Solving Constrained Horn Clauses by Property Directed Reachability

Arie Gurfinkel

HCVS 2017: 4th Workshop on Horn Clauses for Verification and Synthesis

UNIVERSITY OF WATERLOO

Automated Verification

Deductive Verification

- A user provides a program and a verification certificate
  - e.g., inductive invariant, pre- and post-conditions, function summaries, etc.
- A tool automatically checks validity of the certificate
  - this is not easy! (might even be undecidable)
- Verification is manual but machine certified

Algorithmic Verification (My research area)

- A user provides a program and a desired specification
  - e.g., program never writes outside of allocated memory
- A tool automatically checks validity of the specification
  - and generates a verification certificate if the program is correct
  - and generates a counterexample if the program is not correct
- Verification is completely automatic – “push-button”

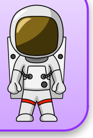
Algorithmic Logic-Based Verification

Safety Properties

Program + Spec

Constrained Horn Clauses

Verification Condition (in Logic)

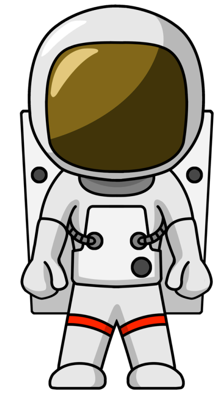
Spacer 

Decision Procedure

Yes

No

Spacer: Solving SMT-constrained CHC



Spacer: a solver for SMT-constrained Horn Clauses

- now part of Z3
  - https://github.com/Z3Prover/z3 since commit `72c4780`
  - use option `fixedpoint.engine=spacer`
- development version at http://bitbucket.org/spacer/code

Supported SMT-Theories

- Best-effort support for many SMT-theories
  - data-structures, bit-vectors, non-linear arithmetic
- Linear Real and Integer Arithmetic
- Quantifier-free theory of arrays
- *Universally quantified theory of arrays + arithmetic (work in progress)*

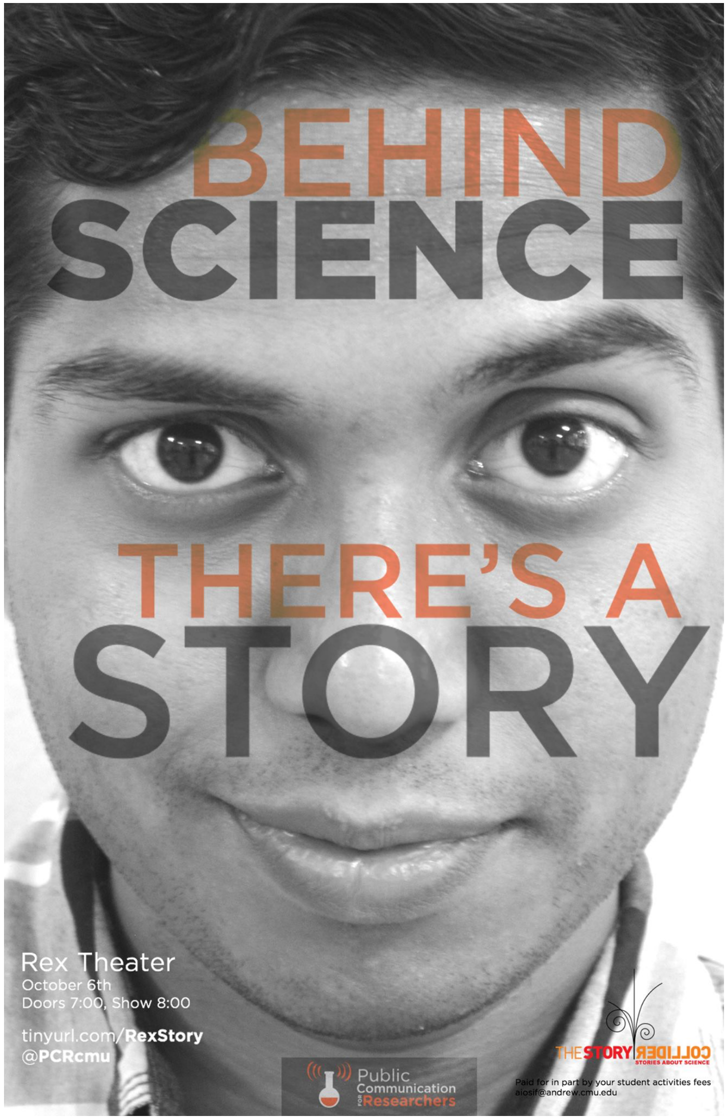
Support for Non-Linear CHC

- for procedure summaries in inter-procedural verification conditions
- for compositional reasoning: abstraction, assume-guarantee, thread modular, etc.

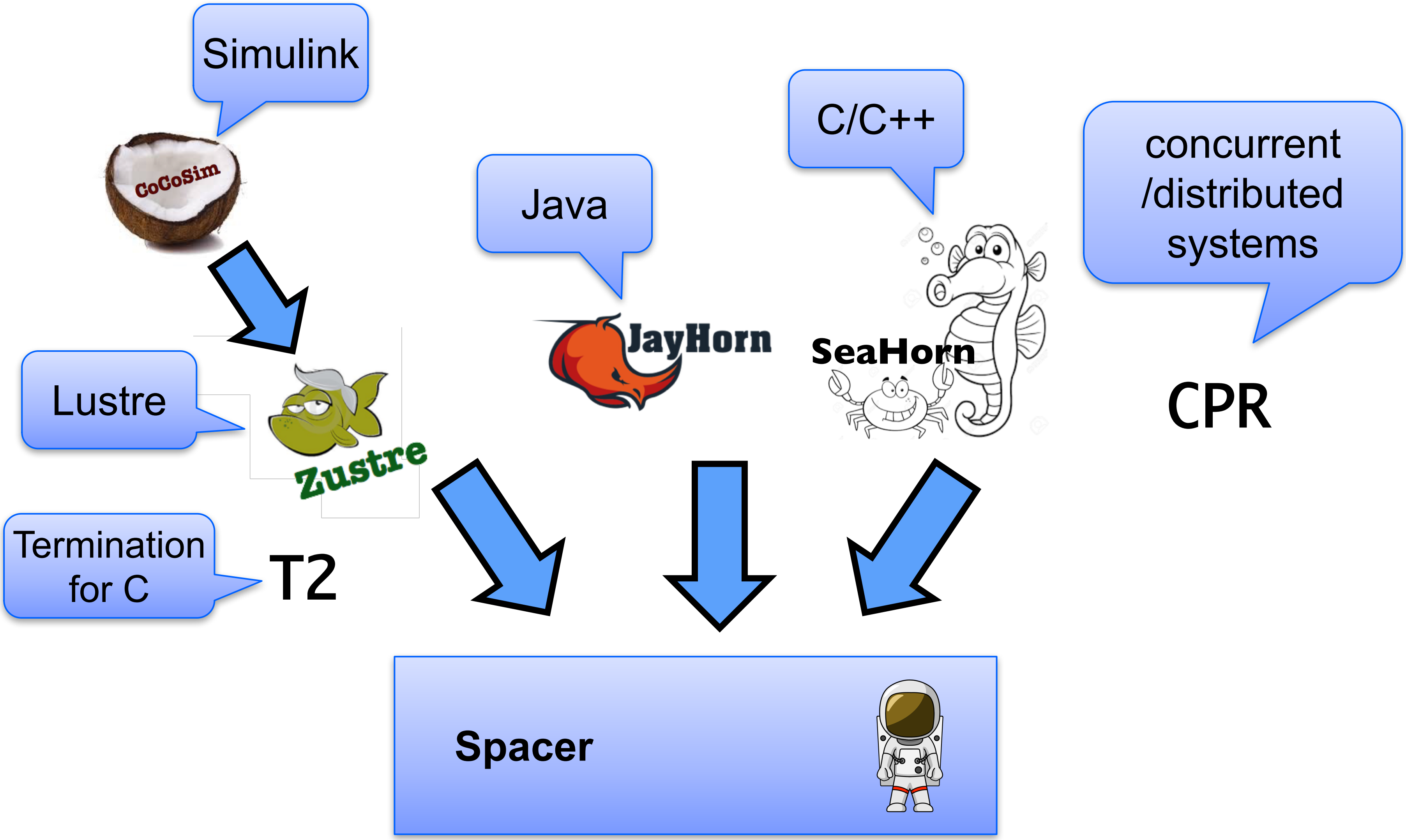
Contributors

Arie Gurfinkel
Anvesh Komuravelli

Nikolaj Bjorner
(Krystof Hoder)
Yakir Vizel
Bernhard Gleiss
Matteo Marescotti



Logic-based Algorithmic Verification



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

UNIVERSITY OF WATERLOO

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

Given a theory A, a model M is **A-definable**, it each p_i in M 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

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

ArrayPDR: CHC with constraints over Airthmetic + Arrays

- Required to model heap manipulating programs
- A. Komuravelli, N. Bjørner, A. Gurfinkel, K. L. McMillan:Compositional Verification of Procedural Programs using Horn Clauses over Integers and Arrays. FMCAD 2015

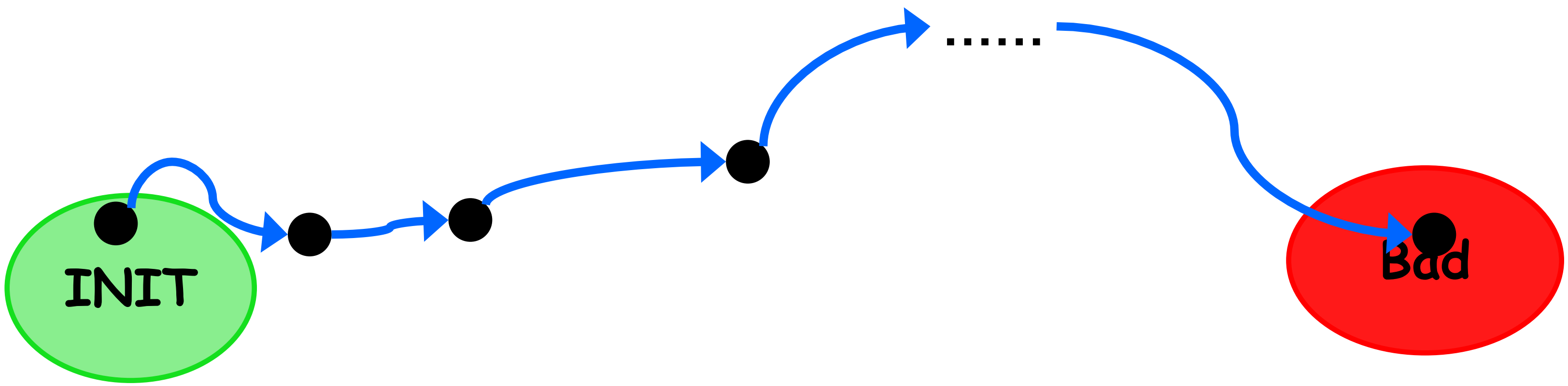
UNIVERSITY OF WATERLOO

Safety Verification Problem

Is Bad reachable?

Safety Verification Problem

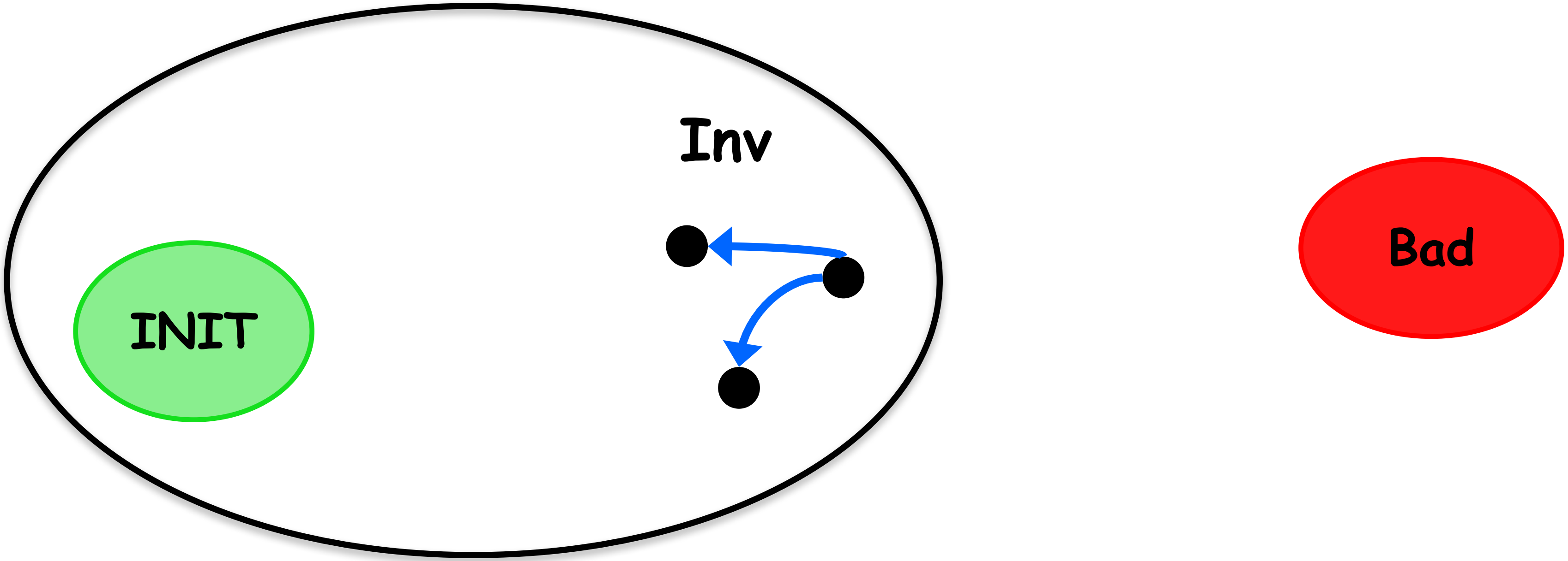
Is Bad reachable?



Yes. There is a counterexample!

Safety Verification Problem

Is Bad reachable?



No. There is an inductive invariant

Programs, Cexs, Invariants

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

- Notation: $\mathcal{F}$(X) = ∃ ***u*** . (X ∧ $\mathcal{Tr}$) ∨ 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.*

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

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

IC3/PDR Overview

bounded safety

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

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

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

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

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

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

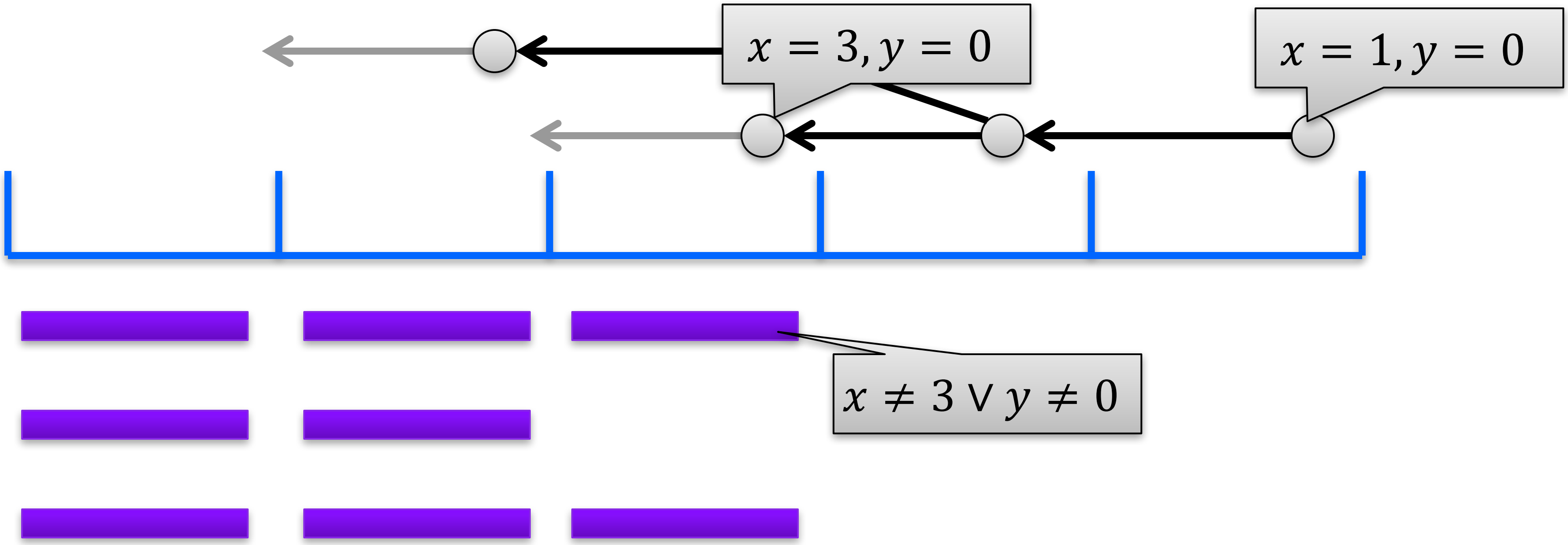
$\quad$ **if** $\exists 0 \leq i < N \cdot F_i = F_{i+1}$ **then return** *Unreachable*;

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

until ∞;

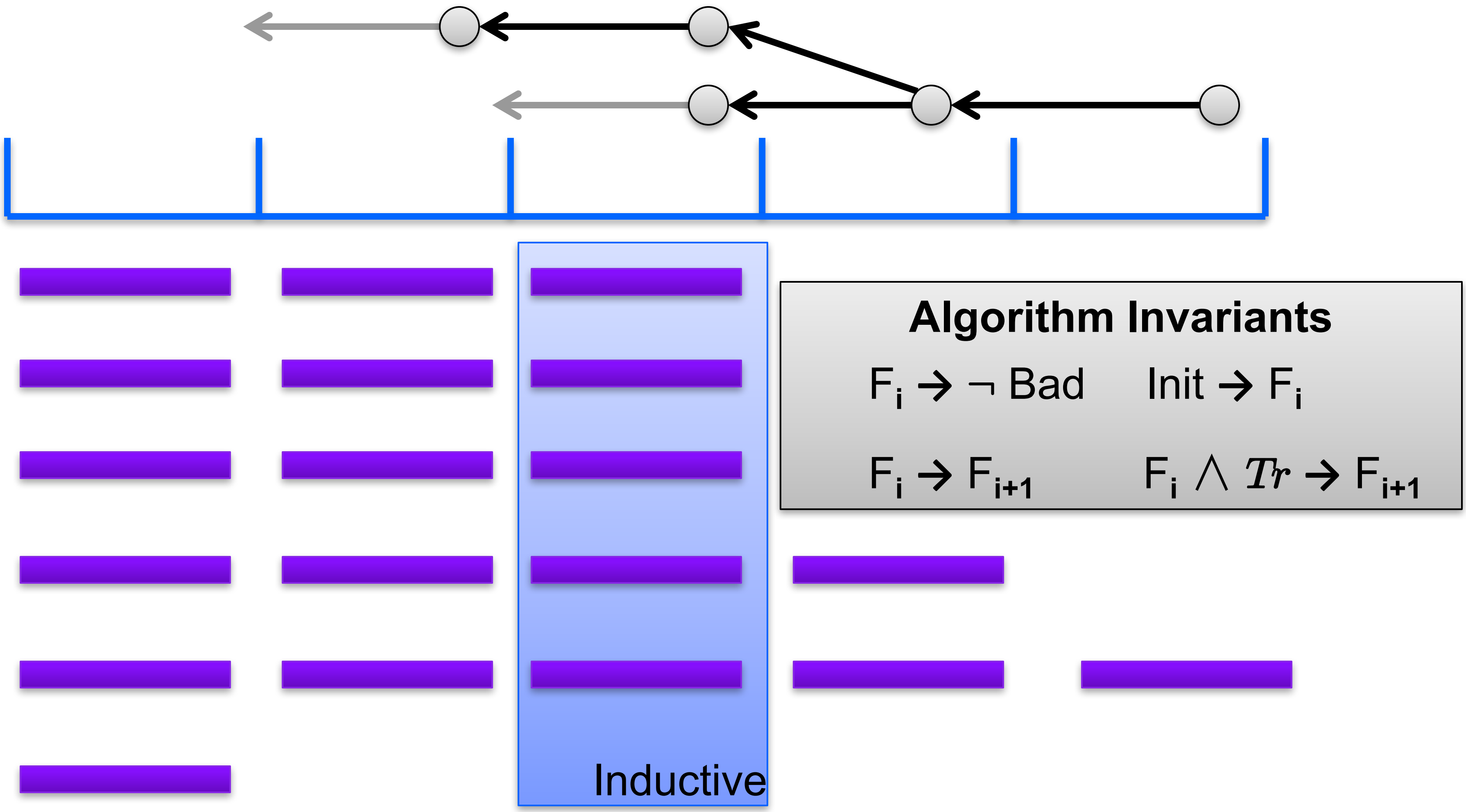
strengthen result

IC3/PDR In Pictures: MkSafe



Push

IC3/PDR in Pictures: Push



IC3/PDR: Solving Linear (Propositional) CHC

Unreachable and Reachable

- terminate the algorithm when a solution is found

Unfold

- increase search bound by 1

Candidate

- choose a bad state in the last frame

Decide

- extend a cex (backward) consistent with the current frame
- choose an assignment ***s*** s.t. $(s \wedge R_i \wedge Tr \wedge cex')$ is SAT

Conflict

- construct a lemma to explain why cex cannot be extended
- Find a clause ***L*** s.t. $L \Rightarrow \neg cex$, $Init \Rightarrow L$, and $L \wedge R_i \wedge Tr \Rightarrow L'$

Induction

- propagate a lemma as far into the future as possible
- (optionally) strengthen by dropping literals

Decide Rule: Generalizing Predecessors

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

- 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

Conflict Rule: Inductive Generalization

Conflict For $0 \leq i < N$: given a candidate model $\langle m, i+1 \rangle \in Q$ and clause φ, such that $\varphi \to \neg m$, if $Init \to \varphi$, and $\varphi \wedge F_i \wedge Tr \to \varphi'$, 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, …, F_N be a valid IC3 trace. If φ is inductive relative to F_i, $0 \cdot i < N$, then, for all $j \cdot i$, φ is inductive relative to F_j.

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

UNIVERSITY OF WATERLOO

From Propositional PDR to Solving CHC

Infinite Theories

- infinitely many satisfying assignments
- can’t simply enumerate (in decide)
- can’t block one assignment at a time (in conflict)

Non-Linear Horn Clauses

- multiple predecessors (in decide)

The problem is undecidable in general, but we want an algorithm that makes progress

- don’t get stuck in a decidable fragment

PDR FOR ARITHMETIC CHC

IC3/PDR: Solving Linear (Propositional) CHC

Unreachable and Reachable

- terminate the algorithm when a solution is found

Unfold

- increase search bound by 1

Candidate

- choose a bad state in the last frame

Theory dependent

Decide

- extend a cex (backward) consistent with the current frame
- choose an assignment ***s*** s.t. $(s \wedge R_i \wedge Tr \wedge cex')$ is SAT

Conflict

- construct a lemma to explain why cex cannot be extended
- Find a clause ***L*** s.t. $L \Rightarrow \neg cex$, $Init \Rightarrow L$, and $L \wedge R_i \wedge Tr \Rightarrow L'$

Induction

- propagate a lemma as far into the future as possible
- (optionally) strengthen by dropping literals

$$((F_i \wedge Tr) \vee Init') \Rightarrow \varphi'$$

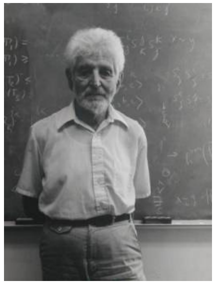
$$\varphi' \Rightarrow \neg c'$$

Looking for φ’

ARITHMETIC CONFLICT

UNIVERSITY OF WATERLOO

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

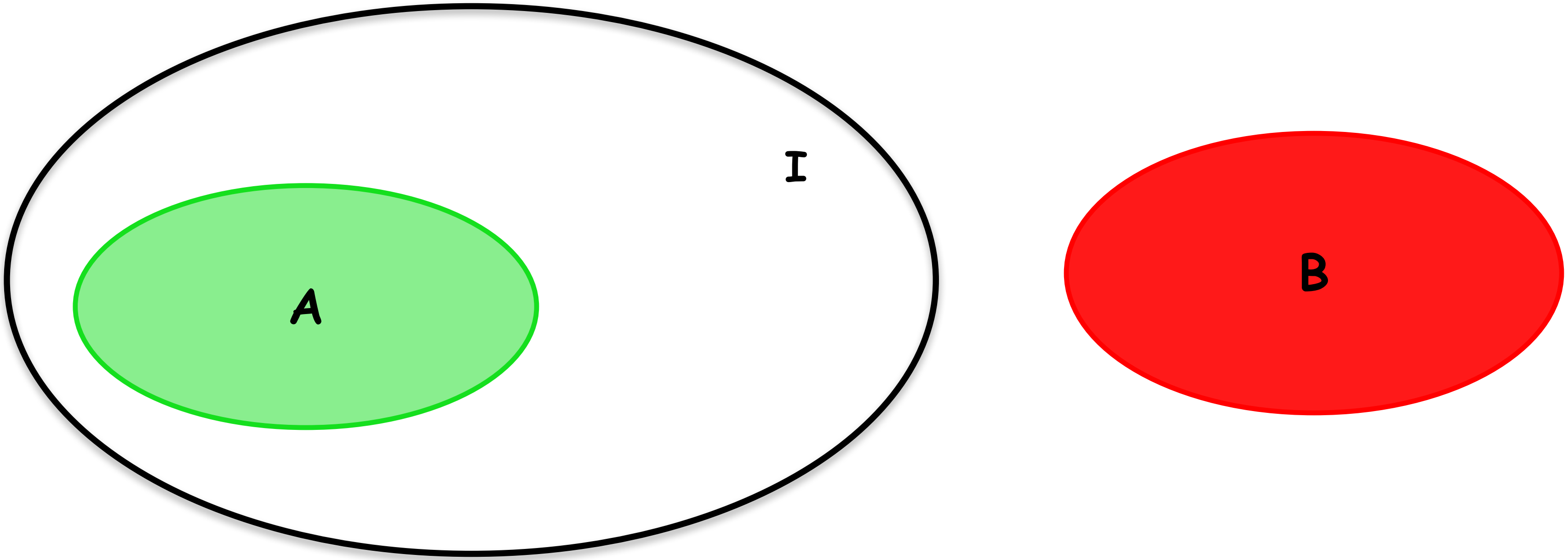
$$A \Rightarrow I \qquad I \Rightarrow \neg B$$

$$\mathit{atoms}(I) \in \mathit{atoms}(A) \cap \mathit{atoms}(B)$$

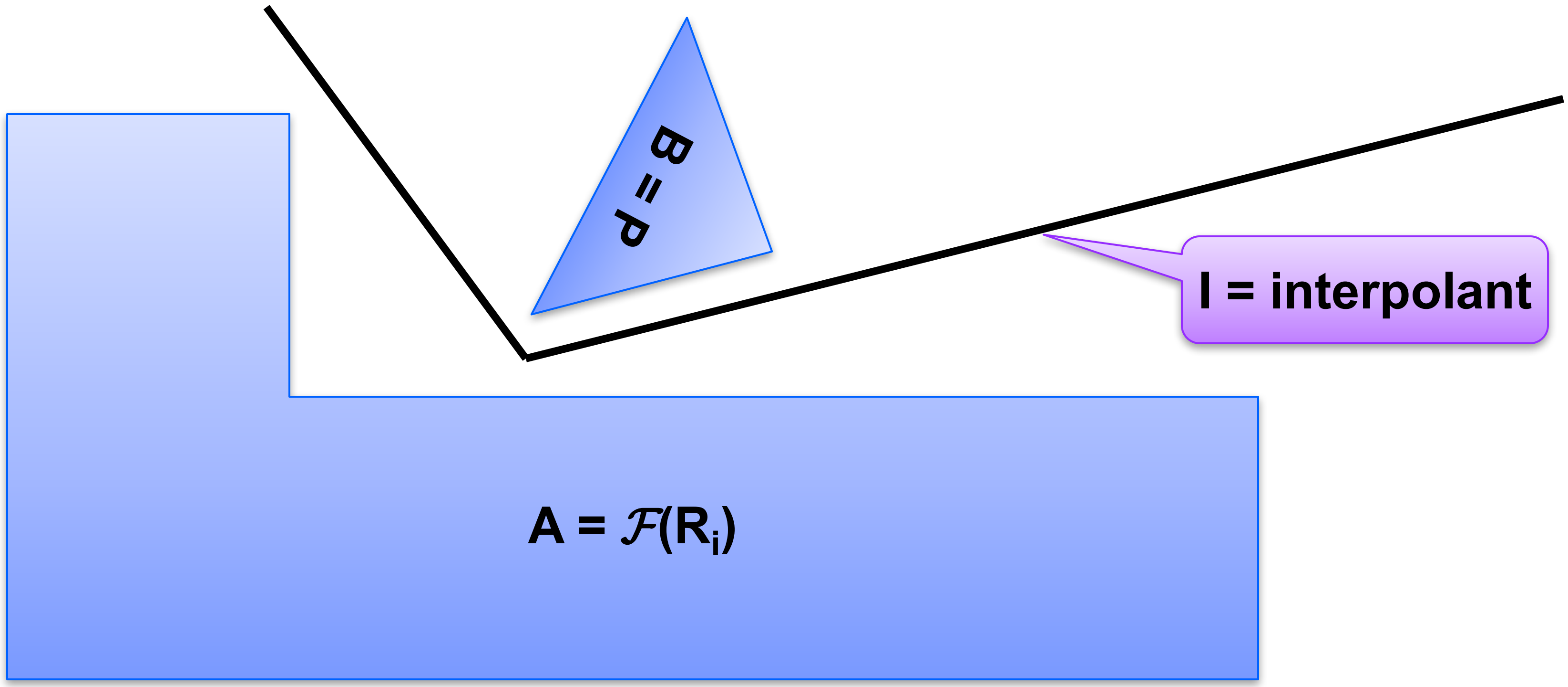
A Craig interpolant ITP(A, B) can be effectively constructed from a resolution proof of unsatisfiability of $A \wedge B$

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

Craig Interpolant



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

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

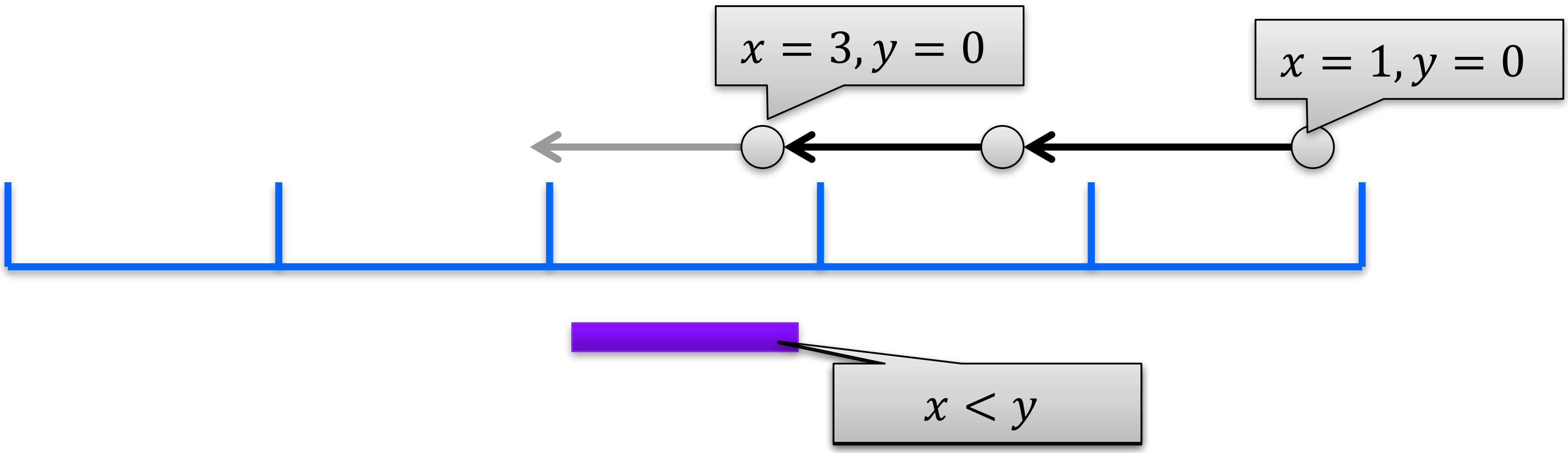
Generalization is not inductive

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

UNIVERSITY OF WATERLOO

IC3/PDR In Pictures: MkSafe

MkSafe



Computing Interpolants for IC3/PDR

Much simpler than general interpolation problem for $A \wedge B$

- B is always a conjunction of literals
- A is dynamically split into DNF by the SMT solver
- DPLL(T) proofs do not introduce new literals

Interpolation algorithm is reduced to analyzing all theory lemmas in a DPLL(T) proof produced by the solver

- every theory-lemma that mixes B-pure literals with other literals is interpolated to produce a single literal in the final solution
- interpolation is restricted to clauses of the form $(\bigwedge B_i \Rightarrow \vee A_j)$

Interpolating (UNSAT) Cores *(ongoing work with Bernhard Gleiss)*

- improve interpolation algorithms and definitions to the specific case of PDR
- classical interpolation focuses on eliminating non-shared literals
- in PDR, the focus is on finding good generalizations

$$s \subseteq pre(c)$$

$$\equiv \quad s \Rightarrow \exists X' \,.\, Tr \wedge c'$$

Computing a predecessor ***s*** of a counterexample ***c***

ARITHMETIC DECIDE

UNIVERSITY OF WATERLOO

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 Projection of U, M and φ iff

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

Model Based Projection under-approximates existential quantifier elimination relative to a given model (i.e., satisfying assignment)

Loos-Weispfenning Quantifier Elimination

φ is LRA formula in Negation Normal Form

E is set of x=t atoms, U set of x < t atoms, and L set of s < x atoms

There are no other occurrences of x in φ[x]

$$\exists x . \varphi[x] \equiv \varphi[\infty] \lor \bigvee_{x=t \in E} \varphi[t] \lor \bigvee_{x<t \in U} \varphi[t - \epsilon]$$

where

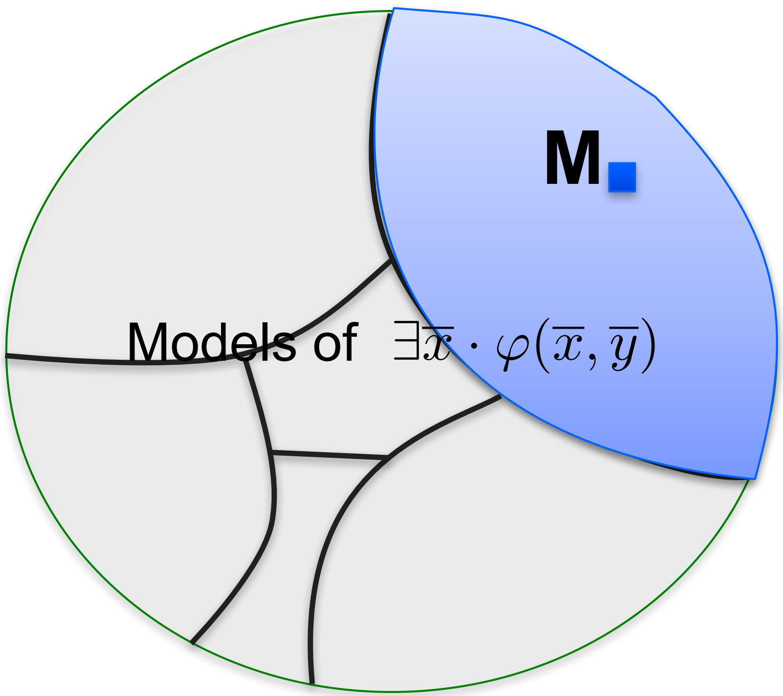
$$(x < t')[t - \epsilon] \equiv t \leq t' \qquad (s < x)[t - \epsilon] \equiv s < t \qquad (x = e)[t - \epsilon] \equiv \mathit{false}$$

The case of lower bounds is dual

- using –∞ and t+ϵ

Model Based Projection

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



1. Find model M of φ (x,y)

2. Compute a partition containing M

UNIVERSITY OF WATERLOO

MBP for Linear Rational Arithmetic

Compute a **single** disjunct from LW-QE that includes the model

- Use the Model to uniquely pick a substitution term for x

$$Mbp_x(M, x = s \wedge L) = L[x \leftarrow s]$$

$$Mbp_x(M, x \neq s \wedge L) = Mbp_x(M, s < x \wedge L) \text{ if } M(x) > M(s)$$

$$Mbp_x(M, x \neq s \wedge L) = Mbp_x(M, -s < -x \wedge L) \text{ if } M(x) < M(s)$$

$$Mbp_x(M, \bigwedge_i s_i < x \wedge \bigwedge_j x < t_j) = \bigwedge_i s_i < t_0 \wedge \bigwedge_j t_0 \leq t_j \text{ where } M(t_0) \leq M(t_i), \forall i$$

MBP techniques have been developed for

- Linear Rational Arithmetic, Linear Integer Arithmetic
- Theories of Arrays, and Recursive Data Types

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**

UNIVERSITY OF WATERLOO

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

$$Init(X) \to P(X)$$

$$P(X) \to !Bad(X)$$

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

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

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: $Init \to \neg Bad$
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 $F_N \to \neg Bad$, then set $N \leftarrow N + 1$ and $Q \leftarrow \emptyset$.
- **Candidate** If for some m, $m \to F_N \wedge Bad$, then add $\langle \langle m, N \rangle \rangle$ to Q.
- **Decide** If there is a $t \in Q$, with $c = \langle m, i+1 \rangle \in t$, $m_1 \to m$, $l_0 \wedge m_0^o \wedge m_1'$ is satisfiable, and $l_0 \wedge m_0^o \wedge m_1' \to F_i \wedge F_i^o \wedge Tr \wedge m'$ 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 $0 \le j \le i + 1$.
- **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 $0 \le i < N$ and a clause $(\varphi \vee \psi) \in F_i$, if $\varphi \notin F_{i+1}$, $\mathcal{F}(\phi \wedge F_i) \to \phi'$, then add φ to F_j, for all $j \le i + 1$.

until ∞;

counterexample is a tree

two predecessors

theory-aware **Conflict**

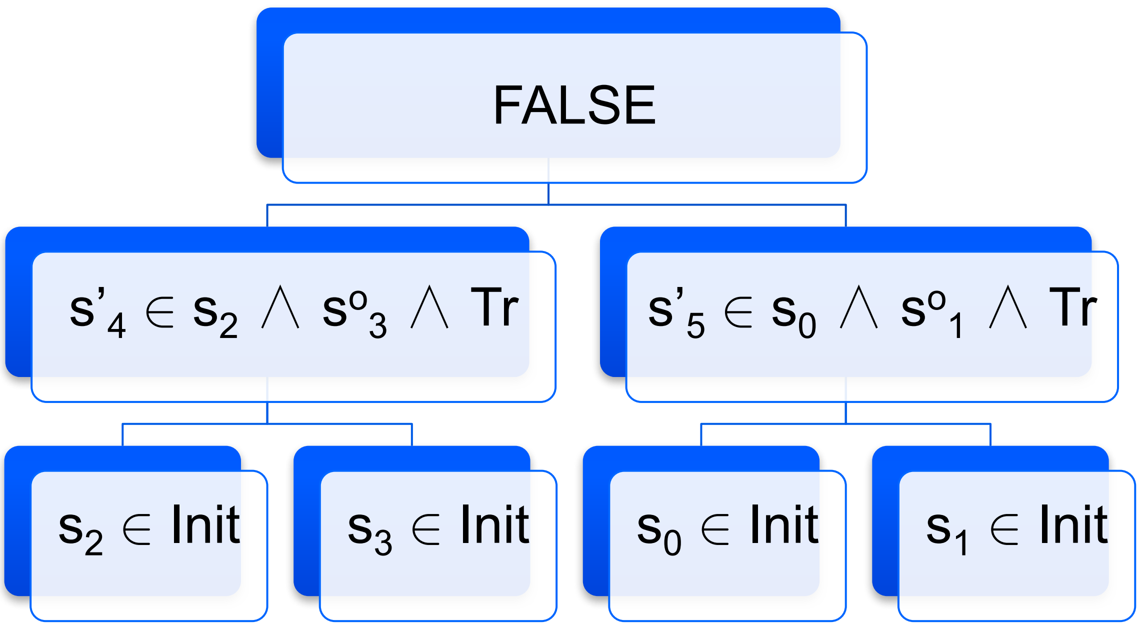
Counterexamples to non-linear CHC

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

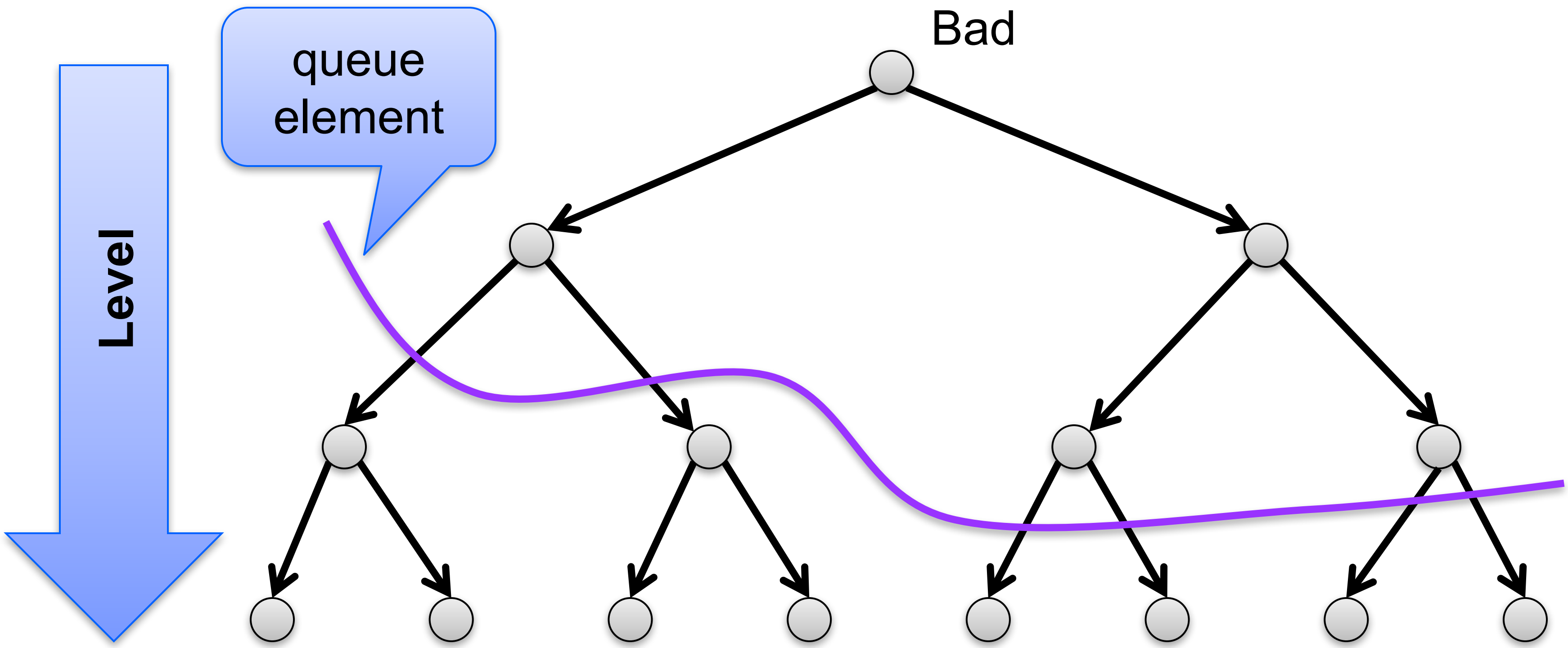
- we call such a derivation a counterexample

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)

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 $\textsc{Reach}$. 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: $Q = \emptyset$, $N = 0$, $F_0 = Init$, $\forall i > 0 \cdot F_i = \emptyset$, $\textsc{Reach} = Init$
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 $\textsc{Reach} \wedge Bad$ 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 $\psi = \mathcal{F}(\vee\textsc{Reach}) \wedge m'$. Then, add s to $\textsc{Reach}$, 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 $\psi = \mathcal{F}(F_i, \vee\textsc{Reach}) \wedge m'$. 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 $s^o \in \text{MBP}(\{X, X'\}, \psi)$.

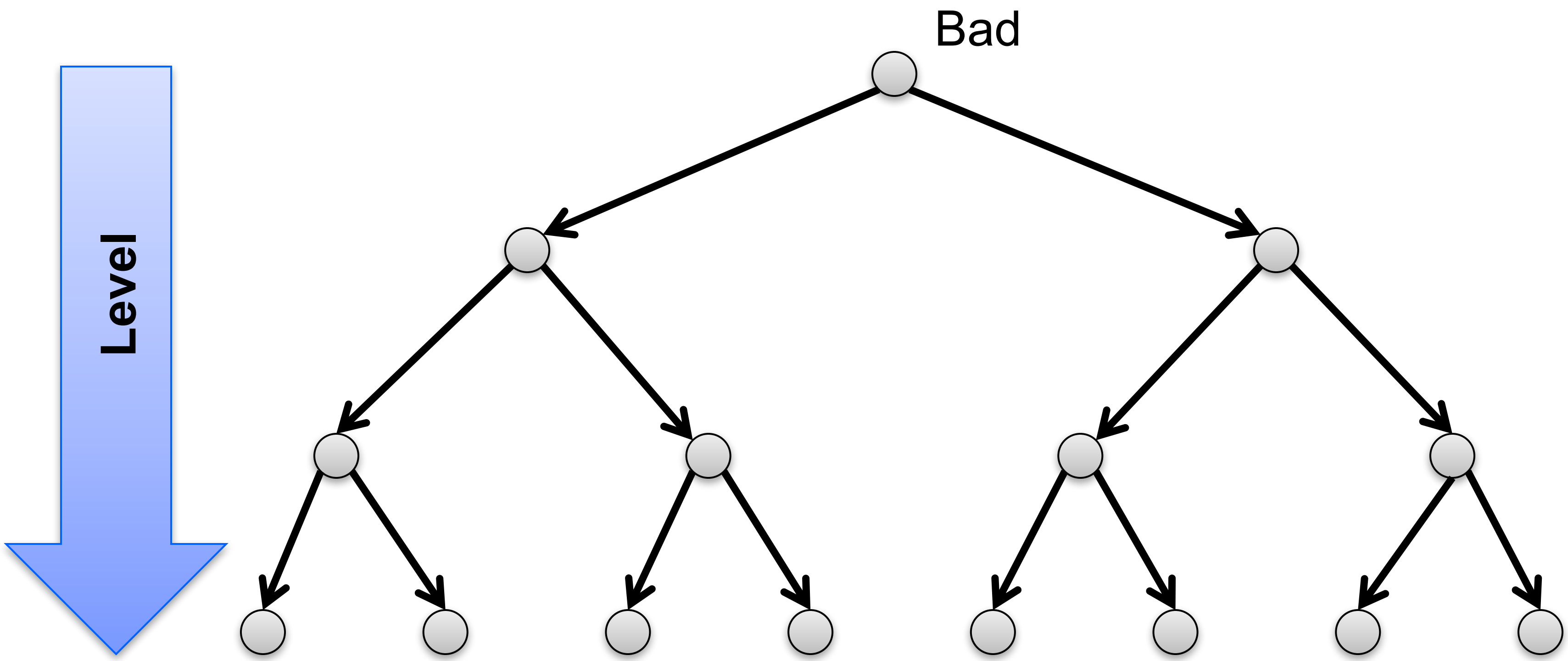
Conflict If there is an $\langle m, i+1\rangle \in Q$, s.t. $\mathcal{F}(F_i) \wedge m'$ is unsatisfiable. Then, add $\varphi = \textsc{Itp}(\mathcal{F}(F_i), m')$ to F_j, for all $0 \le j \le i+1$.

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 $0 \le i < N$ 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 $j \le i+1$.

until ∞;

SPACER Search Space



Unfold the derivation tree in a fixed depth-first order

- use MBP to decide on counterexamples

Learn new facts (reachable states) on the way up

- use MBP to propagate facts bottom up

Successor Rule: 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 $s' \in \mathrm{MBP}(\{X, X^o\}, \psi)$.

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

Decide Rule: Must and May refinement

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

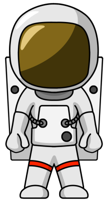
DecideMust

- 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 and Future Work



Spacer: an SMT-based procedure for deciding CHC modulo theories

- extends IC3/PDR from SAT to SMT
- interpolation to over-approximate a possible model
- model-based projection to summarize derivations

The curse of interpolation

- interpolation is fantastic at quickly discovering good lemmas
- BUT it is highly unstable: small changes to input (or code) drastically change what is discovered
- what is easy today might be difficult tomorrow ☹

Harnessing the power of parallelism (see FMCAD'17)

- Spacer is highly non-deterministic: many sound choices for bounded exploration and lemma generation
- Lemmas (invariants) are easy to share between multiple instances
- Problems are naturally partitioned in Decide rule

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

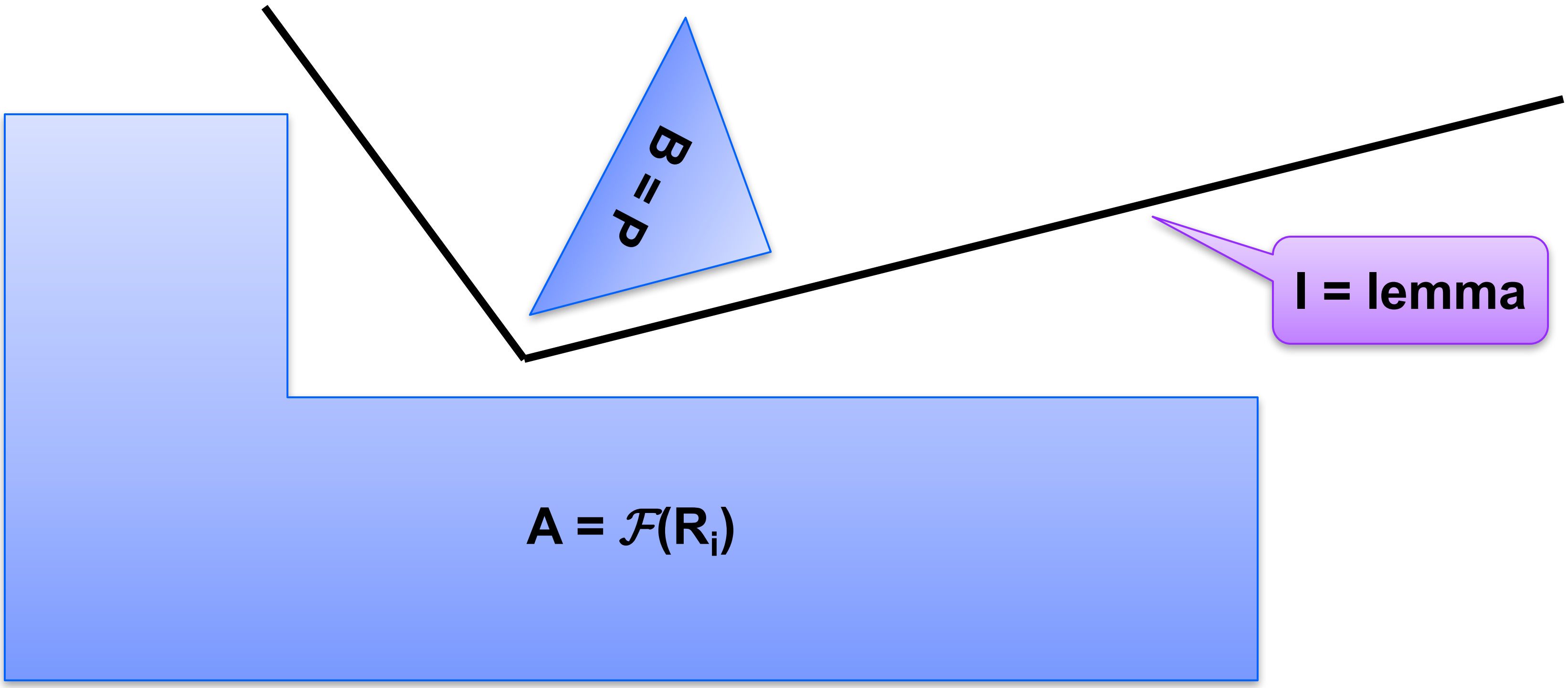
Then

- $g_1 \times (t_1 \geq b_1) + \ldots + g_i \times (t_i \geq b_i)$ 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

UNIVERSITY OF WATERLOO

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