Automating Induction for Solving Horn Clauses

Hiroshi Unno, Sho Torii, and Hiroki Sakamoto

University of Tsukuba, Japan

Program Verification via Horn Constraint Solving

Verification Problems of Programs in

Various Paradigms (e.g., functional [U.+ '08, '09, Rondon+ '08, …], procedural [Grebenshchikov+ '12, Gurfinkel+ '15], object-oriented [Kahsai+ '16], multi-threaded [Gupta+ '11], constraint logic) with

Advanced Language Features (e.g., algebraic data structures, linked data structures, exceptions, higher-order functions) with

Side-Effects (e.g., non-termination, non-determinism, concurrency, assertions, destructive updates)

Reduce

Horn Constraint Solving Problems

Overall Flow of Horn Constraint based Program Verification

$$P(x, 0, 0)$$
$$P(x, y, x + r) \Leftarrow P(x, y - 1, r) \wedge y \neq 0$$
$$r \geq 0 \Leftarrow P(x, y, r) \wedge x \geq 0 \wedge y \geq 0$$

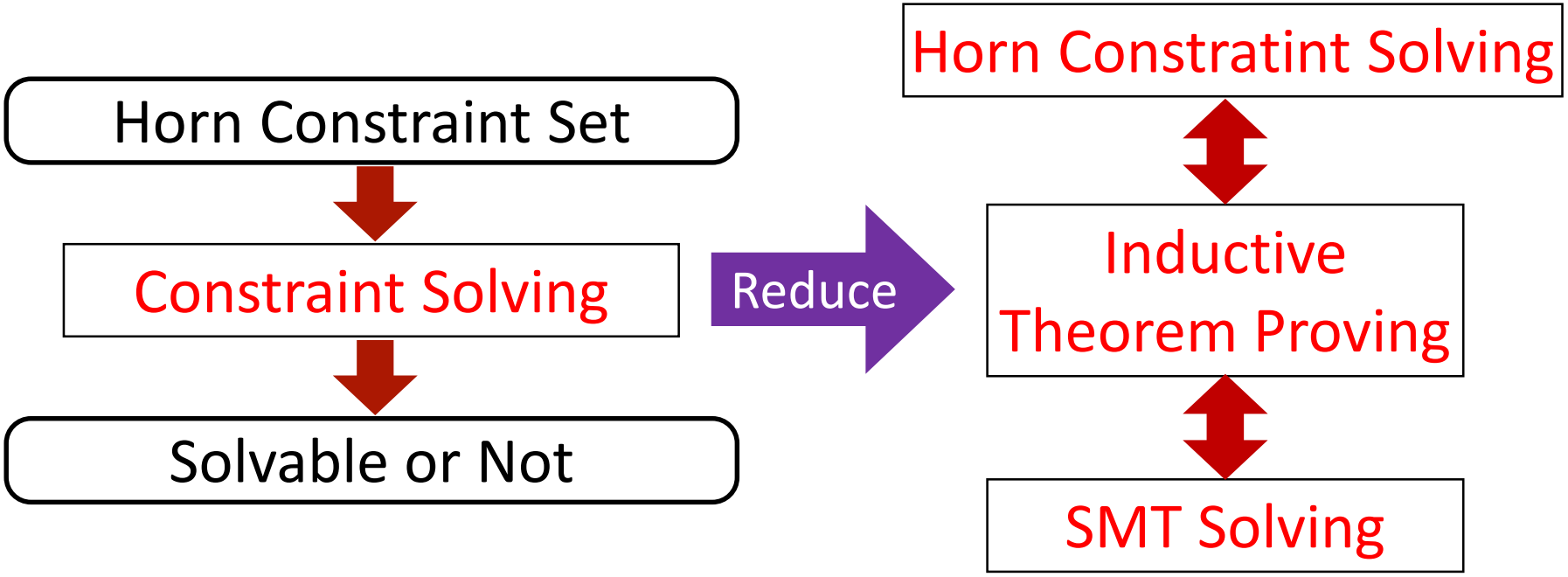
```
(* OCaml *)
let rec mult x y =
  if y = 0 then 0
  else x + mult x (y - 1)
```

```
{- Haskell -}
mult :: Int -> Int -> Int
mult x 0 = 0
mult x y = x + mult x (y - 1)
```

```
int s = 0;
while(y != 0){
}
return s;
}
```

$$P(x, y, r) \equiv r = x \times y$$

This Work

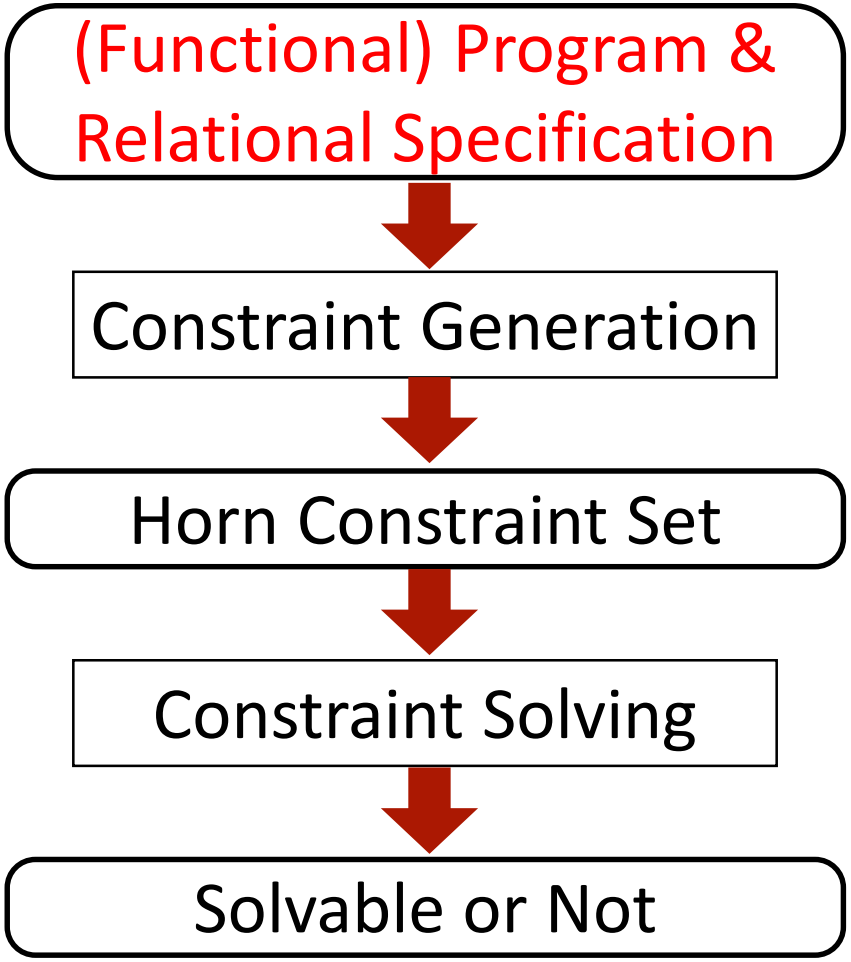


- Enable verification of ***relational specifications*** across programs in various paradigms
- Support constraints over any background theories (if the backend SMT solver does)

Relational Specifications

- Specifications that involve multiple function calls
  - Equivalence
  - Invertibility
  - Non-interference
  - Associativity
  - Commutativity
  - Distributivity
  - Monotonicity
  - Idempotency
  - …

Overall Flow of Horn Constraint based Program Verification



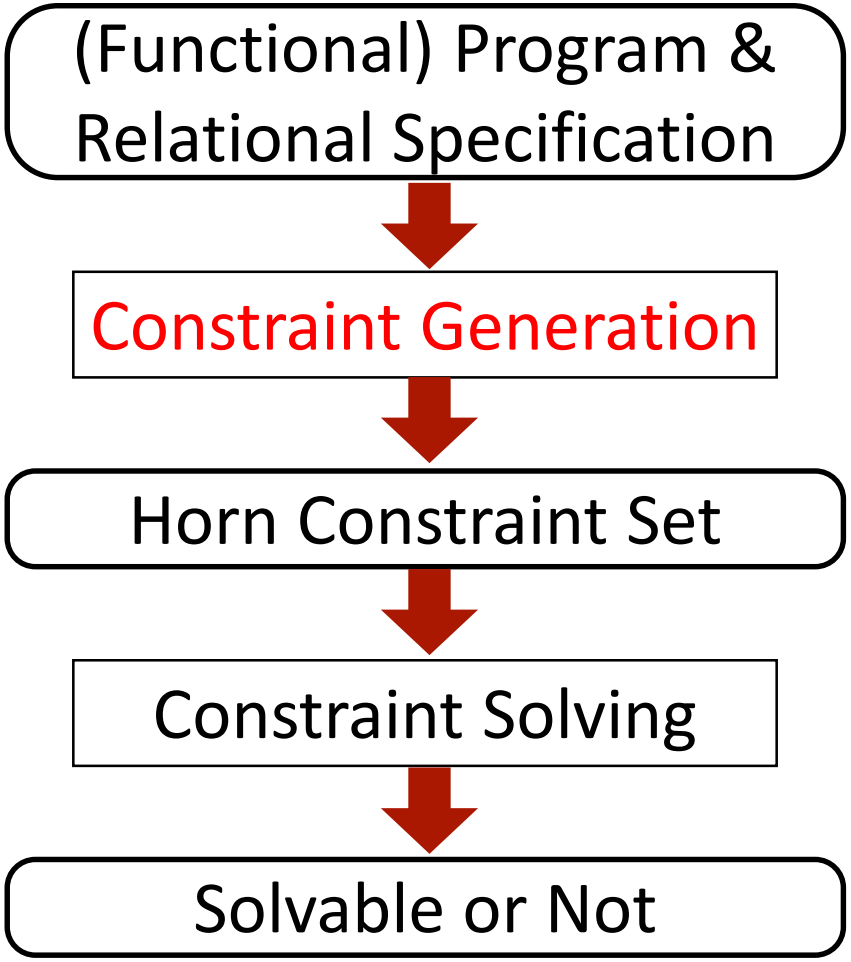
Example: (Functional) Program and Relational Specification

```
(* recursive function to compute "x × y" *)
let rec mult x y =
  if y = 0 then 0 else x + mult x (y - 1)

(* tail recursive function to compute "x × y + a" *)
let rec mult_acc x y a =
  if y = 0 then a else mult_acc x (y - 1) (a + x)

(* functional equivalence of mult and mult_acc *)
let main x y a = assert (mult x y + a = mult_acc x y a)
```

Overall Flow of Horn Constraint based Program Verification



Horn Constraint Generation [U.+ '09]

```
let rec mult x y =
  if y = 0 then 0
  else x + mult x (y - 1)
let rec mult_acc x y a =
  if y = 0 then a
  else mult_acc x (y - 1) (a + x)
```

```
let main x y a =
  assert (mult x y + a
          = mult_acc x y a)
```

$$P(x, 0, 0)$$

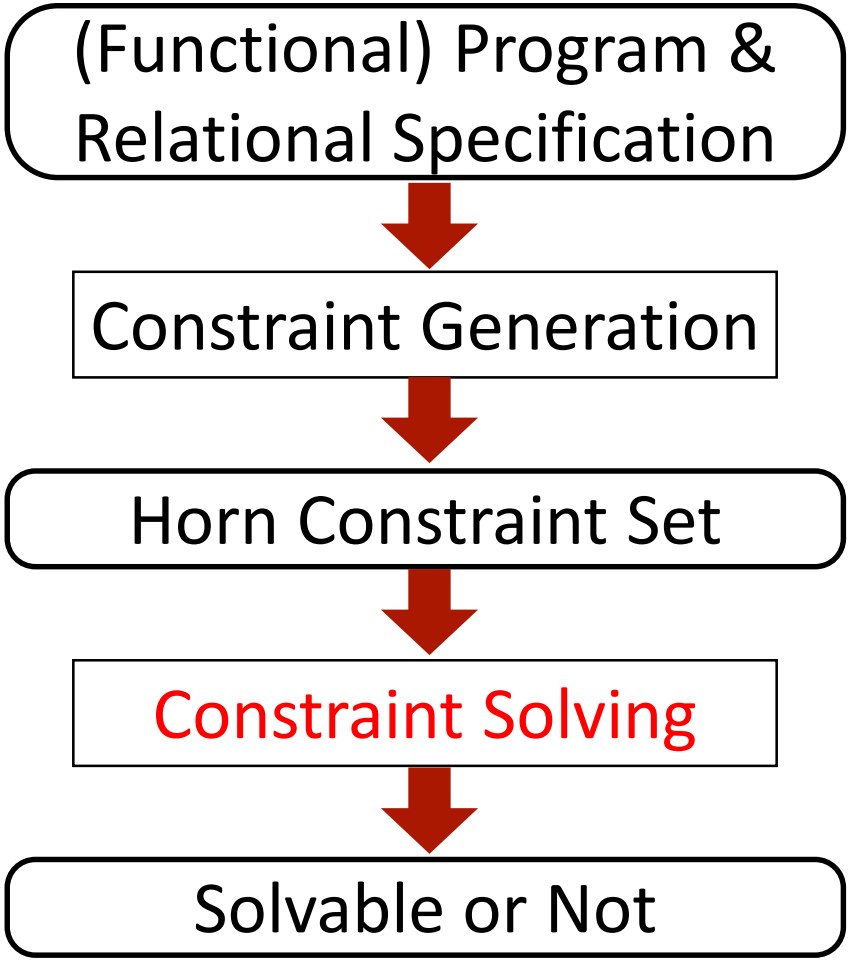
$$P(x, y, x + r) \Leftarrow P(x, y - 1, r) \wedge y \neq 0$$

$$Q(x, 0, a, a)$$

$$Q(x, y, a, r) \Leftarrow Q(x, y - 1, a + x, r) \wedge y \neq 0$$

$$s_1 + a = s_2 \Leftarrow P(x, y, s_1) \wedge Q(x, y, a, s_2)$$

Overall Flow of Horn Constraint based Program Verification



Horn Constraint Solving

- Check the existence of a solution for predicate variables satisfying all the Horn constraints
  - If a solution exists, the original program is guaranteed to satisfy the specification

Example (Non-relational) specification:

```
let main x y = if x >= 0 && y >= 0 then assert (mult x y >= 0)
```

$$P(x, 0, 0)$$

$$P(x, y, x + r) \Leftarrow P(x, y - 1, r) \wedge y \neq 0$$

$$r \geq 0 \Leftarrow P(x, y, r) \wedge x \geq 0 \wedge y \geq 0$$

Solution 1: $P(x, y, r) \equiv x \geq 0 \wedge y \geq 0 \Rightarrow r \geq 0$

Solution 2: $P(x, y, r) \equiv r = x \times y$

Nonlinear

QF-LIA

QF-NIA

Previous Methods for Solving Horn Clause Constraints [U.+ '08,'09, Rondon+ '08, Gupta+ '11, Hoder+ '11,'12, McMillan+ '13, Rümmer+ '13, …]

Find a solution expressible in QF-LIA (or QF-LRA)

$$P(x, 0, 0)$$
$$P(x, y, x + r) \Leftarrow P(x, y - 1, r) \wedge y \neq 0$$
$$r \geq 0 \Leftarrow P(x, y, r) \wedge x \geq 0 \wedge y \geq 0$$

Solution 1: $P(x, y, r) \equiv x \geq 0 \wedge y \geq 0 \Rightarrow r \geq 0$

QF-LIA

~~Solution 2: $P(x, y, r) \equiv r = x \times y$~~

QF-NIA

Example Constraints that Can Not be Solved by Previous Methods

Constraint Solving Fails!

$$P(x, 0, 0)$$

$$P(x, y, x + r) \Leftarrow P(x, y - 1, \ldots)$$

$$Q(x, 0, a, a)$$

$$Q(x, y, a, r) \Leftarrow Q(x, y - 1, a + x, r) \wedge y \neq 0$$

$$s_1 + a = s_2 \Leftarrow P(x, y, s_1) \wedge Q(x, y, a, s_2)$$

QF-NIA

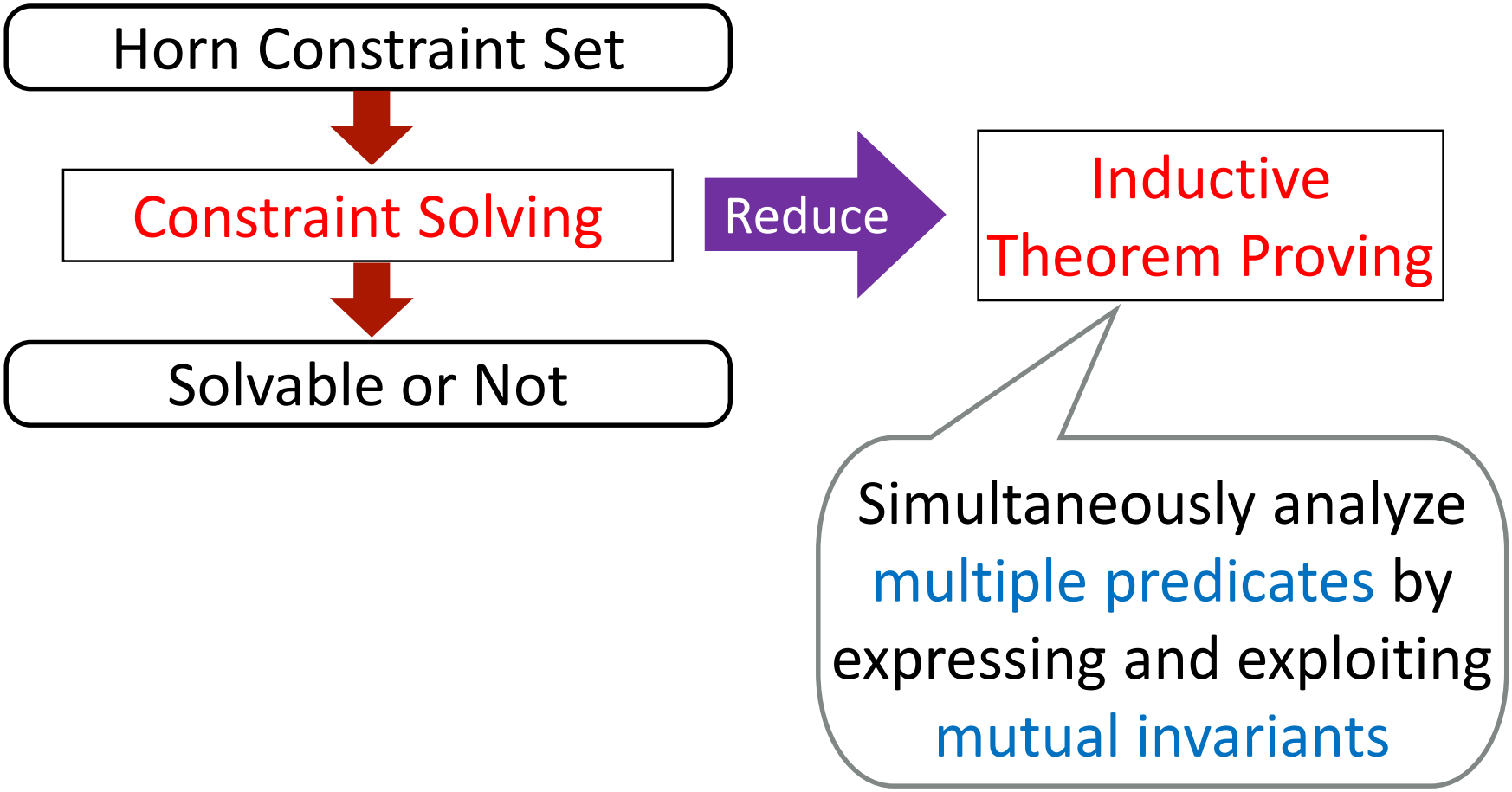
Analyzed separately from Q

Analyzed separately from P

$P(x, y, s_1) \equiv s_1 = x \times y$

$Q(x, y, a, s_2) \equiv s_2 = x \times y + a$

Our Constraint Solving Method



Reduction from Constraint Solving to Inductive Theorem Proving

$P(x,0,0) \quad P(x,y,x+r) \Leftarrow P(x,y-1,r) \wedge y \neq 0$

$Q(x,0,a,a) \quad Q(x,y,a,r) \Leftarrow Q(x,y-1,a+x,r) \wedge y \neq 0$

$s_1 + a = s_2 \Leftarrow P(x,y,s_1) \wedge Q(x,y,a,s_2)$

Prove this by induction on derivation of $P(x,y,s_1)$, $Q(x,y,s_2)$

$$\frac{\models y = 0 \wedge r = 0}{P(x,y,r)} \qquad \frac{P(x,y-1,r-x) \quad \models y \neq 0}{P(x,y,r)}$$

$$\frac{\models y = 0 \wedge a = r}{Q(x,y,a,r)} \qquad \frac{Q(x,y-1,a+x,r) \quad \models y \neq 0}{Q(x,y,a,r)}$$

$\forall x,y,s_1,a,s_2.\, P(x,y,s_1) \wedge Q(x,y,a,s_2) \Rightarrow s_1 + a = s_2$

Principle of Induction on Derivation

$$\forall D.\ \psi(D) \quad \text{if and only if}$$
$$\forall D.\ \left(\forall D'.\ D' \prec D \Rightarrow \psi(D')\right) \Rightarrow \psi(D)$$

where $D' \prec D$ represents that
D' is a strict sub-derivation of D

$$D = \frac{\dfrac{D_1}{J_3} \quad \dfrac{D_2}{J_2} \quad D_3 \quad \dfrac{D_4}{J_4}}{J_1}$$

Assume
$\psi(D_1), \psi(D_2),$
$\psi(D_3), \psi(D_4)$
and prove $\psi(D)$

Horn Constraint Solving:

$$P(x, 0, 0)$$

$$P(x, y, x + r) \Leftarrow P(x, y - 1, r) \wedge y \neq 0$$

$$Q(x, 0, a, a)$$

$$Q(x, y, a, r) \Leftarrow Q(x, y - 1, a + x, r) \wedge y \neq 0$$

$$s_1 + a = s_2 \Leftarrow P(x, y, s_1) \wedge Q(x, y, a, s_2)$$

Induction hypotheses and lemmas

Judgment

$$\emptyset; P(x, y, s_1), Q(x, y, a, s_2) \vdash s_1 + a = s_2$$

Premises

$$\frac{\models y = 0 \wedge \qquad , y - 1, r - x) \qquad \models y \neq 0}{P(x, y, r)} \qquad \frac{}{P(x, y, r)}$$

$$\frac{\models y = 0 \wedge a = r}{Q(x, y, a, r)} \qquad \frac{Q(x, y - 1, a + x, r) \qquad \models y \neq 0}{Q(x, y, a, r)}$$

$$\frac{\models y = 0 \land r = 0}{P(x,y,r)} \qquad \frac{P(x,y-1,r-x) \quad \models y \neq 0}{P(x,y,r)}$$

$$\frac{\models y = 0 \land a = r}{Q(x,y,a,r)} \qquad \frac{Q(x,y-1,a+x,r) \quad \models y \neq 0}{Q(x,y,a,r)}$$

Add an induction hypothesis

Guard to avoid unsound application

$$\gamma = \forall x', y', s_1', a', s_2'.\, D\big(P(x',y',s_1')\big) \prec D\big(P(x,y,s_1)\big) \land P(x',y',{s_1}') \land Q(x',y',a',s_2') \Rightarrow {s_1}' + a' = {s_2}'$$

Induct

Unfold

Case analysis on the last rule used

$$\frac{\gamma; \cdots, y = 0 \land s_1 = 0 \vdash \cdots \qquad \gamma; \cdots, P(x,y-1,s_1-x), y \neq 0 \vdash \cdots}{\emptyset; P(x,y,s_1), Q(x,y,a,s_2) \vdash s_1 + a = s_2}$$

$$\frac{\models y = 0 \land r = 0}{P(x, y, r)} \qquad \frac{P(x, y-1, r-x) \quad \models y \neq 0}{P(x, y, r)}$$

$$\frac{\models y = 0 \land a = r}{Q(x, y, a, r)} \qquad \frac{Q(x, y-1, a+x, r) \quad \models y \neq 0}{Q(x, y, a, r)}$$

$$\frac{\gamma; \cdots, y = 0 \land s_1 = 0 \vdash \cdots}{\emptyset; P(x, y, s_1), Q(x, y, a, s_2) \vdash s_1 + a = s_2}$$

$$\frac{\models y = 0 \land r = 0}{P(x, y, r)} \qquad \frac{P(x, y - 1, r - x) \quad \models y \neq 0}{P(x, y, r)}$$

$$\frac{\models y = 0 \land a = r}{Q(x, y, a, r)} \qquad \frac{Q(x, y - 1, a + x, r) \quad \models y \neq 0}{Q(x, y, a, r)}$$

Case analysis on the last rule used

Unfold

$$\frac{\gamma; \cdots, \cdots \land y = 0 \land a = s_2 \vdash \cdots \qquad \gamma; \cdots, Q(x, y - 1, a + x, s_2), \cdots \land y \neq 0 \vdash \cdots}{\gamma; P(x, y, s_1), Q(x, y, a, s_2), y = 0 \land s_1 = 0 \vdash s_1 + a = s_2}$$

$$\emptyset; P(x, y, s_1), Q(x, y, a, s_2) \vdash s_1 + a = s_2$$

$$\frac{\models y = 0 \wedge r = 0}{P(x, y, r)} \qquad \frac{P(x, y-1, r-x) \quad \models y \neq 0}{P(x, y, r)}$$

$$\frac{\models y = 0 \wedge a = r}{Q(x, y, a, r)} \qquad \frac{Q(x, y-1, a+x, r) \quad \models y \neq 0}{Q(x, y, a, r)}$$

$$\frac{\gamma; \cdots, \cdots \wedge y = 0 \wedge a = s_2 \vdash \ \cdots}{\gamma; P(x, y, s_1), Q(x, y, a, s_2), y = 0 \wedge s_1 = 0 \vdash s_1 + a = s_2}$$

$$\frac{}{\emptyset; P(x, y, s_1), Q(x, y, a, s_2) \vdash s_1 + a = s_2}$$

$$\frac{\models y = 0 \wedge r = 0}{P(x, y, r)} \qquad \frac{P(x, y-1, r-x) \quad \models y \neq 0}{P(x, y, r)}$$

$$\frac{\models y = 0 \wedge a = r}{Q(x, y, a, r)} \qquad \frac{Q(x, y-1, a+x, r) \quad \models y \neq 0}{Q(x, y, a, r)}$$

Validity checking

Valid

$$\frac{\models y = 0 \wedge s_1 = 0 \wedge a = s_2 \Rightarrow s_1 + a = s_2}{\gamma; \cdots, y = 0 \wedge s_1 = 0 \wedge a = s_2 \vdash s_1 + a = s_2}$$

$$\frac{\gamma; \cdots, y = 0 \wedge s_1 = 0 \wedge a = s_2 \vdash s_1 + a = s_2}{\gamma; P(x, y, s_1), Q(x, y, a, s_2), y = 0 \wedge s_1 = 0 \vdash s_1 + a = s_2}$$

$$\frac{\gamma; P(x, y, s_1), Q(x, y, a, s_2), y = 0 \wedge s_1 = 0 \vdash s_1 + a = s_2}{\emptyset; P(x, y, s_1), Q(x, y, a, s_2) \vdash s_1 + a = s_2}$$

$$\frac{\models y = 0 \wedge r = 0}{P(x, y, r)} \qquad \frac{P(x, y-1, r-x) \quad \models y \neq 0}{P(x, y, r)}$$

$$\frac{\models y = 0 \wedge a = r}{Q(x, y, a, r)} \qquad \frac{Q(x, y-1, a+x, r) \quad \models y \neq 0}{Q(x, y, a, r)}$$

$$\frac{\gamma; \cdots, Q(x, y-1, a+x, s_2), \cdots \wedge y \neq 0 \vdash \cdots}{\gamma; P(x, y, s_1), Q(x, y, a, s_2), y = 0 \wedge s_1 = 0 \vdash s_1 + a = s_2}$$

$$\frac{}{\emptyset; P(x, y, s_1), Q(x, y, a, s_2) \vdash s_1 + a = s_2}$$

$$\frac{\models y = 0 \wedge r = 0}{P(x, y, r)} \qquad \frac{P(x, y-1, r-x) \quad \models y \neq 0}{P(x, y, r)}$$

$$\frac{\models y = 0 \wedge a = r}{Q(x, y, a, r)} \qquad \frac{Q(x, y-1, a+x, r) \quad \models y \neq 0}{Q(x, y, a, r)}$$

Valid

$$\frac{\models y = 0 \wedge s_1 = 0 \wedge y \neq 0 \Rightarrow s_1 + a = s_2}{\gamma; \cdots, Q(x, y-1, a+x, s_2), y = 0 \wedge s_1 = 0 \wedge y \neq 0 \vdash s_1 + a = s_2}$$
$$\frac{}{\gamma; P(x, y, s_1), Q(x, y, a, s_2), y = 0 \wedge s_1 = 0 \vdash s_1 + a = s_2}$$
$$\frac{}{\emptyset; P(x, y, s_1), Q(x, y, a, s_2) \vdash s_1 + a = s_2}$$

$$\frac{\models y = 0 \wedge r = 0}{P(x, y, r)} \qquad \frac{P(x, y-1, r-x) \quad \models y \neq 0}{P(x, y, r)}$$

$$\frac{\models y = 0 \wedge a = r}{Q(x, y, a, r)} \qquad \frac{Q(x, y-1, a+x, r) \quad \models y \neq 0}{Q(x, y, a, r)}$$

$$\frac{\gamma; \cdots, P(x, y-1, s_1 - x), y \neq 0 \vdash \cdots}{\emptyset; P(x, y, s_1), Q(x, y, a, s_2) \vdash s_1 + a = s_2}$$

$$\frac{\models y = 0 \land r = 0}{P(x, y, r)} \qquad \frac{P(x, y-1, r-x) \quad \models y \neq 0}{P(x, y, r)}$$

$$\frac{\models y = 0 \land a = r}{Q(x, y, a, r)} \qquad \frac{Q(x, y-1, a+x, r) \quad \models y \neq 0}{Q(x, y, a, r)}$$

Unfold

Case analysis on the last rule used

$$\frac{\gamma; \cdots, \cdots \land y = 0 \land a = s_2 \vdash \cdots \qquad \gamma; \cdots, Q(x, y-1, a+x, s_2), \cdots \land y \neq 0 \vdash \cdots}{\gamma; P(x, y, s_1), Q(x, y, a, s_2), P(x, y-1, s_1 - x), y \neq 0 \vdash s_1 + a = s_2}$$

$$\emptyset; P(x, y, s_1), Q(x, y, a, s_2) \vdash s_1 + a = s_2$$

$$\frac{\models y = 0 \wedge r = 0}{P(x, y, r)}$$

$$\frac{\models y = 0 \wedge a = r}{Q(x, y, a, r)}$$

$$\frac{P(x, y-1, r-x) \quad \models y \neq 0}{P(x, y, r)}$$

$$\frac{Q(x, y-1, a+x, r) \quad \models y \neq 0}{Q(x, y, a, r)}$$

$$\frac{\gamma; \cdots, \cdots \wedge y = 0 \wedge a = s_2 \vdash \ \cdots}{\gamma; P(x, y, s_1), Q(x, y, a, s_2), P(x, y-1, s_1 - x), y \neq 0 \vdash s_1 + a = s_2}$$

$$\emptyset; P(x, y, s_1), Q(x, y, a, s_2) \vdash s_1 + a = s_2$$

$$\frac{\models y = 0 \wedge r = 0}{P(x, y, r)} \qquad \frac{P(x, y-1, r-x) \quad \models y \neq 0}{P(x, y, r)}$$

$$\frac{\models y = 0 \wedge a = r}{Q(x, y, a, r)} \qquad \frac{Q(x, y-1, a+x, r) \quad \models y \neq 0}{Q(x, y, a, r)}$$

Valid

$$\dfrac{\dfrac{\models y \neq 0 \wedge y = 0 \wedge a = s_2 \Rightarrow s_1 + a = s_2}{\gamma; \cdots, y \neq 0 \wedge y = 0 \wedge a = s_2 \vdash s_1 + a = s_2}}{\gamma; P(x, y, s_1), Q(x, y, a, s_2), P(x, y-1, s_1 - x), y \neq 0 \vdash s_1 + a = s_2}$$

$$\emptyset; P(x, y, s_1), Q(x, y, a, s_2) \vdash s_1 + a = s_2$$

$$\frac{\models y = 0 \wedge r = 0}{P(x, y, r)} \qquad \frac{P(x, y-1, r-x) \quad \models y \neq 0}{P(x, y, r)}$$

$$\frac{\models y = 0 \wedge a = r}{Q(x, y, a, r)} \qquad \frac{Q(x, y-1, a+x, r) \quad \models y \neq 0}{Q(x, y, a, r)}$$

$$\frac{\gamma; \cdots, Q(x, y-1, a+x, s_2), \cdots \wedge y \neq 0 \vdash \cdots}{\gamma; P(x, y, s_1), Q(x, y, a, s_2), P(x, y-1, s_1 - x), y \neq 0 \vdash s_1 + a = s_2}$$

$$\emptyset; P(x, y, s_1), Q(x, y, a, s_2) \vdash s_1 + a = s_2$$

$$\frac{\models y = 0 \wedge r = 0}{P(x, y, r)} \qquad \frac{P(x, y-1, r-x) \quad \models y \neq 0}{P(x, y, r)}$$

$$\frac{\models y = 0 \wedge a = r}{Q(x, y, a, r)} \qquad \frac{Q(x, y-1, a+x, r) \quad \models y \neq 0}{Q(x, y, a, r)}$$

$$\sigma(\gamma) = D\big(P(x, y-1, s_1 - x)\big) \prec D\big(P(x, y, s_1)\big) \wedge P(x, y-1, s_1 - x) \wedge Q(x, y-1, a+x, s_2) \Rightarrow (s_1 - x) + (a + x) = s_2$$

IndHyp Apply induction hypothesis

$$\frac{\gamma; \cdots, y \neq 0 \wedge (s_1 - x) + (a + x) = s_2 \vdash s_1 + a = s_2}{\gamma; \cdots, P(x, y-1, s_1 - x), Q(x, y-1, a+x, s_2), y \neq 0 \vdash s_1 + a = s_2}$$

$$\frac{\gamma; P(x, y, s_1), Q(x, y, a, s_2), P(x, y-1, s_1 - x), y \neq 0 \vdash s_1 + a = s_2}{\emptyset; P(x, y, s_1), Q(x, y, a, s_2) \vdash s_1 + a = s_2}$$

$$\frac{\models y = 0 \land r = 0}{P(x, y, r)} \qquad \frac{P(x, y-1, r-x) \quad \models y \neq 0}{P(x, y, r)}$$

$$\frac{\models y = 0 \land a = r}{Q(x, y, a, r)} \qquad \frac{Q(x, y-1, a+x, r) \quad \models y \neq 0}{Q(x, y, a, r)}$$

Valid

$$\frac{\models y \neq 0 \land (s_1 - x) + (a + x) = s_2 \Rightarrow s_1 + a = s_2}{\gamma; \cdots, y \neq 0 \land (s_1 - x) + (a + x) = s_2 \vdash s_1 + a = s_2}$$

$$\frac{\gamma; \cdots, y \neq 0 \land (s_1 - x) + (a + x) = s_2 \vdash s_1 + a = s_2}{\gamma; \cdots, P(x, y-1, s_1 - x), Q(x, y-1, a+x, s_2), y \neq 0 \vdash s_1 + a = s_2}$$

$$\frac{\gamma; \cdots, P(x, y-1, s_1 - x), Q(x, y-1, a+x, s_2), y \neq 0 \vdash s_1 + a = s_2}{\gamma; P(x, y, s_1), Q(x, y, a, s_2), P(x, y-1, s_1 - x), y \neq 0 \vdash s_1 + a = s_2}$$

$$\frac{\gamma; P(x, y, s_1), Q(x, y, a, s_2), P(x, y-1, s_1 - x), y \neq 0 \vdash s_1 + a = s_2}{\emptyset; P(x, y, s_1), Q(x, y, a, s_2) \vdash s_1 + a = s_2}$$

QED

Properties of Inductive Proof System for Horn Constraint Solving

- Soundness: If the goal is proved, the original Horn constraints have a solution (which may not be expressible in the underlying logic)
- *Relative* Completeness: If the original constraints have a solution *expressible in the underlying logic*, the goal is provable

Automating Induction

- Use the following rule application strategy:
  - Repeatedly apply INDHYP until no new premises are added
  - Apply VALID whenever a new premise is added
  - Select some $P(\tilde{t})$ and apply INDUCT and UNFOLD
- Close a proof branch by using:
  - SMT solvers: provide efficient and powerful reasoning about **data structures** (e.g., integers, reals, algebraic data structures) but predicates are abstracted as uninterpreted functions
  - Horn constraint solvers: provide bit costly but powerful reasoning about **inductive predicates**

Prototype Constraint Solver

- Use **Z3** and **μZ PDR** engine respectively as the backend SMT and Horn constraint solvers
- Integrated with a refinement type based verification tool **RCaml** for the OCaml functional language

- Can exploit lemmas which are:
  - User-supplied,
  - Heuristically obtained from the given constraints, or
  - Automatically generated by an abstract interpreter
- Can generate a counterexample (if any)

Experiments on IsaPlanner Benchmark Set

- 85 (mostly) relational verification problems of total functions on inductively defined data structures

Inductive Theorem Prover	#Successfully Proved
RCaml	68
Zeno	82 [Sonnex+ '12]
HipSpec	80 [Claessen+ '13]
CVC4	80 [Reynolds+ '15]
ACL2s	74 (according to [Sonnex+ '12])
IsaPlanner	47 (according to [Sonnex+ '12])
Dafny	45 (according to [Sonnex+ '12])

Support automatic lemma discovery & goal generalization

Experiments on Benchmark Programs with Advanced Language Features & Side-Effects

- 30 (mostly) relational verification problems for:
  - Complex integer functions: Ackermann, McCarthy91
  - Nonlinear real functions: dyn_sys
  - Higher-order functions: fold_left, fold_right, repeat, find, ...
  - Exceptions: find
  - Non-terminating functions: mult, sum, …
  - Non-deterministic functions: randpos
  - Imperative procedures: mult_Ccode

ID	specification	kind	features	result	time (sec.)
1	mult $x\ y + a =$ mult_acc $x\ y\ a$	equiv	P	✓	0.378
2	mult $x\ y =$ mult_acc $x\ y\ 0$	equiv	P	✓†	0.803
3	mult $(1+x)\ y = y +$ mult $x\ y$	equiv	P	✓	0.403
4	$y \geq 0 \Rightarrow$ mult $x\ (1+y) = x +$ mult $x\ y$	equiv	P	✓	0.426
5	mult $x\ y =$ mult $y\ x$	comm	P	✓‡	0.389
6	mult $(x+y)\ z =$ mult $x\ z +$ mult $y\ z$	dist	P	✓	1.964
7	mult $x\ (y+z) =$ mult $x\ y +$ mult $x\ z$	dist	P	✓	4.360
8	mult (mult $x\ y$) $z =$ mult x (mult $y\ z$)	assoc	P	✗	n/a
9	$0 \leq x_1 \leq x_2 \wedge 0 \leq y_1 \leq y_2 \Rightarrow$ mult $x_1\ y_1 \leq$ mult $x_2\ y_2$	mono	P	✓	0.416
10	sum $x + a =$ sum_acc $x\ a$	equiv		✓	0.576
11	sum $x = x +$ sum $(x-1)$	equiv		✓	0.452
12	$x \leq y \Rightarrow$ sum $x \leq$ sum y	mono		✓	0.593
24	noninter $h_1\ l_1\ l_2\ l_3 =$ noninter $h_2\ l_1\ l_2\ l_3$	nonint	P	✓	1.203
25	try find_opt $p\ l =$ Some (find $p\ l$) with Not_Found $\rightarrow$ find_opt $p\ l =$ None	equiv	H, E	✓	1.065
26	try mem (find $((=)\ x)\ l$) l with Not_Found $\rightarrow \neg$(mem $x\ l$)	equiv	H, E	✓	1.056
27	sum_list $l =$ fold_left $(+)\ 0\ l$	equiv	H	✓	6.148
28	sum_list $l =$ fold_right $(+)\ l\ 0$	equiv	H	✓	0.508
29	sum_fun randpos $n > 0$	equiv	H,D	✓	0.319
30	mult $x\ y =$ mult_Ccode(x, y)	equiv	P, C	✓	0.303

† A lemma $P_{\texttt{mult\_acc}}(x, y, a, r) \Rightarrow P_{\texttt{mult\_acc}}(x, y, a - x, r - x)$ is used

‡ A lemma $P_{\texttt{mult}}(x, y, r) \Rightarrow P_{\texttt{mult}}(x - 1, y, r - y)$ is used

Used a machine with Intel(R) Xeon(R) CPU (2.50 GHz, 16 GB of memory).

- 28 (2 required lemmas) successfully proved by **RCaml**
- 3 proved by Horn constraint solver **μZ PDR**
- 2 proved by inductive theorem prover **CVC4** (if inductive predicates are encoded using uninterpreted functions)

Conclusion

- Proposed an automated verification method combining Horn constraint solving and inductive theorem proving
  - Enable relational verification across programs in various paradigms with advanced language features and side-effects
  - Support constraints over any background theories (if the backend SMT solver does)
- Future and ongoing work:
  - Automatic lemma discovery and goal generalization
  - Relational program synthesis
  - Coinduction