Veriopt Theories

July 14, 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-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

definition *well-formed-equal* :: $Value \Rightarrow Value \Rightarrow bool$
(**infix** $\approx$ *50*) **where**
well-formed-equal $v_1\ v_2 = (v_1 \neq UndefVal \longrightarrow v_1 = v_2)$

lemma *well-formed-equal-defn* [*simp*]:
well-formed-equal v_1 v_2 = ($v_1 \neq$ *UndefVal* $\longrightarrow v_1 = v_2$)
unfolding *well-formed-equal-def* **by** *simp*

end

1.1 Conditional Expression

theory *ConditionalPhase*
imports
Common
begin

phase *Conditional*
terminating *size*
begin

lemma *negates*: *is-IntVal32 e* $\vee$ *is-IntVal64 e* $\Longrightarrow$ *val-to-bool* (*val*[*e*]) $\equiv \neg$(*val-to-bool* (*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 $e \neq$ *UndefVal* $\wedge \neg$(*is-ObjRef e*) $\wedge \neg$(*is-ObjStr e*)
shows *val*[(!*e*) *?* *x* : *y*] = *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*: (*true* *?* *x* : *y*) $\longmapsto$ *x* **.**

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

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

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

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

end

end