Unspecified Veriopt Theory

April 23, 2021

Contents

```
theory ATVA2021
  imports
    Optimizations.CanonicalizationProofs
begin
```

```
notation (latex)
  kind (-⟪-⟫)
```

$$isBinaryArithmeticNodeType :: 'a$$

```
inputs-of :: IRNode ⇒ nat list
inputs-of (ConstantNode const) = []
inputs-of (ParameterNode index) = []
inputs-of (ValuePhiNode nid0.0 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 :: (nat ⇒ IRNode option) ⇒ nat set
ids-fake g = {nid ∈ dom g | g nid ≠ Some NoNode}
```

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

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

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

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

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

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

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

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

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

$$\begin{array}{l} \textit{wff-phis}\ g = \\ (\forall\, n{\in}\textit{ids}\ g. \\ \quad \textit{is-PhiNode}\ g\langle\!\langle n\rangle\!\rangle \longrightarrow \\ \quad |\textit{ir-values}\ g\langle\!\langle n\rangle\!\rangle| = \\ \quad |\textit{ir-ends}\ g\langle\!\langle \textit{ir-merge}\ g\langle\!\langle n\rangle\!\rangle\rangle\!\rangle|) \end{array}$$

$$\begin{array}{l} \textit{wff-ends}\ g = \\ (\forall\, n{\in}\textit{ids}\ g. \\ \quad \textit{is-AbstractEndNode}\ g\langle\!\langle n\rangle\!\rangle \longrightarrow \\ \quad \textit{0} < |\textit{usages}\ g\ n|) \end{array}$$

$$\begin{array}{l} \textit{wff-graph} :: \textit{IRGraph} \Rightarrow \textit{bool} \\ \textit{wff-graph}\ g = (\textit{wff-start}\ g \wedge \textit{wff-closed}\ g \wedge \textit{wff-phis}\ g \wedge \textit{wff-ends}\ g) \end{array}$$

type-synonym $\textit{Signature} = \textit{string}$
type-synonym $\textit{Program} = \textit{Signature} \Rightarrow \textit{IRGraph option}$

print-antiquotations

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

$$\begin{array}{l} \textit{h-load-field} :: \textit{string} \Rightarrow \textit{objref} \Rightarrow \textit{DynamicHeap} \Rightarrow \textit{Value} \\ \textit{h-load-field}\ f\ r\ (h,\ n) = h\ f\ r \end{array}$$

$$\begin{array}{l} \textit{h-store-field} :: \textit{string} \Rightarrow \textit{objref} \Rightarrow \textit{Value} \Rightarrow \textit{DynamicHeap} \Rightarrow \textit{DynamicHeap} \\ \textit{h-store-field}\ f\ r\ v\ (h,\ n) = (h(f := (h\ f)(r := v)),\ n) \end{array}$$

$$\begin{array}{l} \textit{h-new-inst} :: \textit{DynamicHeap} \Rightarrow (\textit{DynamicHeap} \times \textit{Value}) \\ \textit{h-new-inst}\ (h,\ n) = ((h,\ n + \textit{1}),\ \textit{ObjRef}\ (\textit{Some}\ n)) \end{array}$$

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

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

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

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

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

$$\frac{g\langle\!\langle x\rangle\!\rangle = \mathit{ConstantNode}\ \mathit{c\text{-}1} \qquad g\langle\!\langle y\rangle\!\rangle = \mathit{ConstantNode}\ \mathit{c\text{-}2} \qquad \mathit{val} = \mathit{c\text{-}1} +\!* \mathit{c\text{-}2}}{\mathit{CanonicalizeAdd}\ g\ (\mathit{AddNode}\ x\ y)\ (\mathit{ConstantNode}\ \mathit{val})}$$

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

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

$[\![\mathit{CanonicalizeAdd}\ g\ \mathit{before}\ \mathit{after};\ \mathit{wff\text{-}graph}\ g \land \mathit{wff\text{-}stamps}\ g \land \mathit{wff\text{-}values}\ g;\ g\ m \vdash \mathit{before} \mapsto \mathit{IntVal}\ b\ \mathit{res};\ g\ m \vdash \mathit{after} \mapsto \mathit{IntVal}\ b'\ \mathit{res}']\!] \Longrightarrow \mathit{res} = \mathit{res}'$

$$\frac{g \vdash (\mathit{nid},\ m,\ h) \rightarrow (\mathit{nid}',\ m,\ h)}{g\ m\ h \vdash \mathit{nid} \leadsto \mathit{nid}'}$$

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

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

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

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

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 $g' = \mathit{replace\text{-}node\text{-}fake}\ \mathit{nid}\ \mathit{after}\ g$
  assumes $g \vdash (\mathit{nid},\ m,\ h) \rightarrow (\mathit{nid}',\ m,\ h)$
  shows $\mathit{nid} \mid g \sim g'$
  sorry

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

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

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

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

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

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

end