

Unbounded Model Checking: IMC and ITP

Automated Program Verification (APV)
Fall 2019

Prof. Arie Gurfinkel



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 $P = (V, \text{Init}, \text{Tr}, \text{Bad})$

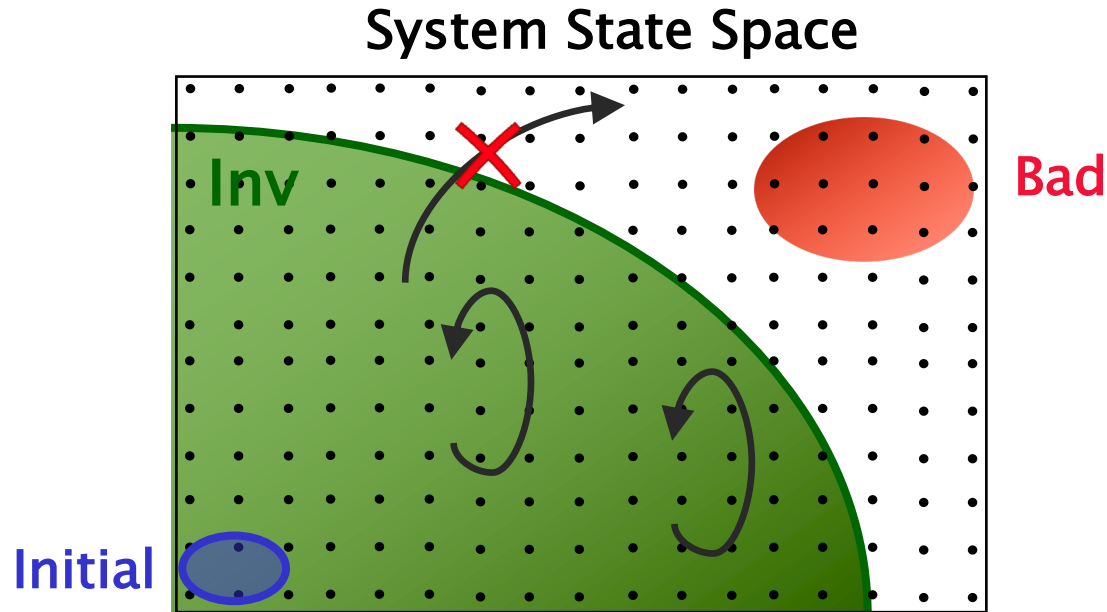
P is UNSAFE if and only if there exists a number N s.t.

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

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

$$\left. \begin{array}{l} \text{Init} \Rightarrow \text{Inv} \\ \text{Inv}(X) \wedge \text{Tr}(X, X') \Rightarrow \text{Inv}(X') \\ \text{Inv} \Rightarrow \neg \text{Bad} \end{array} \right\} \begin{array}{l} \text{Inductive} \\ \text{Safe} \end{array}$$

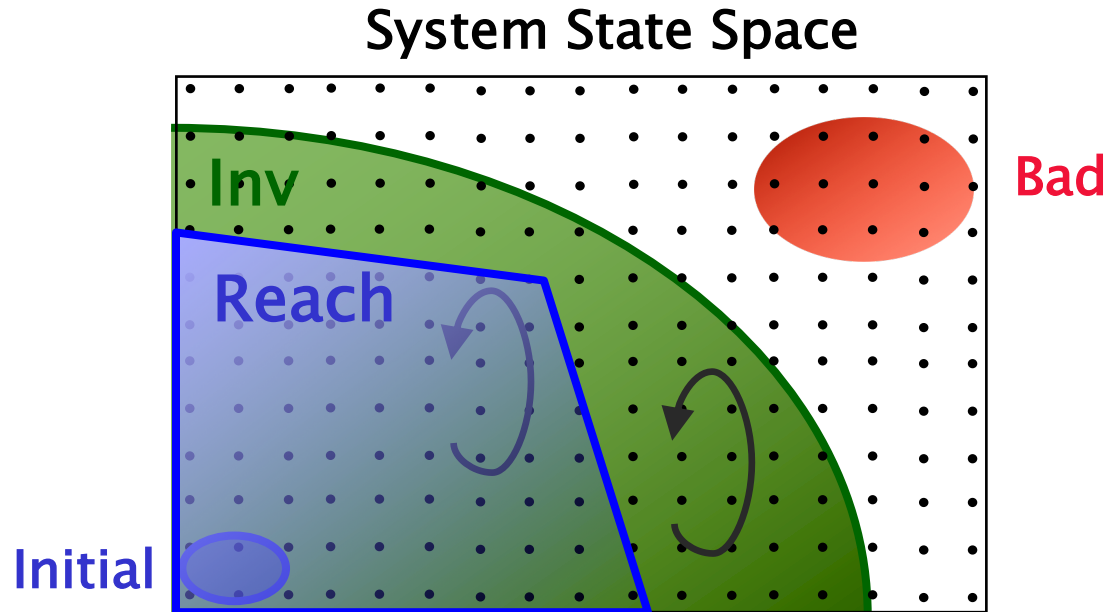
Inductive Invariants



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

- **Initiation:** $\text{Initial} \subseteq \text{Inv}$
- **Safety:** $\text{Inv} \cap \text{Bad} = \emptyset$
- **Consecution:** $\text{TR}(\text{Inv}) \subseteq \text{Inv}$ i.e., if $s \in \text{Inv}$ and $s \rightsquigarrow t$ then $t \in \text{Inv}$

Inductive Invariants



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

- **Initiation:** $\text{Initial} \subseteq \text{Inv}$
- **Safety:** $\text{Inv} \cap \text{Bad} = \emptyset$
- **Consecution:** $\text{TR}(\text{Inv}) \subseteq \text{Inv}$ i.e., if $s \in \text{Inv}$ and $s \rightsquigarrow t$ then $t \in \text{Inv}$

System S is safe if $\text{Reach} \cap \text{Bad} = \emptyset$

Forward Reachability Analysis

Does AG P hold?

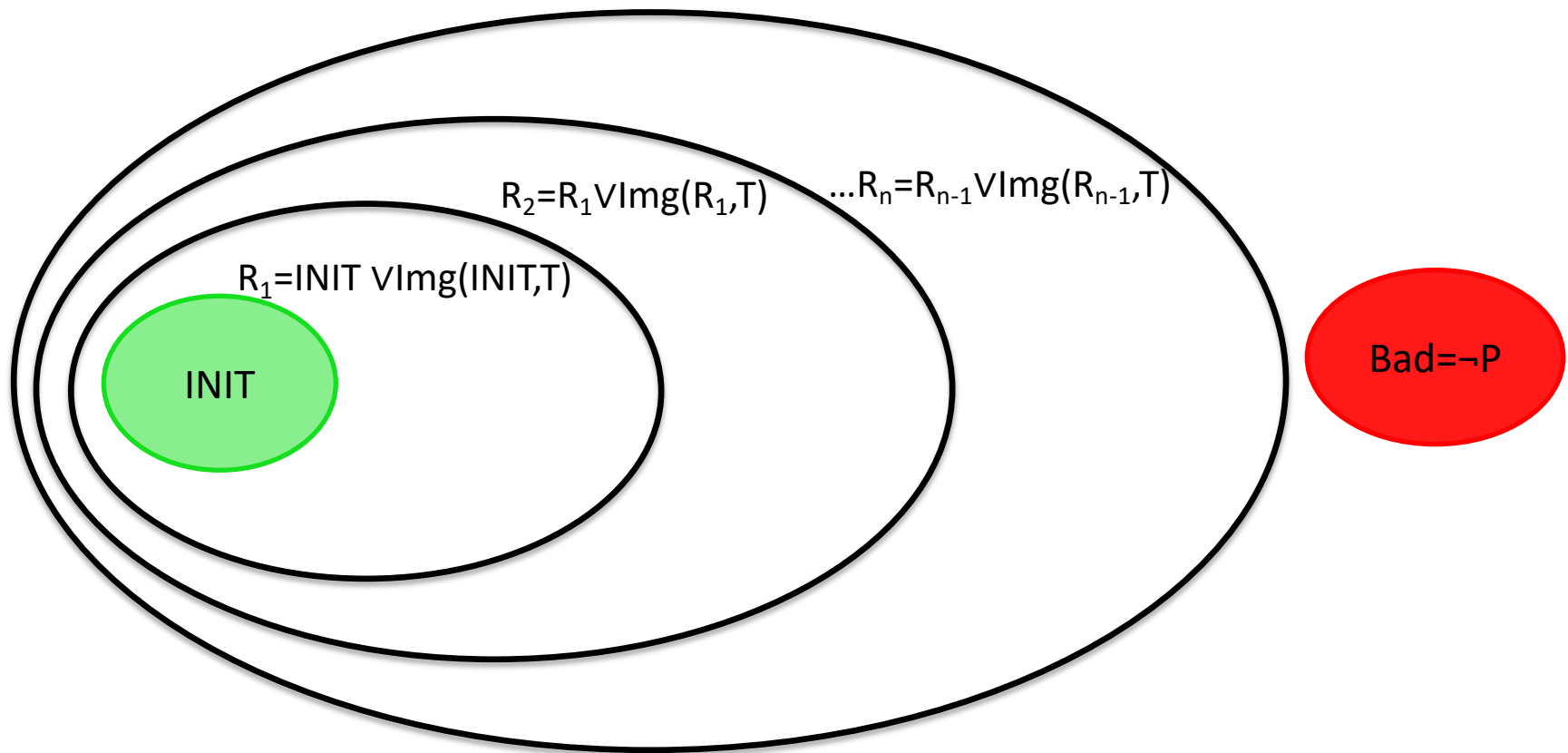


Image operator: $\text{Img}(Q, T) = \exists V. (Q \wedge T)$

Termination when

- either a bad state satisfying $\neg p$ is found
- or a **fixpoint** is reached: $R_j \subseteq \bigcup_{i=0, j-1} R_i$
 $\Rightarrow R_j$ is the set of reachable states

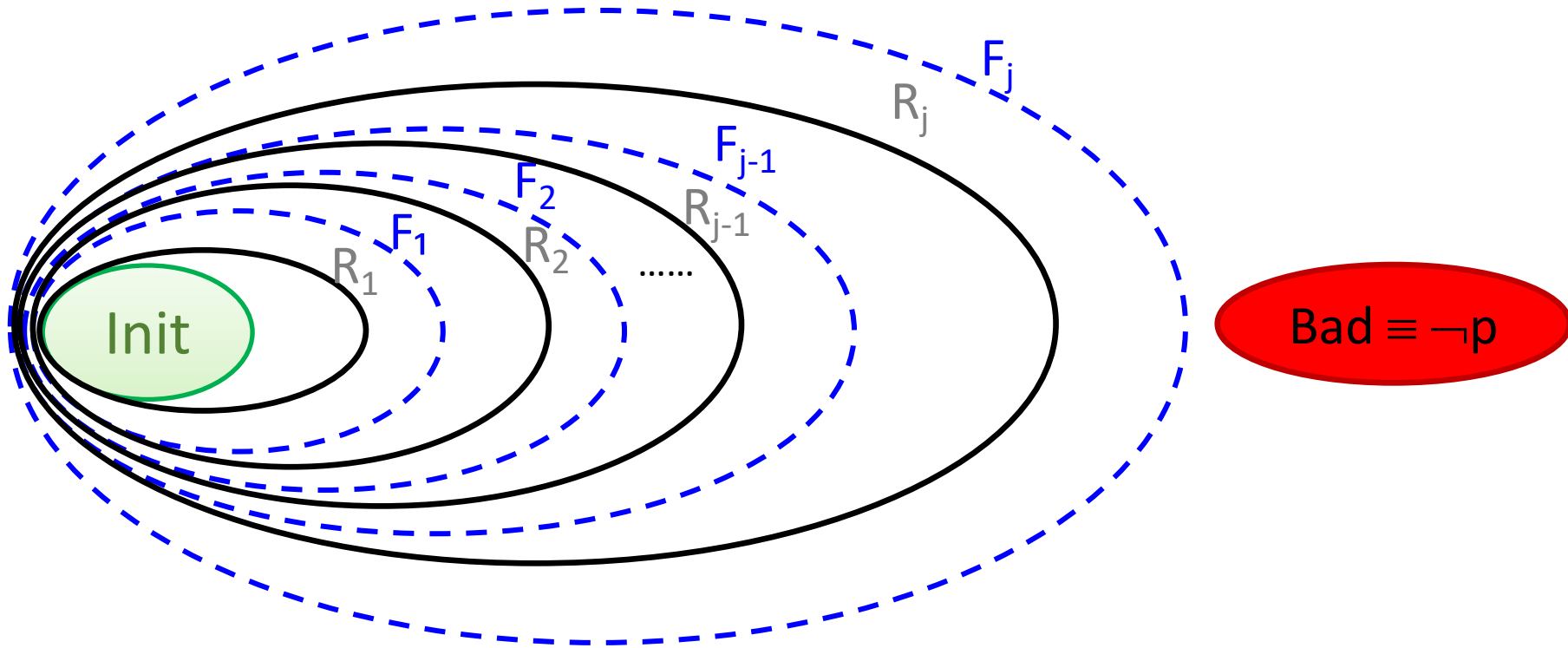
Image computation methods

- Symbolic model checking without BDD's
 - Use SAT solver just for fixed-point detection
 - Abdulla, Bjessse 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 , $\text{Img}(Q,T)$ implies $\text{Img}'(Q,T)$
- Over-approximate reachability:

$$F_0 = I$$

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

$$F = \bigcup F_i$$

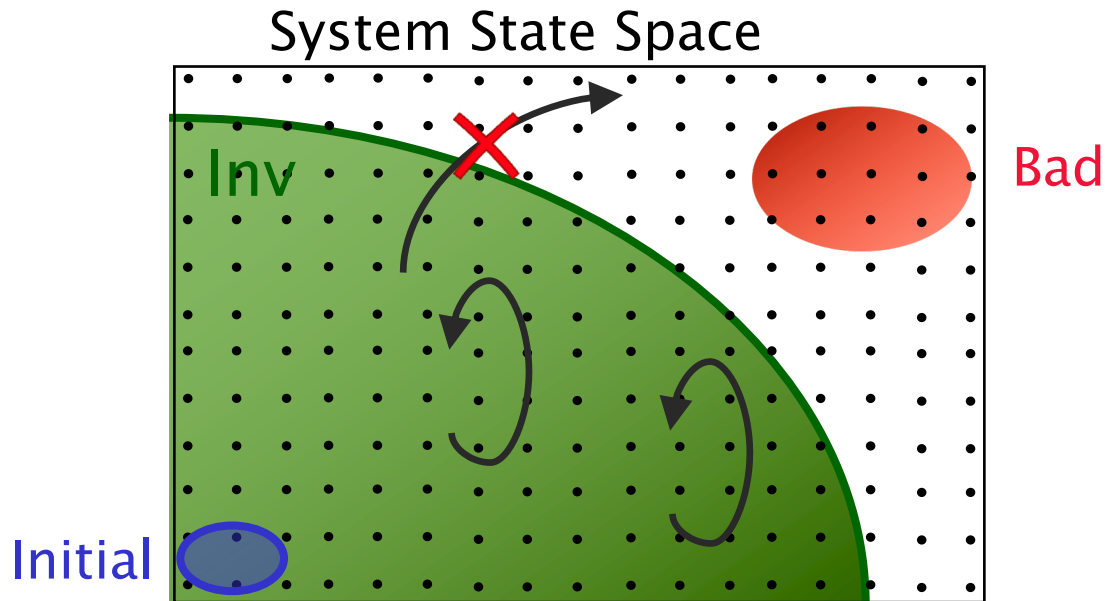
Fixpoint:

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

Inductive Invariants for verifying $\text{AG } p$

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

- Initiation: $\text{Initial} \subseteq \text{Inv}$
- Safety: $\text{Inv} \cap \text{Bad} = \emptyset$
- Consecution: $\text{TR}(\text{Inv}) \subseteq \text{Inv}$ i.e., if $s \in \text{Inv}$ and $s \rightsquigarrow t$ then $t \in \text{Inv}$

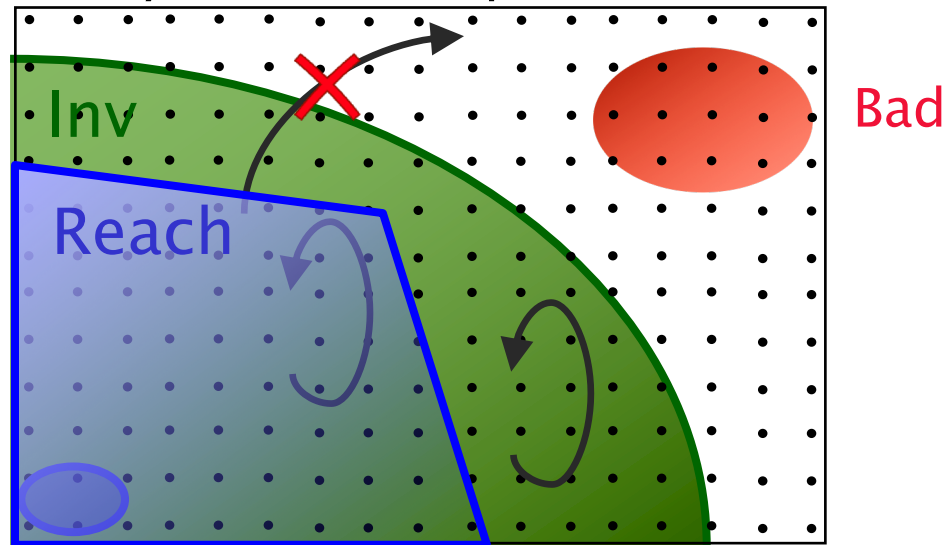


Inductive Invariants for verifying $\text{AG } p$

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

- Initiation: $\text{Initial} \subseteq \text{Inv}$
- Safety: $\text{Inv} \cap \text{Bad} = \emptyset$
- Consecution: $\text{TR}(\text{Inv}) \subseteq \text{Inv}$ i.e., if $s \in \text{Inv}$ and $s \rightsquigarrow t$ then $t \in \text{Inv}$

System State Space



$\text{Reach} \cap \text{Bad} = \emptyset$

Initial

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

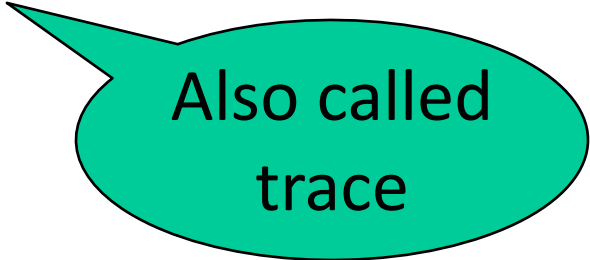
Why is F an inductive invariant?

Recall: forward reachability sequence

$$F_0 = I$$

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

$$F = \bigcup F_i$$



Also called
trace

How to compute an approximate image for reachability analysis?

Adequacy of Approximate Img

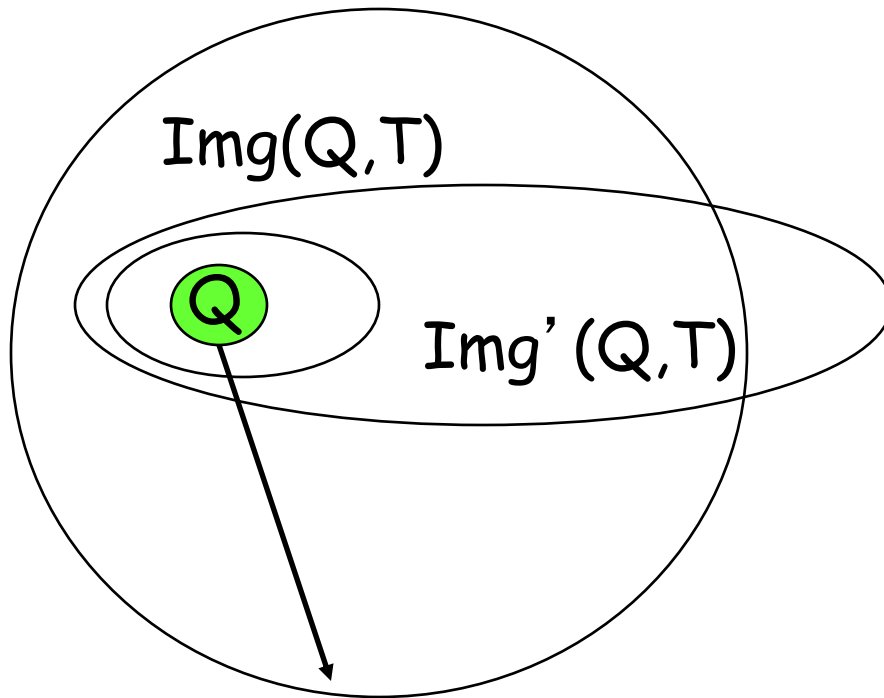
Img' is adequate w.r.t. Bad, when

- if Q cannot reach Bad in any number of steps, then $\text{Img}'(Q, T)$ cannot reach Bad in any number of steps

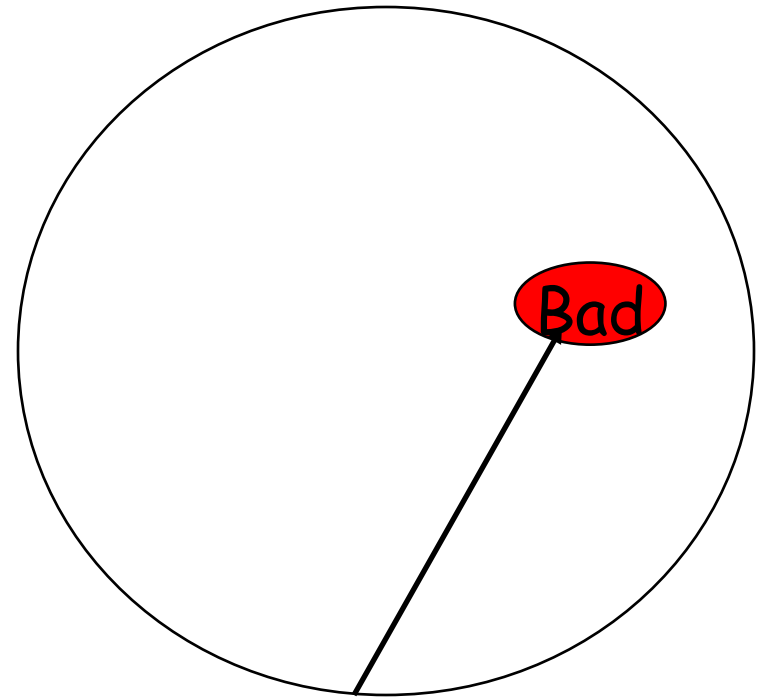
If Img' is adequate, then

- Bad is reachable iff $F \wedge \text{Bad}$ is SAT

Adequate image



Reached from Q



Can reach Bad

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,
 $\text{Img}'(Q,T)$ cannot reach Bad **within k steps**
- Note, if $k > \text{diameter}$, 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 $P = (V, \text{Init}, \text{Tr}, \text{Bad})$ is a sequence of formulas $[F_0, \dots, F_N]$ such that

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

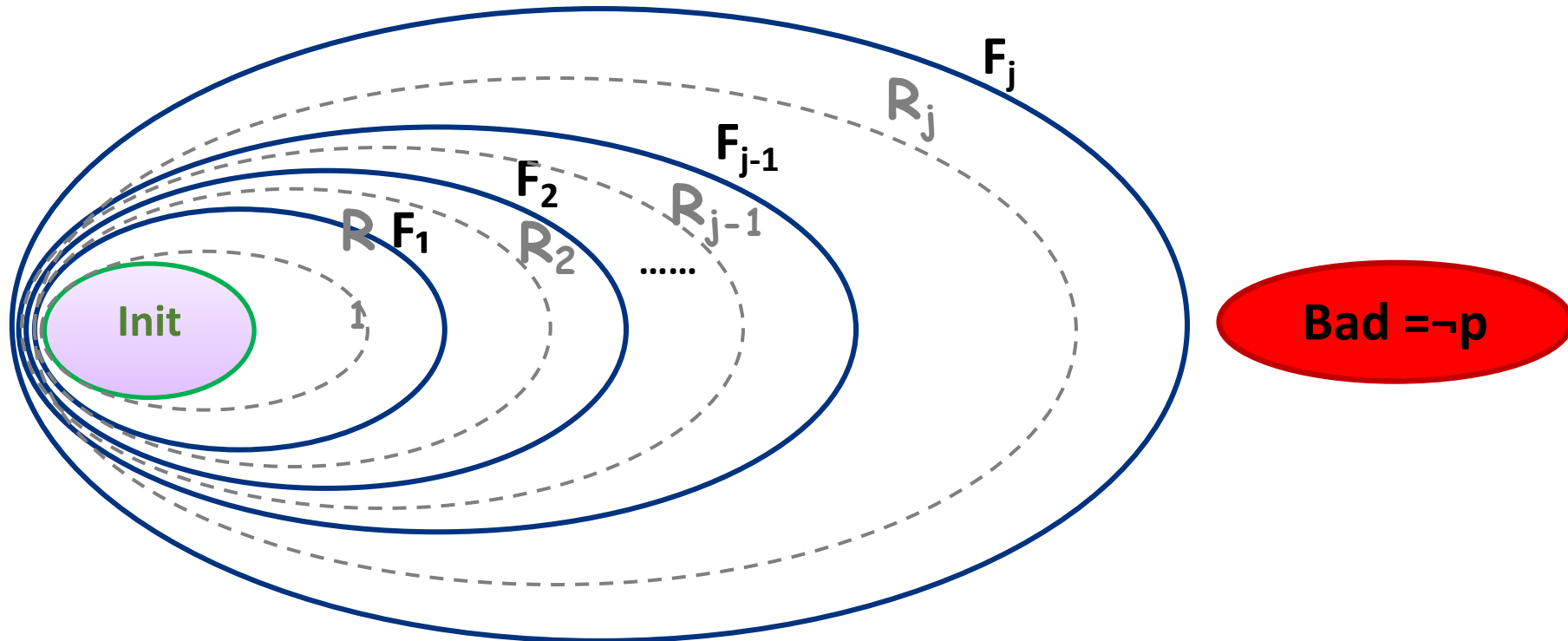
A trace is *safe* iff $\forall 0 \leq i \leq N, F_i \Rightarrow \neg \text{Bad}$

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

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

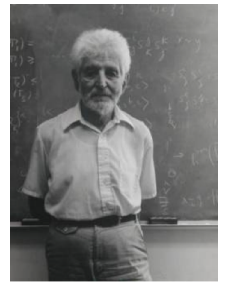
A transition system P is *SAFE* iff it admits a safe closed trace

Safe Monotone Inductive Trace



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

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., $A \Rightarrow \neg B$

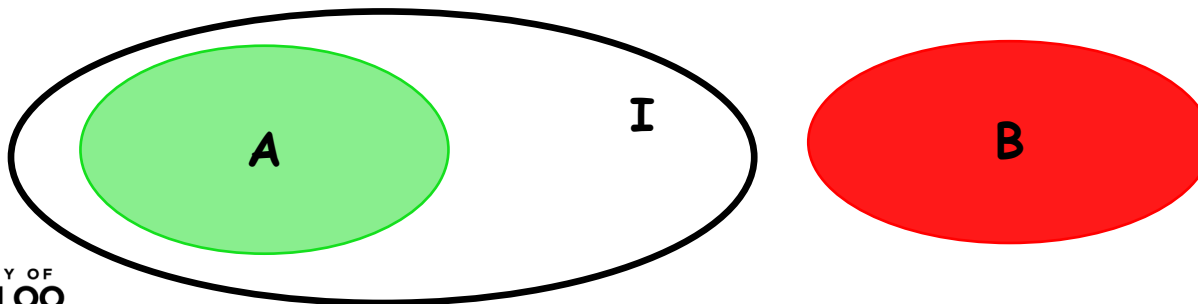
There exists a formula I such that:

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

$$A \Rightarrow \neg B$$

$$A \Rightarrow I$$

$$I \Rightarrow \neg B$$



Example

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

$$I = ?$$

Interpolants from Resolution Proofs

When $A \wedge B$ is unsatisfiable, SAT solvers return 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

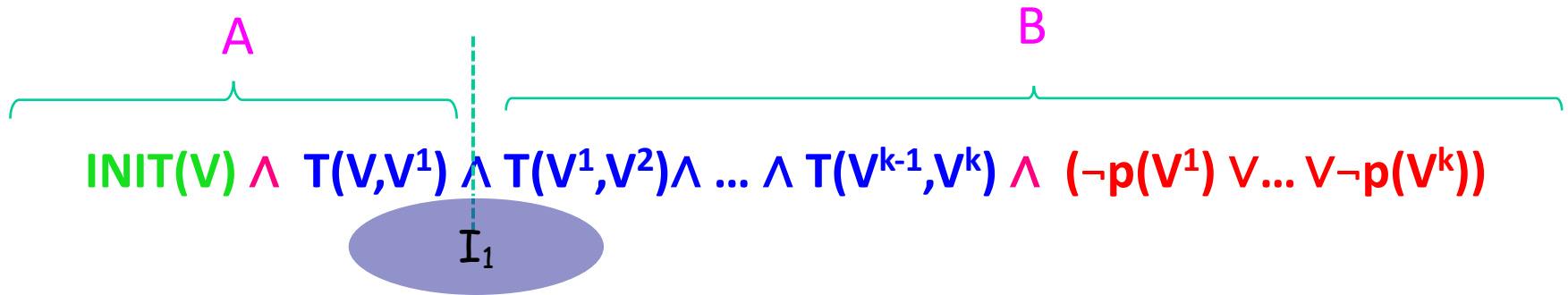
$\text{ITP}(A, B)$ = resulting interpolant

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:

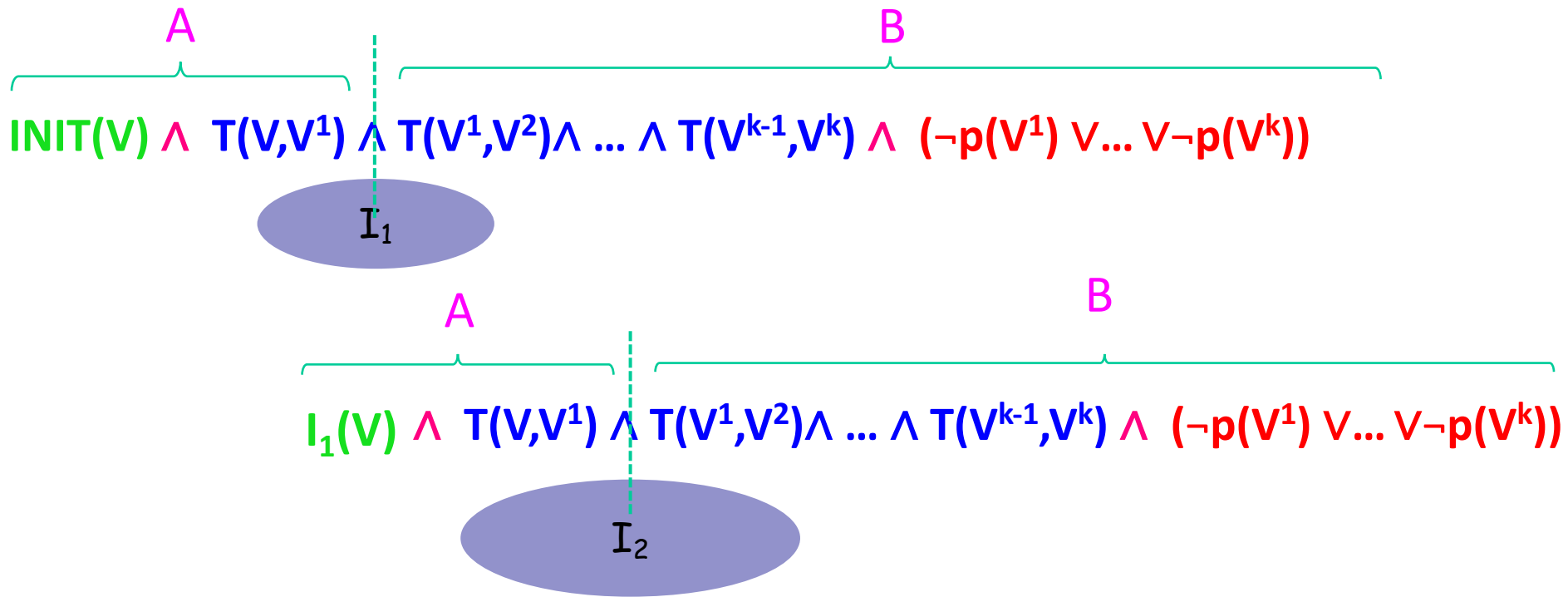


- Interpolant I_1 is computed
 - over-approximates the states reachable from INIT in 1 transition
 - cannot reach a bad state in $\leq k-1$ transitions

k-1-adequate overapprox. image!

*only partial – why?

IMC – Interpolation-based MC



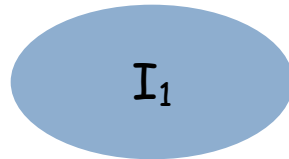
- I_1 is fed back to the BMC solver
- A new interpolant I_2 is computed
 - I_2 over-approximates the states reachable from INIT in 2 transitions
 - cannot reach a bad state in $\leq k-1$ transitions
- Iterative process

IMC – Interpolation-based MC

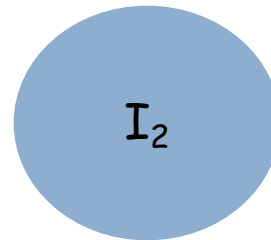
- In IMC, **short** BMC formulas can prove the nonexistence of **long** CEXs
 - INIT is replaced by I_j which over-approximates R_j
- If a satisfying assignment to $I_j(V) \wedge T(V, V^1) \wedge T(V^1, V^2) \wedge \dots \wedge T(V^{k-1}, V^k) \wedge (\neg p(V^1) \vee \dots \vee \neg p(V^k))$ is found, 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)

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



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

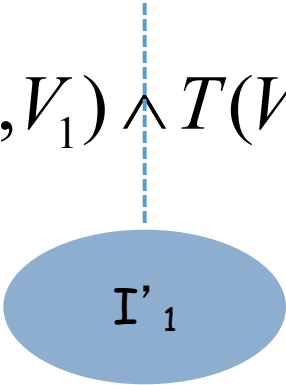


$$I_2(V_0) \wedge T(V_0, V_1) \wedge \neg p(V_1)$$



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

$$I'_1(V_0) \wedge T(V_0, V_1) \wedge T(V_1, V_2) \wedge (\neg q(V_1) \vee \neg q(V_2))$$

▪

▪

▪

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

- 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**
 - $I_n \Rightarrow \text{INIT} \vee \bigvee_{j=1, n-1} I_j$

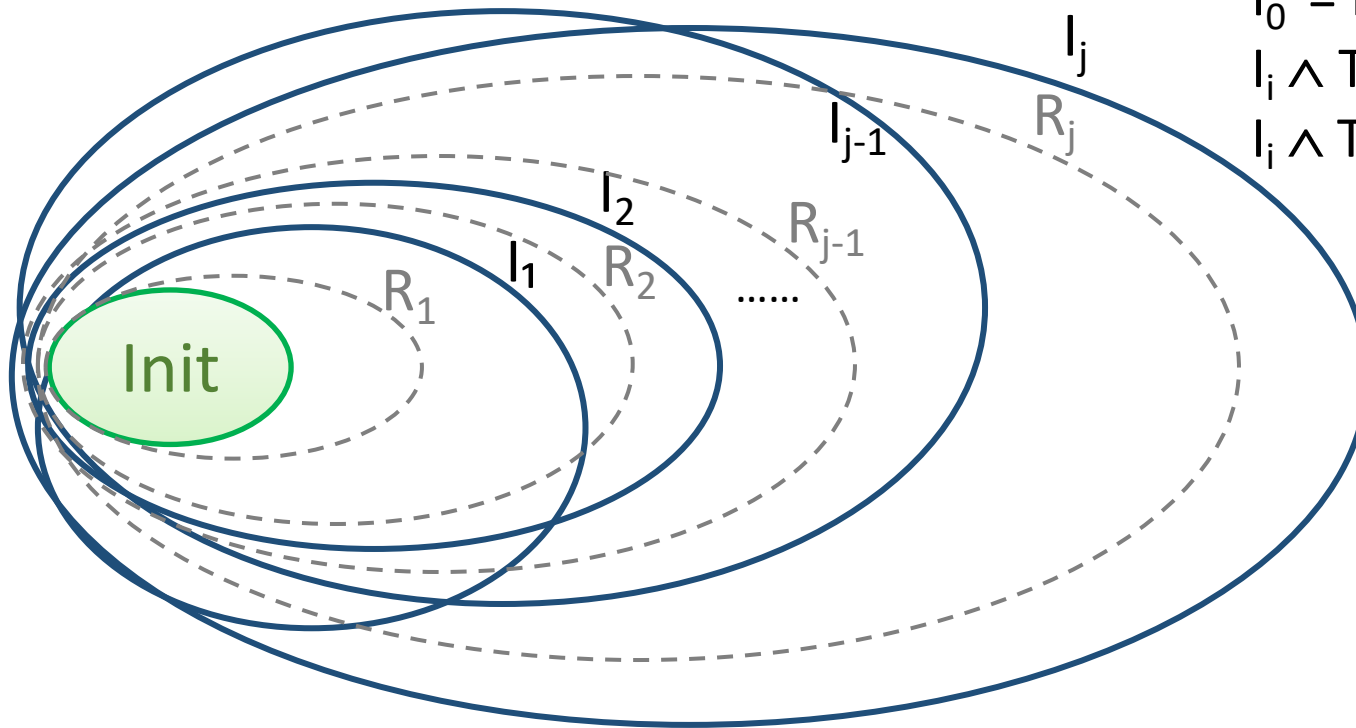
IMC as Approximate Forward Reachability (1)

Inductive trace:

$$I_0 = \text{Init}$$

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

$$I_i \wedge \text{TR}^j \Rightarrow p_j \text{ for } j=0 \dots k-1$$



Bad $\equiv \neg p$

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

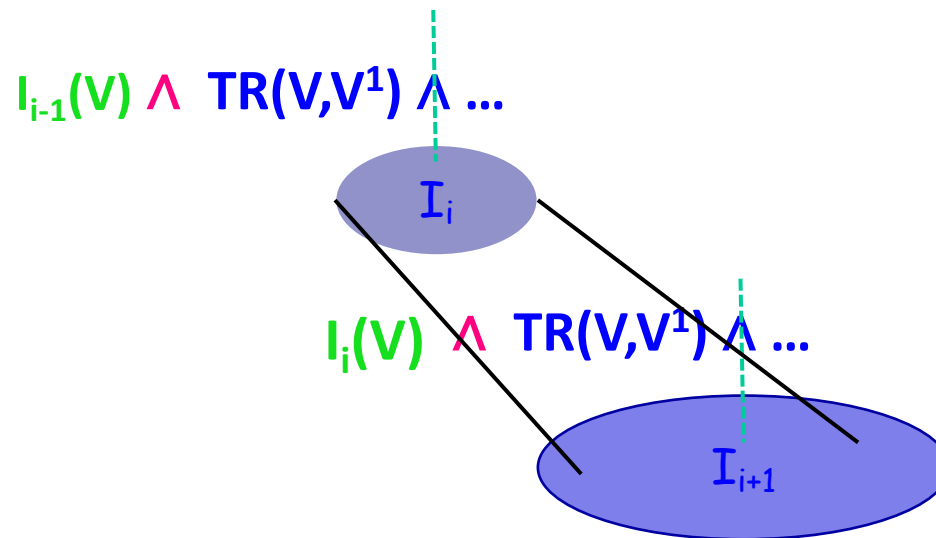
Fixpoint: If $I_j \Rightarrow \text{INIT} \vee I_1 \vee \dots \vee I_{j-1}$ then $\text{INIT} \vee I_1 \vee \dots \vee I_{j-1}$ is an inductive invariant

Inductive Invariants in IMC

Claim: If $I_j \Rightarrow \text{INIT} \vee I_1 \vee \dots \vee I_{j-1}$ then $\text{INIT} \vee I_1 \vee \dots \vee I_{j-1}$ is an ind. invariant

Proof:

Consecution: $(\text{INIT} \vee I_1 \vee \dots \vee I_{j-1}) \wedge \text{TR} \Rightarrow (I_1 \vee I_2 \vee \dots \vee I_j)' \Rightarrow (\text{INIT} \vee I_1 \vee \dots \vee I_{j-1})'$



Initiation: $\text{INIT} \Rightarrow \text{INIT} \vee I_1 \vee \dots \vee I_{j-1}$

Safety: $\text{INIT} \vee I_1 \vee \dots \vee I_{j-1} \Rightarrow p$

IMC Pseudocode (1)

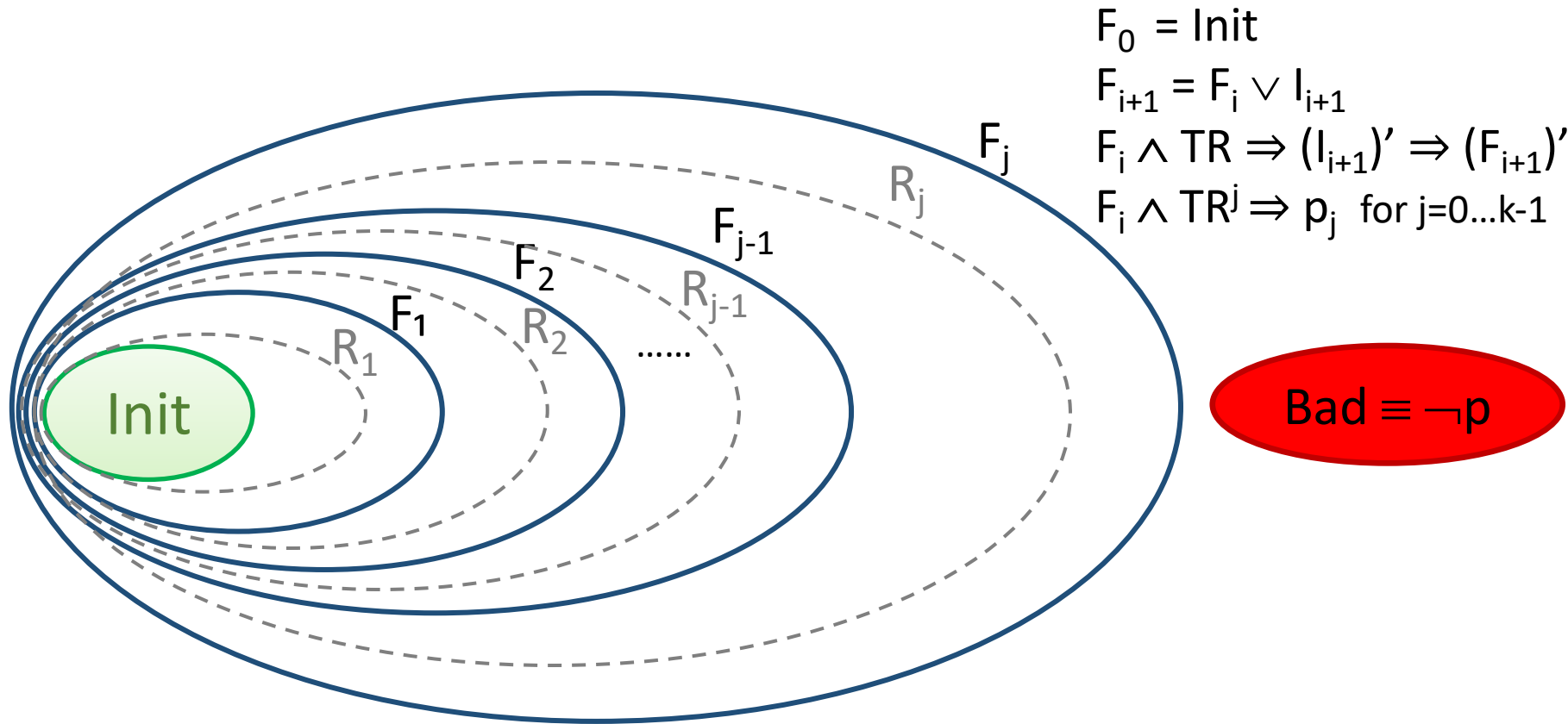
```
k=1
while (true) {
  j = 0
  I0 = INIT
  while(true) {
    if (SAT(Ij ∧ TRk ∧ (¬p1 ∨ ... ∨ ¬pk)) {
      if (j==0) then return CEX
      k++; break;
    } else // UNSAT
      Ij+1 = ITP(Ij ∧ TR, TRk-1 ∧ (¬p1 ∨ ... ∨ ¬pk));
      if (Ij+1 => INIT ∨ I1 ∨ ... ∨ Ij) then return SAFE
      j++;
    }
  }
}
```

IMC Pseudocode (2)

McMillan 2003

```
k=1
while (true) {
  I = INIT
  while(true) {
    if (SAT(I  $\wedge$  TRk  $\wedge$  ( $\neg$ p1 $\vee$ ...  $\vee$  $\neg$ pk))) {
      if (I = INIT) then return CEX
      k++; break;
    } else // UNSAT
      I' = ITP(I  $\wedge$  TR, TRk-1  $\wedge$  ( $\neg$ p1 $\vee$ ...  $\vee$  $\neg$ pk));
      if (I'  $\Rightarrow$  I) then return SAFE
      I = I  $\vee$  I';
  }
}
```

Approximate Reachability Sequence in IMC (2)



- 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 $k \ll d$

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

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

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

In particular:

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

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

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

Interpolation Sequence

- If $A_1 \wedge \dots \wedge A_{k+1}$ is **unsatisfiable**, then there exists an **interpolation-sequence** $l_0, l_1, \dots, l_k, l_{k+1}$ for $(A_1, \dots, A_{k+1})$ s.t.:

$$l_0 = T \text{ and } l_{k+1} = F$$

$$l_j \wedge A_{j+1} \Rightarrow l_{j+1}$$

In particular:

$$A_1 \Rightarrow l_1 \quad l_k \Rightarrow \neg A_{k+1}$$

l_j - over **common** variables of $A_1, \dots, A_j$ and $A_{j+1}, \dots, A_k$

$$\begin{array}{cccccc} \text{Init} \wedge \text{Tr}_1 & \text{Tr}_2 & \text{Tr}_3 & \text{Tr}_4 & \text{Tr}_5 & \neg p \\ l_1 & l_2 & l_3 & l_4 & l_5 & \end{array}$$

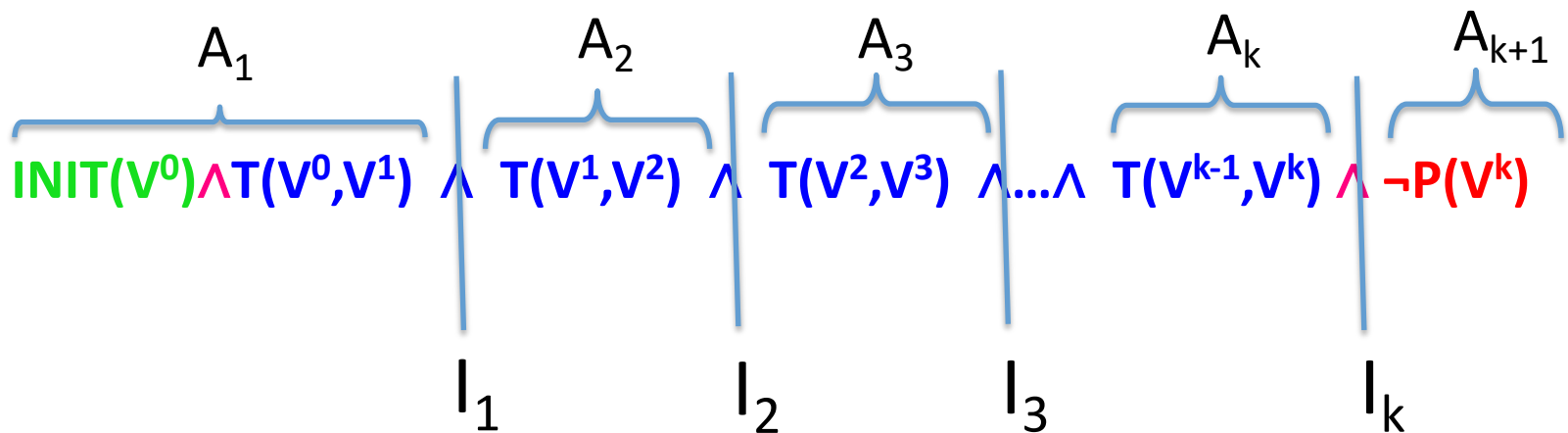
Computed by pairwise interpolation applied to different cuts of a fixed resolution proof

- All l_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:

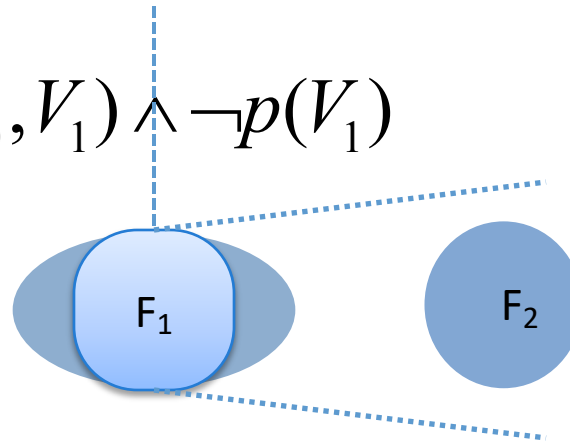


$$I_j \wedge A_{j+1} \Rightarrow I_{j+1}$$

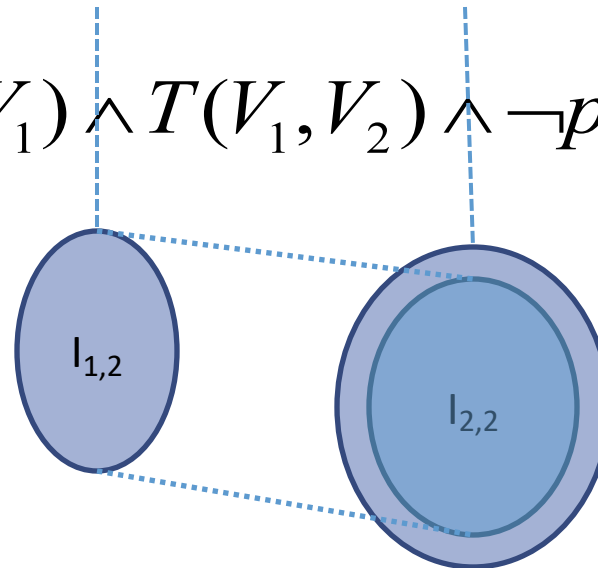
I_j - over **common** variables of $A_1, \dots, A_j$ and $A_{j+1}, \dots, A_{k+1}$

Using Interpolation Sequence

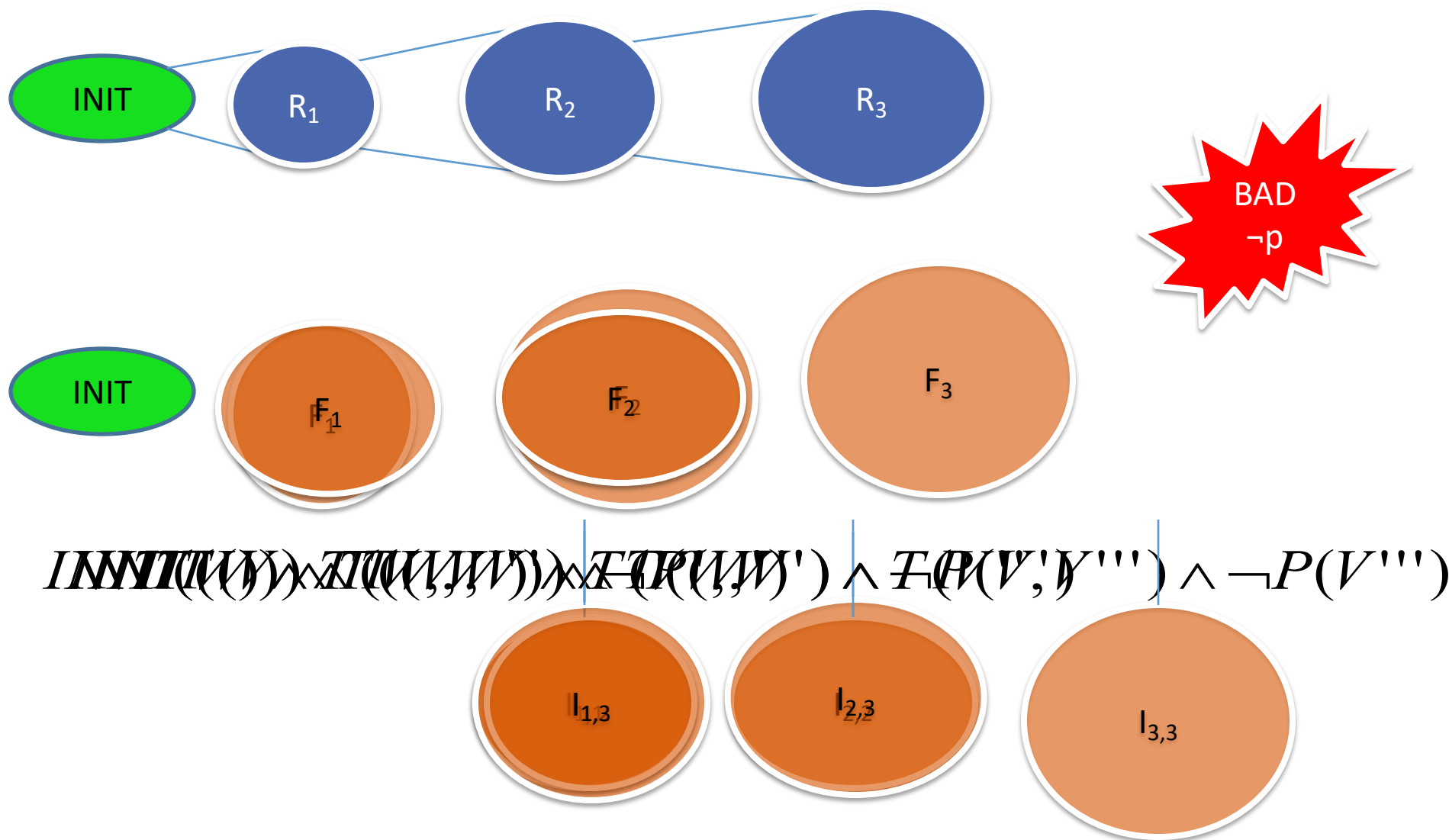
$$INIT(V_0) \wedge T(V_0, V_1) \wedge \neg p(V_1)$$



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



Analogy to Forward Reachability



Reachability with Interpolation-Sequence

- BMC is used for **bug finding**
- Interpolation-sequence computes an **inductive trace**:
 $\langle F_0, F_1, \dots, F_k \rangle$ 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 $F_n \Rightarrow \bigvee_{j=1 \dots n-1} F_j$?
- Similar to checking fixpoint in forward reachability analysis:

$$R_k \subseteq \bigcup_{j=1 \dots k-1} R_j$$

- 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

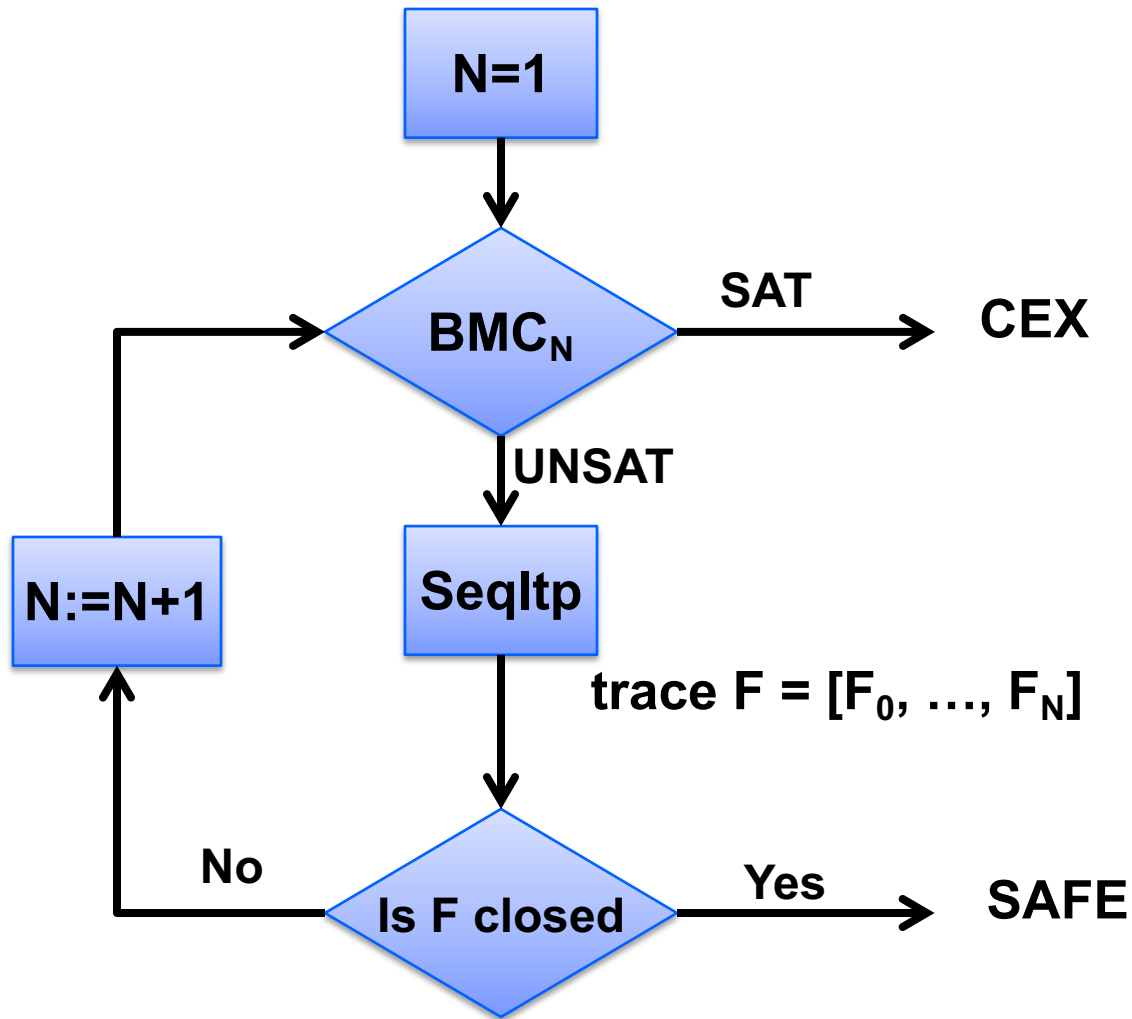
Always terminates

- 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

Alternative Definition of an Interpolant

Let $F = A(x, z) \wedge B(z, y)$ 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

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 $A \wedge B$ 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: $I(Y) = \exists X . A(X, Y)$

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 $A \wedge B$ are UNSAT

Recall that $\text{Res}^*(A, X)$ stands for all clauses obtained from A by exhaustively resolving on variables in X

Let $I(Y)$ be defined as follows

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

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

Pf: $I(Y) = \exists X . A(X, Y)$

Question: Is that a good ITP procedure for IMC?

Interpolation with MUS

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

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

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

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 $F = A(x, z) \wedge B(z, y)$ 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

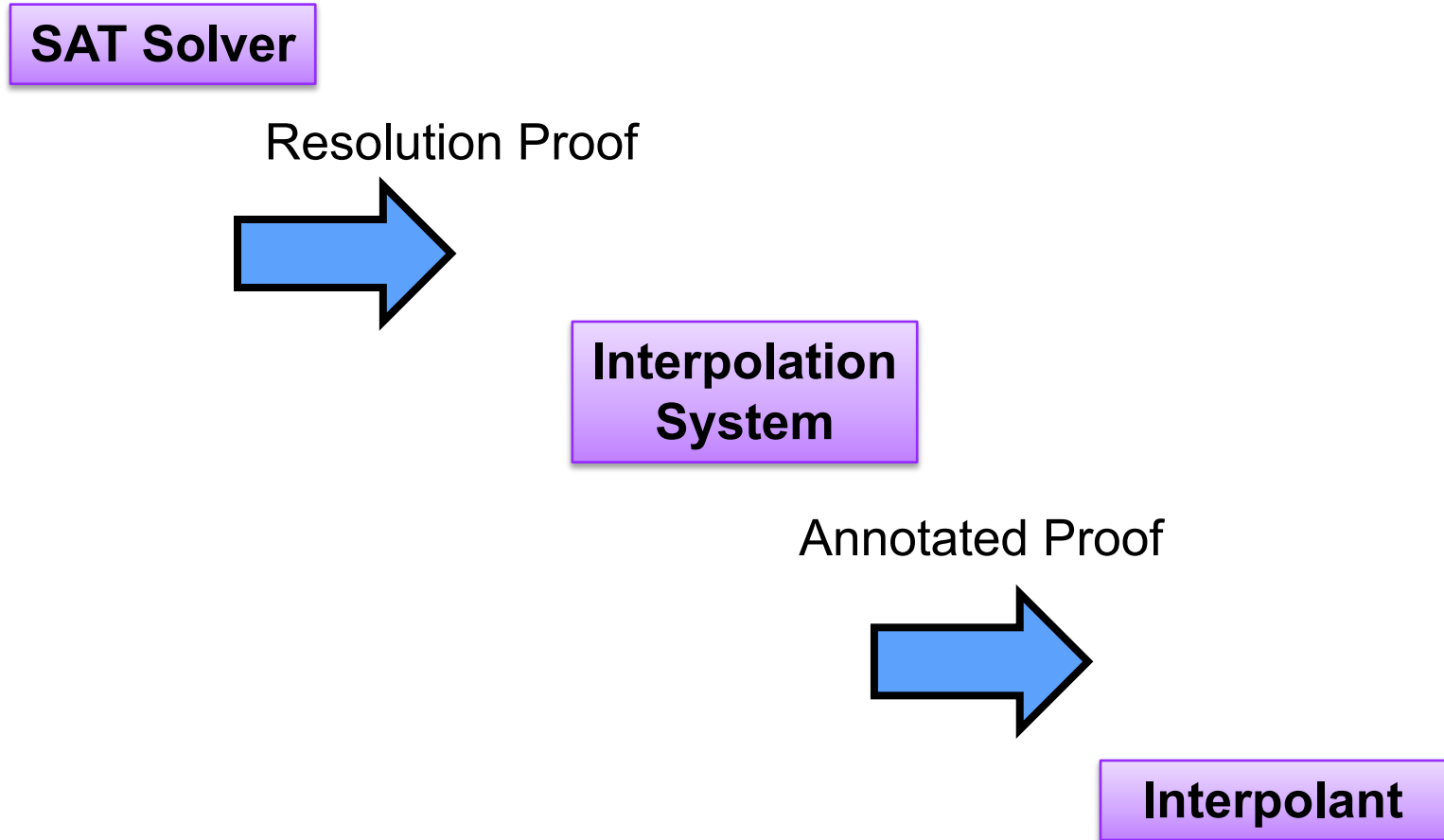
An interpolant is a circuit $I(z)$ such that for every assignment v to z

- $I(v) = A$ 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 $A \wedge B \rightarrow \text{false}$

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

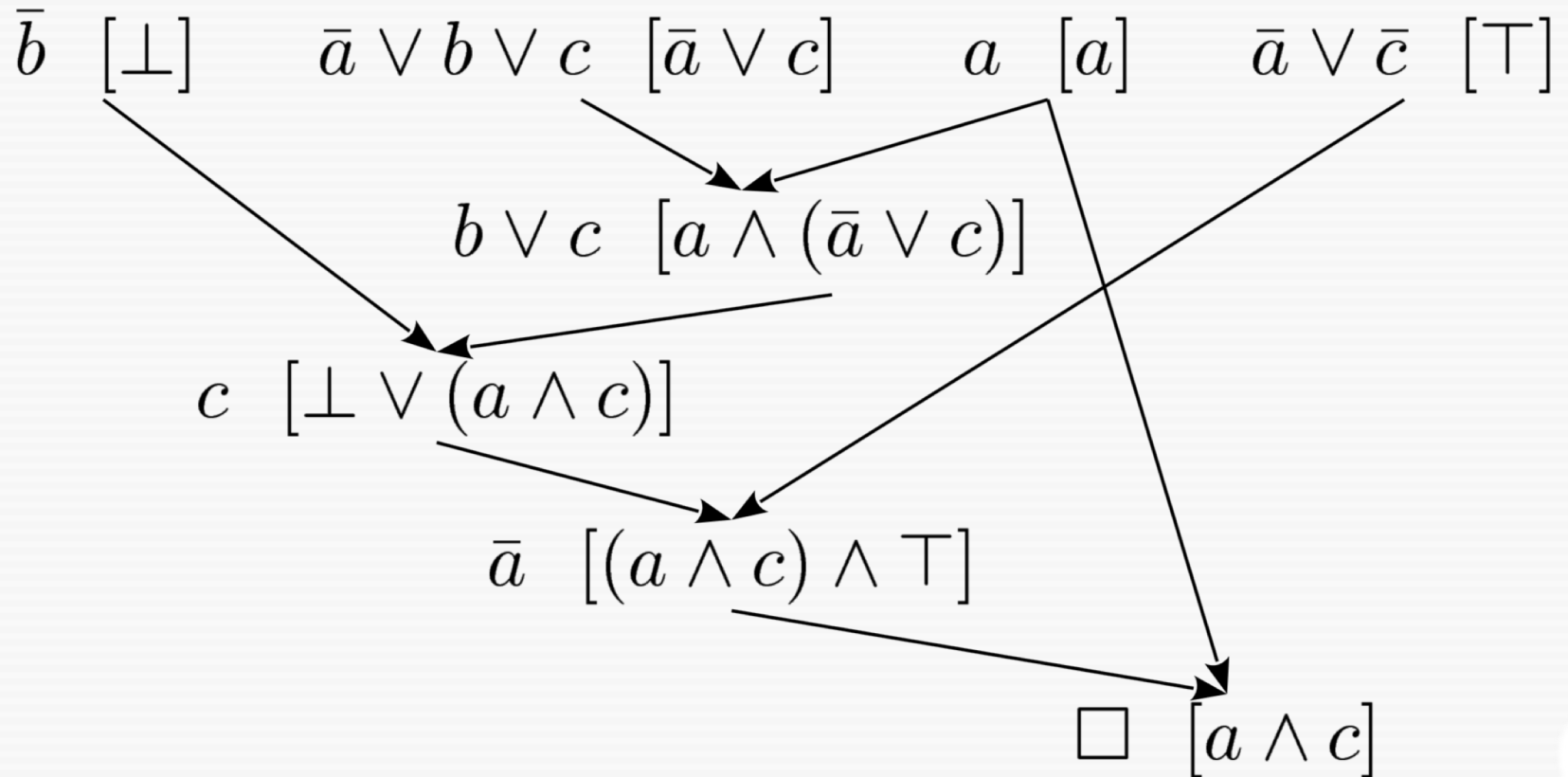
- Notation: $c [p]$ mean formula p is a partial interpolant of clause c

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

$$\begin{array}{c}
 \frac{}{c \quad [g(c)]} \quad c \in A \qquad \qquad \frac{}{c \quad [\top]} \quad c \in B \\
 \\
 \frac{v \vee c \quad [I_1] \qquad \neg v \vee d \quad [I_2]}{c \vee d \quad [I_1 \vee I_2]} \quad v \text{ local to } A \\
 \\
 \frac{v \vee c \quad [I_1] \qquad \neg v \vee d \quad [I_2]}{c \vee d \quad [I_1 \wedge I_2]} \quad v \text{ not local to } A
 \end{array}$$

Interpolation Example

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



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 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: $ITP(A,B) \models \neg ITP(B, A)$
- 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 $S_0, S_1, S_2, \dots, S_n$ be n formulas whose conjunction is UNSAT

Let $ITP(A,B)$ be any interpolation algorithm

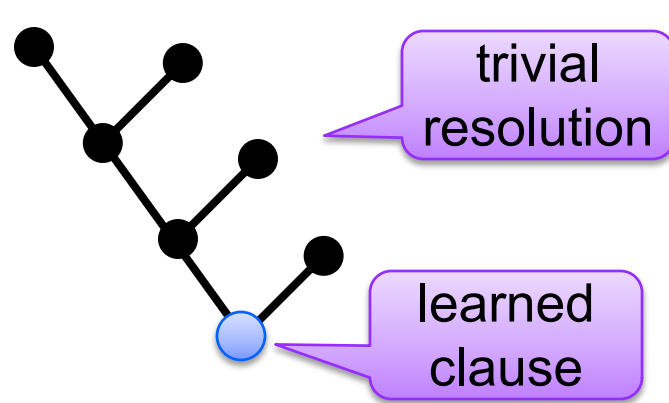
Then, the sequence $I_0, I_1, \dots$ is a sequence interpolant

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

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

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

DRUPing for Interpolants



A CDCL proof is build out of trivial resolutions

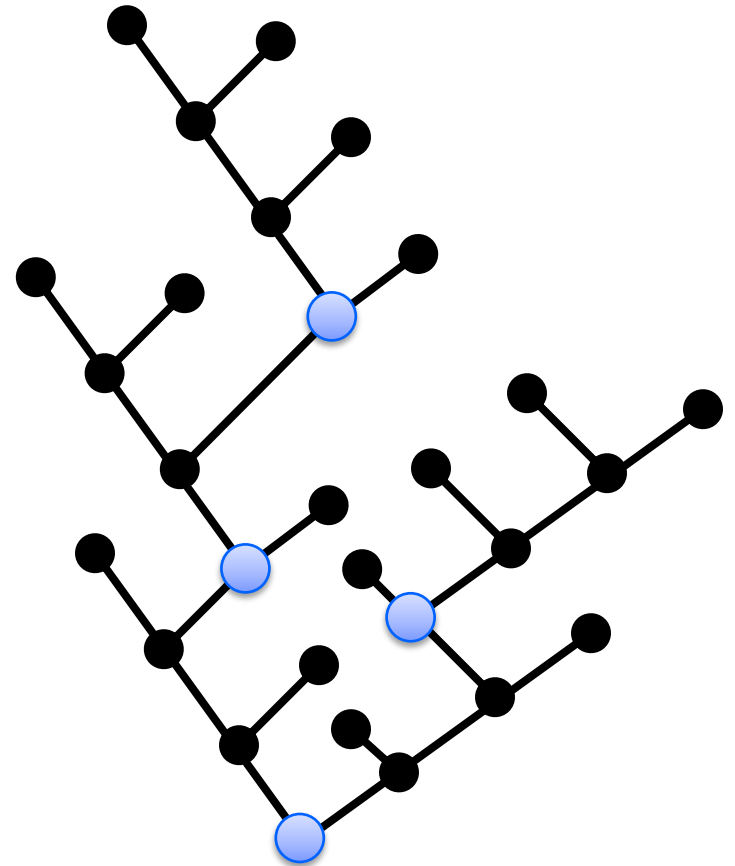
- terminated by a learned clause

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

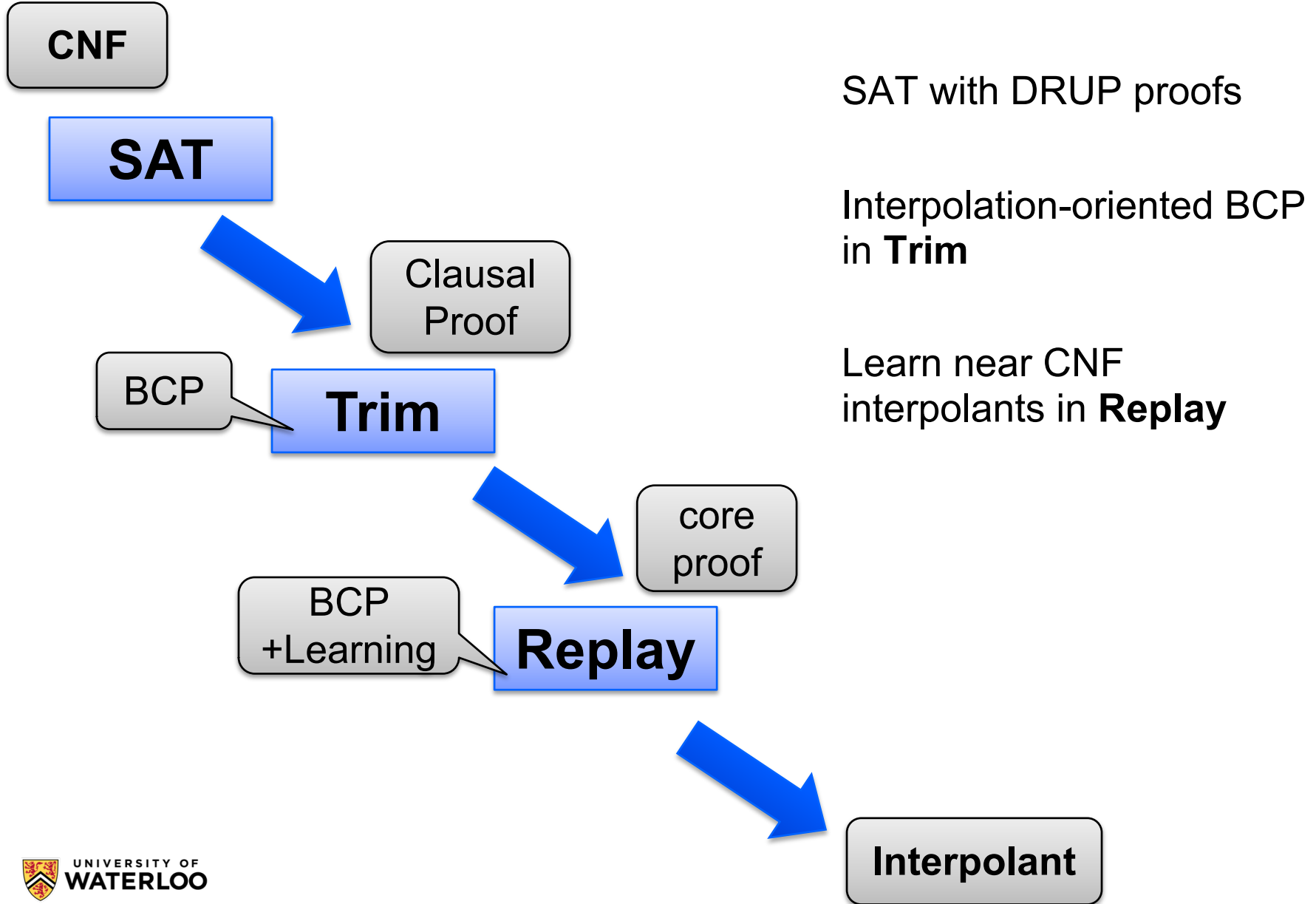
- negation of clause + BCP leads to a conflict

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

Interpolate while replaying the proof



MiniDRUP



DAG Interpolants [TACAS'12]

Given a DAG $G = (V, E)$ and a labeling of edges $\pi: E \rightarrow \text{Expr}$. A **DAG Interpolant** (if it exists) is a labeling $I: V \rightarrow \text{Expr}$ such that

- for any path $v_0, \dots, v_n$, and $0 < k < n$,
 $I(v_k) = \text{ITP}(\pi(v_0) \wedge \dots \wedge \pi(v_{k-1}), \pi(v_k) \wedge \dots \wedge \pi(v_n))$
- $\forall (u, v) \in E. (I(u) \wedge \pi(u, v)) \Rightarrow I(v)$

$$I_2 = \text{ITP}(\pi_1, \pi_8)$$

$$I_2 = \text{ITP}(\pi_1, \pi_2 \wedge \pi_3 \wedge \pi_6 \wedge \pi_7)$$

...

$$(I_1 \wedge \pi_1) \Rightarrow I_2$$

$$(I_2 \wedge \pi_8) \Rightarrow I_7$$

$$(I_2 \wedge \pi_2) \Rightarrow I_3$$

...

