, IC3, CEGAR

Di Marco, Okwieka

University of Rome "La Sapienza"

May 22th, 2023

K-induction

Invariant Checking

- We can check invariants with **inductive reasoning**:
  1. If all the initial states $\mathcal{I}$ satisfy the property.
  2. If from *"good"* states we can only reach other *"good"* states.
  3. Then the System satisfies the formula for all **reachable states**.

SAT-based Inductive Reasoning on Invariants

- Consider a function Θ that returns 1 if a state s^i satisfies our invariant $\mathbf{AG}\,\theta$:

$$\Theta : \mathcal{S} \rightarrow \{0, 1\}$$

- We can call this a **query** on the model.

SAT-based Inductive Reasoning on Invariants (2)

- The inductive reasoning will be:

  1. If all the initial states satisfy θ:

$$\mathcal{I}(s^0) \rightarrow \Theta(s^0) \tag{1}$$

  2. If from *"good"* states we can only reach other *"good"* states:

$$\Big(\Theta(s^{k-1}) \wedge \mathcal{T}(s^{k-1}, s^k)\Big) \rightarrow \Theta(s^k) \tag{2}$$

     i.e. its negation is *unsatisfiable*

  3. Then the Model $\mathcal{M}$ satisfies θ for all **reachable states**.

SAT-based Inductive Reasoning on Invariants (3)

- We will then check for the satisfiability of the negated formula.

$$\mathcal{I}(s^0) \wedge \neg\Theta(s^0) \tag{3}$$

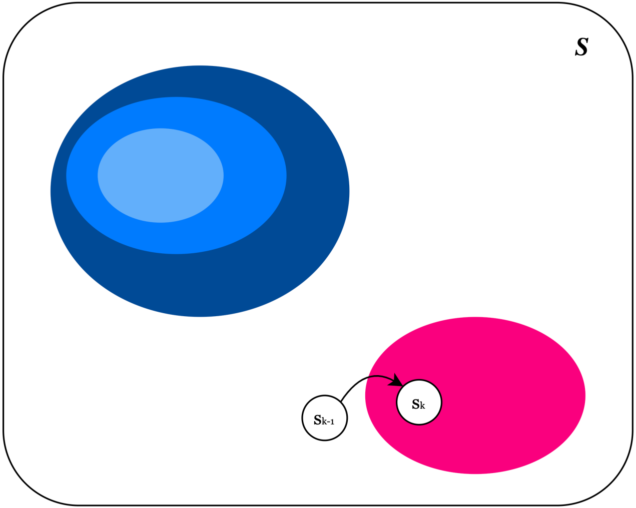
$$\Theta(s^{k-1}) \wedge \mathcal{T}(s^{k-1}, s^k) \wedge \neg\Theta(s^k) \tag{4}$$

- Note that (3) is the equivalent of a 0-step BMC SAT encoding of $\mathbf{F}\,\neg\Theta$

Unreachable States

- The inductive formula alone might fail because of unreachable states.
- If (4) is not valid, it doesn't mean that the model $\mathcal{M}$ does not satisfy the property.
- Both s^{k-1} and s^k might be **unreachable**.
- We need to look at the counterexamples.

Unreachable States (example)



Inductive Steps

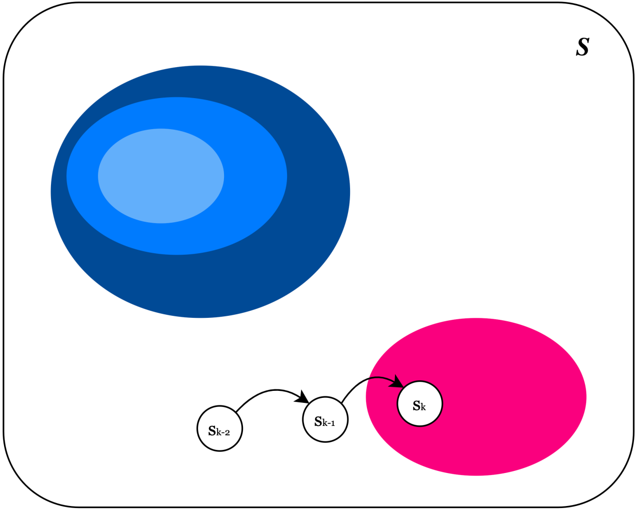
- We can simply increase the depth of the formula by an **inductive step**.

$$\left(\Theta(s^{k-2}) \wedge \mathcal{T}(s^{k-2}, s^{k-1}) \wedge \Theta(s^{k-1}) \wedge \mathcal{T}(s^{k-1}, s^{k}) \wedge \overbrace{\neg(s^{k-2} = s^{k-1})}^{\text{no loops}} \right) \rightarrow \Theta(s^{k}) \tag{5}$$

- Equivalent to looking for a counterexamples with 1-step BMC:

$$\mathcal{I}(s^{0}) \wedge \Theta(s^{0}) \wedge \mathcal{T}(s^{0}, s^{1}) \wedge \neg\Theta(s^{1})$$

Inductive Steps (example)



Inductive Steps (2)

- We will check for the unsatisfiability of the formulas for **increasing values of** k.
- By increasing k we will eventually stop considering **spurious counterexamples**:
  - Chains $s^{k-n}, \ldots, s^k$ of **unreachable** and **different** states.
- K-induction steps can share the formulas of the previous steps:

$$\underbrace{\bigwedge_{i=0}^{k-2} \left(\mathcal{T}(s^i, s^{i+1}) \wedge \Theta(s^i)\right)}_{\text{old steps}} \wedge \underbrace{\mathcal{T}(s^{k-1}, s^k) \wedge \neg\Theta(s^k)}_{\text{new step}} \tag{6}$$

BMC & K-induction

Check for the (un)satisfiability of the invariant Θ

- **Bounded Model Checking**:
  [BMC$_0$] $\mathcal{I}(s^0) \wedge \neg\Theta(s^0)$
  [BMC$_1$] $\mathcal{I}(s^0) \wedge \Theta(s^0) \wedge \mathcal{T}(s^0, s^1) \wedge \neg\Theta(s^1)$
  [BMC$_2$] $\mathcal{I}(s^0) \wedge \Theta(s^0) \wedge \mathcal{T}(s^0, s^1) \wedge \Theta(s^1) \wedge \mathcal{T}(s^1, s^2) \wedge \neg\Theta(s^2)$
- **K-induction**:
  [Kind$_0$] $\Theta(s^0) \wedge \mathcal{T}(s^0, s^1) \wedge \neg\Theta(s^1)$
  [Kind$_1$] $\Theta(s^0) \wedge \mathcal{T}(s^0, s^1) \wedge \Theta(s^1) \wedge \mathcal{T}(s^1, s^2) \wedge \neg\Theta(s^2)$
  ...

Note that in the K-induction above s^0 is **not** the initial state, it is just the first of the sequence of states.

BMC & K-induction (2)

Sheeran et al. 2000

$$\mathrm{Unique}_k := \bigwedge_{i=0}^{k-1} \bigwedge_{j=i+1}^{k} \neg(s^i = s^j) \tag{7}$$

$$\mathrm{Step}_k := \bigwedge_{i=0}^{k} \left(\mathcal{T}(s^i, s^{i+1}) \wedge \Theta(s^i)\right) \wedge \neg\Theta(s^{k+1}) \tag{8}$$

$$\mathrm{Base}_k := \mathcal{I}(s^0) \wedge \bigwedge_{i=0}^{k-1} \left(\mathcal{T}(s^i, s^{i+1}) \wedge \Theta(s^i)\right) \wedge \neg\Theta(s^k) \tag{9}$$

BMC & K-induction (3)

Sheeran et al. 2000

Algorithm Check_Invariant $(\mathcal{I}, \mathcal{T}, \Theta)$

for $k \in \mathbb{N}$ **do**
    if $\mathrm{DPLL}(\mathrm{Base}_k) = \mathit{Sat}$ **then**
        then return *Property_Violated*
    else if $\mathrm{DPLL}(\mathrm{Step}_k \wedge \mathrm{Unique}_k) = \mathit{Unsat}$ **then**
        then return *Property_Verified*
    end if
end for

$\Rightarrow$ Reuses previous searches because the algorithm is incremental!

IC3/PDR

Incremental Construction of Inductive Clauses for Indubitable Correctness (IC3)

Bradley, "SAT-Based Model Checking without Unrolling" VMCAI 2010

- Also known as **Property Directed Reachability** (PDR).
  - IC3 is the first implementation.
  - PDR is the general name of the method.
- As of today it is the *state-of-the-art* symbolic model checking algorithm for *invariants*.
  - Uses **SAT** solving as a subroutine

Notation

- $\mathcal{I}$: the set of **initial states**.
- $\mathcal{V}$: the set of **boolean variables** (or encodings) of the Model $\mathcal{M}$.
  - $\mathcal{V}'$: duplicate of $\mathcal{V}$, it represents the variables evaluation in the **next state**.
- $\mathcal{R}_i$: the set of states **reachable** in i steps or less.
- $\mathcal{T}$: the **transition relation**.
- $\mathcal{P}$: the set of states that satisfy the **invariant property** we want to check.
  - $\mathcal{B}$: the set of states with the **bad prefixes** of our property.
- *cube*: a **conjunction** of literals.
  - *clause*: the negation of a *cube*.

Key Ideas

- We want to find an inductive invariant $\mathcal{F}$ *stronger* than $\mathcal{P}$:
  1. $\mathcal{I} \Rightarrow \mathcal{F}$
  2. $\mathcal{F} \wedge \mathcal{T} \Rightarrow \mathcal{F}'$
  3. $\mathcal{F} \Rightarrow \mathcal{P}$
- By learning inductive facts **incrementally**.
- Remember that:

$$\mathcal{M} \models \mathcal{P} \Leftrightarrow \mathcal{R} \subseteq \mathcal{P} \tag{10}$$

- We will check the satisfiability of:

$$\mathcal{R} \wedge \neg \mathcal{P} \tag{11}$$

Intuitions

- Note that $\mathcal{R}$ is a **fix point** of $\mathcal{T}$:

$$\mathcal{R} \wedge \mathcal{T} \Rightarrow \mathcal{R}' \tag{12}$$

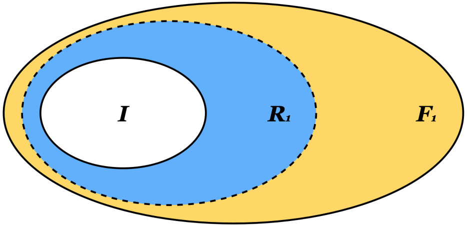
- So if a set of states $\mathcal{S}$ with $\mathcal{I} \subseteq \mathcal{S}$ is also a fix point then it must include $\mathcal{R}$:
  - **If $\mathcal{I} \subseteq \mathcal{S}$ and $\mathcal{S} \wedge \mathcal{T} \Rightarrow \mathcal{S}'$ then $\mathcal{R} \subseteq \mathcal{S}$**
- If we can find such set $\mathcal{S}$ and prove that $\mathcal{S} \subseteq \mathcal{P}$ then we are done!

OARS

Mishchenko, Brayton, "Efficient implementation of property directed reachability", FMCAD 2011

- Iteratively compute **O**ver-**A**pproximated **R**eachability **S**equence:
- The frame $\mathcal{F}_i$ is *over-approximating* $\mathcal{R}_i$ by as much as possible:

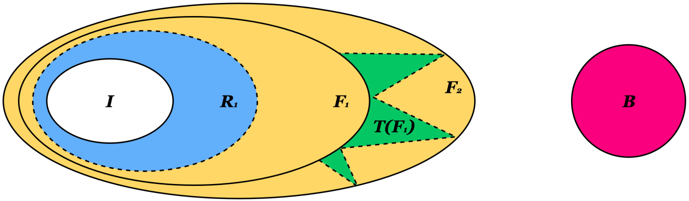
1. $\mathcal{R}_i \subseteq \mathcal{F}_i$
2. $\mathcal{R}_i << \mathcal{F}_i$



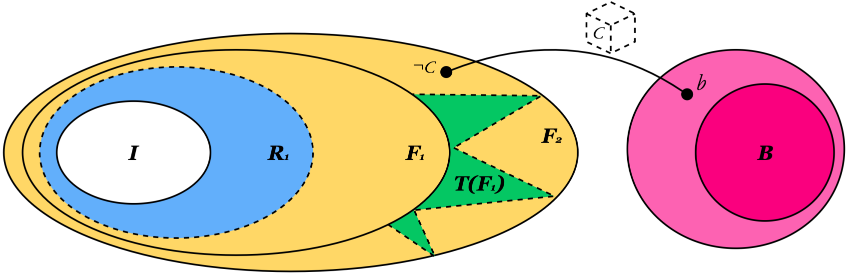
Frames

- Properties of Frames $\mathcal{F}$:

1. $\mathcal{F}_0 \equiv \mathcal{I}$
2. $\forall i > 0$ $\mathcal{F}_i$ is a **set of clauses**
3. $\forall i > 0$ the **clauses** of $\mathcal{F}_{i+1}$ are also in $\mathcal{F}_i$
4. $\mathcal{F}_i \subseteq \mathcal{F}_{i+1}$
5. $\mathcal{T}(\mathcal{F}_i) \subseteq \mathcal{F}_{i+1}$
6. $\forall i$ $\mathcal{F}_i \Rightarrow \neg\mathcal{B}$



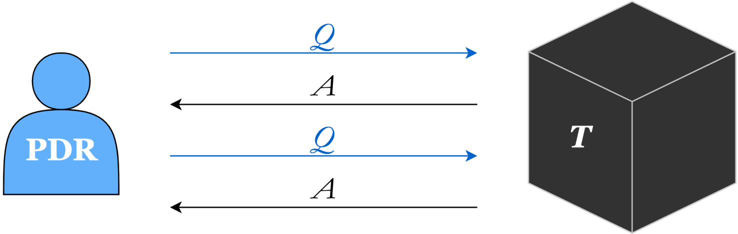
Frames (2)



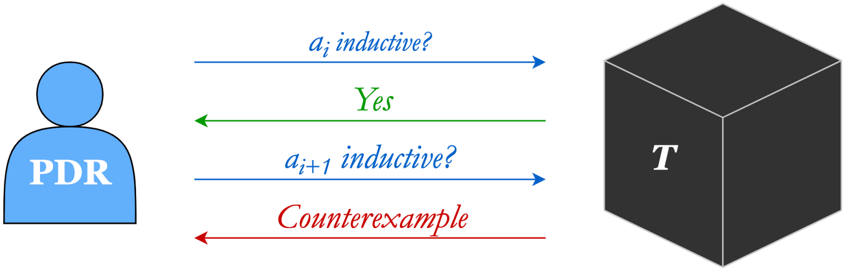
Black-Box Queries

Shoham, "PDR as Abstract Interpretation in the Monotone Theory" ACM SIGPLAN 2023

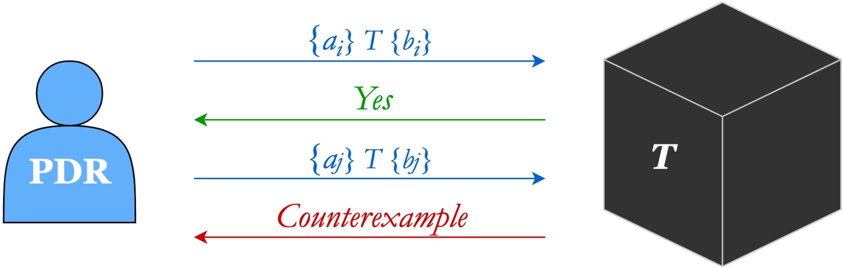
- PDR cannot access the transition relation $\mathcal{T}$ directly.
  - Only though SAT queries



Inductiveness-Query Model



Hoare-Query Model



Proof-Obligation

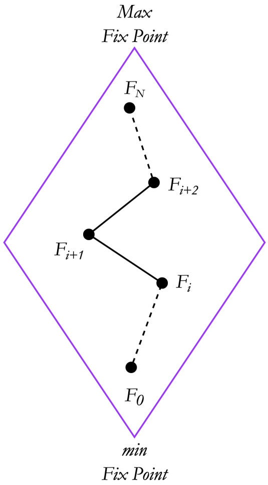
- Given a cube c **extracted from a state** s that can reach $\mathcal{B}$ in a number of steps, consider the query:

$$\text{SAT?}\left[\mathcal{F}_k \wedge \neg c \wedge \mathcal{T} \wedge c'\right] \tag{13}$$

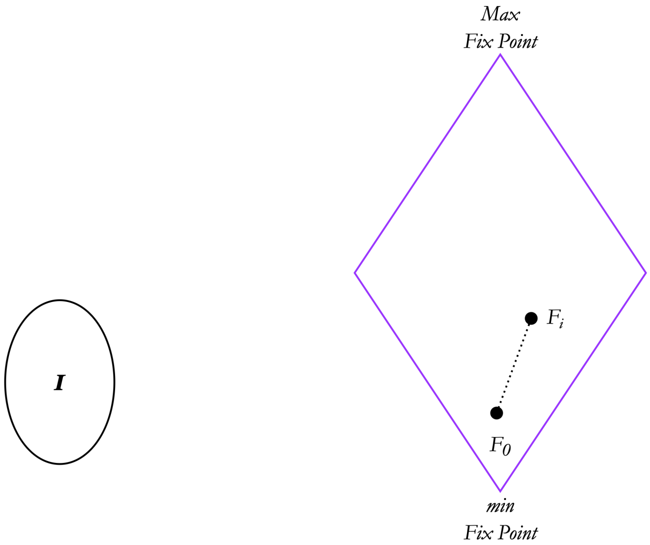
- If it is *Unsat* then the *clauses* in $\mathcal{F}_k$ are strong enough to **block** c at frame $\mathcal{F}_{k+1}$.
  - we can then add $\neg c$ to the **clauses** in $\mathcal{F}_{k+1}$.
- However the properties of the frames require us to **backpropagate** the cube and add c to all preceding frames.
  - If the cube intersects with $\mathcal{F}_0 \equiv \mathcal{I}$ we can no longer block the cube and this leads us to a **counterexample**.

How we build Frames

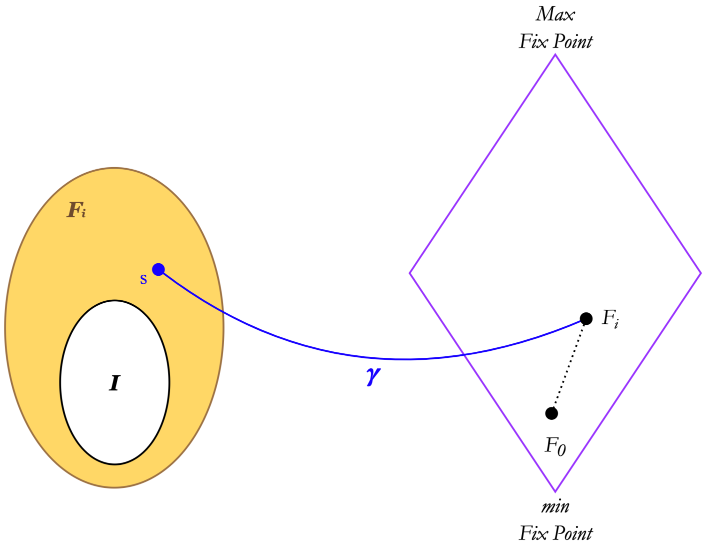
Shoham, "PDR as Abstract Interpretation in the Monotone Theory" ACM SIGPLAN 2023



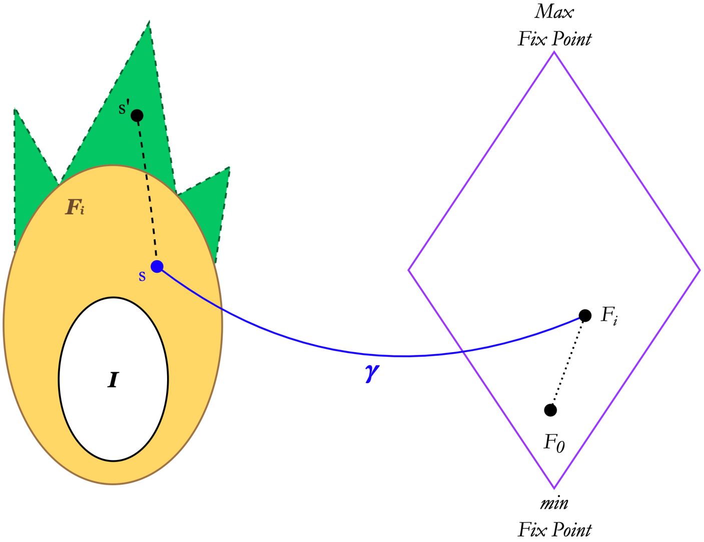
How we build Frames (2)



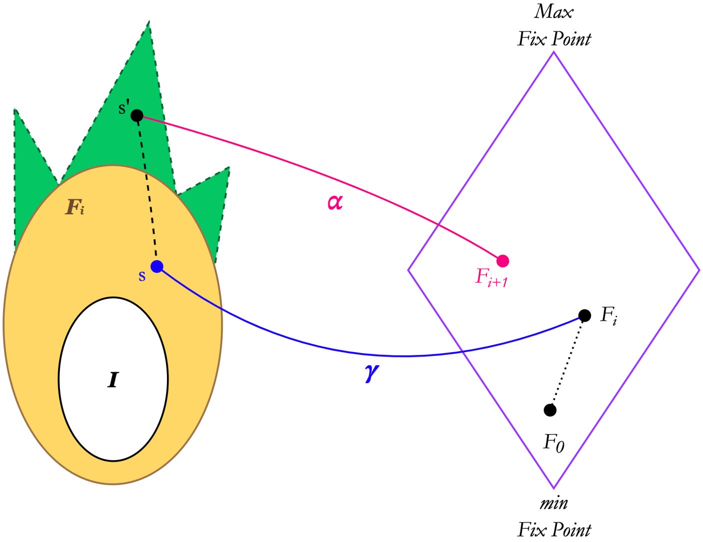
How we build Frames (3)



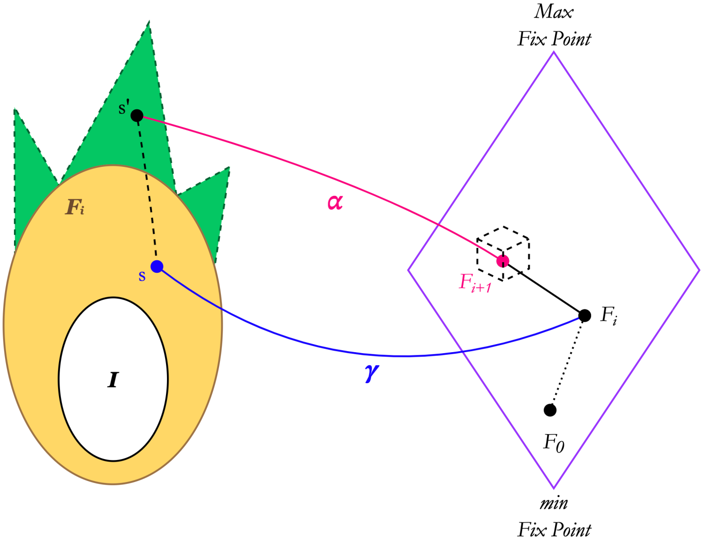
How we build Frames (4)



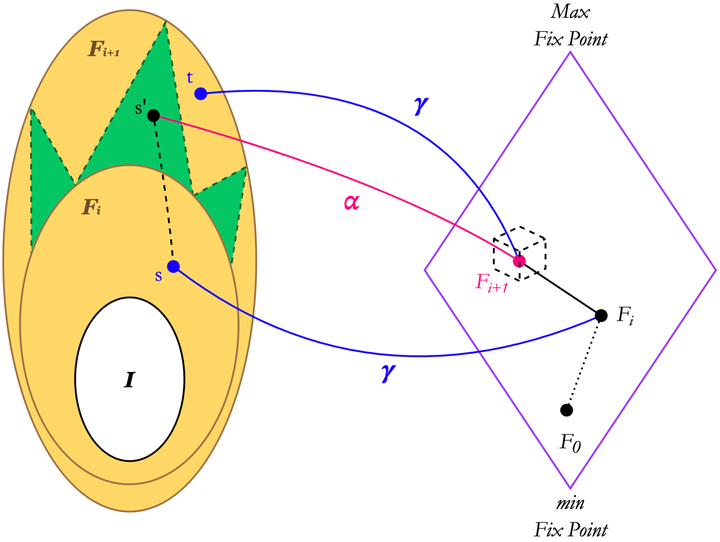
How we build Frames (5)



How we build Frames (6)



How we build Frames (7)



PDR General Algorithm

- The PDR Algorithm proceeds by refining the frame $\mathcal{F}$ at every step
  - adding new clauses and removing old clauses *when possible*
  - while maintaining the properties
- PDR will terminate when:
  1. We find a fix point, i.e. $\mathcal{F}^i \equiv \mathcal{F}^{i+1}$
  2. we find a state $s \in \mathcal{I}$ from which exists a path to $\mathcal{B}$, in that case $\mathcal{M} \not\models \mathcal{P}$

PDR Algorithm

$\mathcal{F}_0 \leftarrow \mathcal{I}$
for $k \in \mathbb{N}$ **do**
  if SAT?$[\mathcal{F}_k \wedge \mathcal{T} \wedge \mathcal{B}'] = \mathit{Unsat}$ **then**
    if $\mathcal{F}_k \equiv \mathcal{F}_{k-1}$ **then** ▷ fix point reached
      return $\mathcal{M} \models \mathcal{P}$
    else
      $\mathcal{F}_{k+1}, c \leftarrow \mathit{update\_frame}(\mathcal{F}_k, \mathcal{B})$
      if $\mathit{back\_propagate}(c, k) = \mathit{BLOCKED}$ **then**
        return counterexample
      end if
    end if
  end if
  return counterexample
end for

PDR Algorithm (2)

update_frame$(\mathcal{F}_k, \mathcal{B})$

1: $b \leftarrow$ some $s \in \mathcal{B}_{-i}$ for some $i \in \mathbb{N}$
2: $c \leftarrow cube(b)$
3: $\mathcal{F}_k \leftarrow \mathcal{F}_k \wedge \neg c$
4: $\mathcal{F}_{k+1} \leftarrow create\_new\_frame(\mathcal{F}_k)$
5: **return** $\mathcal{F}_{k+1}, c$

PDR Algorithm (3)

back_propagate(c, k)

```
for i ∈ [0, k] do                                          ▷ propagation phase
    if SAT?[F_i ∧ ¬c ∧ T ∧ c′] = Unsat then                 ▷ proof-obligation
        F_i ← F_i ∧ ¬c
        F_i ← F_i \ {redundant clauses}
    else
        return BLOCKED
    end if
end for
```

CEGAR

Counter-Example Guided Abstraction Refinement (CEGAR)

Clarke et al. 2003

Problem: model too large to check comfortably

Observation: not all details relevant to check property

$\Longrightarrow$ Create more abstract model, check that instead

Caveat: Abstraction possibly introduces spurious counterexamples

$\Longrightarrow$ Refine abstraction using counterexample

Iterate until successful

CEGAR in paper: based on ACTL* (no existential quantifiers)

Existential Abstraction

Clarke et al. 2003

Surjective abstraction function $h : \mathrm{S} \to \widehat{\mathrm{S}}$ on states, inducing $\equiv$

Abstract model $\widehat{\mathrm{M}} = (\widehat{\mathrm{S}}, \widehat{\mathrm{I}}, \widehat{\mathrm{R}}, \widehat{\mathrm{L}})$ with

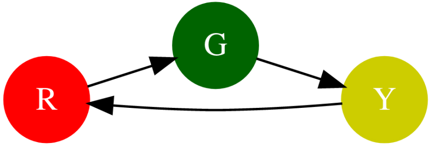
1. $\widehat{\mathrm{S}} = \widehat{\mathrm{D}}$ abstract domain
2. Initial states and transitions in abstraction if there *exist* corresponding concrete states
3. $\widehat{\mathrm{L}}(\widehat{\mathrm{D}}) = \bigcup_{h(d)=\widehat{d}} L(d)$

AP f respects h if for all $d, d' \in \mathrm{D}$ it holds that

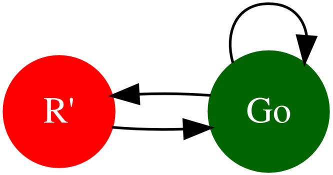
$$h(d) = h(d') \implies (d \models f \iff d' \models f)$$

Spurious Counterexamples

Clarke et al. 2003



Concrete



Abstract

Spurious Counterexample: **AGAF** R holds, but **AGAF** R' does not!

For abstracted model $\widehat{\mathrm{M}}$ it holds that for every φ whose subformulae respect *h*:

$$\widehat{\mathrm{M}} \models \varphi \implies \mathrm{M} \models \varphi$$

Abstraction Function and Variable Clusters

Clarke et al. 2003

Abstraction function $h : D \to \widehat{D}$ with $h = (h_1, \dots, h_N)$ and $h_i : D_i \to \widehat{D}_i$ for $D = D_i \times \dots \times D_N$, analogous for $\widehat{D}$

Naively $D_i = D_{v_i}$ for variables v_i causes combinatorial explosion.

Better approach:

1. Two APs f_1, f_2 *interfere* if they share variables.
2. Build formula clusters FC_i of APs using interference equivalence relation.
3. Build variable clusters VC_i: $v \equiv w \iff v$ and w appear in APs of same FC

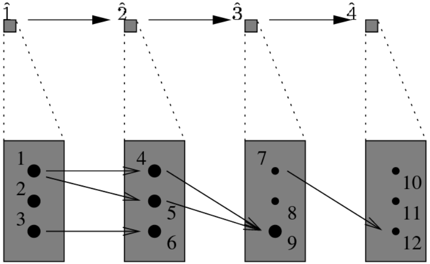
Define $D = D_{\mathrm{VC}_1} \times \dots \times D_{\mathrm{VC}_M}$, analogous for $\widehat{D}$.

SplitPATH: Identifying spurious finite path counterexamples

Clarke et al. 2003 (figure and pseudocode copied verbatim)

Define $h^{-1}(\widehat{s}) = \{s \mid h(s) = \widehat{s}\}$ and lift to paths $\widehat{\mathrm{T}} = \langle \widehat{s_1}, \ldots, \widehat{s_n} \rangle$

$$\text{Concrete counterexample} \iff h^{-1}_{path}(\widehat{\mathrm{T}}) \neq \emptyset \iff S_i \neq \emptyset$$



Algorithm SplitPATH

$S := h^{-1}(\widehat{s_1}) \cap I$

$j := 1$

while $(S \neq \emptyset$ and $j < n)$ $\{$

$\quad j := j + 1$

$\quad S_{\mathrm{prev}} := S$

$\quad S := Img(S, R) \cap h^{-1}(\widehat{s_j})$ $\}$

if $S \neq \emptyset$ **then** output counterexample

else output j, S_{prev}

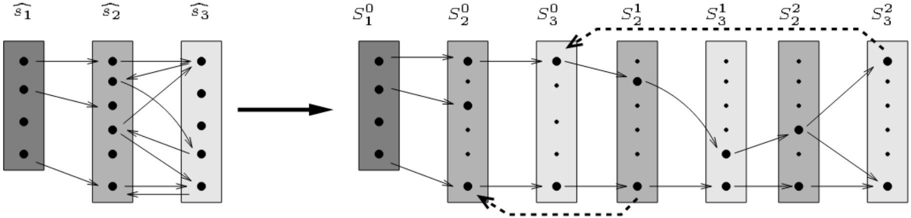
SplitLOOP: Identifying spurious loop counterexamples

Clarke et al. 2003 (figure copied verbatim)

Abstract loop $\widehat{=}$ multiple concrete loops; unwinding periodic *eventually*

Surprising: loop unwindings bounded by $\min_{i+1 \leq j \leq n} |h^{-1}(\widehat{s}_j)|$ for loop from $i+1$ to n

$$\text{Concrete counterexample} \iff h^{-1}_{path}(\widehat{\mathrm{T}}_{unwind}) \neq \emptyset$$



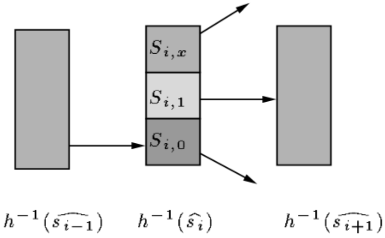
PolyRefine: Refinement Step

Clarke et al. 2003 (figure copied verbatim)

For i so that $Img(S_i, R) \cap h^{-1}(\widehat{s_{i+1}}) = \emptyset$ and S_i reachable, we partition $h^{-1}(\widehat{s_i})$ into $S_{i,0}$ (reachable), $S_{i,1}$ (outgoing), $S_{i,x}$ (isolated)

$$\text{Spurious } \widehat{s_i} \to \widehat{s_{i+1}} \iff S_{i,1} \neq \emptyset$$

Split abstraction relation $\equiv$ into coarsest $\equiv'$ s.t. $S_{i,0}$ and $S_{i,1} \cup S_{i,x}$ are separate. Can be done in polynomial time if no need for optimal (coarsest) refinement.



Fin.