Jacques Garrigue, 2014 年 6 月 23 日

証明論の完全性

1 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.
(先週の解答を含めて)

```
(* あるリテラルだけを false にする割り当て *)
Definition neg_val v a : {v' | eval_lit v' a = false /\
                               forall b, b <> atom_lit a -> v' b = v b}.
  destruct a;
    [ exists (fun x => if x == a then false else v x)
    | exists (fun x => if x == a then true else v x) ]; simpl;
    (split; intros; [destruct (a == a) | destruct (b == a)]; intuition).
Defined.

(* 完全性の証明 *)

(* tt と ff *)
Lemma excluded_middle : forall h p, h |- Disj p (Neg p).
Proof.
  intros.
  apply NegE.
  apply (NegI p).
    apply NegE.
    apply (NegI (Disj p (Neg p))); auto.
  apply (NegI (Disj p (Neg p))); auto.
Qed.
Hint Resolve excluded_middle.

Parameter ex_falso : forall p q h, h |- Conj p (Neg p) -> h |- q.

(* nnf への変換が証明可能性を保存する *)
Lemma provable_nnf : forall p h, h |- p <-> h |- nnf p.
Proof.
  intro.
  functional induction (nnf p); simpl; split; intros; auto;
    try apply -> IHp0; try apply <- IHp0 in H.
  (* 1 *)
  auto.
  (* 2 *)
  apply (NegI p1). destruct (IHp0 h); auto.
  auto.
  (* 3 *)
  apply NegE.
  apply (NegI (Conj p1 p2)); auto.
  apply ConjI.
    apply NegE.
    apply (NegI (Disj (Neg p1) (Neg p2))); auto.
  apply NegE.
  apply (NegI (Disj (Neg p1) (Neg p2))); auto.
Admitted.

(* Clause に関する完全性 *)
Parameter true_clause : forall h,
  tauto_clause h = true -> exists b, In (Posi b) h / In (Nega b) h.

Fixpoint prop_of_lit (l : lit) :=
  match l with
  | Posi a => Atom a
  | Nega a => Neg (Atom a)
  end.

Definition prop_of_clause :=
  fold_right (fun x => Disj (prop_of_lit x)) ff.

Definition prop_of_cnf :=
  fold_right (fun x => Conj (prop_of_clause x)) tt.

Parameter prop_of_cnf_ok : forall v c,
  eval_cnf v c = eval v (prop_of_cnf c).

Parameter provable_clause : forall a c h,
  In a c -> In (prop_of_lit a) h -> h |- prop_of_clause c.

(* CNF の clause 表記に関する完全性 *)
Parameter completeness_cnf : forall p,
  (forall v, eval_cnf v p = true) -> nil |- prop_of_cnf p.

(* 必要な補題の一部 *)
Parameter provable_clause_weaken_l : forall b a h,
  h |- prop_of_clause a -> h |- prop_of_clause (b ++ a).
Parameter provable_clause_weaken_r : forall b a h,
  h |- prop_of_clause a -> h |- prop_of_clause (a ++ b).
Parameter provable_cnf_app : forall c1 c2 h,
  h |- prop_of_cnf c1 ->
  h |- prop_of_cnf c2 -> h |- prop_of_cnf (c1 ++ c2).
Parameter provable_cnf_app_inv : forall c1 c2 h,
  h |- prop_of_cnf (c1 ++ c2) ->
  h |- prop_of_cnf c1 /\ h |- prop_of_cnf c2.
Parameter prop_of_clause_app_inv : forall c1 c2 h,
  h |- prop_of_clause (c1 ++ c2) ->
  h |- Disj (prop_of_clause c1) (prop_of_clause c2).
Parameter provable_cnf_weaken_r : forall a h f,
  h |- prop_of_cnf f ->
  h |- prop_of_cnf (map (fun c => c ++ a) f).
Parameter provable_cnf_disj_l : forall c1 c2 h,
  h |- prop_of_cnf c1 -> h |- prop_of_cnf (disj c1 c2).
Parameter provable_disj_conj : forall a b c h,
  h |- Conj (Disj a c) (Disj b c) <-> h |- Disj (Conj a b) c.
Parameter provable_cnf_distr : forall a h f,
  h |- prop_of_cnf (map (fun c : list lit => a ++ c) f) ->
  h |- Disj (prop_of_clause a) (prop_of_cnf f).

(* 論理和の証明の逆変換 *)
Parameter provable_cnf_disj : forall c1 c2 h,
  h |- prop_of_cnf (disj c1 c2) ->
  h |- Disj (prop_of_cnf c1) (prop_of_cnf c2).

(* CNF での証明の逆変換 *)
Lemma provable_cnf : forall h p,
  is_nnf p -> h |- prop_of_cnf (cnf p) -> h |- p.
Proof.
  intros h p Hnnf. revert h.
  induction Hnnf; simpl; intros.
Admitted.

(* 完全性 *)
Theorem completeness : forall p, tautology p -> nil |- p.
Proof.
  intros.
  apply <- provable_nnf.
  apply provable_cnf; auto.
  apply completeness_cnf.
  intros.
  rewrite cnf_correct, nnf_correct; auto.
Qed.
(* 証明論の決定可能性 *)
Parameter provable_dec : forall p, {nil |- p}+{~nil |- p}.
```