Refinement Type Inference via Horn Constraint Optimization

Kodai Hashimoto and Hiroshi Unno
(University of Tsukuba, Japan)

Our Goal: Path-Sensitive Program Analysis of Higher-order Non-det. Functional Programs

- Precondition inference
- Bug finding
- (Conditional) termination analysis
- Non-termination analysis
- Modular verification
- …

Refinement type optimization

a generalization of ordinary refinement type inference

Refinement Type Inference

Program

```
let rec sum x =
  ...
```

Infer →

Refinement Types

sum : $(x\!: \mathrm{int}) \rightarrow \{y \mid y \geq 0\}$,

...

Refinement types can precisely express **program behaviors**

- $\{x : \mathrm{int} \mid x \geq 0\}$
  Non-negative integers

FOL predicates (e.g., QFLIA)

- $(x : \mathrm{int}) \rightarrow \{y : \mathrm{int} \mid y \geq x\}$
  Functions that take an integer x and return an integer y not less than x

A Challenge in Refinement Type Inference

Which refinement type should be inferred?

```
let rec sum x = if x = 0 then 0 else x + sum (x-1)
```

contradiction

$\{x \mid x = -1\} \to \{y \mid \bot\}$ $:>$ $\{x \mid x < 0\} \to \{y \mid \bot\}$

$\vee$ (:>)

$\mathrm{int} \to \{y \mid y \geq 0\}$ incomparable $x < -5\} \to \{y \mid \bot\}$

$\{x \mid x = 0\} \to \{y \mid y = 0\}$ …

The most general types are often not expressible in the underlying logic (e.g., QFLIA)

Existing Refinement Type Inference Tools

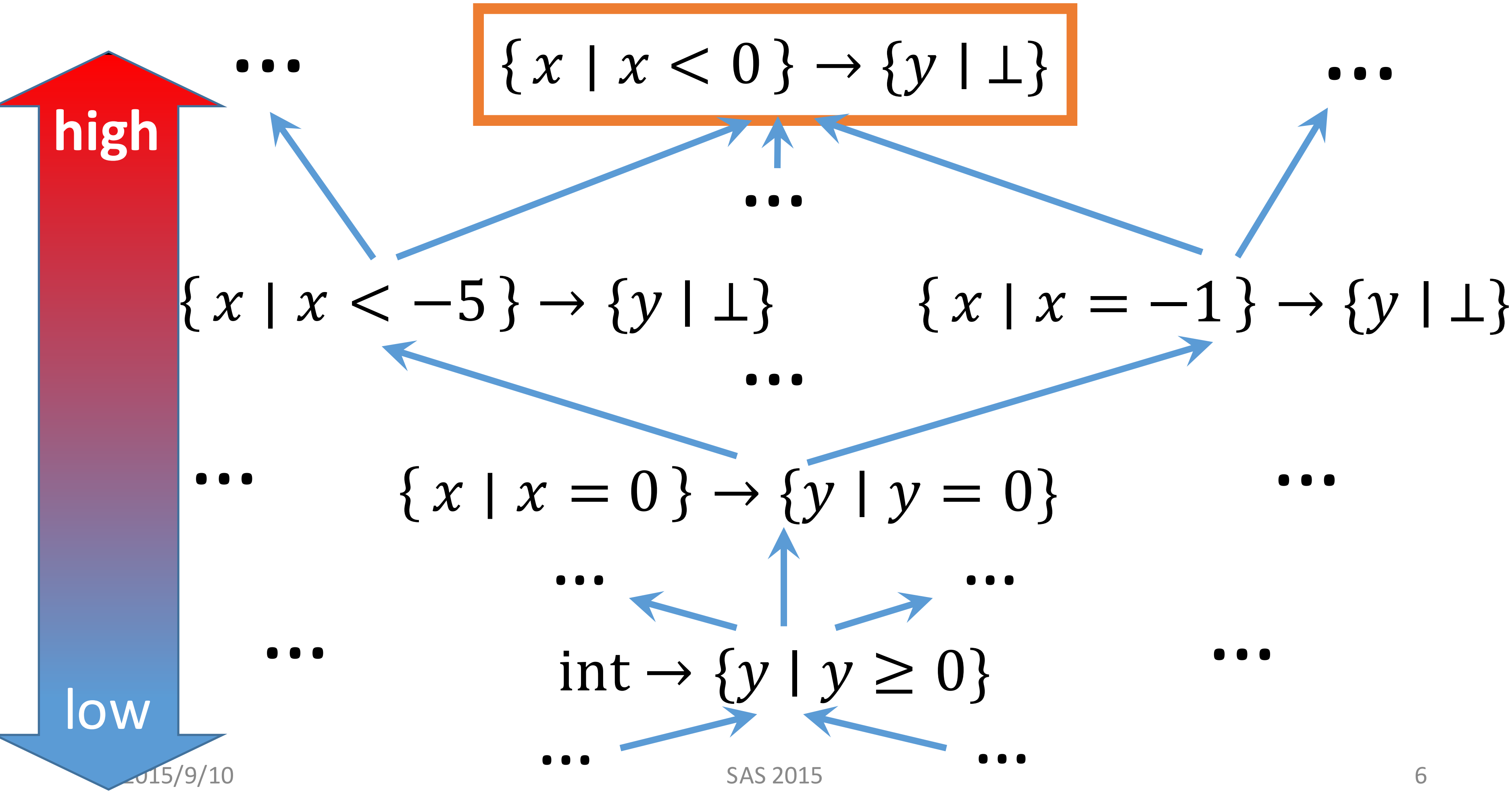
Infer refinement types precise enough to verify a given safety specification

- Refinement Caml [Unno+ ’08, ’09, ’13, ’15]
- Liquid Types [Jhala+ ’08, ’09, …, ’15]
- MoCHi [Kobayashi+ ’11, ’13, ’14, ’15, ’15]
- Depcegar [Terauchi ’10]
- HMC [Jhala+ ’11]
- Popeye [Zhu & Jagannathan ’13]

Inferred types are often too specific to the spec.
→ Limited applications

Our Approach: Refinement Type Optimization

Infer maximally preferred (i.e. **Pareto optimal**) refinement types with respect to **a user-specified preference order**



How to Specify Preference Orders (1/3)

- Refinement type template

Predicate variables

$$(x : \{x \mid \boldsymbol{P}(\boldsymbol{x})\}) \to \{y \mid \boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{y})\}$$

$P(x) \mapsto x < 0$,
$Q(x, y) \mapsto \bot$

$P(x) \mapsto x = 0$,
$Q(x, y) \mapsto y = 0$

$\{x \mid x < 0\} \to \{y \mid \bot\}$

$\{x \mid x = 0\} \to \{y \mid y = 0\}$

How to Specify Preference Orders (2/3)

max/min **optimization constraints**

$\boldsymbol{max}(\boldsymbol{P})$: infer a maximally-**weak** predicate for $\boldsymbol{P}$
$\boldsymbol{min}(\boldsymbol{Q})$: infer a maximally-**strong** predicate for $\boldsymbol{Q}$

Precondition **Postcondition**

sum : $(x : \{x \mid \boldsymbol{P}(\boldsymbol{x})\}) \rightarrow \{y \mid \boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{y})\}$

```
let rec sum x = if x = 0 then 0 else x + sum (x-1)
```

$\boldsymbol{max}(\boldsymbol{P})$
$\boldsymbol{min}(\boldsymbol{Q})$

$\{x \mid x < 0\} \rightarrow \{y \mid \bot\}$ $\quad$ $\mathrm{int} \rightarrow \{y \mid y \geq 0\}$

$\{x \mid x = 0\} \rightarrow \{y \mid y = 0\}$

How to Specify Preference Orders (3/3)

a priority order $\sqsubset$ **on predicate variables**

$\boldsymbol{Q} \sqsubset \boldsymbol{P}$: Q is given higher priority over P

sum : $(x : \{x \mid \mathrm{P}(x)\}) \rightarrow \{y \mid \mathrm{Q}(x, y)\}$

```
let rec sum x = if x = 0 then 0 else x + sum (x-1)
```

$\boldsymbol{max(P)}$

$\boldsymbol{min(Q)}$

$\boldsymbol{Q \sqsubset P}$ $\boldsymbol{P \sqsubset Q}$

$(x : \{ x \mid x < 0 \}) \rightarrow \{y \mid \bot\}$ $(x : \mathrm{int}) \rightarrow \{y \mid y \geq 0\}$

Outline

- Refinement Type Optimization
  - Applications
  - Our Type Optimization Method
- Implementation & Experiments
- Summary

Applications of Refinement Type Optimization

- Non-termination analysis
- Conditional termination analysis
- Precondition inference
- Bug finding
- Modular verification
- …

Non-Termination Analysis

Find a program input that violates the termination property

No return value = **Non-terminating**

$\bot \Leftarrow \boldsymbol{Q(x,y)}$

$\text{sum} : (x : \{x \mid \mathrm{P}(x)\}) \rightarrow \{y \mid \mathrm{Q}(x,y)\}$

```
let rec sum x = if x = 0 then 0 else x + sum (x-1)
```

infer a maximally-weak precondition $\boldsymbol{P}$

$\boldsymbol{max(P)}$

Existing non-termination analysis tool may infer: $\{x \mid x = -1\} \rightarrow \{y \mid \bot\}$

$(x : \{x \mid \boldsymbol{x < 0}\}) \rightarrow \{y \mid \bot\}$

sum never terminates if $\boldsymbol{x < 0}$

Non-Termination Analysis of Non-Deterministic Programs

f : (x: int) → {r | Q(r)} $\bot \Leftarrow Q(r)$

```
let rec f x =
  let n  = read_int() in
  if n = x then f (x+1) else x
```

non-determinism

infer a maximally-weak condition $\boldsymbol{P}$

$\boldsymbol{max(P)}$

n : { n | $\boldsymbol{n = x}$ }

f : (x: int) → {r | ⊥ }, …

f never terminates if the user always inputs same value as an argument $\boldsymbol{x}$

Non-Termination Analysis of Higher-Order Programs

main : $(x : \{x \mid \mathrm{P}(x)\}) \rightarrow \{y \mid \mathrm{Q}(x, y)\}$, … $\bot \Leftarrow Q(x, y)$

```
let rec fix (f:int -> int) x =
  let x’ = f x in
  if x’ = x then x else fix f x’
let to_zero x = if x = 0 then 0 else x - 1
let main x = fix to_zero x
```

infer a maximally-weak precondition $\boldsymbol{P}$

$\boldsymbol{max(P)}$

main never terminates if $x < 0$

main : $(x \colon \{\, \boldsymbol{x} \mid x < 0 \,\}) \rightarrow \{r \mid \bot\}$,

fix : $(f \colon (a : \{a \mid a < 0\}) \rightarrow \{b \mid b < a)\})$
$\rightarrow (x : \{x \mid x < 0)\}) \rightarrow \{y \mid \bot\}$,

to_zero : $(x : \{x \mid x < 0\}) \rightarrow \{y \mid y < x\}$

Applications of Refinement Type Optimization

- Non-termination analysis
- Conditional termination analysis
- Precondition inference
- Bug finding
- Modular verification
- …

Conditional Termination Analysis (1/2)

- Infer a sufficient condition for termination
- Our approach is inspired by a program transformation approach to termination analysis of imperative programs [Gulwani+ '08, '09]

```
let rec sum x = if x = 0 then 0 else x + sum (x-1)
```

the initial value of x

the number of recursive calls

```
let rec sum_t x i c =
  if x = 0 then 0 else x + sum_t (x-1) i (c+1)
```

Conditional Termination Analysis (2/2)

Infer a sufficient condition for termination

$$\exists f. c \leq f(i) \Leftarrow Bnd(i,c). \quad Bnd(i,c) \Leftarrow P(x) \land Inv(x,i,c)$$

sum_t: $(x : \{x \mid \boldsymbol{P(x)}\}) \to (i : \text{int}) \to (c : \{c \mid \boldsymbol{Inv(x,i,c)}\}) \to \text{int}$

```
let rec sum_t x i c =
  if x = 0 then 0 else x + sum_t (x-1) i
```

$Inv(x,i,c) \Leftarrow c = 0 \land i = x$

$\boldsymbol{max(P), min(Bnd)}$

$\boldsymbol{P \sqsubset Bnd}$

sum_t: $(x : \{x \mid \boldsymbol{x \geq 0}\}) \to (i : \text{int}) \to (c : \{c \mid x \leq i \land i = x + c\}) \to \text{int}$

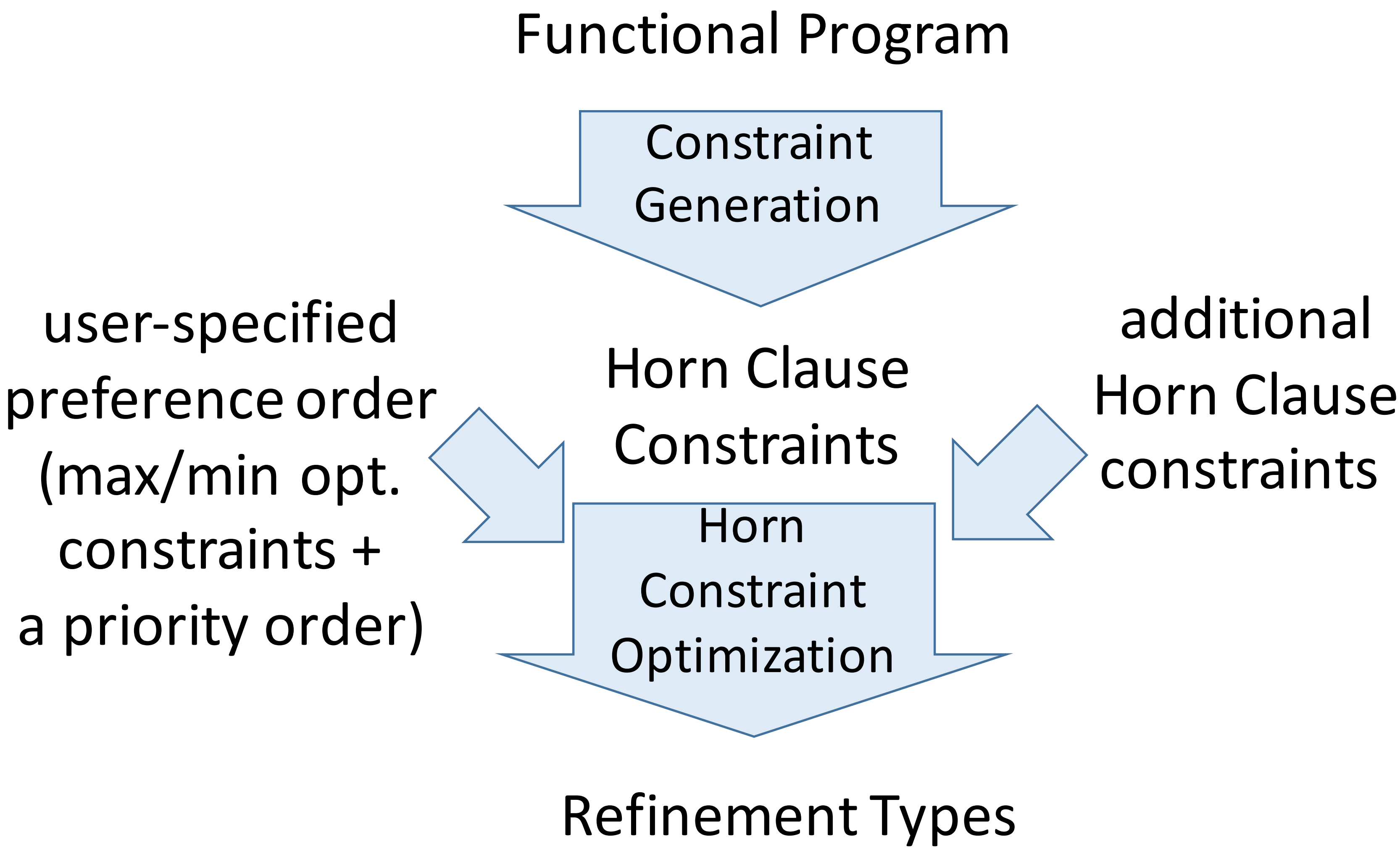
$f(i) = i \quad Bnd(i,c) \mapsto \boldsymbol{c \leq i}$

sum x terminates when $x \geq 0$ because $c \leq i$

Outline

- Refinement Type Optimization
  - Applications
  - Our Type Optimization Method
- Implementation & Experiments
- Summary

Overall Structure



Example: Type Optimization by Our Method

$\text{sum} : (x : \{x \mid \mathrm{P}(x)\}) \rightarrow \{y \mid \bot\}$

```
let rec sum x = if x = 0 then 0 else x + sum (x-1)
```

Constraint Generation

$$H_{sum} = \forall x. \begin{Bmatrix} \bot \Leftarrow P(x) \wedge x = 0, \\ P(x-1) \Leftarrow P(x) \wedge x \neq 0 \end{Bmatrix}$$

Horn Constraint Optimization

$max(P)$

$$(x : \{ x \mid x < 0 \}) \rightarrow \{y \mid \bot\}$$

Example: Horn Constraint Optimization

repeatedly improves a current solution until convergence

H_{sum} → **Solve** ($max(P)$) → $P(x) \mapsto \bot$

Requires any solution for P to be strictly better than $\bot$

Gen. Horn Constraints for improved Sols. → $H' = H_{sum} \cup \left\{ \begin{array}{l} \forall x. (P(x) \Leftarrow \bot), \\ \neg \forall x. (\bot \Leftarrow P(x)) \end{array} \right\}$

Solve → $P(x) \mapsto x < 0$ → Gen. ($max(P)$) → ⋯ → **Solve** → Converge (No Solution)

Horn Constraint Solver *Solve*

- Extended template-based invariant generation techniques [Colon+ '03, Gulwani+ '08] to solve ***existentially-quantified Horn clause constraints***
  - Extend the reach from imperative programs w/o recursion to higher-order non-det. programs
- Any other solver for the class of constraints can be used instead [Unno+ '13, Beyene+ '14, Kuwahara+ '15]

Outline

- Refinement Type Optimization
  - Applications
  - Our Type Optimization Method
- Implementation & Experiments
- Summary

Implementation & Experiments

A refinement type checking and inference tool for OCaml

- Implemented in ***Refinement Caml*** [Unno+ ’08, ’09, …]
  - Z3 [Moura+ ’08] as a backend SMT solver
- Two preliminary experiments:
  - Various program analysis problems for higher-order non-deterministic programs (partly obtained from [Kuwahara+ ’14, Kuwahara+ ’15])
  - Non-termination verification problems for first-order non-deterministic programs (obtained from [Chen+ ’14, Larraz+’14, Kuwahara+ ’15, …])

Results of the Various Program Analysis Problems (excerpt)

Program	Application	#Iter.	Time (sec)	Opt.
foldr_nonterm [Kuwahara+ ’15]	Non-termination	4	**8.04**	✔
fixpoint_nonterm [Kuwahara+ ’15]	Non-termination	2	**0.30**	✔
indirectHO_e [Kuwahara+ ’15]	Non-termination	2	**0.31**	✔
zip [Kuwahara+ ’14]	Conditional Termination Analysis	4	**12.24**	
sum	Conditional Termination Analysis	6	**12.02**	✔
append [Kuwahara+ ’14]	Conditional Termination Analysis	11	**10.66**	✔

Environment: Intel Core i7-3770 (3.40GHz), 16 GB of RAM

Results of the First-Order Non-Termination Verification Problems

	Verified	Time Out	Other
Our tool	**41**	27	13
CppInv [Larraz+ ’14]	**70**	6	5
T2-TACAS [Chen+ ’14]	**51**	0	30
MoCHi [Kuwahara+ ’15]	**48**	26	7
TNT [Emmes+ ’12]	**19**	3	59

Summary

Refinement type optimization problems

- Infer Pareto-optimal refinement types with respect to **a user-specified preference order**
- Has applications to various program analysis problems of higher-order and non-deterministic functional programs

Refinement type optimization method

- Reduction to a Horn constraint optimization problem
- Horn constraint optimization method
  - Repeatedly improve the current solution until convergence

Prototype implementation and preliminary experiments

Thank you!