

Oct 24, 16 7:17

## ImplInterpNock.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.

```

```

(** same as ImpInterp, but without safety checks *)

```

```

Definition lkup' (s : store) (x : string) : val :=
  match lkup s x with
  | Some v => v
  | None => Vint 0
  end.

```

```

Fixpoint updates' (s : store)
  (xs : list string) (vs : list val) : store :=
  match xs, vs with
  | x :: xs', v :: vs' =>
    updates' (update s x v) xs' vs'
  | _, _ => s
  end.

```

```

Definition strget' (v : val) (s : string) : val :=
  match v with
  | Vint i =>
    match String.get (Z.to_nat i) s with
    | Some c => Vstr (String c EmptyString)
    | None => Vint 0
    end
  | _ => Vint 0
  end.

```

```

Definition read' (h : heap) (v : val) : val :=
  match v with
  | Vaddr a =>
    match read h a with
    | Some v' => v'
    | _ => Vint 0
    end
  | _ => Vint 0
  end.

```

```

Definition alloc' (h : heap) (v1 v2 : val) : heap :=
  match v1 with
  | Vint i1 => alloc h i1 v2
  | _ => h
  end.

```

```

Definition write' (h : heap) (v1 v2 : val) : heap :=
  match v1 with
  | Vaddr i1 =>
    match write h i1 v2 with
    | Some h' => h'
    | _ => h
    end
  | _ => h
  end.

```

```

Definition interp_op1
  (op : op1) (v : val) : val :=
  match op, v with
  | Oneg, Vint i =>
    Vint (Z.opp i)

```

Oct 24, 16 7:17

## ImplInterpNock.v

Page 2/4

```

  | Onot, Vbool b =>
    Vbool (negb b)
  | _, _ =>
    Vint 0
  end.

```

```

Definition interp_op2
  (op : op2) (v1 v2 : val) : val :=
  match op, v1, v2 with
  | Oadd, Vint i1, Vint i2 =>
    Vint (Z.add i1 i2)
  | Oadd, Vstr s1, Vstr s2 =>
    Vstr (String.append s1 s2)
  | Osub, Vint i1, Vint i2 =>
    Vint (Z.sub i1 i2)
  | Omul, Vint i1, Vint i2 =>
    Vint (Z.mul i1 i2)
  | Odiv, Vint i1, Vint i2 =>
    Vint (Z.div i1 i2)
  | Omod, Vint i1, Vint i2 =>
    Vint (Z.modulo i1 i2)
  | Oeq, v1, v2 =>
    Vbool (imp_eq v1 v2)
  | Olt, Vint i1, Vint i2 =>
    Vbool (imp_lt i1 i2)
  | Ole, Vint i1, Vint i2 =>
    Vbool (imp_le i1 i2)
  | Oconj, Vbool b1, Vbool b2 =>
    Vbool (andb b1 b2)
  | Odisj, Vbool b1, Vbool b2 =>
    Vbool (orb b1 b2)
  (** hacks *)
  | Oadd, Vaddr i1, Vint i2 =>
    Vaddr (Zsucc (Z.add i1 i2))
  | _, _, _ =>
    Vint 0
  end.

```

```

Fixpoint interp_e (s : store) (h : heap)
  (e : expr) : val :=
  match e with
  | Eval v =>
    v
  | Evar x =>
    lkup' s x
  | Eopl op e1 =>
    interp_op1 op
      (interp_e s h e1)
  | Eop2 op e1 e2 =>
    interp_op2 op
      (interp_e s h e1)
      (interp_e s h e2)
  | Elen e1 =>
    match interp_e s h e1 with
    | Vaddr a => read' h (Vaddr a)
    | Vstr cs => Vint (Z.of_nat (String.length cs))
    | _ => Vint 0
    end
  | Eidx e1 e2 =>
    match interp_e s h e1 with
    | Vaddr a =>
      read' h (interp_op2
        Oadd (Vaddr a) (interp_e s h e2))
    | Vstr cs =>
      strget' (interp_e s h e2) cs
    | _ => Vint 0
    end
  end.

```

Oct 24, 16 7:17

## ImplInterpNock.v

Page 3/4

```

Fixpoint interps_e (s : store) (h : heap)
  (es : list expr) : list val :=
  match es with
  | nil => nil
  | e :: rest =>
    interp_e s h e :: interps_e s h rest
  end.

Fixpoint interp_s (env : env) (s : store) (h : heap)
  (p : stmt) : store * heap * stmt :=
  match p with
  | Snop =>
    (s, h, p)
  | Sset x e =>
    (update s x (interp_e s h e), h, Snop)
  | Salloc x e1 e2 =>
    ( update s x (Vaddr (zlen h))
      , alloc' h (interp_e s h e1) (interp_e s h e2)
      , Snop
    )
  | Swrite x e1 e2 =>
    let h' :=
      write' h (interp_op2 Oadd
                (lkup' s x)
                (interp_e s h e1))
            (interp_e s h e2)
    in
    (s, h', Snop)
  | Scall x f es =>
    let vs := interps_e s h es in
    match locate env f with
    | Some (Func _ params body ret) =>
      ( updates' store_0 params vs
        , h
        , Sincall body ret x s
        )
    | None =>
      let (v', h') := extcall f vs h in
      (update s x v', h', Snop)
    end
  | Sifelse e p1 p2 =>
    match interp_e s h e with
    | Vbool false => (s, h, p2)
    | _ => (s, h, p1)
    end
  | Swhile e p =>
    match interp_e s h e with
    | Vbool false => (s, h, Snop)
    | _ => (s, h, Sseq p (Swhile e p))
    end
  | Sseq p1 p2 =>
    if isNop p1 then
      (s, h, p2)
    else
      match interp_s env s h p1 with
      | (s', h', p1') => (s', h', Sseq p1' p2)
      end
    end
  | Sincall p ret x sr =>
    if isNop p then
      ( update sr x (interp_e s h ret)
        , h
        , Snop
        )
    else
      match interp_s env s h p with
      | (s', h', p') =>
        (s', h', Sincall p' ret x sr)
      end
    end
  end.

```

Oct 24, 16 7:17

## ImplInterpNock.v

Page 4/4

```

Fixpoint interps_p (fuel : nat) (env : env)
  (s : store) (h : heap) (p : stmt)
  (ret : expr) : result :=
  match fuel with
  | 0 => Timeout s h p ret
  | S n =>
    if isNop p then
      Done h (interp_e s h ret)
    else
      match interp_s env s h p with
      | (s', h', p') =>
        interps_p n env s' h' p' ret
      end
    end
  end.

Definition interp_p (fuel : nat) (p : prog) : result :=
  match p with
  | Prog funcs body ret =>
    interps_p fuel funcs store_0 heap_0 body ret
  end.

```