

Oct 10, 16 8:43

ImpCommon.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.

```

```

(** Tests *)

```

```

Definition pred_of_dec {A B}
  (f : forall a1 a2, {B a1 a2} + {~ B a1 a2})
  (a1 a2 : A) : bool :=
  if f a1 a2 then true else false.

```

```

Definition val_eq_dec :
  forall v1 v2 : val,
  {v1 = v2} + {v1 <> v2}.

```

```

Proof.
  decide equality.
- apply Bool.bool_dec.
- apply Z.eq_dec.
- apply String.string_dec.
Defined.

```

```

Definition imp_eq := pred_of_dec val_eq_dec.
Definition imp_lt := pred_of_dec Z_lt_dec.
Definition imp_le := pred_of_dec Z_le_dec.

```

```

(** Stores *)

```

```

Definition store_0 : store :=
  nil.

```

```

Fixpoint update (s : store)
  (x : string) (v : val) : store :=
  match s with
  | nil => (x, v) :: nil
  | (x', v') :: rest =>
    if string_dec x x' then
      (x, v) :: rest
    else
      (x', v') :: update rest x v
  end.

```

```

Fixpoint updates (s : store)
  (xs : list string) (vs : list val) : option store :=
  match xs, vs with
  | nil, nil =>
    Some s
  | x :: xs', v :: vs' =>
    updates (update s x v) xs' vs'
  | _, _ => None
  end.

```

```

Fixpoint lkup (s : store)
  (x : string) : option val :=
  match s with
  | nil => None
  | (x', v) :: rest =>
    if string_dec x x' then
      Some v
    else
      lkup rest x
  end.

```

Oct 10, 16 8:43

ImpCommon.v

Page 2/2

```

(** Common *)

```

```

Fixpoint zlen (h : heap) : Z :=
  match h with
  | nil => 0
  | _ :: rest => 1 + zlen rest
  end.

```

```

Definition isNop (p : stmt) :
  {p = Snop} + {p <> Snop}.
Proof.
  destruct p; auto;
  right; congruence.
Qed.

```

```

Definition store_eq_dec (s1 s2 : store) :
  {s1 = s2} + {s1 <> s2}.

```

```

Proof.
  decide equality.
  decide equality.
- apply val_eq_dec.
- apply string_dec.
Qed.

```

```

Definition expr_eq_dec (e1 e2 : expr) :
  {e1 = e2} + {e1 <> e2}.

```

```

Proof.
  decide equality.
- apply val_eq_dec.
- apply string_dec.
- decide equality.
- decide equality.
Qed.

```

```

Fixpoint size_s (p : stmt) : nat :=
  match p with
  | Snop
  | Sset _ _ => 1
  | Sifelse _ p1 p2 => S (size_s p1 + size_s p2)
  | Swhile _ p1 => S (size_s p1)
  | Sseq p1 p2 => S (size_s p1 + size_s p2)
  end%nat.

```