Unspecified Veriopt Theory

July 3, 2021

Contents

theory *SemanticsSnippets*
  imports
    Optimizations.CanonicalizationProofs
begin

notation (*latex*)
  kind (-⟪-⟫)

syntax (*spaced-type-def* **output**)
  -constrain :: *logic* => *type* => *logic* (- :: - [*4*, *0*] *3*)

is-BinaryArithmeticNode :: *IRNode* ⇒ *bool*

inputs-of :: *IRNode* ⇒ *nat list*
inputs-of (*ConstantNode const*) = []
inputs-of (*ParameterNode index*) = []
inputs-of (*ValuePhiNode nid values merge*) = *merge* · *values*
inputs-of (*AddNode x y*) = [*x*, *y*]
inputs-of (*IfNode condition trueSuccessor falseSuccessor*) = [*condition*]

typedef *IRGraph* = {*g* :: *ID* ⇀ *IRNode* . *finite* (*dom g*)}

fun *ids-fake* :: (*ID* ⇀ *IRNode*) ⇒ *ID set* **where**
  ids-fake g = {*nid* ∈ *dom g* . *g nid* ≠ (*Some NoNode*)}

fun *kind-fake* :: (*ID* ⇀ *IRNode*) ⇒ (*ID* ⇒ *IRNode*) **where**
  kind-fake g = (λ*nid*. (*case g nid of None* ⇒ *NoNode* | *Some v* ⇒ *v*))

$ids\text{-}fake :: (nat \Rightarrow IRNode\ option) \Rightarrow nat\ set$
$ids\text{-}fake\ g = \{nid \in dom\ g \mid g\ nid \neq Some\ NoNode\}$

$kind\text{-}fake :: (nat \Rightarrow IRNode\ option) \Rightarrow nat \Rightarrow IRNode$
$kind\text{-}fake\ g = (\lambda nid.\ \mathsf{case}\ g\ nid\ \mathsf{of}\ None \Rightarrow NoNode \mid Some\ v \Rightarrow v)$

$inputs :: IRGraph \Rightarrow nat \Rightarrow nat\ set$
$inputs\ g\ nid = set\ (inputs\text{-}of\ g\langle\!\langle nid \rangle\!\rangle)$

$succ :: IRGraph \Rightarrow nat \Rightarrow nat\ set$
$succ\ g\ nid = set\ (successors\text{-}of\ g\langle\!\langle nid \rangle\!\rangle)$

$input\text{-}edges :: IRGraph \Rightarrow (nat \times nat)\ set$
$input\text{-}edges\ g = (\bigcup_{i \in ids\ g} \{(i, j) \mid j \in inputs\ g\ i\})$

$usages :: IRGraph \Rightarrow nat \Rightarrow nat\ set$
$usages\ g\ nid = \{j \in ids\ g \mid (j, nid) \in input\text{-}edges\ g\}$

$successor\text{-}edges :: IRGraph \Rightarrow (nat \times nat)\ set$
$successor\text{-}edges\ g = (\bigcup_{i \in ids\ g} \{(i, j) \mid j \in succ\ g\ i\})$

$predecessors :: IRGraph \Rightarrow nat \Rightarrow nat\ set$
$predecessors\ g\ nid = \{j \in ids\ g \mid (j, nid) \in successor\text{-}edges\ g\}$

$wf\text{-}start\ g =$
$(0 \in ids\ g \wedge is\text{-}StartNode\ g\langle\!\langle 0 \rangle\!\rangle)$

$wf\text{-}closed\ g =$
$(\forall\, n \in ids\ g.$
$\quad inputs\ g\ n \subseteq ids\ g \wedge$
$\quad succ\ g\ n \subseteq ids\ g \wedge g\langle\!\langle n \rangle\!\rangle \neq NoNode)$

$wf\text{-}phis\ g =$
$(\forall\, n{\in}ids\ g.$
$\quad is\text{-}PhiNode\ g\langle\!\langle n\rangle\!\rangle \longrightarrow$
$\quad |ir\text{-}values\ g\langle\!\langle n\rangle\!\rangle| =$
$\quad |ir\text{-}ends\ g\langle\!\langle ir\text{-}merge\ g\langle\!\langle n\rangle\!\rangle\rangle\!\rangle|)$

$wf\text{-}ends\ g =$
$(\forall\, n{\in}ids\ g.$
$\quad is\text{-}AbstractEndNode\ g\langle\!\langle n\rangle\!\rangle \longrightarrow$
$\quad 0 < |usages\ g\ n|)$

$wf\text{-}graph :: IRGraph \Rightarrow bool$
$wf\text{-}graph\ g = (wf\text{-}start\ g \wedge wf\text{-}closed\ g \wedge wf\text{-}phis\ g \wedge wf\text{-}ends\ g)$

type-synonym $Signature = string$

type-synonym $Program = Signature \rightharpoonup IRGraph$

print-antiquotations

type-synonym $Heap = objref \Rightarrow string \Rightarrow Value$
type-synonym $Free = nat$
type-synonym $DynamicHeap = Heap \times Free$

$h\text{-}load\text{-}field :: objref \Rightarrow string \Rightarrow DynamicHeap \Rightarrow Value$
$h\text{-}load\text{-}field\ r\ f\ (h,\ n) = h\ r\ f$

$h\text{-}store\text{-}field :: objref \Rightarrow string \Rightarrow Value \Rightarrow DynamicHeap \Rightarrow DynamicHeap$
$h\text{-}store\text{-}field\ r\ f\ v\ (h,\ n) = (h(r := (h\ r)(f := v)),\ n)$

$h\text{-}new\text{-}inst :: DynamicHeap \Rightarrow (DynamicHeap \times Value)$
$h\text{-}new\text{-}inst\ (h,\ n) = ((h,\ n + 1),\ ObjRef\ (Some\ n))$

eval:const eval:add eval:param eval:phi eval:invoke eval:invoke eval:load

$$[g,\ m,\ p] \vdash [] \longmapsto []$$

$$\frac{[g,\ m,\ p] \vdash g\langle\!\langle nid \rangle\!\rangle \mapsto v \qquad [g,\ m,\ p] \vdash xs \longmapsto vs}{[g,\ m,\ p] \vdash nid \cdot xs \longmapsto v \cdot vs}$$

step:seq step:if step:end step:newinst step:load step:store

top:lift top:invoke top:return top:unwind

$$\frac{\begin{array}{c} g\langle\!\langle x \rangle\!\rangle = ConstantNode\ c\text{-}1 \\ g\langle\!\langle y \rangle\!\rangle = ConstantNode\ c\text{-}2 \qquad val = intval\text{-}add\ c\text{-}1\ c\text{-}2 \end{array}}{CanonicalizeAdd\ g\ (AddNode\ x\ y)\ (ConstantNode\ val)}$$

$$\frac{g\langle\!\langle x \rangle\!\rangle = ConstantNode\ c\text{-}1 \qquad \neg\ is\text{-}ConstantNode\ g\langle\!\langle y \rangle\!\rangle \qquad c\text{-}1 = IntVal32\ 0}{CanonicalizeAdd\ g\ (AddNode\ x\ y)\ (RefNode\ y)}$$

$$\frac{\neg\ is\text{-}ConstantNode\ g\langle\!\langle x \rangle\!\rangle \qquad g\langle\!\langle y \rangle\!\rangle = ConstantNode\ c\text{-}2 \qquad c\text{-}2 = IntVal32\ 0}{CanonicalizeAdd\ g\ (AddNode\ x\ y)\ (RefNode\ x)}$$

$[\![CanonicalizeAdd\ g\ before\ after;\ wf\text{-}graph\ g \wedge wf\text{-}stamps\ g;\ [g,\ m,\ p] \vdash before \mapsto IntVal32\ res;\ [g,\ m,\ p] \vdash after \mapsto IntVal32\ res']\!] \Longrightarrow res = res'$

$$\frac{g,\ p \vdash (nid,\ m,\ h) \rightarrow (nid',\ m,\ h)}{g\ m\ p\ h \vdash nid \rightsquigarrow nid'}$$

$$\frac{g,\ p \vdash (nid,\ m,\ h) \rightarrow (nid'',\ m,\ h) \qquad g\ m\ p\ h \vdash nid'' \rightsquigarrow nid'}{g\ m\ p\ h \vdash nid \rightsquigarrow nid'}$$

$$\frac{g\langle\!|cond|\!\rangle = ConstantNode\ condv \qquad val\text{-}to\text{-}bool\ condv}{CanonicalizeIf\ g\ (IfNode\ cond\ tb\ fb)\ (RefNode\ tb)}$$

$$\frac{g\langle\!|cond|\!\rangle = ConstantNode\ condv \qquad \neg\ val\text{-}to\text{-}bool\ condv}{CanonicalizeIf\ g\ (IfNode\ cond\ tb\ fb)\ (RefNode\ fb)}$$

$$\frac{\neg\ is\text{-}ConstantNode\ g\langle\!|cond|\!\rangle \qquad tb = fb}{CanonicalizeIf\ g\ (IfNode\ cond\ tb\ fb)\ (RefNode\ tb)}$$

definition *replace-node-fake* :: *ID* ⇒ *IRNode* ⇒ *IRGraph* ⇒ *IRGraph* **where**
  replace-node-fake nid node g = *replace-node nid* (*node*,*default-stamp*) *g*
lemma *CanonicalizeIfProof-fake*:
  fixes *m*::*MapState* **and** *h*::*RefFieldHeap*
  assumes *kind g nid* = *before*
  assumes *CanonicalizeIf g before after*
  assumes *g'* = *replace-node-fake nid after g*
  assumes *g*, *p* ⊢ (*nid*, *m*, *h*) → (*nid'*, *m*, *h*)
  shows *nid* | *g* ∼ *g'*
 by (*metis CanonicalizeIfProof assms*(*1*) *assms*(*2*) *assms*(*3*) *assms*(*4*) *replace-node-fake-def*)

$[\![g\langle\!|nid|\!\rangle = before;\ CanonicalizeIf\ g\ before\ after;$
$g' = replace\text{-}node\text{-}fake\ nid\ after\ g;\ g,\ p \vdash (nid,\ m,\ h) \rightarrow (nid',\ m,\ h)]\!]$
$\Longrightarrow nid \mid g \sim g'$

notation (*latex* **output**)
  filtered-inputs (*inputs*$^{\text{-}\langle\!\langle\text{-}\rangle\!\rangle}$ _)
notation (*latex* **output**)
  filtered-successors (*succ*$^{\text{-}\langle\!\langle\text{-}\rangle\!\rangle}$ _)
notation (*latex* **output**)
  filtered-usages (*usages*$^{\text{-}\langle\!\langle\text{-}\rangle\!\rangle}$ _)

$inputs^{g}\langle\!\langle^{nid}\rangle\!\rangle_{f}$

notation (*latex* **output**)
  Pure.dummy-pattern (−)

end