Jacques Garrigue, 2014 年 7 月 28 日

単一化の完全性

1 停止性

単一化は本来、書き換えの繰り返しという形で与えられているので、その停止性（アルゴリズムが必ず止まること）が自明ではない。

$$
\begin{array}{lcll}
(E \cup \{f(t_1,\ldots,t_n) = f(t'_1,\ldots,t'_n)\}, \sigma) & \to & (E \cup \{t_1 = t'_1, \ldots, t_n = t'_n\}, \sigma) & \\
(E \cup \{f(t_1,\ldots,t_n) = g(t'_1,\ldots,t'_m)\}, \sigma) & \to & \bot & f \neq g \\
(E \cup \{x = t\}, \sigma) & \to & ([t/x]E, [t/x] \circ \sigma) & x \notin vars(t) \\
(E \cup \{x = x\}, \sigma) & \to & (E, \sigma) & \\
(E \cup \{x = t\}, \sigma) & \to & \bot & x \in vars(t) \wedge x \neq t
\end{array}
$$

前回は残るステップ数の上限を表す変数を渡すことで、取りあえず Fixpoint で定義できるようにした。しかし、その上限が十分であることを証明しなければならない。

使っている上限は $\langle E$ の変数の数, E 全体の文字数 $\rangle$ に対する辞書式順序。すなわち、この順序は整礎なので、停止性を証明するために、各ステップで変数の数が減るか、または変数の数が変わらずに全体の文字数が減っていることを確認すればいい。

3 番目の規則では x が t に入っていないので、$[t/x]E$ には x が入っていない。また、x および t の変数が $E \cup \{x = t\}$ に入っていたので、変数の数が減っている。他の規則では、等式を分解することで、記号の数が減っている。

2 完全性

前回説明したように、以下の定理が成り立つ。

定理 1 任意の単一化問題に対して，解が存在するときには**最も一般的な単一子**を返し，存在しないときにはそれを報告するアルゴリズムが存在する．

もっと具体的に書くと、

定理 1 もしも E が解 σ を認めれば、$\mathcal{U}(E) = \sigma_1$ となるような σ_1 が単一化アルゴリズムにより計算され、しかも σ_1 が σ より一般的である。

今回はこの性質を証明する。

3 実装

今日の実装は `http://www.math.nagoya-u.ac.jp/~garrigue/lecture/2014_SS/` においてある．

```
(* 前回からの修正分 *)

(* 和集合 *)
Definition union : list var -> list var -> list var := set_union eq_nat_dec.

(* 出現変数 *)
Fixpoint vars (t : tree) : list var :=
  match t with
  | Var x => x :: nil
  | Sym _ => nil
  | Fork t1 t2 => union (vars t1) (vars t2)
  end.

(* 変数を集める *)
Fixpoint vars_pairs (l : list (tree * tree)) : list var :=
  match l with
  | nil => nil
                    (* 集合を後部から作るように変更 *)
  | (t1, t2) :: r => union (vars_pairs r) (union (vars t1) (vars t2))
  end.

(* 変数の数だけ unify2 を繰り返す *)
Definition unify t1 t2 :=
  let l := (t1,t2)::nil in unify2 (length (vars_pairs l) + 1) l.


(** 完全性 **)
Require Import Omega. (* 停止性を証明するために *)

(* 循環的な項が作れない *)
Lemma not_unifies_occur : forall v t s,
  Var v <> t -> In v (vars t) -> ~unifies s (Var v) t.
Proof.
  intros.
  unfold unifies; intro Hun.
  (* size_tree で矛盾を導く *)
  assert (Hs: size_tree (subs_list s (Var v)) >= size_tree (subs_list s t)).
    rewrite Hun; auto with arith.
  (* 元の仮定を消してから帰納法を使う *)
  clear Hun. induction t; simpl in *.
Admitted.

(* 集合の要素の数に関する基礎的な補題 *)
Parameter length_set_add : forall a l,
  length (set_add eq_nat_dec a l) >= length l.

Check le_trans.
Check le_lt_trans.
Parameter length_union1 : forall l1 l2, length (union l1 l2) >= length l1.

(* Sym の代入 *)
Parameter subs_list_Sym : forall s f, subs_list s (Sym f) = Sym f.

(* 代入の合成 *)
Parameter unifies_extend : forall s v t t',
  unifies s (Var v) t -> unifies s (subs v t t') t'.

Check in_map_iff.
```

```
Lemma unifies_pairs_extend : forall s v t l,
  unifies_pairs s ((Var v, t) :: l) -> unifies_pairs s (map (subs_pair v t) l).
Proof.
  unfold unifies_pairs, unifies in *.
  intros.
  rewrite in_map_iff in H0. (* apply の代わりに rewrite が使える *)
Admitted.

Parameter unifies_pairs_Fork : forall 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).


(* 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 *)
Parameter moregen_extend : forall s v t s1,
  unifies s (Var v) t -> moregen s1 s -> moregen ((v, t) :: s1) s.

(* 変数の数に関する補題 *)
(* 難しいので、証明しなくて良い *)
Parameter vars_pairs_decrease : forall x t l,
  ~In x (vars t) ->
  length (vars_pairs (map (subs_pair x t) l))
    < length (vars_pairs ((Var x, t) :: l)).
Parameter length_vars_pairs_swap : forall t1 t2 l,
  length (vars_pairs ((t1,t2) :: l)) = length (vars_pairs ((t2,t1) :: l)).
Parameter length_vars_pairs_Fork : forall t1 t2 t'1 t'2 l,
  length (vars_pairs ((Fork t1 t2, Fork t'1 t'2) :: l)) =
  length (vars_pairs ((t1, t'1) :: (t2, t'2) :: l)).

(* 完全性 *)
Theorem unify2_complete : forall s h l,
  h > length (vars_pairs l) ->
  unifies_pairs s l ->
  exists s1, unify2 h l = Some s1 /\ moregen s1 s.
Proof.
  induction h.
    intros. elimtype False. omega.
  simpl.
  intros l Hh.
  remember (size_pairs l + 1) as h'.
  assert (Hh': h' > size_pairs l) by (subst; omega).
  clear Heqh'.
  revert l Hh Hh'.
  induction h'; intros.
    elimtype False. omega.
  simpl.
  destruct l.
    exists nil; split; auto.
    exists s; reflexivity.
  destruct p.
  destruct t; destruct t0.
  (* VarVar *)
  destruct (v == v0).
    subst v0.
```

```
    simpl in *.
    apply IHh'; try omega.
      eapply le_lt_trans; try apply Hh.
      apply length_union1.
    intros t1 t2 Hin. apply H; simpl; auto.
  assert (Var v <> Var v0).
    intro.
    elim n.
    inversion H0; reflexivity.
Lemma unify_subs_complete : forall s h v t l,
  (forall l,
    h > length (vars_pairs l) -> unifies_pairs s l ->
    exists s1, unify2 h l = Some s1 /\ moregen s1 s) ->
  S h > length (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.
Proof.
  intros until l; intros IHh Hh Hs Hv.
  unfold unify_subs.
Admitted.
  apply unify_subs_complete; intuition.
  (* VarSym *)
  apply unify_subs_complete; intuition.
  discriminate.
Admitted.

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

練習問題 3.1 証明の中の Parameter を Theorem に変え，Admitted を Qed に変えよ．ただし，vars_pairs_decrease 以下の Parameter は証明しなくて良い．