ME(LIA) - Model Evolution With Linear Integer Arithmetic Constraints

Peter Baumgartner
NICTA, Canberra, Australia

Alexander Fuchs, Cesare Tinelli
University of Iowa, USA

Motivation

Proof problems in SW verification often require rich theories

- Background theory $\mathcal{T}$ = (Linear) integer arithmetic + Arrays + ...
- Free function and/or predicate symbols
- **Quantifiers**

A Q_AUFLIA proof problem [Ranise]

- Backgroud theory $\mathcal{T}$ = Linear integer arithmetic + Arrays
- Axiom:

$$\forall a, n \ \mathsf{symmetric}(a, n) \leftrightarrow (\forall i, j \ 1 \leq i, j \leq n \rightarrow \mathsf{select}(a, i, j) = \mathsf{select}(a, j, i))$$

- Proof task:

$$\{\mathsf{symmetric}(\mathsf{a}, \mathsf{n})\} \quad \mathsf{a}[0, 0] := \mathsf{e}_0 \,; \ldots ; \mathsf{a}[\mathrm{k}, \mathrm{k}] := \mathsf{e}_\mathrm{k} \quad \{\mathsf{symmetric}(\mathsf{a}, \mathsf{n})\}$$

Form of proof problem: $\forall\, \Phi \ \models_{\mathcal{T}} \ \forall\, \Psi$ (Φ, Ψ with free symbols)

The combination "Background theories + free symbols + quantifiers" makes it difficult

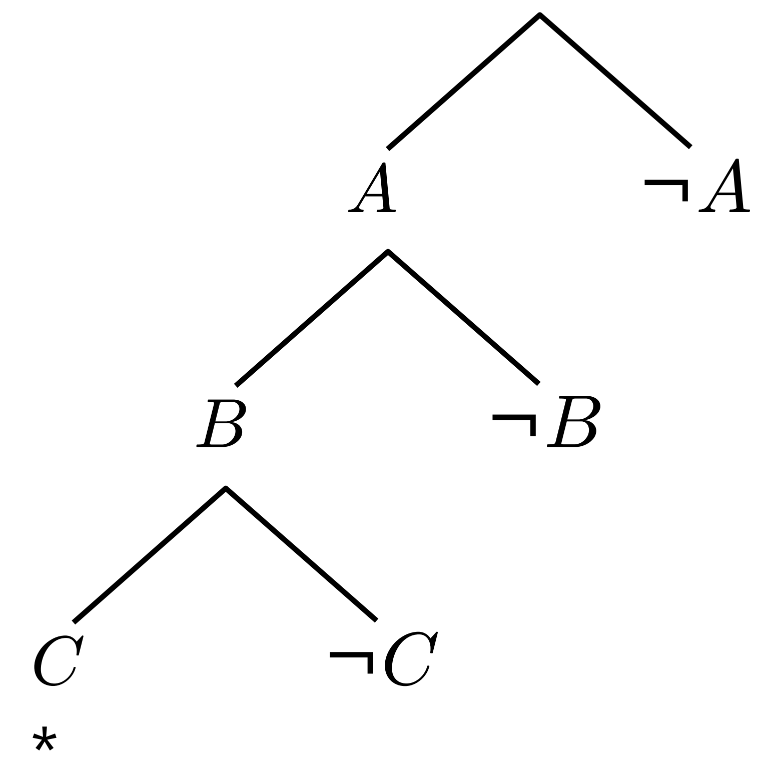
Approaches

- **First-order resolution theorem proving**
  - Support free symbols and quantifiers natively
  - Extensions for reasoning with background theories
    - Theory R [Stickel 85], Constraint R [Bürckert 90], Hierarchical Superposition [BGW 94], R+LIA [Korovin&Voronkov 07]
- **SMT solvers, in particular DPLL($\mathcal{T}$)**
  - Very successful for the quantifier free case, i.e. $\models_{\mathcal{T}} \forall\Phi$
  - Rely on instantiation heuristics for non-quantifier free case, $\forall\Psi \models_{\mathcal{T}} \forall\Phi$
- **ME(LIA)**
  - "DPLL(LIA) with quantifiers treated natively"
  - LIA constraints over $\mathbb{Z}$, free constants over finite domains, e.g. [1 .. 10]
  - Main result: sound and complete

DPLL procedure

Input: Propositional clause set

Output: Model or „unsatisfiable"



Algorithm components:

- Propositional semantic tree
  enumerates interpretations
- Propagation
- Split
- Backjumping

$$\{A, B\} \stackrel{?}{\models} \cancel{\neg A} \vee \cancel{\neg B} \vee C \vee D$$ ✗

$$\{A, B, C\} \stackrel{?}{\models} \cancel{\neg A} \vee \cancel{\neg B} \vee C \vee D$$ ✓

$$\{A, B, C\} \stackrel{?}{\models} \neg B \vee \neg C$$ ✗

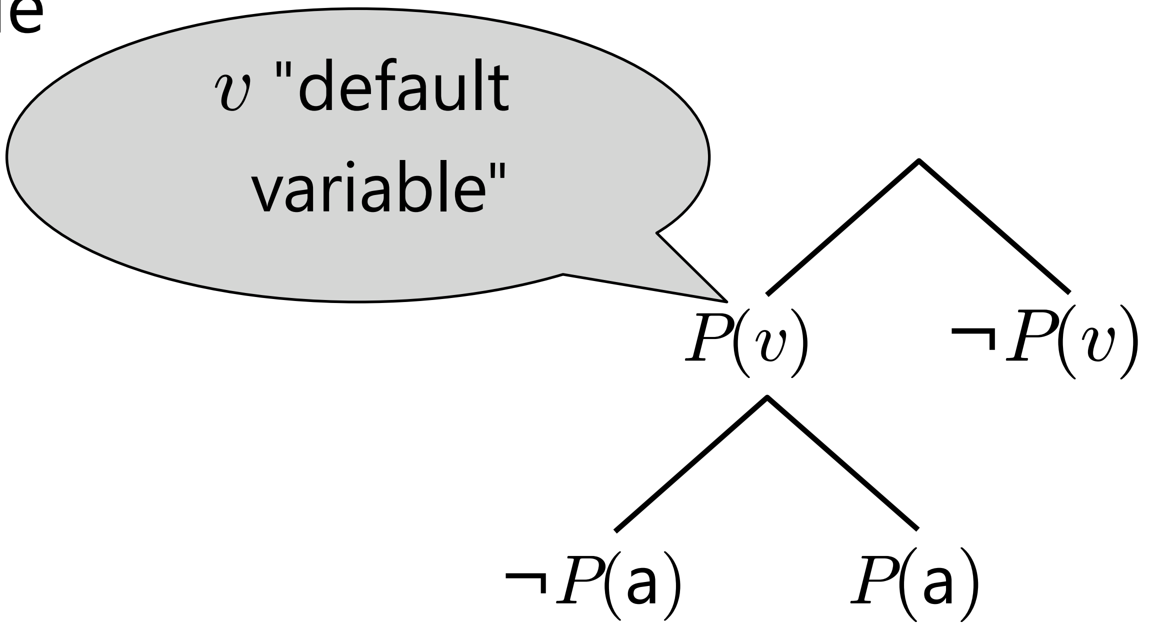
ME - lifting to first-order level

ME as First-Order DPLL

Input: First-order clause set

Output: Model or „unsatisfiable"

if termination



Algorithm components:

- First-order semantic tree

  enumerates interpretations
- Propagation
- Split
- Backjumping

$\{P(\mathsf{b}),$
$P(\mathsf{f}(\mathsf{a})),\ P(\mathsf{f}(\mathsf{b}), \ldots\}$

- A branch literal specifies a truth value for all its ground instances, unless there is a more specific literal specifying the opposite truth value
- ME's tries to compute a model of the input clause set represented this way

ME - Achievements so far

Plan: efficient theorem prover by integrating DPLL and FO techniques

Rationale: sufficient expressivity without compromising efficiency (BS logic)

- FDPLL [CADE-17]
  - Basic ideas, predecessor of ME
- ME Calculus [CADE-19, AIJ 2008]
  - Proper treatment of universal variables and unit propagation
  - Semantically justified redundancy criteria
- Finite model computation [JAL 2007]
- ME+Equality [CADE-20]
- ME+Lemmas [LPAR 2006]
- Darwin prover [JAIT 2006]
  http://combination.cs.uiowa.edu/Darwin/
  - CASC winner of EPR in 2006, 2007, second in 2008

Rest of This Talk - ME(LIA)

- Define the input language
- Generalize semantic trees
- Inference rules overview
- Discussion of calculus properties

Input Language

- Constraint clauses $C \leftarrow c$, where C is a "normalized" clause, e.g.

$$P(x_1, x_2) \vee \neg Q(x_2, x_3) \leftarrow \exists y\; 2 \leq y \wedge y < \mathsf{a} + x_1 \wedge x_2 = x_3$$

  where $P, Q, \ldots$ are free predicate symbols and a is a free constant

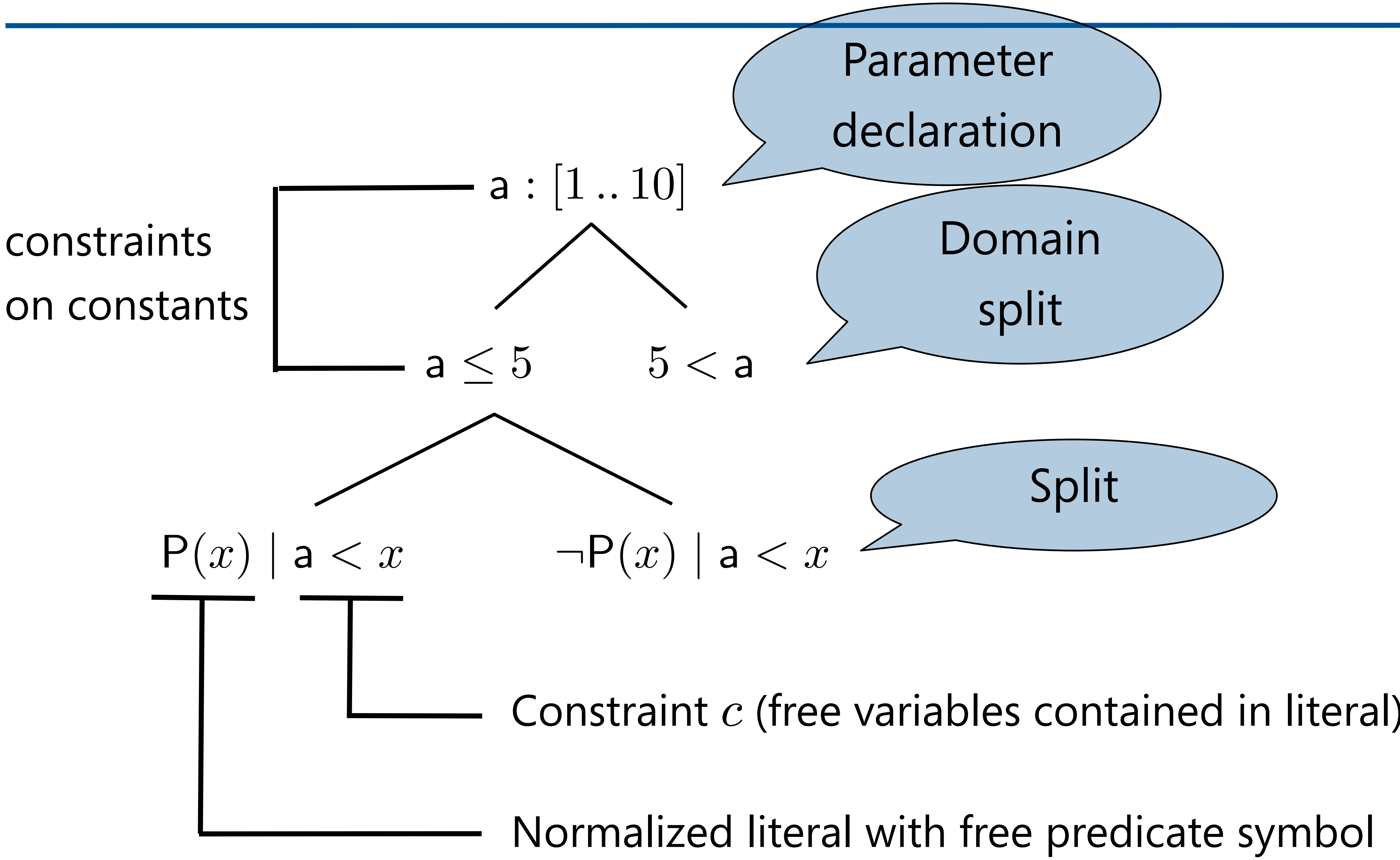
- Constraints c over $\mathbb{Z}$ generated by the syntax

$$\begin{array}{lcl} n & ::= & \text{integer constants } 0, \pm 1, \pm 2, \ldots \\ a & ::= & \text{free constants ("parameters") } \mathsf{a}, \mathsf{b}, \ldots \\ x & ::= & \text{variables } x, y, \ldots \\ t & ::= & n \mid a \mid x \mid t_1 + t_2 \mid t_1 - t_2 \\ l & ::= & \top \mid \bot \mid t_1 = t_2 \mid t_1 < t_2 \\ c & ::= & l \mid c_1 \wedge c_2 \mid \exists x\; c \end{array}$$

- Domain declaration a : $[n_1 \mathinner{..} n_2]$, for every input parameter a

- Constraint solutions must be bounded from below (add e.g. $-10 < x_1 \wedge 3 < x_2 \wedge 0 < x_3$ above)

Generalized Semantic Trees

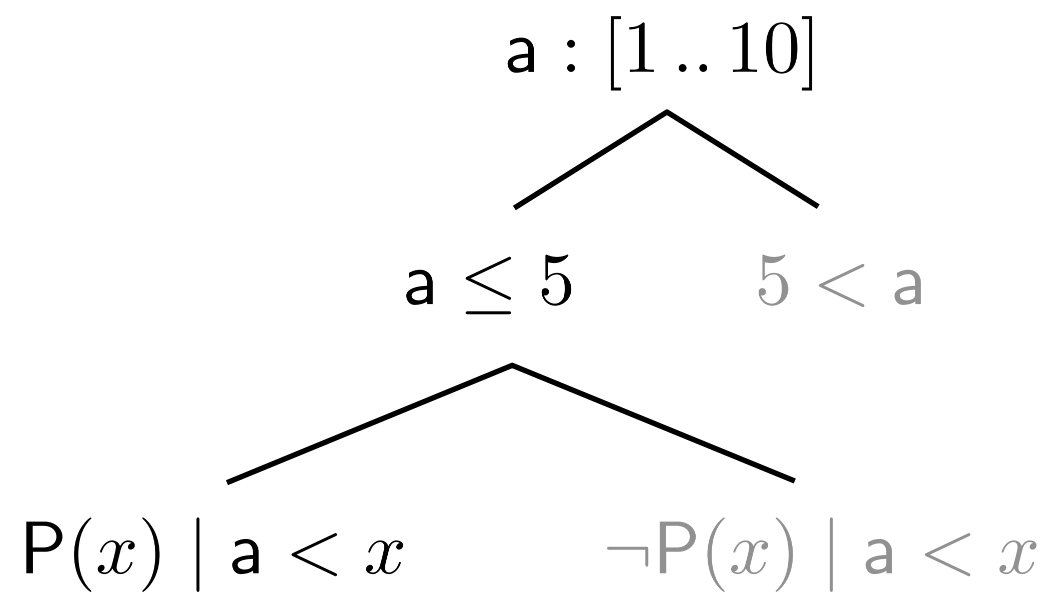


What is the meaning of a branch literal (model construction)?

Model Construction

... parametric in parameters, e.g:

$$\mathsf{a} = 4 : \quad \frac{I}{\begin{array}{c} P(5) \\ P(6) \\ P(7) \\ P(8) \\ \vdots \end{array}}$$



Idea:
For any assignment of constants consistent with the constraints,
a branch literal specifies a truth values for all its ground instances over $\mathbb{Z}$
that satisfy its constraint, unless ... (next slide)

Model Construction

$$\frac{I}{\begin{array}{r} P(5) \\ P(6) \\ \neg P(7) \\ \neg P(8) \\ \vdots \end{array}}$$

a $= 4$:

$\mathsf{a} : [1 \mathinner{..} 10]$

$\mathsf{a} \le 5$ $\quad$ $5 < \mathsf{a}$

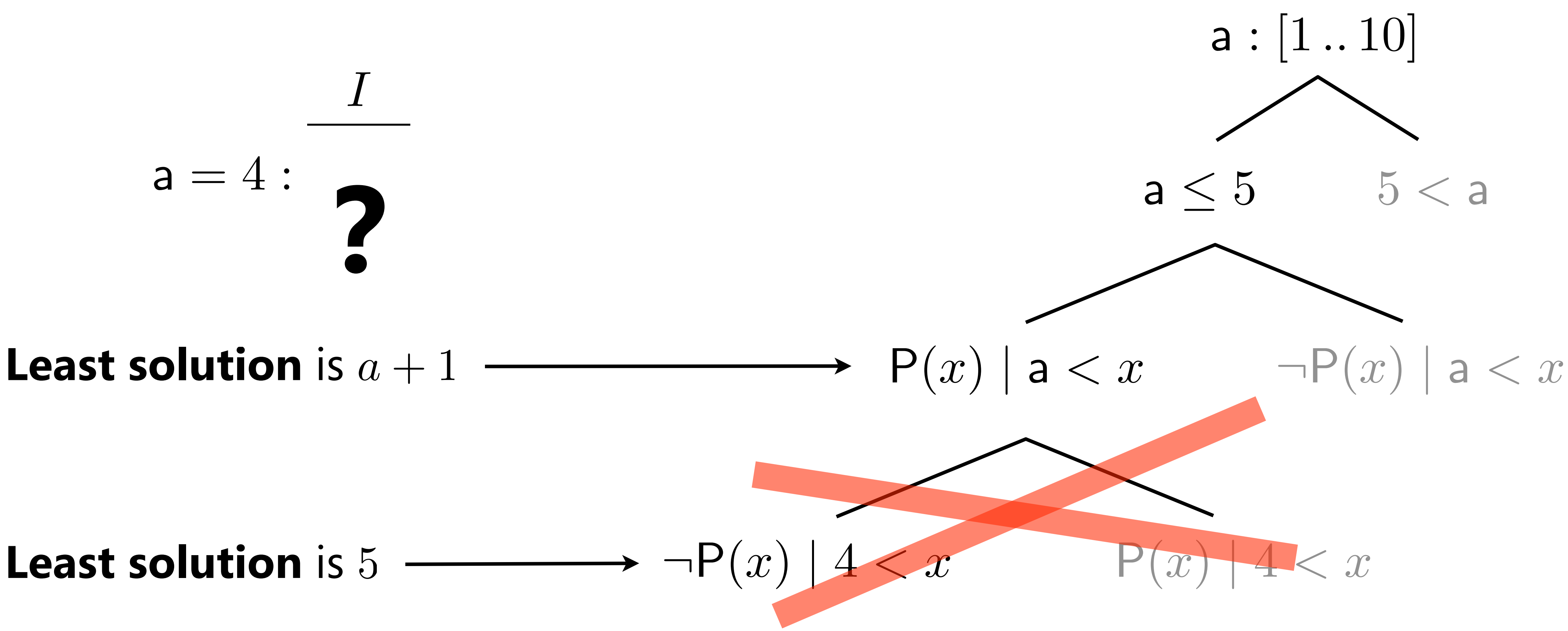
Least solution is $a + 1$ ⟶ $\mathsf{P}(x) \mid \mathsf{a} < x$ $\quad$ $\neg\mathsf{P}(x) \mid \mathsf{a} < x$

Least solution is $a + 3$ ⟶ $\neg\mathsf{P}(x) \mid \mathsf{a} + 2 < x$ $\quad$ $\mathsf{P}(x) \mid \mathsf{a} + 2 < x$

For any assignment of the constants consistent with the constraints: a branch literal specifies a truth value for all its ground instances unless there is a branch literal with a greater least solution specifying the opposite truth value

Non-Contradictory Branches

The model construction works only for non-contradictory branches



- **Contradictory branch**: for some consistent assignment of the constants, two complementary branch literals have the same least solution
- The branch above is contradictory: take a=4
- The calculus will never builds contradictory branches

Inference Rule - Split

$\neg P(x) \leftarrow a + 2 < x$

$\neg P(a+3)$
$\neg P(a+4)$
$\neg P(a+5)$
$\dots$

I: $P(a+1)$, $P(a+2)$, $P(a+3)$, $\dots$

$\mathsf{a} : [1 \,..\, 10]$

$P(x) \mid \mathsf{a} < x$ $\qquad$ $\neg P(x) \mid \mathsf{a} < x$

$\neg P(x) \mid \mathsf{a} + 2 < x$ $\qquad$ $P(x) \mid \mathsf{a} + 2 < x$

I: $P(a+1)$, $P(a+2)$, $\neg P(a+3)$, $\neg P(a+4)$, $\dots$

Repair interpretation:

Context unifier $\mathsf{a} < x \wedge \mathsf{a} + 2 < x$

Equivalently $\mathsf{a} + 2 < x$

Split candidate $\neg P(x) \mid \mathsf{a} + 2 < x$

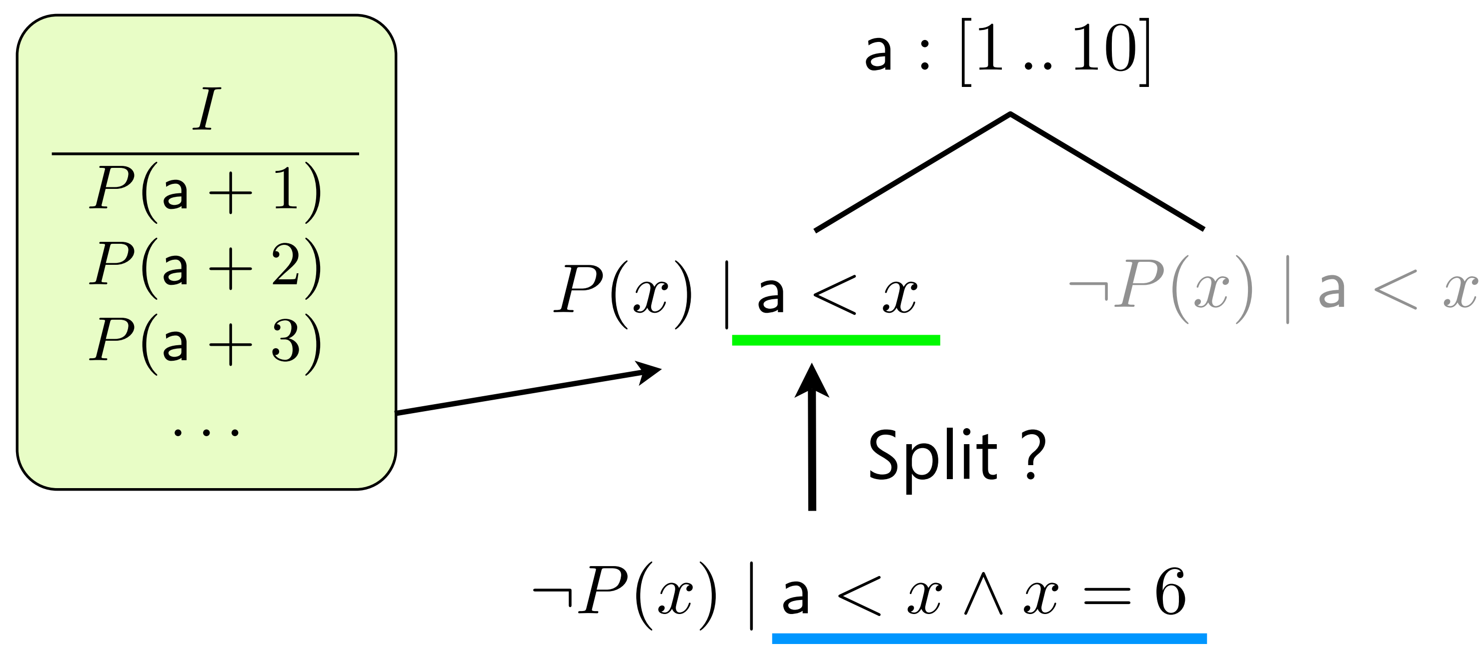
Non-contradictory $\mathsf{a} : [1 \,..\, 10] \models \mathsf{a} + 1 \neq \mathsf{a} + 3$

$\Rightarrow$ Split with candidate is applicable

Inference Rule - Domain Split

$\neg P(x) \leftarrow x = 6$

$\neg P(6)$



Split domain of constant a

Context unifier $\mathsf{a} < x \wedge x = 6$

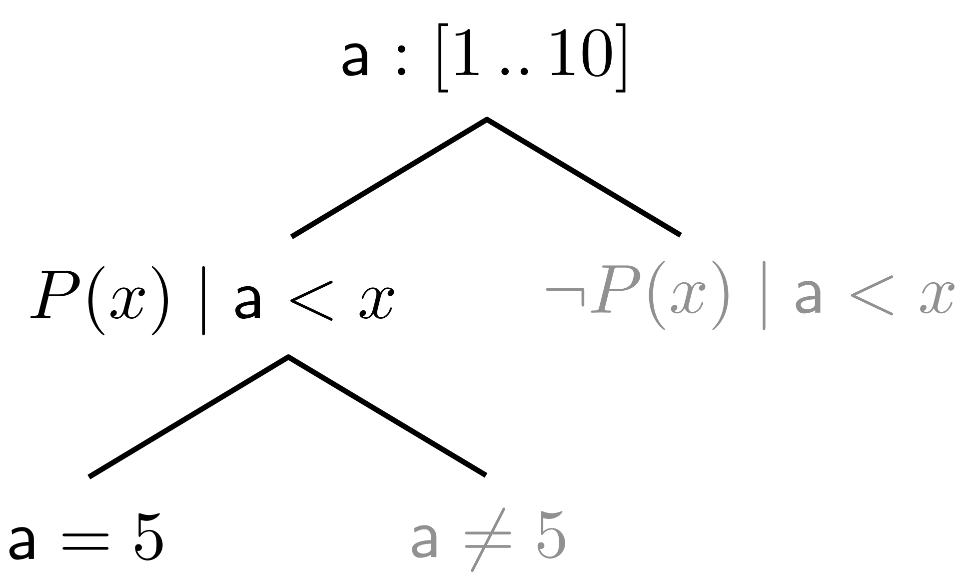
Split candidate $\neg P(x) \mid \mathsf{a} < x \wedge x = 6$

Least solutions of $\mathsf{a} < x$ and $\mathsf{a} < x \wedge x = 6$ are the same if $\mathsf{a} = 5$. Split not applicable:

Contradictory $\mathsf{a} : [1\,..\,10] \not\models \mathsf{a} \neq 5$

(And also $\mathsf{a} : [1\,..\,10] \not\models \mathsf{a} = 5$)

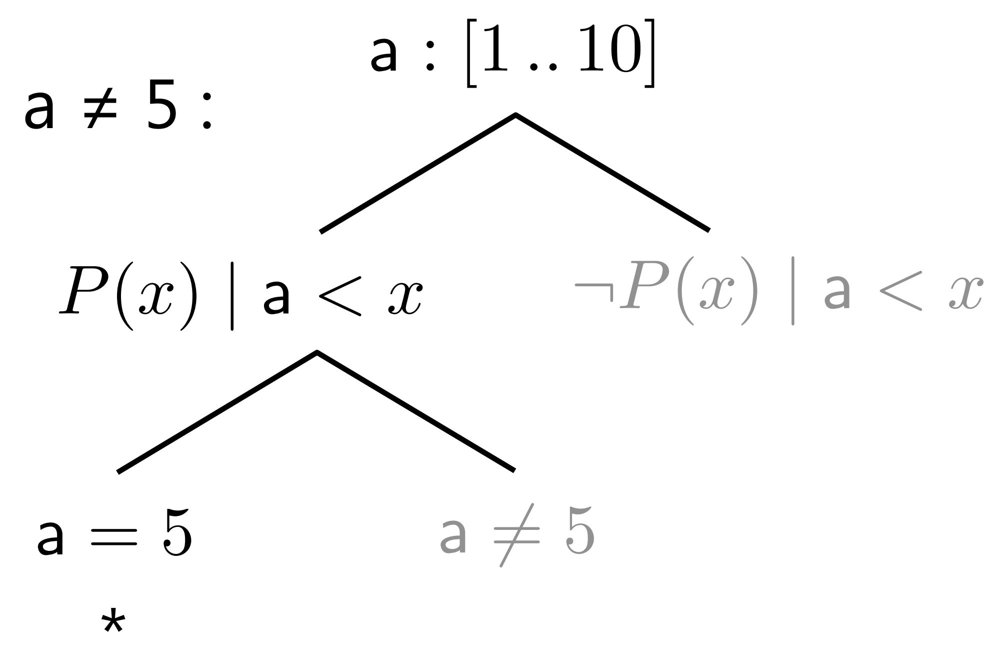
$\Rightarrow$ Domain Split with a = 5 is applicable



Inference Rule - Close

$\neg P(x) \leftarrow x = 6$

$\neg P(6)$



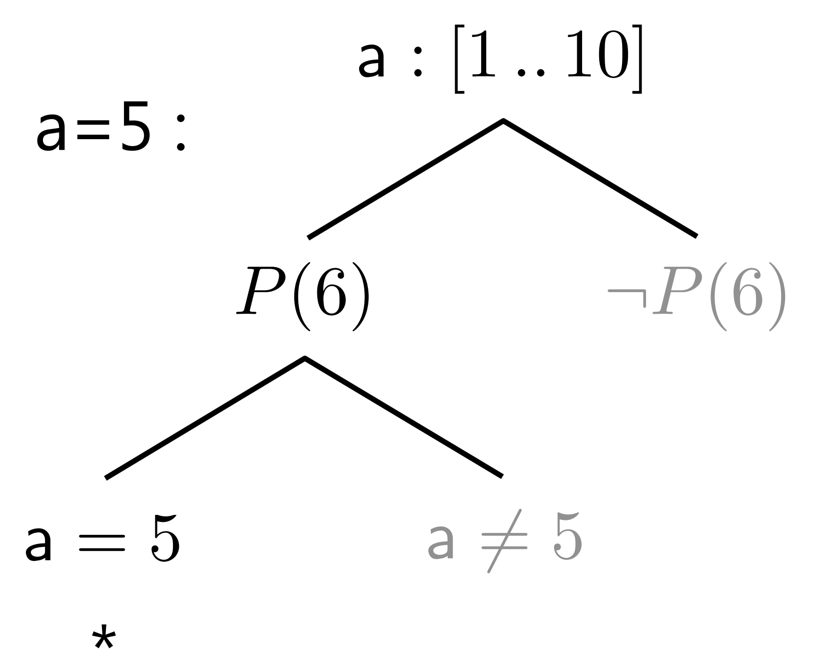
The left branch is closed

- If a ≠ 5 then

  the left branch does not satisfy a = 5

- If a = 5 then

  the least solutions of the branch literal and the context unifier are the same



This is the Soundness argument

In Reality ...

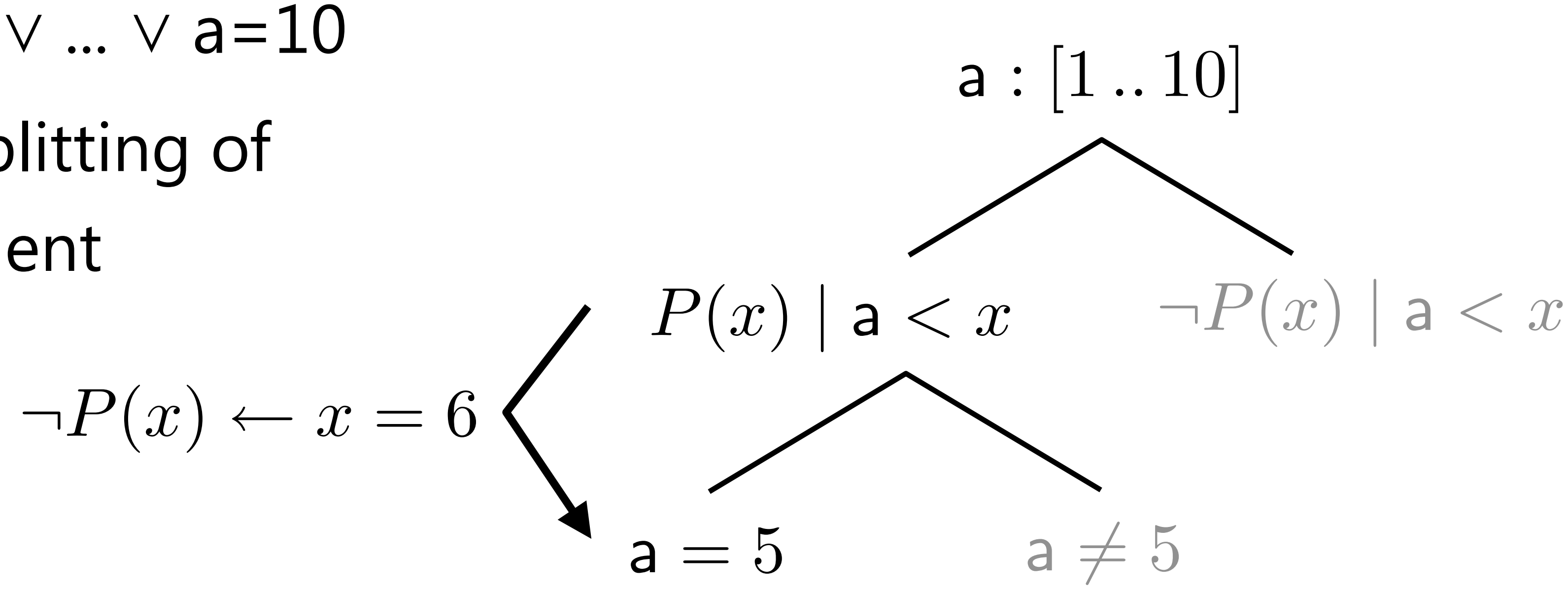
- ...the calculus works not just with unary clauses and unary predicates
- ...n-ary predicates: pointwise **minimal** solutions instead of the **least** ones
  - Example: $P(x,y) \leftarrow x \neq y$ has two minimal solutions: (0,1) and (1,0)
- Can define for a constraint, e.g., $x \neq y$ by formulas over constraint language:
  - The lexicographically least solution of $x \neq y$
  - The pointwise minimal solutions of $x \neq y$
  - The i-th pointwise minimal solution of $x \neq y$, which is the formula expressing the lexicographic least solution of
    - $\mu 1 \; x \neq y$ = "(x,y) is a pointwise minimal solution of $x \neq y$"
    - $\mu 2 \; x \neq y$ = "(x,y) is a pointwise minimal solution of $x \neq y$ and (x,y) does not satisfy $\mu 1 \; x \neq y$"
    - $\mu 3 \; x \neq y$ = "..." is unsatisfiable
  - Inference rules need effective satisfiability test for closed LIA-constraints

Main Result

- **Soundness**
  - As indicated above
- **Completeness**
  - Fair derivations via branch saturation (one branch at a time)
  - Every saturated open (limit) branch B specifies a model of the clause set
  - Proof idea: assume B falsifies a ground instance of a clause C.
    Then show that one of the following cases applies
    - B is closed [contradictory for all assignments]
    - Domain Split is applicable [contradictory for some assignments]
    - An inference rule is applicable to satisfy C
      [contradictory for no assignments)]
  - Each case leads to a contradiction

Why ME(LIA) Could be Good in Practice

- **Semantic Redundancy Criterion**
  - Can ignore clauses that are satisfied in current interpretation
- **Domain Splitting**
  - Domain decl a : [1 .. 10] could be eliminated using a=1 ∨ … ∨ a=10
  - But demand-driven splitting of domains is more efficient



  - Application to finite model computation:

    $\mathsf{a} : [1 \,..\, 10] \qquad \mathrm{P}(\mathsf{a}) \qquad \neg\mathrm{P}(\mathrm{x}) \leftarrow 1 \le \mathrm{x} \le 10$

    can be refuted in O(1) steps.
    Model finders need O(n) steps (here: n=10)

ME(LIA) Variations

- **No constants**
  - ME(LIA) not a decision procedure
    - There are clause sets that don't admit finite model representation with contexts
  - But ME(LIA) is sound and complete

$P(0)$
$\neg P(1)$
$P(x) \leftrightarrow P(x+2)$

- **Parameters unbounded**
  I.e. for "declarations" a : [0 .. ∞]
  - No complete calculus possible then
    - Can express domain emptyness problem of 2-register machines
    - Can express multiplication
  - Ignore? Add induction?

$P(0)$
$P(x+1) \leftarrow P(x)$
$\neg P(a)$

ME(LIA) Variations

- **Variables bounded**
  I.e. additionally finite domain restriction for free variables
  - ME(LIA) derivations are finite then
  - Application e.g. arrays
    (Totality axiom only)

Unfolding into disjunctions "by demand" only

$\forall i : [1 \mathinner{..} 10]\ \exists v : [1 \mathinner{..} 20]\ \mathsf{select\_a1}(i, v)$

becomes

$\mathsf{v}_1 : [1 \mathinner{..} 20] \qquad \mathsf{select\_a1}(i, v) \leftarrow i = 1 \wedge v = \mathsf{v}_1$

$\vdots$

$\mathsf{v}_{10} : [1 \mathinner{..} 20] \qquad \mathsf{select\_a1}(i, v) \leftarrow i = 10 \wedge v = \mathsf{v}_{10}$

Conclusions

Summary

- Sound and complete thanks to native quantifier treatment
- Needs ("only") a satisfiability checker for LIA
- Avoids expanding finite domains into disjunctions
- Model building capabilities
  - Application: countermodels for wrong conjectures
  - Countermodel then is more informative than "don't know" answer from system based on instantiation heuristics

Todo

- Universal literals, unit propagation and related inference rules
- Generalize parameters to functions with finite range ([BGW 94])
- Herbrand terms, equality (e.g. to axiomatize lists, arrays)