

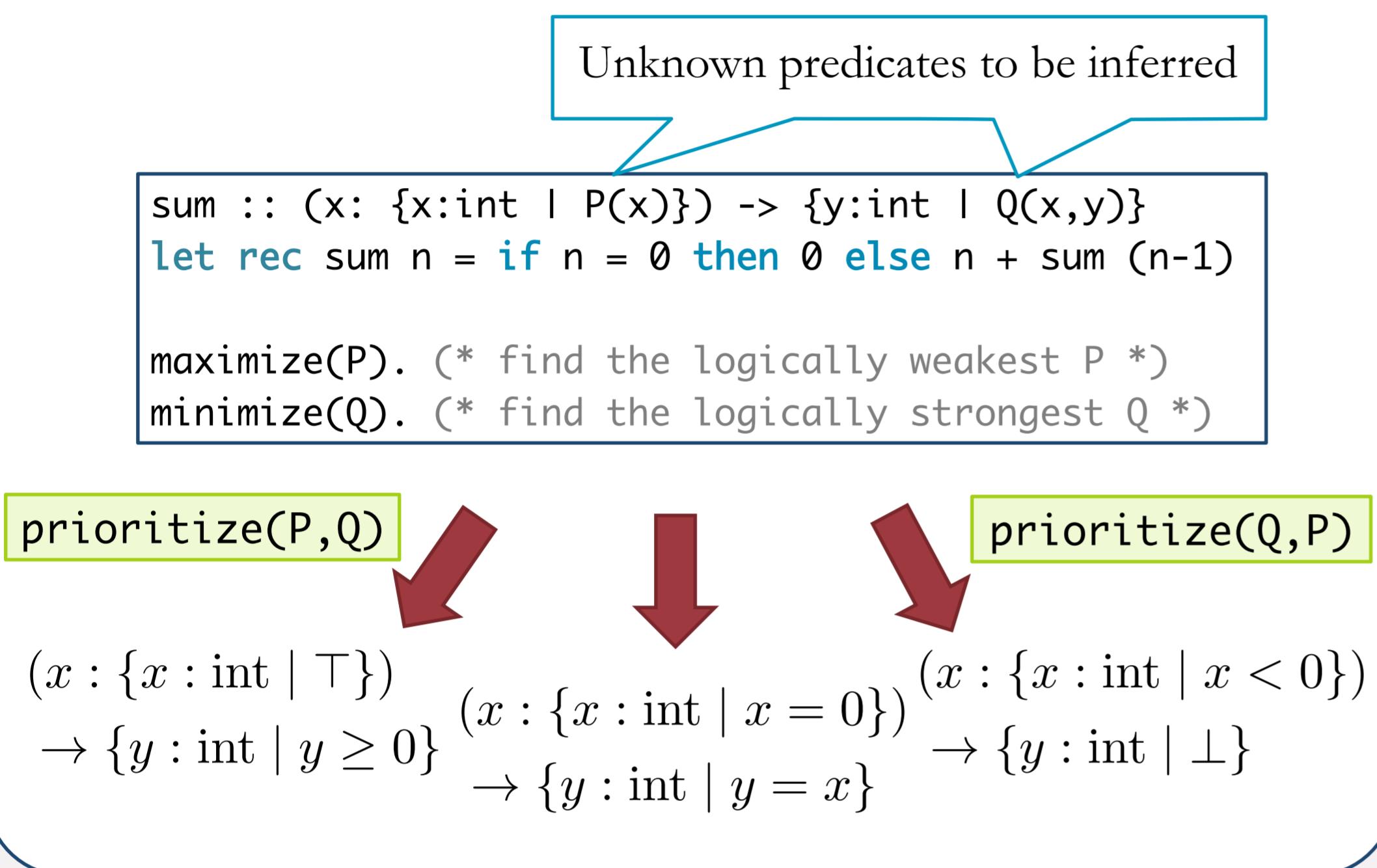
# Refinement Type Inference via Multi-Objective Optimization Subject to Horn Clauses

Kodai Hashimoto and Hiroshi Unno (University of Tsukuba)

{kodai, uhiro}@logic.cs.tsukuba.ac.jp

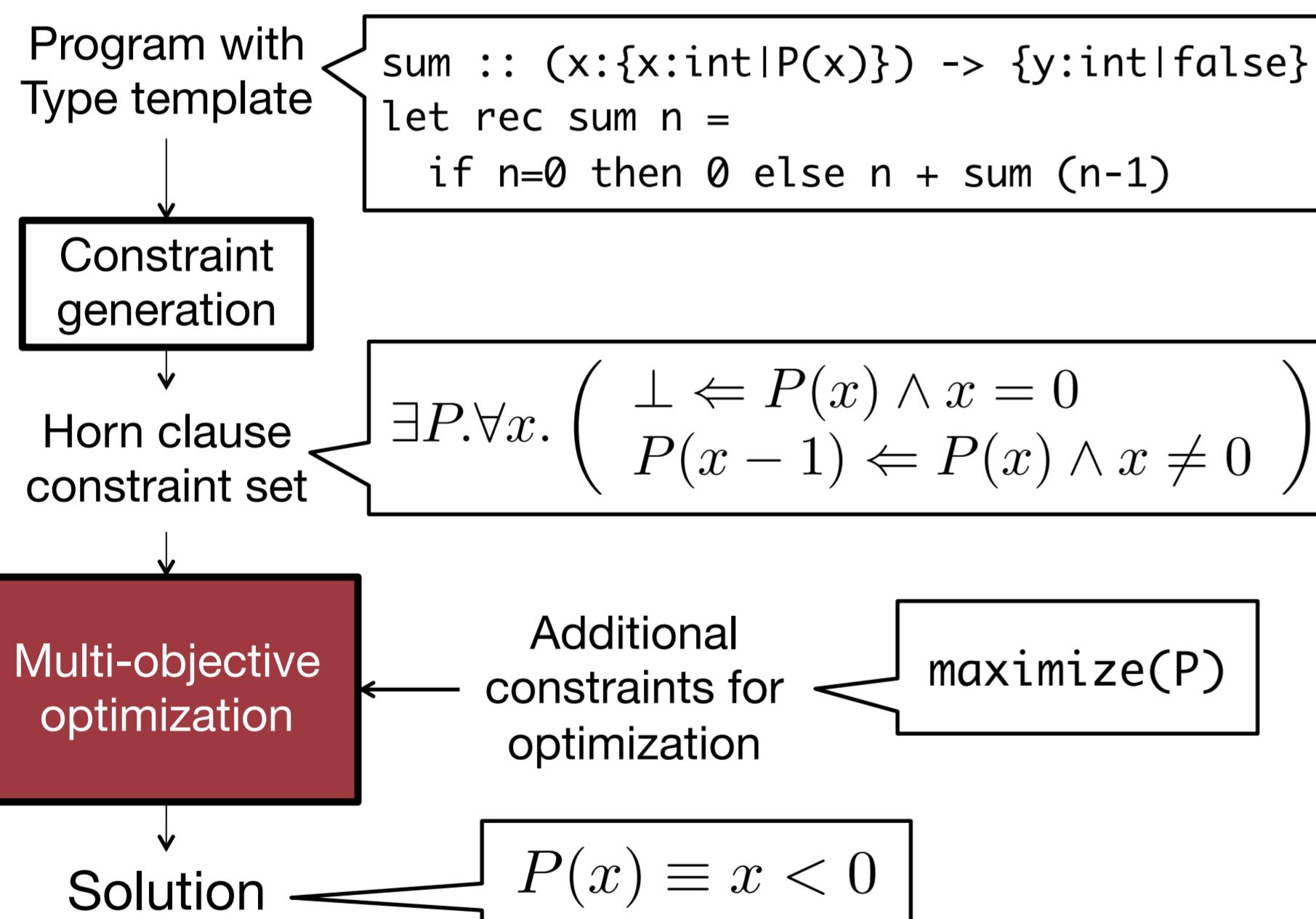
## Our proposal

- We propose a refinement type inference method that allows users to control “the quality” of inferred types.



## Overall structure

(cf. [Unno and Kobayashi 2009])



## Applications

### Precondition inference

Infer the (preferably weakest) precondition that satisfies a given postcondition.

```
sum :: (x: {x:int | P(x)}) -> {y:int | x = y}
let rec sum n = if n<=0 then 0 else n + sum (n-1)

maximize(P).
```

inferred type:

$(x : \{x : \text{int} \mid 0 \leq x \leq 1\}) \rightarrow \{y : \text{int} \mid x = y\}$

### Non-termination analysis

Infer the (preferably weakest) precondition that leads to non-termination.

```
sum :: (x: {x:int | P(x)}) -> {y:int | false}
let rec sum n = if n=0 then 0 else n + sum (n-1)

maximize(P).
```

inferred type:  $(x : \{x : \text{int} \mid x < 0\}) \rightarrow \{y : \text{int} \mid \perp\}$

### Bounds analysis

Infer the upper bound on the number of recursive calls.

```
sum :: (x: {x:int | x>=0}) -> (i: int)
      -> (c : {c:int | P(x,i,c)} -> int
(* i: initial value of x, c: # of recursive calls *)
let rec sum x i c =
  if x = 0 then 0 else x + sum (x-1) i (c+1)

minimize(P).
P(x,i,c) :- x = i && c = 0.

inferred type:  $(x : \{x : \text{int} \mid x \geq 0\}) \rightarrow (i : \text{int})$ 
 $\rightarrow (c : \{c : \text{int} \mid c + x = i \wedge c \geq 0\}) \rightarrow \text{int}$ 
The upper bound of c is i  
because  $x \geq 0 \wedge i = x + c$  is an invariant of sum.
```

## Our optimization algorithm

repeatedly improve an approximate solution until convergence!

We extended [Gulwani et al. 2008] to support:

- Horn clause constraint sets,
- multiple objectives, and
- priority orders

We implemented a prototype type inference system (demo available)

