

単一化の証明 (完全性)

1 実装

今日の実装は http://www.math.nagoya-u.ac.jp/~garrigue/lecture/2023_AW/ においてある。

(* 単一化 *)

```
From mathcomp Require Import all_ssreflect.
```

```
Lemma mem_tail {A:eqType} x a {l : seq A} : x \in l -> x \in a::l.
```

```
Proof. rewrite inE => ->; exact/orbT. Qed.
```

```
#[local] Hint Resolve mem_head mem_tail : core.
```

(* ... *)

(* subs_list と Fork が可換 *)

```
Lemma subs_list_Fork s t1 t2 :
```

```
  subs_list s (Fork t1 t2) = Fork (subs_list s t1) (subs_list s t2).
```

```
Proof. by elim: s t1 t2 => /=. Qed.
```

(* unifies_pairs の性質 *)

```
Lemma unifies_pairs_same s t l :
```

```
  unifies_pairs s l -> unifies_pairs s ((t,t) :: l).
```

```
Proof.
```

```
  move=> H t1 t2; rewrite inE => /orP[ ].
```

```
  - by case/eqP => -> ->.
```

```
  - exact/H.
```

```
Qed.
```

```
Lemma unifies_pairs_swap s t1 t2 l :
```

```
  unifies_pairs s ((t1, t2) :: l) -> unifies_pairs s ((t2, t1) :: l).
```

```
Proof.
```

```
  move=> H x y.
```

```
  rewrite inE => /orP[/eqP[-> ->] | H1].
```

```
  - exact/esym/H.
```

```
  - exact/H/mem_tail.
```

```
Qed.
```

(* 主補題 *)

```
Lemma unify_subs_sound h v t l s :
```

```
  (forall s l, unify2 h l = Some s -> unifies_pairs s l) ->
```

```
  unify_subs (unify2 h) v t l = Some s ->
```

```
  unifies_pairs s ((Var v, t) :: l).
```

Proof.

```
rewrite /unify_subs.
case Hocc: (v \in _) => // IH.
case Hun: (unify2 _ _) => [s' |] // = [] <-.
move=> t1 t2.
rewrite /unifies inE => /orP[/eqP[-> ->] | Hin] /=.
  by rewrite eqxx subs_same // Hocc.
apply (IH _ _ Hun).
exact: (map_f (subs_pair v t) Hin).
```

Qed.

(* unify2 の健全性 *)

Theorem unify2_sound h s l : unify2 h l = Some s -> unifies_pairs s l.

Proof.

```
elim: h s l => // = h IH s l.
move: (size_pairs l + 1) => h'.
elim: h' l => // = h' IH' [] //.
move=> [t1 t2] l.
destruct t1, t2 => //.
- case: ifP. (* VarVar *)
  by move=> /eqP <- /IH' /unifies_pairs_same.
  move=> _ /unify_subs_sound; exact.
- move/unify_subs_sound; exact. (* VarSym *)
- move/unify_subs_sound; exact. (* VarFork *)
- move/unify_subs_sound/unifies_pairs_swap; exact. (* SymVar *)
- by case: symbol_dec => // = -> /IH'/unifies_pairs_same. (* SymSym *)
- move/unify_subs_sound/unifies_pairs_swap; exact. (* ForkVar *)
- move/IH' => Hun t1 t2. (* ForkFork *)
  rewrite inE => /orP[/eqP[-> ->] | H1].
  rewrite /unifies !subs_list_Fork.
  rewrite (Hun t1_1 t2_1) //.
  rewrite (Hun t1_2 t2_2); by auto.
  apply Hun; by auto.
```

Qed.

(* 単一化の健全性 *)

Corollary soundness t1 t2 s : unify t1 t2 = Some s -> unifies s t1 t2.

Proof. move/unify2_sound; exact. Qed.

(* 完全性 *)

Lemma not_unifies_occur v t s : (* 循環的な項が作れない *)

Var v != t -> v \in vars t -> ~ unifies s (Var v) t.

Proof.

```
rewrite /unifies.
move=> vt Ht Hun.
(* size_tree で矛盾を導く *)
have Hs: size_tree (subs_list s (Var v)) >= size_tree (subs_list s t).
  by rewrite Hun.
(* 元の仮定を消してから帰納法を使う *)
elim: t {Hun} vt Ht Hs => // = [v' | t1 IH1 t2 IH2] vt.
Admitted.
```

(* 集合の要素の数に関する基礎的な補題 *)

Lemma size_union2 l1 l2 : size (union l1 l2) >= size l2. Admitted.

(* Sym の代入 *)

Lemma subs_list_Sym s f : subs_list s (Sym f) = Sym f. Admitted.

(* 代入の合成 *)

Lemma unifies_extend s v t t' :
unifies s (Var v) t -> unifies s (subs v t t') t'.
Admitted.

Lemma unifies_pairs_extend s v t l :
unifies_pairs s ((Var v, t) :: l) ->
unifies_pairs s (map (subs_pair v t) l).

Proof.

move=> H t1 t2 /mapP /= [] [t3 t4] H1 [-> ->].

Admitted.

Lemma unifies_pairs_Fork s t1 t2 t'1 t'2 l :
unifies s (Fork t1 t2) (Fork t'1 t'2) ->
unifies_pairs s l ->
unifies_pairs s ((t1,t'1)::(t2,t'2)::l).

Admitted.

(* s が s' より一般的な単一子である *)

Definition moregen s s' :=

exists s2, forall t, subs_list s' t = subs_list s2 (subs_list s t).

(* 一般性を保ちながら拡張 *)

Lemma moregen_extend s v t s1 :

unifies s (Var v) t -> moregen s1 s -> moregen ((v, t) :: s1) s.

Admitted.

(* 変数の数に関する補題 *)

(* 面倒なので、後で証明して良い *)

Lemma subs_del x t t' : x \notin vars t -> x \notin vars (subs x t t').

Admitted.

Lemma subs_pairs_del x t l :

x \notin vars t -> x \notin (vars_pairs (map (subs_pair x t) l)).

Admitted.

Lemma subs_sub x t t' :

{subset vars (subs x t t') <= union (vars t) (vars t')}.

Admitted.

Lemma subs_pairs_sub x t l :

{subset vars_pairs (map (subs_pair x t) l) <= union (vars t) (vars_pairs l)}.

Admitted.

Lemma uniq_vars_pairs l : uniq (vars_pairs l). Admitted.

Check uniq_leq_size.

Lemma vars_pairs_decrease x t l :

x \notin (vars t) ->

```

    size (vars_pairs (map (subs_pair x t) l)) <
    size (vars_pairs ((Var x, t) :: l)).
Admitted.

Lemma size_vars_pairs_swap t1 t2 l :
  size (vars_pairs ((t1,t2) :: l)) = size (vars_pairs ((t2,t1) :: l)).
Admitted.

Lemma size_vars_pairs_Fork t1 t2 t'1 t'2 l :
  size (vars_pairs ((Fork t1 t2, Fork t'1 t'2) :: l)) =
  size (vars_pairs ((t1, t'1) :: (t2, t'2) :: l)).
Admitted.
(* 変数の数に関する補題はここまで *)

Lemma unify_subs_complete s h v t l :
  (forall l,
    h > size (vars_pairs l) -> unifies_pairs s l ->
    exists s1, unify2 h l = Some s1 /\ moregen s1 s) ->
  h.+1 > size (vars_pairs ((Var v, t) :: l)) ->
  unifies_pairs s ((Var v, t) :: l) ->
  Var v != t ->
  exists s1, unify_subs (unify2 h) v t l = Some s1 /\ moregen s1 s.
Admitted.

(* 完全性 *)
Theorem unify2_complete s h l :
  h > size (vars_pairs l) ->
  unifies_pairs s l ->
  exists s1, unify2 h l = Some s1 /\ moregen s1 s.
Proof.
  elim: h l => // = h IH l Hh.
  move Hh': (size_pairs l + 1) => h'.
  have {Hh'} : h' > size_pairs l.
  by rewrite -Hh' addn1 ltnS.
  elim: h' l Hh => // = h' IH' [] // =.
  move=>*; exists nil; split => //; by exists s.
  move=> [t1 t2] l Hh Hh' Hs.
  destruct t1, t2 => /=.
Admitted.

(* 短い完全性定理 *)
Corollary unify_complete s t1 t2 :
  unifies s t1 t2 ->
  exists s1, unify t1 t2 = Some s1 /\ moregen s1 s.
Admitted.

```

練習問題 1.1 証明の中の Admitted を Qed に変えよ.