

Paths.v

```
1 Require Import Coq.Arith.PeanoNat.
2 Require Import Coq.Lists.List.
3 Require Import Coq.Bool.Sumbool.
4 Require Import Coq.Classes.EquivDec.
5 Require Import Autosubst.Autosubst.
6 Require Import Omega.
7
8 Require Import PrincInh.Types.
9 Require Import PrincInh.Utills.
10
11 Import ListNotations.
12 Import EqNotations.
13
14 Inductive dir :Type :=
15 | Src
16 | Tgt
17 .
18
19 Instance dir_eqdec : EqDec dir eq.
20 Proof.
21   unfold EqDec.
22   intros.
23   destruct x; destruct y;
24   try (left; reflexivity);
25   try (right; intros F; discriminate F).
26 Defined.
27
28 Definition path :Type := list dir.
29
30 Hint Rewrite (@nth_error_nil path) app_nil_l app_nil_r.
31
32 Fixpoint P (rho:type) (pi: path) {struct pi} : option type :=
33   match pi with
34   | [] ⇒ Some rho
35   | Src::pi' ⇒ match rho with
36     | (? x) ⇒ None
37     | sigma ~> _ ⇒ P sigma pi'
38   end
39   | Tgt::pi' ⇒ match rho with
40     | (? x) ⇒ None
41     | _ ~> tau ⇒ P tau pi'
42   end
43 end.
44
45 Fixpoint P_default (rho:type) (def : type) (pi: path) {struct pi} : type :=
46   match pi with
47   | [] ⇒ rho
48   | Src::pi' ⇒ match rho with
49     | (? x) ⇒ def
50     | sigma ~> _ ⇒ P_default sigma def pi'
51   end
52   | Tgt::pi' ⇒ match rho with
53     | (? x) ⇒ def
54     | _ ~> tau ⇒ P_default tau def pi'
```

```

55         end
56     end.
57
58 Fixpoint dom_P (rho: type) : list path :=
59     match rho with
60     | ? x ⇒ [[]]
61     | sigma ⇒ tau ⇒ [] :: map (cons Src) (dom_P sigma) ++ map (cons Tgt) (dom_P
        ↪ tau)
62     end.
63
64 Lemma dom_P_some : forall pi rho, In pi (dom_P rho) →
65     { tau & P rho pi = Some tau }.
66 Proof.
67     induction pi.
68     - intros. exists rho. destruct rho; reflexivity.
69     - intros. destruct a.
70       + destruct rho.
71         * simpl in H. exfalso. ainv.
72         * simpl in H. simpl. apply IHpi. destruct H as [F | H]. {inversion F. }
73           ↪ apply in_app_or in H as [H|H];
74             apply in_map_iff in H as [pi' [H1 H2]]; ainv.
75       + destruct rho.
76         * simpl in H. exfalso. ainv.
77         * simpl in H. simpl. apply IHpi. destruct H as [F | H]. {inversion F. }
78           ↪ apply in_app_or in H as [H|H];
79             apply in_map_iff in H as [pi' [H1 H2]]; ainv.
80 Qed.
81
82 Lemma dom_P_none : forall pi rho, ~ In pi (dom_P rho) → P rho pi = None.
83 Proof.
84     induction pi.
85     - intros. exfalso. apply H. destruct rho; simpl; left; reflexivity.
86     - destruct a.
87       + intros. simpl. destruct rho.
88         * reflexivity.
89         * apply IHpi. simpl in H. intros H1. apply H. right. apply in_or_app.
90           ↪ left.
91           apply in_map. assumption.
92       + intros. simpl. destruct rho.
93         * reflexivity.
94         * apply IHpi. simpl in H. intros H1. apply H. right. apply in_or_app.
95           ↪ right.
96           apply in_map. assumption.
97 Qed.
98
99 Lemma dom_P_false : forall pi' d x, In (d :: pi') (dom_P (? x)) → False.
100 Proof.
101     ainv.
102 Qed.
103
104 Lemma dom_P_Src {pi sigma tau} : In (Src :: pi) (dom_P (sigma ⇒ tau)) → In pi
    ↪ (dom_P sigma).
105     intros. asimpl in H. destruct H. discriminate H. apply In_app_sumbool in H.
    ↪ destruct H.
106     + apply in_map_cons in i. assumption.
107     + exfalso. apply in_map_cons_not in i. apply i. intros F. discriminate F.

```

```

104 Qed.
105
106 Lemma dom_P_Src_iff {pi sigma tau} : In (Src :: pi) (dom_P (sigma  $\rightsquigarrow$  tau))  $\rightsquigarrow$ 
     $\hookrightarrow$  In pi (dom_P sigma).
107 Proof.
108   split; intros.
109   - apply dom_P_Src in H. assumption.
110   - asimpl. right. apply in_or_app. left. apply in_map_cons_iff. assumption.
111 Qed.
112
113 Lemma dom_P_Tgt {pi sigma tau} : In (Tgt :: pi) (dom_P (sigma  $\rightsquigarrow$  tau))  $\rightarrow$  In pi
     $\hookrightarrow$  (dom_P tau).
114   intros. asimpl in H. destruct H. discriminate H. apply In_app_sumbool in H.
115    $\hookrightarrow$  destruct H.
116   + ex falso. apply in_map_cons_not in i. apply i. intros F. discriminate F.
117   + apply in_map_cons in i. assumption.
118 Qed.
119
120 Lemma dom_P_Tgt_iff {pi sigma tau} : In (Tgt :: pi) (dom_P (sigma  $\rightsquigarrow$  tau))  $\rightsquigarrow$ 
     $\hookrightarrow$  In pi (dom_P tau).
121 Proof.
122   split; intros.
123   - apply dom_P_Tgt in H. assumption.
124   - asimpl. right. apply in_or_app. right. apply in_map_cons_iff. assumption.
125 Qed.
126
127 Lemma dom_P_last : forall rho pi d, In (pi  $\leftrightarrow$  [d]) (dom_P rho)  $\rightarrow$  In pi (dom_P
     $\hookrightarrow$  rho).
128 Proof.
129   induction rho; intros.
130   - pose proof (app_cons_not_nil pi [] d). inversion H; try contradiction.
131   - asimpl. destruct pi.
132     + left. reflexivity.
133     + right. apply in_or_app. destruct d0.
134       * left. apply in_map_cons_iff. eapply IHrho1. eapply dom_P_Src. exact H.
135       * right. apply in_map_cons_iff. eapply IHrho2. eapply dom_P_Tgt. exact H.
136 Qed.
137
138 Lemma dom_P_prefix : forall pi' pi rho, In (pi  $\leftrightarrow$  pi') (dom_P rho)  $\rightarrow$  In pi
     $\hookrightarrow$  (dom_P rho).
139 Proof.
140   induction pi' using rev_ind.
141   - intros. rewrite app_nil_r in H. assumption.
142   - intros. rewrite app_assoc in H. apply dom_P_last in H. apply IHpi'.  $\hookrightarrow$  assumption.
143 Qed.
144
145 Lemma P_prefix {rho pi pi' tau}: P rho (pi  $\leftrightarrow$  pi') = Some tau  $\rightarrow$  {tau' & P rho
     $\hookrightarrow$  pi = Some tau'}.
146 Proof.
147   intros.
148   revert rho tau pi' H.
149   induction pi.
150   - intros. exists rho. reflexivity.
151   - intros. simpl. destruct a; destruct rho; try discriminate H;
    simpl in H; eapply IHpi; exact H.

```

```

152 Qed.
153
154 Lemma dom_P_nil : forall rho, In [] (dom_P rho).
155 Proof.
156   destruct rho; simpl; left; reflexivity.
157 Qed.
158
159 Definition P_ok rho pi (proof : In pi (dom_P rho)) : type.
160   revert rho pi proof.
161   fix dummy 2. intros.
162   destruct pi.
163   - exact rho.
164   - destruct rho.
165     + exfalso. exact (dom_P_false _ _ _ proof).
166     + destruct d.
167       * exact (dummy rho1 pi (dom_P_Src proof)).
168       * exact (dummy rho2 pi (dom_P_Tgt proof)).
169 Defined.
170
171 Lemma P_ok_Src : forall sigma tau pi pr, P_ok (sigma  $\rightarrow$  tau) (Src::pi) pr = P_ok
   $\hookrightarrow$  (sigma) pi (dom_P_Src pr).
172 Proof.
173   reflexivity.
174 Qed.
175
176 Lemma P_ok_Tgt : forall sigma tau pi pr, P_ok (sigma  $\rightarrow$  tau) (Tgt::pi) pr = P_ok
   $\hookrightarrow$  tau pi (dom_P_Tgt pr).
177 Proof.
178   reflexivity.
179 Qed.
180
181 Lemma P_ok_proof_irl : forall rho pi p1 p2, P_ok rho pi p1 = P_ok rho pi p2.
182 Proof.
183   induction rho.
184   - intros. destruct pi.
185     + reflexivity.
186     + inversion p1. discriminate H. inversion H.
187   - intros. destruct pi.
188     + reflexivity.
189     + destruct d.
190       * rewrite P_ok_Src. rewrite P_ok_Src. apply IHrho1.
191       * rewrite P_ok_Tgt. rewrite P_ok_Tgt. apply IHrho2.
192 Qed.
193
194 Lemma P_ok_P {rho pi tau pr}: P_ok rho pi pr = tau  $\rightarrow$  P rho pi = Some tau.
195 Proof.
196   split.
197   - revert rho tau pr. induction pi.
198     + simpl. intros rho tau _. apply some_eq.
199     + simpl. intros. destruct rho.
200       * inversion pr. discriminate H0. inversion H0.
201       * destruct a; eapply IHpi; exact H.
202   - revert rho tau pr. induction pi.
203     + simpl. intros rho tau _ eq. apply some_eq. exact eq.
204     + simpl. intros. destruct rho.
205       * destruct a; discriminate H.

```

```

206 * destruct a.
207 ** eapply IHpi in H. apply H.
208 ** eapply IHpi in H. apply H.

```

209 Qed.

210
211 **Lemma** P_ok_P_ex {rho pi tau}: (exists pr, P_ok rho pi pr = tau) \rightarrow P rho pi =
 \hookrightarrow Some tau.

212 **Proof.**

213 **split.**

```

214 - revert rho tau. induction pi.
215 + simpl. intros. destruct H. subst. reflexivity.
216 + simpl. intros. destruct rho.
217   * destruct H as [pr H]. inversion pr. discriminate H0. inversion H0.
218   * destruct a.
219     ** eapply IHpi. destruct H as [pr H]. exists (dom_P_Src pr). exact H.
220     ** eapply IHpi. destruct H as [pr H]. exists (dom_P_Tgt pr). exact H.
221 - revert rho tau. induction pi.
222 + simpl. intros rho tau eq. exists (dom_P_nil rho). apply some_eq. exact eq.
223 + simpl. intros. destruct rho.
224   * destruct a; discriminate H.
225   * destruct a.
226     ** apply IHpi in H.
227       destruct H as [pr H]. assert (In (Src::pi) (dom_P (rho1  $\rightarrow$  rho2))).
228       {
229         apply dom_P_Src_iff. assumption.
230       }
231       exists H0. rewrite (P_ok_proof_irl _ _ _ pr). assumption.
232     ** apply IHpi in H.
233       destruct H as [pr H]. assert (In (Tgt::pi) (dom_P (rho1  $\rightarrow$  rho2))).
234       {
235         apply dom_P_Tgt_iff. assumption.
236       }
237       exists H0. rewrite (P_ok_proof_irl _ _ _ pr). assumption.

```

238 Qed.

239
240 **Lemma** P_P_ok_set {rho pi tau}: P rho pi = Some tau \rightarrow { pr & P_ok rho pi pr =
 \hookrightarrow tau }.

241 **Proof.**

```

242 - revert rho tau. induction pi.
243 + simpl. intros rho tau eq. exists (dom_P_nil rho). apply some_eq. exact eq.
244 + simpl. intros. destruct rho.
245   * destruct a; discriminate H.
246   * destruct a.
247     ** apply IHpi in H.
248       destruct H as [pr H]. assert (In (Src::pi) (dom_P (rho1  $\rightarrow$  rho2))).
249       {
250         apply dom_P_Src_iff. assumption.
251       }
252       exists H0. rewrite (P_ok_proof_irl _ _ _ pr). assumption.
253     ** apply IHpi in H.
254       destruct H as [pr H]. assert (In (Tgt::pi) (dom_P (rho1  $\rightarrow$  rho2))).
255       {
256         apply dom_P_Tgt_iff. assumption.
257       }
258       exists H0. rewrite (P_ok_proof_irl _ _ _ pr). assumption.

```

259 Qed.

```

260
261 Definition make_tgt_path (pi: path) (n : nat) :=
262   pi ++ (repeat Tgt n) ++ [Src].
263
264 Definition even_ones pi := Nat.Even (count_occ dir_eqdec pi Src).
265
266 Lemma even_ones_pump pi : even_ones pi = even_ones (pi ++ [Tgt]).
267 Proof.
268   unfold even_ones.
269   rewrite count_occ_last_neq.
270   - reflexivity.
271   - isfalse.
272 Qed.
273
274 Definition odd_repo (Delta : list path) := Forall (fun pi ⇒ Nat.Odd (count_occ
  ⇨ dir_eqdec pi Src)) Delta.
275
276 Lemma odd_repo_head (Delta : list path) : forall pi, odd_repo (pi :: Delta) →
  ⇨ Nat.Odd (count_occ dir_eqdec pi Src).
277 Proof.
278   intros.
279   unfold odd_repo in H.
280   inv H.
281   assumption.
282 Qed.
283
284 Lemma odd_repo_split Delta : forall pi,
285   ~ odd_repo (pi :: Delta) →
286   odd_repo Delta →
287   ~ Nat.Odd (count_occ dir_eqdec pi Src).
288 Proof.
289   intros.
290   unfold odd_repo in *.
291   intros F.
292   apply H.
293   constructor; assumption.
294 Qed.
295
296 Lemma odd_repo_comb Delta : forall pi,
297   odd_repo Delta →
298   Nat.Odd (count_occ dir_eqdec pi Src) →
299   odd_repo (pi :: Delta).
300 Proof.
301   intros. unfold odd_repo. constructor; assumption.
302 Qed.
303
304 Lemma odd_repo_head_eq (Delta : list path) : forall pi pi', (odd_repo (pi ::
  ⇨ Delta) → odd_repo (pi' :: Delta)) →
305 odd_repo Delta → (Nat.Odd (count_occ dir_eqdec pi Src) → Nat.Odd (count_occ
  ⇨ dir_eqdec pi' Src)).
306 Proof.
307   intros.
308   eapply odd_repo_head.
309   apply H.
310   apply odd_repo_comb; assumption.
311 Qed.

```

```

312
313 Lemma odd_repo_head_eq2 Delta : forall pi pi',
314   (Nat.Odd (count_occ dir_eqdec pi Src) → Nat.Odd (count_occ dir_eqdec pi'
    ↪ Src)) →
315
    odd_repo (pi :: Delta) → odd_repo
    ↪ (pi' :: Delta).

316 Proof.
317   intros.
318   unfold odd_repo in H0.
319   unfold odd_repo .
320   constructor.
321   - apply H. ainv.
322   - inversion H0. assumption.
323 Qed.
324
325 Lemma odd_repo_head_tail (Delta : list path) : forall pi, odd_repo ((pi ++
    ↪ [Src]) :: Delta) ↗ odd_repo ((Src :: pi) :: Delta).
326 Proof.
327   intros.
328   split.
329   - apply odd_repo_head_eq2. simpl.
330   rewrite count_occ_split. simpl. rewrite Nat.add_comm. simpl. auto.
331   - apply odd_repo_head_eq2. simpl.
332   rewrite count_occ_split. simpl. rewrite Nat.add_comm. simpl. auto.
333 Qed.
334
335 Lemma tgt_path_even_if_pi_odd : forall n pi, Nat.Odd (count_occ dir_eqdec pi
    ↪ Src) →
336   Nat.Even (count_occ dir_eqdec (make_tgt_path pi n) Src).
337 Proof.
338   unfold make_tgt_path. simpl. intros n pi.
339   repeat rewrite count_occ_split. simpl. rewrite Nat.add_comm. revert n pi.
340   induction n.
341   - intros. simpl.
342   apply Nat.Even_succ. assumption.
343   - intros. simpl. apply IHn. assumption.
344 Qed.
345
346 Lemma tgt_path_even_if_delta_odd: forall (Delta : list path) pi n,
347   odd_repo Delta → In pi Delta →
348   even_ones (make_tgt_path pi n).
349 Proof.
350   intros.
351   apply tgt_path_even_if_pi_odd.
352   unfold odd_repo in H.
353   rewrite Forall_forall in H. apply H. assumption.
354 Qed.
355
356 Lemma P_src {rho pi sigma tau} : P rho pi = Some (sigma ↗ tau) → P rho (pi ++
    ↪ [Src]) = Some sigma.
357 Proof.
358   revert pi rho sigma tau. induction pi; intros.
359   - simpl in H. apply some_eq in H. subst. reflexivity.
360   - destruct a; simpl in H; destruct rho; try discriminate H; simpl; eapply
    ↪ IHpi; apply H.
361 Qed.

```

```

362
363
364 Lemma P_tgt {rho pi sigma tau} : P rho pi = Some (sigma  $\rightarrow$  tau)  $\rightarrow$  P rho (pi ++
 $\hookrightarrow$  [Tgt]) = Some tau.
365 Proof.
366   revert pi rho sigma tau. induction pi; intros.
367   - simpl in H. apply some_eq in H. subst. reflexivity.
368   - destruct a; simpl in H; destruct rho; try discriminate H; simpl; eapply
 $\hookrightarrow$  IHpi; apply H.
369 Qed.
370
371 Lemma P_app_split {pi pi' rho rho' rho''}: P rho pi = Some rho'  $\rightarrow$  P rho' pi' =
 $\hookrightarrow$  Some rho''  $\rightarrow$  P rho (pi ++ pi') = Some rho''.
372 Proof.
373   revert pi' rho rho' rho''.
374   induction pi.
375   - ainv.
376   - intros. asimpl. destruct a.
377     + asimpl in *. destruct rho; try discriminate H. eapply IHpi. apply H. apply
 $\hookrightarrow$  H0.
378     + asimpl in *. destruct rho; try discriminate H. eapply IHpi. apply H. apply
 $\hookrightarrow$  H0.
379 Qed.
380
381 Lemma P_app_proof {a ts rho pi} :
382   P rho pi = Some (make_arrow_type ts a)  $\rightarrow$ 
383   forall n (pr: n < length ts), P rho (pi ++ (repeat Tgt n ++ [Src])) = Some
 $\hookrightarrow$  (nth_ok ts n pr).
384 Proof.
385   intros.
386   eapply P_app_split. apply H.
387   clear H rho pi.
388   revert ts n pr a.
389   induction ts.
390   - ainv.
391   - intros. destruct n.
392     + reflexivity.
393     + simpl. simpl in pr. pose proof (lt_S_n _ _ pr).
394       erewrite nth_ok_proof_irel.
395       apply (IHts n H).
396 Qed.
397
398 Lemma P_app_proof_in {rho pi a ts} : P rho pi = Some (make_arrow_type ts a)  $\rightarrow$ 
399   forall n (pr: n < length ts),
400   In (pi ++ repeat Tgt n ++ [Src])
 $\hookrightarrow$  (dom_P rho).
401 Proof.
402   intros. apply (@P_app_proof a ts rho pi) with (n:=n) (pr:=pr) in H.
403   apply P_ok_P_ex in H. destruct H as [pr0 _]. assumption.
404 Qed.
405
406 Lemma dom_P_head_Src : forall pi rho, In (Src :: pi) (dom_P rho)  $\rightarrow$  {rho1 &
 $\hookrightarrow$  {rho2 & rho = rho1  $\rightarrow$  rho2 /\ In pi (dom_P rho1)}}.
407 Proof.
408   intros pi  $\llbracket$  t1 t2  $\rrbracket$  H.
409   - exfalso. ainv.

```



```

410 - exists t1. exists t2. apply dom_P_Src in H. split. reflexivity. assumption.
411 Qed.
412
413 Lemma dom_P_head_Tgt : forall pi rho, In (Tgt :: pi) (dom_P rho) → {rho1 &
  ↪ {rho2 & rho = rho1 ↪ rho2 /\ In pi (dom_P rho2)}}.
414 Proof.
415   intros pi [[t1 t2] H].
416   - exfalso. ainv.
417   - exists t1. exists t2. apply dom_P_Tgt in H. split. reflexivity. assumption.
418 Qed.
419
420 Lemma dom_P_Src_to_Tgt : forall pi rho dir1 dir2, In (pi ++ [dir1]) (dom_P rho)
  ↪ → In (pi ++ [dir2]) (dom_P rho).
421 Proof.
422   induction pi.
423   - intros. destruct rho. ainv. destruct dir2; (asimpl; right; apply in_or_app).
424     + left. apply in_map_cons_iff. apply dom_P_nil.
425     + right. apply in_map_cons_iff. apply dom_P_nil.
426   - simpl. intros. destruct a.
427     + apply dom_P_head_Src in H. destruct H as [t1 [t2 [Hrho HIn]]]. subst.
428       simpl. right. apply in_or_app. left. apply in_map_cons_iff. eapply IHpi.
429       ↪ apply HIn.
430     + apply dom_P_head_Tgt in H. destruct H as [t1 [t2 [Hrho HIn]]]. subst.
431       simpl. right. apply in_or_app. right. apply in_map_cons_iff. eapply IHpi.
432       ↪ apply HIn.
433 Qed.
434
435 Lemma P_ok_Src_to_Tgt : forall pi rho dir1 dir2 pr1 sigma, P_ok rho (pi ++
  ↪ [dir1]) pr1 = sigma →
436
437                                     {pr2 & {tau & P_ok rho (pi ++
438                                     ↪ [dir2]) pr2 = tau}}.
439 Proof.
440   intros.
441   pose proof dom_P_Src_to_Tgt _ _ _ dir2 pr1.
442   exists H0. pose proof dom_P_some _ _ H0 as [tau HP]. exists tau.
443   apply P_ok_P. assumption.
444 Qed.
445
446 Lemma P_ok_make_arrow : forall ts a, {pr & P_ok (make_arrow_type ts a) (repeat
  ↪ Tgt (length ts)) pr = a}.
447 Proof.
448   intros.
449   induction ts.
450   - intros. simpl. exists (dom_P_nil _). reflexivity.
451   - simpl. destruct IHts as [pr IHts].
452     eexists. erewrite P_ok_proof_irl. exact IHts.
453     Unshelve.
454   right.
455   apply in_or_app. right. apply in_map_cons_iff. assumption.
456 Qed.
457
458 Lemma P_Src2 : forall pi rho sigma, P rho (pi ++ [Src]) = Some sigma → {tau & P
  ↪ rho pi = Some (sigma ↪ tau) /\
459
460                                     P rho (pi ++
461                                     ↪ [Tgt]) =
462                                     ↪ Some tau}.

```

```

456 Proof.
457   induction pi.
458   - intros. rewrite app_nil_l in H. inversion H. destruct rho eqn:Hrho; try
    ↪ discriminate H1.
459     simpl. apply some_eq in H1. subst. exists t2. split; reflexivity.
460   - intros. destruct a.
461     + simpl. destruct rho eqn:Hrho; try discriminate H. apply IHpi.
462       simpl in H. assumption.
463     + simpl. destruct rho eqn:Hrho; try discriminate H. apply IHpi.
464       simpl in H. assumption.
465 Qed.
466
467 Lemma P_Tgt2 : forall pi rho tau, P rho (pi ++ [Tgt]) = Some tau → {sigma & P
    ↪ rho pi = Some (sigma → tau) /\
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505

```

P rho (pi ++ [Src])
↪ = Some sigma}.

```

Proof.
  induction pi.
  - intros. rewrite app_nil_l in H. inversion H. destruct rho eqn:Hrho; try
    ↪ discriminate H1.
    simpl. apply some_eq in H1. subst. exists t1. split; reflexivity.
  - intros. destruct a.
    + simpl. destruct rho eqn:Hrho; try discriminate H. apply IHpi.
      simpl in H. assumption.
    + simpl. destruct rho eqn:Hrho; try discriminate H. apply IHpi.
      simpl in H. assumption.
Qed.

Lemma P_path_make_arrow_type {tau pi n rho}: P tau (pi ++ repeat Tgt n) = Some
  ↪ rho →
    {ts & P tau pi = Some (make_arrow_type ts
    ↪ rho) /\ length ts = n}.

Proof.
  revert pi rho tau.
  induction n; intros pi rho tau.
  - simpl. rewrite app_nil_r. intros. exists []. auto.
  - rewrite repeat_rev. intros. rewrite app_assoc in H. apply P_Tgt2 in H.
    destruct H as [sigma [HP1 HP2]].
    pose proof IHn _ _ HP1 as [ts [Hres HLen]].
    exists (ts ++ [sigma]). simpl. rewrite make_arrow_type_last.
    split. assumption. rewrite app_length. simpl. rewrite HLen. omega.
Qed.

Lemma make_arrow_type_dirs {tau ts a n}:
  make_arrow_type ts (? a) = tau →
  P tau (repeat Tgt n ++ [Src]) = nth_error ts n.

Proof.
  revert ts tau.
  induction n.
  - intros. simpl. destruct tau.
    + pose proof make_arrow_type_ts_is_nil H as [Hts Hrho].
      subst. reflexivity.
    + destruct ts.
      * discriminate H.
      * simpl in H. injection H. intros. subst. reflexivity.
  - intros. asimpl. destruct tau.

```

```

506 + pose proof make_arrow_type_ts_is_nil H as [Hts Hrho].
507 subst. reflexivity.
508 + destruct ts.
509 * discriminate H.
510 * apply IHn. injection H. intros. assumption.
511 Qed.
512
513 Fixpoint replace_at_path b tau pi {struct pi} : type :=
514   match pi with
515   | [] => b
516   | Src :: pi' => match tau with
517                     | (? _) => tau
518                     | sigma ~> tau' => replace_at_path b sigma pi' ~> tau'
519                     end
520   | Tgt :: pi' => match tau with
521                     | (? _) => tau
522                     | sigma ~> tau' => sigma ~> replace_at_path b tau' pi'
523                     end
524   end.

```