

Oct 17, 16 7:08

**ImpEval.v**

Page 1/4

```

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 : opl -> 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 17, 16 7:08

**ImpEval.v**

Page 2/4

```

Inductive eval_e (s : store) (h : heap) :
  expr -> val -> Prop :=
| eval_val :
  forall v,
  eval_e s h (Eval v) v
| eval_var :
  forall x v,
  lkup s x = Some v ->
  eval_e s h (Evar x) v
| eval_op1 :
  forall op e v v',
  eval_e s h e v ->
  eval_unop op v v' ->
  eval_e s h (Eop1 op e) v'
| eval_op2 :
  forall op e1 e2 v1 v2 v',
  eval_e s h e1 v1 ->
  eval_e s h e2 v2 ->
  eval_binop op v1 v2 v' ->
  eval_e s h (Eop2 op e1 e2) v'
| eval_len_a :
  forall e a l,
  eval_e s h e (Vaddr a) ->
  read h a = Some (Vint l) ->
  eval_e s h (Elen e) (Vint l)
| eval_len_s :
  forall e cs l,
  eval_e s h e (Vstr cs) ->
  Z.of_nat (String.length cs) = l ->
  eval_e s h (Elen e) (Vint l)
| eval_idx_a :
  forall e1 e2 a l i v,
  eval_e s h e1 (Vaddr a) ->
  eval_e s h e2 (Vint i) ->
  read h a = Some (Vint l) ->
  0 <= i < l ->
  read h (Zsucc (a + i)) = Some v ->
  eval_e s h (Eidx e1 e2) v
| eval_idx_s :
  forall e1 e2 cs i c,
  eval_e s h e1 (Vstr cs) ->
  eval_e s h e2 (Vint i) ->
  0 <= i ->
  String.get (Z.to_nat i) cs = Some c ->
  eval_e s h (Eidx e1 e2) (Vstr (String c EmptyString)).
Inductive evals_e (s : store) (h : heap) :
  list expr -> list val -> Prop :=
| evals_nil :
  evals_e s h nil nil
| evals_cons :
  forall e es v vs,
  eval_e s h e v ->
  evals_e s h es vs ->
  evals_e s h (e :: es) (v :: vs).

Inductive eval_s (env : env) :
  store -> heap -> stmt -> store -> heap -> Prop :=
| eval_nop :
  forall s h,
  eval_s env
  s h Snop
  s h
| eval_set :
  forall s h x e v,
  eval_e s h e v ->
  eval_s env
  s h (Sset x e)
  (update s x v) h

```

Oct 17, 16 7:08

**ImpEval.v**

Page 3/4

```

| eval_alloc :
  forall s h x e1 e2 i v,
    eval_e s h e1 (Vint i) ->
    eval_e s h e2 v ->
    0 <= i ->
    eval_s env
      s h (Salloc x e1 e2)
      (update s x (Vaddr (zlen h))) (alloc h i v)

| eval_write :
  forall s h x e1 e2 a l i v h',
    lkup s x = Some (Vaddr a) ->
    read h a = Some (Vint l) ->
    eval_e s h e1 (Vint i) ->
    eval_e s h e2 v ->
    0 <= i < l ->
    write h (Zsucc (a + i)) v = Some h' ->
    eval_s env
      s h (Swrite x e1 e2)
      s h'

| eval_call_internal :
  forall s h f params body ret vs sf sf' h' x es v,
    locate env f = Some (Func f params body ret) ->
    evals_e s h es vs ->
    updates store_0 params vs = Some sf ->
    eval_s env
      sf h body
      sf' h' ->
    eval_e sf' h' ret v ->
    eval_s env
      s h (Scall x f es)
      (update s x v) h'

| eval_call_external :
  forall x f es s h vs v' h',
    locate env f = None ->
    evals_e s h es vs ->
    extcall_spec f vs h v' h' ->
    eval_s env
      s h (Scall x f es)
      (update s x v') h'

| eval_ifelse_t :
  forall s h e p1 p2 s' h',
    eval_e s h e (Vbool true) ->
    eval_s env
      s h p1
      s' h' ->
    eval_s env
      s h (Sifelse e p1 p2)
      s' h'

| eval_ifelse_f :
  forall s h e p1 p2 s' h',
    eval_e s h e (Vbool false) ->
    eval_s env
      s h p2
      s' h' ->
    eval_s env
      s h (Sifelse e p1 p2)
      s' h'

| eval_while_t :
  forall s1 h1 e p s2 h2 s3 h3,
    eval_e s1 h1 e (Vbool true) ->
    eval_s env
      s1 h1 p
      s2 h2 ->
    eval_s env
      s2 h2 (Swhile e p)
      s3 h3 ->
    eval_s env
      s1 h1 (Swhile e p)
      s3 h3

```

**ImpEval.v**

Page 4/4

```

| eval_while_f :
  forall s h e p,
    eval_e s h e (Vbool false) ->
    eval_s env
      s h (Swhile e p)
      s h

| eval_seq :
  forall s1 h1 p1 s2 h2 p2 s3 h3,
    eval_s env
      s1 h1 p1
      s2 h2 ->
    eval_s env
      s2 h2 p2
      s3 h3 ->
    eval_s env
      s1 h1 (Sseq p1 p2)
      s3 h3

| eval_incall :
  forall s1 h1 body s2 h2 v rl ret rs,
    eval_s env
      s1 h1 body
      s2 h2 ->
    eval_e s2 h2 ret v ->
    eval_s env
      s1 h1 (Sincall body ret rl rs)
      (update rs rl v) h2.

Inductive eval_p : prog -> val -> Prop :=
| eval_prog :
  forall funcs main ret s' h' v,
    eval_s funcs
      store_0 heap_0 main
      s' h' ->
    eval_e s' h' ret v ->
    eval_p (Prog funcs main ret) v.

```