Veriopt Theories

July 13, 2022

Contents

1 Canonicalization Phase

theory *Common*
  imports
    OptimizationDSL.Canonicalization
    Semantics.IRTreeEvalThms
begin

lemma *size-pos*[*simp*]: $0 < size\ y$
  apply (*induction y*; *auto?*)
  subgoal premises *prems* **for** *op a b*
    using *prems* **by** (*induction op*; *auto*)
  done

lemma *size-non-add*: $op \neq BinAdd \Longrightarrow size\ (BinaryExpr\ op\ a\ b) = size\ a + size\ b$
  by (*induction op*; *auto*)

lemma *size-non-const*:
  $\neg\ is\text{-}ConstantExpr\ y \Longrightarrow 1 < size\ y$
  using *size-pos* **apply** (*induction y*; *auto*)
  subgoal premises *prems* **for** *op a b*
    apply (*cases op = BinAdd*)
    using *size-non-add size-pos* **apply** *auto*
    by (*simp add*: *Suc-lessI one-is-add*)+
  done

end

1.1 Conditional Expression

theory *ConditionalPhase*

imports
Common
begin

phase *Conditional*
terminating *size*
begin

lemma *negates*: $\textit{is-IntVal32}\ e \vee \textit{is-IntVal64}\ e \Longrightarrow \textit{val-to-bool}\ (\textit{val}[e]) \equiv \neg(\textit{val-to-bool}\ (\textit{val}[!e]))$
using *intval-logic-negation.simps* **unfolding** *logic-negate-def*
by (*smt* (*verit*, *best*) *Value.collapse*(*1*) *is-IntVal64-def val-to-bool.simps*(*1*) *val-to-bool.simps*(*2*) *zero-neq-one*)

lemma *negation-condition-intval*:
assumes $e \neq \textit{UndefVal} \wedge \neg(\textit{is-ObjRef}\ e) \wedge \neg(\textit{is-ObjStr}\ e)$
shows $\textit{val}[(!e)\ ?\ x : y] = \textit{val}[e\ ?\ y : x]$
using *assms* **by** (*cases e*; *auto simp*: *negates logic-negate-def*)

optimization *negate-condition*: $((!e)\ ?\ x : y) \longmapsto (e\ ?\ y : x)$
apply *simp* **using** *negation-condition-intval*
by (*smt* (*verit*, *ccfv-SIG*) *ConditionalExpr ConditionalExprE Value.collapse*(*3*) *Value.collapse*(*4*) *Value.exhaust-disc evaltree-not-undef intval-logic-negation.simps*(*4*) *intval-logic-negation.simps*(*5*) *negates unary-eval.simps*(*4*) *unfold-unary*)

optimization *const-true*: $(\textit{true}\ ?\ x : y) \longmapsto x$ **.**

optimization *const-false*: $(\textit{false}\ ?\ x : y) \longmapsto y$ **.**

optimization *equal-branches*: $(e\ ?\ x : x) \longmapsto x$ **.**

definition *wff-stamps* :: *bool* **where**
$\textit{wff-stamps} = (\forall\ m\ p\ \textit{expr}\ \textit{val}\ .\ ([m,p] \vdash \textit{expr} \mapsto \textit{val}) \longrightarrow \textit{valid-value}\ \textit{val}\ (\textit{stamp-expr}\ \textit{expr}))$

definition *wf-stamp* :: *IRExpr* ⇒ *bool* **where**
$\textit{wf-stamp}\ e = (\forall\ m\ p\ v.\ ([m,\ p] \vdash e \mapsto v) \longrightarrow \textit{valid-value}\ v\ (\textit{stamp-expr}\ e))$

end

end