

Oct 10, 16 8:43

ImpEval.v

Page 1/2

```

Require Import List.
Require Import String.
Require Import ZArith.

```

```

Open Scope list_scope.
Open Scope string_scope.
Open Scope Z_scope.

```

```

Require Import ImpSyntax.
Require Import ImpCommon.

```

```

Inductive eval_unop : op1 -> val -> val -> Prop :=

```

```

| eval_neg :
  forall i,
    eval_unop Oneg (Vint i)
      (Vint (Z.opp i))
| eval_not :
  forall b,
    eval_unop Onot (Vbool b)
      (Vbool (negb b)).

```

```

Inductive eval_binop : op2 -> val -> val -> val -> Prop :=

```

```

| eval_add_i :
  forall i1 i2,
    eval_binop Oadd (Vint i1) (Vint i2)
      (Vint (Z.add i1 i2))
| eval_add_s :
  forall s1 s2,
    eval_binop Oadd (Vstr s1) (Vstr s2)
      (Vstr (String.append s1 s2))
| eval_sub :
  forall i1 i2,
    eval_binop Osub (Vint i1) (Vint i2)
      (Vint (Z.sub i1 i2))
| eval_mul :
  forall i1 i2,
    eval_binop Omul (Vint i1) (Vint i2)
      (Vint (Z.mul i1 i2))
| eval_div :
  forall i1 i2,
    i2 <> 0 ->
    eval_binop Odiv (Vint i1) (Vint i2)
      (Vint (Z.div i1 i2))
| eval_mod :
  forall i1 i2,
    i2 <> 0 ->
    eval_binop Omod (Vint i1) (Vint i2)
      (Vint (Z.modulo i1 i2))
| eval_eq :
  forall v1 v2,
    eval_binop Oeq v1 v2
      (Vbool (imp_eq v1 v2))
| eval_lt :
  forall i1 i2,
    eval_binop Olt (Vint i1) (Vint i2)
      (Vbool (imp_lt i1 i2))
| eval_le :
  forall i1 i2,
    eval_binop Ole (Vint i1) (Vint i2)
      (Vbool (imp_le i1 i2))
| eval_conj :
  forall b1 b2,
    eval_binop Oconj (Vbool b1) (Vbool b2)
      (Vbool (andb b1 b2))
| eval_disj :
  forall b1 b2,
    eval_binop Odisj (Vbool b1) (Vbool b2)
      (Vbool (orb b1 b2)).

```

Oct 10, 16 8:43

ImpEval.v

Page 2/2

```

Inductive eval_e (s : store) : expr -> val -> Prop :=

```

```

| eval_val :
  forall v,
    eval_e s (Eval v) v
| eval_var :
  forall x v,
    lkup s x = Some v ->
    eval_e s (Evar x) v
| eval_op1 :
  forall op e v v',
    eval_e s e v ->
    eval_unop op v v' ->
    eval_e s (Eop1 op e) v'
| eval_op2 :
  forall op e1 e2 v1 v2 v',
    eval_e s e1 v1 ->
    eval_e s e2 v2 ->
    eval_binop op v1 v2 v' ->
    eval_e s (Eop2 op e1 e2) v'.

```

```

Inductive eval_s :

```

```

  store -> stmt -> store -> Prop :=
| eval_nop :
  forall s,
    eval_s s Snop s
| eval_set :
  forall s x e v,
    eval_e s e v ->
    eval_s s (Sset x e) (update s x v)
| eval_ifelse_t :
  forall s e p1 p2 s',
    eval_e s e (Vbool true) ->
    eval_s s p1 s' ->
    eval_s s (Sifelse e p1 p2) s'
| eval_ifelse_f :
  forall s e p1 p2 s',
    eval_e s e (Vbool false) ->
    eval_s s p2 s' ->
    eval_s s (Sifelse e p1 p2) s'
| eval_while_t :
  forall s1 e p s2 s3,
    eval_e s1 e (Vbool true) ->
    eval_s s1 p s2 ->
    eval_s s2 (Swhile e p) s3 ->
    eval_s s1 (Swhile e p) s3
| eval_while_f :
  forall s e p,
    eval_e s e (Vbool false) ->
    eval_s s (Swhile e p) s
| eval_seq :
  forall s1 p1 s2 p2 s3,
    eval_s s1 p1 s2 ->
    eval_s s2 p2 s3 ->
    eval_s s1 (Sseq p1 p2) s3.

```