

Filtr.v

```
1 Require Import Coq.Arith.PeanoNat.
2 Require Import Coq.Lists.List.
3 Require Import Autosubst.Autosubst.
4
5 Require Import PrincInh.Terms.
6 Require Import PrincInh.Types.
7 Require Import PrincInh.Typing.
8 Require Import PrincInh.Utills.
9
10 Import ListNotations.
11 Import EqNotations.
12
13 (* Given a type derivation D we define the set  $T(D) = \{ \tau \mid \Gamma \vdash M : \tau \text{ is a}$ 
14     $\hookrightarrow$  judgement in D} *)
15 Inductive TD (Gamma : list type) : forall (m : term) (tau : type), ty_T Gamma m
16     $\hookrightarrow$  tau  $\rightarrow$  type  $\rightarrow$  Type :=
17    | TD_refl m tau : forall (proof : ty_T Gamma m tau), TD Gamma m tau proof tau
18    | TD_lam s sigma tau :
19      forall (tau' : type) (innerproof : ty_T (sigma :: Gamma) s tau),
20      TD (sigma :: Gamma) s tau innerproof tau'
21       $\rightarrow$  TD Gamma ( $\lambda$  s) (sigma  $\rightarrow$  tau) (Ty_Lam Gamma s sigma tau innerproof)
22       $\hookrightarrow$  tau'
23    | TD_app_l p q sigma tau :
24      forall (tau' : type) (leftproof : ty_T Gamma p (sigma  $\rightarrow$  tau)) (rightproof :
25         $\hookrightarrow$  ty_T Gamma q sigma),
26      TD Gamma p (sigma  $\rightarrow$  tau) leftproof tau'
27       $\rightarrow$  TD Gamma (p @ q) tau (Ty_App Gamma p q sigma tau leftproof
28         $\hookrightarrow$  rightproof) tau'
29    | TD_app_r p q sigma tau :
30      forall (tau' : type) (leftproof : ty_T Gamma p (sigma  $\rightarrow$  tau)) (rightproof :
31         $\hookrightarrow$  ty_T Gamma q sigma),
32      TD Gamma q sigma rightproof tau'
33       $\rightarrow$  TD Gamma (p @ q) tau (Ty_App Gamma p q sigma tau leftproof
34         $\hookrightarrow$  rightproof) tau'
35
36 .
37
38 Lemma app_eq_l {m1 m2 m3 m4} : m1 @ m2 = m3 @ m4  $\rightarrow$  m1 = m3.
39 Proof.
40   intros. ainv.
41 Qed.
42
43 Lemma app_eq_r {m1 m2 m3 m4} : m1 @ m2 = m3 @ m4  $\rightarrow$  m2 = m4.
44 Proof.
45   intros. ainv.
46 Qed.
47
48 Lemma lam_eq {s1 s2} :  $\lambda$  s1 =  $\lambda$  s2  $\rightarrow$  s1 = s2.
49 Proof.
50   intros.
51 Qed.
52
53 Lemma tyrewew {Gamma m1 m2 m1' m2' sigma tau} : forall (eq : m1 @ m2 = m1' @ m2')
54    $\hookrightarrow$  (proof1 : ty_T Gamma m1' (sigma  $\rightarrow$  tau)) (proof2 : ty_T Gamma m2' sigma),
```

```

47   Ty_App Gamma m1' m2' sigma tau proof1 proof2 =
48   rew[fun m => ty_T Gamma m tau] eq in
49   Ty_App Gamma m1 m2 sigma tau
50   (rew -[fun m => ty_T Gamma m (sigma ~> tau)] app_eq_l eq in
51   (rew -[fun m => ty_T Gamma m sigma] app_eq_r eq in proof2)).

52 Proof.
53   intros.
54   revert proof1 proof2.
55   inversion eq.
56   revert eq.
57   rewrite H1.
58   rewrite H0.
59   intro eq.
60   rewrite ← (Coq.Logic.Eqdep_dec.UIP_dec eq_dec_term eq_refl eq).
61   rewrite ← (Coq.Logic.Eqdep_dec.UIP_dec eq_dec_term eq_refl (app_eq_l
62   ↪ eq_refl)).
63   rewrite ← (Coq.Logic.Eqdep_dec.UIP_dec eq_dec_term eq_refl (app_eq_r
64   ↪ eq_refl)).
65   unfold eq_rect_r.
66   reflexivity.

67 Qed.

68 Lemma False_Ty : forall (T : Type), False → T.
69 Proof.
70   intros.
71   exfalso.
72   apply H.
73 Qed.

74 Fixpoint filtration (X : type → bool) (a : var) (rho : type) :=
75   match rho with
76   | ? b => ? a
77   | sigma ~> tau => if ( andb (X (sigma ~> tau)) (X tau)) then
78     (filtration X a sigma) ~> (filtration X a tau)
79   else
80     ? a
81   end.

82 Definition repo_filt (X : type → bool) (a : var) : repo → list type :=
83   map (filtration X a).

84 Lemma filt_split : forall sigma tau X a, X tau = true → X (sigma ~> tau) = true
85   → filtration X a (sigma ~> tau) = (filtration X a sigma)
86   ↪ ~> (filtration X a tau).

87 Proof.
88   intros.
89   unfold filtration. rewrite H. rewrite H0. reflexivity.
90 Qed.

91 Lemma repo_filt_split : forall X a rho Gamma, repo_filt X a (rho :: Gamma) =
92   ↪ ((filtration X a rho) :: (repo_filt X a (Gamma))).

93 Proof.
94   intros.
95   unfold repo_filt.
96   reflexivity.

```

```

98 Qed.
99
100 Fixpoint TD_f {Gamma m tau} (proof : ty_T Gamma m tau) : list type :=
101   match proof with
102   | Ty_Var _ x rho eqproof  $\Rightarrow$  [rho]
103   | Ty_Lam _ s sigma tau' innerproof  $\Rightarrow$  (sigma  $\leadsto$  tau') :: TD_f innerproof
104   | Ty_App _ s t A B proof1 proof2  $\Rightarrow$  B::(TD_f proof1 ++ TD_f proof2)
105   end.
106
107
108 Lemma TD_last {Gamma m tau}: forall (proof : ty_T Gamma m tau), In tau (TD_f
   $\hookrightarrow$  proof).
109 Proof.
110   intros.
111   destruct proof.
112   - simpl. left. reflexivity.
113   - simpl. left. reflexivity.
114   - simpl. left. reflexivity.
115 Qed.
116
117
118
119 Lemma filter_deriv {m Gamma tau}: forall (X : type  $\rightarrow$  bool) (proof : ty_T Gamma
   $\hookrightarrow$  m tau),
120   (forall tau', In tau' (TD_f proof)  $\rightarrow$  X tau' = true)  $\rightarrow$  (forall a, ty_T
     $\hookrightarrow$  (repo_filt X a Gamma) m (filtration X a tau)).
121 Proof.
122   intros.
123   induction proof.
124   - constructor. unfold repo_filt.
125     apply map_nth_error. assumption.
126   - rewrite filt_split.
127     + constructor. rewrite  $\leftarrow$  repo_filt_split. apply IHproof. intros. apply H.
128        $\hookrightarrow$  simpl. right. assumption.
129     + apply H. simpl. right. apply TD_last.
130     + apply H. simpl. left. reflexivity.
131   - econstructor.
132     + instantiate (1:=filtration X a A). rewrite  $\leftarrow$  filt_split.
133       * apply IHproof1. intros. apply H. simpl. right. apply in_or_app. left.
134          $\hookrightarrow$  assumption.
135       * apply H. simpl. left. reflexivity.
136       * apply H. simpl. right. apply in_or_app. left. apply TD_last.
137     + apply IHproof2. intros. apply H. simpl. right. apply in_or_app. right.
138        $\hookrightarrow$  assumption.
139 Qed.
140
141
142 Lemma In_TD_dec {Gamma m tau} : forall (deriv : ty_T Gamma m tau) tau', {In tau'
   $\hookrightarrow$  (TD_f deriv)} + {~(In tau' (TD_f deriv))}.
143 Proof.
144   intros.
145   apply In_dec.
146   apply eq_dec_type.
147 Defined.
148
149
150 Definition TD_b {Gamma m tau} (deriv : ty_T Gamma m tau) tau' : bool :=
151   if (In_TD_dec deriv tau') then true else false.

```

```

147
148 Lemma TD_b_corr {Gamma m tau} {proof : ty_T Gamma m tau}: (forall tau' : type,
    ↪ In tau' (TD_f proof) → TD_b proof tau' = true) .
149 Proof.
150   intros.
151   unfold TD_b.
152   destruct (In_TD_dec proof tau') eqn:HIn.
153   + reflexivity.
154   + contradiction.
155 Qed.
156
157 Lemma filt_mtTy :forall a X, repo_filt a X [] = [].
158 Proof.
159   auto.
160 Qed.
161
162 Lemma subformula_subst : forall tau rho Su, subformula rho tau → subformula
    ↪ rho.[Su] tau.[Su].
163 Proof.
164   induction 1.
165   - constructor.
166   - constructor. assumption.
167   - constructor 3. assumption.
168 Qed.
169
170 Lemma subst_subformula : forall sigma tau rho, subformula (sigma ↪ tau) rho →
    ↪ forall X a Su, rho.[Su] = filtration X a rho →
171                                     (sigma ↪ tau).[Su] = filtration X a
    ↪ (sigma ↪ tau).
172 Proof.
173   intros.
174   remember (sigma ↪ tau) as tau0.
175   induction H.
176   - ainv.
177   - apply IHsubformula.
178     + assumption.
179     + inversion H0.
180     destruct ((X (sigma0 ↪ tau0) && X tau0)%bool).
181     * ainv.
182     * ainv.
183   - apply IHsubformula.
184     + assumption.
185     + inversion H0.
186     destruct ((X (sigma0 ↪ tau0) && X tau0)%bool).
187     * ainv.
188     * ainv.
189 Qed.
190
191 Lemma princ_var : forall A x, princ A (! x) → False.
192 Proof.
193   intros.
194   unfold princ in X.
195   destruct X.
196   inv t.
197   rewrite nth_error_nil in H0.
198   ainv.

```

```

199 Qed.
200
201 Lemma filter_princ_nec : forall m rho (D : ty_T [] m rho) sigma tau, subformula
     $\hookrightarrow$  (sigma  $\rightsquigarrow$  tau) rho  $\rightarrow$  (TD_b D tau = false)  $\rightarrow$  princ rho m  $\rightarrow$  False.
202 Proof.
203   intros.
204   pose proof filter_deriv _ D TD_b_corr (first_fresh_type rho) as filtD.
205   simpl in filtD.
206   unfold princ in X.
207   destruct X as [alsoD Hsubst].
208   pose proof Hsubst _ filtD as [Su sucorr].
209   pose proof subst_subformula sigma tau rho H (TD_b D) (first_fresh_type rho) Su
     $\hookrightarrow$  sucorr.
210   simpl in H1. rewrite H0 in H1. rewrite Bool.andb_false_r in H1. inversion H1.
211 Qed.

```