SMT and Z3

Nikolaj Bjørner
Microsoft Research
ReRISE Winter School, Linz, Austria
February 6, 2014

Plan

Mon An invitation to SMT with Z3

Tue Equalities and Theory Combination

Wed Theories: Arithmetic, Arrays, Data types

Thu Quantifiers and Theories

Fri Programming Z3: Interfacing and Solving

Plan

- Model-based Quantifier Instantiation
- Quantifier Elimination
- Solving Horn Clauses

Quiz

Show: A *difference logic graph* without negative cycles has a model. Give a procedure for extracting a model.

True or false: A formula over integer difference logic has a model over reals **iff** it has a model over integers?

Give an *efficient* algorithm to extract models for UTVPI over integers.

Encode lambda Calculus into $map, K, read$ (without I).

A taste of

MODEL BASED QUANTIFIER INSTANTIATION

MBQI: Motivation

Template Based Invariant Geneneration

```
pre
while (c) {
   T
}
post
```

$$\varphi_I: \begin{array}{l} \forall s.\ pre[s] \rightarrow I(s) \wedge \\ \forall s,s'.\ I(s) \wedge c[s] \wedge T[s,s'] \rightarrow I(s') \wedge \\ \forall s.\ I(s) \wedge \neg c[s] \rightarrow post[s] \end{array}$$

If φ_I is *satisfiable,* then *post* holds

MBQI: Example

```
assert(n >= 0);
x = 0; y = 0;
while (x < n) {
  x = x + 1;
  y = y + 2;
}
assert(y == 2*n);
```

$$\forall x, y, n \,.\, I(x, y, n) \wedge \neg(x < n) \rightarrow y = 2n$$

$$\forall x, y, n, x', y', n' .\; I(x, y, n) \wedge x' = x + 1 \wedge y' = y + 2 \wedge n' = n \rightarrow I(x', y', n')$$

$$\forall x, y, n \,.\, I(x, y, n) \wedge \neg(x < n) \rightarrow y = 2x$$

If φ_I is *satisfiable,* then *post* holds

MBQI: Example

$\forall x, y, n \,.\, I(x, y, n) \wedge \neg(x < n) \rightarrow y = 2n$
$\forall x, y, n, x', y', n' .\ I(x, y, n) \wedge x' = x + 1 \wedge y' = y + 2 \wedge n' = n \rightarrow I(x', y', n')$
$\forall x, y, n \,.\, I(x, y, n) \wedge \neg(x < n) \rightarrow y = 2x$

Is $\varphi_I \coloneqq \forall \vec{x} \psi[\vec{x}]$ *satisfiable?*

Initial model $I^M(x, y, n) \coloneqq true$

Then $\mathrm{M} \vDash \forall x, y, n \,.\, I(x, y, n) \wedge \neg(x < n) \rightarrow y = 2n$

Iff $\vDash \forall x, y, n \,.\, (true \wedge \neg(x < n) \rightarrow y = 2n$

Iff $\neg(true \wedge \neg(x < n) \rightarrow y = 2n)$ is unsat

But it is satisfied with: $x = y = n = 1$

MBQI: Example

$\forall x, y, n\,.\, I(x, y, n) \wedge \neg(x < n) \rightarrow y = 2n$
$\forall x, y, n, x', y', n'.\ I(x, y, n) \wedge x' = x + 1 \wedge y' = y + 2 \wedge n' = n \rightarrow I(x', y', n')$
$\forall x, y, n\,.\, I(x, y, n) \wedge \neg(x < n) \rightarrow y = 2x$

$\neg I(1,1,1)$
$\neg I(2,2,1)$
$\neg I(3,3,1)$
$\neg I(4,4,1)$
$\neg I(5,6,1)$
$\neg I(6,7,1)$
$\neg I(7,7,1)$
$\neg I(8,7,1)$
$\neg I(9,7,1)$
$\neg I(10,7,1)$
$\neg I(11,7,1)$

$I^M(x, y, n) \coloneqq if\ x = 1 \wedge y = 1 \wedge n = 1\ then\ false\ else\ if\ \ldots$

MBQI: Example

- Idea: Add template macros for *I:*

$\forall x, y, n\,.\, I(x, y, n) \wedge \neg(x < n) \rightarrow y = 2n$
$\forall x, y, n, x', y', n'.\ I(x, y, n) \wedge x' = x + 1 \wedge y' = y + 2 \wedge n' = n \rightarrow I(x', y', n')$
$\forall x, y, n\,.\, I(x, y, n) \wedge \neg(x < n) \rightarrow y = 2x$

$\forall x, y, n\,.\, I(x, y, n) \leftrightarrow ax + by + cn + d \leq 0 \wedge \ \ldots$

$a = -1, b = 0, c = 1, I = \lambda x, y, n\,.\, ax + by + cn + d \leq 0 \wedge \ \ldots$

From virtual substitutions to constraints for

QUANTIFIER ELIMINATION

Loos-Weispfenning $\Rightarrow$ Abstract QE(LRA)

Terms $\quad t, s \ ::= a_1 x_1 + a_2 x_2 + \ \ldots + a_k x_k + c$

Atoms $\quad atom ::= x > s \mid x = t \mid \ x \ < \ t$

Formulas $\quad \varphi ::= \varphi \wedge \varphi \mid \varphi \vee \varphi \mid atom$

$$\exists x \,.\, \varphi\big[x < t_i, x > s_j, x = t_k\big] \equiv$$

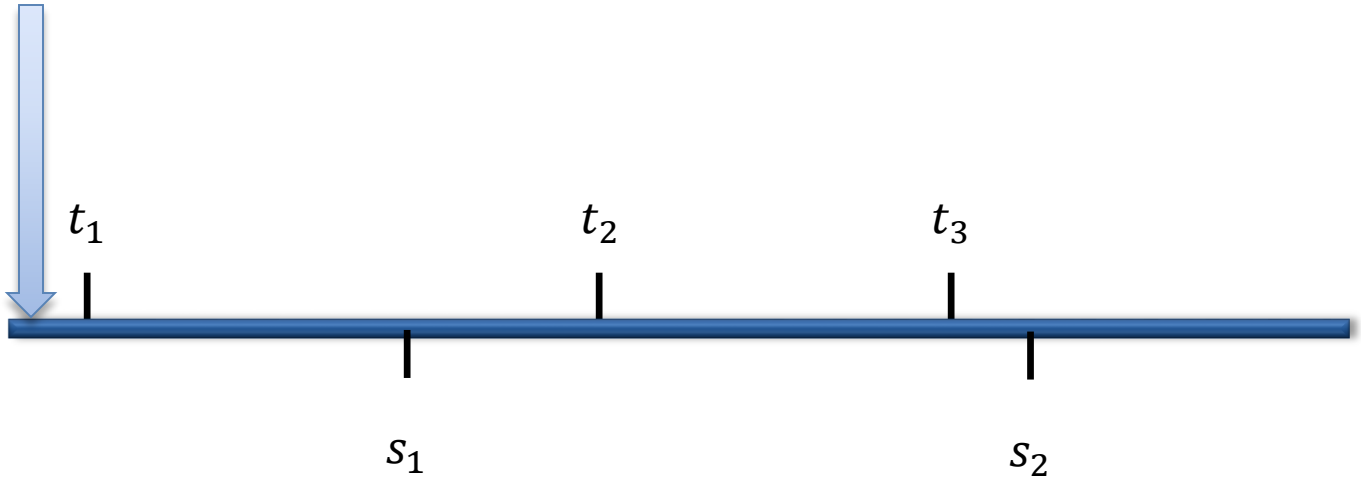
$$\varphi\big[\infty < t_i, \infty > s_j, \infty = t_k\big] \ \vee \qquad x \ is \ large$$

$$\bigvee_k \varphi\big[t_k < t_i, t_k > s_j, t_k = t_{k'}\big] \vee \qquad x \ is \ t_k$$

$$\bigvee_i \varphi\big[t_i - \epsilon < t_{i'}, t_i - \epsilon > s_j, t_i - \epsilon = t_k\big] \qquad lub. of \ x \ is \ t_i$$

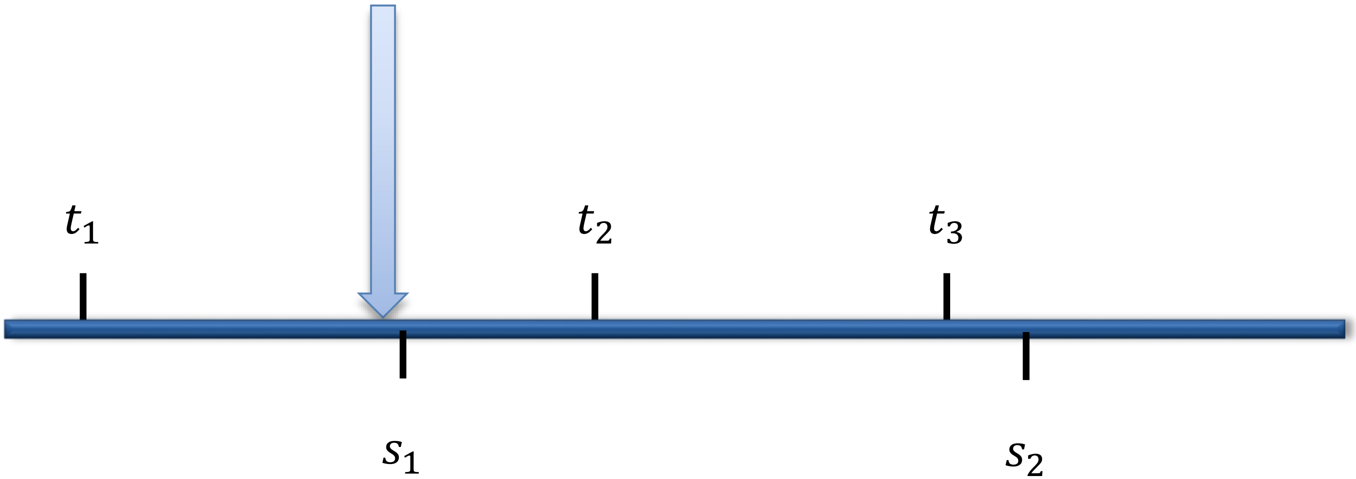
$\varphi[x < t_1, x < t_2, x = t_3, x > s_1, x > s_2]$ t_1 *is lub. for* x

false true false false false



$\varphi[x < t_1, x < t_2, x = t_3, x > s_1, x > s_2]$ $x < t_2$

false ***true*** ***false*** ***true*** ***false***



Observation

$\varphi[x < t_1, x < t_2, x = t_3, x > s_1, x > s_2]$

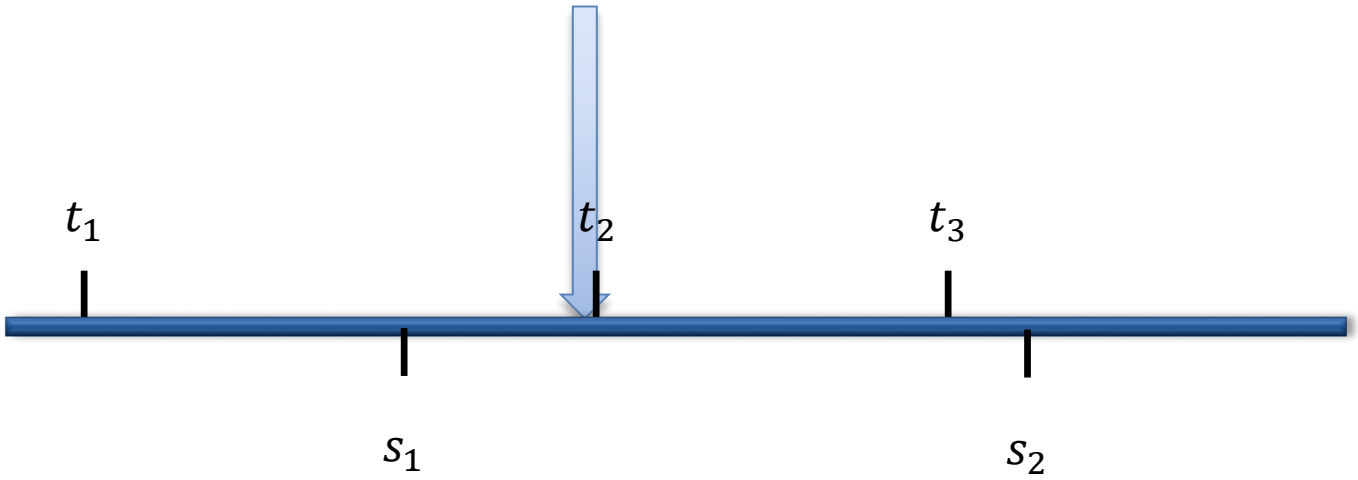
$\boldsymbol{false\ \ true\ \ false\ \ false\ \ false}$ Non-tight value $x < t_2$

$\boldsymbol{false\ \ true\ \ false\ \ true\ \ false}$ Tightest value $x < t_2$

Only have to consider least upper bounds for $x\ wrt\ t_1, t_2$

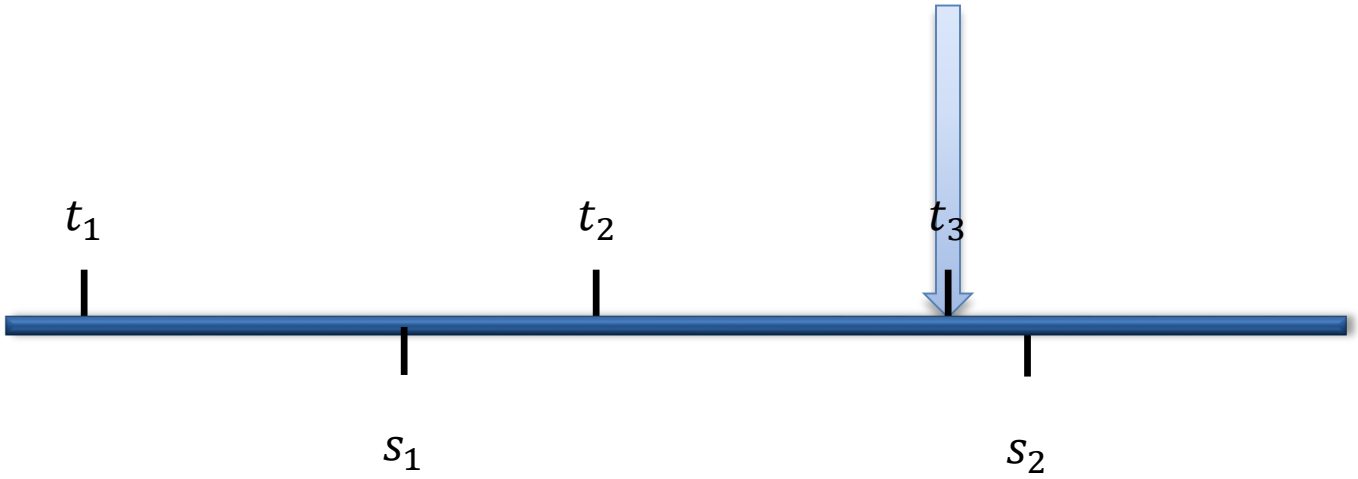
$\varphi[x < t_1, x < t_2, x = t_3, x > s_1, x > s_2]$ $t_3 = x$

false false true true false



$\varphi[x < t_1, x < t_2, x = t_3, x > s_1, x > s_2]$ $x\ is\ bigger\ than\ t_1, t_2, t_3, s_1, s_2$

false false false true true



$t_1 is\ lub.\ for\ x,$ $t_2\ is\ lub.\ for\ x,$ $t_3 = x,$ $x\ is\ bigger\ than\ t_1, t_2, t_3, s_1, s_2$

Loos-Weispfenning $\Rightarrow$ Abstract QE(LRA)

Terms $\quad t, s \ ::= a_1 x_1 + a_2 x_2 + \ \ldots + a_k x_k + c$

Atoms $\quad atom ::= x > s \mid x = t \mid \ x \ < \ t$

Formulas $\quad \varphi ::= \varphi \wedge \varphi \mid \varphi \vee \varphi \mid atom$

$$\exists x \,.\, \varphi\big[x < t_i, x > s_j, x = t_k\big] \equiv$$

$$\varphi\big[\infty < t_i, \infty > s_j, \infty = t_k\big] \ \vee \qquad x \ is \ large$$

$$\bigvee_k \varphi\big[t_k < t_i, t_k > s_j, t_k = t_{k'}\big] \vee \qquad x \ is \ t_k$$

$$\bigvee_i \varphi\big[t_i - \epsilon < t_{i'}, t_i - \epsilon > s_j, t_i - \epsilon = t_k\big] \qquad lub. of \ x \ is \ t_i$$

Loos-Weispfenning $\Rightarrow$ Abstract QE(LRA)

Terms $\quad t, s \ ::= a_1 x_1 + a_2 x_2 + \ \ldots + a_k x_k + c$

Atoms $\quad atom ::= x > s \mid x = t \mid \ x \ < \ t$

Formulas $\quad \varphi ::= \varphi \wedge \varphi \mid \varphi \vee \varphi \mid atom$

$$\exists x \,.\, \varphi\big[x < t_i, x > s_j, x = t_k\big] \equiv$$

$$\varphi\big[false, \infty > s_j, \infty = t_k\big] \ \vee \qquad x \ is \ large$$

$$\bigvee_k \varphi\big[t_k < t_i, t_k > s_j, t_k = t_{k'}\big] \vee \qquad x \ is \ t_k$$

$$\bigvee_i \varphi\big[t_i - \epsilon < t_{i'}, t_i - \epsilon > s_j, t_i - \epsilon = t_k\big] \qquad lub. of \ x \ is \ t_i$$

Loos-Weispfenning $\Rightarrow$ Abstract QE(LRA)

Terms $\quad t, s \ ::= a_1 x_1 + a_2 x_2 + \ \ldots + a_k x_k + c$

Atoms $\quad atom ::= x > s \mid x = t \mid \ x \ < \ t$

Formulas $\quad \varphi ::= \varphi \wedge \varphi \mid \varphi \vee \varphi \mid atom$

$$\exists x \,.\, \varphi\big[x < t_i, x > s_j, x = t_k\big] \equiv$$

$$\varphi[false, true, \infty = t_k] \ \vee \qquad x \ is \ large$$

$$\bigvee_k \varphi\big[t_k < t_i, t_k > s_j, t_k = t_{k'}\big] \vee \qquad x \ is \ t_k$$

$$\bigvee_i \varphi\big[t_i - \epsilon < t_{i'}, t_i - \epsilon > s_j, t_i - \epsilon = t_k\big] \qquad lub. of \ x \ is \ t_i$$

Loos-Weispfenning $\Rightarrow$ Abstract QE(LRA)

Terms $\quad t, s \ ::= a_1 x_1 + a_2 x_2 + \ \ldots + a_k x_k + c$

Atoms $\quad atom ::= x > s \mid x = t \mid \ x \ < \ t$

Formulas $\quad \varphi ::= \varphi \wedge \varphi \mid \varphi \vee \varphi \mid atom$

$$\exists x \,.\, \varphi\big[x < t_i, x > s_j, x = t_k\big] \equiv$$

$$\varphi[false, true, false] \ \vee \qquad x \ is \ large$$

$$\bigvee_k \varphi\big[t_k < t_i, t_k > s_j, t_k = t_{k'}\big] \vee \qquad x \ is \ t_k$$

$$\bigvee_i \varphi\big[t_i - \epsilon < t_{i'}, t_i - \epsilon > s_j, t_i - \epsilon = t_k\big] \qquad lub. of \ x \ is \ t_i$$

Loos-Weispfenning $\Rightarrow$ Abstract QE(LRA)

Terms $t, s \ ::= a_1 x_1 + a_2 x_2 + \ \ldots + a_k x_k + c$

Atoms $atom ::= x > s \mid x = t \mid \ x \ < \ t$

Formulas $\varphi ::= \varphi \wedge \varphi \mid \varphi \vee \varphi \mid atom$

$$\exists x \,.\, \varphi\big[x < t_i, x > s_j, x = t_k\big] \equiv$$

$$\varphi[false, true, false] \ \vee \qquad x \ is \ large$$

$$\bigvee_k \varphi\big[t_k < t_i, t_k > s_j, t_k = t_{k'}\big] \vee \qquad x \ is \ t_k$$

$$\bigvee_i \varphi\big[t_i \leq t_{i'}, t_i - \epsilon > s_j, t_i - \epsilon = t_k\big] \qquad lub. of \ x \ is \ t_i$$

Loos-Weispfenning $\Rightarrow$ Abstract QE(LRA)

Terms $t, s ::= a_1 x_1 + a_2 x_2 + \ldots + a_k x_k + c$

Atoms $atom ::= x > s \mid x = t \mid x < t$

Formulas $\varphi ::= \varphi \wedge \varphi \mid \varphi \vee \varphi \mid atom$

$$\exists x \,.\, \varphi\big[x < t_i, x > s_j, x = t_k\big] \equiv$$

$$\varphi[false, true, false] \;\vee \qquad x \text{ is large}$$

$$\bigvee_k \varphi\big[t_k < t_i, t_k > s_j, t_k = t_{k'}\big] \vee \qquad x \text{ is } t_k$$

$$\bigvee_i \varphi\big[t_i \leq t_{i'}, t_i > s_j, t_i - \epsilon = t_k\big] \qquad lub. of \; x \text{ is } t_i$$

Loos-Weispfenning $\Rightarrow$ Abstract QE(LRA)

Terms $t, s \ ::= a_1 x_1 + a_2 x_2 + \ \ldots + a_k x_k + c$

Atoms $atom ::= x > s \mid x = t \mid \ x \ < \ t$

Formulas $\varphi ::= \varphi \wedge \varphi \mid \varphi \vee \varphi \mid atom$

$$\exists x \,.\, \varphi\big[x < t_i, x > s_j, x = t_k\big] \equiv$$
$$\varphi[false, true, false] \ \vee$$
$$\bigvee_k \varphi\big[t_k < t_i, t_k > s_j, t_k = t_{k'}\big] \vee$$
$$\bigvee_i \varphi\big[t_i \leq t_{i'}, t_i > s_j, false\big]$$

Loos-Weispfenning $\Rightarrow$ Abstract QE(LRA)

$$\exists x\,.\,\varphi\big[x < t_i, x > s_j, x = t_k\big] \equiv$$

$$\varphi\big[\infty < t_i, \infty > s_j, \infty = t_k\big] \;\vee$$

$$\bigwedge_i \neg(x < t_i) \wedge \bigwedge_k \neg(x = t_k) \wedge \bigwedge_j x > s_j$$

$$\bigvee_k \varphi\big[t_k < t_i, t_k > s_j, t_k = t_{k'}\big] \vee$$

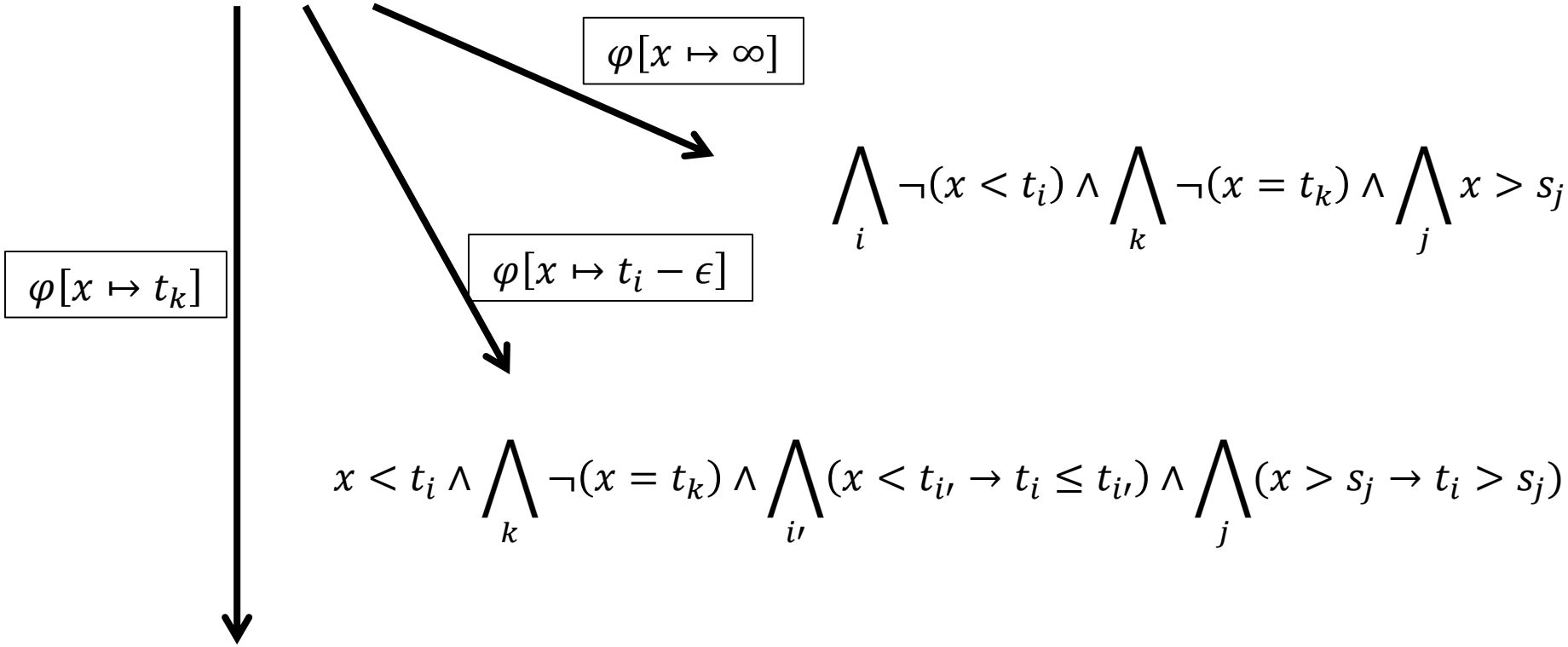
$$x = t_k \wedge \bigwedge_i (x < t_i \rightarrow t_k < t_i) \wedge \bigwedge_j (x > s_j \rightarrow t_k > s_j)$$

$$\bigvee_i \varphi\big[t_i - \epsilon < t_{i'}, t_i - \epsilon > s_j, t_i - \epsilon = t_k\big]$$

$$x < t_i \wedge \bigwedge_k \neg(x = t_k) \wedge \bigwedge_{i'} (x < t_{i'} \rightarrow t_i \leq t_{i'}) \wedge \bigwedge_j (x > s_j \rightarrow t_i > s_j)$$

Loos-Weispfenning $\Rightarrow$ Abstract QE(LRA)

$\exists x \,.\, \varphi[x < t_i, x > s_j, x = t_k]$

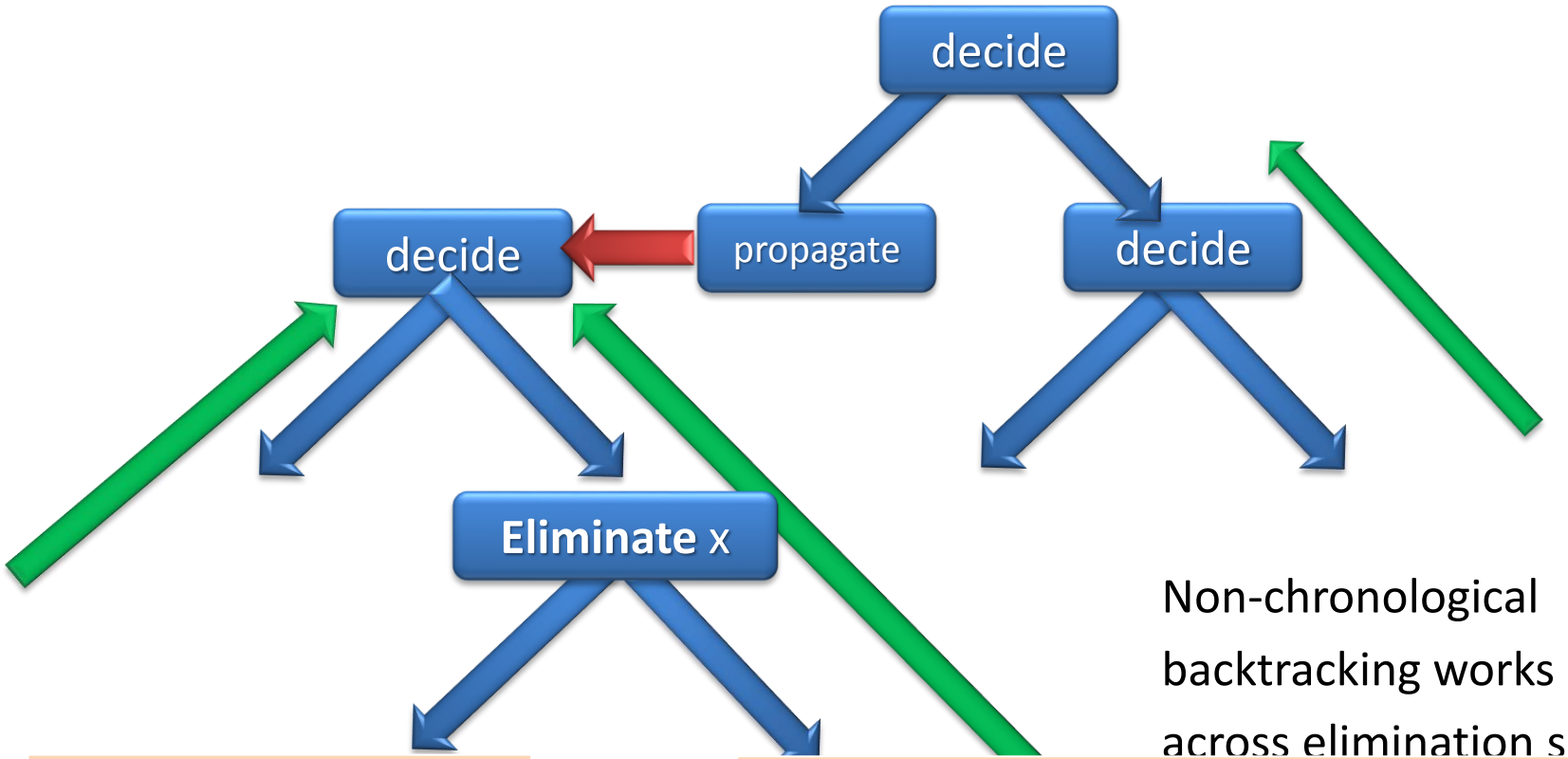


$$\bigwedge_i \neg(x < t_i) \wedge \bigwedge_k \neg(x = t_k) \wedge \bigwedge_j x > s_j$$

$$x < t_i \wedge \bigwedge_k \neg(x = t_k) \wedge \bigwedge_{i\prime}(x < t_{i\prime} \rightarrow t_i \leq t_{i\prime}) \wedge \bigwedge_j (x > s_j \rightarrow t_i > s_j)$$

$$x = t_k \wedge \bigwedge_i (x < t_i \rightarrow t_k < t_i) \wedge \bigwedge_j (x > s_j \rightarrow t_k > s_j)$$

The Abstract Decision Procedure



Non-chronological backtracking works across elimination splits

$$\bigwedge_i \neg(x < t_i) \wedge \bigwedge_k \neg(x = t_k) \wedge \bigwedge_j x > s_j$$

$$x < t_i \wedge \bigwedge_k \neg(x = t_k) \wedge \bigwedge_{i'} (x < t_{i'} \rightarrow t_i \leq t_{i'}) \wedge \bigwedge_j (x > s_j \rightarrow t_i > s_j)$$

$$\varphi[x \mapsto t_i - \epsilon] \vee \varphi[x \mapsto \infty]$$

Cooper+Ω $\Rightarrow$ Abstract QE(LIA)

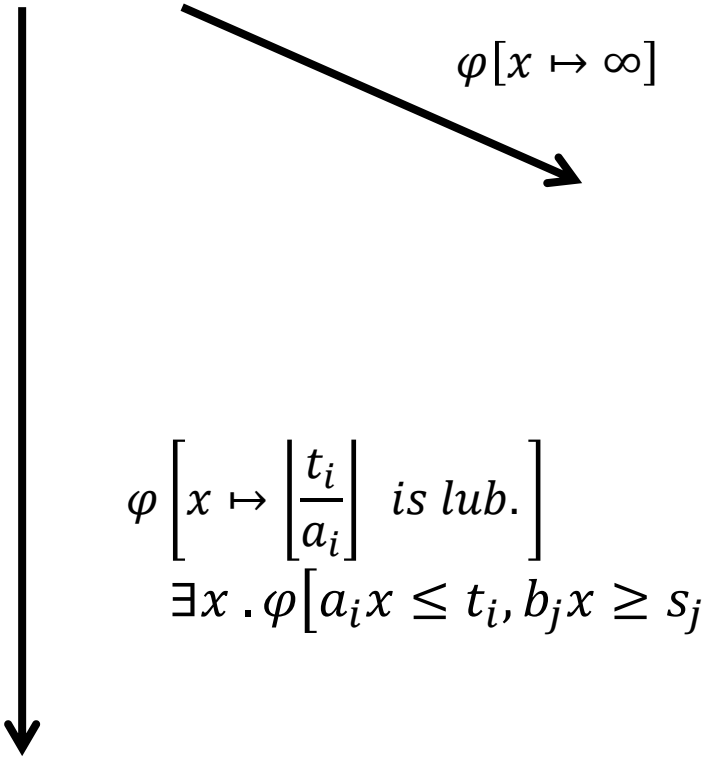
Terms $t, s ::= a_1 x_1 + a_2 x_2 + \ldots + a_k x_k + c$

Atoms $atom ::= ax \leq t \mid bx \geq s \mid \pm (c \uparrow ax + t)$

Formulas $\varphi ::= \varphi \wedge \varphi \mid \varphi \vee \varphi \mid atom$

$$\exists x \,.\, \varphi\left[a_i x \leq t_i, b_j x \geq s_j, \pm(c_k \uparrow a_k x + t_k)\right]$$

Cooper+Ω $\Rightarrow$ Abstract QE(LIA)



$\varphi[x \mapsto \infty]$

$$\bigwedge_i \neg(ax \leq t_i) \wedge \bigwedge_j (bx \geq t_j)$$

$$\varphi\left[x \mapsto \left\lfloor \frac{t_i}{a_i} \right\rfloor \ \ is\ lub.\right]$$

$$\exists x\,.\,\varphi\left[a_i x \leq t_i, b_j x \geq s_j, \pm(c_k \uparrow a_k x + t_k)\right]$$

$$a_i x \leq t_i \wedge \bigwedge_{i'} (a_{i'} x \leq t_{i'} \rightarrow a_{i'} t_i \leq a_i t_{i'}) \wedge \bigwedge_j (b_j x \geq s_j \rightarrow resolve(a_i x \leq t_i, b_j x \geq s_j))$$

Cooper+Ω $\Rightarrow$ Abstract QE(LIA)

Resolving integer inequalities:

$$
\begin{aligned}
&resolve(ax \leq t, bx \geq s) = \\
&\quad as + (a-1)(b-1) \leq bt \vee \\
&\quad a \geq b \wedge as \leq bt \wedge (\exists z \in [0 \ldots b-2])(b|(s+z) \wedge a(s+z) \leq bt) \vee \\
&\quad b > a \wedge as \leq bt \wedge (\exists z \in [0 \ldots a-2])(a|(t-z) \wedge as \leq b(t-z))
\end{aligned}
$$

$$(\exists x.\, ax \leq t \wedge bx \geq s) \quad \equiv \quad resolve(ax \leq t, bx \geq s)$$

n x *m*-ary version in [Pugh 92]

Cooper+Ω $\Rightarrow$ Abstract QE(LIA)

Eliminating divisibility

$$\exists x\,.\,\varphi\left[a_i x \le t_i, b_j x \ge s_j, \pm(\boldsymbol{c_k}\uparrow \boldsymbol{a_k x} + \boldsymbol{t_k})\right] \equiv$$

$$\delta = lcm(c_k) - 1$$

$$\exists x\,.\,\exists u \in \{0..\delta-1\}.\begin{pmatrix} \delta \uparrow (x-u) \wedge \\ \left((c_k \uparrow a_k x + t_k) \leftrightarrow (c_k \uparrow a_k u + t_k)\right) \wedge \\ \varphi\left[a_i x \le t_i, b_j x \ge s_j, \pm(c_k\uparrow a_k u + t_k)\right] \end{pmatrix} \equiv$$

$$x \mapsto x\delta + u$$

$$\exists x\,.\,\exists u \in \{0..\delta-1\}.\begin{pmatrix} \left((c_k \uparrow a_k(\delta x+u) + t_k) \leftrightarrow (c_k \uparrow a_k u + t_k)\right) \wedge \\ \varphi[a_i(\delta x+u) \le t_i, b_j(\delta x+u) \ge s_j, \pm(c_k\uparrow a_k u + t_k)] \end{pmatrix} \equiv$$

$$\exists x\,.\,\exists u \in \{0..\delta-1\}.\varphi\left[a_i(\delta x+u) \le t_i, b_j(\delta x+u) \ge s_j, \pm(c_k\uparrow a_k u + t_k)\right]$$

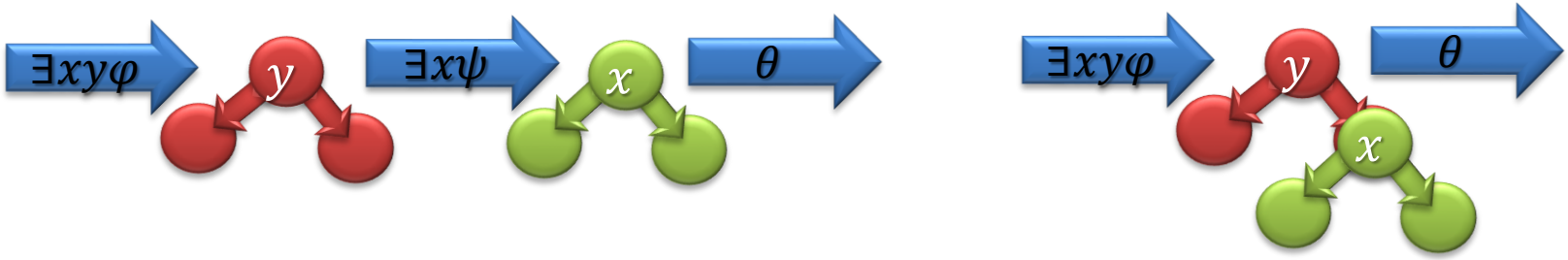
Practicalities

Use LA solvers to prune search early

- Efficient LA solvers eliminate infeasible cases
- Identify satisfiable pure formulas

Linear Diophantine Equation solving, e.g., [Pugh 92]

Elimination Order: Sequential vs. Parallel



Handling finite range arithmetic efficiently

- In context of Z3:
  Reduce finite range arithmetic to bit-vector theory

QDT: Quantfied Data-types

Eliminate quantifiers for Data-types using virtual substitution style approach

Other algorithms: Maher, Hodges

Convention: $C(\ldots) - constructor$: $cons(x, l)$

$acc_j(..) - accessors$: $head(l)$

$IsC(\ldots) - recognizer$: $cons?\,(l)$

Non-recursive case

Non-recursive data-types are particularly easy.
Quantifier elimination can simply expand the finite set of cases corresponding to the set of possible constructors.

$$\exists\, x \;.\; \varphi[x] \;\equiv \bigvee_i \exists\, \vec{y} \;.\varphi[C_i(\vec{y})]$$

Recursive case

The recursive case requires essentially three steps. The second step is only applied if the formula contains an accessor term over the variable to be eliminated.

Fix recognizer

$$\exists\, x \;.\; \varphi[x] \equiv \bigvee_i \exists\, x \;.\; IsC_i(x) \wedge \;\; \varphi[x]$$

Eliminate accessors

$$\exists\, x\; IsC(x) \wedge \varphi[acc_j(x), x] \quad \equiv \exists\, \vec{y} \;.\; \varphi[y_j, C(\vec{y})]$$

$\exists\, x.\ IsC(x) \wedge\ \varphi[x]$ x is not under accessor

Solve for *x* using unification

x in φ as: $(\neg)t[x] \simeq y,\ (\neg)x \simeq t\ (x \notin t)$

Eliminate *x*: $\varphi\left[x \simeq t_i, \neg(x \simeq t_j)\right]$:

$$\varphi[false, true] \vee \bigvee_i Is(x) \wedge \varphi \wedge Unifier_i$$

Computing *Unifier*

Solve for *x:*

$$\frac{C(t_1, x) = t_2 \wedge \psi}{x = accC_2(t_2) \ \wedge IsC(t_2) \wedge \psi}$$

Solve equalities After the recognizer has been fixed, and x is no longer in scope of an accessor, we further simplify $R_C(x) \wedge \varphi$ using the equivalences:

$$\begin{aligned}
R_C(x) \wedge \varphi[R_C(x)] &\equiv R_C(x) \wedge \varphi[\mathit{true}] \\
R_C(x) \wedge \varphi[R_{C'}(x)] &\equiv R_C(x) \wedge \varphi[\mathit{false}] \quad \text{for } C \neq C' \\
\varphi[C(t[x], t'[x]) \simeq C(s[x], s'[x])] &\equiv \varphi[t[x] \simeq s[x] \wedge t'[x] \simeq s'[x]] \\
\varphi[x \simeq x] &\equiv \varphi[\mathit{true}] \\
\varphi[x \simeq C(..x..)] &\equiv \varphi[\mathit{false}]
\end{aligned}$$

So the equalities involving x are of one of the forms $t[x] \simeq y$, where y is a variable different from x, or $x \simeq t$, where $x \notin t$.

We can now eliminate x:

$$\exists x \,.\, R_C(x) \wedge \varphi[t_i[x] \simeq t'_i, s_j[x] \not\simeq s'_j] \equiv \varphi[\mathit{false}, \mathit{true}] \vee \bigvee_i (R_C(x) \wedge \psi_i \wedge \varphi)[u_i/x] \quad (5)$$

where

$$\begin{aligned}
u_i, \psi_i &= \mathit{solve}(x, t_i[x], t'_i, \mathit{true}) \\
\mathit{solve}(x, x, t, \psi) &= t, \psi \\
\mathit{solve}(x, C(.., t_k[x], ..), t, \psi) &= \mathit{solve}(x, t_k[x], \mathit{acc}_k(t), R_C(t) \wedge \psi)
\end{aligned}$$

The justification for elimination is as follows: Given an interpretation I, a positive equality $t_i[x] \simeq t'_i$ is true under I then let $t, \varphi_i = \mathit{solve}(x, t_i[x], t'_i, \mathit{true})$, and I satisfies $\varphi_i \wedge x \simeq t$. In the case every equality is false under I, consider the interpretation I' obtained from I by modifying x to the diagonal of $s_j[s'_j]$ (a term that contains as subterms all $s_j[s'_j]$). In this case if $I \models \varphi[\mathit{false}, s_j[x] \not\simeq s'_j]$, then also $I' \models \varphi[\mathit{false}, s_j[x] \not\simeq s'_j]$. Since I and I' only differ in the interpretation of x we have $I \models \exists x \,.\, \varphi$ if and only if $I' \models \exists x \,.\, \varphi$.

From QE to QSMT

- If you just want to check satsifiability, then Quantifier Elimination is an over-fit.

- Research proposition: take inspiration from QBF solving for QAF/QDF solving

SOLVING HORN CLAUSES

Claim

Symbolic Model Checking = Solving Horn Clauses

Motivation: Recursive Procedures

mc(x) = x-10 **if x > 100**

mc(x) = mc(mc(x+11)) **if x ≤ 100**

assert ($x \leq 101 \rightarrow$mc(x) = 91)

Motivation: Recursive Procedures

Formulate as Horn clauses:

$$\forall X.\ X > 100 \rightarrow \mathrm{mc}(X, X - 10)$$

$$\forall X, Y, R.\ X \leq 100 \wedge \mathrm{mc}(X + 11, Y) \wedge \mathrm{mc}(Y, R) \rightarrow \mathrm{mc}(X, R)$$

$$\forall X, R.\ \mathrm{mc}(X, R) \wedge x \leq 101 \rightarrow \mathrm{R} = 91$$

Solve for mc

Motivation: Recursive Procedures

Formulate as Predicate Transformer:

$$\mathscr{F}(\textbf{mc})(X,R) = \begin{array}{c} X > 100 \wedge R = X - 10 \\ \vee\ X \leq 100 \wedge \exists Y.\, \textbf{mc}(X + 11, Y) \wedge \textbf{mc}(Y, R) \end{array}$$

Check: $\mu\mathscr{F}(\textbf{mc})(X,R) \wedge \boldsymbol{x \leq 101 \rightarrow R = 91}$

Motivation: Recursive Procedures

Instead of computing $\mu\mathscr{F}(\mathbf{mc})(X,R)$,

then checking $\mu\mathscr{F}(\mathbf{mc})(X,R) \wedge x \leq 101 \rightarrow \mathbf{mc}(x) = 91$

Suffices to find post-fixed point $\mathbf{mc}_{post}$ **satisfying:**

$$\forall X, R.\quad \mathscr{F}(\mathbf{mc}_{post})(X,R) \rightarrow \mathbf{mc}_{post}(X,R)$$

$$\forall X, R.\quad \mathbf{mc}_{post}(X,R) \wedge x \leq 101 \rightarrow \mathbf{mc}_{post}(x) = 91$$

Program Verification as SMT

[Bjørner, McMillan, Rybalchenko, SMT workshop 2012]
Hilbert Sausage Factory: [Grebenshchikov, Lopes, Popeea, Rybalchenko, PLDI 2012]

Program Verification (Safety)

as ***Solving fixed-points***

as ***Satisfiability of Horn clauses***

IC3/PDR: Property Directed Reachability

The IC3 Algorithm for Symbolic Model Checking by Aaron Bradley

Procedures	*Regular vs. Push Down systems* As a Conflict-driven solver for recursive Horn clauses
Beyond Propositional Logic	*Linear Real Arithmetic* - *Timed Automata Decision Procedure* - *Interpolants from models*

[SAT 2012. Kryštof Hoder & Nikolaj Bjørner]

PDR – the algorithm

Objective is to solve for R such that

$$\mathscr{F}(R)(X) \rightarrow R(X), \quad R(X) \rightarrow Safe(X), \ \forall X$$

Key elements of PDR algorithm:

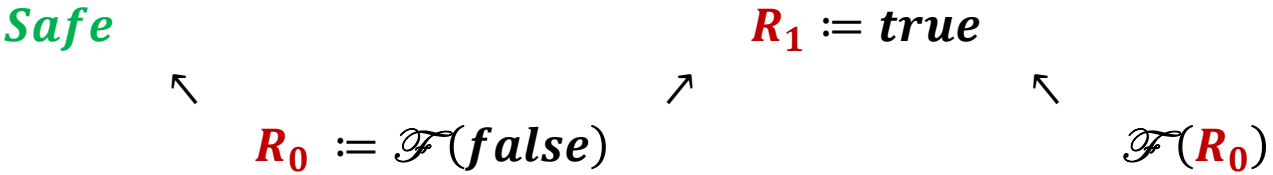
- **Over-approximate reachable states**
  $R_0 := \mathscr{F}(\text{false}), R_1 \rightarrow R_2 \rightarrow \cdots \rightarrow R_N := \text{true}$
- **Propagate back from** $\neg Safe$
- **Resolve conflicts**
- **Strengthen/propagate using induction**

PDR – the algorithm

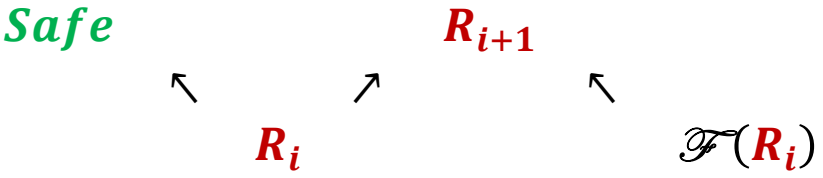
Objective is to solve for R such that

$$\mathscr{F}(R)(X) \rightarrow R(X), \quad R(X) \rightarrow Safe(X), \ \forall X$$

Initialize:

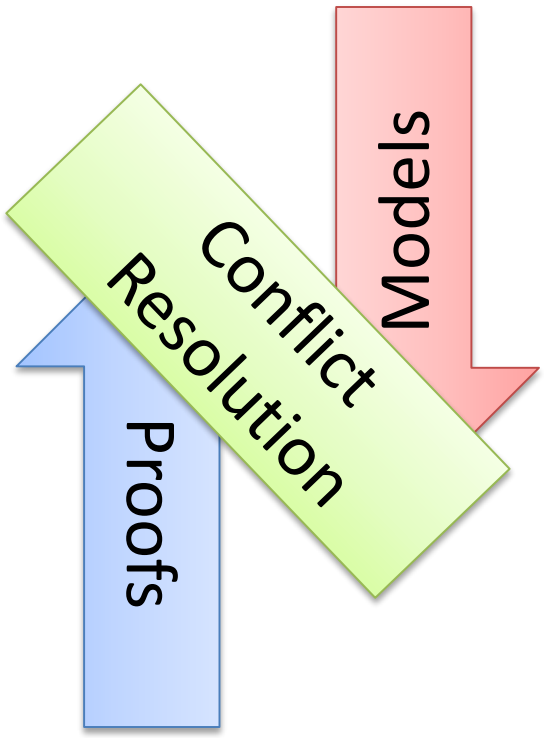


Main invariant:



A digression

Dualities – Recurring Theme



Core DPLL(T) engine

Fixed Points engine

Nonlinear solver

Linear Integer solver

Core Engine in Z3: Modern DPLL/CDCL

Initialize	$\epsilon \mid F$	F *is a set of clauses*
Decide	$M \mid F \Longrightarrow M, \ell \mid F$	ℓ *is unassigned*
Propagate	$M \mid F, C \vee \ell \Longrightarrow M, \ell^{C \vee \ell} \mid F, C \vee \ell$	C *is false under* M
Sat	$M \mid F \Longrightarrow M$	F *true under* M
Conflict	$M \mid F, C \Longrightarrow M \mid F, C \mid C$	C *is false under* M
Learn	$M \mid F \mid C \Longrightarrow M \mid F, C \mid C$	
Unsat	$M \mid F \mid \emptyset \Longrightarrow Unsat$	
Backjump	$MM' \mid F \mid C \vee \ell \Longrightarrow M\ell^{C \vee \ell} \mid F$	$\neg\ell \in M', \quad M' \cap \neg C = \emptyset$
Resolve	$M \mid F \mid C' \vee \neg\ell \Longrightarrow M \mid F \mid C' \vee C$	$\ell^{C \vee \ell} \in M$
Restart	$M \mid F \Longrightarrow \epsilon \mid F$	
Forget	$M \mid F, C \Longrightarrow M \mid F$	C *is a learned clause*

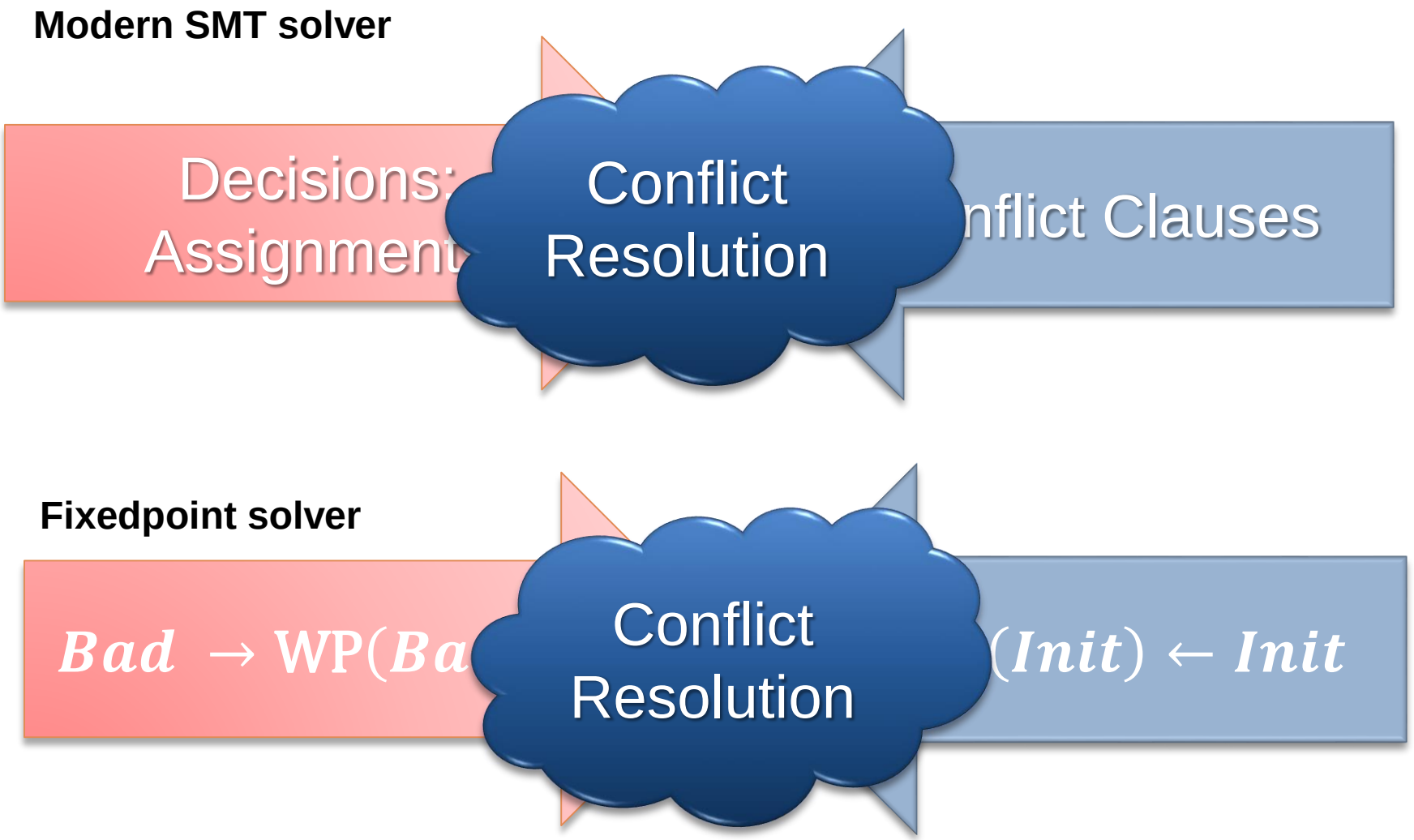
Model

Proof

Conflict Resolution

[Nieuwenhuis, Oliveras, Tinelli J.ACM 06] customized

Search: Mile-high perspective



Conflict resolution with arithmetic

$$\textbf{initially } y_1 := y_2 := 0;$$
$$P_1 :: \left[\begin{array}{l}\textbf{loop forever do}\\ \left[\begin{array}{l}\ell_0 : y_1 := y_2 + 1;\\ \ell_1 : \textbf{await } y_2 = 0 \vee y_1 \leq y_2;\\ \ell_2 : \textbf{critical};\\ \ell_3 : y_1 := 0;\end{array}\right]\end{array}\right] \;||\; P_2 :: \left[\begin{array}{l}\textbf{loop forever do}\\ \left[\begin{array}{l}\ell_0 : y_2 := y_1 + 1;\\ \ell_1 : \textbf{await } y_1 = 0 \vee y_2 \leq y_1;\\ \ell_2 : \textbf{critical};\\ \ell_3 : y_2 := 0;\end{array}\right]\end{array}\right]$$

R(0,0,0,0). *Initial states*

T(L,M,Y1,Y2,L',M',Y1',Y2')∧R(L,M,Y1,Y2) → R(L',M',Y1',Y2') *Reachable states*

R(2,2,Y1,Y2) → false *Is unsafe state reachable?*

Conflict resolution with arithmetic

$$\text{initially } y_1 := y_2 := 0;$$

$$P_1 :: \left[\begin{array}{l}\text{loop forever do}\\ \left[\begin{array}{l}\ell_0 : y_1 := y_2 + 1;\\ \ell_1 : \text{await } y_2 = 0 \vee y_1 \leq y_2;\\ \ell_2 : \text{critical};\\ \ell_3 : y_1 := 0;\end{array}\right]\end{array}\right] \;||\; P_2 :: \left[\begin{array}{l}\text{loop forever do}\\ \left[\begin{array}{l}\ell_0 : y_2 := y_1 + 1;\\ \ell_1 : \text{await } y_1 = 0 \vee y_2 \leq y_1;\\ \ell_2 : \text{critical};\\ \ell_3 : y_2 := 0;\end{array}\right]\end{array}\right]$$

R(0,0,0,0). *Initial states*

T(L,M,Y1,Y2,L',M',Y1',Y2')∧R(L,M,Y1,Y2) → R(L',M',Y1',Y2') *Reachable states*

R(2,2,Y1,Y2) → false *Is unsafe state reachable?*

Step(L,L',Y1,Y2,Y1') → T(L,M,Y1,Y2,L',M,Y1',Y2) *P_1 takes a step*

Step(M,M',Y2,Y1,Y2') → T(L,M,Y1,Y2,L,M',Y1,Y2') *P_2 takes a step*

Step(0,1,Y1,Y2,Y2+1). $\ell_0 : y := \hat{y} + 1; \; goto \; \ell_1$

(Y1 ≤ Y2 ∨ Y2 = 0) → Step(1,2,Y1,Y2,Y1). $\ell_1 : await \; \hat{y} = 0 \vee y \leq \hat{y} \; ; goto \; \ell_2$

Step(2,3,Y1,Y2,Y1). $\ell_2 : critical \; ; goto \; \ell_3$

Step(3,0,Y1,Y2,0). $\ell_3 : y := 0; goto \; \ell_0$

Conflict resolution with arithmetic

$$\text{initially } y_1 := y_2 := 0;$$
$$P_1 :: \left[\begin{array}{l}\text{loop forever do}\\ \left[\begin{array}{l}\ell_0 : y_1 := y_2 + 1;\\ \ell_1 : \text{await } y_2 = 0 \vee y_1 \leq y_2;\\ \ell_2 : \text{critical};\\ \ell_3 : y_1 := 0;\end{array}\right]\end{array}\right] \;||\; P_2 :: \left[\begin{array}{l}\text{loop forever do}\\ \left[\begin{array}{l}\ell_0 : y_2 := y_1 + 1;\\ \ell_1 : \text{await } y_1 = 0 \vee y_2 \leq y_1;\\ \ell_2 : \text{critical};\\ \ell_3 : y_2 := 0;\end{array}\right]\end{array}\right]$$

R(0,0,0,0).

T(L,M,Y1,Y2,L',M',Y1',Y2')∧R(L,M,Y1,Y2) → R(L',M

R(2,2,Y1,Y2) → false

Step(L,L',Y1,Y2,Y1') → T(L,M,Y1,Y2,L',M,Y1',Y2)

Step(M,M',Y2,Y1,Y2') → T(L,M,Y1,Y2,L,M',Y1,Y2')

Step(0,1,Y1,Y2,Y2+1).

(Y1 ≤ Y2 ∨ Y2 = 0) → Step(1,2,Y1,Y2,Y1).

Step(2,3,Y1,Y2,Y1).

Step(3,0,Y1,Y2,0).

Mutual Exclusion

⇔

Clauses have model

P_2 takes a step

$\ell_0 : y := \hat{y} + 1;\ goto\ \ell_1$

$\ell_1 : await\ \hat{y} = 0 \vee y \leq \hat{y}\ ;\ goto\ \ell_2$

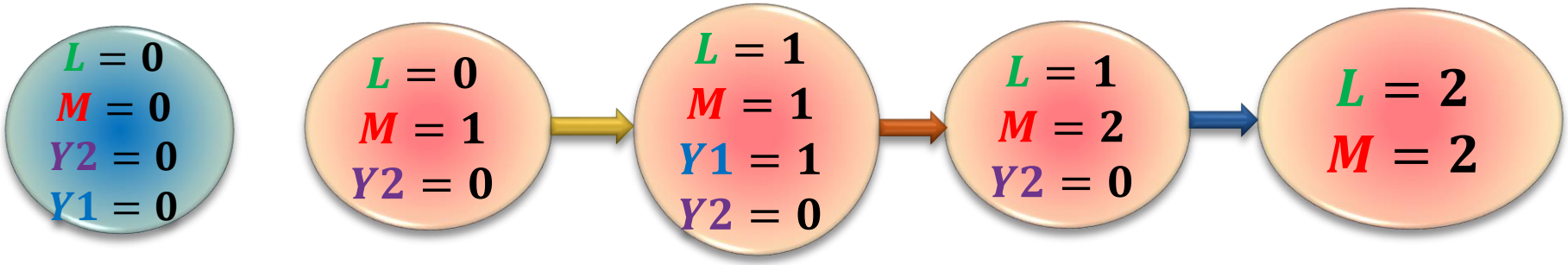
$\ell_2 : critical\ ;\ goto\ \ell_3$

$\ell_3 : y := 0;\ goto\ \ell_0$

PDR(T): Conflict Resolution

initially $y_1 := y_2 := 0$;

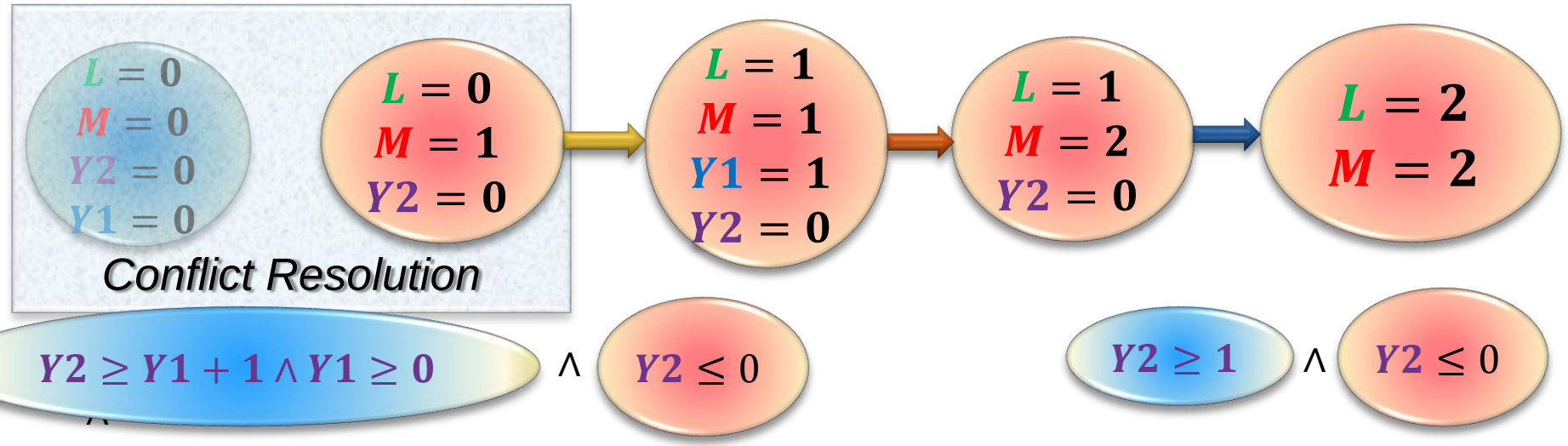
$$P_1 :: \left[\begin{array}{l}\text{loop forever do}\\ \left[\begin{array}{l}\ell_0 : y_1 := y_2 + 1;\\ \ell_1 : \text{await } y_2 = 0 \vee y_1 \leq y_2;\\ \ell_2 : \text{critical};\\ \ell_3 : y_1 := 0;\end{array}\right]\end{array}\right] \;||\; P_2 :: \left[\begin{array}{l}\text{loop forever do}\\ \left[\begin{array}{l}\ell_0 : y_2 := y_1 + 1;\\ \ell_1 : \text{await } y_1 = 0 \vee y_2 \leq y_1;\\ \ell_2 : \text{critical};\\ \ell_3 : y_2 := 0;\end{array}\right]\end{array}\right]$$



PDR(T): Conflict Resolution

initially $y_1 := y_2 := 0$;

$$P_1 :: \left[\begin{array}{l}\text{loop forever do}\\ \left[\begin{array}{l}\ell_0 : y_1 := y_2 + 1;\\ \ell_1 : \text{await } y_2 = 0 \vee y_1 \leq y_2;\\ \ell_2 : \text{critical};\\ \ell_3 : y_1 := 0;\end{array}\right]\end{array}\right] \;||\; P_2 :: \left[\begin{array}{l}\text{loop forever do}\\ \left[\begin{array}{l}\ell_0 : y_2 := y_1 + 1;\\ \ell_1 : \text{await } y_1 = 0 \vee y_2 \leq y_1;\\ \ell_2 : \text{critical};\\ \ell_3 : y_2 := 0;\end{array}\right]\end{array}\right]$$



Conflict

Resolution

Get Generalization from Farkas Lemma

Eg., resolve away blue internal variables

Side-effect: Horn Craig Interpolants

Suppose $A \Rightarrow B$

A Craig Interpolant is formula I:

$$Lang(I) \subseteq Lang(A) \cap Lang(B)$$

$$A \Rightarrow I, I \Rightarrow B$$

Horn version. Establish satisfiability of:

$$\forall x, y. A[x, y] \Rightarrow I(x),$$

$$\forall x, z. I(x) \Rightarrow B[x, z]$$

and find solution for I.

Side-effect: Horn Craig Interpolants

Suppose $A \Rightarrow B$

A Craig Interpolant is formula I:

$$Lang(I) \subseteq Lang(A) \cap Lang(B)$$

$$A \Rightarrow I, I \Rightarrow B$$

$\forall X.\quad X > 100 \rightarrow \mathbf{mc_0}(X, X-10)$

$\forall X, Y, R.\quad X \leq 100 \wedge \mathbf{mc_0}(X+11, Y) \wedge \mathbf{mc_0}(Y, R) \rightarrow \mathbf{mc_1}(X, R)$

$\forall X.\quad X > 100 \rightarrow \mathbf{mc_1}(X, X-10)$

$\forall X, Y, R.\quad X \leq 100 \wedge \mathbf{mc_1}(X+11, Y) \wedge \mathbf{mc_1}(Y, R) \rightarrow \mathbf{mc_2}(X, R)$

$\forall X.\quad X > 100 \rightarrow \mathbf{mc_2}(X, X-10)$

$\forall X, R.\quad \mathbf{mc_2}(X, R) \rightarrow R \geq 91$

Solve for $\mathbf{mc_0}, \mathbf{mc_1}, \mathbf{mc_2}$