

**Input:** Function *func* is a function; Argument Bounds *args* of function *func*.

**Output:** The domain of return variable *ret* of function *func*

```
1: Variables
2:   blk.d is the domain set of block blk; blk.d(var) is the domain (lower and
   upper bounds) of variable var in block blk.
3: end Variables
4: procedure IS_REACHABLE(blk) // Check the reachability of block blk
5:   return (Any domain  $\in blk.d == \emptyset$ ) ? false : true
6: end procedure // Return true if blk does not have empty domain
7: // Infer bounds of function func using breath-first or depth-first traversal
8: procedure INFER_BOUNDS(func, args)
9:   cfg = BUILD_CFG(func) // Build control flow graph of function func
10:  EXTRACT_CONSTRAINTS(cfg) // Extract constraints in each block
11:  INIT(func) // Initialise each domain in each block with  $\emptyset$ 
12:  deque.add(cfg.getEntry()) // Put entry to deque as starting block
13:  while deque is NOT empty do
14:    blk = deque.poll() // Retrieve block in breath-first or depth-first order
15:    if blk is a function call then // Bound inference on a function call
16:      callee = GET_CALLED_FUNCTION(blk)
17:      args = GET_ARGUMENT_BOUNDS(bounds, blk)
18:      // Infer the bounds of called function
19:      ret = INFER_BOUNDS(callee, args)
20:      // Add the domain of variable ret as a constraint to block blk
21:      ADD_CONSTRAINT(ret, blk)
22:    end if
23:    // Infer the domains of all variables in block blk
24:    blk.din := blk.d // Store domain set of block blk before inference
25:    blk.d := {}
26:    for each parent block in blk do
27:      for each var  $\in$  parent.vars do
28:        if var is a live variable in parent then
29:          // Propagate domains of live variables from parents to blk
30:          blk.d := blk.d  $\cup$  parent.d(var)
31:        end if
32:      end for
33:    end for // Produce initial value of blk.d from parent blocks
34:    for each constraint  $\in$  blk.constraints do
35:      Apply the bound propagation rules of constraint on blk.d
36:    end for // Produce blk.d domains consistent with all constraints
37:    if blk.d has any change (blk.d  $\neq$  blk.din) then
38:      Add children blocks of blk (except EXIT) to deque
39:    end if // We start inferring the bounds of child blocks
40:  end while // Repeat until the domains of all blocks become stable
41:  // Produce final domains of each variable in function func at EXIT block
42:  exit :=  $\bigcup \{blk.d \bullet (\forall blk : BLOCKS \bullet is\_reachable(blk))\}$ 
43:  return exit.d('ret') // Return the domain of return variable ret
44: end procedure
```