## Printed by Zach Tatlock

```
StructTactics.v
                                                                                                                             StructTactics.v
 Oct 24, 16 7:17
                                                                           Page 1/7
                                                                                          Oct 24, 16 7:17
                                                                                                                                                                    Page 2/7
(** https://github.com/uwplse/StructTact *)
                                                                                                    | [H : exists (name : _), _ |- _ ] =>
                                                                                                      let x := fresh name in
                                                                                                      destruct H as [x]
Ltac subst max :=
 repeat match goal with
                                                                                                  end.
          | [ H : ?X = _ |- _ ] => subst X
| [H : _ = ?X |- _] => subst X
                                                                                        Ltac break_exists_exists :=
         end.
                                                                                          repeat match goal with
                                                                                                    | H:exists _, _ |- _ =>
                                                                                                      let x := fresh "x" in
Ltac inv H := inversion H; subst_max.
Ltac invc H := inv H; clear H.
                                                                                                      destruct H as [x]; exists x
Ltac invcs H := invc H; simpl in *.
                                                                                                  end.
Ltac break if :=
                                                                                        Ltac break and :=
 match goal with
                                                                                          repeat match goal with
   | [ |- context [ if ?X then _ else _ ] ] =>
                                                                                                  | [H : _ /\ _ |- _ ] => destruct H
      match type of X with
                                                                                                  end.
        | sumbool _ _ => destruct X
        | _ => destruct X eqn:?
                                                                                         Ltac break_and_goal :=
      end
                                                                                            repeat match goal with
                                                                                                    | [ |- _ /\ _ ] => split
    | [ H : context [ if ?X then _ else _ ] |- _] =>
      match type of X with
                                                                                                    end
        | sumbool _ _ => destruct X
        | _ => destruct X eqn:?
                                                                                        Ltac solve_by_inversion' tac :=
                                                                                          match goal with
      end
 end.
                                                                                           | [H : _ |- _] => solve [inv H; tac]
                                                                                           end.
Ltac break_match_hyp :=
 match goal with
                                                                                         Ltac solve by inversion := solve by inversion' auto.
   | [ H : context [ match ?X with _ => _ end ] |- _] =>
      match type of X with
                                                                                         Ltac apply_fun f H:=
       | sumbool _ _ => destruct X
                                                                                          match type of H with
        | _ => destruct X eqn:?
                                                                                            | ?X = ?Y => assert (f X = f Y)
      end
                                                                                           end.
 end.
                                                                                        Ltac conclude H tac :=
                                                                                           (let H' := fresh in
Ltac break_match_goal :=
 match goal with
                                                                                           match type of H with
   [ [ - context [ match ?X with => end ] ] =>
                                                                                            | ?P -> => assert P as H' by (tac)
      match type of X with
                                                                                            end; specialize (H H'); clear H').
       | sumbool _ _ => destruct X
        | _ => destruct X eqn:?
                                                                                         Ltac concludes :=
      end
                                                                                          match goal with
 end.
                                                                                            | [ H : ?P -> _ |- _ ] => conclude H auto
                                                                                           end.
Ltac break_match := break_match_goal || break_match_hyp.
                                                                                         Ltac forward H :=
Ltac break_inner_match' t :=
                                                                                          let H' := fresh in
match t with
                                                                                           match type of H with
  context[match ?X with _ => _ end] =>
                                                                                            | ?P -> _ => assert P as H'
    break_inner_match' X || destruct X eqn:?
                                                                                            end.
  | _ => destruct t eqn:?
end.
                                                                                         Ltac forwards :=
                                                                                          match goal with
                                                                                           | [ H : ?P -> _ |- _ ] => forward H
Ltac break_inner_match_goal :=
match goal with
                                                                                           end.
 [ [ - context[match ?X with _ => _ end] ] =>
                                                                                         Ltac find_contradiction :=
    break_inner_match' X
end.
                                                                                          match goal with
                                                                                           | [ H : ?X = _, H' : ?X = _ |- _ ] => rewrite H in H'; solve_by_inversion
Ltac break_inner_match_hyp :=
                                                                                           end.
match goal with
 [ H : context[match ?X with _ => _ end] |- _ ] =>
                                                                                         Ltac find_rewrite :=
    break_inner_match' X
                                                                                          match goal with
                                                                                            | [ H : ?X _ _ _ = _, H' : ?X _ _ _ = _ |- _ ] => rewrite H in H'
| [ H : ?X = _, H' : ?X = _ |- _ ] => rewrite H in H'
| [ H : ?X = _, H' : context [ ?X ] |- _ ] => rewrite H in H'
| [ H : ?X = _ |- context [ ?X ] ] => rewrite H
end.
Ltac break_inner_match := break_inner_match_goal || break_inner_match_hyp.
Ltac break_exists :=
                                                                                           end.
 repeat match goal with
```

Monday October 24, 2016

## Printed by Zach Tatlock

Page 4/7

```
StructTactics.v
                                                                                                                                      StructTactics.v
                                                                                                 Oct 24, 16 7:17
 Oct 24, 16 7:17
                                                                                Page 3/7
Ltac find erewrite :=
                                                                                                  end.
 match goal with
    | [ H : ?X _ _ _ = _, H' : ?X _ _ _ = _ |- _ ] => erewrite H in H'
| [ H : ?X = _, H' : ?X = _ |- _ ] => erewrite H in H'
| [ H : ?X = _, H' : context [ ?X ] |- _ ] => erewrite H in H'
                                                                                               Ltac break_let :=
                                                                                                 match goal with
                                                                                                   | [ H : context [ (let (_,_) := ?X in _) ] |- _ ] => destruct X eqn:?
    | [ H : ?X = _ |- context [ ?X ] ] => erewrite H
                                                                                                   | [ |- context [ (let (_,_) := ?X in _) ] ] => destruct X eqn:?
 end.
                                                                                                 end.
Ltac find_rewrite_lem lem :=
                                                                                               Ltac break_or_hyp :=
 match goal with
                                                                                                 match goal with
    | [ H : _ |- _ ] =>
                                                                                                  | [ H : _ \/ _ |- _ ] => invc H
      rewrite lem in H; [idtac]
                                                                                                 end.
 end.
                                                                                               Ltac copy_apply lem H :=
Ltac find_rewrite_lem_by lem t :=
                                                                                                let x := fresh in
                                                                                                pose proof H as x;
 match goal with
   | [ H : _ |- _ ] =>
                                                                                                   apply lem in x.
     rewrite lem in H by t
 end.
                                                                                               Ltac copy_eapply lem H :=
                                                                                                let x := fresh in
Ltac find_erewrite_lem lem :=
                                                                                                 pose proof H as x;
 match goal with
                                                                                                   eapply lem in x.
  | [ H : _ |- _] => erewrite lem in H by eauto
                                                                                               Ltac conclude_using tac :=
 end.
                                                                                                 match goal with
Ltac find_reverse_rewrite :=
                                                                                                  | [ H : ?P -> _ |- _ ] => conclude H tac
 match goal with
                                                                                                 end.
   | [ H : _ = ?X _ _ _, H' : ?X _ _ _ = _ |- _ ] => rewrite <- H in H'
| [ H : _ = ?X, H' : context [ ?X ] |- _ ] => rewrite <- H in H'
                                                                                               Ltac find_higher_order_rewrite :=
    | [ H : _ = ?X |- context [ ?X ] ] => rewrite <- H
                                                                                                 match goal with
 end.
                                                                                                   | [ H : _ = _ |- _ ] => rewrite H in *
                                                                                                   | [ H : forall _, _ = _ |- _ ] => rewrite H in *
Ltac find_inversion :=
                                                                                                   | [ H : forall ____ = _ |- _ ] => rewrite H in *
 match goal with
                                                                                                 end
        H : ?X _ _ _ = ?X _ _ _ |- _ ] => invc H
    | [ H : ?X _ _ _ = ?X _ _ _ |- ] => invc H
| [ H : ?X _ _ = ?X _ _ _ |- ] => invc H
                                                                                               Ltac find_reverse_higher_order_rewrite :=
                                                                                                 match goal with
   | [ H : 2X _ _ = 2X _ _ |-_] => invc H
| [ H : 2X _ _ = 2X _ _ |-_] => invc H
| [ H : 2X _ = 2X _ |-_] => invc H
                                                                                                   | [ H : _ = _ |- _ ] => rewrite <- H in *
                                                                                                   | [ H : forall _, _ = _ |- _ ] => rewrite <- H in *
| [ H : forall _, _ = _ |- _ ] => rewrite <- H in *
 end.
                                                                                                 end.
Ltac prove eg :=
                                                                                               Ltac clean :=
 match goal with
                                                                                                match goal with
   | [ H : ?X ?x1 ?x2 ?x3 = ?X ?y1 ?y2 ?y3 |- _ ] =>
                                                                                                  | [ H : ?X = ?X |- _ ] => clear H
      assert (x1 = y1) by congruence;
                                                                                                 end.
        assert (x^2 = y^2) by congruence;
        assert (x3 = y3) by congruence;
                                                                                               Ltac find_apply_hyp_goal :=
        clear H
                                                                                                match goal with
    | [ H : ?X ?x1 ?x2 = ?X ?y1 ?y2 |- _ ] =>
                                                                                                  | [ H : _ |- _ ] => solve [apply H]
      assert (x1 = y1) by congruence;
                                                                                                 end.
        assert (x^2 = y^2) by congruence;
        clear H
                                                                                               Ltac find_copy_apply_lem_hyp lem :=
    | [ H : ?X ?x1 = ?X ?y1 |- _ ] =>
                                                                                                 match goal with
      assert (x1 = y1) by congruence;
                                                                                                  | [ H : _ |- _ ] => copy_apply lem H
        clear H
                                                                                                 end.
 end.
                                                                                               Ltac find_apply_hyp_hyp :=
Ltac tuple_inversion :=
                                                                                                 match goal with
                                                                                                  | [ H : forall _, _ -> _,
H' : _ |- _ ] =>
apply H in H'; [idtac]
 match goal with
   | [ H : (_, _, _, _) = (_, _, _, _) |- _ ] => invc H
| [ H : (_, _, _) = (_, _, _) |- _ ] => invc H
    | [ H : (_, _) = (_, _) |- _ ] => invc H
                                                                                                   | [ H : _ -> _ , H' : _ |- _ ] =>
 end.
                                                                                                      apply H in H'; auto; [idtac]
                                                                                                 end.
Ltac f_apply H f :=
match type of H with
                                                                                               Ltac find_copy_apply_hyp_hyp :=
   | ?X = ?Y =>
                                                                                                 match goal with
      assert (f X = f Y) by (rewrite H; auto)
                                                                                                   | [ H : forall _, _ -> _,
```

Oct 24, 16 7:17	StructTactics.v	Page 5/7	Oct 24, 16 7:17	StructTactics.v	Page 6/7
H' : _   ] => copy_apply H H'; [idta   [ H :> _ , H' : _ copy_apply H H'; auto, end.	] =>		remGenIfNotVar f; remGenIfNotVar g   _ ?a ?b ?c ?d ?e ? remGenIfNotVar a; remGenIfNotVar b; remGenIfNotVar c;	f =>	
<pre>Ltac find_apply_lem_hyp lem match goal with</pre>			remGenIfNotVar d; remGenIfNotVar e; remGenIfNotVar f  ?a ?b ?c ?d ?e = remGenIfNotVar a;	>	
Ltac find_eapply_lem_hyp lem match goal with			remGenIfNotVar b; remGenIfNotVar c; remGenIfNotVar d; remGenIfNotVar e		
<pre>Ltac insterU H := match type of H with      forall _: ?T, _ =&gt;     let x := fresh "x" in     evar (x : T);     let x' := (eval unfold         clear x; specialize end.</pre>			<pre>  _ ?a ?b ?c ?d =&gt;     remGenIfNotVar a;     remGenIfNotVar b;     remGenIfNotVar c;     remGenIfNotVar d   _ ?a ?b ?c =&gt;     remGenIfNotVar a;     remGenIfNotVar b;     remGenIfNotVar c</pre>		
Ltac find_insterU := match goal with   [ H : forall _, _   end.	_] => insterU H		<pre>  _ ?a ?b =&gt;     remGenIfNotVar a;     remGenIfNotVar b   _ ?a =&gt;     remGenIfNotVar a end.</pre>		
Ltac eapply_prop P := match goal with   H : P _   => eapply H end. Ltac isVar t := match goal with   V : _   =>			Ltac generalizeEverythin repeat match goal with   [ x : ?T  - first [ match H   x = end   match t	_ ] => with > fail 2 ype of H with	
<pre>match t with      v =&gt; idtac    end end.</pre>			end   revert	text [x] => fail 2 x]	
<pre>Ltac remGen t := let x := fresh in let H := fresh in remember t as x eqn:H;</pre>			<pre>Ltac prep_induction H :=     rememberNonVars H;     generalizeEverythingEl</pre>		
<pre>generalize dependent H. Ltac remGenIfNotVar t := fi: Ltac rememberNonVars H :=</pre>	rst [isVar t  remGen t].		Ltac econcludes := match goal with   [ H : ?P -> _   end.	] => conclude H eauto	
<pre>match type of H with     _ ?a ?b ?c ?d ?e ?f ?c     remGenIfNotVar a;     remGenIfNotVar b;     remGenIfNotVar c;</pre>	g ?h =>		Ltac find_copy_eapply_lew match goal with		
<pre>remGenIfNotVar d; remGenIfNotVar e; remGenIfNotVar f; remGenIfNotVar g; remGenIfNotVar g; remGenIfNotVar h   _ ?a ?b ?c ?d ?e ?f ?d remGenIfNotVar a;</pre>	J =>		Ltac apply_prop_hyp P Q match goal with   [ H : context [ P ], apply H in H' end.	:= H' : context [ Q ]   ] =>	
remGenIfNotVar b; remGenIfNotVar c; remGenIfNotVar d; remGenIfNotVar e;			Ltac eapply_prop_hyp P Q match goal with   [ H : context [ P ], eapply H in H'	H' : context [ Q ]   ] =>	

Oct 24, 16 7:17	StructTactics.v	Page 7/7
end.		
Ltac copy_eapply_prop_hyp match goal with   [ H : context [ P ] copy_eapply H H' end.	p P Q := ], H' : context [ Q ]   ] =>	
Ltac find_false := match goal with   H :> False   end.	_ => exfalso; apply H	
Ltac injc H := injection H; clear H; :	intros; subst_max.	
[ H : ?X	_ = ?X  ] => injc H = ?X ] => injc H ?X [] => injc H X [] => injc H ] => injc H  ] => injc H	
Ltac aggressive_rewrite_ match goal with H : _		
Ltac break_exists_name x match goal with   [ H : exists _, _  - end.	:= _ ] => destruct H as [x H]	
let Htmp := fresh "H	alse -> x := fun false : False =>	
Tactic Notation "on" ucor match goal with   [H:?y  ] => unify x with tac H end.	nstr(x) "," tactic3(tac) := y;	
(** generic forward reas	oning *)	
Tactic Notation "fwd" tac simple refine (let H [ shelve   tac   clearbody H ].	<pre>ctic3(tac) "as" ident(H) := : _ := _ in _);</pre>	
Tactic Notation "fwd" tac let H := fresh "H" i fwd tac as H.		
Ltac ee := econstructor; eauto.		