Simplified Interprocedural Analysis Algorithm for Non-Recursive Programs

17-355/17-665/17-819: Program Analysis (Spring 2021)
Jonathan Aldrich and Rohan Padhye

This simplified context-sensitive interprocedural analysis algorithm is capable of analyzing programs that do not contain recursion.

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

function FLOW($[\![n\text{: } x := f(y)]\!], ctx, \sigma_n$) ▷ called by intraprocedural analysis
  $\sigma_{in} \leftarrow [formal(f) \mapsto \sigma_n(y)]$ ▷ map f's formal parameter to info on actual from σ_n
  $calleeCtx \leftarrow$ GETCTX(f, ctx, n, σ_{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, σ_{in})
  if $ctx \in \mathrm{dom}(results)$ **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, σ_{in})
  end if
end function

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