Relational Verification of Functional Programs via Induction-based Horn Constraint Solving

Hiroshi Unno (University of Tsukuba)

Joint work with Sho Torii

Refinement Caml (RCaml)

- A fully-automated verification tool for OCaml based on refinement type checking and inference
- Supported Language Features:
  - Higher-order recursive functions
  - Algebraic data types (ADTs)
  - Integers, booleans, unit, tuples, conditionals, pattern matching
- Supported Analyses:
  - Static assertion checking [U. and Kobayashi '08, '09, U.+ '13]
  - Termination and non-termination analysis [Hashimoto and U. '15, …]
  - (Maximally-weak) precondition inference [Hashimoto and U. '15, …]
  - Relational verification (this talk)
  - …

This Talk: Automated Verification of Relational Specifications of Higher-Order Functional Programs with ADTs

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

Demo of RCaml!

Relational Verification of Functional Programs via Induction-based Horn Constraint Solving

Hiroshi Unno (University of Tsukuba)

Joint work with Sho Torii

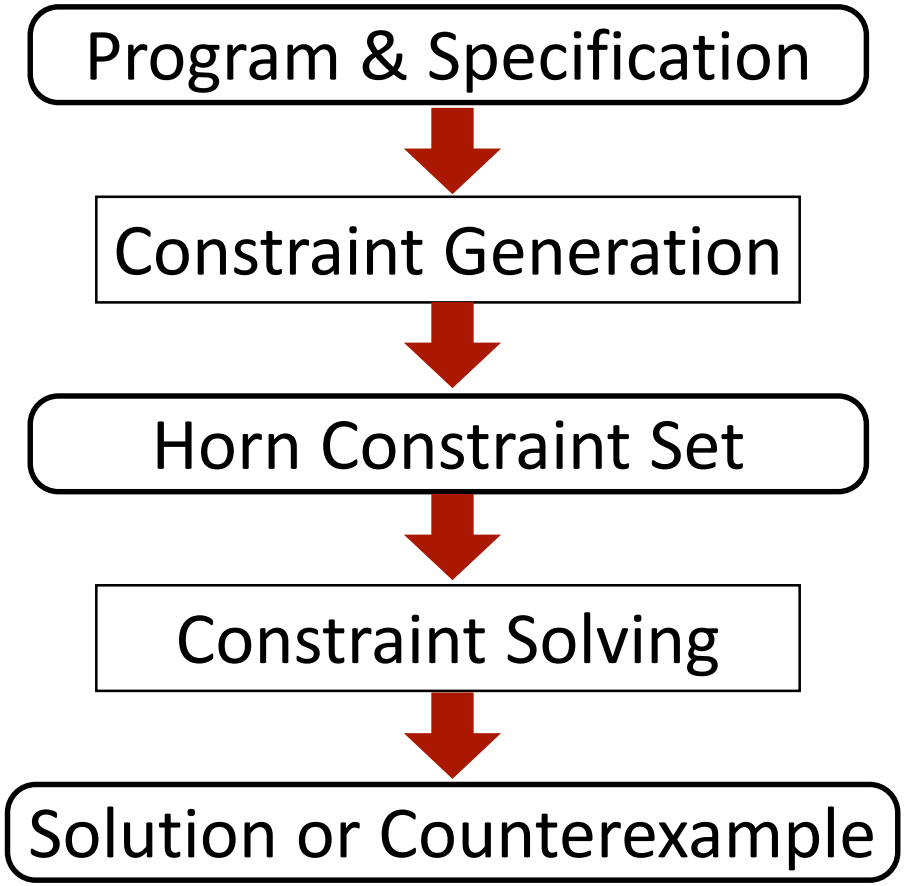
Program Verification via Horn Constraint Solving

[U.+ ’08,’09, Gupta+ ’11, Grebenshchikov+ ’12, …]

Verification Problems of Programs in Various Paradigms
(e.g., imperative, functional, logic, multi-threaded)
with Advanced Language Features
(e.g., higher-order functions, exceptions, ADTs)
with Side-Effects
(e.g., assertion failure, non-termination, evaluation orders)

Horn Constraint Solving Problems

Overall Flow of Horn Constraint based Program Verification



Overall Flow of Horn Constraint based Program Verification

Program & Specification

$$\forall x, y.\ x \geq 0 \wedge y \geq 0 \Rightarrow \mathrm{mult}\ x\ y \geq 0$$

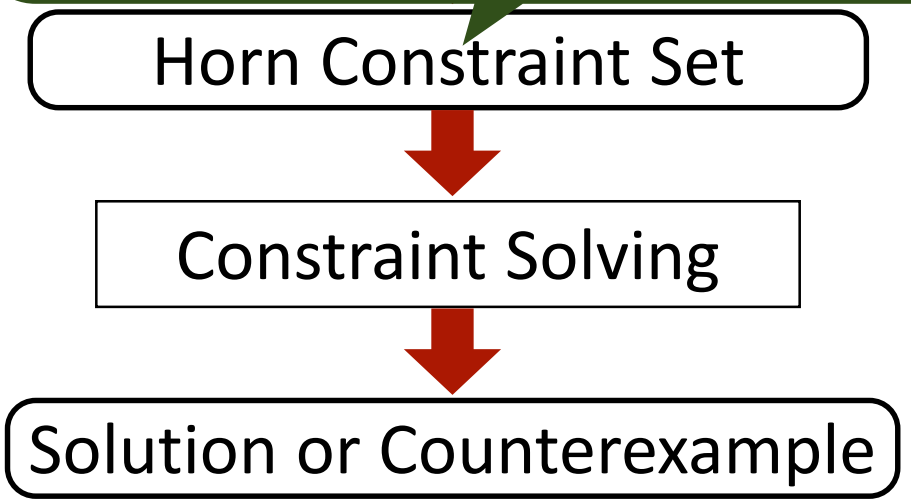
```
(* OCaml *)
let rec mult x y =
  if y = 0 then 0
  else x + mult x (y - 1)
```

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

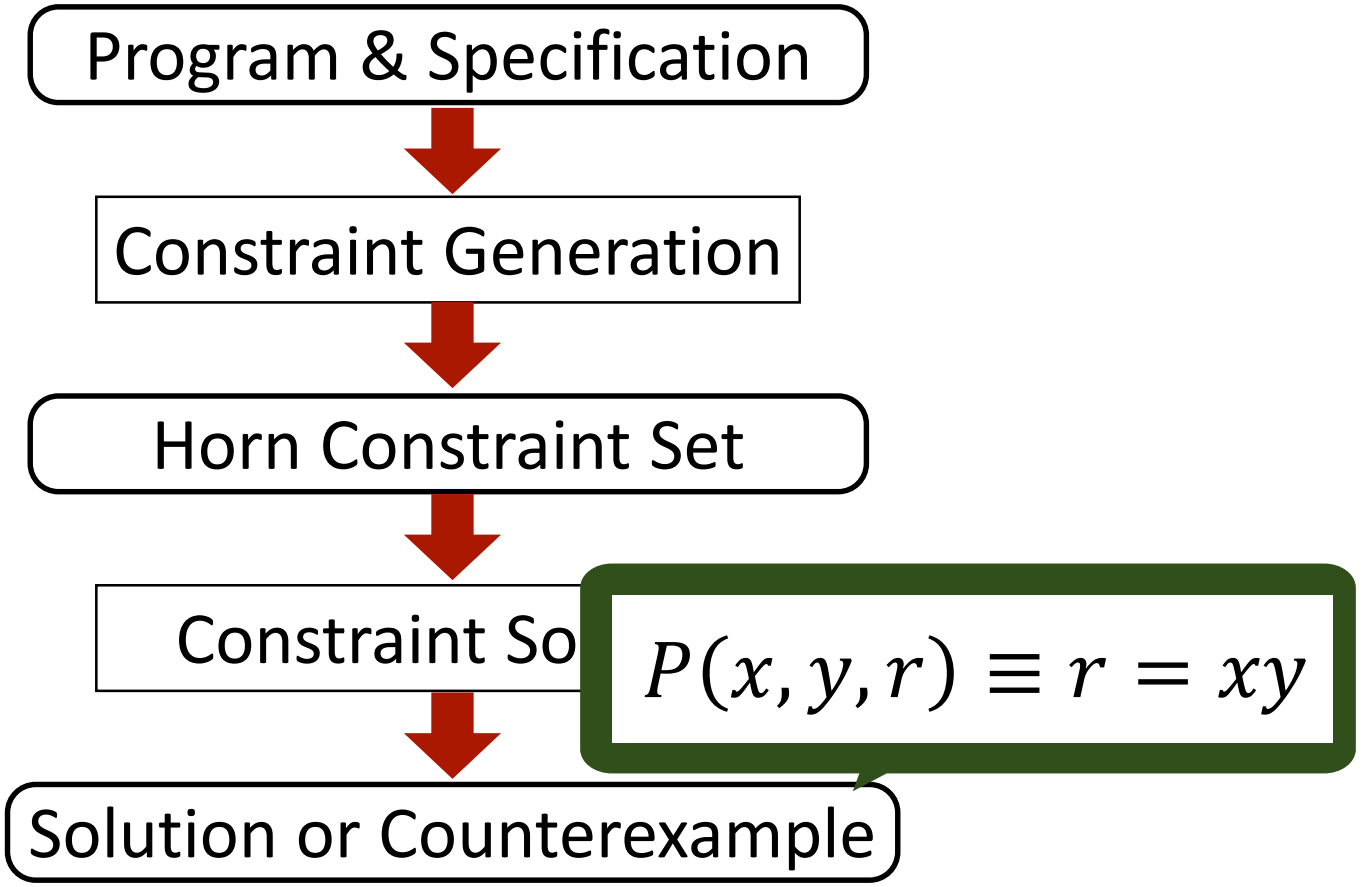
```
/* C */
int mult(int x, int y) {
  int s = 0;
  while(y != 0){
    s += x;
    y--;
  }
  return s;
}
```

Overall Flow of Horn Constraint based Program Verification

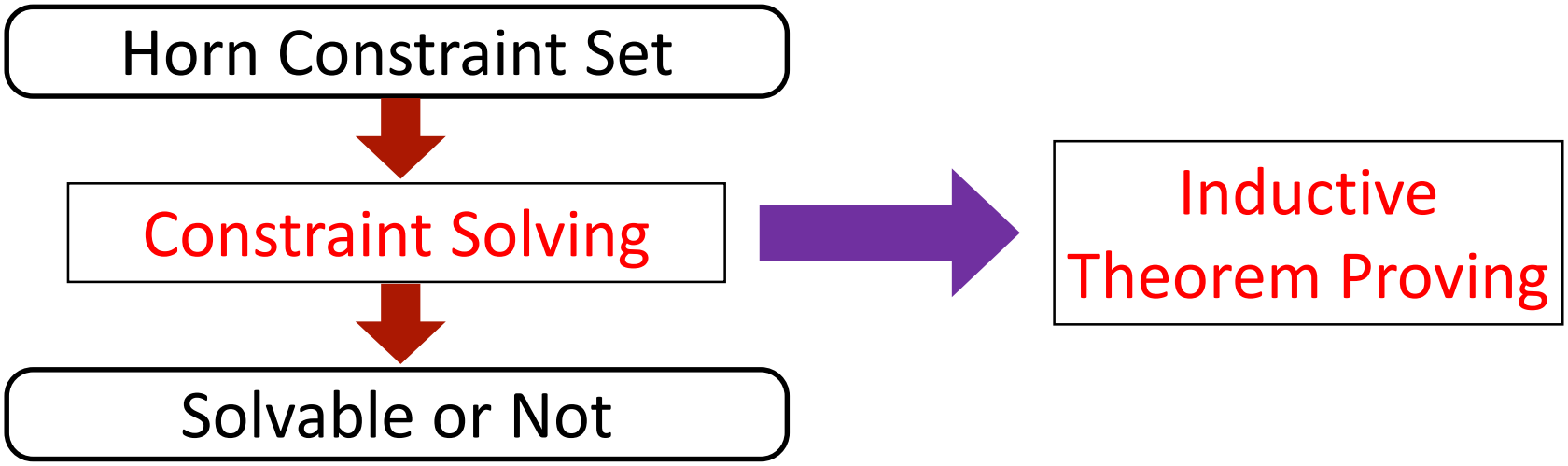
$$P(x, 0, 0)$$
$$P(x, y, x + r) \Leftarrow P(x, y - 1, r) \wedge y \neq 0$$
$$r \geq 0 \Leftarrow P(x, y, r) \wedge x \geq 0 \wedge y \geq 0$$



Overall Flow of Horn Constraint based Program Verification

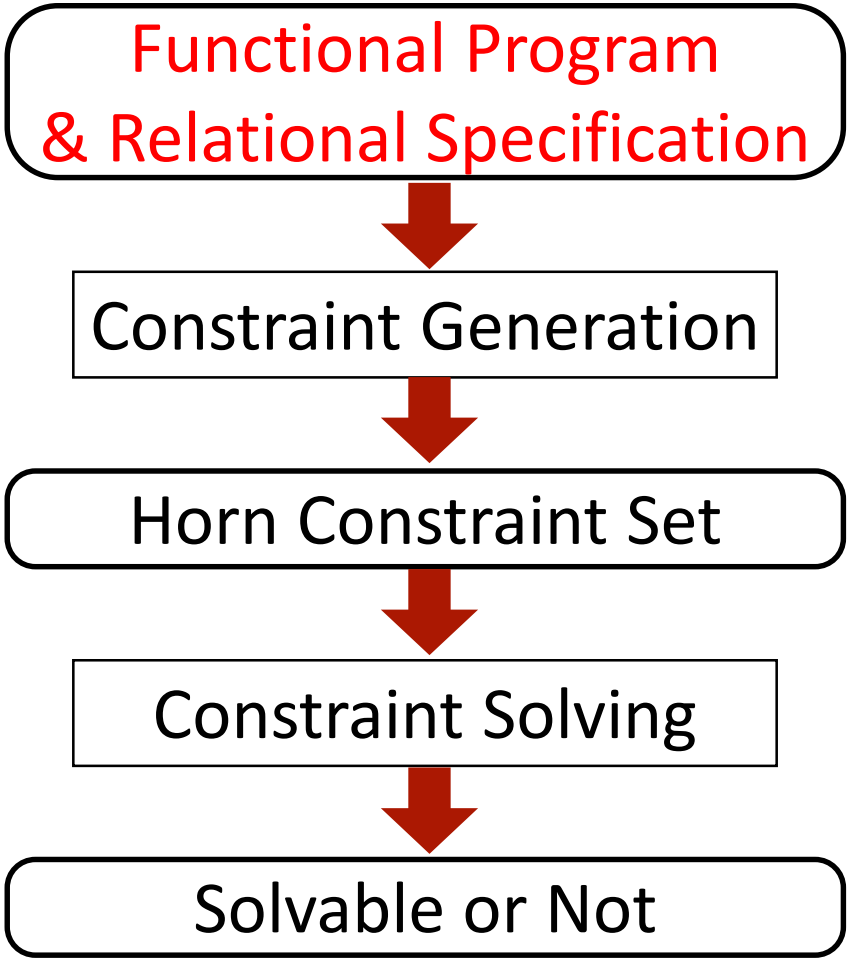


This Work



Enable us to solve Horn constraints reduced from verification problems of ***relational specifications*** of higher-order functional programs with ADTs

Overall Flow of Horn Constraint based Program Verification



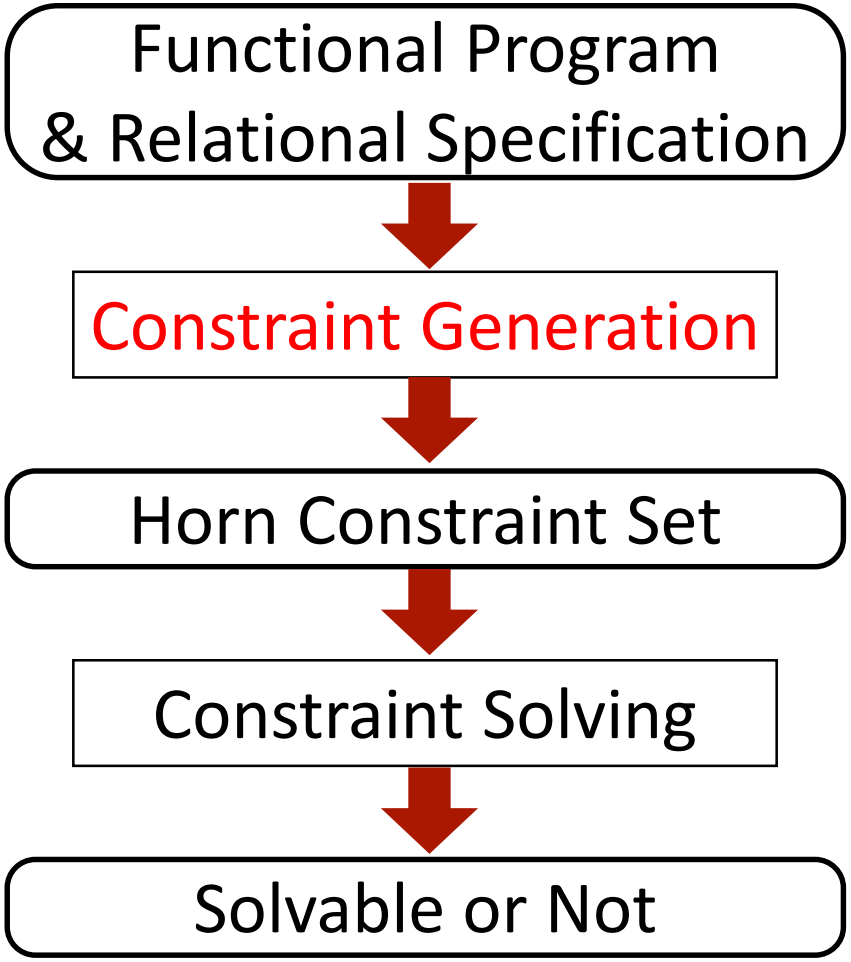
Example: Functional Program and Relational Specification

```
(* 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)

(* relational specification of multiple fun. calls *)
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.+ '08,'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)
```

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

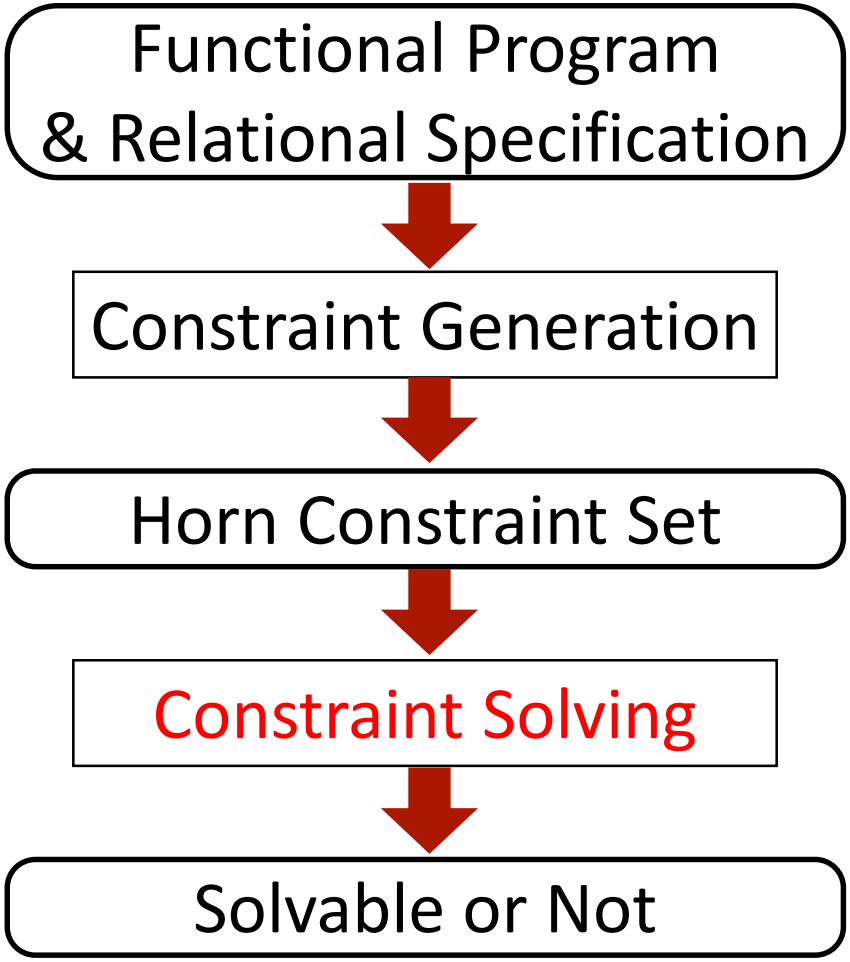
$$P(x, y, x + r) \Leftarrow P(x, y - 1, r) \wedge y \neq 0$$

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

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

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

Overall Flow of Horn Constraint based Program Verification



Horn Constraint Solving

- Check the existence of a *solution* (i.e., a substitution for predicate variables that validates all the Horn clauses)
  - If a solution exists, program satisfies spec.
  - There possibly exist multiple solutions

Example: Solutions

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

QF−LIA

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

QF−NIA

Nonlinear

Previous Methods for Solving Horn Clauses [U.+ '09, Terauchi '10, Gupta+ '11, …]

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

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

OK Sol. 1: $P(x,y,r) \equiv x \geq 0 \wedge y \geq 0 \Rightarrow r \geq 0$ QF−LIA

NG Sol. 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

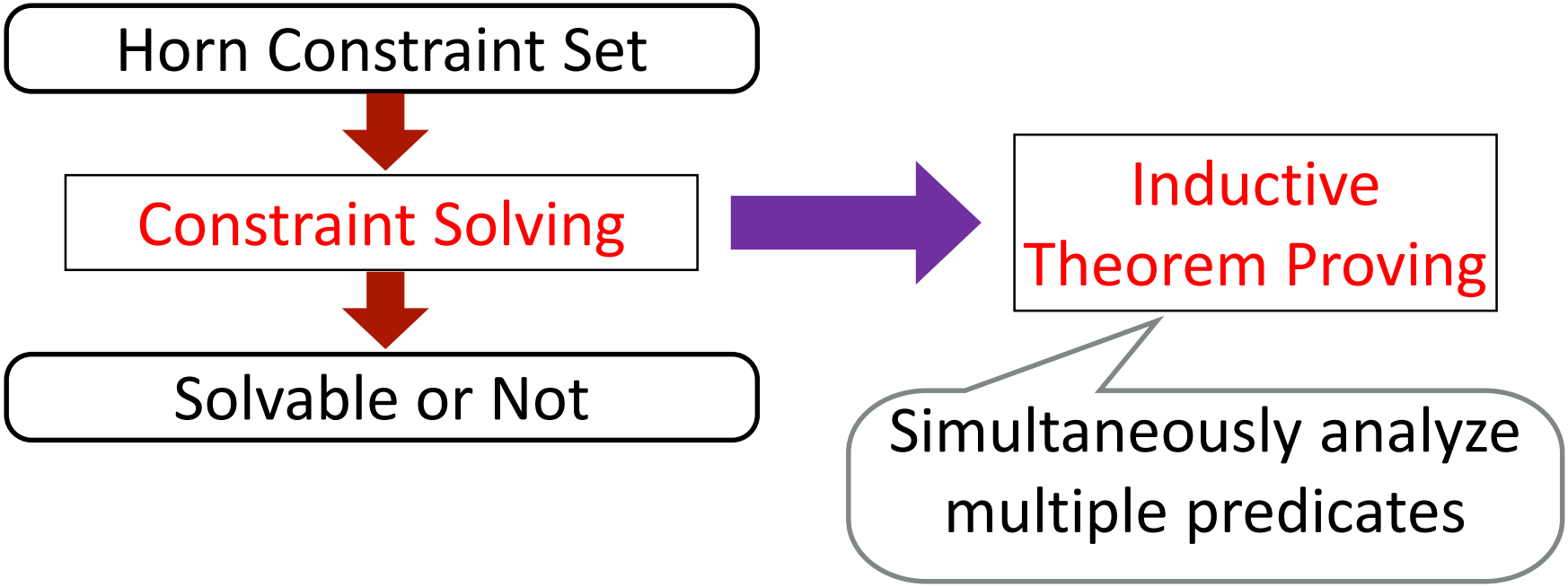
Analyzed separately from P

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

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

Our Constraint Solving Method

Reduce Horn constraint solving to inductive theorem proving



Our Constraint Solving Method

Reduce Horn 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 goals by induction on derivation of pred. apps.

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

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

Induction on Derivation

$$(\forall D.\left(\forall D'.D' \prec D \Rightarrow \psi(D')\right) \Rightarrow \psi(D))$$
$$\Downarrow$$
$$\forall D.\psi(D)$$

$$\dfrac{\dfrac{\overline{D_4} \quad \vdots \quad \overline{D_2}}{\vdots} \quad \cdots \quad \dfrac{\vdots \quad \overline{D_1}}{\vdots} \quad \cdots \quad \dfrac{\vdots \quad \overline{D_3}}{\vdots}}{D}$$

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

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

Premises

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

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

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

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

Add an induction hypothesis:

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

Induct

Case

Case analysis on the last rule used

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

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

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

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

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

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

Case analysis on the last rule used

Case

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

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

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

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

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

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

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

Validity checking

Valid

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

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

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

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

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

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

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

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

Valid

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

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

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

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

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

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

Case

Case analysis on the last rule used

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

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

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

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

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

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

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

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

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

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

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

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

Valid

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

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

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

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

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

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

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

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

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

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

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

QED

Automating Induction

- Use an off-the-shelf SMT solver
  - Provide powerful and efficient reasoning about data structures such as integers, reals, ADTs, arrays, …
- Adopt the following rule application strategy:
  - Repeatedly apply INDHYP until no new premises are added
  - Select some pred. app. $P(\tilde{t})$ in a fair manner and apply INDUCT
  - Apply CASE whenever INDUCT is applied
  - Apply VALID whenever a new premise is added

Implementation

- As a backend solver for Refinement Caml
- Use Z3 as the underlying SMT solver
- Support:
  - User specified lemmas
  - Refutation

Experiments on IsaPlanner benchmark set

- Consist of 85 relational verification problems of total recursive functions on inductively defined ADTs

Inductive Theorem Prover	#Successfully Proved
Refinement Caml	68
Zeno	82
ACL2s	74
CVC4 (dti encoding)	80
CVC4 (dtt encoding)	64
IsaPlanner	47
Dafny	45

Experiments on Various Programs that Use Advanced Language Features

- Integer functions with complex recursion:
  - Ackermann, McCarthy91
- Higher-order functions:
  - fold_left, fold_right, apply, repeat, find, ...
- Exceptions
  - find
- Non-terminating functions:
  - mult, sum, …
- Imperative procedures

Related Work: Inductive Theorem Provers

- Boyer-Moore theorem proving [Kaufmann+ '00]
  - ACL2s, …
- Rewriting induction [Reddy '90, …]
  - SPIKE, …
- SMT-based Induction [Suter+ '11, Leino '12, Reynolds+ '15, …]
  - Dafny, CVC4, Leon, …

Support only pure terminating functions on inductively-defined data structures

Conclusion

- Proposed an automated verification method combining Horn constraint solving and inductive theorem proving
  - Enable relational verification of higher-order functional programs with ADTs
- Future work:
  - Automatic lemma discovery
  - Automatic coinductive proof
  - Verification of specifications expressed by using inductively-defined predicates