Veriopt Theories

December 16, 2022

Contents

1   Graph Type Class Instantiations

theory
  IRGraphSort
imports
  Graph.IRGraph
begin

An *IRGraph* can be treated as an instantiation of various type classes such as the equal type class.

Here we define the instantiations for all type classes of which *IRGraph* belongs.

1.1   Equal Instance

The following is an incorrect definition of equality which is used to allow code generation of the Canonicalization phase. Without this we have to use the values command to generate all possible results of the canonicalization phase.

Note that we should be able to find a correct equality definition in terms of the ids, kind, and stamp function. However, we are yet to find a satisfactory definition as yet.

definition *IRGraph-equal* :: *IRGraph* $\Rightarrow$ *IRGraph* $\Rightarrow$ *bool* **where**
  IRGraph-equal g1 g2 = True

instantiation *IRGraph* :: *equal*
begin

definition *equal-IRGraph :: IRGraph ⇒ IRGraph ⇒ bool* **where**
  equal-IRGraph = IRGraph-equal

instance proof
  fix *g1 g2 :: IRGraph*
  show *equal-class.equal g1 g2 ⟷ (g1 = g2)*
    apply *standard*
    unfolding *equal-IRGraph-def IRGraph-equal-def*
    unfolding *as-list-def*
    apply *transfer*
    sorry
qed
end

end
theory *ValidationSnippets*
  imports
    IRGraphSort
    Snippets.Snipping
    Graph.Comparison
    ConditionalElimination.ConditionalElimination
begin

notation (*latex*)
  kind (*-⟪-⟫*)

StepSemantics

intval-mod (IntVal (b1.0::nat) (v1.0::64 word))
 (IntVal (b2.0::nat) (v2.0::64 word)) =
new-int-bin b1.0 b2.0
 (word-of-int
   (int-signed-value b1.0 v1.0 smod
    int-signed-value b2.0 v2.0))

eval:rem

ModuloTestSnippet

static-test   moduloSnippet   [(IntVal   32   (1)),   (Intval   32
(−2147483648))] (IntVal 32 (1))

2

ConditionalInitialEncoding

definition *test1-initial* :: *IRGraph* **where**
test1-initial = *irgraph* [
(*0*, (*StartNode* (*Some 3*) *7*), *VoidStamp*),
(*1*, (*ParameterNode 0*), *default-stamp*),
(*2*, (*ParameterNode 1*), *default-stamp*),
(*3*, (*FrameState* [] *None None None*), *IllegalStamp*),
(*4*, (*IntegerLessThanNode 2 1*), *VoidStamp*),
(*5*, (*BeginNode 8*), *VoidStamp*),
(*6*, (*BeginNode 13*), *VoidStamp*),
(*7*, (*IfNode 4 6 5*), *VoidStamp*),
(*8*, (*EndNode*), *VoidStamp*),
(*9*, (*MergeNode* [*8, 10*] (*Some 16*) *18*), *VoidStamp*),
(*10*, (*EndNode*), *VoidStamp*),
(*11*, (*BeginNode 15*), *VoidStamp*),
(*12*, (*BeginNode 10*), *VoidStamp*),
(*13*, (*IfNode 4 11 12*), *VoidStamp*),
(*14*, (*ConstantNode* (*IntVal 32* (*1*))), *IntegerStamp 32* (*1*) (*1*)),
(*15*, (*ReturnNode* (*Some 14*) *None*), *VoidStamp*),
(*16*, (*FrameState* [] *None None None*), *IllegalStamp*),
(*17*, (*ConstantNode* (*IntVal 32* (*2*))), *IntegerStamp 32* (*2*) (*2*)),
(*18*, (*ReturnNode* (*Some 17*) *None*), *VoidStamp*)
]

definition *test1-final* :: *IRGraph* **where**
test1-final = *irgraph* [
(*0*, (*StartNode* (*Some 3*) *7*), *VoidStamp*),
(*1*, (*ParameterNode 0*), *default-stamp*),
(*2*, (*ParameterNode 1*), *default-stamp*),
(*3*, (*FrameState* [] *None None None*), *IllegalStamp*),
(*4*, (*IntegerLessThanNode 2 1*), *VoidStamp*),
(*5*, (*BeginNode 8*), *VoidStamp*),
(*6*, (*BeginNode 13*), *VoidStamp*),
(*7*, (*IfNode 4 6 5*), *VoidStamp*),
(*8*, (*EndNode*), *VoidStamp*),
(*9*, (*MergeNode* [*8*, *10*] (*Some 16*) *18*), *VoidStamp*),
(*10*, (*EndNode*), *VoidStamp*),
(*11*, (*BeginNode 15*), *VoidStamp*),
(*12*, (*BeginNode 10*), *VoidStamp*),
(*13*, (*IfNode 19 11 12*), *VoidStamp*),
(*14*, (*ConstantNode* (*IntVal 32* (*1*))), *IntegerStamp 32* (*1*) (*1*)),
(*15*, (*ReturnNode* (*Some 14*) *None*), *VoidStamp*),
(*16*, (*FrameState* [] *None None None*), *IllegalStamp*),
(*17*, (*ConstantNode* (*IntVal 32* (*2*))), *IntegerStamp 32* (*2*) (*2*)),
(*18*, (*ReturnNode* (*Some 17*) *None*), *VoidStamp*),
(*19*, (*ConstantNode* (*IntVal 1* (*1*))), *VoidStamp*)
]

value *runConditionalElimination test1-initial*

corollary (*runConditionalElimination test1-initial*) $\approx_s$ *test1-final*
  by *eval*

end