Building Program Verifiers from Compilers and Theorem Provers

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Arie Gurfinkel

based on joint work with Teme Kahsai, Jorge A. Navas, Anvesh Komuravelli, and Nikolaj Bjorner

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM-0002433

The Lab

Download SeaHorn v.0.1.0-rc2

- http://github.com/seahorn/seahorn/releases

(Optionally) If you need a virtual machine, see instructions at:

- http://arieg.bitbucket.org/ssft15.html

Clone http://github.com/seahorn/seahorn-tutoral for examples

PLEASE DO THIS BEFORE THE LAB ☺

For THIS lecture, additional material at

- http://arieg.bitbucket.org/pdf/gurfinkel_ssft15.pdf

Constrained Horn Clauses (CHC)

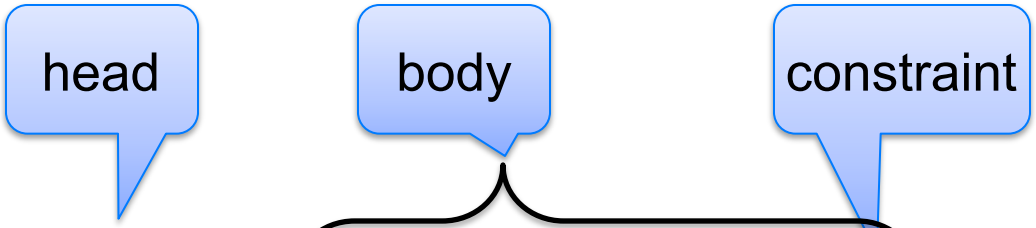
A Constrained Horn Clause (CHC) is a FOL formula of the form

$$\forall V . (\phi \wedge p_1[X_1] \wedge \ldots \wedge p_n[X_n] \rightarrow h[X]),$$

where

- A is a background theory (e.g., Linear Arithmetic, Arrays, Bit-Vectors, or combinations of the above)
- ϕ is a constrained in the background theory A
- $p_1, \ldots, p_n$, h are n-ary predicates
- $p_i[X]$ is an application of a predicate to first-order terms

CHC Terminology



Rule $h[X] \leftarrow p_1[X_1], \ldots, p_n[X_n], \phi.$

Query $\text{false} \leftarrow p_1[X_1], \ldots, p_n[X_n], \phi.$

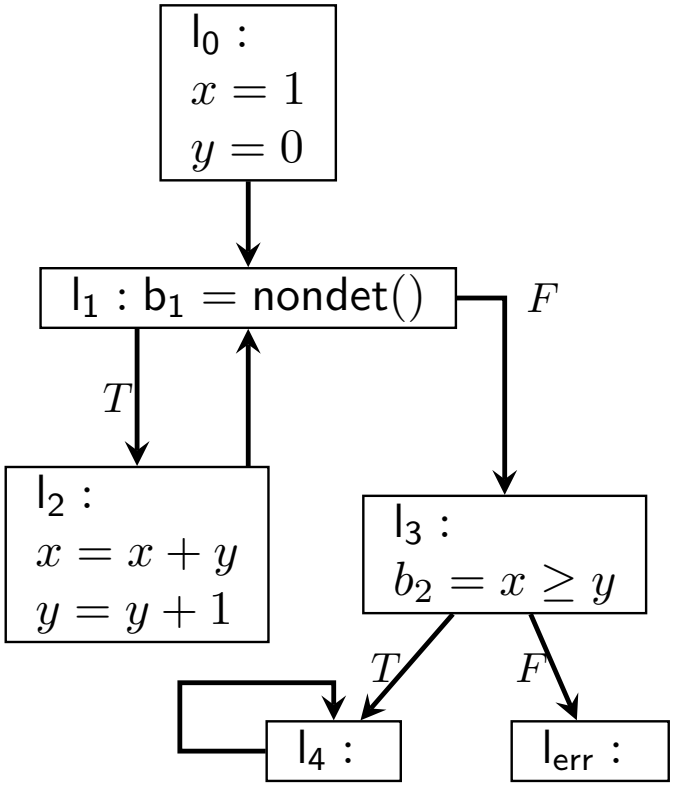
Fact $h[X] \leftarrow \phi.$

Linear CHC $h[X] \leftarrow p[X_1], \phi.$

Non-Linear CHC $h[X] \leftarrow p_1[X_1], \ldots, p_n[X_n], \phi.$ for $n > 1$

Example Horn Encoding

```
int x = 1;
int y = 0;
while (*) {
    x = x + y;
    y = y + 1;
}
assert(x ≥ y);
```



$\langle 1 \rangle$ p_0.

$\langle 2 \rangle$ $\mathsf{p}_1(x, y) \leftarrow$
$\mathsf{p}_0, x = 1, y = 0$.

$\langle 3 \rangle$ $\mathsf{p}_2(x, y) \leftarrow \mathsf{p}_1(x, y)$.

$\langle 4 \rangle$ $\mathsf{p}_3(x, y) \leftarrow \mathsf{p}_1(x, y)$.

$\langle 5 \rangle$ $\mathsf{p}_1(x', y') \leftarrow$
$\mathsf{p}_2(x, y)$,
$x' = x + y$,
$y' = y + 1$.

$\langle 6 \rangle$ $\mathsf{p}_4 \leftarrow (x \geq y), \mathsf{p}_3(x, y)$.

$\langle 7 \rangle$ $\mathsf{p}_{\mathsf{err}} \leftarrow (x < y), \mathsf{p}_3(x, y)$.

$\langle 8 \rangle$ $\mathsf{p}_4 \leftarrow \mathsf{p}_4$.

$\langle 9 \rangle$ $\bot \leftarrow \mathsf{p}_{\mathsf{err}}$.

CHC Satisfiability

A **model** of a set of clauses Π is an interpretation of each predicate p_i that makes all clauses in Π valid

A set of clauses is **satisfiable** if it has a model, and is unsatisfiable otherwise

A model is **A-definable**, it each p_i is definable by a formula ψ_i in A

In the context of program verification

- a program satisfies a property iff corresponding CHCs are satisfiable
- verification certificates correspond to models
- counterexamples correspond to derivations of false

SOLVING CHC WITH SMT

IC3, PDR, and Friends (1)

IC3: A SAT-based Hardware Model Checker

- Incremental Construction of Inductive Clauses for Indubitable Correctness
- A. Bradley: SAT-Based Model Checking without Unrolling. VMCAI 2011

PDR: Explained and extended the implementation

- Property Directed Reachability
- N. Eén, A. Mishchenko, R. K. Brayton: Efficient implementation of property directed reachability. FMCAD 2011

PDR with Predicate Abstraction (easy extension of IC3/PDR to SMT)

- A. Cimatti, A. Griggio, S. Mover, St. Tonetta: IC3 Modulo Theories via Implicit Predicate Abstraction. TACAS 2014
- J. Birgmeier, A. Bradley, G. Weissenbacher: Counterexample to Induction-Guided Abstraction-Refinement (CTIGAR). CAV 2014

IC3, PDR, and Friends (2)

GPDR: Non-Linear CHC with Arithmetic constraints

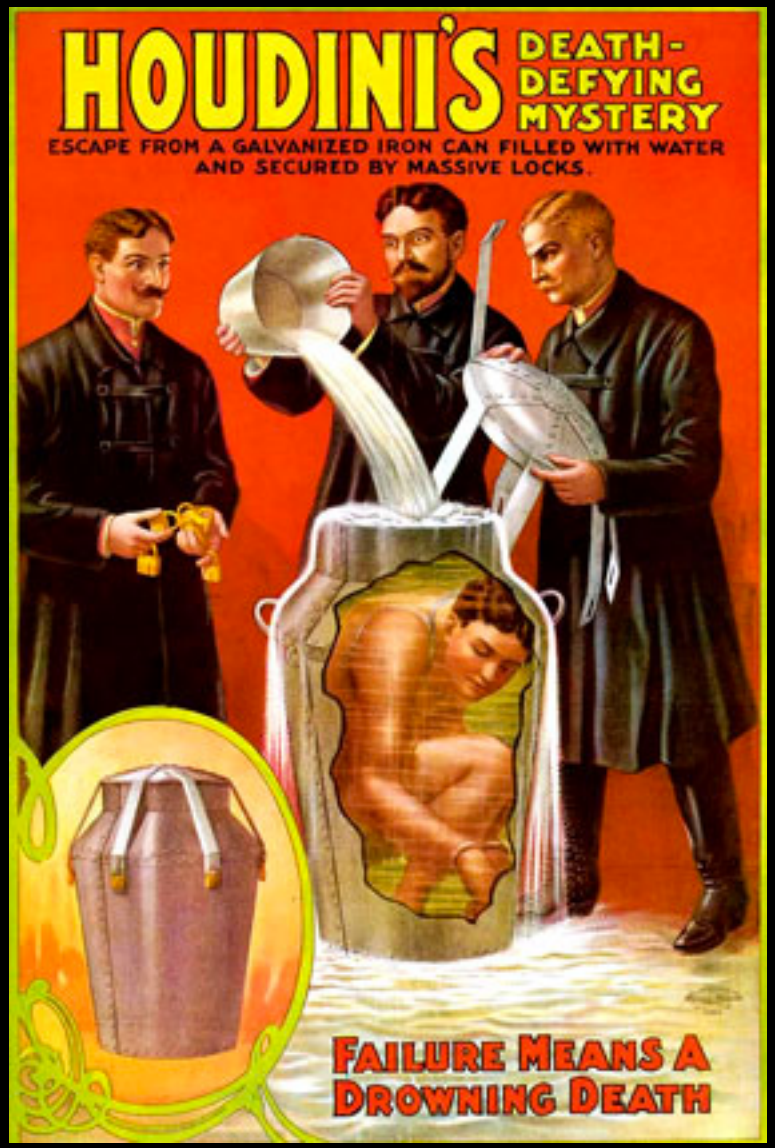
- Generalized Property Directed Reachability
- K. Hoder and N. Bjørner: Generalized Property Directed Reachability. SAT 2012

SPACER: Non-Linear CHC with Arithmetic

- fixes an incompleteness issue in GPDR and extends it with under-approximate summaries
- A. Komuravelli, A. Gurfinkel, S. Chaki: SMT-Based Model Checking for Recursive Programs. CAV 2014

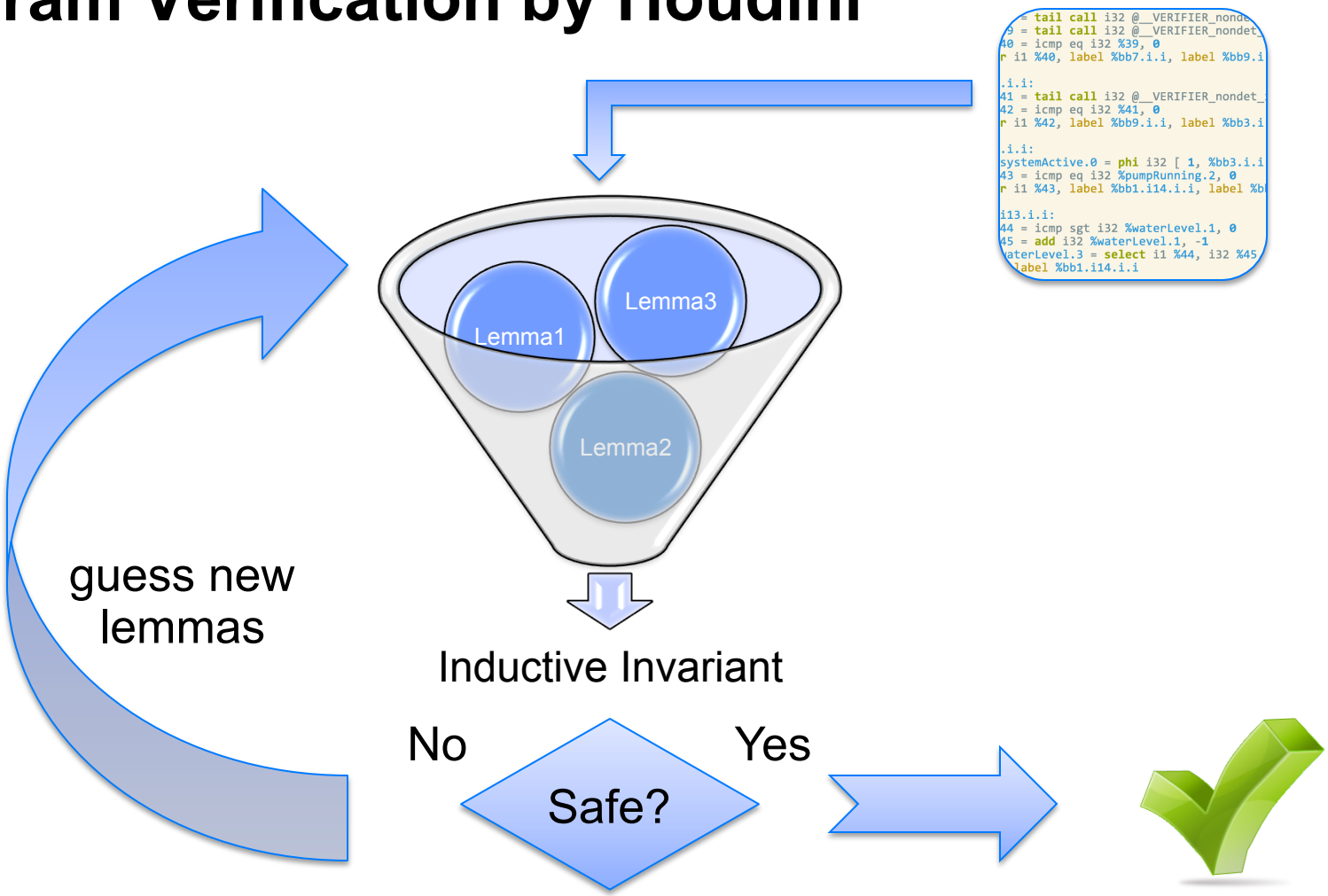
PolyPDR: Convex models for Linear CHC

- simulating Numeric Abstract Interpretation with PDR
- N. Bjørner and A. Gurfinkel: Property Directed Polyhedral Abstraction. VMCAI 2015

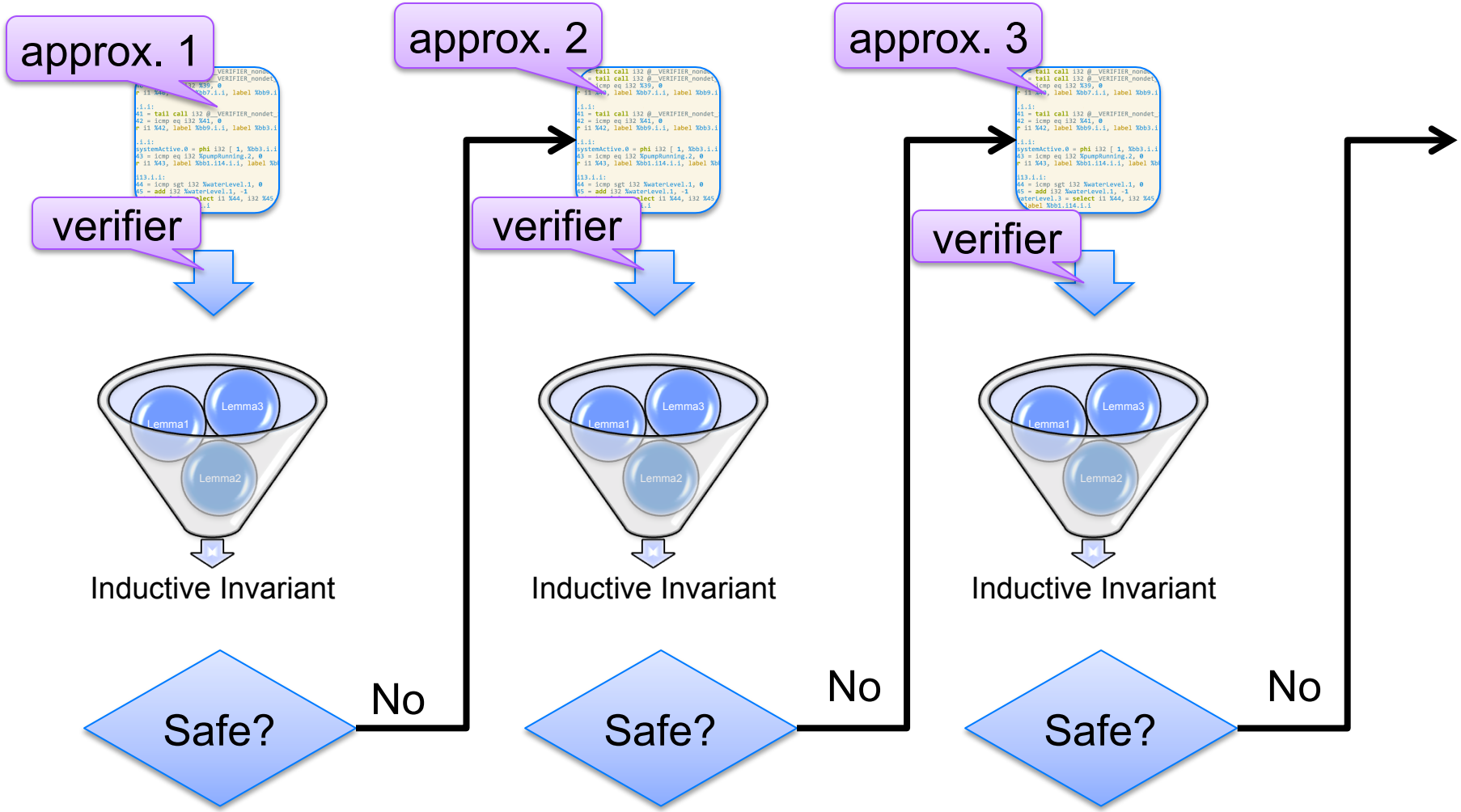


Cormac Flanagan, K. Rustan M. Leino: Houdini, an Annotation Assistant for ESC/Java. FME 2001: 500-517

Program Verification by Houdini



Guessing Lemmas by Evolving Approximations



Linear CHC Satisfiability

Satisfiability of a set of linear CHCs is reducible to satisfiability of THREE clauses of the form

$$Init(X) \rightarrow P(X)$$
$$P(X) \rightarrow Bad(X)$$
$$P(X) \wedge Tr(X, X') \rightarrow P(X')$$

where, X' = {x' | x $\in$ X}, P a fresh predicate, and *Init*, *Bad*, and *Tr* are constraints

Proof:

add extra arguments to distinguish between predicates

$$\frac{Q(y) \wedge \phi \rightarrow W(y, z)}{P(id='Q', y) \wedge \phi \rightarrow P(id='W', y, z)}$$

Programs, Cexs, Invariants

A program *P = (V, Init, $\mathcal{T}r$, Bad)*

- Notation: $\mathcal{F}(X) = \exists\ \boldsymbol{u}\ .\ (X \wedge \mathcal{T}r) \vee \text{Init}$

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.*

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

$$Inv \Rightarrow \neg Bad \qquad \text{Safe}$$

IC3/PDR Algorithm Overview

Input: Safety problem $\langle Init(X), Tr(X, X'), Bad(X) \rangle$

$F_0 \leftarrow Init$; $N \leftarrow 0$ **repeat**

  $\mathbf{G} \leftarrow \textsc{PdrMkSafe}([F_0, \ldots, F_N], Bad)$

  if $\mathbf{G} = [\,]$ **then return** *Reachable*;

  $\forall 0 \leq i \leq N \cdot F_i \leftarrow \mathbf{G}[i]$

  $F_0, \ldots, F_N \leftarrow \textsc{PdrPush}([F_0, \ldots, F_N])$

  if $\exists 0 \leq i < N \cdot F_i = F_{i+1}$ **then return** *Unreachable*;

  $N \leftarrow N + 1$; $F_N \leftarrow \emptyset$

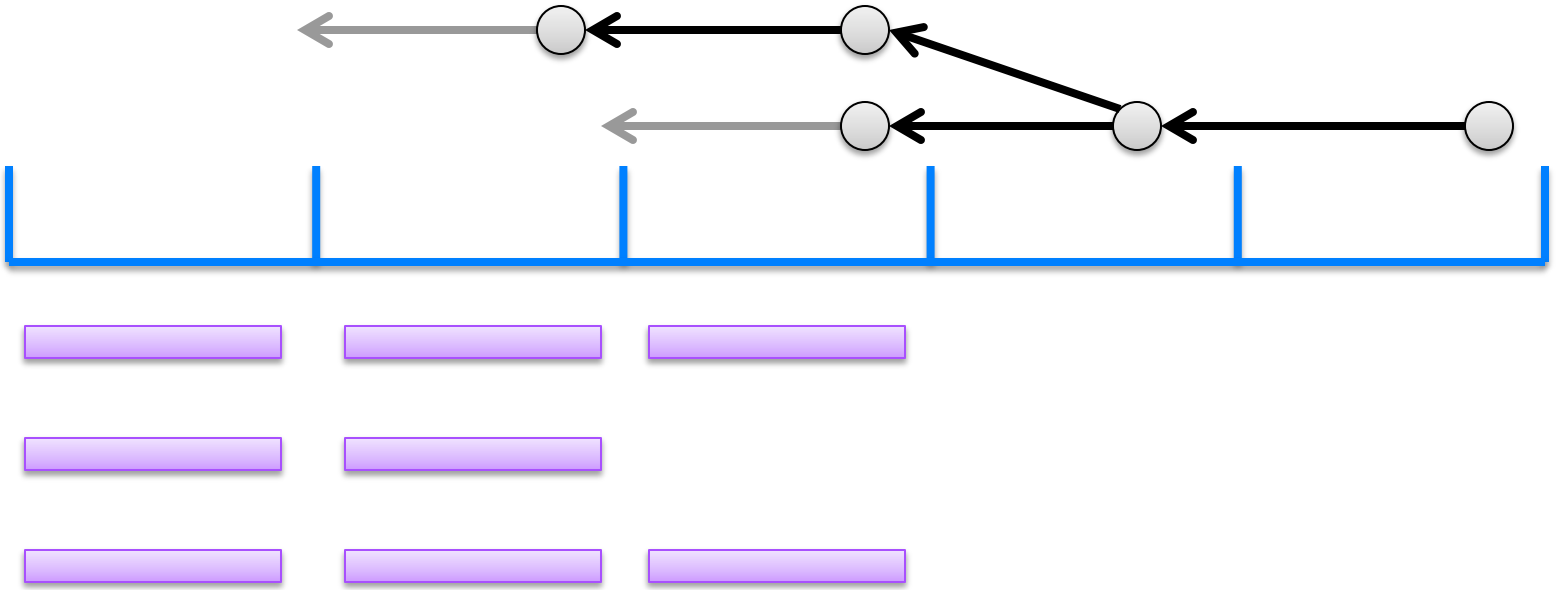
until ∞;

bounded safety

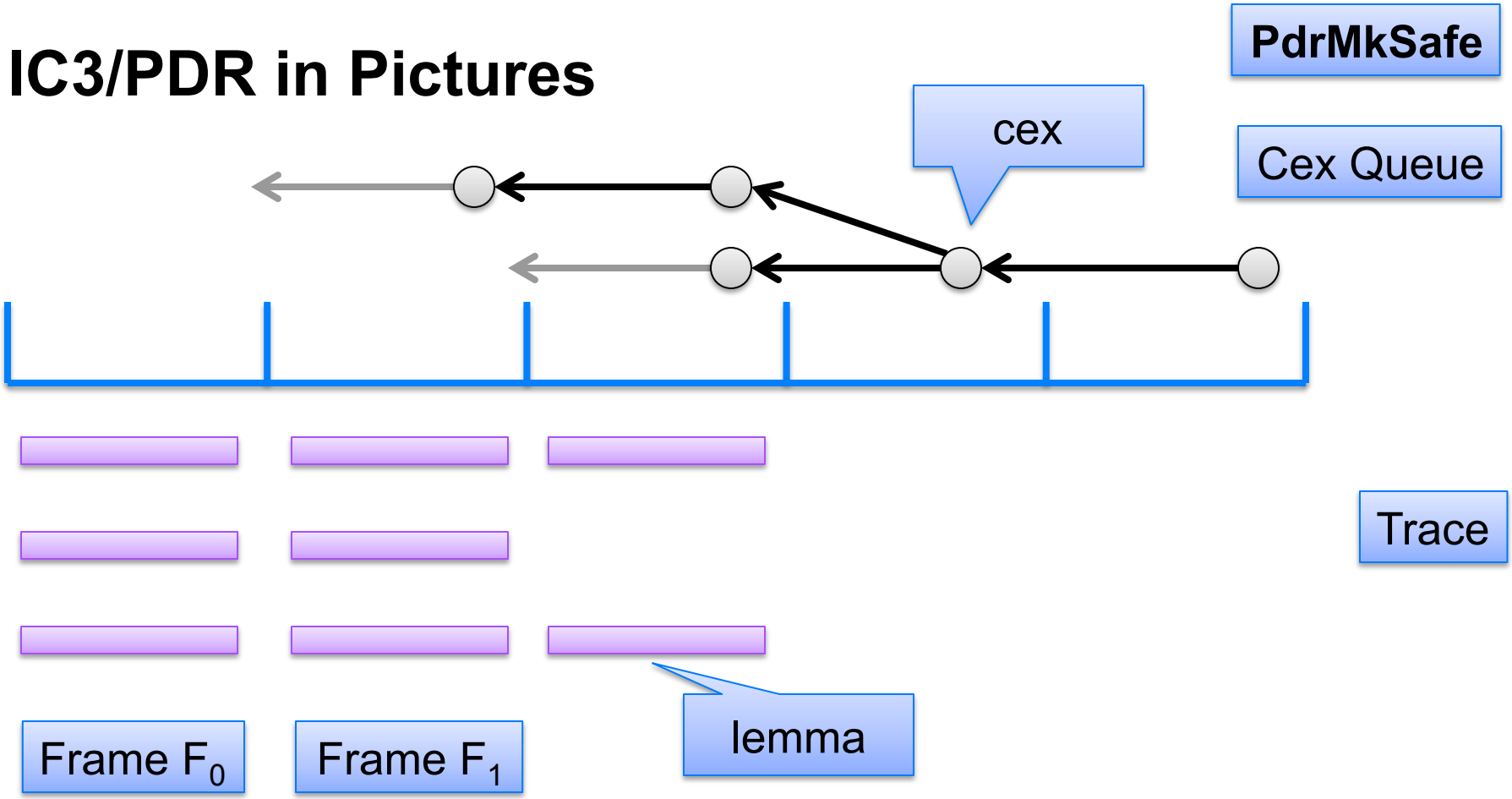
strengthen result

IC3/PDR in Pictures

PdrMkSafe

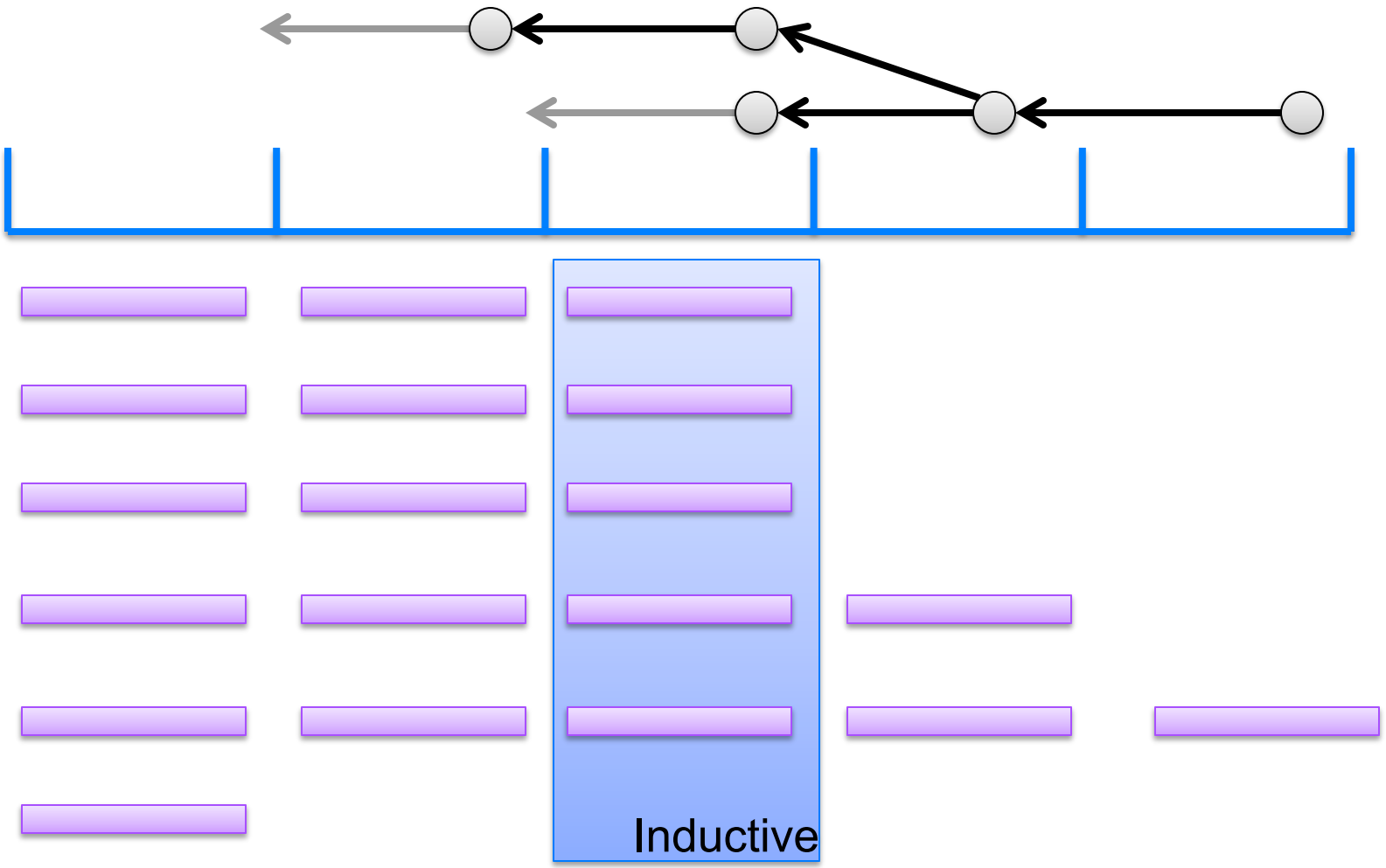


IC3/PDR in Pictures



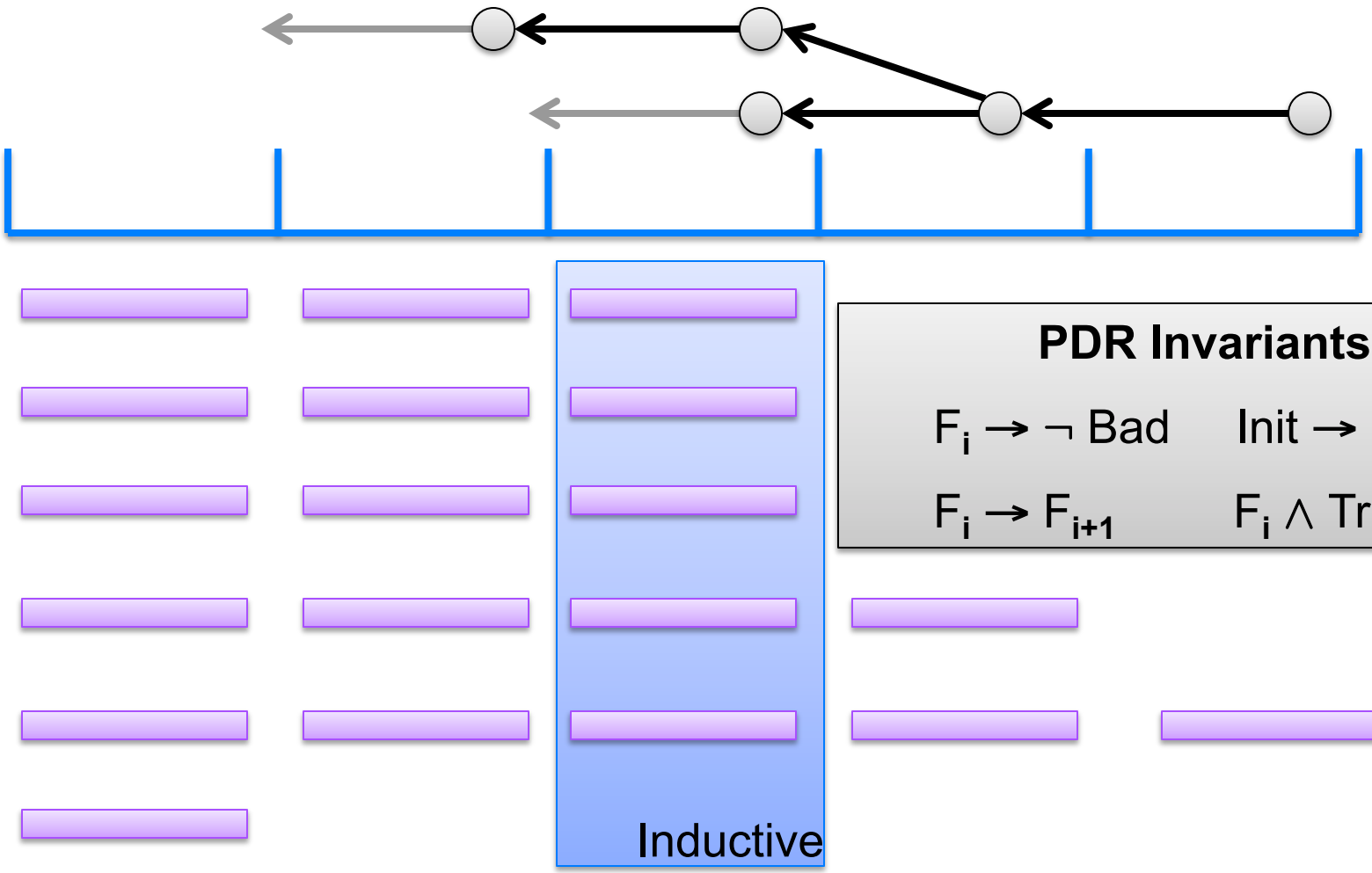
IC3/PDR in Pictures

PdrPush



IC3/PDR in Pictures

PdrPush



PDR Invariants

$F_i \rightarrow \neg$ Bad Init $\rightarrow F_i$

$F_i \rightarrow F_{i+1}$ $F_i \wedge Tr \rightarrow F_{i+1}$

IC3/PDR

Input: A safety problem $\langle Init(X), Tr(X, X'), Bad(X)\rangle$.
Output: *Unreachable* or *Reachable*
Data: A cex queue Q, where $c \in Q$ is a pair $\langle m, i\rangle$, m is a cube over state variables, and $i \in \mathbb{N}$. A level N. A trace $F_0, F_1, \ldots$
Initially: $Q = \emptyset$, $N = 0$, $F_0 = Init$, $\forall i > 0 \cdot F_i = \emptyset$.
repeat

Unreachable If there is an $i < N$ s.t. $F_i \subseteq F_{i+1}$ **return** *Unreachable*.

Reachable If there is an m s.t. $\langle m, 0\rangle \in Q$ **return** *Reachable*.

Unfold If $F_N \rightarrow \neg Bad$, then set $N \leftarrow N + 1$.

Candidate If for some m, $m \rightarrow F_N \wedge Bad$, then add $\langle m, N\rangle$ to Q.

Decide If $\langle m, i + 1\rangle \in Q$ and there are m_0 and m_1 s.t. $m_1 \rightarrow m$, $m_0 \wedge m_1'$ is satisfiable, and $m_0 \wedge m_1' \rightarrow F_i \wedge Tr \wedge m'$, then add $\langle m_0, i\rangle$ to Q.

Conflict For $0 \leq i < N$: given a candidate model $\langle m, i + 1\rangle \in Q$ and clause φ, such that $\varphi \rightarrow \neg m$, if $Init \rightarrow \varphi$, and $\varphi \wedge F_i \wedge Tr \rightarrow \varphi'$, then add φ to F_j, for $j \leq i + 1$.

Leaf If $\langle m, i\rangle \in Q$, $0 < i < N$ and $F_{i-1} \wedge Tr \wedge m'$ is unsatisfiable, then add $\langle m, i + 1\rangle$ to Q.

Induction For $0 \leq i < N$ and a clause $(\varphi \vee \psi) \in F_i$, if $\varphi \notin F_{i+1}$, $Init \rightarrow \varphi$ and $\varphi \wedge F_i \wedge Tr \rightarrow \varphi'$, then add φ to F_j, for each $j \leq i + 1$.

until ∞;

IC3 Data-Structures

A trace $F = F_0, \ldots, F_N$ is a sequence of frames.

- A frame F_i is a set of clauses. Elements of F_i are called lemmas.
- Invariants:
  - Bounded Safety: $\forall i < N . F_i \rightarrow \neg Bad$
  - Monotonicity: $\forall i < N . F_{i+1} \subseteq F_i$
  - Inductiveness: $\forall i < N . F_i \wedge Tr \rightarrow F'_{i+1}$

A priority queue Q of counterexamples to induction (CTI)

- $(m, i) \in Q$ is a pair, where m is a cube and i a level
- if $(m, i) \in Q$ then there exists a path of length (N-i) from a state in m to a state in Bad
- Q is ordered by level
  - $(m, i) < (k, j)$ iff $i < j$

Termination and Progress

Unreachable If there is an $i < N$ s.t. $F_i \subseteq F_{i+1}$
return *Unreachable*.

Reachable If there is an m s.t. $\langle m, 0 \rangle \in Q$
return *Reachable*.

Unfold If $F_N \rightarrow \neg Bad$, then set $N \leftarrow N + 1$.

Candidate If for some m, $m \rightarrow F_N \wedge Bad$,
then add $\langle m, N \rangle$ to Q.

Inductive Generalization

Conflict For $0 \leq i < N$: given a candidate model $\langle m, i+1 \rangle \in Q$ and clause φ, such that $\varphi \rightarrow \neg m$, if $Init \rightarrow \varphi$, and $\varphi \wedge F_i \wedge Tr \rightarrow \varphi'$, then add φ to F_j, for $j \leq i+1$.

A clause φ is inductive relative to F iff

- Init → φ (Initialization) and φ ∧ F ∧ Tr → φ' (Inductiveness)

Implemented by first letting φ = ¬m and generalizing φ by iteratively dropping literals while checking the inductiveness condition

Theorem: Let $F_0, F_1, \ldots, F_N$ be a valid IC3 trace. If φ is inductive relative to F_i, $0 \leq i < N$, then, for all $j \leq i$, φ is inductive relative to F_j.

- Follows from the monotonicity of the trace
  - if $j < i$ then $F_j \rightarrow F_i$
  - if $F_j \rightarrow F_i$ then $(\varphi \wedge F_i \wedge Tr \rightarrow \varphi') \rightarrow (\varphi \wedge F_j \wedge Tr \rightarrow \varphi')$

Prime Implicants

A formula φ is an *implicant* of a formula psi iff $\varphi \Rightarrow \psi$

A *propositional implicant* of ψ is a conjunction of literals φ such that φ is an implicant of ψ

- φ is a conjunction of literals
- $\varphi \Rightarrow \psi$
- φ is a partial assignment that makes ψ true

A propositonal implicant φ of ψ is called *prime* if no subset of φ is an implicant of ψ

- φ is a conjunction of literals
- $\varphi \Rightarrow \psi$
- $\forall\, p\, .\, (p \neq \varphi \wedge \varphi \Rightarrow p) \Rightarrow (p \not\Rightarrow \psi)$

Generalizing Predecessors

Decide If $\langle m, i+1 \rangle \in Q$ and there are m_0 and m_1 s.t. $m_1 \rightarrow m$, $m_0 \wedge m_1'$ is satisfiable, and $m_0 \wedge m_1' \rightarrow F_i \wedge Tr \wedge m'$, then add $\langle m_0, i \rangle$ to Q.

Decide rule chooses a (generalized) predecessor m_0 of m that is consistent with the current frame

Simplest implementation is to extract a predecessor m_o from a satisfying assignment of $M \models Fi \wedge Tr \wedge m'$

- m_0 cab be further generalized using ternary simulation by dropping literals and checking that m' remains forced

An alternative is to let m_0 be an implicant (not necessarily prime) of $F_i \wedge \exists X'.(Tr \wedge m')$

- finding a prime implicant is difficult because of the existential quantification
- we settle for an arbitrary implicant. The side conditions ensure it is not trivial

Strengthening a trace

Induction For $0 \leq i < N$ and a clause $(\varphi \vee \psi) \in F_i$, if $\varphi \notin F_{i+1}$, $Init \rightarrow \varphi$ and $\varphi \wedge F_i \wedge Tr \rightarrow \varphi'$, then add φ to F_j, for each $j \leq i + 1$.

Also known as **Push** or **Propagate**

Bounded safety proofs are usually very weak towards the end

- not much is needed to show that error will not happen in one or two steps

This tends to make them non-inductive

- a weakness of interpolation-based model checking, like IMPACT
- in IMPACT, this is addressed by forced covering heuristic

Induction "applies" forced cover one lemma at a time

- whenever all lemmas are pushed F_{i+1} is inductive (and safe)
- (optionally) combine strengthening with generalization

Implementation

- Apply Induction from 0 to N whenever **Conflict** and **Decide** are not applicable

Long Counterexamples

Leaf If $\langle m, i \rangle \in Q$, $0 < i < N$ and $F_{i-1} \wedge Tr \wedge m'$ is unsatisfiable, then add $\langle m, i+1 \rangle$ to Q.

Whenever a counterexample *m* is blocked at level *i*, it is known that

- there is no path of length *i* from *Init* to *m* (because got blocked)
- there is a path of length *(N-i)* from *m* to *Bad*

Can check whether there exists a path of length *(i+1)* from *Init* to *m*

- (**Leaf**) check eagerly by placing the CTI back into the queue at a higher level
- **(No Leaf)** check lazily by waiting until the same (or similar) CTI is discovered after N is increased by **Unfold**

Leaf allows IC3 to discover counterexamples much longer than the current unfolding depth N

- each CTI re-enqueued by **Leaf** adds one to the depth of the longest possible counterexample found
- a real counterexample might chain through multiple such CTI's

Queue Management for Long Counterexamples

A queue element is a triple *(m, i, d)*

- *m* is a CTI, *i* a level, *d* a depth

Decide sets *m* and *i* as before, and sets *d* to 0

Leaf increases *i* and *d* by one

- *i* determines how far the CTI can be pushed back
- *d* counts number of times the CTI was pushed forward

Queue is ordered first by level, then by depth

- $(m, i, d) < (k, j, e) \Leftrightarrow i < j \vee (i=j \wedge d < e)$

Overall exploration mimics iterative deepening with non-uniform exploration depth

- go deeper each time before backtracking

PDR FOR ARITHMETIC

Arithmetic PDR

Input: A safety problem $\langle Init(X), Tr(X, X'), Bad(X)\rangle$.
Output: *Unreachable* or *Reachable*
Data: A cex queue Q, where $c \in Q$ is a pair $\langle m, i\rangle$, m is a cube over state variables, and $i \in \mathbb{N}$. A level N. A trace $F_0, F_1, \ldots$
Initially: $Q = \emptyset$, $N = 0$, $F_0 = Init$, $\forall i > 0 \cdot F_i = \emptyset$.
repeat
  Unreachable If there is an $i < N$ s.t. $F_i \subseteq F_{i+1}$ **return** *Unreachable*.

Notation: $\mathcal{F}(A) = A(X) \wedge Tr(X, X') \vee Init(X')$.

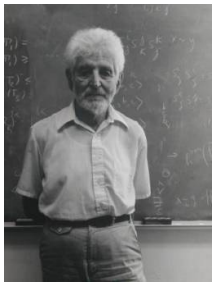
Decide If $\langle P, i+1\rangle \in Q$ and there is a model $m(X, X')$ s.t. $m \models \mathcal{F}(F_i) \wedge P'$, add $\langle P_\downarrow, i\rangle$ to Q, where $P_\downarrow = \mathrm{MBP}(X', m, \mathcal{F}(F_i) \wedge P')$.

Conflict For $0 \leq i < N$, given a counterexample $\langle P, i+1\rangle \in Q$ s.t. $\mathcal{F}(F_i) \wedge P'$ is unsatisfiable, add $P^\uparrow = \textsc{Itp}(\mathcal{F}(F_i)(X^o, X), P)$ to F_j for $j \leq i+1$.

  Induction For $0 \leq i < N$ and a clause $(\varphi \vee \psi) \in F_i$, if $\varphi \notin F_{i+1}$, $Init \rightarrow \varphi$ and $\varphi \wedge F_i \wedge Tr \rightarrow \varphi'$, then add φ to F_j, for each $j \leq i+1$.

until ∞;

Craig Interpolation Theorem



Theorem (Craig 1957)

Let A and B be two First Order (FO) formulae such that $A \Rightarrow \neg B$, then there exists a FO formula I, denoted ITP(A, B), such that

$$A \Rightarrow I \qquad I \Rightarrow \neg B$$

$$atoms(I) \in atoms(A) \cap atoms(B)$$

A Craig interpolant ITP(A, B) can be effectively constructed from a resolution proof of unsatisfiability of $A \wedge B$

In Model Cheching, Craig Interpolation Theorem is used to safely over-approximate the set of (finitely) reachable states

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

Farkas Lemma

Let $M = t_1 \geq b_1 \wedge \ldots \wedge t_n \geq b_n$, where t_i are linear terms and b_i are constants M is *unsatisfiable* iff $0 \geq 1$ is derivable from M by resolution

M is *unsatisfiable* iff $M \vdash 0 \geq 1$

- e.g., $x + y > 10, -x > 5, -y > 3 \vdash (x+y-x-y) > (10 + 5 + 3) \vdash 0 > 18$

M is unsatisfiable iff there exist *Farkas* coefficients $g_1, \ldots, g_n$ such that

- $g_i \geq 0$
- $g_1 \times t_1 + \ldots + g_n \times t_n = 0$
- $g_1 \times b_1 + \ldots + g_n \times b_n \geq 1$

Interpolation for Linear Real Arithmetic

Let $M = A \wedge B$ be UNSAT, where

- $A = t_1 \geq b_1 \wedge \ldots \wedge t_i \geq b_i$, and
- $B = t_{i+1} \geq b_i \wedge \ldots \wedge t_n \geq b_n$

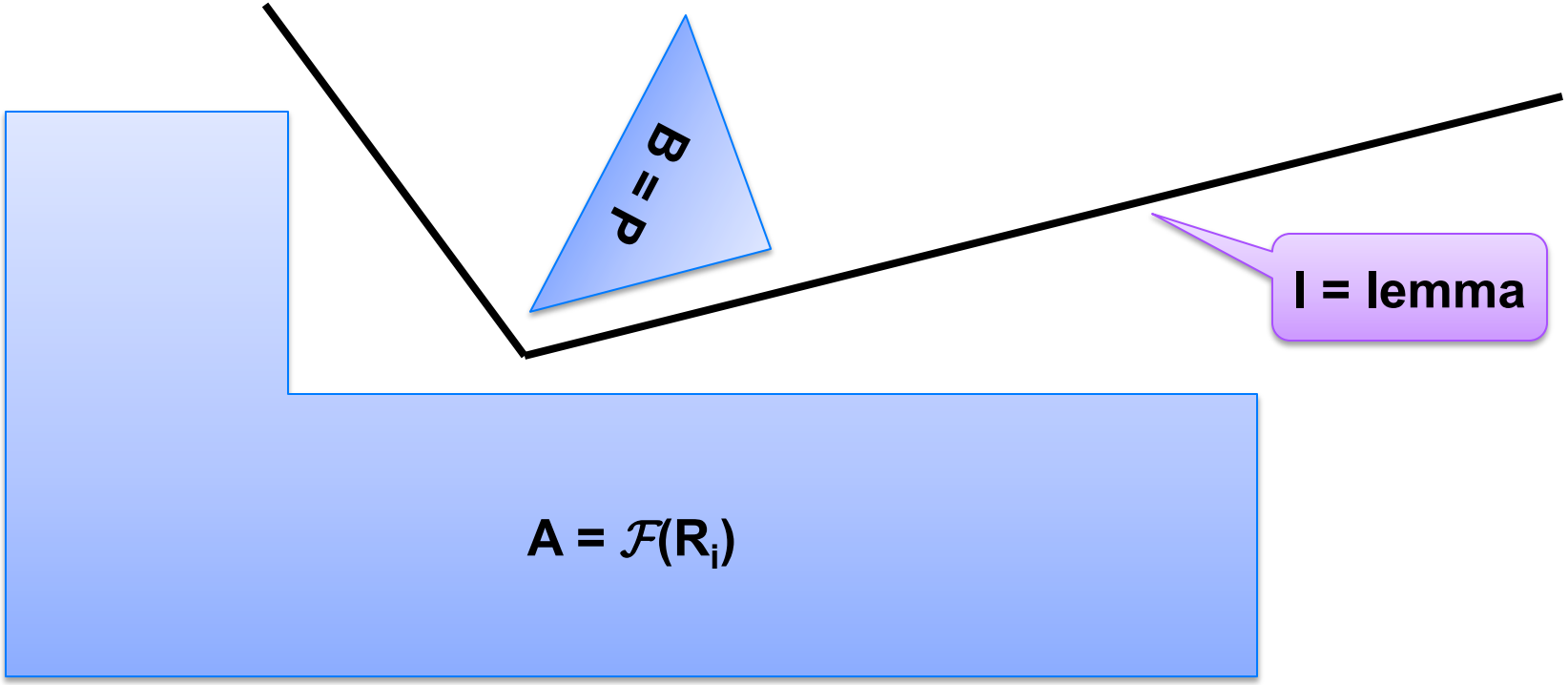
Let $g_1, \ldots, g_n$ be the Farkas coefficients witnessing UNSAT

Then

- $g_1 \times (t_1 \geq b_1) + \ldots + g_i \times (t_i \geq b_i)$ is an interpolant between A and B
- $g_{i+1} \times (t_{i+1} \geq b_i) + \ldots + g_n \times (t_n \geq b_n)$ is an interpolant between B and A

- $g_1 \times t_1 + \ldots + g_i \times t_i = -(g_{i+1} \times t_{i+1} + \ldots + g_n \times t_n)$
- $\neg(g_{i+1} \times (t_{i+1} \geq b_i) + \ldots + g_n \times (t_n \geq b_n))$ is an interpolant between A and B

Craig Interpolation for Linear Arithmetic



Useful properties of existing interpolation algorithms [CGS10] [HB12]

- $I \in ITP(A, B)$ then $\neg I \in ITP(B, A)$
- if A is syntactically convex (a monomial), then I is convex
- if B is syntactically convex, then I is co-convex (a clause)
- if A and B are syntactically convex, then I is a half-space

Arithmetic Conflict

Notation: $\mathcal{F}(A) = (A(X) \wedge Tr) \vee Init(X')$.

Conflict For $0 \leq i < N$, given a counterexample $\langle P, i+1 \rangle \in Q$ s.t. $\mathcal{F}(F_i) \wedge P'$ is unsatisfiable, add $P^{\uparrow} = \textsc{Itp}(\mathcal{F}(F_i), P')$ to F_j for $j \leq i+1$.

Counterexample is blocked using Craig Interpolation

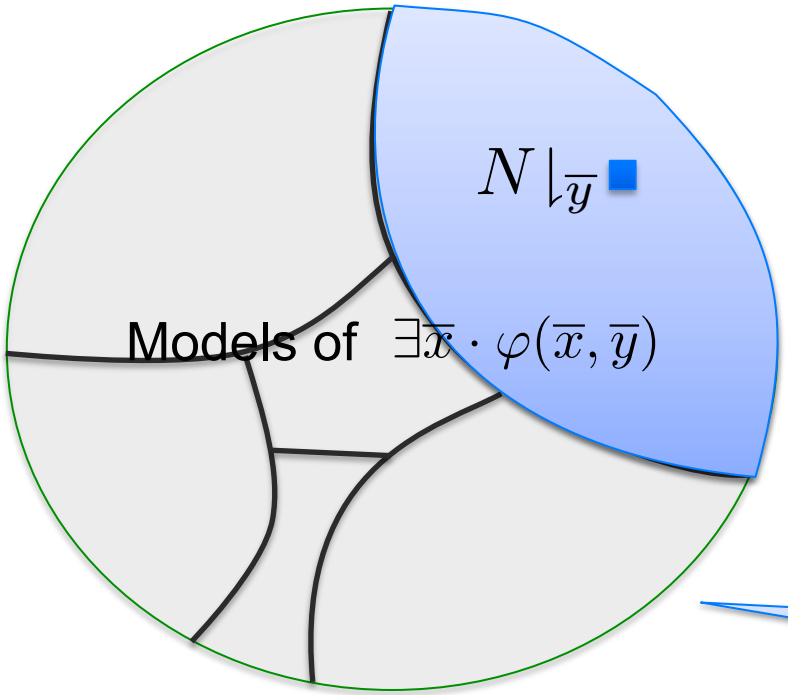
- summarizes the reason why the counterexample cannot be extended

Generalization is not inductive

- weaker than IC3/PDR
- inductive generalization for arithmetic is still an open problem

Model Based Projection

Expensive to find a quantifier-free $\psi(\overline{y}) \equiv \exists \overline{x} \cdot \varphi(\overline{x}, \overline{y})$



1. find $N \models \varphi(\overline{x}, \overline{y})$

(e.g. specific pre-post pair that needs to be generalized)

2. choose disjunct "covering" *N*

using virtual substitution

Lazy Quantifier Elimination!

Model Based Projection

Definition: Let φ be a formula, U a set of variables, and M a model of φ. Then ψ = MBP (U, M, φ) is a Model Based Project of U, M and φ iff

1. ψ is a monomial (optional)
2. Vars(ψ) $\subseteq$ Vars(φ) $\setminus$ U
3. M $\models \psi$
4. $\psi \Rightarrow \exists$ U . φ

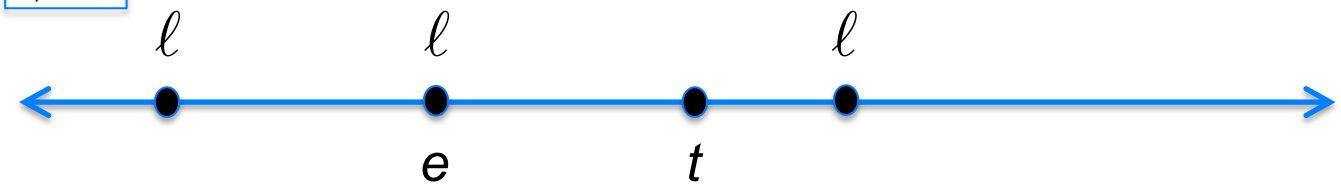
For a fixed set of variables U and a formula φ, MBP is a function from models to formulas

MBP is *finite* if its range (as a function defined above) is finite

MBP for Linear Rational Arithmetic

$$\exists \ell \cdot (\ell = e \wedge \phi_1) \vee (t < \ell \wedge \ell < u) \vee (\ell < u \wedge \phi_2)$$

$$\equiv \quad (\phi_1 \vee (t < e \wedge e < u) \vee (e < u \wedge \phi_2))$$
$$\vee \; (t < u \vee (t < u \wedge \phi_2))$$
$$\vee \; \phi_2$$



pick a disjunct that covers a given model

[1] Loos and Weispfenning, *Applying Linear Quantifier Elimination*, 1993
[2] Tobias Nipkow, *Linear Quantifier Elimination*, 2008
[3] Bjorner, *Linear Quantifier Elimination as an Abstract Decision Procedure*, 2010

Arithmetic Decide

Notation: $\mathcal{F}(A) = (A(X) \wedge Tr(X, X') \vee Init(X')$.

Decide If $\langle P, i+1 \rangle \in Q$ and there is a model $m(X, X')$ s.t. $m \models \mathcal{F}(F_i) \wedge P'$,
add $\langle P_{\downarrow}, i \rangle$ to Q, where $P_{\downarrow} = \mathrm{MBP}(X', m, \mathcal{F}(F_i) \wedge P')$.

Compute a predecessor using an under-approximation of quantifier elimination – called Model Based Projection

To ensure progress, Decide must be finite

- finitely many possible predecessors when all other arguments are fixed

Alternatives

- Completeness can follow from the **Conflict** rule only
  - for Linear Arithmetic this means using Fourier-Motzkin implicants
- Completeness can follow from an interaction of **Decide** and **Conflict**

PDR FOR NON-LINEAR CHC

Non-Linear CHC Satisfiability

Satisfiability of a set of arbitrary (i.e., linear or non-linear) CHCs is reducible to satisfiability of THREE clauses of the form

$$Init(X) \rightarrow P(X)$$

$$P(X) \rightarrow Bad(X)$$

$$P(X) \wedge P(X^o) \wedge Tr(X, X^o, X') \rightarrow P(X')$$

where, X' = {x' | x ∈ X}, X^o = {x^o | x ∈ X}, P a fresh predicate, and Init, Bad, and Tr are constraints

Proof:

- factor rules with more than 2 predicates in the body

  replace $P_1(x) \wedge P_2(y) \wedge P_3(z) \wedge \phi(x,y,z) \rightarrow H(x, y, z)$

  by $P_1(x) \wedge W(y,z) \wedge phi(x,y,z) \rightarrow H(x,y,z)$. $P_2(y) \wedge P_3(z) \rightarrow W(y, z)$.
- add extra arguments to distinguish between predicates

  $P(id='P_2', y) \wedge P(id='P_3', z) \rightarrow P(id='W', y, z)$.

Non-linear CHC by reduction to linear CHC

Can non-linear CHC satisfiability be reduced to (multiple) linear CHC satisfiability problems?

$$Init(X) \rightarrow P(X)$$
$$P(X) \rightarrow Bad(X)$$
$$P(X) \wedge P(X^o) \wedge Tr(X, X^o, X') \rightarrow P(X')$$

Generalized GPDR

Input: A safety problem $\langle Init(X), Tr(X, X^o, X'), Bad(X)\rangle$.
Output: *Unreachable* or *Reachable*
Data: A cex queue Q, where a cex $\langle c_0, \ldots, c_k\rangle \in Q$ is a tuple, each $c_j = \langle m, i\rangle$, m is a cube over state variables, and $i \in \mathbb{N}$. A level N. A trace $F_0, F_1, \ldots$
Notation: $\mathcal{F}(A, B) = Init(X') \vee (A(X) \wedge B(X^o) \wedge Tr)$, and $\mathcal{F}(A) = \mathcal{F}(A, A)$
Initially: $Q = \emptyset$, $N = 0$, $F_0 = Init$, $\forall i > 0 \cdot F_i = \emptyset$
Require: $Init \rightarrow \neg Bad$
repeat

Unreachable If there is an $i < N$ s.t. $F_i \subseteq F_{i+1}$ **return** *Unreachable*.

Reachable if exists $t \in Q$ s.t. for all $\langle c, i\rangle \in t$, $i = 0$, **return** *Reachable*.

Unfold If $F_N \rightarrow \neg Bad$, then set $N \leftarrow N + 1$ and $Q \leftarrow \emptyset$.

Candidate If for some m, $m \rightarrow F_N \wedge Bad$, then add $\langle\langle m, N\rangle\rangle$ to Q.

Decide If there is a $t \in Q$, with $c = \langle m, i+1\rangle \in t$, $m_1 \rightarrow m$, $l_0 \wedge m_0^o \wedge m_1'$ is satisfiable, and $l_0 \wedge m_0^o \wedge m_1' \rightarrow F_i \wedge F_i^o \wedge Tr \wedge m'$ then add $\hat{t}$ to Q, where $\hat{t} = t$ with c replaced by two tuples $\langle l_0, i\rangle$, and $\langle m_0, i\rangle$.

Conflict If there is a $t \in Q$ with $c = \langle m, i+1\rangle \in t$, s.t. $\mathcal{F}(F_i) \wedge m'$ is unsatisfiable. Then, add $\varphi = \textsc{Itp}(\mathcal{F}(F_i), m')$ to F_j, for all $0 \leq j \leq i+1$.

Leaf If there is $t \in Q$ with $c = \langle m, i\rangle \in t$, $0 < i < N$ and $\mathcal{F}(F_{i-1}) \wedge m'$ is unsatisfiable, then add $\hat{t}$ to Q, where $\hat{t}$ is t with c replaced by $\langle m, i+1\rangle$.

Induction For $0 \leq i < N$ and a clause $(\varphi \vee \psi) \in F_i$, if $\varphi \notin F_{i+1}$, $\mathcal{F}(\phi \wedge F_i) \rightarrow \phi'$, then add φ to F_j, for all $j \leq i+1$.

until ∞;

counterexample is a tree

two predecessors

theory-aware **Conflict**

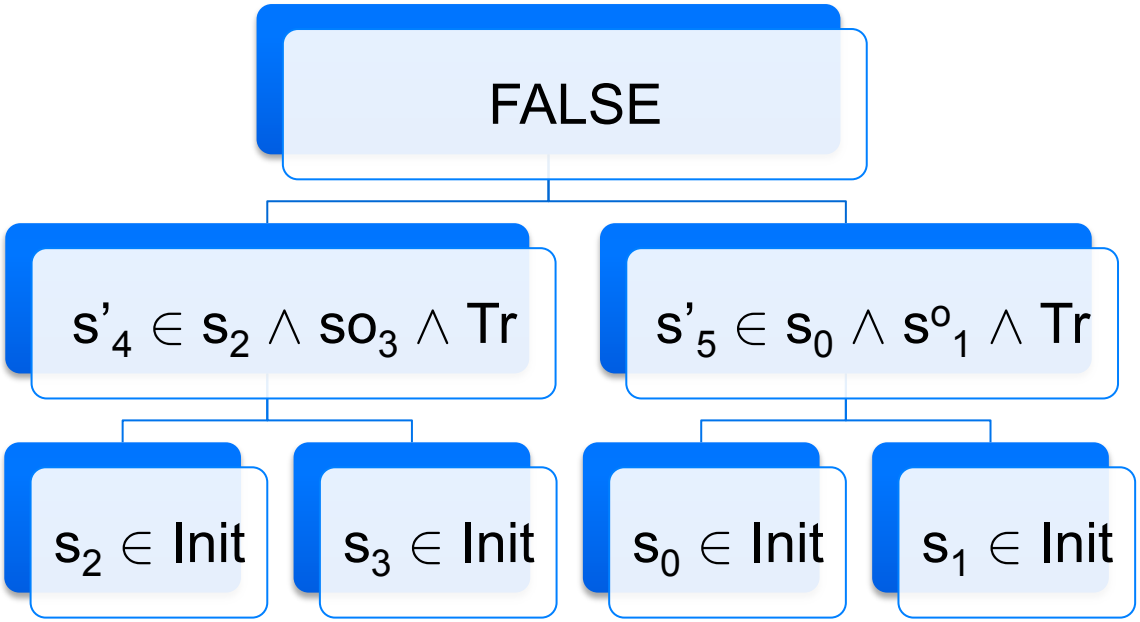
Counterexamples to non-linear CHC

A set S of CHC is unsatisfiable iff S can derive FALSE

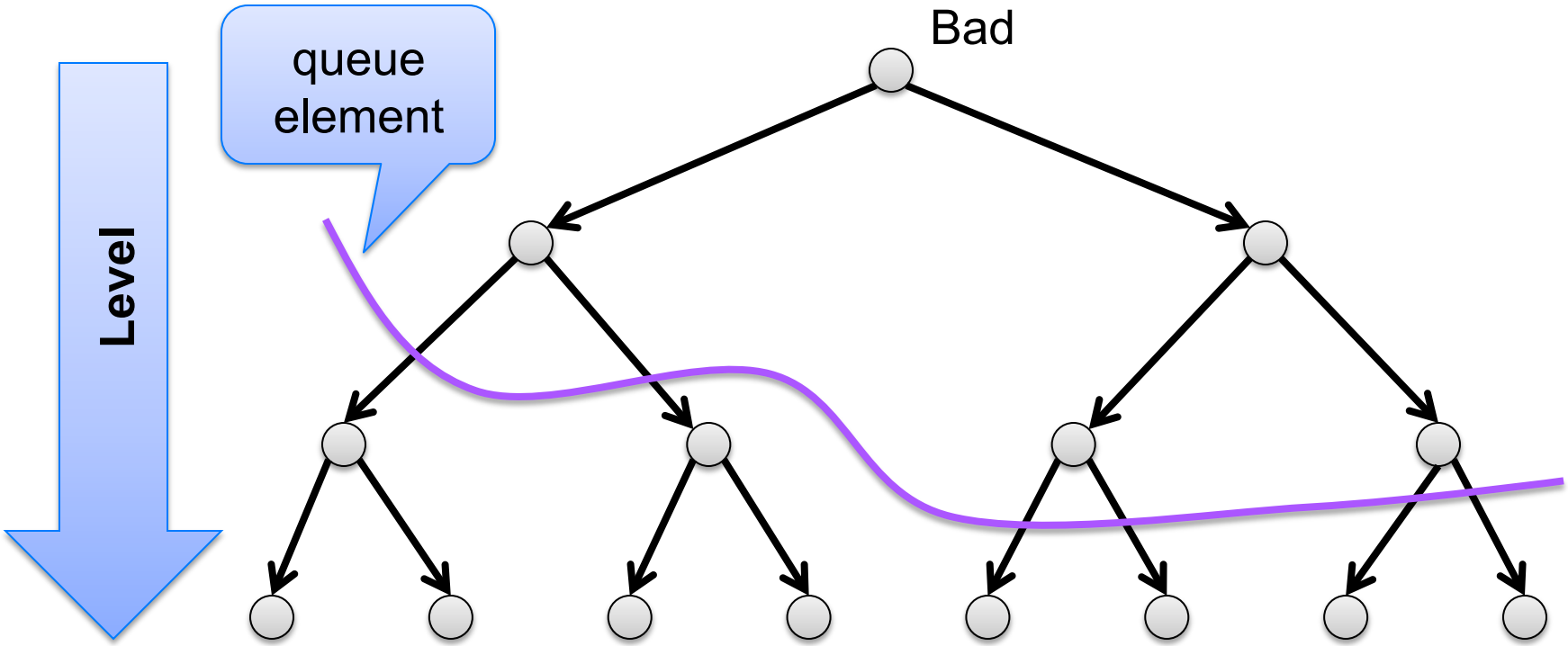
- we call such a derivation a counterexample

For linear CHC, the counterexample is a path

For non-linear CHC, the counterexample is a tree



GPDR Search Space



At each step, one CTI in the frontier is chosen and its two children are expanded

GPDR: Deciding predecessors

Decide If there is a $t \in Q$, with $c = \langle m, i+1 \rangle \in t$, $m_1 \rightarrow m$, $l_0 \wedge m_0^o \wedge m_1'$ is satisfiable, and $l_0 \wedge m_0^o \wedge m_1' \rightarrow F_i \wedge F_i^o \wedge Tr \wedge m'$ then add $\hat{t}$ to Q, where $\hat{t} = t$ with c replaced by two tuples $\langle l_0, i \rangle$, and $\langle m_0, i \rangle$.

Compute two predecessors at each application of **GPDR/Decide**

Can explore both predecessors in parallel

- e.g., BFS or DFS exploration order

Number of predecessors is unbounded

- incomplete even for finite problem (i.e., non-recursive CHC)
- Is compatible with MBP approach of APDR?

No caching/summarization of previous decisions

- worst-case exponential for Boolean Push-Down Systems

Spacer

Same queue as in IC3/PDR

Cache Reachable states

Three variants of **Decide**

Same **Conflict** as in APDR/GPDR

Input: A safety problem $\langle Init(X), Tr(X, X^o, X'), Bad(X)\rangle$.
Output: *Unreachable* or *Reachable*
Data: A cex queue Q, where a cex $c \in Q$ is a pair $\langle m, i\rangle$, m is a cube over state variables, and $i \in \mathbb{N}$. A level N. A set of reachable states REACH. A trace $F_0, F_1, \ldots$
Notation: $\mathcal{F}(A, B) = Init(X') \vee (A(X) \wedge B(X^o) \wedge Tr)$, and $\mathcal{F}(A) = \mathcal{F}(A, A)$
Initially: $Q = \emptyset$, $N = 0$, $F_0 = Init$, $\forall i > 0 \cdot F_i = \emptyset$, REACH $= Init$
Require: $Init \rightarrow \neg Bad$
repeat

Unreachable If there is an $i < N$ s.t. $F_i \subseteq F_{i+1}$ **return** *Unreachable*.

Reachable If REACH $\wedge$ *Bad* is satisfiable, **return** *Reachable*.

Unfold If $F_N \rightarrow \neg Bad$, then set $N \leftarrow N + 1$ and $Q \leftarrow \emptyset$.

Candidate If for some m, $m \rightarrow F_N \wedge Bad$, then add $\langle m, N\rangle$ to Q.

Successor If there is $\langle m, i+1\rangle \in Q$ and a model M $M \models \psi$, where $\psi = \mathcal{F}(\vee\text{REACH}) \wedge m'$. Then, add s to REACH, where $s' \in \text{MBP}(\{X, X^o\}, \psi)$.

DecideMust If there is $\langle m, i+1\rangle \in Q$, and a model M $M \models \psi$, where $\psi = \mathcal{F}(F_i, \vee\text{REACH}) \wedge m'$. Then, add s to Q, where $s \in \text{MBP}(\{X^o, X'\}, \psi)$.

DecideMay If there is $\langle m, i+1\rangle \in Q$ and a model M $M \models \psi$, where $\psi = \mathcal{F}(F_i) \wedge m'$. Then, add s to Q, where $s^o \in \text{MBP}(\{X, X'\}, \psi)$.

Conflict If there is an $\langle m, i+1\rangle \in Q$, s.t. $\mathcal{F}(F_i) \wedge m'$ is unsatisfiable. Then, add $\varphi = \text{ITP}(\mathcal{F}(F_i), m')$ to F_j, for all $0 \leq j \leq i+1$.

Leaf If $\langle m, i\rangle \in Q$, $0 < i < N$ and $\mathcal{F}(F_{i-1}) \wedge m'$ is unsatisfiable, then add $\langle m, i+1\rangle$ to Q.

Induction For $0 \leq i < N$ and a clause $(\varphi \vee \psi) \in F_i$, if $\varphi \notin F_{i+1}$, $\mathcal{F}(\phi \wedge F_i) \rightarrow \phi'$, then add φ to F_j, for all $j \leq i+1$.

until ∞;

Computing Reachable States

Successor If there is $\langle m, i+1\rangle \in Q$ and a model M $M \models \psi$, where $\psi = \mathcal{F}(\vee \textsc{Reach}) \wedge m'$. Then, add s to $\textsc{Reach}$, where $s' \in \text{MBP}(\{X, X^o\}, \psi)$.

Computing new reachable states by under-approximating forward image using MBP

- since MBP is finite, guarantee to exhaust all reachable states

Second use of MBP

- orthogonal to the use of MBP in Decide
- REACH can contain auxiliary variables, but might get too large

For Boolean CHC, the number of reachable states is bounded

- complexity is polynomial in the number of states
- same as reachability in Push Down Systems

Must and May refinement

DecideMust If there is $\langle m, i+1 \rangle \in Q$, and a model M $M \models \psi$, where $\psi = \mathcal{F}(F_i, \vee\text{REACH}) \wedge m'$. Then, add s to Q, where $s \in \text{MBP}(\{X^o, X'\}, \psi)$.

DecideMay If there is $\langle m, i+1 \rangle \in Q$ and a model M $M \models \psi$, where $\psi = \mathcal{F}(F_i) \wedge m'$. Then, add s to Q, where $s^o \in \text{MBP}(\{X, X'\}, \psi)$.

DecideMust

- use computed summary to skip over a call site

DecideMay

- use over-approximation of a calling context to guess an approximation of the call-site
- the call-site either refutes the approximation (**Conflict**) or refines it with a witness (**Successor**)

Conclusion

A program verifier is a compiler

- reusing an existing compiler is good idea, but comes with many caveats

Verification is Logic

- reduce verification to decidability of logic formulas
- CHC is a great target fragment for many verification tasks
- Greatly simplifies reasoning by discharging program semantics

An exciting direction with many extensions and open problems

- termination and model counting
- abstraction refinement and predicate abstraction
- abstract interpretation as model finding
- beyond arithmetic: arrays, memory, quantified models, separation logic, …
- program transformation for verification
- proof search strategies
- …

Contact Information

Arie Gurfinkel, Ph. D.
Sr. Researcher
CSC/SSD
Telephone: +1 412-268-5800
Email: info@sei.cmu.edu

U.S. Mail
Software Engineering Institute
Customer Relations
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
USA

Web
www.sei.cmu.edu
www.sei.cmu.edu/contact.cfm

Customer Relations
Email: info@sei.cmu.edu
Telephone: +1 412-268-5800
SEI Phone: +1 412-268-5800
SEI Fax: +1 412-268-6257