

## Terms2.v

```
1 Require Import List.
2
3 Require Import Autosubst.Autosubst.
4
5 Require Import PrincInh.Terms.
6
7 Import ListNotations.
8
9 Inductive term2 :=
10 | App_var (ms : list term2) (x : var)
11 | App_lam (ms : list term2) (s : {bind term2})
12 .
13
14 Definition term2_ind' : forall P : term2 → Prop,
15   (forall (ms : list term2) (x : var),
16     (Forall P ms) →
17     P (App_var (ms) x)) →
18   (forall (ms : list term2) (s : {bind term2}),
19     Forall P ms →
20     P s → P (App_lam ms s)) →
21   forall t : term2, P t :=
22 fun P app_var_case app_lam_case ⇒
23   fix term2_ind'_rec (t : term2) :=
24     match t with
25     | App_var ms x ⇒ app_var_case ms x ((fix term2_ind'_var_rec (ms : list
26       ↪ term2) : Forall P ms :=
27         match ms with
28         | [] ⇒ Forall_nil _
29         | t::ts ⇒ @Forall_cons _ _ t ts (term2_ind'_rec t)
30           ↪ (term2_ind'_var_rec ts)
31         end) ms)
32     | App_lam ms s ⇒ app_lam_case ms s ((fix term2_ind'_lam_rec (ms : list
33       ↪ term2) : Forall P ms :=
34         match ms with
35         | [] ⇒ Forall_nil _
36         | t::ts ⇒ @Forall_cons _ _ t ts (term2_ind'_rec t)
37           ↪ (term2_ind'_lam_rec ts)
38         end) ms) (term2_ind'_rec s)
39     end.
40
41 Notation "'!!!" x" := (App_var [] x) (at level 15).
42 Notation "'!!' p '@@' q" := (App_var q p) (at level 31, left associativity).
43 Notation "'\___' p" := (App_lam [] p) (at level 35, right associativity).
44 Notation "'\___' p '@@' q" := (App_lam q p) (at level 35, right associativity).
45
46 Fixpoint term2_term (m : term2) : term :=
47   match m with
48   | !! x @@ q ⇒ curry (! x) (map term2_term q)
49   | [ ] s @@ q ⇒ curry ([ ] (term2_term s)) (map term2_term q)
50   end.
51
52 Fixpoint term_term2 (m : term) : term2 :=
53   match m with
```

```

51 | !x ⇒ !!!x
52 |  $\lambda$ _ s ⇒  $\lambda$ ____ (term_term2 s)
53 | p @ q ⇒ let p' := term_term2 p in
54     match p' with
55     | !!x @ q' ⇒ !!x @ (q' ++ [term_term2 q])
56     |  $\lambda$ _ s @ q' ⇒  $\lambda$ _ s @ (q' ++ [term_term2 q])
57     end
58 end.

```

59 **Lemma** term2\_term\_id : forall m, (term2\_term >>> term\_term2) m = m.

60 **Proof.**

```

61 induction m using term2_ind'.
62 - unfold "_ >>>_" in *.
63   simpl in *. rewrite (Forall_forall _ ms)in H.
64   induction ms using rev_ind.
65   + reflexivity.
66   + rewrite map_app. simpl. rewrite curry_tail. simpl.
67     rewrite IHms.
68     { rewrite (H x0).
69       - reflexivity.
70       - apply (in_or_app). right. constructor. reflexivity. }
71     { intros.
72       apply H. apply (in_or_app). left. assumption.
73     }
74 - unfold "_ >>>_" in *.
75   simpl in *. rewrite (Forall_forall _ ms) in H.
76   induction ms using rev_ind.
77   + simpl. rewrite IHm. reflexivity.
78   + rewrite map_app. simpl. rewrite curry_tail. simpl.
79     rewrite IHms.
80     { rewrite (H x).
81       - reflexivity.
82       - apply (in_or_app). right. constructor. reflexivity. }
83     { intros.
84       apply H. apply (in_or_app). left. assumption. }
85   }

```

86 **Qed.**

87 **Lemma** term\_term2\_id : forall m, (term\_term2 >>> term2\_term) m = m.

88 **Proof.**

```

89 unfold "_ >>>_".
90 induction m.
91 - reflexivity.
92 - simpl. destruct (term_term2 m1) eqn:Htm1.
93   + simpl. rewrite map_app. simpl. rewrite curry_tail.
94     rewrite IHm2. ainv.
95   + simpl. rewrite map_app. simpl. rewrite curry_tail.
96     rewrite IHm2. ainv.
97   - simpl. rewrite IHm. reflexivity.
98 
```

99 **Qed.**

100 **Lemma** term2\_term\_if : forall m n, term2\_term m = term2\_term n → m = n.

101 **Proof.**

```

102 intros.
103 rewrite ← (term2_term_id m).
104 rewrite ← (term2_term_id n).
105 
```

```

106   unfold "_ >>> _".
107   rewrite H.
108   reflexivity.
109 Qed.
110
111 Lemma term_term2_if : forall m n, term_term2 m = term_term2 n → m = n.
112 Proof.
113   intros.
114   rewrite ← (term_term2_id m).
115   rewrite ← (term_term2_id n).
116   unfold "_ >>> _".
117   rewrite H.
118   reflexivity.
119 Qed.
120
121 Instance Ids_term2 : Ids term2. unfold Ids. apply (App_var []). Defined.
122 Instance Rename_term2 : Rename term2 (*:= fun xi ⇒ (term2_term >>> Rename_term
123   ↪ xi >>> term_term2). *)
124 := fix ren_term2 (xi: var → var) (s : term2) {struct s} : term2
125   := match s as t return (annot term2 t) with
126     | !! x @@ ms ⇒ !! (xi x) @@ (map (ren_term2 xi) ms)
127     | [ ] s @@ ms ⇒ [ ] (ren_term2 (upren xi) s) @@ (map (ren_term2
128       ↪ xi) ms)
129   end.
130
131 Instance Subst_term2 : Subst term2 (*:= fun sigma ⇒ (term2_term >>> subst
132   ↪ (sigma >>> term2_term) >>> term_term2). *)
133 :=
134 fix dummy (sigma : var → term2) (s : term2) {struct s} : term2 :=
135   match s as t return (annot term2 t) with
136   | !! x @@ ms ⇒ match sigma x with
137     | !! y @@ ns ⇒ !! y @@ (ns ++ map (dummy sigma) ms)
138     | [ ] s @@ ns ⇒ [ ] s @@ (ns ++ map (dummy sigma) ms)
139   end
140   | [ ] s0 @@ ms ⇒ [ ] dummy (up sigma) s0 @@ map (dummy sigma) ms
141   end.
142
143 Lemma rename_subst_term2 (xi : var → var) (s : term2) :
144   rename xi s = s.[ren xi].
145 Proof.
146   intros.
147   revert xi.
148   induction s using term2_ind'.
149   - intros. simpl. rewrite Forall_forall in H. erewrite map_ext_in.
150     + reflexivity.
151     + intros. apply H. assumption.
152   - intros. simpl. rewrite Forall_forall in H.
153     rewrite IHs. rewrite up_upren_internal.
154     + erewrite map_ext_in.
155       { reflexivity. }
156       { intros. apply H. assumption. }
157     + auto.
158 Defined.
159
160 Lemma subst_up_ids : up ids = ids.

```

```

158 Proof.
159   unfold up.
160   unfold ids.
161   unfold "_ >>> _".
162   f_ext.
163   induction x.
164   - reflexivity.
165   - simpl. simpl in IHx. auto.
166 Qed.
167
168 Lemma subst_id_term2 (s : term2) :
169   s.[ids] = s.
170 Proof.
171   induction s using term2_ind'.
172   - simpl. rewrite Forall_forall in H. erewrite map_ext_in.
173     + instantiate (1:=id). rewrite map_id. reflexivity.
174     + auto.
175   - simpl. rewrite subst_up_ids. unfold subst in IHs. rewrite IHs.
176     rewrite Forall_forall in H. erewrite map_ext_in.
177     + instantiate (1:=id). rewrite map_id. reflexivity.
178     + auto.
179 Qed.
180
181 Lemma id_subst_term2 (sigma : var → term2) (x : var) :
182   (ids x).[sigma] = sigma x.
183 Proof.
184   simpl.
185   destruct (sigma x);
186   rewrite app_nil_r;
187   reflexivity.
188 Qed.
189
190 Lemma term2_var_app_split_eq : forall x y ms ns, x = y → ms = ns → !! x @@ ms
191   ⇔ = !! y @@ ns.
192 Proof.
193   intros. subst. reflexivity.
194 Qed.
195
196 Lemma term2_var_app_apps_eq : forall x ms ns, ms = ns → !! x @@ ms = !! x @@
197   ⇔ ns.
198 Proof.
199   intros. subst. reflexivity.
200 Qed.
201
202 Lemma term2_lam_app_split_eq : forall x y ms ns, x = y → ms = ns → λ__ x @@ ms
203   ⇔ = λ__ y @@ ns.
204 Proof.
205   intros. subst. reflexivity.
206 Qed.
207
208 Lemma term2_lam_app_apps_eq : forall x ms ns, ms = ns → λ__ x @@ ms = λ__ x @@
209   ⇔ ns.
210 Proof.
211   intros. subst. reflexivity.
212 Qed.

```

```

209
210 Lemma list_split_eq_l {A}: forall (l : list A) l1 l2, l1 = l2 → l ++ l1 = l ++
    ↪ l2.
211 Proof.
212   intros. subst. reflexivity.
213 Qed.
214
215 Lemma ren_subst_comp_term2 : forall xi sigma (s : term2), (rename xi s).[sigma]
    ↪ = s.[xi >>> sigma].
216 Proof.
217 fix ih 3. intros xi sigma s. destruct s.
218   - simpl. destruct (sigma (xi x)) eqn:H.
219     + apply term2_var_app_apps_eq. apply list_split_eq_l. unfold subst in ih.
220       rewrite map_map. induction ms.
221         { reflexivity. }
222         { simpl. rewrite ih. rewrite IHms. reflexivity. }
223     + apply term2_lam_app_apps_eq. apply list_split_eq_l. unfold subst in ih.
224       rewrite map_map. induction ms.
225         { reflexivity. }
226         { simpl. rewrite ih. rewrite IHms. reflexivity. }
227   - simpl. apply term2_lam_app_split_eq.
228     + rewrite up_comp_ren_subst. unfold subst in ih. apply ih.
229     + rewrite map_map. induction ms.
230       { reflexivity. }
231       { simpl. rewrite ih. rewrite IHms. reflexivity. }
232 Qed.
233
234 Lemma up_comp_subst_ren_term2 :
235   forall sigma xi, up (sigma >>> rename xi) = up sigma >>> rename (upren xi).
236 Proof.
237   apply up_comp_subst_ren_internal.
238   - reflexivity.
239   - apply rename_subst_term2.
240   - apply ren_subst_comp_term2.
241 Qed.
242
243 Lemma up_comp_subst_ren_n_term2 :
244   forall sigma xi n, upn n (sigma >>> rename xi) = upn n sigma >>> rename
    ↪ (iterate upren n xi).
245 Proof.
246   apply up_comp_subst_ren_n_internal. apply up_comp_subst_ren_term2.
247 Qed.
248
249 Lemma subst_ren_comp_term2 : forall sigma xi (s : term2),
    rename xi s.[sigma] = s.[sigma >>> rename xi].
250 Proof.
251 fix ih 3. intros. destruct s.
252   - simpl. destruct (sigma x).
253     + simpl. apply term2_var_app_apps_eq. rewrite map_app. apply
    ↪ list_split_eq_l.
254       unfold subst in ih. rewrite map_map. induction ms.
255         { reflexivity. }
256         { simpl. rewrite ih. rewrite IHms. reflexivity. }
257     + simpl. apply term2_lam_app_apps_eq. rewrite map_app. apply
    ↪ list_split_eq_l.
258       unfold subst in ih. rewrite map_map. induction ms.
259

```

```

260     { reflexivity. }
261     { simpl. rewrite ih. rewrite IHms. reflexivity. }
262 - simpl. apply term2_lam_app_split_eq.
263 + unfold rename in ih. unfold rename. unfold subst in ih. rewrite
    ↪ up_comp_subst_ren_term2.
264     apply ih.
265 + rewrite map_map. induction ms.
266     { reflexivity. }
267     { simpl. rewrite ih. rewrite IHms. reflexivity. }
268 Qed.
269
270 Lemma subst_comp_term2 : forall (sigma tau : var → term2) (s : term2),
271     s.[sigma].[tau] = s.[sigma >> tau].
272 Proof.
273 intros.
274 revert sigma tau.
275 induction s using term2_ind'.
276 - intros. simpl. destruct (sigma x); simpl.
277   + destruct (tau x0);
278     try apply term2_var_app_apps_eq;
279     try apply term2_lam_app_apps_eq;
280     try rewrite ← app_assoc;
281     apply list_split_eq_l;
282     rewrite map_app;
283     apply list_split_eq_l;
284     rewrite Forall_forall in H;
285     rewrite map_map;
286     apply map_ext_in;
287     intros; apply H; assumption.
288 + apply term2_lam_app_apps_eq.
289     rewrite map_app.
290     apply list_split_eq_l.
291     rewrite Forall_forall in H.
292     rewrite map_map.
293     apply map_ext_in.
294     intros. apply H. assumption.
295 - intros. simpl. unfold subst in IHs.
296     apply term2_lam_app_split_eq.
297 + rewrite IHs. rewrite up_comp_internal.
298     { reflexivity. }
299     { apply id_subst_term2. }
300     { apply ren_subst_comp_term2. }
301     { apply subst_ren_comp_term2. }
302 + rewrite map_map. rewrite Forall_forall in H. erewrite map_ext_in.
303     { reflexivity. }
304     { intros. apply H. assumption. }
305 Qed.
306
307 Instance SubstLemmas_term2 : SubstLemmas term2.
308 Proof.
309     split.
310     { apply rename_subst_term2. }
311     { apply subst_id_term2. }
312     { apply id_subst_term2. }
313     { apply subst_comp_term2. }
314 Qed.

```

```
315
316      (*)
317  Coercion term_term2 : term → term2.
318  Coercion term2_term : term2 → term.
319  *)
```