

Simplified Interprocedural Analysis Algorithm for Non-Recursive Programs

17-355/17-665/17-819: Program Analysis (Spring 2022)
Rohan Padhye

```

type Context
  val fn : Function           ▷ the function being called
  val input :  $\sigma$          ▷ input for this set of calls

type Summary                 ▷ the input/output summary for a context
  val input :  $\sigma$ 
  val output :  $\sigma$ 

val results : Map[Context, Summary]           ▷ the analysis results

function ANALYZE(ctx,  $\sigma_{in}$ )
   $\sigma'_{out}$   $\leftarrow$  INTRAPROCEDURAL(ctx,  $\sigma_{in}$ )
  results[ctx]  $\leftarrow$  Summary( $\sigma_{in}$ ,  $\sigma'_{out}$ )
  return  $\sigma'_{out}$ 
end function

function FLOW( $\llbracket n: x := f(y) \rrbracket$ , ctx,  $\sigma_n$ )           ▷ called by intraprocedural analysis
   $\sigma_{in}$   $\leftarrow$  [formal(f)  $\mapsto$   $\sigma_n(y)$ ]         ▷ map f's formal parameter to info on actual from  $\sigma_n$ 
  calleeCtx  $\leftarrow$  GETCTX(f, ctx, n,  $\sigma_{in}$ )
   $\sigma_{out}$   $\leftarrow$  RESULTSFOR(calleeCtx,  $\sigma_{in}$ )
  return  $\sigma_n[x \mapsto \sigma_{out}[result]]$            ▷ update dataflow with the function's result
end function

function RESULTSFOR(ctx,  $\sigma_{in}$ )
  if ctx  $\in$  dom(results) then
    if  $\sigma_{in} \sqsubseteq$  results[ctx].input then
      return results[ctx].output           ▷ existing results are good
    else
      return ANALYZE(ctx, results[ctx].input  $\sqcup$   $\sigma_{in}$ )   ▷ possibly more general input
    end if
  else
    return ANALYZE(ctx,  $\sigma_{in}$ )
  end if
end function

function GETCTX(f, callingCtx, n,  $\sigma_{in}$ )
  return Context(f,  $\sigma_{in}$ )           ▷ constructs a new Context with f and  $\sigma_{in}$ 
end function

```