Unbounded Model Checking: IMC and ITP

Automated Program Verification (APV)
Fall 2018

Prof. Arie Gurfinkel

UNIVERSITY OF WATERLOO

SAT-based Model Checking

Bounded Model Checking

- Is there a counterexample of k-steps

Unbounded Model Checking

- Induction and K-Induction (k-IND)
- Interpolation Based Model Checking (IMC)
- Property Directed Reachability (IC3/PDR)

SAT-Based Unbounded Model Checking

Uses BMC for falsification

Simulates forward reachability analysis for verification

Identifies a termination condition

- all reachable states have been found: “fixed-point”

Symbolic Safety and Reachability

A transition system P = (V, Init, Tr, Bad)

P is UNSAFE if and only if there exists a number N s.t.

$$Init(X_0) \wedge \left(\bigwedge_{i=0}^{N-1} Tr(X_i, X_{i+1}) \right) \wedge Bad(X_N) \not\Rightarrow \bot$$

P is SAFE if and only if there exists a safe inductive invariant Inv s.t.

$$Init \Rightarrow Inv$$

$$Inv(X) \wedge Tr(X, X') \Rightarrow Inv(X')$$

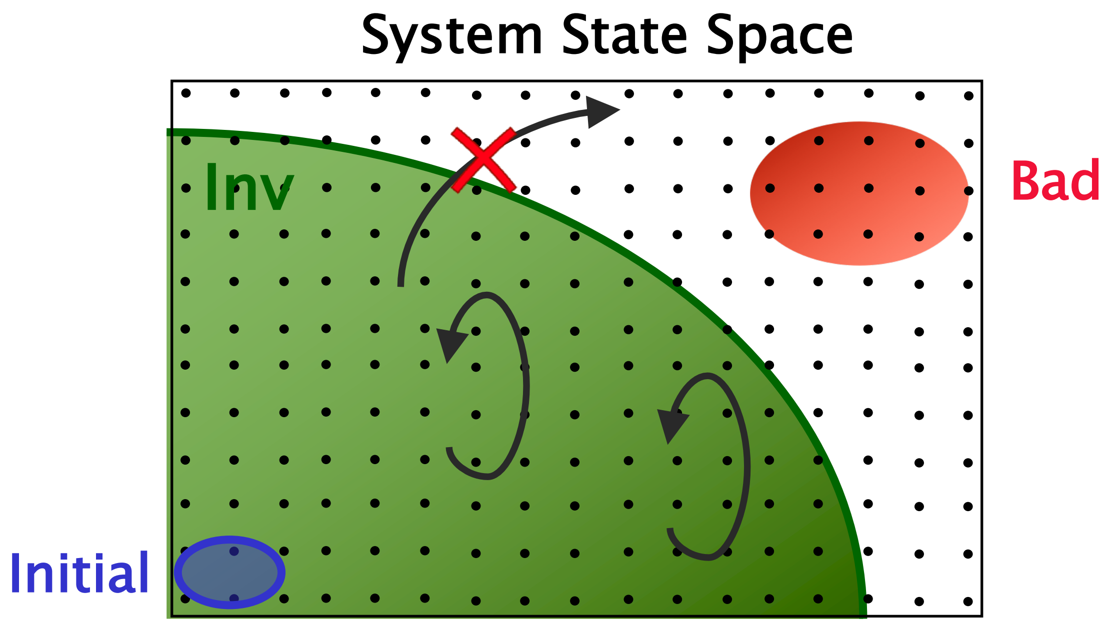
Inductive

$$Inv \Rightarrow \neg Bad$$

Safe

UNIVERSITY OF WATERLOO

Inductive Invariants



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

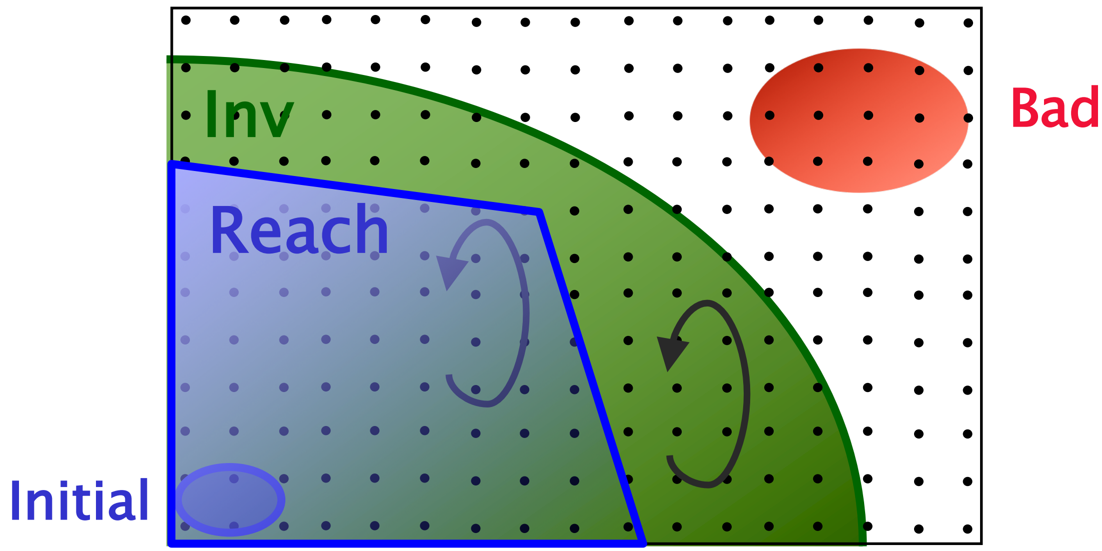
- **Initiation:** Initial ⊆ Inv
- **Safety:** Inv ∩ Bad = ∅
- **Consecution:** TR(Inv) ⊆ Inv

i.e., if s ∈ Inv and s⤳t then t ∈ Inv

UNIVERSITY OF WATERLOO

Inductive Invariants

System State Space



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

- **Initiation:** Initial ⊆ Inv
- **Safety:** Inv ∩ Bad = ∅
- **Consecution:** TR(Inv) ⊆ Inv

i.e., if s ∈ Inv and s⇝t then t ∈ Inv

System S is safe if Reach ∩ Bad = ∅

Forward Reachability Analysis

Does AG P hold?

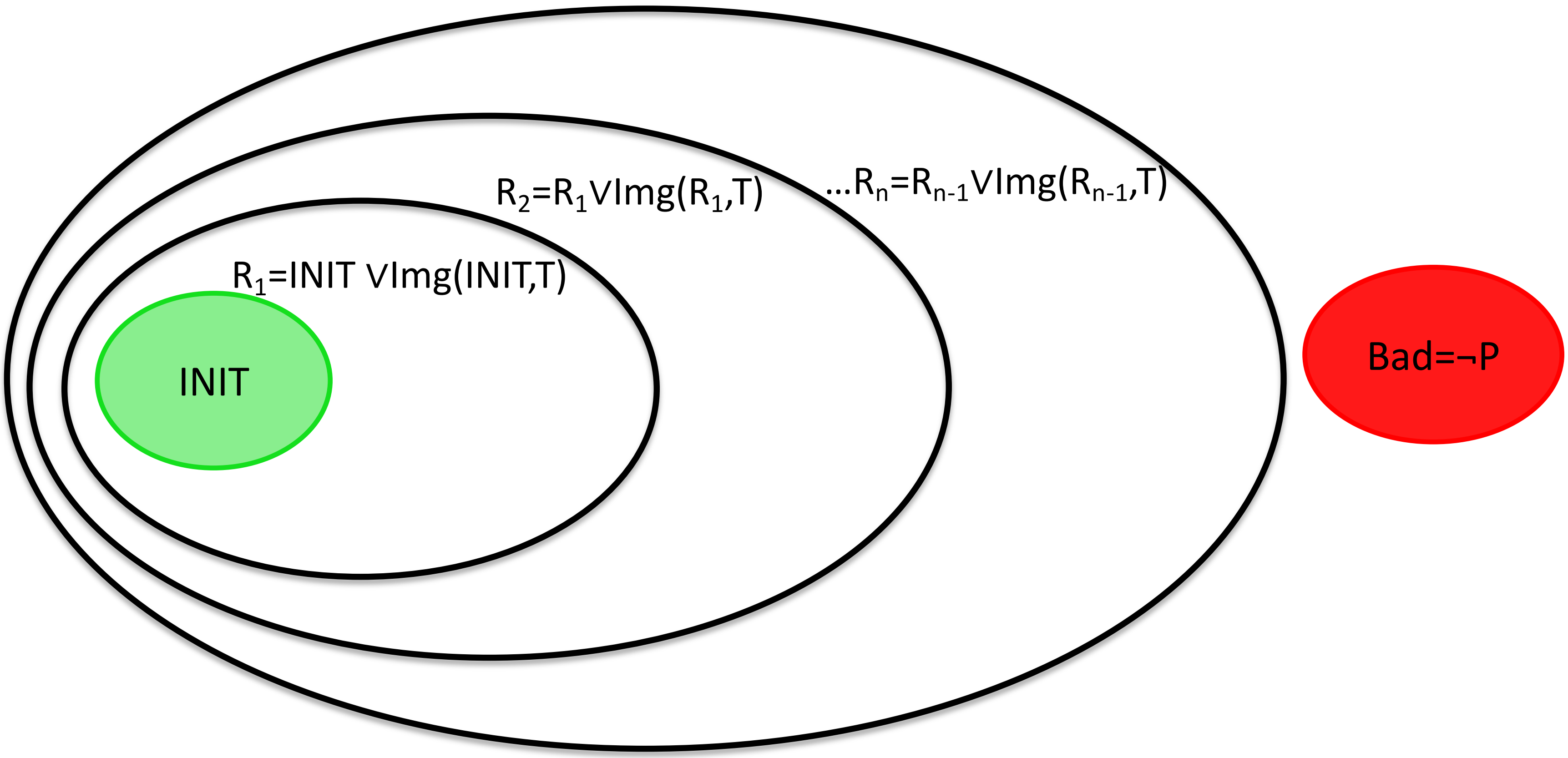


Image operator: Img(Q,T) = ∃ V. (Q ∧ T)

Termination when

- either a bad state satisfying **$\neg$p** is found
- or a **fixpoint** is reached: $\mathbf{R_j \subseteq \cup_{i=0,j-1} R_i}$

➔ **R_j is the set of reachable states**

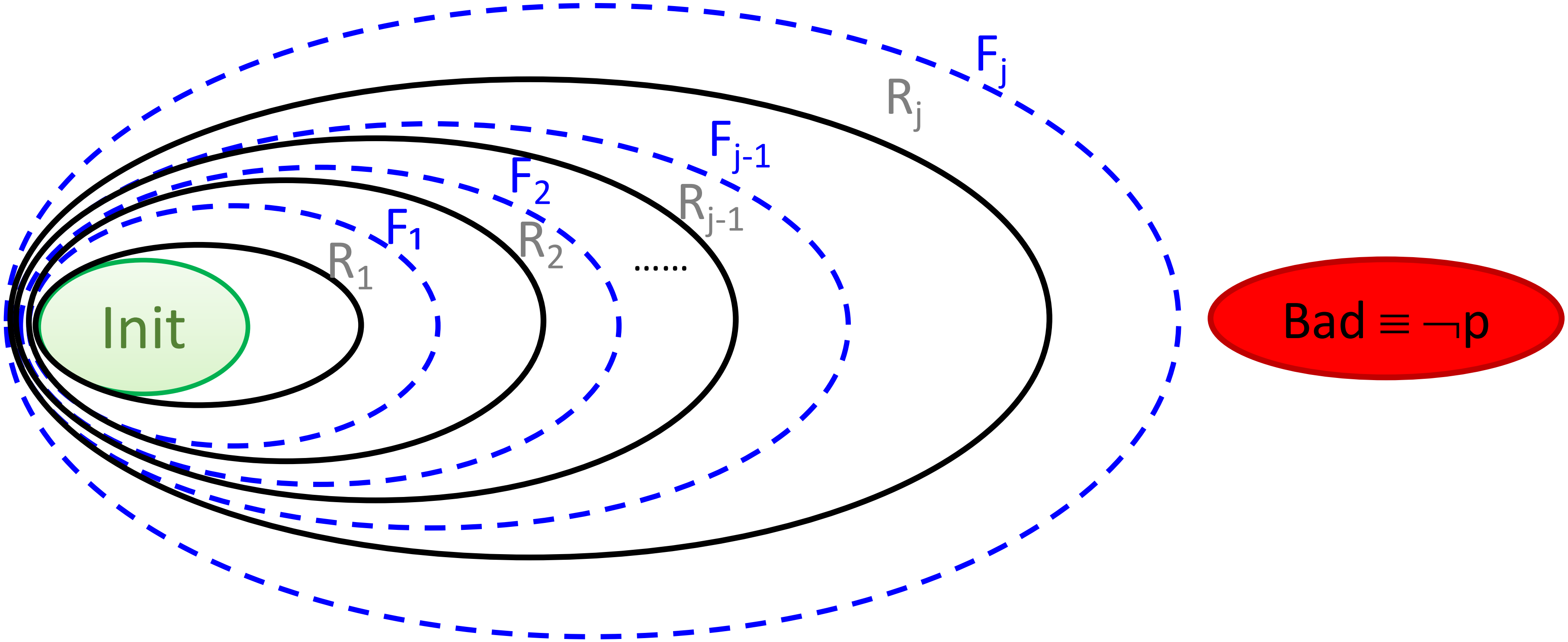
Image computation methods

- Symbolic model checking without BDD's
  - Use SAT solver just for fixed-point detection
    - Abdulla, Bjesse and Een 2000
    - Williams, Biere, Clarke and Gupta 2000
  - Adapt SAT solver to compute image directly
    - McMillan, 2002

Image over-approximation

- BMC and Craig interpolation allow us to compute image over-approximation relative to property.
  - Avoid computing exact image.
  - Maintain SAT solver's advantage of filtering out irrelevant facts.

Approximate Reachability Analysis



- $F_i = F_{i-1} \vee \text{AppxImg}(F_{i-1})$
- F_i over-approximates the states reachable in at most i steps

Over-approximation

- An over-approximate image op. is Img' s.t.

  for all Q, Img(Q,T) implies Img'(Q,T)

- Over-approximate reachability:

$$F_0 = I$$

$$F_{i+1} = F_i \cup Img'(F_i,T)$$

$$F = \cup F_i$$

Fixpoint:

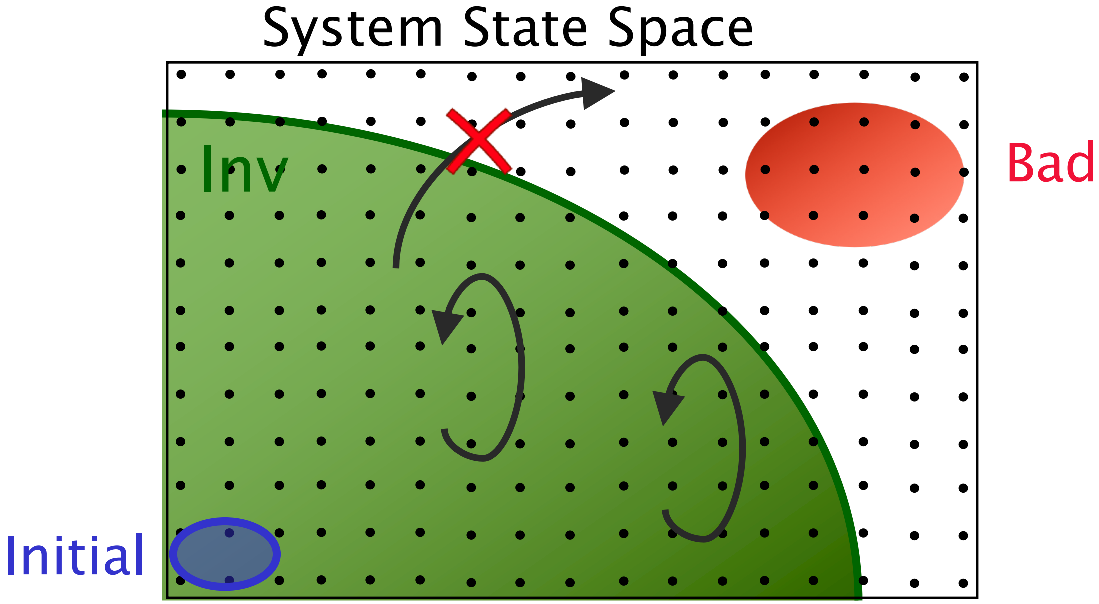
- If $F_{j+1} \equiv F_j$ no new reachable states will be discovered
- F_j is an inductive invariant

Inductive Invariants for verifying AG p

A set of states Inv is an **inductive invariant** if

- Initiation: Initial ⊆ Inv
- Safety: Inv ∩ Bad = ∅
- Consecution: TR(Inv) ⊆ Inv

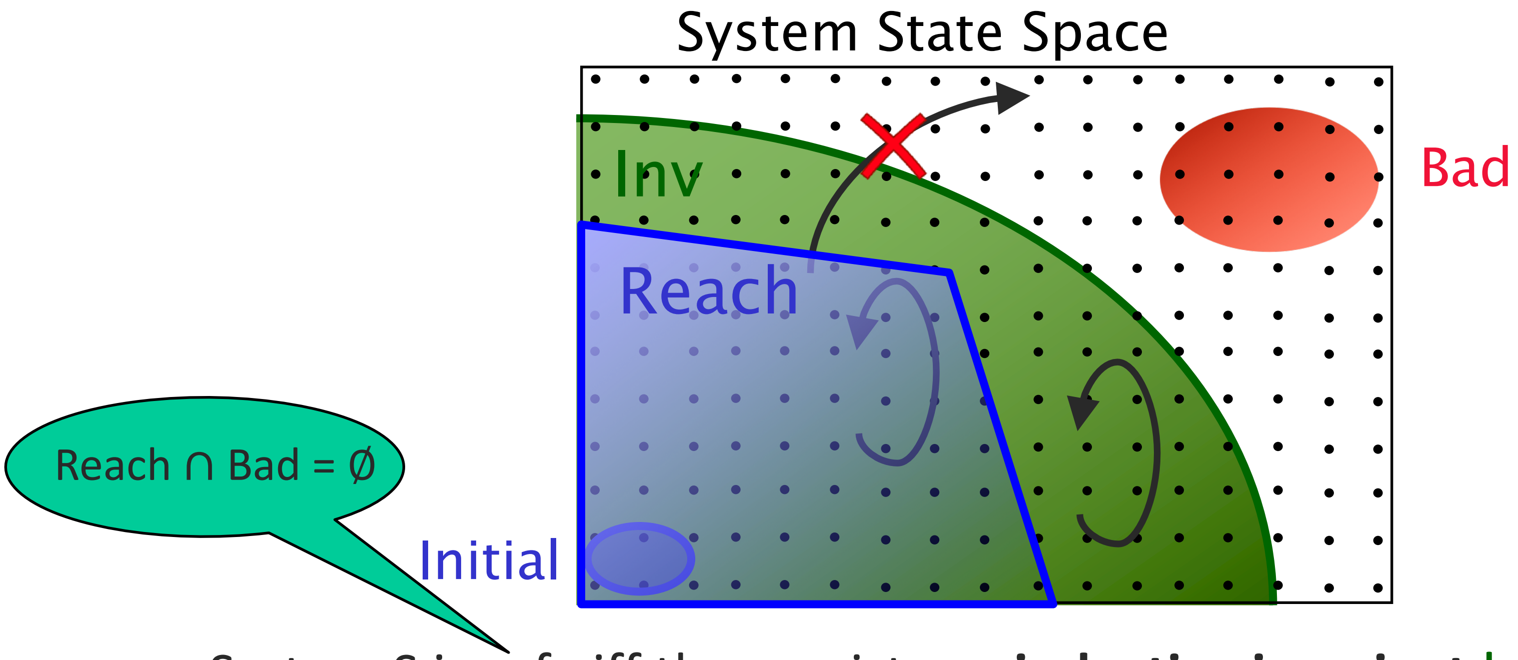
i.e., if s ∈ Inv and s↝t then t ∈ Inv



Inductive Invariants for verifying AG p

A set of states Inv is an **inductive invariant** if
- Initiation: Initial ⊆ Inv
- Safety: Inv ∩ Bad = ∅
- Consecution: TR(Inv) ⊆ Inv

i.e., if s ∈ Inv and s↝t then t ∈ Inv



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

Why is F an inductive invariant?

Recall: forward reachability sequence

$$F_0 = I$$

$$F_{i+1} = F_i \cup Img'(F_i,T)$$

$$F = \cup F_i$$

Also called trace

How to compute an approximate image for reachability analysis?

Adequacy of Approximate Img

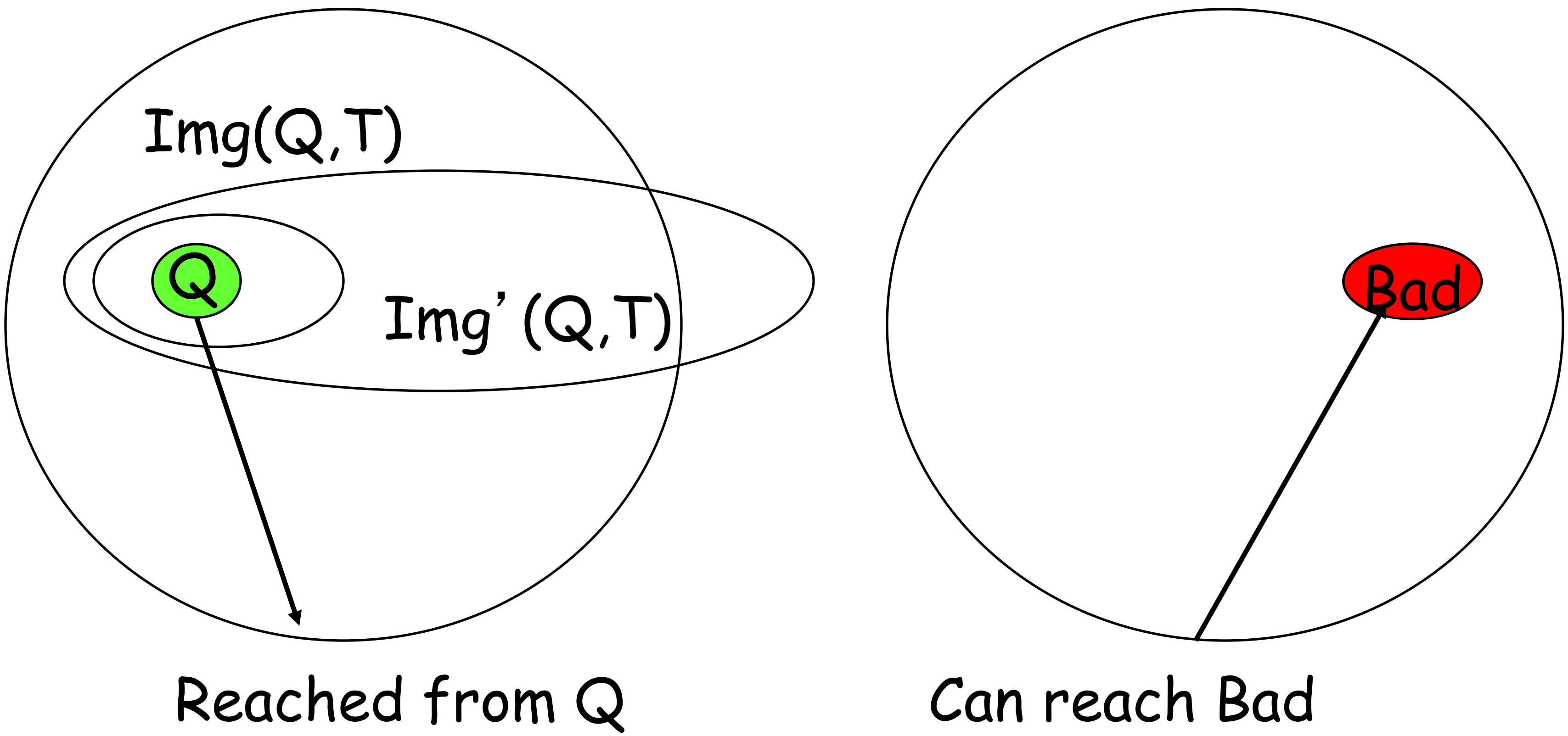
Img' is adequate w.r.t. Bad, when

- if Q cannot reach Bad in any number of steps, then Img'(Q,T) cannot reach Bad in any number of steps

If Img' is adequate, then

- Bad is reachable iff $F \wedge$ Bad is SAT

Adequate image



But how do you get an adequate Img'?

k-adequate image operator

- Img' is k-adequate w.r.t. Bad, when
  - if Q cannot reach Bad,

    Img'(Q,T) cannot reach Bad within k steps

- Note, if k > diameter, then k-adequate is equivalent to adequate.

Interpolating Model Checking (IMC)

Key Idea

- turn SAT/SMT proofs of bounded safety to inductive traces
- repeat forever until a counterexample or inductive invariant are found

Introduced by McMillan in 2003

- Kenneth L. McMillan: Interpolation and SAT-Based Model Checking. CAV2003: 1-13
- based on pairwise Craig interpolation

Extended to sequences

- Yakir Vizel, Orna Grumberg: Interpolation-sequence based model checking. FMCAD 2009: 1-8
  - uses interpolation sequence
- Kenneth L. McMillan: Lazy Abstraction with Interpolants. CAV 2006: 123-136
  - IMPACT: interpolation sequence on each program path

Inductive Trace

An inductive trace of a transition system $P = (V, Init, Tr, Bad)$ is a sequence of formulas $[F_0, \ldots, F_N]$ such that

- $Init \Rightarrow F_0$
- $\forall 0 \leq i < N \, , F_i(v) \wedge Tr(v, u) \Rightarrow F_{i+1}(u)$

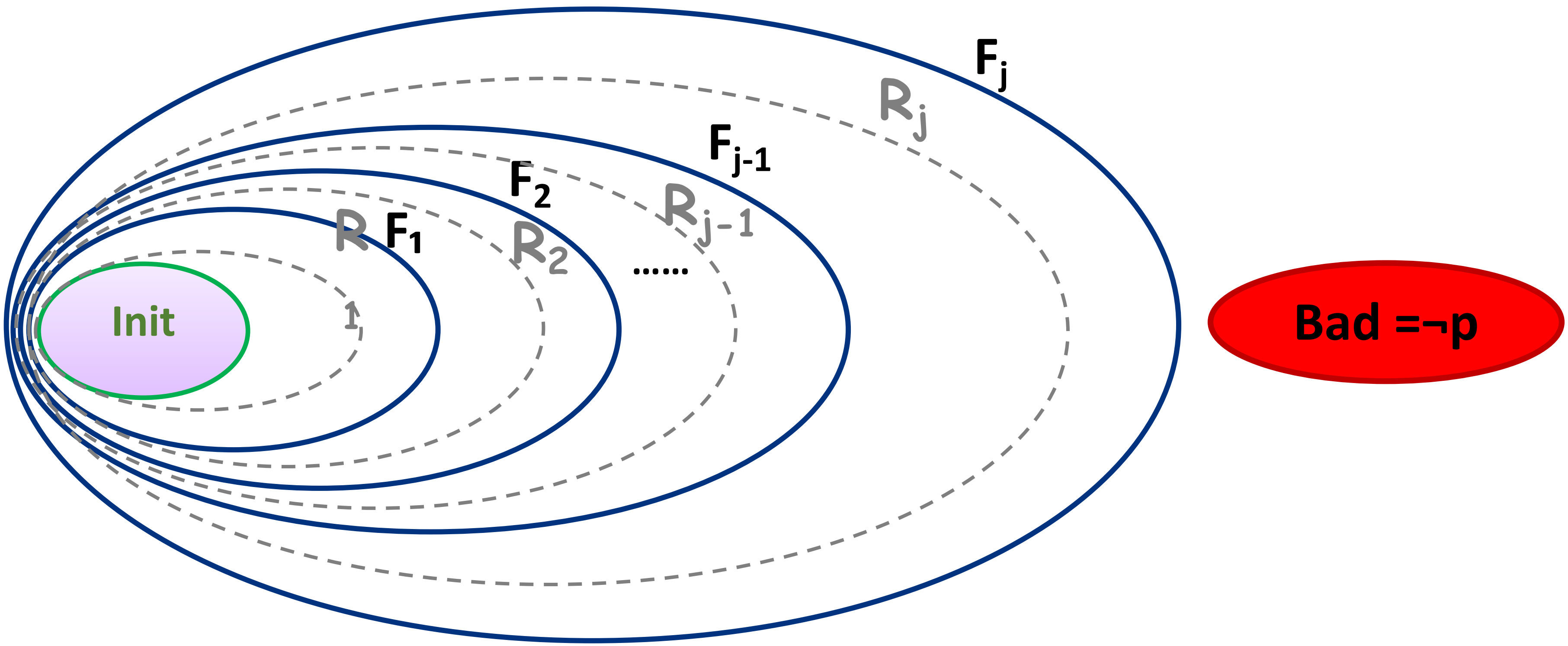
A trace is *safe* iff $\forall\, 0 \leq i \leq N \, , F_i \Rightarrow \neg Bad$

A trace is *monotone* iff $\forall\, 0 \leq i < N \, , F_i \Rightarrow F_{i+1}$

A trace is *closed* iff $\exists\, 1 \leq i \leq N, F_i \Rightarrow (F_0 \vee \ldots \vee F_{i-1})$

A transition system P is *SAFE* iff it admits a safe closed trace

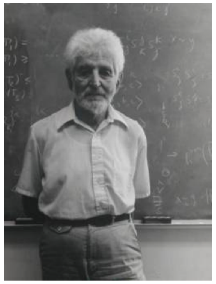
Safe Monotone Inductive Trace



- F_i over-approximates the states that are reachable in at most i steps
- If F_{j+1} -> F_j then F_j is an inductive invariant

UNIVERSITY OF WATERLOO

Craig Interpolants [Craig 57]



Given a pair (A,B) of propositional formulas s.t.

- $A(X,Y) \wedge B(Y,Z)$ is unsatisfiable
- i.e., $A \Rightarrow \neg B$

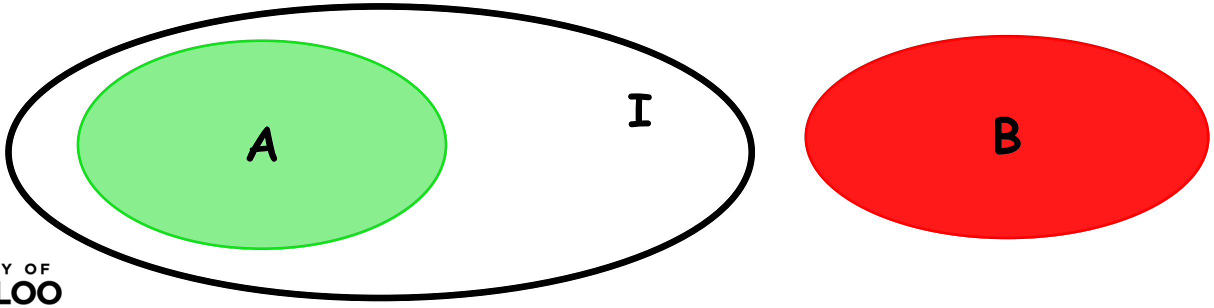
There exists a formula I such that:

- $A \Rightarrow I$
- $I \wedge B$ is unsatisfiable
- I is over Y, the common variables of A and B

$A \Rightarrow \neg B$

$A \Rightarrow I$

$I \Rightarrow \neg B$



UNIVERSITY OF WATERLOO

Example

$$A = p \wedge q, \quad B = \neg q \wedge r,$$

I = ?

Interpolants from Resolution Proofs

When A ∧ B is unsatisfiable, SAT solvers return a a resolution graph deriving false

An interpolant I can be derived from the resolution graph

- In linear time
- In the worst case, I is linear in the resolution graph (i.e., exponential in the size of A and B)

ITP = procedure for computing an interpolant

ITP(A,B) = resulting interpolant

Pudlak,Krajicek 97, McMillan 03

IMC – Interpolation-based MC

McMillan, CAV 2003

Craig Interpolation Theorem is used to safely over-approximate sets of reachable states:

$$\underbrace{INIT(V) \wedge T(V,V^1)}_{A} \wedge \underbrace{T(V^1,V^2) \wedge \ldots \wedge T(V^{k-1},V^k) \wedge (\neg p(V^1) \vee \ldots \vee \neg p(V^k))}_{B}$$

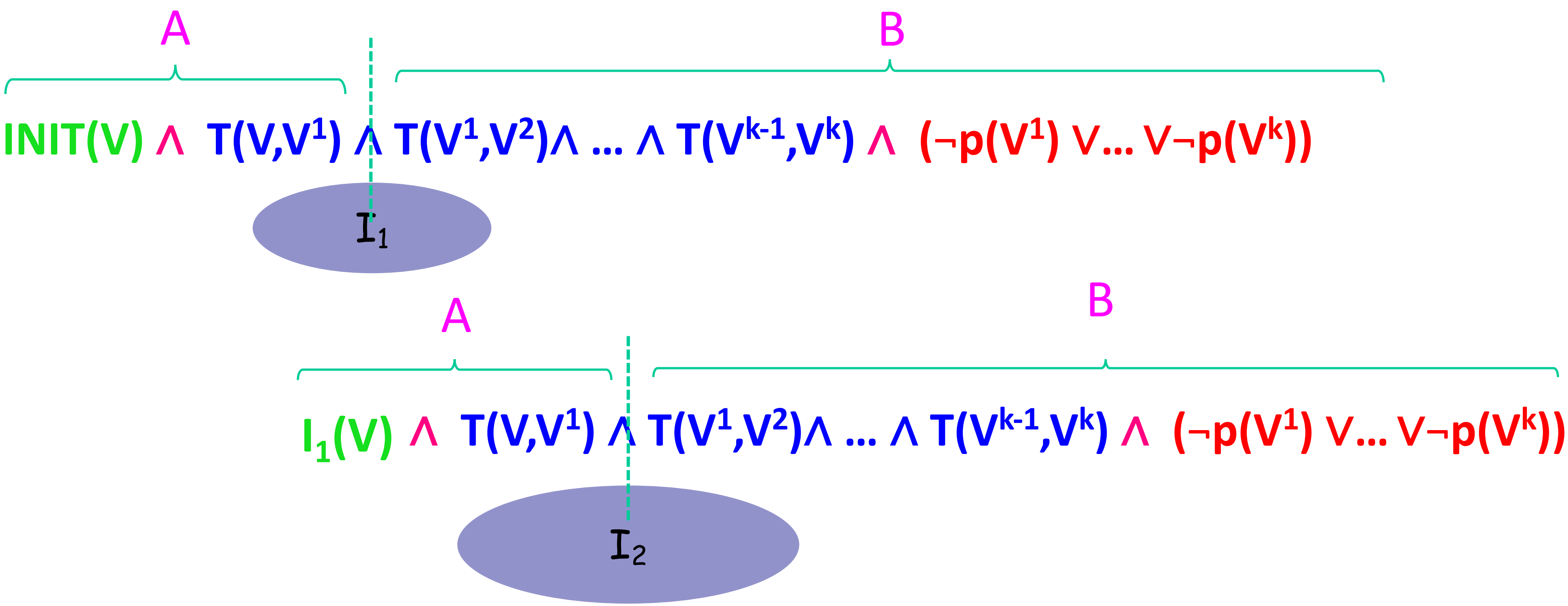
I_1

- Interpolant I_1 is computed
  - over-approximates the states reachable from INIT in 1 transition
  - cannot reach a bad state in $\leq$ k-1 transitions

k-1-adequate overapprox. image!

*only partial – why?

IMC – Interpolation-based MC



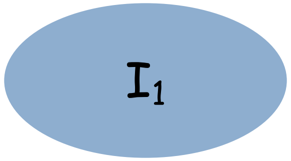
- I_1 is fed back to the BMC solver
- A new interpolant I_2 is computed
  - I_2 over-approximates the states reachable from INIT in 2 transitions
  - cannot reach a bad state in $\leq$ k-1 transitions
- Iterative process

IMC – Interpolation-based MC

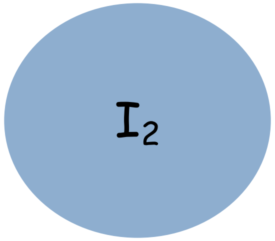
- In IMC, short BMC formulas can prove the nonexistence of long CEXs
  - INIT is replaced by I_j which over-approximates R_j
- If a satisfying assignment to
  $I_j(V) \wedge T(V,V^1) \wedge T(V^1,V^2) \wedge \ldots \wedge T(V^{k-1},V^k) \wedge (\neg p(V^1) \vee \ldots \vee \neg p(V^k))$
  is found, the counterexample might be spurious
  - Since $I_j(V)$ is over-approximated
- Increase precision:
  Increase k and start over with the **original INIT**

Using Interpolation (k=1)

$$INIT(V_0) \wedge T(V_0, V_1) \wedge \neg p(V_1)$$



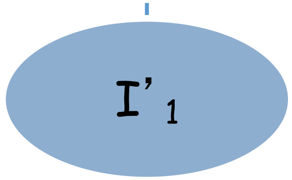
$$I_1(V_0) \wedge T(V_0, V_1) \wedge \neg p(V_1)$$



$$I_2(V_0) \wedge T(V_0, V_1) \wedge \neg p(V_1)$$

Using Interpolation (k=2)

$$INIT(V_0) \wedge T(V_0, V_1) \wedge T(V_1, V_2) \wedge (\neg q(V_1) \vee \neg q(V_2))$$



$$I_1'(V_0) \wedge T(V_0, V_1) \wedge T(V_1, V_2) \wedge (\neg q(V_1) \vee \neg q(V_2))$$

.

.

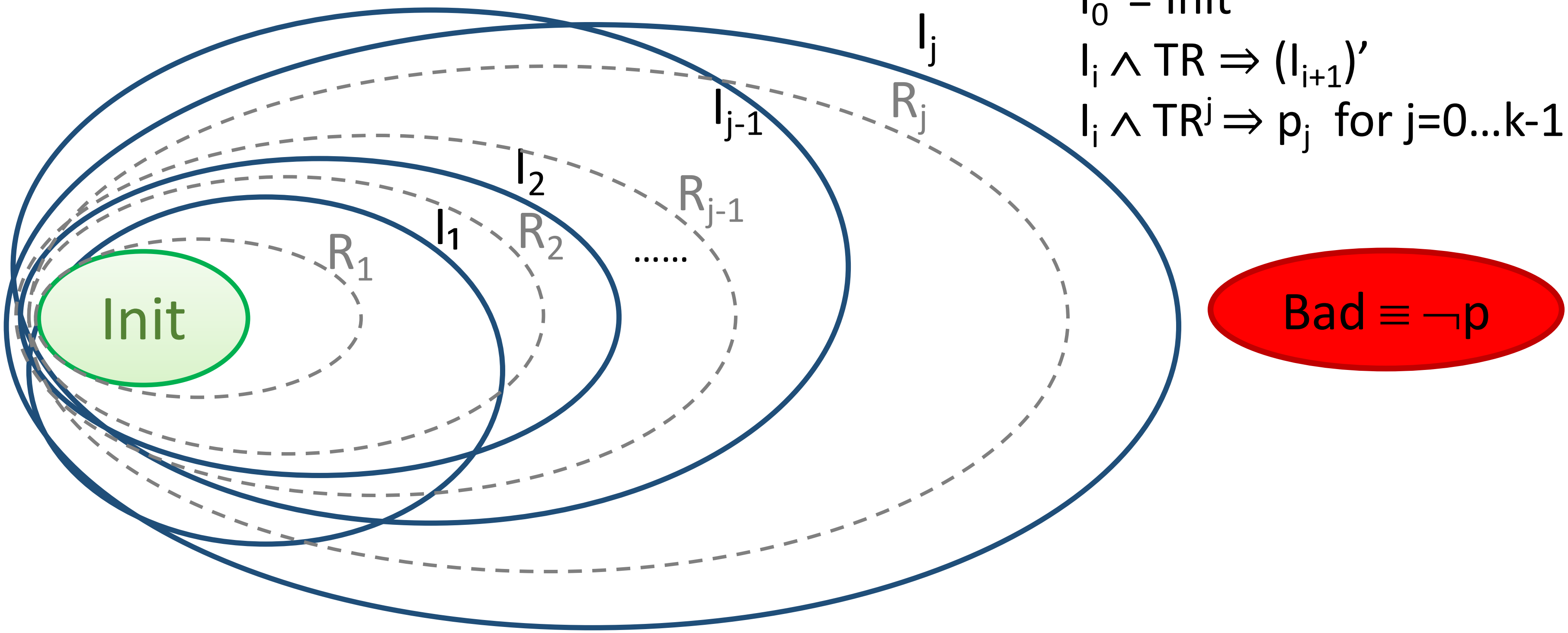
.

$$I_k'(V_0) \wedge T(V_0, V_1) \wedge T(V_1, V_2) \wedge (\neg q(V_1) \vee \neg q(V_2))$$

- A fixpoint is checked whenever a new interpolant is computed

- For iteration i, every new interpolant is checked for inclusion in all previously computed interpolants for the same i
  - $\mathbf{I_n \Rightarrow INIT \vee V_{j=1,n-1}\, I_j}$

IMC as Approximate Forward Reachability (1)



Inductive trace:

I_0 = Init

$I_i \wedge TR \Rightarrow (I_{i+1})'$

$I_i \wedge TR^j \Rightarrow p_j$ for j=0…k-1

I_i over-approximates the states that are reachable in (exactly) i steps

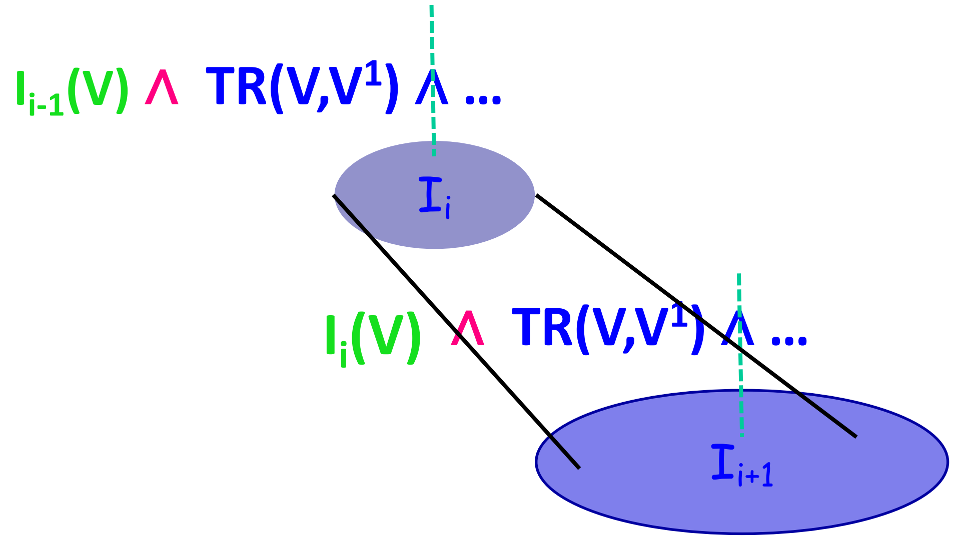
Fixpoint: If $I_j \Rightarrow INIT \vee I_1 \vee \ldots \vee I_{j-1}$ then $INIT \vee I_1 \vee \ldots \vee I_{j-1}$ is an inductive invariant

Inductive Invariants in IMC

Claim: If $\mathbf{I_j => INIT \lor I_1 \lor ... \lor I_{j-1}}$ then $\mathbf{INIT \lor I_1 \lor ... \lor I_{j-1}}$ is an ind. invariant

Proof:

Consecution: $(INIT \lor I_1 \lor ... \lor I_{j-1}) \land TR => (I_1 \lor I_2 \lor ... \lor I_j)' => (INIT \lor I_1 \lor ... \lor I_{j-1})'$



Initiation: $INIT => INIT \lor I_1 \lor ... \lor I_{j-1}$

Safety: $INIT \lor I_1 \lor ... \lor I_{j-1} => p$

IMC Pseudocode (1)

```
k=1
while (true) {
  j = 0
  I_0 = INIT
  while(true) {
    if (SAT(I_j ∧ TR^k ∧ (¬p_1∨… ∨¬p_k)) {
      if (j==0) then return CEX
      k++; break;
    } else  // UNSAT
      I_{j+1} = ITP(I_j∧ TR, TR^{k-1} ∧ (¬p_1∨… ∨¬p_k));
      if (I_{j+1} => INIT ∨ I_1 ∨… ∨ I_j ) then return SAFE
      j++;
    }
}
```

IMC Pseudocode (2)

McMillan 2003

```
k=1
while (true) {
  I = INIT
  while(true) {
    if (SAT(I ∧ TR^k ∧ (¬p_1∨… ∨¬p_k)) {
      if (I = INIT) then return CEX
      k++; break;
    } else  // UNSAT
      I’ = ITP(I ∧ TR, TR^(k-1) ∧ (¬p_1∨… ∨¬p_k));
      if (I’ => I ) then return SAFE
      I = I ∨ I’;
   }
}
```

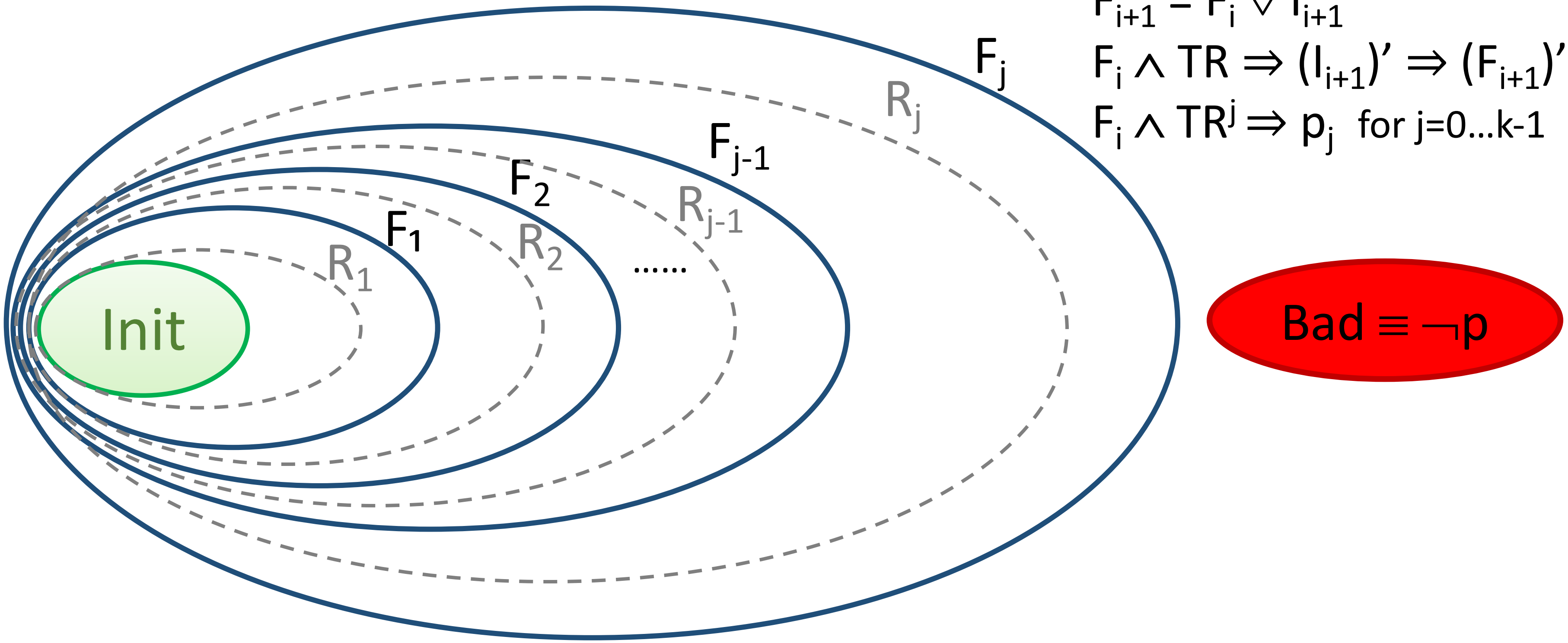
Approximate Reachability Sequence in IMC (2)

$F_0 = Init$

$F_{i+1} = F_i \vee I_{i+1}$

$F_i \wedge TR \Rightarrow (I_{i+1})' \Rightarrow (F_{i+1})'$

$F_i \wedge TR^j \Rightarrow p_j$ for j=0…k-1



- F_i over-approximates the states that are reachable in at most i steps
- If $F_{j+1} \Rightarrow F_j$ then F_j is an inductive invariant

Termination

- Since k increases at every iteration, eventually $k > d$, the diameter, in which case Img' is adequate, and hence we terminate.

Notes:

- don't need to know when $k > d$ in order to terminate
- often termination occurs with $k << d$

Interpolation-based MC

- Fully SAT-based
- Inherits SAT solvers ability to concentrate on facts relevant to a property
- Most effective when
  - Very large set of facts is available
  - Only a small subset are relevant to property
- For true properties, appears to converge for smaller k values
- Disadvantage: start from scratch for each k

IMC WITH INTERPOLATION SEQUENCE

Interpolation Sequence

- If $A_1 \wedge \ldots \wedge A_{k+1}$ is unsatisfiable, then there exists an *interpolation-sequence* $I_0, I_1, \ldots, I_k, I_{k+1}$ for $(A_1, \ldots, A_{k+1})$ s.t.:

  $I_0 = T$ and $I_{k+1} = F$

  $I_j \wedge A_{j+1} \Rightarrow I_{j+1}$

  In particular: $A_1 \Rightarrow I_1$ $I_k \Rightarrow \neg A_{k+1}$

  I_j - over common variables of $A_1, \ldots, A_j$ and $A_{j+1}, \ldots, A_k$

- Each I_j can be computed as the interpolant of $A = A_1 \wedge \ldots \wedge A_j$ and $B = A_{j+1} \wedge \ldots \wedge A_k$
  - **All I_j's should be computed on the same resolution graph**

Interpolation Sequence

- If $A_1 \wedge \ldots \wedge A_{k+1}$ is unsatisfiable, then there exists an *interpolation-sequence* $I_0, I_1, \ldots, I_k, I_{k+1}$ for $(A_1, \ldots, A_{k+1})$ s.t.:

$$I_0 = T \quad \text{and} \quad I_{k+1} = F$$

$$I_j \wedge A_{j+1} \Rightarrow I_{j+1}$$

I_j - over common variables of $A_1, \ldots, A_j$ and $A_{j+1}, \ldots, A_k$

In particular:
$A_1 \Rightarrow I_1 \quad I_k \Rightarrow \neg A_{k+1}$

$$\mathrm{Init} \wedge Tr_1 \quad Tr_2 \quad Tr_3 \quad Tr_4 \quad Tr_5 \quad \neg p$$

$$I_1 \quad I_2 \quad I_3 \quad I_4 \quad I_5$$

Computed by pairwise interpolation applied to different cuts of a fixed resolution proof

- **All I_j's should be computed on the same resolution graph**

Reachability with Interpolation-Sequence

Vizel , Grumberg, FMCAD 2009

- Unsatisfiable BMC formula partitioned in the following manner:

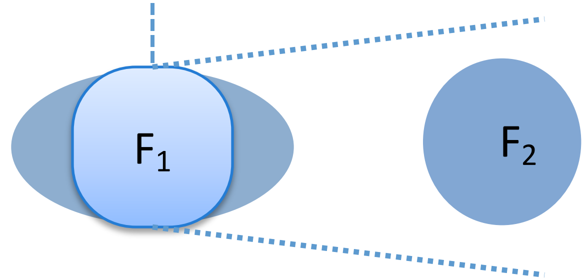
$$\underbrace{INIT(V^0) \wedge T(V^0,V^1)}_{A_1} \;\Big|_{I_1}\; \wedge \underbrace{T(V^1,V^2)}_{A_2} \;\Big|_{I_2}\; \wedge \underbrace{T(V^2,V^3)}_{A_3} \;\Big|_{I_3}\; \wedge \ldots \wedge \underbrace{T(V^{k-1},V^k)}_{A_k} \;\Big|_{I_k}\; \wedge \underbrace{\neg P(V^k)}_{A_{k+1}}$$

$$I_j \wedge A_{j+1} \Rightarrow I_{j+1}$$

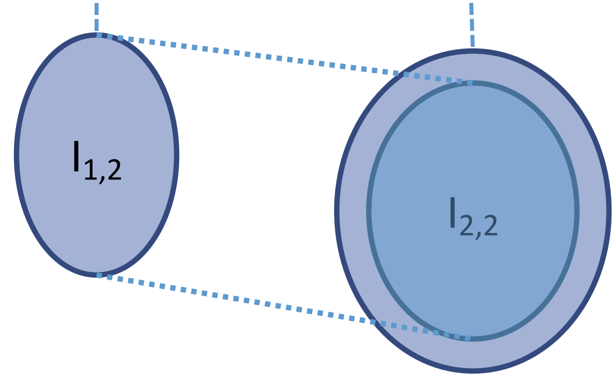
I_j - over common variables of $A_1, \ldots, A_j$ and $A_{j+1}, \ldots, A_{k+1}$

Using Interpolation Sequence

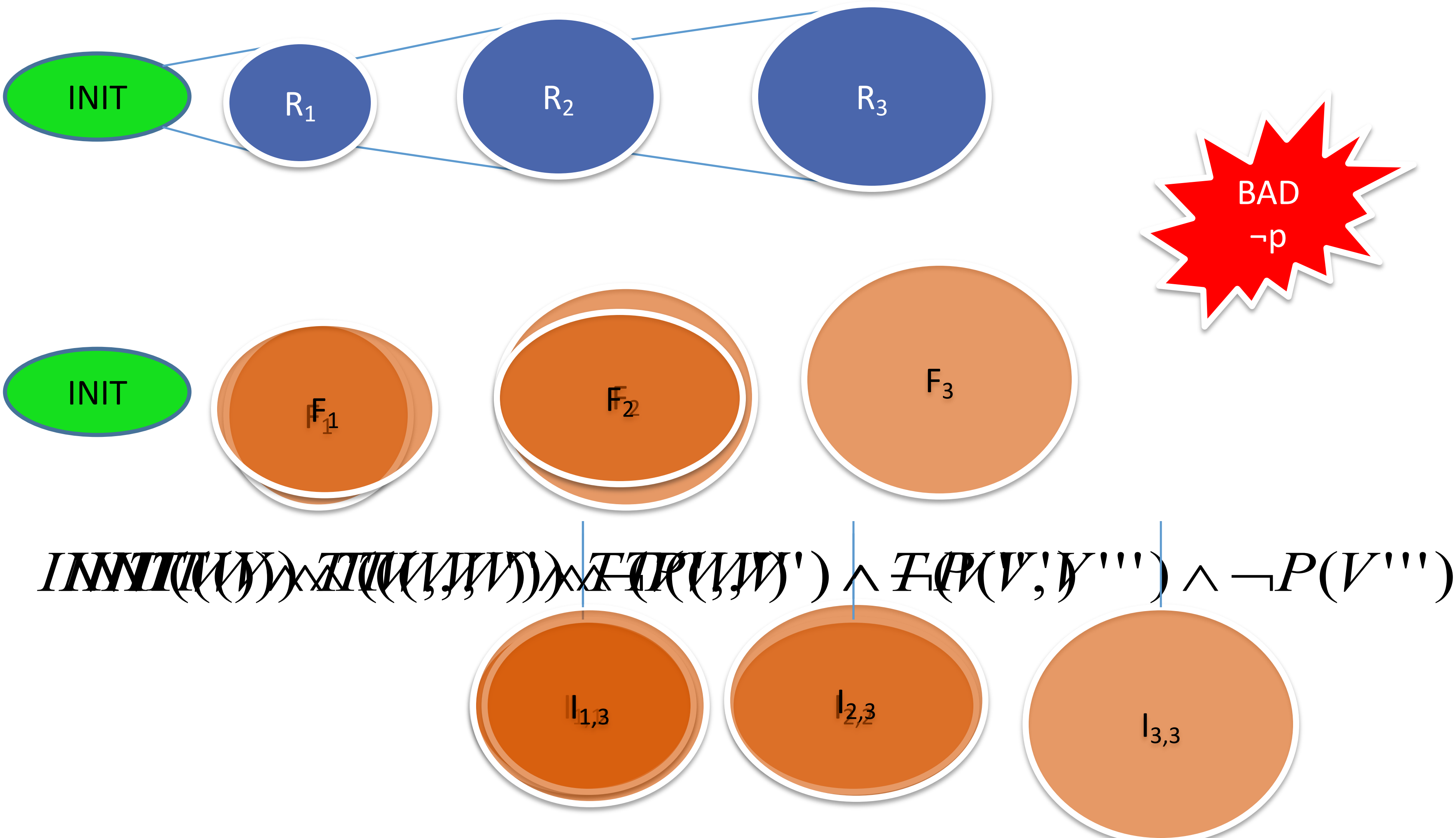
$$INIT(V_0) \wedge T(V_0, V_1) \wedge \neg p(V_1)$$



$$INIT(V_0) \wedge T(V_0, V_1) \wedge T(V_1, V_2) \wedge \neg p(V_2)$$



Analogy to Forward Reachability



Reachability with Interpolation-Sequence

- BMC is used for bug finding
- Interpolation-sequence computes an inductive trace: $<F_0,F_1,\ldots,F_k>$ from BMC formulas
  - Safe over-approximations of reachable states
  - $F_i(V) \wedge T(V,V') \Rightarrow F_{i+1}(V')$
  - $F_i \Rightarrow P$
- Integrated into the BMC loop to detect termination

Checking if a "fixpoint" has been reached

- Does there exist $2 \leq n \leq k$ such that $F_n \Rightarrow \vee_{j=1\ldots n-1} F_j$?
- Similar to checking fixpoint in forward reachability analysis:
  $R_k \subseteq \cup_{j=1\ldots k-1} R_j$
- But here we check inclusion for every $2 \leq k \leq n$
  - No monotonicity because of the approximation
- "Fixpoint" is checked with a SAT solver

Termination

Always terminates

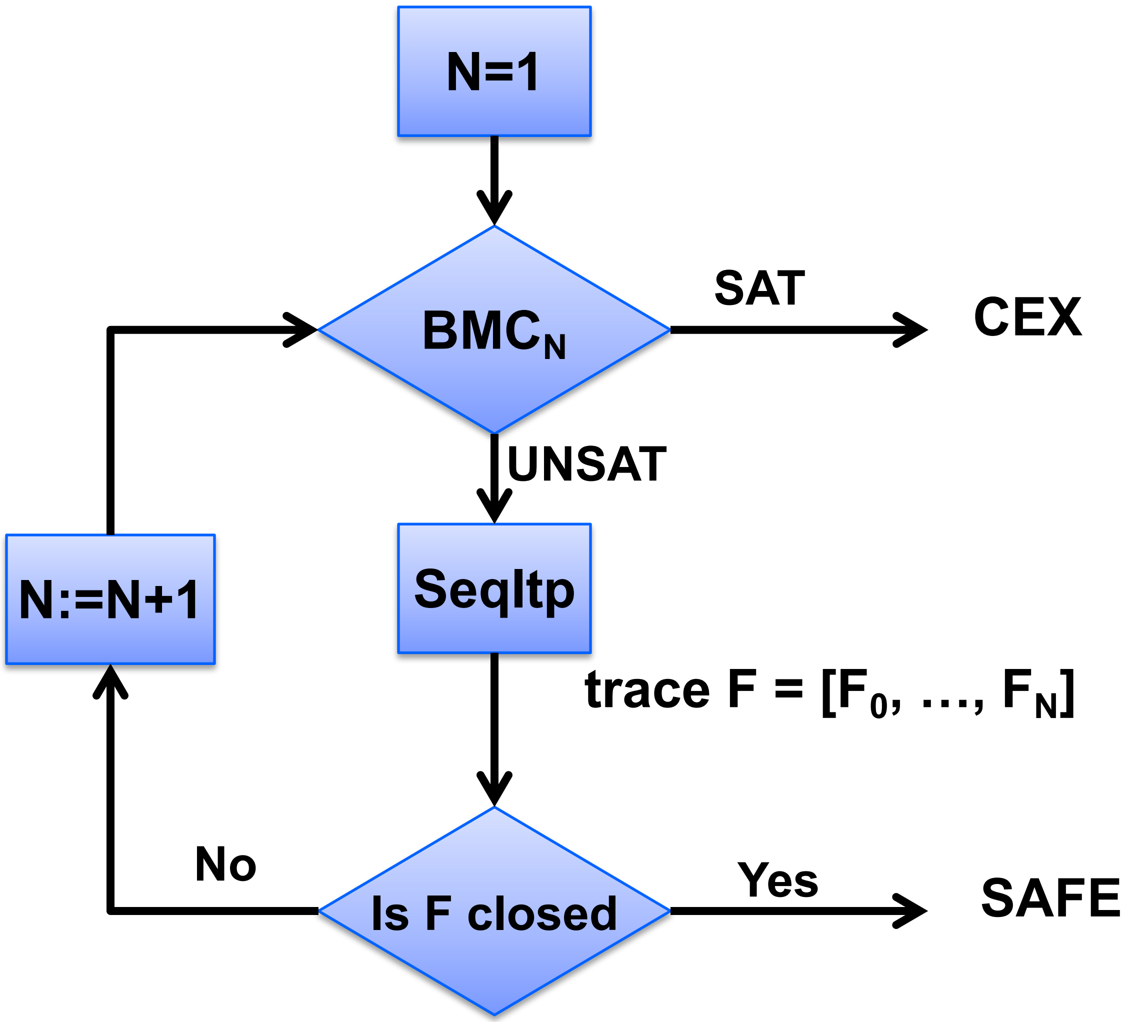
- either when BMC finds a bug: $M \not\models AGp$
- or when all reachable states have been found: $M \models AGp$

Problems:

- As the BMC formula grows – Interpolants grow
  - keep conjoining interpolants from subsequent runs: conjunctions grow

1. “Big” formulas cause the BMC problems to be hard to solve
2. Non-CNF interpolants need to be translated to CNF
3. Unrolling of TR multiplies number of variables

IMC: Interpolating Model Checking



IMC: Strength and Weaknesses

Strength

- elegant
- global bounded safety proof
- many different interpolation algorithms available
- easy to extend to SMT theories

Weaknesses

- the naïve version does not converge easily
  - interpolants are weaker towards the end of the sequence
- not incremental
  - no information is reused between BMC queries
- size of interpolants
- hard to guide

INTERPOLATION

Algorithms for Computing Interpolants

Variable Elimination by Substitution

Variable Elimination by Resolution

Optimizing using an MUS

Interpolating a resolution proof

Interpolation via Variable Elimination (1)

A(X, Y) and B(Y, Z) be two sets of clauses such that A ∧ B are UNSAT

Let I(Y) be a formula defined as follows:

$$I(Y) = \bigvee_{\vec{x} \in \mathbb{B}^n} A(\vec{x}), \text{where } n = |X|$$

Then, I(Y) is an interpolant between A and B

Pf: I(Y) = ∃ X . A(X, Y)

Question: Is that a good ITP procedure for IMC?

Interpolation via Variable Elimination (2)

A(X, Y) and B(Y, Z) be two sets of clauses such that A ∧ B are UNSAT

Recall that Res*(A, X) stands for all clauses obtained from A by exhaustively resolving on variables in X

Let I(Y) be defined as follows

$$I(Y) = \{c \in Res^*(A) \mid Vars(c) \cap X = \emptyset\}$$

Then, I(Y) is an interpolant between A and B.

Pf: I(Y) = ∃ X . A(X, Y)

Question: Is that a good ITP procedure for IMC?

UNIVERSITY OF WATERLOO

Interpolation with MUS

A(X, Y) and B(Y, Z) be two sets of clauses such that A ∧ B are UNSAT

Let U(X, Y) be a minimal subset of A(X, Y) such that U ∧ B are UNSAT

- U can be computed by iteratively querying a SAT solver
- or by examining the refutation proof of A ∧ B

Let I(Y) be an interpolant of U and B computed using either of previous methods

Then, I is an interpolant for A and B

Pf: ???

Question: Is I(Y) a good interpolant for IMC?

Alternative Definition of an Interpolant

Let F = A(x, z) ∧ B(z, y) be UNSAT, where x and y are distinct

- Note that for any assignment v to z either
  - A(x, v) is UNSAT, or
  - B(v, y) is UNSAT

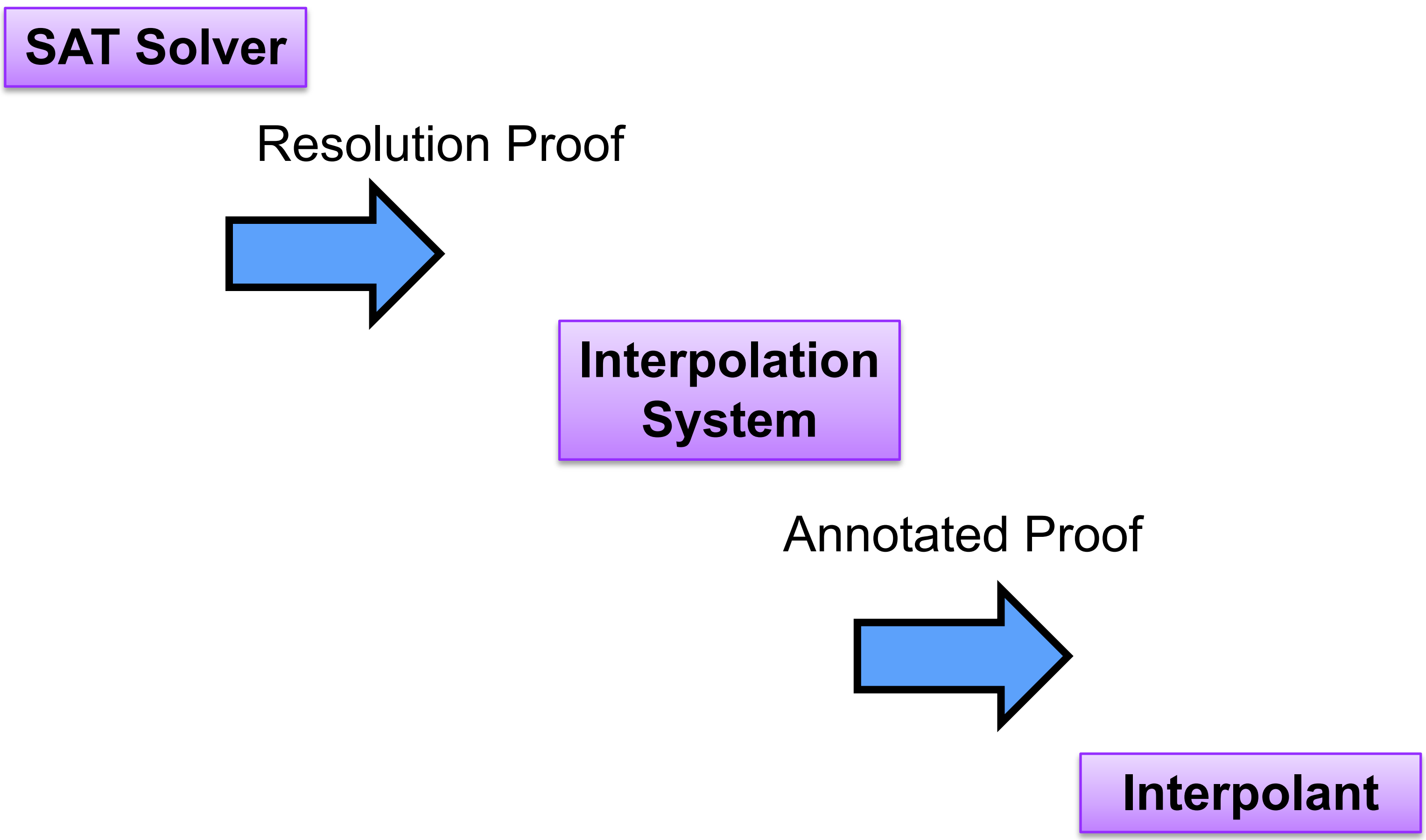
An interpolant is a circuit I(z) such that for every assignment v to z

- I(v) = A only if A(x, v) is UNSAT
- I(v) = B only if B(v, y) is UNSAT

A proof system S has a *feasible interpolation* if for every refutation π of F in S, F has an interpolant polynomial in the size of π

- propositional resolution has feasible interpolation
- extended resolution does not have feasible interpolation

Interpolants from Proof



McMillan Interpolation Procedure

Let A and B be two sets of clauses

Let *Π* be a resolution proof of A ∧ B ➔ false

Annotated clauses in *Π* with *partial interpolants* (Boolean formulae)

- Notation: c [p] mean formula *p* is a partial interpolant of clause *c*

Augmented resolution calculus, where *g(c)* is a sub-clause of only global variables

$$\frac{}{c \quad [g(c)]} \; c \in A \qquad\qquad \frac{}{c \quad [\top]} \; c \in B$$

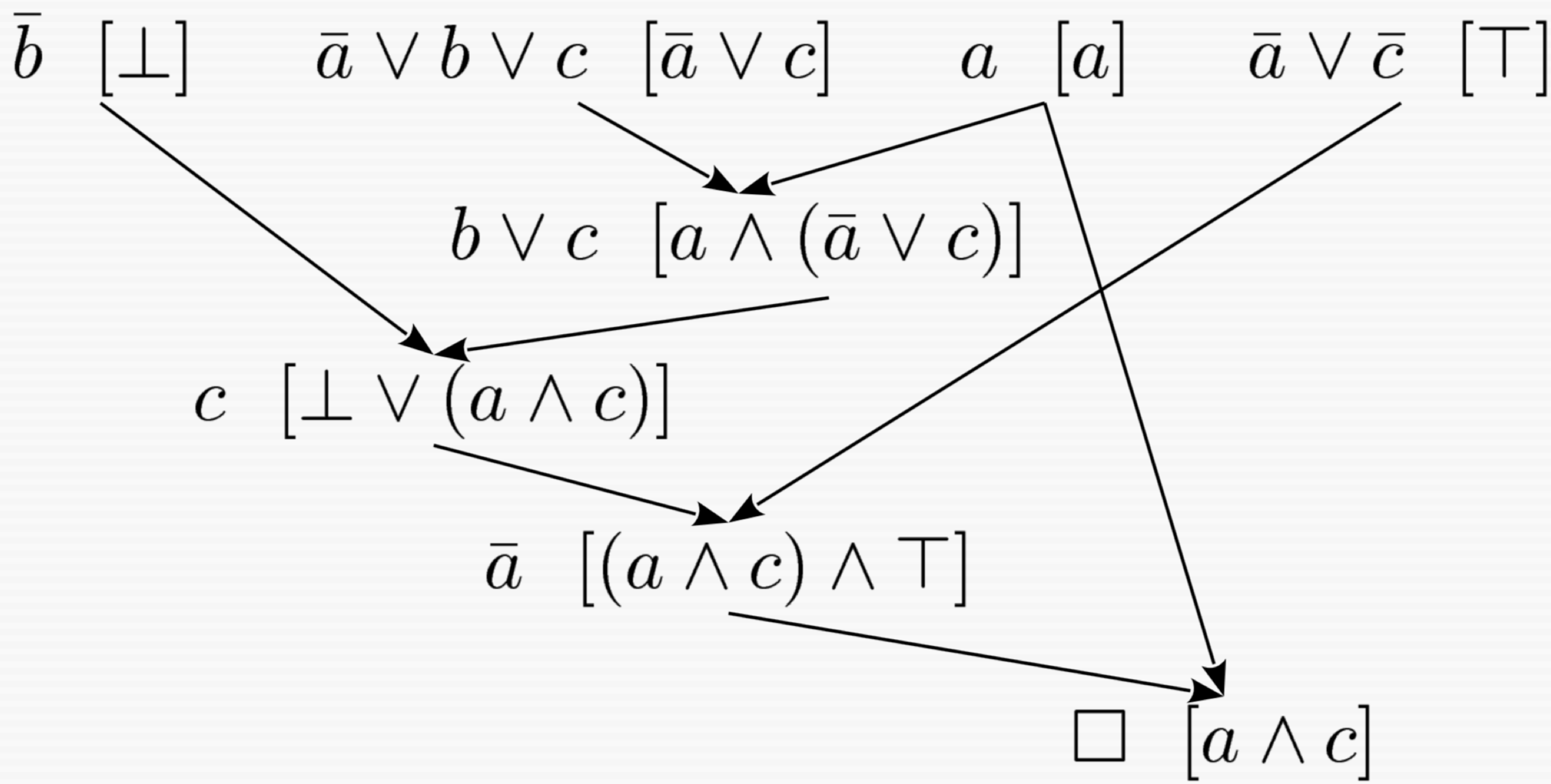
$$\frac{v \vee c \quad [I_1] \qquad \neg v \vee d \quad [I_2]}{c \vee d \quad [I_1 \vee I_2]} \; v \text{ local to } A$$

$$\frac{v \vee c \quad [I_1] \qquad \neg v \vee d \quad [I_2]}{c \vee d \quad [I_1 \wedge I_2]} \; v \text{ not local to } A$$

UNIVERSITY OF WATERLOO

Interpolation Example

$$A = \{\bar{b}, \bar{a} \vee b \vee c, a\} \qquad B = \{\bar{a} \vee \bar{c}\}$$



Correctness of McMillan Interpolation

Lemma: In any annotated proof of A and B, for every clause node *c [p_c]* the following are true

$$A \models p_c \vee (c \setminus g(c))$$

$$B, p_c \models g(c)$$

$$p_c \text{ only contains global symbols}$$

Corollary: The root of resolution proof is the empty clause, and its partial interpolant is the interpolant!

Other Interpolation Systems

A single resolution proof can be annotated in different ways giving different interpolants

McMillan interpolation is the strongest interpolant obtained by annotated proof rules from a given proof

There are other annotation strategies and systems for interpolation

- Symmetric interpolants: ITP(A,B) == ¬ ITP(B, A)
- Labelled interpolation: framework in which McMillan ITP is an instance

So far, no correlation between strength / technique and usefulness for verification ☺

Computing Sequence Interpolant

Let S_0, S_1, S_2, …, S_n be n formulas whose conjunction is UNSAT
Let ITP(A,B) be any interpolation algorithm

Then, the sequence I_0, I_1, … is a sequence interpolant

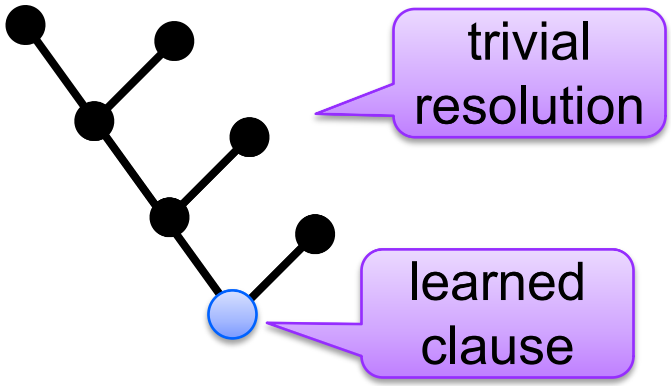
$$I_0 = ITP(S_0, \bigwedge_{i=1..n} S_i)$$

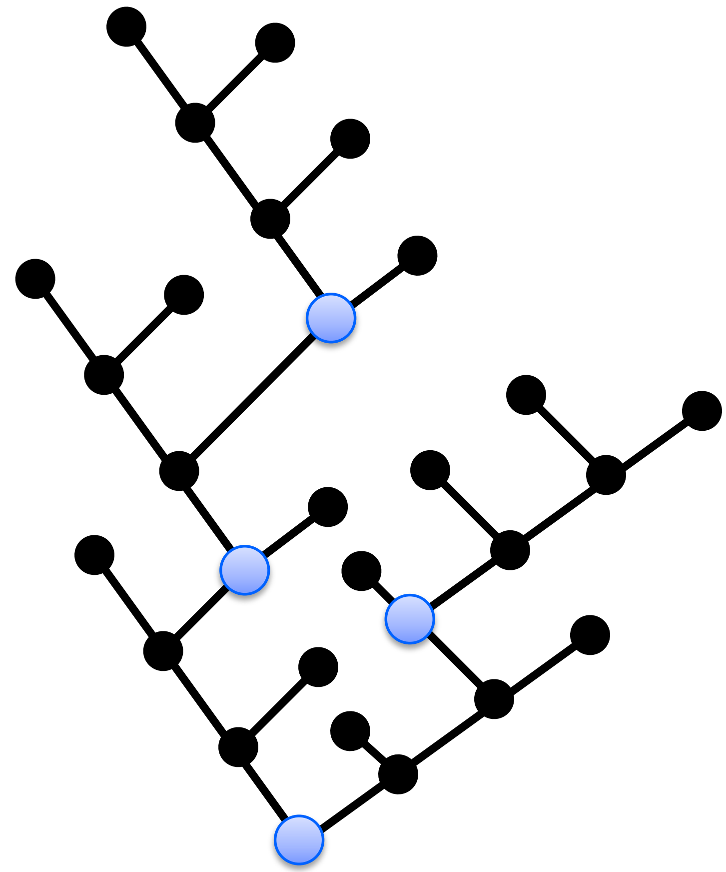
$$I_1 = ITP(I_0 \wedge S_1, \bigwedge_{i=2..n} S_i)$$

$$I_k = ITP(I_{k-1} \wedge S_k, \bigwedge_{i=k..n} S_i), \text{for } k \in [1..(n-1)]$$

UNIVERSITY OF WATERLOO

DRUPing for Interpolants





A CDCL proof is build out of trivial resolutions

- terminated by a learned clause

A sub-proof for each learned clause can be re-constructed in polynomial time

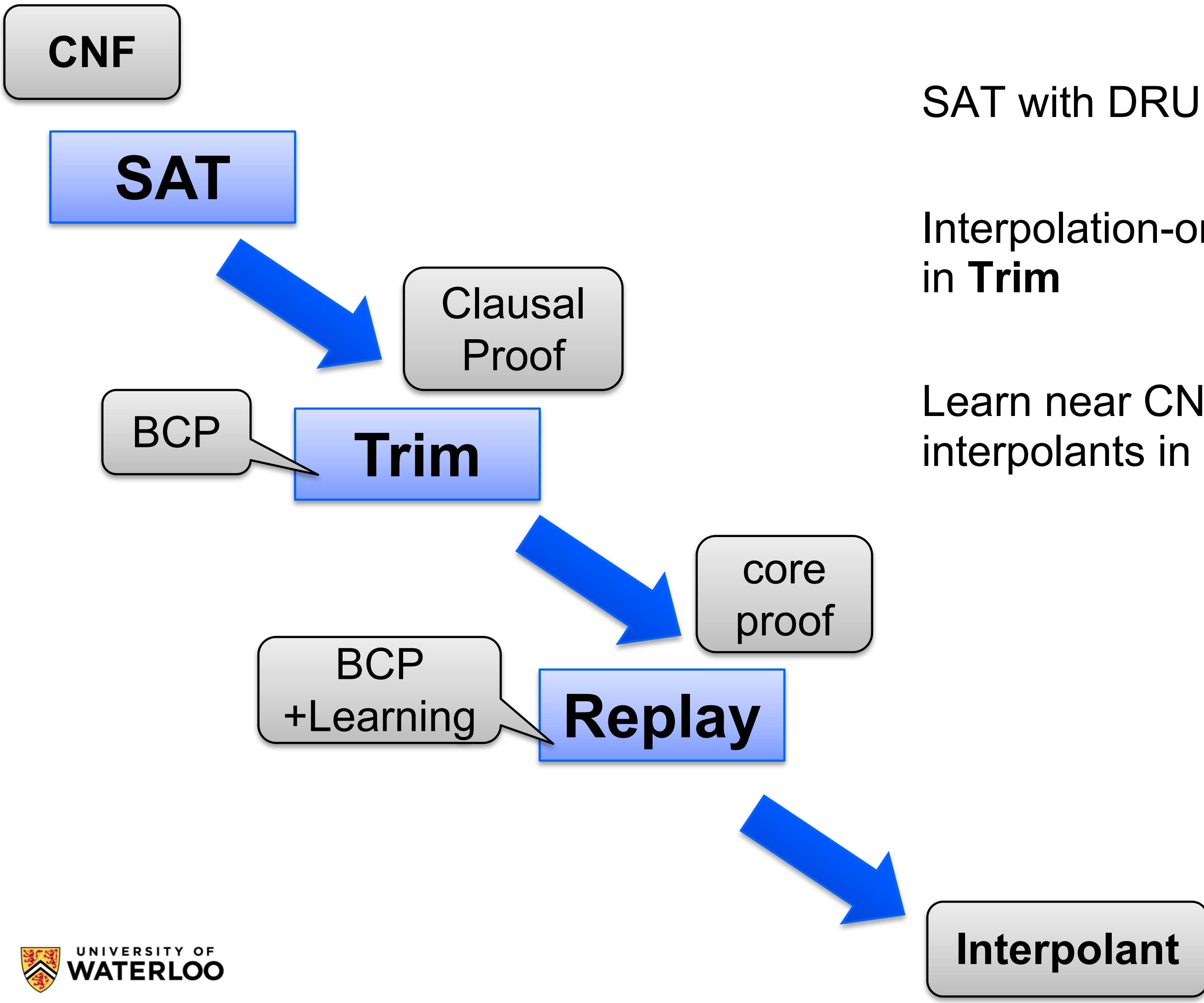
- negation of clause + BCP leads to a conflict

A clausal proof is a sequence of learned clauses in the order they are learned

Interpolate while replaying the proof

A. Gurfinkel & Y. Vizel: DRUPing for interpolats. FMCAD 2014: 99-106

MiniDRUP



SAT with DRUP proofs

Interpolation-oriented BCP in **Trim**

Learn near CNF interpolants in **Replay**