SAV07 Mini-Project

Variable Range Analysis

Presented by:

Simon Blanchoud

Yuanjian Wang Zufferey

Overview

- Grammar
- Functions
- Computation Steps
  - Flow of computation
  - Translation Example
  - Convergence
- Simulation
- Future improvement

Grammar

- type system = System of (int*range*string*int*equation) list
- type equation =
  - | Intersection of equation * equation
  - | Union of equation * equation
  - | Addition of equation * equation
  - | Substraction of equation * equation
  - | Multiplication of equation * equation
  - | Division of equation * equation
  - | Widening of equation * equation
  - | Narrowing of equation * equation
  - | Range of range
  - | LowerThen of equation
  - | GreaterThen of equation

Grammar

- type range =
  | Index of int
  | Ref of string
  | Is of bool*bound * bound*bool

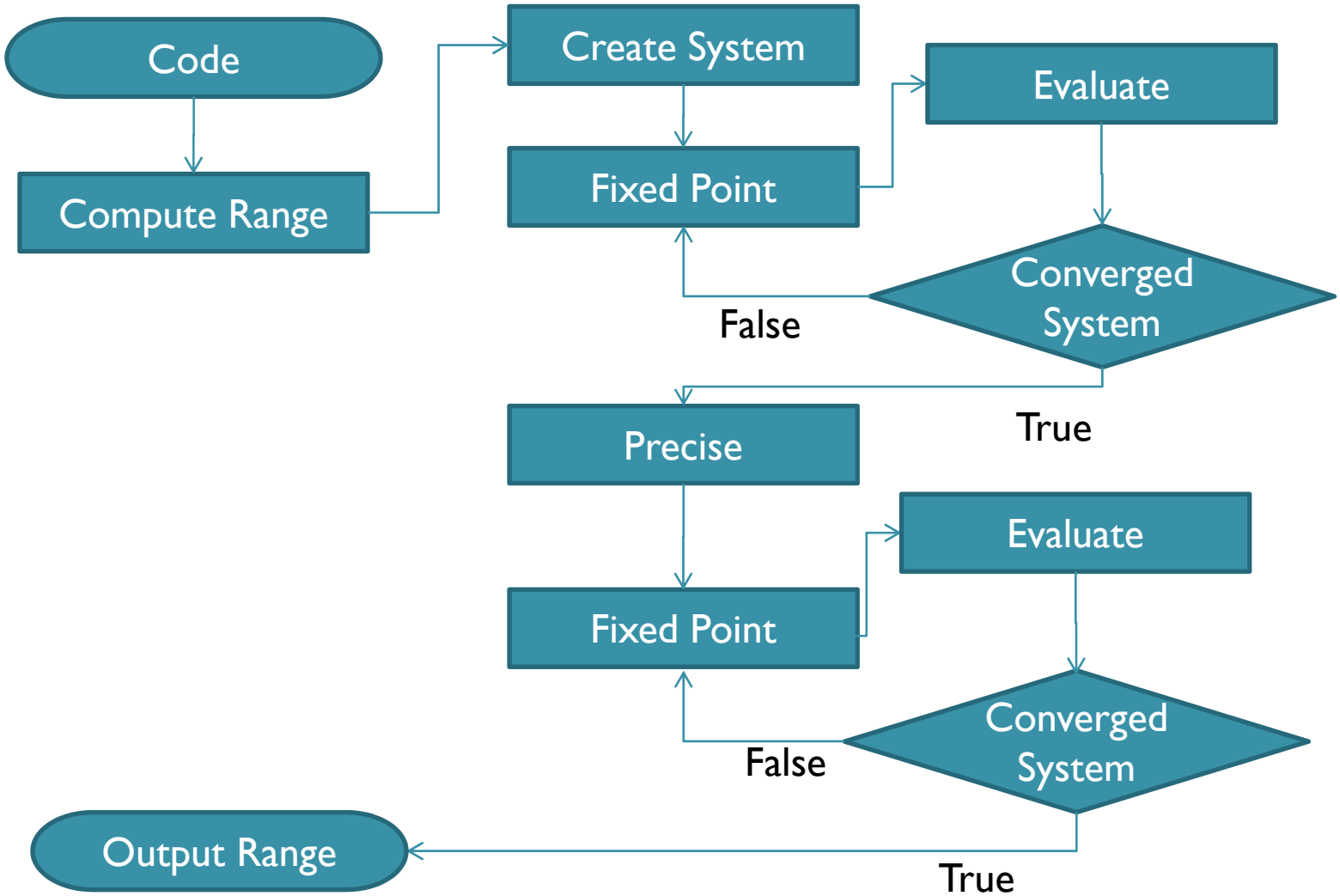
- type bound =
  | Inf
  | InfNeg
  | Empty
  | Val of float

Functions

- Printing
- String list operations
- Search
- Code analysis
- Bound computation
- Range computation
- Converting
- Equations computation

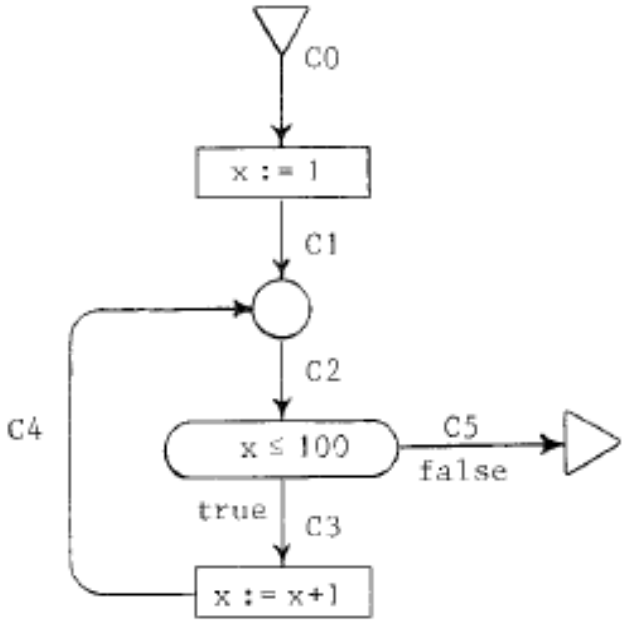
Computation Steps

Flow of Functions



Computation Steps

Translation Example



(0) C0 = [,]
(1) C1 = [1, 1]
(2) C2 = C2 ▽ (C1 ∪ C4)
(3) C3 = C2 ∩ [−∞, 100]
(4) C4 = C3 + [1, 1]
(5) C5 = C2 ∩ [101, +∞]

Computation Steps

Translation Example

```
While(f1, c1) -> let f2 = (to_left_form f1) in
        let f3 = (to_revert_form f2) in
        let s1 = (main_var f2) in
        let l1 = (last s s1 0) in
        let f4 = (func_to_equation f2) in
        let f5 = (func_to_equation f3) in
        let s2_temp = create_system (c1) (add s (0, (Is(true, Empty, Empty, true)), s1, (l1+1),
            Intersection((Range(Index l1)), f4))) in
        let l2 = (last s2_temp s1 0) in
        let s2 = (add s (0, (Is(true, Empty, Empty, true)), s1, (l1+1),Widening((Range(Index(l1+1))),
            (Union((Range(Index l1)), (Range (Index (l2+1)))))))) in
        let s3 = create_system (c1) (add s2 (0, (Is(true, Empty, Empty, true)), s1, (l1+2),
            Intersection((Range(Index (l1+1))), f4))) in
        add s3 (0, (Is(true, Empty, Empty, true)), s1, (l2+2), Intersection((Range(Index (l1+1))), f5))
```

```
(0)   C0 = [ , ]
(1)   C1 = [1, 1]
(2)   C2 = C2 ▽ (C1 ∪ C4)
(3)   C3 = C2 ∩ [-∞, 100]
(4)   C4 = C3 + [1, 1]
(5)   C5 = C2 ∩ [101, +∞]
```

Computation Steps

Convergence

```
let rec evaluate (s : system)(s_past : system) : system =
        match s with
        | System [] -> s
        | System (e1::eq_rest) ->
            match e1 with
              | (c1, r1, v1, i1, eq1) ->
                let sol = (evaluate_equation eq1 s_past s v1) in
                let c1 = if ((equal_range r1 sol)
                        & (converged_predecessors eq1 s_past v1 )) then 1 else 0 in
                adds (System ((c1, sol, v1, i1, eq1)::[]))
                        (evaluate (System eq_rest) (add s_past e1))
```

Simulation

- Demo…

Future improvement

- Multiple variables
- Call of procedures (Functions ?)
- Arrays ?

Thanks for your attentions!

- Questions ?