

Oct 17, 16 6:59

ImpInterp.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 StructTactics.
Require Import ImpSyntax.
Require Import ImpCommon.

Definition interp_op1
  (op : op1) (v : val) : option val :=
  match op, v with
  | Oneg, Vint i =>
    Some (Vint (Z.opp i))
  | Onot, Vbool b =>
    Some (Vbool (negb b))
  | _, _ =>
    None
  end.

Definition interp_op2
  (op : op2) (v1 v2 : val) : option val :=
  match op, v1, v2 with
  | Oadd, Vint i1, Vint i2 =>
    Some (Vint (Z.add i1 i2))
  | Oadd, Vstr s1, Vstr s2 =>
    Some (Vstr (String.append s1 s2))
  | Osub, Vint i1, Vint i2 =>
    Some (Vint (Z.sub i1 i2))
  | Omul, Vint i1, Vint i2 =>
    Some (Vint (Z.mul i1 i2))
  | Odiv, Vint i1, Vint i2 =>
    if Z.eq_dec i2 0 then
      None
    else
      Some (Vint (Z.div i1 i2))
  | Omod, Vint i1, Vint i2 =>
    if Z.eq_dec i2 0 then
      None
    else
      Some (Vint (Z.modulo i1 i2))
  | Oeq, v1, v2 =>
    Some (Vbool (imp_eq v1 v2))
  | Olt, Vint i1, Vint i2 =>
    Some (Vbool (imp_lt i1 i2))
  | Ole, Vint i1, Vint i2 =>
    Some (Vbool (imp_le i1 i2))
  | Oconj, Vbool b1, Vbool b2 =>
    Some (Vbool (andb b1 b2))
  | Odisj, Vbool b1, Vbool b2 =>
    Some (Vbool (orb b1 b2))
  | _, _, _ =>
    None
  end.

Fixpoint interp_e (s : store) (h : heap)
  (e : expr) : option val :=
  match e with
  | Eval v => Some v
  | Evar x => lkup s x
  | Eop1 op e1 =>
    match interp_e s h e1 with
    | Some v1 => interp_op1 op v1
    | _ => None
    end
  | Eop2 op e1 e2 =>

```

Oct 17, 16 6:59

ImpInterp.v

Page 2/4

```

    match interp_e s h e1, interp_e s h e2 with
    | Some v1, Some v2 => interp_op2 op v1 v2
    | _, _ => None
    end
  | Elen e1 =>
    match interp_e s h e1 with
    | Some (Vaddr a) =>
      match read h a with
      | Some (Vint l) => Some (Vint l)
      | _ => None
      end
    | Some (Vstr cs) =>
      Some (Vint (Z.of_nat (String.length cs)))
    | _ => None
    end
  | Eidx e1 e2 =>
    match interp_e s h e1, interp_e s h e2 with
    | Some (Vaddr a), Some (Vint i) =>
      match read h a with
      | Some (Vint l) =>
        if Z.le_dec 0 i then
          if Z.lt_dec i l then
            read h (Zsucc (a + i))
          else
            None
          else
            None
        | _ => None
        end
    | Some (Vstr cs), Some (Vint i) =>
      if Z.le_dec 0 i then
        match String.get (Z.to_nat i) cs with
        | Some c =>
          Some (Vstr (String c EmptyString))
        | None => None
        end
      else
        None
      end
    | _, _ => None
    end
  end.

Fixpoint interps_e (s : store) (h : heap)
  (es : list expr) : option (list val) :=
  match es with
  | nil => Some nil
  | e :: t =>
    match interp_e s h e, interps_e s h t with
    | Some v, Some vs => Some (v :: vs)
    | _, _ => None
    end
  end.

Fixpoint interp_s (s : store) (h : heap)
  (p : stmt) : option (store * heap * stmt) :=
  match p with
  | Snop => None
  | Sset x e =>
    match interp_e s h e with
    | Some v => Some (update s x v, h, Snop)
    | _ => None
    end
  | Salloc x e1 e2 =>
    match interp_e s h e1, interp_e s h e2 with
    | Some (Vint i), Some v =>
      if Z.le_dec 0 i then
        Some ( update s x (Vaddr (zlen h))
              , alloc h i v
              , Snop

```

Oct 17, 16 6:59

ImplInterp.v

Page 3/4

```

    else
      None
    | _, _ => None
  end
| Swrite x e1 e2 =>
  match lkup s x
  , interp_e s h e1
  , interp_e s h e2 with
  | Some (Vaddr a), Some (Vint i), Some v =>
    match read h a with
    | Some (Vint l) =>
      if Z_le_dec 0 i then
        if Z_lt_dec i l then
          match write h (Zsucc (a + i)) v with
          | Some h' => Some (s, h', Snop)
          | None => None
        end
      else
        None
      end
    | _ => None
  end
| _, _, _ => None
end
| Sifelse e p1 p2 =>
  match interp_e s h e with
  | Some (Vbool true) => Some (s, h, p1)
  | Some (Vbool false) => Some (s, h, p2)
  | _ => None
  end
| Swhile e p =>
  match interp_e s h e with
  | Some (Vbool true) =>
    Some (s, h, Sseq p (Swhile e p))
  | Some (Vbool false) =>
    Some (s, h, Snop)
  | _ =>
    None
  end
| Sseq p1 p2 =>
  if isNop p1 then
    Some (s, h, p2)
  else
    match interp_s s h p1 with
    | Some (s', h', p1') =>
      Some (s', h', Sseq p1' p2)
    | None => None
    end
  end
end.

Fixpoint interps_p (fuel : nat)
(s : store) (h : heap) (p : stmt)
(ret : expr) : result :=
match fuel with
| 0 => Timeout s h p ret
| S n =>
  if isNop p then
    match interp_e s h ret with
    | Some v => Done h v
    | None => Stuck s h p ret
    end
  else
    match interp_s s h p with
    | Some (s', h', p') =>
      interps_p n s' h' p' ret
    | _ =>
      Stuck s h p ret
  end
end.

```

Oct 17, 16 6:59

ImplInterp.v

Page 4/4

```

end
end.

```