Unspecified Veriopt Theory

April 24, 2021

Contents

```
theory ATVA2021
  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 = (λnid. (case g nid of None ⇒ NoNode | Some v ⇒ v))
```

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

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

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

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

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

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

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

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

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

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

wf-phis g =
(∀ *n*∈*ids g*.
    is-PhiNode g⟪*n*⟫ ⟶
    |*ir-values g*⟪*n*⟫| =
    |*ir-ends g*⟪*ir-merge g*⟪*n*⟫⟫|)

wf-ends g =
(∀ *n*∈*ids g*.
    is-AbstractEndNode g⟪*n*⟫ ⟶
    0 < |*usages g n*|)

wf-graph :: *IRGraph* ⇒ *bool*
wf-graph g = (*wf-start g* ∧ *wf-closed g* ∧ *wf-phis g* ∧ *wf-ends g*)

type-synonym *Signature* = *string*

type-synonym *Program* = *Signature* ⇀ *IRGraph*

print-antiquotations

type-synonym *Heap* = *string* ⇒ *objref* ⇒ *Value*
type-synonym *Free* = *nat*
type-synonym *DynamicHeap* = *Heap* × *Free*

h-load-field :: *string* ⇒ *objref* ⇒ *DynamicHeap* ⇒ *Value*
h-load-field f r (*h*, *n*) = *h f r*

h-store-field :: *string* ⇒ *objref* ⇒ *Value* ⇒ *DynamicHeap* ⇒ *DynamicHeap*
h-store-field f r v (*h*, *n*) = (*h*(*f* := (*h f*)(*r* := *v*)), *n*)

h-new-inst :: *DynamicHeap* ⇒ (*DynamicHeap* × *Value*)
h-new-inst (*h*, *n*) = ((*h*, *n* + *1*), *ObjRef* (*Some n*))

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

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

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

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

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

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

$$\frac{g\langle\!\langle x\rangle\!\rangle = \mathit{ConstantNode}\ \mathit{c\text{-}1} \qquad \neg\ \mathit{is\text{-}ConstantNode}\ g\langle\!\langle y\rangle\!\rangle \qquad \mathit{c\text{-}1} = \mathit{IntVal}\ 32\ 0}{\mathit{CanonicalizeAdd}\ g\ (\mathit{AddNode}\ x\ y)\ (\mathit{RefNode}\ y)}$$

$$\frac{\neg\ \mathit{is\text{-}ConstantNode}\ g\langle\!\langle x\rangle\!\rangle \qquad g\langle\!\langle y\rangle\!\rangle = \mathit{ConstantNode}\ \mathit{c\text{-}2} \qquad \mathit{c\text{-}2} = \mathit{IntVal}\ 32\ 0}{\mathit{CanonicalizeAdd}\ g\ (\mathit{AddNode}\ x\ y)\ (\mathit{RefNode}\ x)}$$

$\llbracket \textit{CanonicalizeAdd}\ g\ \textit{before}\ \textit{after};\ \textit{wf-graph}\ g \land \textit{wf-stamps}\ g \land \textit{wf-values}\ g;\ g\ m \vdash \textit{before} \mapsto \textit{IntVal}\ b\ \textit{res};\ g\ m \vdash \textit{after} \mapsto \textit{IntVal}\ b'\ \textit{res}'\rrbracket \Longrightarrow \textit{res} = \textit{res}'$

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

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

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

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

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

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*::*FieldRefHeap*
  assumes *kind g nid* = *before*
  assumes *CanonicalizeIf g before after*
  assumes g' = *replace-node-fake nid after g*
  assumes $g \vdash (\textit{nid},\ m,\ h) \rightarrow (\textit{nid}',\ m,\ h)$
  shows $\textit{nid} \mid g \sim g'$
  sorry

$\llbracket g\langle\langle \textit{nid}\rangle\rangle = \textit{before};\ \textit{CanonicalizeIf}\ g\ \textit{before}\ \textit{after};$
$g' = \textit{replace-node-fake nid after}\ g;\ g \vdash (\textit{nid},\ m,\ h) \rightarrow (\textit{nid}',\ m,\ h)\rrbracket$
$\Longrightarrow \textit{nid} \mid g \sim g'$

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

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

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

notation (*latex* **output**)
  IntVal (*IntVal* (*2* -))

end