Contents

Chapter 1

Library tarski_axioms

```
Require Export general_tactics.
```

This version of the axioms of Tarski is the one given in Wolfram Schwabhäuser, Wanda Szmielew and Alfred Tarski, Metamathematische Methoden in der Geometrie, Springer-Verlag, Berlin, 1983.

This axioms system is the result of a long history of axiom systems for geometry studied by Tarski, Schwabhäuser, Szmielew and Gupta.

```
Class Tarski_neutral_dimensionless := {
 Tpoint : Type;
 Bet : Tpoint → Tpoint → Tpoint → Prop;
 Cong : Tpoint → Tpoint → Tpoint → Tpoint → Prop;
 between_identity : ∀ A B, Bet A B A → A=B;
 cong_pseudo_reflexivity : ∀ A B : Tpoint, Cong A B B A;
 cong_identity : ∀ A B C : Tpoint, Cong A B C C → A = B;
 cong_inner_transitivity : ∀ A B C D E F : Tpoint,
   Cong A B C D → Cong A B E F → Cong C D E F;
 inner_pasch : ∀ A B C P Q : Tpoint,
       Bet A P C → Bet B Q C →
       ∃ x, Bet P x B ∧ Bet Q x A;
 five_segments : ∀ A A' B B' C C' D D' : Tpoint,
     Cong A B A' B' →
     Cong B C B' C' →
     Cong A D A' D' →
     Cong B D B' D' →
     Bet A B C → Bet A' B' C' → A ≠ B → Cong C D C' D';
 segment_construction : ∀ A B C D : Tpoint,
     ∃ E : Tpoint, Bet A B E ∧ Cong B E C D;
 lower_dim : ∃ A, ∃ B, ∃ C, ¬ (Bet A B C ∨ Bet B C A ∨ Bet C A B)
}.
Class Tarski_2D `(Tn : Tarski_neutral_dimensionless) := {
```

```
  upper_dim : ∀ A B C P Q : Tpoint,
    P ≠ Q → Cong A P A Q → Cong B P B Q → Cong C P C Q →
    (Bet A B C ∨ Bet B C A ∨ Bet C A B)
}.
Class Tarski_2D_euclidean ‘(T2D : Tarski_2D) := {
  euclid : ∀ A B C D T : Tpoint,
    Bet A D T → Bet B D C → A≠D →
    ∃ x, ∃ y,
    Bet A B x ∧ Bet A C y ∧ Bet x T y
}.
Class EqDecidability U := {
 eq_dec_points : ∀ A B : U, A=B ∨ ¬ A=B
}.
Class InterDecidability U (Col : U → U → U → Prop) := {
 inter_dec : ∀ A B C D,
   (∃ I, Col I A B ∧ Col I C D) ∨
 ¬ (∃ I, Col I A B ∧ Col I C D)
}.
```

We describe here a variant of the axiom system proposed by T.J.M. Makarios in June 2013. This variant has a slightly different five_segments axioms and allows to remove the cong_pseudo_reflexivity axiom. All axioms have been shown to be independent except cong_identity and inner_pasch.

```
Class Tarski_neutral_dimensionless_variant := {
 MTpoint : Type;
 BetM : MTpoint → MTpoint → MTpoint → Prop;
 CongM : MTpoint → MTpoint → MTpoint → MTpoint → Prop;
 Mbetween_identity : ∀ A B : MTpoint, BetM A B A → A = B;
 Mcong_identity : ∀ A B C : MTpoint, CongM A B C C → A = B;
 Mcong_inner_transitivity : ∀ A B C D E F : MTpoint,
   CongM A B C D → CongM A B E F → CongM C D E F;
 Minner_pasch : ∀ A B C P Q : MTpoint,
      BetM A P C → BetM B Q C →
      ∃ x, BetM P x B ∧ BetM Q x A;
 Mfive_segments : ∀ A A' B B' C C' D D' : MTpoint,
    CongM A B A' B' →
    CongM B C B' C' →
    CongM A D A' D' →
    CongM B D B' D' →
    BetM A B C → BetM A' B' C' → A ≠ B → CongM D C C' D';
 Msegment_construction : ∀ A B C D : MTpoint,
    ∃ E : MTpoint, BetM A B E ∧ CongM B E C D;
```

```
Mlower_dim : ∃ A, ∃ B, ∃ C, ¬ (BetM A B C ∨ BetM B C A ∨ BetM C A B)
}.
```

We describe here an intuitionistic version of Tarski's axiom system proposed by Michael Beeson.

```
Class intuitionistic_Tarski_neutral_dimensionless := {
 ITpoint : Type;
 IBet : ITpoint → ITpoint → ITpoint → Prop;
 ICong : ITpoint → ITpoint → ITpoint → ITpoint → Prop;
 Cong_stability : ∀ A B C D, ¬ ¬ ICong A B C D → ICong A B C D;
 Bet_stability : ∀ A B C, ¬ ¬ IBet A B C → IBet A B C;
 IT (A B C : ITpoint) := ¬ (A≠B ∧ B≠C ∧ ¬ IBet A B C);
 ICol (A B C : ITpoint) := A≠B ∧ ¬ (~ IT C A B ∧ ¬ IT A C B ∧ ¬ IT A B C);
 Ibetween_identity : ∀ A B, ¬ IBet A B A;
 Icong_identity : ∀ A B C, ICong A B C C → A = B;
 Icong_pseudo_reflexivity : ∀ A B : ITpoint, ICong A B B A;
 Icong_inner_transitivity : ∀ A B C D E F,
   ICong A B C D → ICong A B E F → ICong C D E F;
 Iinner_pasch : ∀ A B C P Q,
   IBet A P C → IBet B Q C → ¬ ICol A B C →
   ∃ x, IBet P x B ∧ IBet Q x A;
 Ibetween_symmetry : ∀ A B C, IBet A B C → IBet C B A;
 Ibetween_inner_transitivity : ∀ A B C D, IBet A B D → IBet B C D → IBet A B C;
 Ifive_segments : ∀ A A' B B' C C' D D',
     ICong A B A' B' →
     ICong B C B' C' →
     ICong A D A' D' →
     ICong B D B' D' →
     IT A B C → IT A' B' C' → A ≠ B → ICong C D C' D';
 Isegment_construction : ∀ A B C D,
     A≠B → ∃ E, IT A B E ∧ ICong B E C D;
 Ilower_dim : ∃ A, ∃ B, ∃ C, ¬ IT C A B ∧ ¬ IT A C B ∧ ¬ IT A B C
}.
```

Chapter 2

Library hilbert_axioms

```
Require Export Setoid.
Require Export aux.
```

We circumvent a limitation of type class definition by defining a polymorphic type for a triple of elements which will be used to define an angle by instantiating A with Point

Class **Hilbert** := {
 Point : Type;
 Line : Type;
 EqL : Line → Line → Prop;
 EqL_Equiv : **Equivalence** EqL;
 Incid : Point → Line → Prop;

 Group I Incidence line_existence : ∀ A B, $A \neq B$ → ∃ l, Incid A l ∧ Incid B l;
 line_unicity : ∀ A B l m, $A \neq B$ → Incid A l → Incid B l → Incid A m → Incid B m → EqL l m;
 two_points_on_line : ∀ l, ∃ A, ∃ B, Incid B l ∧ Incid A l ∧ $A \neq B$;
 ColH := fun A B C ⇒ ∃ l, Incid A l ∧ Incid B l ∧ Incid C l;
 plan : ∃ A, ∃ B, ∃ C, ¬ ColH A B C;

 Group II Order BetH : Point → Point → Point → Prop;
 between_col : ∀ A B C : Point, BetH A B C → ColH A B C;
 between_comm : ∀ A B C : Point, BetH A B C → BetH C B A;
 between_out : ∀ A B : Point, $A \neq B$ → ∃ C : Point, BetH A B C;
 between_only_one : ∀ A B C : Point, BetH A B C → ¬ BetH B C A ∧ ¬ BetH B A C;
 between_one : ∀ A B C, $A \neq B$ → $A \neq C$ → $B \neq C$ → ColH A B C → BetH A B C ∨ BetH B C A ∨ BetH B A C;
 cut := fun l A B ⇒ ¬ Incid A l ∧ ¬ Incid B l ∧ ∃ I, Incid I l ∧ BetH A I B;
 pasch : ∀ A B C l, ¬ ColH A B C → ¬ Incid C l → cut l A B → cut l A C ∨ cut l B C;

Group III Parallels $Para$:= fun l m ⇒ ¬ ∃ X, Incid X l ∧ Incid X m;
euclid_existence : ∀ l P, ¬ Incid P l → ∃ m, Para l m;
euclid_unicity : ∀ l P $m1$ $m2$, ¬ Incid P l → Para l $m1$ → Incid P $m1$→ Para l $m2$ → Incid P $m2$ → EqL $m1$ $m2$;

Group IV Congruence CongH : Point → Point → Point → Point → Prop;
cong_pseudo_transitivity : ∀ A B C D E F, CongH A B C D → CongH A B E F → CongH C D E F;
cong_refl : ∀ A B, CongH A B A B;
cong_existence : ∀ A B l M, $A \neq B$ → Incid M l → ∃ A', ∃ B',
Incid A' l ∧ Incid B' l ∧ BetH A' M B' ∧ CongH M A' A B ∧ CongH M B' A B;
cong_unicity : ∀ A B l M A' B' A'' B'', $A \neq B$ → Incid M l →
Incid A' l → Incid B' l →
Incid A'' l → Incid B'' l →
BetH A' M B' → CongH M A' A B →
CongH M B' A B → BetH A'' M B'' →
CongH M A'' A B →
CongH M B'' A B →
(A' = A'' ∧ B' = B'') ∨ (A' = B'' ∧ B' = A'');

$disjoint$:= fun A B C D ⇒ ¬ ∃ P, BetH A P B ∧ BetH C P D;
addition: ∀ A B C A' B' C', ColH A B C → ColH A' B' C' →
disjoint A B B C → disjoint A' B' B' C' →
CongH A B A' B' → CongH B C B' C' → CongH A C A' C';
$Angle$:= @**Triple** Point;
$angle$:= build_triple Point;
CongaH : Angle → Angle → Prop;
cong_5 : ∀ A B C A' B' C', ∀ $H1$: ($B \neq A$ ∧ $C \neq A$), ∀ $H2$: $B' \neq A'$ ∧ $C' \neq A'$,
∀ $H3$: ($A \neq B$ ∧ $C \neq B$), ∀ $H4$: $A' \neq B'$ ∧ $C' \neq B'$,
CongH A B A' B' → CongH A C A' C' → CongaH (angle B A C $H1$) (angle B' A' C' $H2$) →
CongaH (angle A B C $H3$) (angle A' B' C' $H4$);

$same\_side$:= fun A B l ⇒ ∃ P, cut l A P ∧ cut l B P;

$outH$:= fun P A B ⇒ BetH P A B ∨ BetH P B A ∨ ($P \neq A$ ∧ A = B);

$InAngleH$:= fun a P ⇒
(∃ M, BetH (V1 a) M (V2 a) ∧ ((outH (V a) M P) ∨ M = (V a))) ∨
outH (V a) (V1 a) P ∨ outH (V a) (V2 a) P;

```
Hline := @Couple Point;
line_of_hline : Hline → Line;
hline_construction := fun a (h: Hline) P (hc:Hline) H ⇒
(P1 h) = (P1 hc) ∧
CongaH a (angle (P2 h) (P1 h) (P2 hc) (conj (sym_not_equal (Cond h)) H)) ∧
 (∀ M, InAngleH (angle (P2 h) (P1 h) (P2 hc) (conj (sym_not_equal (Cond h)) H)) M →
  same_side P M (line_of_hline h));


 aux : ∀ (h h1 : Hline), P1 h = P1 h1 → P2 h1 ≠ P1 h;

 hcong_4_existence: ∀ a h P,
 ¬Incid P (line_of_hline h) → ¬ BetH (V1 a)(V a)(V2 a) →
 ∃ h1 , (P1 h) = (P1 h1) ∧ (∀ CondAux : P1 h = P1 h1,
                                          CongaH a (angle (P2 h) (P1 h) (P2 h1) (conj (sym_not_equal
(Cond h)) (aux h h1 CondAux)))
                                          ∧ (∀ M, ¬ Incid M (line_of_hline h) ∧ InAngleH
(angle (P2 h) (P1 h) (P2 h1) (conj (sym_not_equal (Cond h)) (aux h h1 CondAux))) M
                                                        → same_side P M (line_of_hline
h)));


hEq : relation Hline := fun h1 h2 ⇒ (P1 h1) = (P1 h2) ∧
                          ((P2 h1) = (P2 h2) ∨ BetH (P1 h1) (P2 h2) (P2 h1) ∨
                                                 BetH (P1 h1) (P2 h1) (P2 h2));

hcong_4_unicity :
 ∀ a h P h1 h2 HH1 HH2,
 ¬Incid P (line_of_hline h) → ¬ BetH (V1 a)(V a)(V2 a) →
 hline_construction a h P h1 HH1 → hline_construction a h P h2 HH2 →
 hEq h1 h2

}.
```

Chapter 3

Library Ch02_cong

```
Require Export tarski_axioms.

Ltac prolong A B x C D :=
 assert (sg:= segment_construction A B C D);
 ex_and sg x.

Ltac cases_equality A B := elim (eq_dec_points A B);intros.

Section T1_1.

Context `{M:Tarski_neutral_dimensionless}.

Lemma cong_reflexivity : ∀ A B,
 Cong A B A B.

Lemma cong_symmetry : ∀ A B C D : Tpoint,
 Cong A B C D → Cong C D A B.

Lemma cong_transitivity : ∀ A B C D E F : Tpoint,
 Cong A B C D → Cong C D E F → Cong A B E F.

Lemma cong_left_commutativity : ∀ A B C D,
 Cong A B C D → Cong B A C D.

Lemma cong_right_commutativity : ∀ A B C D,
 Cong A B C D → Cong A B D C.

Lemma cong_trivial_identity : ∀ A B : Tpoint,
 Cong A A B B.

Lemma cong_reverse_identity : ∀ A C D,
 Cong A A C D → C=D.

Lemma cong_commutativity : ∀ A B C D,
 Cong A B C D → Cong B A D C.

End T1_1.

Hint Resolve cong_commutativity cong_reverse_identity cong_trivial_identity
             cong_left_commutativity cong_right_commutativity
```

```
                cong_transitivity cong_symmetry cong_reflexivity cong_identity : cong.
Ltac Cong := auto with cong.
Ltac eCong := eauto with cong.

Section T1_2.

Context `{M:Tarski_neutral_dimensionless}.

Lemma cong_dec_eq_dec :
 (∀ A B C D, Cong A B C D ∨ ¬ Cong A B C D) →
 (∀ A B:Tpoint, A=B ∨ A≠B).

Definition OFSC := fun A B C D A' B' C' D' ⇒
  Bet A B C ∧ Bet A' B' C' ∧
  Cong A B A' B' ∧ Cong B C B' C' ∧
  Cong A D A' D' ∧ Cong B D B' D'.

Lemma five_segments_with_def : ∀ A B C D A' B' C' D',
 OFSC A B C D A' B' C' D' → A≠B → Cong C D C' D'.

Lemma cong_diff : ∀ A B C D : Tpoint, A ≠ B → Cong A B C D → C ≠ D.

Lemma cong_diff_2 : ∀ A B C D ,
  B ≠ A → Cong A B C D → C ≠ D.

Lemma cong_diff_3 : ∀ A B C D ,
  C ≠ D → Cong A B C D → A ≠ B.

Lemma cong_diff_4 : ∀ A B C D ,
  D ≠ C → Cong A B C D → A ≠ B.

Definition Cong_3 := fun A1 A2 A3 B1 B2 B3 ⇒ Cong A1 A2 B1 B2 ∧ Cong A1 A3
B1 B3 ∧ Cong A2 A3 B2 B3.

Lemma cong_3_sym : ∀ A B C A' B' C',
  Cong_3 A B C A' B' C' → Cong_3 A' B' C' A B C.

Lemma cong_3_swap : ∀ A B C A' B' C',
  Cong_3 A B C A' B' C' → Cong_3 B A C B' A' C'.

Lemma cong_3_swap_2 : ∀ A B C A' B' C',
  Cong_3 A B C A' B' C' → Cong_3 A C B A' C' B'.

Lemma cong3_transitivity : ∀ A0 B0 C0 A1 B1 C1 A2 B2 C2,
        Cong_3 A0 B0 C0 A1 B1 C1 → Cong_3 A1 B1 C1 A2 B2 C2 → Cong_3 A0 B0
C0 A2 B2 C2.

End T1_2.

Hint Resolve cong_3_sym : cong.
Hint Resolve cong_3_swap cong_3_swap_2 cong3_transitivity : cong3.
Hint Unfold Cong_3 : cong3.

Section T1_3.
```

```
Context `{M:Tarski_neutral_dimensionless}.
Context `{EqDec:EqDecidability Tpoint}.

Lemma l2_11 : ∀ A B C A' B' C',
 Bet A B C → Bet A' B' C' → Cong A B A' B' → Cong B C B' C' → Cong A C A' C'.

Lemma construction_unicity : ∀ Q A B C X Y,
 Q ≠ A → Bet Q A X → Cong A X B C → Bet Q A Y → Cong A Y B C → X=Y.

Lemma Cong_cases :
  ∀ A B C D,
  Cong A B C D ∨ Cong A B D C ∨ Cong B A C D ∨ Cong B A D C ∨
  Cong C D A B ∨ Cong C D B A ∨ Cong D C A B ∨ Cong D C B A →
  Cong A B C D.

Lemma Cong_perm :
  ∀ A B C D,
  Cong A B C D →
  Cong A B C D ∧ Cong A B D C ∧ Cong B A C D ∧ Cong B A D C ∧
  Cong C D A B ∧ Cong C D B A ∧ Cong D C A B ∧ Cong D C B A.

End T1_3.
```

Chapter 4

Library Ch03_bet

```
Require Export Ch02_cong.
Section T2_1.
Context `{M:Tarski_neutral_dimensionless}.
Definition Col (A B C : Tpoint) : Prop := Bet A B C ∨ Bet B C A ∨ Bet C A B.
Lemma bet_col : ∀ A B C, Bet A B C → Col A B C.
Lemma between_trivial : ∀ A B : Tpoint, Bet A B B.
Lemma bet_dec_eq_dec :
  (∀ A B C, Bet A B C ∨ ¬ Bet A B C) →
  (∀ A B:Tpoint, A=B ∨ A≠B).
Lemma between_symmetry : ∀ A B C : Tpoint, Bet A B C → Bet C B A.
```

This lemma is used by tactics for trying several permutations.

```
Lemma Bet_cases :
 ∀ A B C,
 Bet A B C ∨ Bet C B A →
 Bet A B C.
Lemma Bet_perm :
  ∀ A B C,
 Bet A B C →
 Bet A B C ∧ Bet C B A.
Lemma between_trivial2 : ∀ A B : Tpoint, Bet A A B.
Lemma between_egality : ∀ A B C : Tpoint, Bet A B C → Bet B A C → A = B.
End T2_1.
Ltac assert_cols :=
repeat
 match goal with
      | H:Bet ?X1 ?X2 ?X3 ⊢ _ ⇒
     not_exist_hyp (Col X1 X2 X3);assert (Col X1 X2 X3) by (apply bet_col;apply H)
```

```
 end.

Ltac clean_trivial_hyps :=
  repeat
  match goal with
   | H:(Cong ?X1 ?X1 ?X2 ?X2) ⊢ _ ⇒ clear H
   | H:(Cong ?X1 ?X2 ?X2 ?X1) ⊢ _ ⇒ clear H
   | H:(Cong ?X1 ?X2 ?X1 ?X2) ⊢ _ ⇒ clear H
   | H:(Bet ?X1 ?X1 ?X2) ⊢ _ ⇒ clear H
   | H:(Bet ?X2 ?X1 ?X1) ⊢ _ ⇒ clear H
   | H:(Col ?X1 ?X1 ?X2) ⊢ _ ⇒ clear H
   | H:(Col ?X2 ?X1 ?X1) ⊢ _ ⇒ clear H
   | H:(Col ?X1 ?X2 ?X1) ⊢ _ ⇒ clear H
end.

Ltac clean_reap_hyps :=
  repeat
  match goal with

   | H:(¬Col ?A ?B ?C), H2 : ¬Col ?A ?B ?C ⊢ _ ⇒ clear H2
   | H:(Col ?A ?B ?C), H2 : Col ?A ?C ?B ⊢ _ ⇒ clear H2
   | H:(Col ?A ?B ?C), H2 : Col ?A ?B ?C ⊢ _ ⇒ clear H2
   | H:(Col ?A ?B ?C), H2 : Col ?B ?A ?C ⊢ _ ⇒ clear H2
   | H:(Col ?A ?B ?C), H2 : Col ?B ?C ?A ⊢ _ ⇒ clear H2
   | H:(Col ?A ?B ?C), H2 : Col ?C ?B ?A ⊢ _ ⇒ clear H2
   | H:(Col ?A ?B ?C), H2 : Col ?C ?A ?B ⊢ _ ⇒ clear H2
   | H:(Bet ?A ?B ?C), H2 : Bet ?C ?B ?A ⊢ _ ⇒ clear H2
   | H:(Bet ?A ?B ?C), H2 : Bet ?A ?B ?C ⊢ _ ⇒ clear H2
   | H:(Cong ?A ?B ?C ?D), H2 : Cong ?A ?B ?D ?C ⊢ _ ⇒ clear H2
   | H:(Cong ?A ?B ?C ?D), H2 : Cong ?A ?B ?C ?D ⊢ _ ⇒ clear H2
   | H:(Cong ?A ?B ?C ?D), H2 : Cong ?C ?D ?A ?B ⊢ _ ⇒ clear H2
   | H:(Cong ?A ?B ?C ?D), H2 : Cong ?C ?D ?B ?A ⊢ _ ⇒ clear H2
   | H:(Cong ?A ?B ?C ?D), H2 : Cong ?D ?C ?B ?A ⊢ _ ⇒ clear H2
   | H:(Cong ?A ?B ?C ?D), H2 : Cong ?D ?C ?A ?B ⊢ _ ⇒ clear H2
   | H:(Cong ?A ?B ?C ?D), H2 : Cong ?B ?A ?C ?D ⊢ _ ⇒ clear H2
   | H:(Cong ?A ?B ?C ?D), H2 : Cong ?B ?A ?D ?C ⊢ _ ⇒ clear H2
   | H:(?A<>?B), H2 : (?B<>?A) ⊢ _ ⇒ clear H2
   | H:(?A<>?B), H2 : (?A<>?B) ⊢ _ ⇒ clear H2
end.

Ltac clean := clean_trivial_hyps;clean_duplicated_hyps;clean_reap_hyps.

Ltac smart_subst X := subst X;clean.

Ltac treat_equalities_aux :=
  repeat
```

```
  match goal with
   | H:(?X1 = ?X2) ⊢ _ ⇒ smart_subst X2
end.

Ltac treat_equalities :=
try treat_equalities_aux;
repeat
  match goal with
   | H:(Cong ?X3 ?X3 ?X1 ?X2) ⊢ _ ⇒
      apply cong_symmetry in H; apply cong_identity in H;smart_subst X2
   | H:(Cong ?X1 ?X2 ?X3 ?X3) ⊢ _ ⇒
      apply cong_identity in H;smart_subst X2
   | H:(Bet ?X1 ?X2 ?X1) ⊢ _ ⇒ apply between_identity in H;smart_subst X2
end.

Ltac show_distinct X Y := assert (X≠Y);[unfold not;intro;treat_equalities|idtac].

Hint Resolve between_symmetry bet_col : between.
Hint Resolve between_trivial between_trivial2 : between_no_eauto.

Ltac eBetween := treat_equalities;eauto with between.
Ltac Between := treat_equalities;auto with between between_no_eauto.

Section T2_2.

Context `{M:Tarski_neutral_dimensionless}.
Context `{EqDec:EqDecidability Tpoint}.

Lemma between_inner_transitivity : ∀ A B C D, Bet A B D → Bet B C D → Bet A B C.

Lemma between_exchange3 : ∀ A B C D, Bet A B C → Bet A C D → Bet B C D.

Lemma outer_transitivity_between2 : ∀ A B C D, Bet A B C → Bet B C D → B≠C → Bet
A C D.

End T2_2.

Hint Resolve outer_transitivity_between2 between_inner_transitivity between_exchange3 : between.

Section T2_3.

Context `{M:Tarski_neutral_dimensionless}.
Context `{EqDec:EqDecidability Tpoint}.

Lemma between_exchange2 : ∀ A B C D, Bet A B D → Bet B C D → Bet A C D.

Lemma outer_transitivity_between : ∀ A B C D, Bet A B C → Bet B C D → B≠C → Bet
A B D.

End T2_3.

Hint Resolve between_exchange2 : between.

Section T2_4.

Context `{M:Tarski_neutral_dimensionless}.
```

Context `{$EqDec$:**EqDecidability** Tpoint}.

Lemma between_exchange4 : ∀ A B C D, Bet A B C → Bet A C D → Bet A B D.

End T2_4.

Hint Resolve outer_transitivity_between between_exchange4 : *between*.

Section T2_5.

Context `{M:**Tarski_neutral_dimensionless**}.
Context `{$EqDec$:**EqDecidability** Tpoint}.

Definition Bet_4 := fun $A1$ $A2$ $A3$ $A4$ ⇒
Bet $A1$ $A2$ $A3$ ∧ Bet $A2$ $A3$ $A4$ ∧ Bet $A1$ $A3$ $A4$ ∧ Bet $A1$ $A2$ $A4$.

Lemma l_3_9_4 : ∀ $A1$ $A2$ $A3$ $A4$, Bet_4 $A1$ $A2$ $A3$ $A4$ → Bet_4 $A4$ $A3$ $A2$ $A1$.

Lemma l3_17 : ∀ A B C A' B' P,
Bet A B C → Bet A' B' C → Bet A P A' → ∃ Q , Bet P Q C ∧ Bet B Q B'.

Lemma two_distinct_points : ∃ X , ∃ Y: Tpoint , $X \neq Y$.

Lemma point_construction_different : ∀ A B, ∃ C , Bet A B C ∧ $B \neq C$.

Lemma another_point : ∀ A: Tpoint, ∃ B , $A \neq B$.

End T2_5.

Section Beeson_1.

Another proof of l2_11 without eq_dec_points but using Cong stability inspired by Micheal Beeson.

Context `{M:**Tarski_neutral_dimensionless**}.

Variable *Cong_stability* : ∀ A B C D, ¬ ¬ Cong A B C D → Cong A B C D.

Lemma l2_11_b : ∀ A B C A' B' C',
Bet A B C → Bet A' B' C' → Cong A B A' B' → Cong B C B' C' → Cong A C A' C'.

Lemma cong_dec_eq_dec_b :
(∀ A B:Tpoint, ¬ $A \neq B$ → $A = B$).

End Beeson_1.

Section Beeson_2.

Context `{M:**Tarski_neutral_dimensionless**}.

Variable *Bet_stability* : ∀ A B C, ¬ ¬ Bet A B C → Bet A B C.

Lemma bet_dec_eq_dec_b :
(∀ A B:Tpoint, ¬ $A \neq B$ → $A = B$).

End Beeson_2.

Chapter 5

Library Ch04_cong_bet

```
Require Export Ch03_bet.

Section T3.

Context `{M:Tarski_neutral_dimensionless}.
Context `{EqDec:EqDecidability Tpoint}.

Definition IFSC := fun A B C D A' B' C' D' ⇒
    Bet A B C ∧ Bet A' B' C' ∧
    Cong A C A' C' ∧ Cong B C B' C' ∧
    Cong A D A' D' ∧ Cong C D C' D'.

Lemma l4_2 : ∀ A B C D A' B' C' D', IFSC A B C D A' B' C' D' → Cong B D B' D'.

Lemma l4_3 : ∀ A B C A' B' C',
  Bet A B C → Bet A' B' C' → Cong A C A' C' → Cong B C B' C' → Cong A B A' B'.

Lemma l4_5 : ∀ A B C A' C',
  Bet A B C → Cong A C A' C' →
  ∃ B', Bet A' B' C' ∧ Cong_3 A B C A' B' C'.

Lemma l4_6 : ∀ A B C A' B' C', Bet A B C → Cong_3 A B C A' B' C' → Bet A' B' C'.

Lemma cong3_bet_eq : ∀ A B C X,
 Bet A B C → Cong_3 A B C A X C → X = B.

End T3.
```

Chapter 6

Library Ch04_col

Require Export Ch04_cong_bet.

Section T4_1.

Context ‘{M:**Tarski_neutral_dimensionless**}.
Context ‘{$EqDec$:**EqDecidability** Tpoint}.

Lemma col_permutation_1 : ∀ A B C,Col A B C → Col B C A.

Lemma col_permutation_2 : ∀ A B C, Col A B C → Col C A B.

Lemma col_permutation_3 : ∀ A B C, Col A B C → Col C B A.

Lemma col_permutation_4 : ∀ A B C, Col A B C → Col B A C.

Lemma col_permutation_5 : ∀ A B C, Col A B C → Col A C B.

End T4_1.

Hint Resolve bet_col col_permutation_1 col_permutation_2
col_permutation_3 col_permutation_4 col_permutation_5 : *col*.

Ltac *Col* := auto 3 with *col*.
Ltac *Col5* := auto with *col*.

Section T4_2.

Context ‘{M:**Tarski_neutral_dimensionless**}.
Context ‘{$EqDec$:**EqDecidability** Tpoint}.

Lemma not_col_permutation_1 :
∀ (A B C : Tpoint), ¬ Col A B C → ¬ Col B C A.

Lemma not_col_permutation_2 :
∀ (A B C : Tpoint), ¬ Col A B C → ¬ Col C A B.

Lemma not_col_permutation_3 :
∀ (A B C : Tpoint), ¬ Col A B C → ¬ Col C B A.

Lemma not_col_permutation_4 :
∀ (A B C : Tpoint), ¬ Col A B C → ¬ Col B A C.

Lemma not_col_permutation_5 :
∀ (A B C : Tpoint), ¬ Col A B C → ¬ Col A C B.

End T4_2.

Hint Resolve not_col_permutation_1 not_col_permutation_2 not_col_permutation_3 not_col_permutation_4 not_col_permutation_5 : *col*.

Section T4_3.

Context ‘{M:**Tarski_neutral_dimensionless**}.
Context ‘{EqDec:**EqDecidability** Tpoint}.

This lemma is used by tactics for trying several permutations. Lemma Col_cases :
∀ A B C,
Col A B C ∨ Col A C B ∨ Col B A C ∨
Col B C A ∨ Col C A B ∨ Col C B A →
Col A B C.

Lemma Col_perm :
∀ A B C,
Col A B C →
Col A B C ∧ Col A C B ∧ Col B A C ∧
Col B C A ∧ Col C A B ∧ Col C B A.

Lemma col_trivial_1 : ∀ A B, Col A A B.

Lemma col_trivial_2 : ∀ A B, Col A B B.

Lemma col_trivial_3 : ∀ A B, Col A B A.

End T4_3.

Hint Immediate col_trivial_1 col_trivial_2 col_trivial_3: *col*.

Section T4_4.

Context ‘{M:**Tarski_neutral_dimensionless**}.
Context ‘{EqDec:**EqDecidability** Tpoint}.

Lemma l4_13 : ∀ A B C A' B' C',
Col A B C → Cong_3 A B C A' B' C' → Col A' B' C'.

Lemma l4_14 : ∀ A B C A' B',
Col A B C → Cong A B A' B' → ∃ C', Cong_3 A B C A' B' C'.

Definition FSC := fun A B C D A' B' C' D' ⇒
Col A B C ∧ Cong_3 A B C A' B' C' ∧ Cong A D A' D' ∧ Cong B D B' D'.

Lemma l4_16 : ∀ A B C D A' B' C' D',
FSC A B C D A' B' C' D' → A≠B → Cong C D C' D'.

Lemma l4_17 : ∀ A B C P Q, A≠B → Col A B C → Cong A P A Q → Cong B P B Q → Cong C P C Q.

Lemma l4_18 : ∀ A B C C',

$A \neq B \rightarrow$ Col A B $C \rightarrow$ Cong A C A $C' \rightarrow$ Cong B C B $C' \rightarrow C = C'$.

Lemma l4_19 : $\forall$ A B C C',
Bet A C $B \rightarrow$ Cong A C A $C' \rightarrow$ Cong B C B $C' \rightarrow C = C'$.

Lemma not_col_distincts : $\forall$ A B C ,
$\neg$Col A B $C \rightarrow$
$\neg$Col A B $C \land A \neq B \land B \neq C \land A \neq C$.

Lemma NCol_cases :
$\forall$ A B C,
$\neg$ Col A B $C \lor \neg$ Col A C $B \lor \neg$ Col B A $C \lor$
$\neg$ Col B C $A \lor \neg$ Col C A $B \lor \neg$ Col C B $A \rightarrow$
$\neg$ Col A B C.

Lemma NCol_perm :
$\forall$ A B C,
$\neg$ Col A B $C \rightarrow$
$\neg$ Col A B $C \land \neg$ Col A C $B \land \neg$ Col B A $C \land$
$\neg$ Col B C $A \land \neg$ Col C A $B \land \neg$ Col C B A.

End T4_4.

Chapter 7

Library Ch05_bet_le

```
Require Export Ch04_col.

Section T5.

Context `{M:Tarski_neutral_dimensionless}.
Context `{EqDec:EqDecidability Tpoint}.

Lemma l5_1 : ∀ A B C D,
 A≠B → Bet A B C → Bet A B D → Bet A C D ∨ Bet A D C.

Lemma l5_2 : ∀ A B C D,
 A≠B → Bet A B C → Bet A B D → Bet B C D ∨ Bet B D C.

Lemma l5_3 : ∀ A B C D,
 Bet A B D → Bet A C D → Bet A B C ∨ Bet A C B.

Definition le := fun A B C D ⇒
   ∃ y , Bet C y D ∧ Cong A B C y.

Definition ge := fun A B C D ⇒ le C D A B.

Lemma l5_5_1 : ∀ A B C D,
  le A B C D → ∃ x , Bet A B x ∧ Cong A x C D.

Lemma l5_5_2 : ∀ A B C D,
 (∃ x , Bet A B x ∧ Cong A x C D) → le A B C D.

Lemma l5_6 : ∀ A B C D A' B' C' D',
 le A B C D ∧ Cong A B A' B' ∧ Cong C D C' D' → le A' B' C' D'.

Lemma le_reflexivity : ∀ A B, le A B A B.

Lemma le_transitivity : ∀ A B C D E F, le A B C D → le C D E F → le A B E F .

Lemma between_cong : ∀ A B C, Bet A C B → Cong A C A B → C=B.

Lemma cong3_symmetry : ∀ A B C A' B' C' : Tpoint , Cong_3 A B C A' B' C' → Cong_3
A' B' C' A B C.

Lemma between_cong_2 : ∀ A B D E, Bet A D B → Bet A E B → Cong A D A E → D = E.
```

Lemma between_cong_3 : $\forall$ A B D E, $A \neq B \rightarrow$ Bet A B $D \rightarrow$ Bet A B $E \rightarrow$ Cong B D B $E \rightarrow D = E$.

Lemma le_anti_symmetry : $\forall$ A B C D, le A B C $D \rightarrow$ le C D A $B \rightarrow$ Cong A B C D.

Lemma segment_construction_2 : $\forall$ A Q B C, $A \neq Q \rightarrow \exists$ X, (Bet Q A X $\vee$ Bet Q X A) $\wedge$ Cong Q X B C.

Lemma Cong_dec : $\forall$ A B C D,
  Cong A B C D $\vee$ $\neg$ Cong A B C D.

Lemma Bet_dec : $\forall$ A B C, Bet A B C $\vee$ $\neg$ Bet A B C.

Lemma Col_dec : $\forall$ A B C, Col A B C $\vee$ $\neg$ Col A B C.

Lemma le_trivial : $\forall$ A C D, le A A C D .

Lemma le_cases : $\forall$ A B C D, le A B C D $\vee$ le C D A B.

Lemma le_zero : $\forall$ A B C, le A B C $C \rightarrow A = B$.

Lemma bet_cong_eq :
 $\forall$ A B C D,
  Bet A B $C \rightarrow$
  Bet A C $D \rightarrow$
  Cong B C A $D \rightarrow$
  $C = D \wedge A = B$.

Definition lt := fun A B C $D \Rightarrow$ le A B C D $\wedge$ $\neg$ Cong A B C D.
Definition gt := fun A B C $D \Rightarrow$ lt C D A B.

Lemma fourth_point : $\forall$ A B C P, $A \neq B \rightarrow B \neq C \rightarrow$ Col A B $P \rightarrow$ Bet A B $C \rightarrow$
  Bet P A B $\vee$ Bet A P B $\vee$ Bet B P C $\vee$ Bet B C P.

Lemma third_point : $\forall$ A B P, Col A B $P \rightarrow$ Bet P A B $\vee$ Bet A P B $\vee$ Bet A B P.

End T5.

Chapter 8

Library Ch06_out_lines

Require Export Ch05_bet_le.

Ltac $eCol$:= eauto with col.

Section T6_1.

Context `{M:**Tarski_neutral_dimensionless**}.
Context `{$EqDec$:**EqDecidability** Tpoint}.

Definition out := fun P A B ⇒ $A{\neq}P$ ∧ $B{\neq}P$ ∧ (Bet P A B ∨ Bet P B A).

Lemma out_dec : ∀ P A B, out P A B ∨ ¬ out P A B.

Lemma out_diff1 : ∀ A B C, out A B C → $B \neq A$.

Lemma out_diff2 : ∀ A B C, out A B C → $C \neq A$.

Lemma out_distinct : ∀ A B C, out A B C → $B \neq A$ ∧ $C \neq A$.

Lemma out_col : ∀ A B C, out A B C → Col A B C.

Lemma l6_2 : ∀ A B C P, $A{\neq}P$ → $B{\neq}P$ → $C{\neq}P$ → Bet A P C → (Bet B P C ↔ out P A B).

Lemma l6_3_1 : ∀ A B P, out P A B → ($A{\neq}P$ ∧ $B{\neq}P$ ∧ ∃ C , $C{\neq}P$ ∧ Bet A P C ∧ Bet B P C).

Lemma l6_3_2 : ∀ A B P,
 ($A{\neq}P$ ∧ $B{\neq}P$ ∧ ∃ C , $C{\neq}P$ ∧ Bet A P C ∧ Bet B P C) → out P A B.

Lemma l6_4_1 : ∀ A B P, out P A B → Col A P B ∧ ¬ Bet A P B.

Lemma l6_4_2 : ∀ A B P, Col A P B ∧ ¬ Bet A P B → out P A B.

out reflexivity. l6_5

Lemma out_trivial : ∀ P A, $A{\neq}P$ → out P A A.

out symmetry.

Lemma l6_6 : ∀ P A B, out P A B → out P B A.

out transitivity.

Lemma l6_7 : ∀ $P\ A\ B\ C$, out $P\ A\ B$ → out $P\ B\ C$ → out $P\ A\ C$.

Lemma bet_out_out_bet : ∀ $A\ B\ C\ A'\ C'$,
Bet $A\ B\ C$ → out $B\ A\ A'$ → out $B\ C\ C'$ → Bet $A'\ B\ C'$.

Lemma out2_bet_out : ∀ $A\ B\ C\ X\ P$,
out $B\ A\ C$ → out $B\ X\ P$ → Bet $A\ X\ C$ → out $B\ A\ P$ ∧ out $B\ C\ P$.

Lemma l6_11_unicity : ∀ $A\ B\ C\ R\ X\ Y$,
$R{\neq}A$ → $B{\neq}C$ →
out $A\ X\ R$ → Cong $A\ X\ B\ C$ →
out $A\ Y\ R$ → Cong $A\ Y\ B\ C$ →
$X{=}Y$.

Lemma l6_11_existence : ∀ $A\ B\ C\ R$,
$R{\neq}A$ → $B{\neq}C$ → ∃ X , out $A\ X\ R$ ∧ Cong $A\ X\ B\ C$.

Lemma l6_13_1 : ∀ $P\ A\ B$, out $P\ A\ B$ → le $P\ A\ P\ B$ → Bet $P\ A\ B$.

Lemma l6_13_2 : ∀ $P\ A\ B$, out $P\ A\ B$ → Bet $P\ A\ B$ → le $P\ A\ P\ B$.

Lemma l6_16_1 : ∀ $P\ Q\ S\ X$, $P{\neq}Q$ → $S{\neq}P$ → Col $S\ P\ Q$ → Col $X\ P\ Q$ → Col $X\ P\ S$.

Lemma col_transitivity_1 : ∀ $P\ Q\ A\ B$,
$P{\neq}Q$ → Col $P\ Q\ A$ → Col $P\ Q\ B$ → Col $P\ A\ B$.

Lemma col_transitivity_2 : ∀ $P\ Q\ A\ B$,
$P{\neq}Q$ → Col $P\ Q\ A$ → Col $P\ Q\ B$ → Col $Q\ A\ B$.

Unicity of intersection

Lemma l6_21 : ∀ $A\ B\ C\ D\ P\ Q$,
¬ Col $A\ B\ C$ → $C{\neq}D$ → Col $A\ B\ P$ → Col $A\ B\ Q$ → Col $C\ D\ P$ → Col $C\ D\ Q$ → $P{=}Q$.

End T6_1.

Hint Resolve col_transitivity_1 col_transitivity_2 : *col*.

Section T6_2.

Context ‘{M:**Tarski_neutral_dimensionless**}.
Context ‘{$EqDec$:**EqDecidability** Tpoint}.

Lemma l6_25 : ∀ $A\ B$,
$A{\neq}B$ → ∃ C , ¬ Col $A\ B\ C$.

Lemma t2_8 : ∀ $A\ B\ C\ D\ E$: Tpoint,
Bet $A\ B\ C$ → Bet $D\ B\ E$ → Cong $A\ B\ D\ B$ → Cong $B\ C\ B\ E$ → Cong $A\ E\ C\ D$.

Lemma col3 : ∀ $X\ Y\ A\ B\ C$,
$X \neq Y$ →
Col $X\ Y\ A$ → Col $X\ Y\ B$ → Col $X\ Y\ C$ →
Col $A\ B\ C$.

End T6_2.

Chapter 9

Library Ch07_midpoint

Require Export Ch06_out_lines.

Section T7_1.

Context `{MT:**Tarski_neutral_dimensionless**}.
Context `{$EqDec$:**EqDecidability** Tpoint}.

Definition is_midpoint := fun M A B ⇒ Bet A M B ∧ Cong A M M B.

Lemma is_midpoint_dec :
∀ I A B, is_midpoint I A B ∨ ¬ is_midpoint I A B.

Lemma is_midpoint_id : ∀ A B, is_midpoint A A B → A = B.

Lemma is_midpoint_id_2 : ∀ A B, is_midpoint A B A → A=B.

Lemma l7_2 : ∀ M A B, is_midpoint M A B → is_midpoint M B A.

Lemma l7_3 : ∀ M A, is_midpoint M A A → M=A.

Lemma l7_3_2 : ∀ A, is_midpoint A A A.

Lemma symmetric_point_construction : ∀ A P, ∃ P', is_midpoint P A P'.

Lemma symmetric_point_unicity : ∀ A P $P1$ $P2$, is_midpoint P A $P1$ → is_midpoint P A $P2$ → $P1$=$P2$.

Lemma l7_9 : ∀ P Q A X, is_midpoint A P X → is_midpoint A Q X → P=Q.

Lemma diff_bet : ∀ A B C, $A \neq B$ → Bet A B C → $A \neq C$.

Lemma l7_13 : ∀ A P Q P' Q', is_midpoint A P' P → is_midpoint A Q' Q → Cong P Q P' Q'.

Lemma l7_15 : ∀ P Q R P' Q' R' A,
is_midpoint A P P' → is_midpoint A Q Q' → is_midpoint A R R' → Bet P Q R → Bet P' Q' R'.

Lemma l7_16 : ∀ P Q R S P' Q' R' S' A,
is_midpoint A P P' → is_midpoint A Q Q' →
is_midpoint A R R' → is_midpoint A S S' →

```
  Cong P Q R S → Cong P' Q' R' S'.

Lemma symmetry_preserves_midpoint :
   ∀ A B C D E F Z,
 is_midpoint Z A D → is_midpoint Z B E →
 is_midpoint Z C F → is_midpoint B A C → is_midpoint E D F.

End T7_1.

Hint Resolve l7_13 : cong.
Hint Resolve l7_2 l7_3 l7_3_2 symmetric_point_construction symmetry_preserves_midpoint :
midpoint.

Ltac Midpoint := auto with midpoint.

Section T7_2.

Context `{MT:Tarski_neutral_dimensionless}.
Context `{EqDec:EqDecidability Tpoint}.

Lemma Mid_cases :
  ∀ A B C,
  is_midpoint A B C ∨ is_midpoint A C B →
  is_midpoint A B C.

Lemma Mid_perm :
  ∀ A B C,
  is_midpoint A B C →
  is_midpoint A B C ∧ is_midpoint A C B.

Lemma l7_17 : ∀ P P' A B, is_midpoint A P P' → is_midpoint B P P' → A=B.

Lemma l7_17_bis : ∀ P P' A B, is_midpoint A P P' → is_midpoint B P' P → A=B.

Lemma l7_20 : ∀ M A B,
  Col A M B → Cong M A M B → A=B ∨ is_midpoint M A B.

Lemma cong_col_mid : ∀ A B C,
 A ≠ C → Col A B C → Cong A B B C →
 is_midpoint B A C.

Lemma l7_21 : ∀ A B C D P,
  ¬ Col A B C → B≠D →
  Cong A B C D → Cong B C D A →
  Col A P C → Col B P D →
  is_midpoint P A C ∧ is_midpoint P B D.

Lemma l7_22_aux : ∀ A1 A2 B1 B2 C M1 M2,
    Bet A1 C A2 → Bet B1 C B2 →
    Cong C A1 C B1 → Cong C A2 C B2 →
    is_midpoint M1 A1 B1 → is_midpoint M2 A2 B2 →
    le C A1 C A2 →
```

Bet $M1\ C\ M2$.

This is Krippen lemma , proved by Gupta in its PhD in 1965 as Theorem 3.45

Lemma l7_22 : ∀ $A1\ A2\ B1\ B2\ C\ M1\ M2$,
Bet $A1\ C\ A2$ → Bet $B1\ C\ B2$ →
Cong $C\ A1\ C\ B1$ → Cong $C\ A2\ C\ B2$ →
is_midpoint $M1\ A1\ B1$ → is_midpoint $M2\ A2\ B2$ →
Bet $M1\ C\ M2$.

Lemma bet_col1 : ∀ $A\ B\ C\ D$, Bet $A\ B\ D$ → Bet $A\ C\ D$ → Col $A\ B\ C$.

Lemma bet_col2 : ∀ $A\ B\ C\ D$,
$A \neq B$ → Bet $A\ B\ C$ → Bet $A\ B\ D$ →
Col $A\ C\ D$.

Lemma l7_25 : ∀ $A\ B\ C$,
Cong $C\ A\ C\ B$ →
∃ X , is_midpoint $X\ A\ B$.

Lemma midpoint_distinct_1 : ∀ $I\ A\ B$,
$A \neq B$ →
is_midpoint $I\ A\ B$ →
$I \neq A \wedge I \neq B$.

Lemma midpoint_distinct_2 : ∀ $I\ A\ B$,
$I \neq A$ →
is_midpoint $I\ A\ B$ →
$A \neq B \wedge I \neq B$.

Lemma midpoint_distinct_3 : ∀ $I\ A\ B$,
$I \neq B$ →
is_midpoint $I\ A\ B$ →
$A \neq B \wedge I \neq A$.

Lemma midpoint_def : ∀ $A\ B\ C$, Bet $A\ B\ C$ → Cong $A\ B\ B\ C$ → is_midpoint $B\ A\ C$.

Lemma midpoint_bet : ∀ $A\ B\ C$, is_midpoint $B\ A\ C$ → Bet $A\ B\ C$.

Lemma midpoint_col : ∀ $A\ M\ B$, is_midpoint $M\ A\ B$ → Col $M\ A\ B$.

Lemma midpoint_cong : ∀ $A\ B\ C$, is_midpoint $B\ A\ C$ → Cong $A\ B\ B\ C$.

Lemma midpoint_not_midpoint : ∀ $I\ A\ B$,
$A \neq B$ →
is_midpoint $I\ A\ B$ →
¬ is_midpoint $B\ A\ I$.

Lemma swap_diff : ∀ ($A\ B$: Tpoint), $A \neq B$ → $B \neq A$.

Lemma cong_cong_half_1 : ∀ $A\ M\ B\ A'\ M'\ B'$,
is_midpoint $M\ A\ B$ → is_midpoint $M'\ A'\ B'$ →
Cong $A\ B\ A'\ B'$ → Cong $A\ M\ A'\ M'$.

Lemma cong_cong_half_2 : ∀ A M B A' M' B',
is_midpoint M A B $\rightarrow$ is_midpoint M' A' B' $\rightarrow$
Cong A B A' B' $\rightarrow$ Cong B M B' M'.

End T7_2.

Chapter 10

Library Ch08_orthogonality

```
Require Export Ch07_midpoint.
Require Export ColR.

Ltac not_exist_hyp_comm A B := not_exist_hyp (A≠B);not_exist_hyp (B≠A).

Ltac not_exist_hyp2 A B C D := first [not_exist_hyp_comm A B | not_exist_hyp_comm
C D].

Ltac not_exist_hyp3 A B C D E F := first [not_exist_hyp_comm A B | not_exist_hyp_comm
C D | not_exist_hyp_comm E F].

Ltac not_exist_hyp_perm_col A B C := not_exist_hyp (¬ Col A B C); not_exist_hyp (¬ Col
A C B);
                                     not_exist_hyp (¬ Col B A C); not_exist_hyp (¬ Col B
C A);
                                     not_exist_hyp (¬ Col C A B); not_exist_hyp (¬ Col C
B A).

Ltac finish := match goal with
 | ⊢ Col ?A ?B ?C ⇒ Col
 | ⊢ ¬ Col ?A ?B ?C ⇒ Col
 | ⊢ Cong ?A ?B ?C ?D ⇒ Cong
 | ⊢ is_midpoint ?A ?B ?C ⇒ Midpoint
 | ⊢ ?A<>?B ⇒ apply swap_diff;assumption
 | ⊢ _ ⇒ try assumption
end.

Ltac assert_all_diffs_by_cases :=
 repeat match goal with
 | A: Tpoint, B: Tpoint ⊢ _ ⇒ not_exist_hyp_comm A B;induction (eq_dec_points A
B);[treat_equalities;solve [finish] |idtac]
end.

Ltac assert_cols :=
repeat
```

```
 match goal with
       | H:Bet ?X1 ?X2 ?X3 ⊢ _ ⇒
      not_exist_hyp (Col X1 X2 X3);assert (Col X1 X2 X3) by (apply bet_col;apply H)

       | H:is_midpoint ?X1 ?X2 ?X3 ⊢ _ ⇒
      not_exist_hyp (Col X1 X2 X3);let N := fresh in assert (N := midpoint_col X2 X1
X3 H)

       | H:out ?X1 ?X2 ?X3 ⊢ _ ⇒
      not_exist_hyp (Col X1 X2 X3);let N := fresh in assert (N := out_col X1 X2 X3
H)
 end.

Ltac assert_bet :=
repeat
 match goal with
       | H:is_midpoint ?B ?A ?C ⊢ _ ⇒ let T := fresh in not_exist_hyp (Bet A B C);
assert (T := midpoint_bet A B C H)
 end.

Ltac clean_reap_hyps :=
  repeat
  match goal with
   | H:(is_midpoint ?A ?B ?C), H2 : is_midpoint ?A ?C ?B ⊢ _ ⇒ clear H2
   | H:(is_midpoint ?A ?B ?C), H2 : is_midpoint ?A ?B ?C ⊢ _ ⇒ clear H2
   | H:(¬Col ?A ?B ?C), H2 : ¬Col ?A ?B ?C ⊢ _ ⇒ clear H2
   | H:(Col ?A ?B ?C), H2 : Col ?A ?C ?B ⊢ _ ⇒ clear H2
   | H:(Col ?A ?B ?C), H2 : Col ?A ?B ?C ⊢ _ ⇒ clear H2
   | H:(Col ?A ?B ?C), H2 : Col ?B ?A ?C ⊢ _ ⇒ clear H2
   | H:(Col ?A ?B ?C), H2 : Col ?B ?C ?A ⊢ _ ⇒ clear H2
   | H:(Col ?A ?B ?C), H2 : Col ?C ?B ?A ⊢ _ ⇒ clear H2
   | H:(Col ?A ?B ?C), H2 : Col ?C ?A ?B ⊢ _ ⇒ clear H2
   | H:(Bet ?A ?B ?C), H2 : Bet ?C ?B ?A ⊢ _ ⇒ clear H2
   | H:(Bet ?A ?B ?C), H2 : Bet ?A ?B ?C ⊢ _ ⇒ clear H2
   | H:(Cong ?A ?B ?C ?D), H2 : Cong ?A ?B ?D ?C ⊢ _ ⇒ clear H2
   | H:(Cong ?A ?B ?C ?D), H2 : Cong ?A ?B ?C ?D ⊢ _ ⇒ clear H2
   | H:(Cong ?A ?B ?C ?D), H2 : Cong ?C ?D ?A ?B ⊢ _ ⇒ clear H2
   | H:(Cong ?A ?B ?C ?D), H2 : Cong ?C ?D ?B ?A ⊢ _ ⇒ clear H2
   | H:(Cong ?A ?B ?C ?D), H2 : Cong ?D ?C ?B ?A ⊢ _ ⇒ clear H2
   | H:(Cong ?A ?B ?C ?D), H2 : Cong ?D ?C ?A ?B ⊢ _ ⇒ clear H2
   | H:(Cong ?A ?B ?C ?D), H2 : Cong ?B ?A ?C ?D ⊢ _ ⇒ clear H2
   | H:(Cong ?A ?B ?C ?D), H2 : Cong ?B ?A ?D ?C ⊢ _ ⇒ clear H2
   | H:(?A<>?B), H2 : (?B<>?A) ⊢ _ ⇒ clear H2
   | H:(?A<>?B), H2 : (?A<>?B) ⊢ _ ⇒ clear H2
```

```
end.

Ltac assert_diffs :=
repeat
 match goal with
      | H:(¬Col ?X1 ?X2 ?X3) ⊢ _ ⇒
      let h := fresh in
      not_exist_hyp3 X1 X2 X1 X3 X2 X3;
      assert (h := not_col_distincts X1 X2 X3 H);decompose [and] h;clear h;clean_reap_hyps


      | H:Cong ?A ?B ?C ?D, H2 : ?A ≠ ?B ⊢_ ⇒
      let T:= fresh in (not_exist_hyp_comm C D);
        assert (T:= cong_diff A B C D H2 H);clean_reap_hyps
      | H:Cong ?A ?B ?C ?D, H2 : ?B ≠ ?A ⊢_ ⇒
      let T:= fresh in (not_exist_hyp_comm C D);
        assert (T:= cong_diff_2 A B C D H2 H);clean_reap_hyps
      | H:Cong ?A ?B ?C ?D, H2 : ?C ≠ ?D ⊢_ ⇒
      let T:= fresh in (not_exist_hyp_comm A B);
        assert (T:= cong_diff_3 A B C D H2 H);clean_reap_hyps
      | H:Cong ?A ?B ?C ?D, H2 : ?D ≠ ?C ⊢_ ⇒
      let T:= fresh in (not_exist_hyp_comm A B);
        assert (T:= cong_diff_4 A B C D H2 H);clean_reap_hyps

      | H:is_midpoint ?I ?A ?B, H2 : ?A<>?B ⊢ _ ⇒
      let T:= fresh in (not_exist_hyp2 I B I A);
       assert (T:= midpoint_distinct_1 I A B H2 H);
       decompose [and] T;clear T;clean_reap_hyps
      | H:is_midpoint ?I ?A ?B, H2 : ?B<>?A ⊢ _ ⇒
      let T:= fresh in (not_exist_hyp2 I B I A);
       assert (T:= midpoint_distinct_1 I A B (swap_diff B A H2) H);
       decompose [and] T;clear T;clean_reap_hyps

      | H:is_midpoint ?I ?A ?B, H2 : ?I<>?A ⊢ _ ⇒
      let T:= fresh in (not_exist_hyp2 I B A B);
       assert (T:= midpoint_distinct_2 I A B H2 H);
       decompose [and] T;clear T;clean_reap_hyps
      | H:is_midpoint ?I ?A ?B, H2 : ?A<>?I ⊢ _ ⇒
      let T:= fresh in (not_exist_hyp2 I B A B);
       assert (T:= midpoint_distinct_2 I A B (swap_diff A I H2) H);
       decompose [and] T;clear T;clean_reap_hyps

      | H:is_midpoint ?I ?A ?B, H2 : ?I<>?B ⊢ _ ⇒
```

```
      let T:= fresh in (not_exist_hyp2 I A A B);
       assert (T:= midpoint_distinct_3 I A B H2 H);
       decompose [and] T;clear T;clean_reap_hyps
      | H:is_midpoint ?I ?A ?B, H2 : ?B<>?I ⊢ _ ⇒
      let T:= fresh in (not_exist_hyp2 I A A B);
       assert (T:= midpoint_distinct_3 I A B (swap_diff B I H2) H);
       decompose [and] T;clear T;clean_reap_hyps

      | H:out ?A ?B ?C ⊢ _ ⇒
      let T:= fresh in (not_exist_hyp2 A B A C);
       assert (T:= out_distinct A B C H);
       decompose [and] T;clear T;clean_reap_hyps
 end.

Ltac clean_trivial_hyps :=
  repeat
  match goal with
   | H:(Cong ?X1 ?X1 ?X2 ?X2) ⊢ _ ⇒ clear H
   | H:(Cong ?X1 ?X2 ?X2 ?X1) ⊢ _ ⇒ clear H
   | H:(Cong ?X1 ?X2 ?X1 ?X2) ⊢ _ ⇒ clear H
   | H:(Bet ?X1 ?X1 ?X2) ⊢ _ ⇒ clear H
   | H:(Bet ?X2 ?X1 ?X1) ⊢ _ ⇒ clear H
   | H:(Col ?X1 ?X1 ?X2) ⊢ _ ⇒ clear H
   | H:(Col ?X2 ?X1 ?X1) ⊢ _ ⇒ clear H
   | H:(Col ?X1 ?X2 ?X1) ⊢ _ ⇒ clear H
   | H:(is_midpoint ?X1 ?X1 ?X1) ⊢ _ ⇒ clear H
end.

Ltac treat_equalities :=
try treat_equalities_aux;
repeat
  match goal with
   | H:(Cong ?X3 ?X3 ?X1 ?X2) ⊢ _ ⇒
      apply cong_symmetry in H; apply cong_identity in H;smart_subst X2
   | H:(Cong ?X1 ?X2 ?X3 ?X3) ⊢ _ ⇒
      apply cong_identity in H;smart_subst X2
   | H:(Bet ?X1 ?X2 ?X1) ⊢ _ ⇒
      apply between_identity in H;smart_subst X2
   | H:(is_midpoint ?X ?Y ?Y) ⊢ _ ⇒ apply l7_3 in H; smart_subst Y
   | H : Bet ?A ?B ?C, H2 : Bet ?B ?A ?C ⊢ _ ⇒
     let T := fresh in not_exist_hyp (A=B); assert (T := between_egality A B C H
H2);smart_subst A
   | H : is_midpoint ?P ?A ?P1, H2 : is_midpoint ?P ?A ?P2 ⊢ _ ⇒
     let T := fresh in not_exist_hyp (P1=P2); assert (T := symmetric_point_unicity A
```

```
P P1 P2 H H2);smart_subst P1
    | H : is_midpoint ?A ?P ?X, H2 : is_midpoint ?A ?Q ?X ⊢ _ ⇒
      let T := fresh in not_exist_hyp (P=Q); assert (T := l7_9 P Q A X H H2);smart_subst
P
    | H : is_midpoint ?M ?A ?A ⊢ _ ⇒
      let T := fresh in not_exist_hyp (M=A); assert (T : l7_3 M A H);smart_subst M
    | H : is_midpoint ?A ?P ?P', H2 : is_midpoint ?B ?P ?P' ⊢ _ ⇒
      let T := fresh in not_exist_hyp (A=B); assert (T := l7_17 P P' A B H H2);smart_subst
A
    | H : is_midpoint ?A ?P ?P', H2 : is_midpoint ?B ?P' ?P ⊢ _ ⇒
      let T := fresh in not_exist_hyp (A=B); assert (T := l7_17_bis P P' A B H
H2);smart_subst A
    | H : is_midpoint ?A ?B ?A ⊢ _ ⇒
      let T := fresh in not_exist_hyp (A=B); assert (T := is_midpoint_id_2 A B H);smart_subst
A
    | H : is_midpoint ?A ?A ?B ⊢ _ ⇒
      let T := fresh in not_exist_hyp (A=B); assert (T := is_midpoint_id A B H);smart_subst
A
end.

Ltac ColR :=
 let tpoint := constr:(Tpoint) in
 let col := constr:(Col) in
   treat_equalities; Col_refl tpoint col.

Ltac search_contradiction :=
 match goal with
  | H: ?A ≠ ?A ⊢ _ ⇒ exfalso;apply H;reflexivity
  | H: ¬ Col ?A ?B ?C ⊢ _ ⇒ exfalso;apply H;ColR
 end.

Ltac show_distinct' X Y :=
 assert (X≠Y);
 [intro;treat_equalities; (solve [search_contradiction])|idtac].

Ltac assert_all_diffs_by_contradiction :=
repeat match goal with
 | A: Tpoint, B: Tpoint ⊢ _ ⇒ not_exist_hyp_comm A B;show_distinct' A B
end.

Ltac update_cols :=
 let tpoint := constr:(Tpoint) in
 let col := constr:(Col) in
   update_cols_gen tpoint col.

Ltac deduce_cols :=
 let tpoint := constr:(Tpoint) in
```

```
 let col := constr:(Col) in
   treat_equalities; deduce_cols_hide_gen tpoint col.

Ltac cols :=
 let tpoint := constr:(Tpoint) in
 let col := constr:(Col) in
   cols_gen tpoint col.

Ltac tag_hyps :=
 let tpoint := constr:(Tpoint) in
 let col := constr:(Col) in
   tag_hyps_gen tpoint col.

Ltac untag_hyps :=
 let tpoint := constr:(Tpoint) in
 let col := constr:(Col) in
   untag_hyps_gen tpoint col.

Ltac clear_cols :=
 let tpoint := constr:(Tpoint) in
 let col := constr:(Col) in
   clear_cols_gen tpoint col.

Ltac smart_subst' := update_cols;clean.

Ltac treat_equalities_aux' :=
  match goal with
   | H:(?X1 = ?X2) ⊢ _ ⇒ smart_subst'
end.

Ltac treat_equalities' :=
try treat_equalities_aux';
repeat
  match goal with
   | H:(Cong ?X3 ?X3 ?X1 ?X2) ⊢ _ ⇒
      apply cong_symmetry in H; apply cong_identity in H; smart_subst'
   | H:(Cong ?X1 ?X2 ?X3 ?X3) ⊢ _ ⇒
      apply cong_identity in H; smart_subst'
   | H:(Bet ?X1 ?X2 ?X1) ⊢ _ ⇒
      apply between_identity in H; smart_subst'
   | H:(is_midpoint ?X ?Y ?Y) ⊢ _ ⇒ apply l7_3 in H; smart_subst'
   | H : Bet ?A ?B ?C, H2 : Bet ?B ?A ?C ⊢ _ ⇒
     let T := fresh in not_exist_hyp (A=B); assert (T : between_egality A B C H H2);
smart_subst'
   | H : is_midpoint ?P ?A ?P1, H2 : is_midpoint ?P ?A ?P2 ⊢ _ ⇒
     let T := fresh in not_exist_hyp (P1=P2); assert (T : symmetric_point_unicity A P
P1 P2 H H2); smart_subst'
   | H : is_midpoint ?A ?P ?X, H2 : is_midpoint ?A ?Q ?X ⊢ _ ⇒
```

```
    let T := fresh in not_exist_hyp (P=Q); assert (T : l7_9 P Q A X H H2);
smart_subst'
  | H : is_midpoint ?M ?A ?A ⊢ _ ⇒
    let T := fresh in not_exist_hyp (M=A); assert (T : l7_3 M A H); smart_subst'
  | H : is_midpoint ?A ?P ?P', H2 : is_midpoint ?B ?P ?P' ⊢ _ ⇒
    let T := fresh in not_exist_hyp (A=B); assert (T := l7_17 P P' A B H H2);
smart_subst'
  | H : is_midpoint ?A ?B ?A ⊢ _ ⇒
    let T := fresh in not_exist_hyp (A=B); assert (T := is_midpoint_id_2 A B H);
smart_subst'
  | H : is_midpoint ?A ?A ?B ⊢ _ ⇒
    let T := fresh in not_exist_hyp (A=B); assert (T := is_midpoint_id A B H);
smart_subst'
end.

Ltac search_contradiction' :=
 match goal with
  | H: ?A ≠ ?A ⊢ _ ⇒ exfalso;apply H;reflexivity
  | H: ¬ Col ?A ?B ?C ⊢ _ ⇒ exfalso;apply H;cols
 end.

Ltac show_distinct'' X Y :=
 assert (X≠Y);
 [intro; treat_equalities'; (solve [search_contradiction'])|idtac].

Ltac show_not_col X Y Z :=
 assert (¬ Col X Y Z);
 [intro; update_cols; (solve [search_contradiction'])|idtac].

Ltac assert_all_diffs_by_contradiction_aux :=
repeat match goal with
 | A: Tpoint, B: Tpoint ⊢ _ ⇒ untag_hyps; not_exist_hyp_comm A B; tag_hyps; show_distinct''
A B
end.

Ltac assert_all_not_cols_by_contradiction_aux :=
repeat match goal with
 | A: Tpoint, B: Tpoint, C: Tpoint ⊢ _ ⇒ untag_hyps; not_exist_hyp_perm_col A B C;
tag_hyps; show_not_col A B C
end.

Ltac assert_all_diffs_by_contradiction' :=
  deduce_cols; assert_all_diffs_by_contradiction_aux; untag_hyps; clear_cols.

Ltac assert_all_not_cols_by_contradiction :=
  deduce_cols; assert_all_not_cols_by_contradiction_aux; untag_hyps; clear_cols.

Ltac assert_ndc_by_contradiction :=
```

```
  assert_all_diffs_by_contradiction'; assert_all_not_cols_by_contradiction.
Section T8_1.
Context `{MT:Tarski_neutral_dimensionless}.
Context `{EqDec:EqDecidability Tpoint}.
Definition Per := fun A B C => ∃ C', is_midpoint B C C' ∧ Cong A C A C'.
Lemma Per_dec : ∀ A B C, Per A B C ∨ ¬ Per A B C.
Lemma l8_2 : ∀ A B C, Per A B C → Per C B A.
End T8_1.
Hint Resolve l8_2 : perp.
Ltac Perp := auto with perp.
Section T8_2.
Context `{MT:Tarski_neutral_dimensionless}.
Context `{EqDec:EqDecidability Tpoint}.
Lemma Per_cases :
 ∀ A B C,
 Per A B C ∨ Per C B A →
 Per A B C.
Lemma Per_perm :
 ∀ A B C,
 Per A B C →
 Per A B C ∧ Per C B A.
Lemma l8_3 : ∀ A B C A',
 Per A B C → A≠B → Col B A A' → Per A' B C.
Lemma l8_4 : ∀ A B C C', Per A B C → is_midpoint B C C' → Per A B C'.
Lemma l8_5 : ∀ A B, Per A B B.
Lemma l8_6 : ∀ A B C A', Per A B C → Per A' B C → Bet A C A' → B=C.
End T8_2.
Hint Resolve l8_5 : perp.
Ltac let_symmetric C P A :=
let id1:=fresh in (assert (id1:(∃ A', is_midpoint P A A'));
[apply symmetric_point_construction|ex_and id1 C]).
Ltac symmetric B' A B :=
assert(sp:= symmetric_point_construction B A); ex_and sp B'.
Section T8_3.
Context `{MT:Tarski_neutral_dimensionless}.
Context `{EqDec:EqDecidability Tpoint}.
```

Lemma l8_7 : $\forall$ A B C, Per A B C $\rightarrow$ Per A C B $\rightarrow$ B=C.

Lemma l8_8 : $\forall$ A B, Per A B A $\rightarrow$ A=B.

Lemma l8_9 : $\forall$ A B C, Per A B C $\rightarrow$ Col A B C $\rightarrow$ A=B $\vee$ C=B.

Lemma l8_10 : $\forall$ A B C A' B' C', Per A B C $\rightarrow$ Cong_3 A B C A' B' C' $\rightarrow$ Per A' B' C'.

Definition Perp_in := fun X A B C D $\Rightarrow$
$A \neq B$ $\wedge$ $C \neq D$ $\wedge$ Col X A B $\wedge$ Col X C D $\wedge$ ($\forall$ U V, Col U A B $\rightarrow$ Col V C D $\rightarrow$ Per U X V).

Lemma col_col_per_per : $\forall$ A X C U V,
$A \neq X$ $\rightarrow$ $C \neq X$ $\rightarrow$
Col U A X $\rightarrow$
Col V C X $\rightarrow$
Per A X C $\rightarrow$
Per U X V.

Lemma Perp_in_dec : $\forall$ X A B C D, Perp_in X A B C D $\vee$ $\neg$ Perp_in X A B C D.

Definition Perp := fun A B C D $\Rightarrow$ $\exists$ X , Perp_in X A B C D.

Lemma perp_distinct : $\forall$ A B C D, Perp A B C D $\rightarrow$ $A \neq B$ $\wedge$ $C \neq D$.

Lemma l8_12 : $\forall$ A B C D X, Perp_in X A B C D $\rightarrow$ Perp_in X C D A B.

Lemma l4_3_1 : $\forall$ A B C A' B' C',
Bet A B C $\rightarrow$ Bet A' B' C' $\rightarrow$ Cong A B A' B' $\rightarrow$ Cong A C A' C' $\rightarrow$ Cong B C B' C'.

Lemma per_col : $\forall$ A B C D,
$B \neq C$ $\rightarrow$ Per A B C $\rightarrow$ Col B C D $\rightarrow$ Per A B D.

Lemma l8_13_2 : $\forall$ A B C D X,
$A \neq B$ $\rightarrow$ $C \neq D$ $\rightarrow$ Col X A B $\rightarrow$ Col X C D $\rightarrow$
($\exists$ U , $\exists$ V :Tpoint, Col U A B $\wedge$ Col V C D $\wedge$ $U \neq X$ $\wedge$ $V \neq X$ $\wedge$ Per U X V) $\rightarrow$
Perp_in X A B C D.

Definition DistLn := fun A B C D $\Rightarrow$
($\exists$ X , Col X A B $\wedge$ $\neg$Col X C D) $\vee$ ($\exists$ X , $\neg$ Col X A B $\wedge$ Col X C D).

Lemma perBAB : $\forall$ A B, Per B A B $\rightarrow$ A = B.

Lemma l8_14_1 : $\forall$ A B, $\neg$ Perp A B A B.

Lemma l8_14_2_1a : $\forall$ X A B C D, Perp_in X A B C D $\rightarrow$ Perp A B C D.

Lemma perp_in_distinct : $\forall$ X A B C D , Perp_in X A B C D $\rightarrow$ $A \neq B$ $\wedge$ $C \neq D$.

Lemma l8_14_2_1b : $\forall$ X A B C D Y, Perp_in X A B C D $\rightarrow$ Col Y A B $\rightarrow$ Col Y C D $\rightarrow$ X=Y.

Lemma l8_14_2_1b_bis : $\forall$ A B C D X, Perp A B C D $\rightarrow$ Col X A B $\rightarrow$ Col X C D $\rightarrow$ Perp_in X A B C D.

Lemma l8_14_2_2 : $\forall$ X A B C D,
Perp A B C D $\rightarrow$ ($\forall$ Y, Col Y A B $\rightarrow$ Col Y C D $\rightarrow$ X=Y) $\rightarrow$ Perp_in X A B C D.

Lemma l8_14_3 : $\forall$ A B C D X Y, Perp_in X A B C D $\rightarrow$ Perp_in Y A B C D $\rightarrow$ $X=Y$.

Lemma l8_15_1 : $\forall$ A B C X, $A\neq B$ $\rightarrow$ Col A B X $\rightarrow$ Perp A B C X $\rightarrow$ Perp_in X A B C X.

Lemma l8_15_2 : $\forall$ A B C X, $A\neq B$ $\rightarrow$ Col A B X $\rightarrow$ Perp_in X A B C X $\rightarrow$ Perp A B C X.

Lemma perp_in_per : $\forall$ A B C, Perp_in B A B B $C\rightarrow$ Per A B C.

Lemma perp_sym : $\forall$ A B C D, Perp A B C D $\rightarrow$ Perp C D A B.

Lemma perp_col0 : $\forall$ A B C D X Y, Perp A B C D $\rightarrow$ $X \neq Y$ $\rightarrow$ Col A B X $\rightarrow$ Col A B Y $\rightarrow$ Perp C D X Y.

Lemma l8_16_1 : $\forall$ A B C U X,
$A\neq B$ $\rightarrow$ Col A B X $\rightarrow$ Col A B U $\rightarrow$ $U\neq X$ $\rightarrow$ Perp A B C X $\rightarrow$ $\neg$ Col A B C $\wedge$ Per C X U.

Lemma per_perp_in : $\forall$ A B C, $A \neq B$ $\rightarrow$ $B \neq C$ $\rightarrow$ Per A B C $\rightarrow$ Perp_in B A B B C.

Lemma per_perp : $\forall$ A B C, $A \neq B$ $\rightarrow$ $B \neq C$ $\rightarrow$ Per A B C $\rightarrow$ Perp A B B C.

Lemma perp_left_comm : $\forall$ A B C D, Perp A B C D $\rightarrow$ Perp B A C D.

Lemma perp_right_comm : $\forall$ A B C D, Perp A B C D $\rightarrow$ Perp A B D C.

Lemma perp_comm : $\forall$ A B C D, Perp A B C D $\rightarrow$ Perp B A D C.

Lemma perp_in_sym :
$\forall$ A B C D X,
Perp_in X A B C D $\rightarrow$ Perp_in X C D A B.

Lemma perp_in_left_comm :
$\forall$ A B C D X,
Perp_in X A B C D $\rightarrow$ Perp_in X B A C D.

Lemma perp_in_right_comm : $\forall$ A B C D X, Perp_in X A B C D $\rightarrow$ Perp_in X A B D C.

Lemma perp_in_comm : $\forall$ A B C D X, Perp_in X A B C D $\rightarrow$ Perp_in X B A D C.

End T8_3.

Hint Resolve perp_sym perp_left_comm perp_right_comm perp_comm per_perp_in
perp_in_per perp_in_left_comm perp_in_right_comm perp_in_comm perp_in_sym
: *perp*.

Ltac *double* A B A' :=
assert (*mp*:= symmetric_point_construction A B);
elim *mp*; intros A' ; intro; clear *mp*.

Section T8_4.

Context ‘{MT:**Tarski_neutral_dimensionless**}.
Context ‘{$EqDec$:**EqDecidability** Tpoint}.

Lemma Perp_cases :
$\forall$ A B C D,
Perp A B C D $\vee$ Perp B A C D $\vee$ Perp A B D C $\vee$ Perp B A D C $\vee$

Perp $C\ D\ A\ B$ ∨ Perp $C\ D\ B\ A$ ∨ Perp $D\ C\ A\ B$ ∨ Perp $D\ C\ B\ A$ →
Perp $A\ B\ C\ D$.

Lemma Perp_perm :
∀ $A\ B\ C\ D$,
Perp $A\ B\ C\ D$ →
Perp $A\ B\ C\ D$ ∧ Perp $B\ A\ C\ D$ ∧ Perp $A\ B\ D\ C$ ∧ Perp $B\ A\ D\ C$ ∧
Perp $C\ D\ A\ B$ ∧ Perp $C\ D\ B\ A$ ∧ Perp $D\ C\ A\ B$ ∧ Perp $D\ C\ B\ A$.

Lemma Perp_in_cases :
∀ $X\ A\ B\ C\ D$,
Perp_in $X\ A\ B\ C\ D$ ∨ Perp_in $X\ B\ A\ C\ D$ ∨ Perp_in $X\ A\ B\ D\ C$ ∨ Perp_in $X\ B\ A\ D\ C$ ∨
Perp_in $X\ C\ D\ A\ B$ ∨ Perp_in $X\ C\ D\ B\ A$ ∨ Perp_in $X\ D\ C\ A\ B$ ∨ Perp_in $X\ D\ C\ B\ A$ →
Perp_in $X\ A\ B\ C\ D$.

Lemma Perp_in_perm :
∀ $X\ A\ B\ C\ D$,
Perp_in $X\ A\ B\ C\ D$ →
Perp_in $X\ A\ B\ C\ D$ ∧ Perp_in $X\ B\ A\ C\ D$ ∧ Perp_in $X\ A\ B\ D\ C$ ∧ Perp_in $X\ B\ A\ D\ C$ ∧
Perp_in $X\ C\ D\ A\ B$ ∧ Perp_in $X\ C\ D\ B\ A$ ∧ Perp_in $X\ D\ C\ A\ B$ ∧ Perp_in $X\ D\ C\ B\ A$.

Lemma l8_16_2 : ∀ $A\ B\ C\ U\ X$,
$A \neq B$ → Col $A\ B\ X$ → Col $A\ B\ U$ → $U \neq X$ → ¬ Col $A\ B\ C$ → Per $C\ X\ U$ → Perp $A\ B\ C\ X$.

Lemma l8_18_unicity : ∀ $A\ B\ C\ X\ Y$,
¬ Col $A\ B\ C$ → Col $A\ B\ X$ → Perp $A\ B\ C\ X$ → Col $A\ B\ Y$ → Perp $A\ B\ C\ Y$ → $X = Y$.

Lemma distinct : ∀ $A\ B\ X\ C\ C'$, ¬ Col $A\ B\ C$ → Col $A\ B\ X$ → is_midpoint $X\ C\ C'$ → $C \neq C'$.

Lemma l8_20_1 : ∀ $A\ B\ C\ C'\ D\ P$,
Per $A\ B\ C$ → is_midpoint $P\ C'\ D$ → is_midpoint $A\ C'\ C$ → is_midpoint $B\ D\ C$ → Per $B\ A\ P$.

Lemma l8_20_2 : ∀ $A\ B\ C\ C'\ D\ P$,
Per $A\ B\ C$ → is_midpoint $P\ C'\ D$ → is_midpoint $A\ C'\ C$ → is_midpoint $B\ D\ C$ → $B \neq C$ → $A \neq P$.

Lemma perp_col1 : ∀ $A\ B\ C\ D\ X$,
$C \neq X$ → Perp $A\ B\ C\ D$ → Col $C\ D\ X$ → Perp $A\ B\ C\ X$.

Lemma l8_18_existence : ∀ $A\ B\ C$, ¬ Col $A\ B\ C$ → ∃ X, Col $A\ B\ X$ ∧ Perp $A\ B\ C\ X$.

Lemma l8_21_aux : ∀ $A\ B\ C$,
$A \neq B$ → ¬ Col $A\ B\ C$ → ∃ P, ∃ T, Perp $A\ B\ P\ A$ ∧ Col $A\ B\ T$ ∧ Bet $C\ T\ P$.

Lemma l8_21 : $\forall$ A B C,
$A\neq$ B $\rightarrow$ $\exists$ P , $\exists$ T , Perp A B P A $\wedge$ Col A B T $\wedge$ Bet C T P.

Lemma perp_in_col : $\forall$ A B C D X, Perp_in X A B C D $\rightarrow$ Col A B X $\wedge$ Col C D X.

Lemma perp_perp_in : $\forall$ A B C, Perp A B C A $\rightarrow$ Perp_in A A B C A.

Lemma perp_per_1 : $\forall$ A B C, $A\neq B$ $\rightarrow$ Perp A B C A $\rightarrow$ Per B A C.

Lemma perp_per_2 : $\forall$ A B C, $A\neq B$ $\rightarrow$ Perp A B A C $\rightarrow$ Per B A C.

Lemma perp_col : $\forall$ A B C D E, $A\neq E$ $\rightarrow$ Perp A B C D $\rightarrow$ Col A B E $\rightarrow$ Perp A E C D.

Lemma perp_col2 : $\forall$ A B C D X Y,
Perp A B X Y $\rightarrow$
$C\neq D$ $\rightarrow$ Col A B C $\rightarrow$ Col A B D $\rightarrow$ Perp C D X Y.

Lemma perp_not_eq_1 : $\forall$ A B C D, Perp A B C D $\rightarrow$ $A\neq B$.

Lemma perp_not_eq_2 : $\forall$ A B C D, Perp A B C D $\rightarrow$ $C\neq D$.

Lemma le_bet : $\forall$ A B C D, le C D A B $\rightarrow$ $\exists$ X , Bet A X B $\wedge$ Cong A X C D.

Lemma bet_cong3 : $\forall$ A B C A' B', Bet A B C $\rightarrow$ Cong A B A' B' $\rightarrow$ $\exists$ C' , Cong_3 A B C A' B' C'.

Lemma diff_per_diff : $\forall$ A B P R ,
$A\neq B$ $\rightarrow$ Cong A P B R $\rightarrow$ Per B A P $\rightarrow$ Per A B R $\rightarrow$ $P\neq R$.

Lemma per_not_colp : $\forall$ A B P R, $A\neq B$ $\rightarrow$ $A\neq P$ $\rightarrow$ $B\neq R$ $\rightarrow$ Per B A P $\rightarrow$ Per A B R $\rightarrow$ $\neg$Col P A R.

Lemma per_not_col : $\forall$ A B C, $A\neq B$ $\rightarrow$ $B\neq C$ $\rightarrow$ Per A B C $\rightarrow$ $\neg$Col A B C.

Lemma per_cong : $\forall$ A B P R X ,
$A\neq B$ $\rightarrow$ $A\neq P$ $\rightarrow$
Per B A P $\rightarrow$ Per A B R $\rightarrow$
Cong A P B R $\rightarrow$ Col A B X $\rightarrow$ Bet P X R $\rightarrow$
Cong A R P B.

Lemma perp_cong : $\forall$ A B P R X,
$A\neq B$ $\rightarrow$ $A\neq P$ $\rightarrow$
Perp A B P A $\rightarrow$ Perp A B R B $\rightarrow$
Cong A P B R $\rightarrow$ Col A B X $\rightarrow$ Bet P X R $\rightarrow$
Cong A R P B.

Lemma midpoint_existence_aux : $\forall$ A B P Q T,
$A\neq B$ $\rightarrow$ Perp A B Q B $\rightarrow$ Perp A B P A $\rightarrow$
Col A B T $\rightarrow$ Bet Q T P $\rightarrow$ le A P B Q $\rightarrow$
$\exists$ X : Tpoint , is_midpoint X A B.

Lemma midpoint_existence : $\forall$ A B, $\exists$ X , is_midpoint X A B.

Lemma perp_in_id : $\forall$ A B C X, Perp_in X A B C A $\rightarrow$ X = A.

Lemma l8_22 : $\forall$ A B P R X ,

$A \neq B \rightarrow A \neq P \rightarrow$
Per $B\ A\ P$ → Per $A\ B\ R$ →
Cong $A\ P\ B\ R$ → Col $A\ B\ X$ → Bet $P\ X\ R$ →
Cong $A\ R\ P\ B$ ∧ is_midpoint $X\ A\ B$ ∧ is_midpoint $X\ P\ R$.

Lemma l8_22_bis : ∀ $A\ B\ P\ R\ X$,
$A \neq B \rightarrow A \neq P \rightarrow$
Perp $A\ B\ P\ A$ → Perp $A\ B\ R\ B$ →
Cong $A\ P\ B\ R$ → Col $A\ B\ X$ → Bet $P\ X\ R$ →
Cong $A\ R\ P\ B$ ∧ is_midpoint $X\ A\ B$ ∧ is_midpoint $X\ P\ R$.

Lemma perp_in_perp : ∀ $A\ B\ C\ D\ X$, Perp_in $X\ A\ B\ C\ D$ → Perp $A\ B\ C\ D$.

End T8_4.

Hint Resolve perp_col perp_perp_in perp_in_perp : *perp*.

Section T8_5.

Context ‘{*MT*:**Tarski_neutral_dimensionless**}.
Context ‘{*EqDec*:**EqDecidability** Tpoint}.

Lemma perp_proj : ∀ $A\ B\ C\ D$, Perp $A\ B\ C\ D$ → ¬Col $A\ C\ D$ → ∃ X , Col $A\ B\ X$ ∧ Perp $A\ X\ C\ D$.

Lemma l8_24 : ∀ $A\ B\ P\ Q\ R\ T$,
Perp $P\ A\ A\ B$ →
Perp $Q\ B\ A\ B$ →
Col $A\ B\ T$ →
Bet $P\ T\ Q$ →
Bet $B\ R\ Q$ →
Cong $A\ P\ B\ R$ →
∃ X , is_midpoint $X\ A\ B$ ∧ is_midpoint $X\ P\ R$.

Lemma perp_not_col2 : ∀ $A\ B\ C\ D$, Perp $A\ B\ C\ D$ → ¬ Col $A\ B\ C$ ∨ ¬ Col $A\ B\ D$.

Lemma ex_col2 : ∀ $A\ B$, ∃ C , Col $A\ B\ C$ ∧ $A \neq C$ ∧ $B \neq C$.

Lemma perp_in_col_perp_in : ∀ $A\ B\ C\ D\ E\ P$, $C \neq E$ → Col $C\ D\ E$ → Perp_in $P\ A\ B\ C\ D$ → Perp_in $P\ A\ B\ C\ E$.

End T8_5.

Chapter 11

Library Ch09_plane

```
Require Export Ch08_orthogonality.

Ltac clean_reap_hyps :=
  repeat
  match goal with
   | H:(is_midpoint ?A ?B ?C), H2 : is_midpoint ?A ?C ?B ⊢ _ ⇒ clear H2
   | H:(is_midpoint ?A ?B ?C), H2 : is_midpoint ?A ?B ?C ⊢ _ ⇒ clear H2
   | H:(¬Col ?A ?B ?C), H2 : ¬Col ?A ?B ?C ⊢ _ ⇒ clear H2
   | H:(Col ?A ?B ?C), H2 : Col ?A ?C ?B ⊢ _ ⇒ clear H2
   | H:(Col ?A ?B ?C), H2 : Col ?A ?B ?C ⊢ _ ⇒ clear H2
   | H:(Col ?A ?B ?C), H2 : Col ?B ?A ?C ⊢ _ ⇒ clear H2
   | H:(Col ?A ?B ?C), H2 : Col ?B ?C ?A ⊢ _ ⇒ clear H2
   | H:(Col ?A ?B ?C), H2 : Col ?C ?B ?A ⊢ _ ⇒ clear H2
   | H:(Col ?A ?B ?C), H2 : Col ?C ?A ?B ⊢ _ ⇒ clear H2
   | H:(Bet ?A ?B ?C), H2 : Bet ?C ?B ?A ⊢ _ ⇒ clear H2
   | H:(Bet ?A ?B ?C), H2 : Bet ?A ?B ?C ⊢ _ ⇒ clear H2
   | H:(Cong ?A ?B ?C ?D), H2 : Cong ?A ?B ?D ?C ⊢ _ ⇒ clear H2
   | H:(Cong ?A ?B ?C ?D), H2 : Cong ?A ?B ?C ?D ⊢ _ ⇒ clear H2
   | H:(Cong ?A ?B ?C ?D), H2 : Cong ?C ?D ?A ?B ⊢ _ ⇒ clear H2
   | H:(Cong ?A ?B ?C ?D), H2 : Cong ?C ?D ?B ?A ⊢ _ ⇒ clear H2
   | H:(Cong ?A ?B ?C ?D), H2 : Cong ?D ?C ?B ?A ⊢ _ ⇒ clear H2
   | H:(Cong ?A ?B ?C ?D), H2 : Cong ?D ?C ?A ?B ⊢ _ ⇒ clear H2
   | H:(Cong ?A ?B ?C ?D), H2 : Cong ?B ?A ?C ?D ⊢ _ ⇒ clear H2
   | H:(Cong ?A ?B ?C ?D), H2 : Cong ?B ?A ?D ?C ⊢ _ ⇒ clear H2
   | H:(Perp ?A ?B ?C ?D), H2 : Perp ?A ?B ?D ?C ⊢ _ ⇒ clear H2
   | H:(Perp ?A ?B ?C ?D), H2 : Perp ?A ?B ?C ?D ⊢ _ ⇒ clear H2
   | H:(Perp ?A ?B ?C ?D), H2 : Perp ?C ?D ?A ?B ⊢ _ ⇒ clear H2
   | H:(Perp ?A ?B ?C ?D), H2 : Perp ?C ?D ?B ?A ⊢ _ ⇒ clear H2
   | H:(Perp ?A ?B ?C ?D), H2 : Perp ?D ?C ?B ?A ⊢ _ ⇒ clear H2
   | H:(Perp ?A ?B ?C ?D), H2 : Perp ?D ?C ?A ?B ⊢ _ ⇒ clear H2
```

```
    | H:(Perp ?A ?B ?C ?D), H2 : Perp ?B ?A ?C ?D ⊢ _ ⇒ clear H2
    | H:(Perp ?A ?B ?C ?D), H2 : Perp ?B ?A ?D ?C ⊢ _ ⇒ clear H2
    | H:(?A<>?B), H2 : (?B<>?A) ⊢ _ ⇒ clear H2
    | H:(?A<>?B), H2 : (?A<>?B) ⊢ _ ⇒ clear H2
    | H:(Per ?A ?D ?C), H2 : (Per ?C ?D ?A) ⊢ _ ⇒ clear H2
    | H:(Per ?A ?D ?C), H2 : (Per ?A ?D ?C) ⊢ _ ⇒ clear H2
    | H:(Perp_in ?X ?A ?B ?C ?D), H2 : Perp_in ?X ?A ?B ?D ?C ⊢ _ ⇒ clear H2
    | H:(Perp_in ?X ?A ?B ?C ?D), H2 : Perp_in ?X ?A ?B ?C ?D ⊢ _ ⇒ clear H2
    | H:(Perp_in ?X ?A ?B ?C ?D), H2 : Perp_in ?X ?C ?D ?A ?B ⊢ _ ⇒ clear H2
    | H:(Perp_in ?X ?A ?B ?C ?D), H2 : Perp_in ?X ?C ?D ?B ?A ⊢ _ ⇒ clear H2
    | H:(Perp_in ?X ?A ?B ?C ?D), H2 : Perp_in ?X ?D ?C ?B ?A ⊢ _ ⇒ clear H2
    | H:(Perp_in ?X ?A ?B ?C ?D), H2 : Perp_in ?X ?D ?C ?A ?B ⊢ _ ⇒ clear H2
    | H:(Perp_in ?X ?A ?B ?C ?D), H2 : Perp_in ?X ?B ?A ?C ?D ⊢ _ ⇒ clear H2
    | H:(Perp_in ?X ?A ?B ?C ?D), H2 : Perp_in ?X ?B ?A ?D ?C ⊢ _ ⇒ clear H2
end.

Ltac assert_diffs :=
repeat
 match goal with
      | H:(¬Col ?X1 ?X2 ?X3) ⊢ _ ⇒
      let h := fresh in
      not_exist_hyp3 X1 X2 X1 X3 X2 X3;
      assert (h := not_col_distincts X1 X2 X3 H);decompose [and] h;clear h;clean_reap_hyps


      | H:Cong ?A ?B ?C ?D, H2 : ?A ≠ ?B ⊢_ ⇒
      let T:= fresh in (not_exist_hyp_comm C D);
        assert (T:= cong_diff A B C D H2 H);clean_reap_hyps
      | H:Cong ?A ?B ?C ?D, H2 : ?B ≠ ?A ⊢_ ⇒
      let T:= fresh in (not_exist_hyp_comm C D);
        assert (T:= cong_diff_2 A B C D H2 H);clean_reap_hyps
      | H:Cong ?A ?B ?C ?D, H2 : ?C ≠ ?D ⊢_ ⇒
      let T:= fresh in (not_exist_hyp_comm A B);
        assert (T:= cong_diff_3 A B C D H2 H);clean_reap_hyps
      | H:Cong ?A ?B ?C ?D, H2 : ?D ≠ ?C ⊢_ ⇒
      let T:= fresh in (not_exist_hyp_comm A B);
        assert (T:= cong_diff_4 A B C D H2 H);clean_reap_hyps

      | H:is_midpoint ?I ?A ?B, H2 : ?A<>?B ⊢ _ ⇒
      let T:= fresh in (not_exist_hyp2 I B I A);
       assert (T:= midpoint_distinct_1 I A B H2 H);
       decompose [and] T;clear T;clean_reap_hyps
      | H:is_midpoint ?I ?A ?B, H2 : ?B<>?A ⊢ _ ⇒
```

```
      let T:= fresh in (not_exist_hyp2 I B I A);
       assert (T:= midpoint_distinct_1 I A B (swap_diff B A H2) H);
       decompose [and] T;clear T;clean_reap_hyps

      | H:is_midpoint ?I ?A ?B, H2 : ?I<>?A ⊢ _ ⇒
      let T:= fresh in (not_exist_hyp2 I B A B);
       assert (T:= midpoint_distinct_2 I A B H2 H);
       decompose [and] T;clear T;clean_reap_hyps
      | H:is_midpoint ?I ?A ?B, H2 : ?A<>?I ⊢ _ ⇒
      let T:= fresh in (not_exist_hyp2 I B A B);
       assert (T:= midpoint_distinct_2 I A B (swap_diff A I H2) H);
       decompose [and] T;clear T;clean_reap_hyps

      | H:is_midpoint ?I ?A ?B, H2 : ?I<>?B ⊢ _ ⇒
      let T:= fresh in (not_exist_hyp2 I A A B);
       assert (T:= midpoint_distinct_3 I A B H2 H);
       decompose [and] T;clear T;clean_reap_hyps
      | H:is_midpoint ?I ?A ?B, H2 : ?B<>?I ⊢ _ ⇒
      let T:= fresh in (not_exist_hyp2 I A A B);
       assert (T:= midpoint_distinct_3 I A B (swap_diff B I H2) H);
       decompose [and] T;clear T;clean_reap_hyps

      | H:Perp ?A ?B ?C ?D ⊢ _ ⇒
      let T:= fresh in (not_exist_hyp2 A B C D);
       assert (T:= perp_distinct A B C D H);
       decompose [and] T;clear T;clean_reap_hyps
      | H:Perp_in ?X ?A ?B ?C ?D ⊢ _ ⇒
      let T:= fresh in (not_exist_hyp2 A B C D);
       assert (T:= perp_in_distinct X A B C D H);
       decompose [and] T;clear T;clean_reap_hyps
      | H:out ?A ?B ?C ⊢ _ ⇒
      let T:= fresh in (not_exist_hyp2 A B A C);
       assert (T:= out_distinct A B C H);
       decompose [and] T;clear T;clean_reap_hyps
  end.

Ltac clean_trivial_hyps :=
  repeat
  match goal with
   | H:(Cong ?X1 ?X1 ?X2 ?X2) ⊢ _ ⇒ clear H
   | H:(Cong ?X1 ?X2 ?X2 ?X1) ⊢ _ ⇒ clear H
   | H:(Cong ?X1 ?X2 ?X1 ?X2) ⊢ _ ⇒ clear H
   | H:(Bet ?X1 ?X1 ?X2) ⊢ _ ⇒ clear H
```

```
  | H:(Bet ?X2 ?X1 ?X1) ⊢ _ ⇒ clear H
  | H:(Col ?X1 ?X1 ?X2) ⊢ _ ⇒ clear H
  | H:(Col ?X2 ?X1 ?X1) ⊢ _ ⇒ clear H
  | H:(Col ?X1 ?X2 ?X1) ⊢ _ ⇒ clear H
  | H:(Per ?X1 ?X2 ?X2) ⊢ _ ⇒ clear H
  | H:(Per ?X1 ?X1 ?X2) ⊢ _ ⇒ clear H
  | H:(is_midpoint ?X1 ?X1 ?X1) ⊢ _ ⇒ clear H
end.

Ltac finish := match goal with
 | ⊢ Col ?A ?B ?C ⇒ Col
 | ⊢ ¬ Col ?A ?B ?C ⇒ Col
 | ⊢ Perp ?A ?B ?C ?D ⇒ Perp
 | ⊢ Cong ?A ?B ?C ?D ⇒ Cong
 | ⊢ is_midpoint ?A ?B ?C ⇒ Midpoint
 | ⊢ ?A<>?B ⇒ apply swap_diff;assumption
 | ⊢ _ ⇒ try assumption
end.

Section T9.

Context `{MT:Tarski_neutral_dimensionless}.
Context `{EqDec:EqDecidability Tpoint}.

Definition two_sides := fun A B P Q ⇒
  A≠B ∧ ¬ Col P A B ∧ ¬ Col Q A B ∧ ∃ T, Col T A B ∧ Bet P T Q.

Lemma l9_2 : ∀ A B P Q, two_sides A B P Q → two_sides A B Q P.

Lemma invert_two_sides : ∀ A B P Q,
 two_sides A B P Q → two_sides B A P Q.

Lemma inter_unicity : ∀ A B X Y M N,
 Col A B M→ Col X Y M → Col A B N → Col X Y N →
 ¬ Col A X B → X ≠ Y → M = N.

Lemma colx : ∀ A B M N X, A ≠ B → N ≠ M → X ≠ M → Col A B M → Col A B N →
Col M N X → Col A B X.

Lemma l9_3 : ∀ P Q A C M R B,
 two_sides P Q A C → Col M P Q →
 is_midpoint M A C → Col R P Q →
 out R A B → two_sides P Q B C.

Definition is_symmetric (A A' C : Tpoint) := is_midpoint C A A'.

Lemma sym_sym : ∀ A C A', is_symmetric A A' C → is_symmetric A' A C.

Lemma distinct : ∀ P Q R : Tpoint, P ≠ Q → (R ≠ P ∨ R ≠ Q).

Lemma diff_col_ex : ∀ A B, ∃ C, A ≠ C ∧ B ≠ C ∧ Col A B C.

Lemma diff_bet_ex3 : ∀ A B C,
```

Bet A B C $\rightarrow$
$\exists$ D, $A \neq D \wedge B \neq D \wedge C \neq D \wedge$ Col A B D.

Lemma diff_col_ex3 : $\forall$ A B C,
Col A B C $\rightarrow$ $\exists$ D, $A \neq D \wedge B \neq D \wedge C \neq D \wedge$ Col A B D.

Lemma mid_preserves_col : $\forall$ A B C M A' B' C',
Col A B C $\rightarrow$
is_midpoint M A A' $\rightarrow$
is_midpoint M B B' $\rightarrow$
is_midpoint M C C' $\rightarrow$
Col A' B' C'.

Lemma per_mid_per : $\forall$ A B X Y M,
$A \neq B$ $\rightarrow$ Per X A B $\rightarrow$
is_midpoint M A B $\rightarrow$ is_midpoint M X Y $\rightarrow$
Cong A X B Y $\wedge$ Per Y B A.

Lemma perp_in_perp : $\forall$ A B C D X,
Perp_in X A B C D $\rightarrow$ Perp X B C D $\vee$ Perp A X C D.

Lemma sym_preserve_diff : $\forall$ A B M A' B',
$A \neq B$ $\rightarrow$ is_midpoint M A A' $\rightarrow$ is_midpoint M B B' $\rightarrow$ $A' \neq B'$.

Lemma perp_col2 : $\forall$ P Q R S A B,
Perp P Q A B $\rightarrow$ Col P Q R $\rightarrow$ Col P Q S $\rightarrow$ $R \neq S$ $\rightarrow$
Perp R S A B.

Lemma l9_4_1_aux : $\forall$ P Q A C R S M,
le S C R A $\rightarrow$
two_sides P Q A C $\rightarrow$ Col R P Q $\rightarrow$ Perp P Q A R $\rightarrow$ Col S P Q $\rightarrow$
Perp P Q C S $\rightarrow$ is_midpoint M R S $\rightarrow$
($\forall$ U C',is_midpoint M U C' $\rightarrow$ (out R U A $\leftrightarrow$ out S C C')).

Lemma per_col_eq : $\forall$ A B C, Per A B C $\rightarrow$ Col A B C $\rightarrow$ $B \neq C$ $\rightarrow$ $A = B$.

Lemma l9_4_1 : $\forall$ P Q A C R S M,
two_sides P Q A C $\rightarrow$ Col R P Q $\rightarrow$
Perp P Q A R $\rightarrow$ Col S P Q $\rightarrow$
Perp P Q C S $\rightarrow$ is_midpoint M R S $\rightarrow$
($\forall$ U C',is_midpoint M U C' $\rightarrow$ (out R U A $\leftrightarrow$ out S C C')).

Lemma mid_two_sides : $\forall$ A B M X Y,
$A \neq B$ $\rightarrow$ is_midpoint M A B $\rightarrow$ $\neg$ Col A B X $\rightarrow$ is_midpoint M X Y $\rightarrow$
two_sides A B X Y.

Lemma col_preserves_two_sides : $\forall$ A B C D X Y,
$A \neq B$ $\rightarrow$ $C \neq D$ $\rightarrow$ Col A B C $\rightarrow$ Col A B D $\rightarrow$
two_sides A B X Y $\rightarrow$
two_sides C D X Y.

Lemma out_out_two_sides : $\forall$ A B X Y U V I,
  $A \neq B \rightarrow$
  two_sides A B X Y $\rightarrow$
  Col I A B $\rightarrow$ Col I X Y $\rightarrow$
  out I X U $\rightarrow$ out I Y V $\rightarrow$
  two_sides A B U V.

Lemma l9_4_2_aux : $\forall$ P Q A C R S U V, le S C R A $\rightarrow$
two_sides P Q A C $\rightarrow$ Col R P Q $\rightarrow$ Perp P Q A R $\rightarrow$ Col S P Q $\rightarrow$
Perp P Q C S $\rightarrow$ out R U A $\rightarrow$ out S V C $\rightarrow$ two_sides P Q U V.

Lemma l9_4_2 : $\forall$ P Q A C R S U V,
two_sides P Q A C $\rightarrow$ Col R P Q $\rightarrow$ Perp P Q A R $\rightarrow$ Col S P Q $\rightarrow$
Perp P Q C S $\rightarrow$ out R U A $\rightarrow$ out S V C $\rightarrow$ two_sides P Q U V.

Lemma l9_5 : $\forall$ P Q A C R B,
 two_sides P Q A C $\rightarrow$ Col R P Q $\rightarrow$ out R A B $\rightarrow$ two_sides P Q B C.

This lemma used to be an axiom in previous versions of Tarski's axiom system. It is a been shown to a theorem by Gupta in his Phd 1965.

Lemma outer_pasch : $\forall$ A B C P Q,
 Bet A C P $\rightarrow$ Bet B Q C $\rightarrow$ $\exists$ X, Bet A X B $\wedge$ Bet P Q X.

Definition one_side := fun P Q A B $\Rightarrow$
 $\exists$ C, two_sides P Q A C $\wedge$ two_sides P Q B C.

Lemma invert_one_side : $\forall$ A B P Q,
 one_side A B P Q $\rightarrow$ one_side B A P Q.

Lemma l9_8_1 : $\forall$ P Q A B C, two_sides P Q A C $\rightarrow$ two_sides P Q B C $\rightarrow$ one_side P Q A B.

Lemma not_two_sides_id : $\forall$ A P Q, $\neg$ two_sides P Q A A.

Lemma l9_8_2 : $\forall$ P Q A B C,
 two_sides P Q A C $\rightarrow$
 one_side P Q A B $\rightarrow$
 two_sides P Q B C.

Lemma l9_9 : $\forall$ P Q A B, two_sides P Q A B $\rightarrow$ $\neg$ one_side P Q A B.

Lemma l9_9_bis : $\forall$ P Q A B, one_side P Q A B $\rightarrow$ $\neg$ two_sides P Q A B.

Lemma one_side_chara : $\forall$ P Q A B,
 $P \neq Q$ $\rightarrow$ $\neg$ Col A P Q $\rightarrow$ $\neg$ Col B P Q $\rightarrow$
 one_side P Q A B $\rightarrow$ ($\forall$ X, Col X P Q $\rightarrow$ $\neg$ Bet A X B).

Lemma l9_10 : $\forall$ P Q A,
 $P \neq Q$ $\rightarrow$ $\neg$ Col A P Q $\rightarrow$ $\exists$ C, two_sides P Q A C.

Lemma one_side_reflexivity : $\forall$ P Q A,
 $\neg$ Col A P Q $\rightarrow$ one_side P Q A A.

Lemma one_side_symmetry : $\forall\ P\ Q\ A\ B$,
one_side $P\ Q\ A\ B \rightarrow$ one_side $P\ Q\ B\ A$.

Lemma one_side_transitivity : $\forall\ P\ Q\ A\ B\ C$,
one_side $P\ Q\ A\ B \rightarrow$ one_side $P\ Q\ B\ C \rightarrow$ one_side $P\ Q\ A\ C$.

Lemma col_eq : $\forall\ A\ B\ X\ Y$,
$A \neq X \rightarrow$ Col $A\ X\ Y \rightarrow$ Col $B\ X\ Y \rightarrow$
$\neg$ Col $A\ X\ B \rightarrow$
$X = Y$.

Lemma l9_17 : $\forall\ A\ B\ C\ P\ Q$, one_side $P\ Q\ A\ C \rightarrow$ Bet $A\ B\ C \rightarrow$ one_side $P\ Q\ A\ B$.

Lemma l9_18 : $\forall\ X\ Y\ A\ B\ P$,
$X \neq Y \rightarrow$ Col $X\ Y\ P \rightarrow$ Col $A\ B\ P \rightarrow$ (two_sides $X\ Y\ A\ B \leftrightarrow$ (Bet $A\ P\ B \wedge \neg$Col $X\ Y\ A \wedge \neg$Col $X\ Y\ B$)).

Lemma l9_19 : $\forall\ X\ Y\ A\ B\ P$,
$X \neq Y \rightarrow$ Col $X\ Y\ P \rightarrow$ Col $A\ B\ P \rightarrow$ (one_side $X\ Y\ A\ B \leftrightarrow$ (out $P\ A\ B \wedge \neg$Col $X\ Y\ A$)).

End T9.

Chapter 12

Library Ch10_line_reflexivity

```
Require Export Ch09_plane.

Section T10.

Context `{MT:Tarski_2D}.
Context `{EqDec:EqDecidability Tpoint}.

Definition is_image_spec P' P A B :=
  (∃ X, is_midpoint X P P' ∧ Col A B X) ∧
  (Perp A B P P' ∨ P=P').

Definition is_image P' P A B :=
 (A≠B ∧ is_image_spec P' P A B) ∨ (A=B ∧ is_midpoint A P P').

Lemma ex_sym : ∀ A B X, ∃ Y, (Perp A B X Y ∨ X = Y) ∧
   (∃ M, Col A B M ∧ is_midpoint M X Y).

Lemma is_image_is_image_spec : ∀ P P' A B, A≠B → (is_image P' P A B ↔ is_image_spec
P' P A B).

Require Import Setoid.

Lemma ex_sym1 : ∀ A B X, A≠B → ∃ Y, (Perp A B X Y ∨ X = Y) ∧
 (∃ M, Col A B M ∧ is_midpoint M X Y ∧ is_image X Y A B).

Lemma l10_2_unicity : ∀ A B P P1 P2,
 is_image P1 P A B → is_image P2 P A B → P1=P2.

Lemma l10_2_existence_spec : ∀ A B P,
 ∃ P', is_image_spec P' P A B.

Lemma l10_2_existence : ∀ A B P,
 ∃ P', is_image P' P A B.

Lemma l10_4_spec : ∀ A B P P',
 is_image_spec P P' A B →
 is_image_spec P' P A B.

Lemma l10_4 : ∀ A B P P', is_image P P' A B → is_image P' P A B.
```

Lemma l10_5 : ∀ A B P P' P'',
 is_image P' P A B →
 is_image P'' P' A B → P=P''.

Lemma l10_6_unicity : ∀ A B P P1 P2, is_image P P1 A B → is_image P P2 A B → P1 = P2.

Lemma l10_6_existence_spec : ∀ A B P', A≠B → ∃ P , is_image_spec P' P A B.

Lemma l10_6_existence : ∀ A B P', ∃ P , is_image P' P A B.

Lemma l10_7 : ∀ A B P P' Q Q',
 is_image P' P A B → is_image Q' Q A B →
  P'=Q' → P = Q.

Lemma l10_8 : ∀ A B P, is_image P P A B → Col P A B.

Here we need the assumption that A<>B Lemma is_image_col_cong : ∀ A B P P' X, A≠B →
 is_image P P' A B → Col A B X → Cong P X P' X.

Lemma is_image_spec_col_cong : ∀ A B P P' X,
 is_image_spec P P' A B → Col A B X → Cong P X P' X.

Lemma perp_not_col : ∀ A B P, Perp A B P A → ¬ Col A B P.

Lemma image_id : ∀ A B T T',
  A≠B →
  Col A B T →
  is_image T T' A B →
  T = T'.

Lemma osym_not_col : ∀ A B P P',
 is_image P P' A B →
 ¬ Col A B P → ¬ Col A B P'.

We use l9_33 as a definition for coplanar.

Definition coplanar A B C D :=
  ∃ X , (Col A B X ∧ Col C D X) ∨ (Col A C X ∧ Col B D X) ∨ (Col A D X ∧ Col B C X).

Lemma coplanar_perm_1 : ∀ A B C D,
 coplanar A B C D → coplanar A B D C.

Lemma coplanar_perm_2 : ∀ A B C D,
 coplanar A B C D → coplanar A C B D.

Lemma coplanar_perm_3 : ∀ A B C D,
 coplanar A B C D → coplanar A C D B.

Lemma coplanar_perm_4 : ∀ A B C D,
 coplanar A B C D → coplanar A D B C.

Lemma coplanar_perm_5 : ∀ A B C D,

coplanar A B C $D \rightarrow$ coplanar A D C B.

Lemma coplanar_perm_6 : $\forall$ A B C D,
coplanar A B C $D \rightarrow$ coplanar B A C D.

Lemma coplanar_perm_7 : $\forall$ A B C D,
coplanar A B C $D \rightarrow$ coplanar B A D C.

Lemma coplanar_perm_8 : $\forall$ A B C D,
coplanar A B C $D \rightarrow$ coplanar B C A D.

Lemma coplanar_perm_9 : $\forall$ A B C D,
coplanar A B C $D \rightarrow$ coplanar B C D A.

Lemma coplanar_perm_10 : $\forall$ A B C D,
coplanar A B C $D \rightarrow$ coplanar B D A C.

Lemma coplanar_perm_11 : $\forall$ A B C D,
coplanar A B C $D \rightarrow$ coplanar B D C A.

Lemma per_per_col : $\forall$ A B C X, Per A X $C \rightarrow X \neq C \rightarrow$ Per B X $C \rightarrow$ Col A B X.

Lemma per_per_perp : $\forall$ A B X Y,
$A \neq B \rightarrow X \neq Y \rightarrow$
$(B \neq X \vee B \neq Y) \rightarrow$ Per A B $X \rightarrow$ Per A B $Y \rightarrow$
Perp A B X Y.

Lemma midpoint_preserves_image : $\forall$ A B P P' Q Q' M,
$A \neq B \rightarrow$ Col A B $M \rightarrow$ is_image P P' A $B \rightarrow$
is_midpoint M P $Q \rightarrow$ is_midpoint M P' $Q' \rightarrow$ is_image Q Q' A B.

Definition is_image_spec_in M P' P A B :=
(is_midpoint M P P' $\wedge$ Col A B M) $\wedge$ (Perp A B P P' $\vee$ P=P').

Definition is_image_spec_in_gen M P' P A B :=
($A \neq B$ $\wedge$ is_image_spec_in M P' P A B) $\vee$ (A=B $\wedge$ A=M $\wedge$ is_midpoint M P P').

Lemma image_in_is_image_spec :
$\forall$ M A B P P',
is_image_spec_in M P P' A $B \rightarrow$ is_image_spec P P' A B.

Lemma image_in_gen_is_image : $\forall$ M A B P P',
is_image_spec_in_gen M P P' A $B \rightarrow$ is_image P P' A B.

Lemma image_image_in : $\forall$ A B P P' M,
$P \neq P' \rightarrow$ is_image_spec P P' A $B \rightarrow$ Col A B $M \rightarrow$ Col P M $P' \rightarrow$
is_image_spec_in M P P' A B.

Lemma image_in_col : $\forall$ A B P P' Q Q' M,
is_image_spec_in M P P' A $B \rightarrow$ is_image_spec_in M Q Q' A $B \rightarrow$
Col M P Q.

Lemma image_in_col0 : $\forall$ A B P P' Y : Tpoint, is_image_spec_in Y P P' A B → Col P P' Y.

Lemma l10_10_spec : ∀ A B P Q P' Q',
$A \neq B$ → is_image_spec P' P A B → is_image_spec Q' Q A B →
Cong P Q P' Q'.

Lemma l10_10 : ∀ A B P Q P' Q',
is_image P' P A B → is_image Q' Q A B →
Cong P Q P' Q'.

Lemma is_image_spec_rev : ∀ P P' A B, is_image_spec P P' A B → is_image_spec P P' B A.

Lemma is_image_rev : ∀ P P' A B,
is_image P P' A B → is_image P P' B A.

Lemma per_double_cong : ∀ A B C C',
Per A B C → is_midpoint B C C' → Cong A C A C'.

Lemma midpoint_preserves_per : ∀ A B C $A1$ $B1$ $C1$ M,
Per A B C →
is_midpoint M A $A1$ →
is_midpoint M B $B1$ →
is_midpoint M C $C1$ →
Per $A1$ $B1$ $C1$.

Lemma image_preserves_bet : ∀ A B C A' B' C' X Y,
$X \neq Y$ →
is_image_spec A A' X Y → is_image_spec B B' X Y → is_image_spec C C' X Y →
Bet A B C →
Bet A' B' C'.

Lemma image_gen_preserves_bet : ∀ A B C A' B' C' X Y,
$X \neq Y$ →
is_image A A' X Y →
is_image B B' X Y →
is_image C C' X Y →
Bet A B C →
Bet A' B' C'.

Lemma image_preserves_midpoint :
∀ A B C A' B' C' X Y, $X \neq Y$ →
is_image_spec A A' X Y → is_image_spec B B' X Y → is_image_spec C C' X Y →
is_midpoint A B C →
is_midpoint A' B' C'.

Lemma image_preserves_per : ∀ A B C A' B' C' X Y,
$X \neq Y$ →
is_image_spec A A' X Y → is_image_spec B B' X Y → is_image_spec C C' X Y →
Per A B C →

Per $A' \ B' \ C'$.

Lemma col_per_perp : $\forall \ A \ B \ C \ D$,
$A \neq B \rightarrow B \neq C \rightarrow D \neq B \rightarrow D \neq C \rightarrow$
Col $B \ C \ D \rightarrow$ Per $A \ B \ C \rightarrow$ Perp $C \ D \ A \ B$.

Lemma image_col : $\forall \ A \ B \ X$, Col $A \ B \ X \rightarrow$ is_image_spec $X \ X \ A \ B$.

Lemma is_image_spec_triv : $\forall \ A \ B$, is_image_spec $A \ A \ B \ B$.

Lemma is_image_spec_dec :
$\forall \ A \ B \ C \ D$, is_image_spec $A \ B \ C \ D \lor \neg$ is_image_spec $A \ B \ C \ D$.

Lemma l10_12 : $\forall \ A \ B \ C \ A' \ B' \ C'$,
Per $A \ B \ C \rightarrow$ Per $A' \ B' \ C' \rightarrow$
Cong $A \ B \ A' \ B' \rightarrow$ Cong $B \ C \ B' \ C' \rightarrow$
Cong $A \ C \ A' \ C'$.

Lemma l10_14 : $\forall \ P \ P' \ A \ B$, $P \neq P' \rightarrow A \neq B \rightarrow$
is_image $P \ P' \ A \ B \rightarrow$ two_sides $A \ B \ P \ P'$.

Lemma l10_15 : $\forall \ A \ B \ C \ P$,
Col $A \ B \ C \rightarrow \neg$ Col $A \ B \ P \rightarrow$
$\exists \ Q$, Perp $A \ B \ Q \ C \land$ one_side $A \ B \ P \ Q$.

Lemma l10_16 : $\forall \ A \ B \ C \ A' \ B' \ P$,
$\neg$ Col $A \ B \ C \rightarrow \neg$ Col $A' \ B' \ P \rightarrow$ Cong $A \ B \ A' \ B' \rightarrow$
$\exists \ C'$, Cong_3 $A \ B \ C \ A' \ B' \ C' \land$ one_side $A' \ B' \ P \ C'$.

Lemma not_col_exists : $\forall \ A \ B$, $A \neq B \rightarrow \exists \ P$, $\neg$Col $A \ B \ P$.

Lemma perp_exists : $\forall \ O \ A \ B$, $A \neq B \rightarrow \exists \ X$, Perp $O \ X \ A \ B$.

End T10.

Chapter 13

Library Ch10_line_reflexivity_2D

```
Require Export Ch10_line_reflexivity.

Section T10.

Context `{MT:Tarski_2D}.
Context `{EqDec:EqDecidability Tpoint}.

Lemma image_cong_col : ∀ A B P P' X,
  P ≠ P' → is_image P P' A B → Cong P X P' X →
  Col A B X.

Lemma per_per_cong_1 :
  ∀ A B X Y, A ≠ B → Per A B X → Per A B Y →
  Cong B X B Y → X = Y ∨ is_midpoint B X Y.

Lemma per_per_cong : ∀ A B X Y,
  A ≠ B → Per A B X → Per A B Y → Cong B X B Y →
  X = Y ∨ is_image_spec X Y A B.

Lemma per_per_cong_gen : ∀ A B X Y,
  A ≠ B → Per A B X → Per A B Y → Cong B X B Y →
  X = Y ∨ is_image X Y A B.

End T10.
```

Chapter 14

Library Ch11_angles

Require Export Ch10_line_reflexivity.

Ltac *permut* :=
match goal with
  |H : (Col $?X$ $?Y$ $?Z$) ⊢ Col $?X$ $?Y$ $?Z$ ⇒ assumption
  |H : (Col $?X$ $?Y$ $?Z$) ⊢ Col $?Y$ $?Z$ $?X$ ⇒ apply col_permutation_1; assumption
  |H : (Col $?X$ $?Y$ $?Z$) ⊢ Col $?Z$ $?X$ $?Y$ ⇒ apply col_permutation_2; assumption
  |H : (Col $?X$ $?Y$ $?Z$) ⊢ Col $?X$ $?Z$ $?Y$ ⇒ apply col_permutation_5; assumption
  |H : (Col $?X$ $?Y$ $?Z$) ⊢ Col $?Y$ $?X$ $?Z$ ⇒ apply col_permutation_4; assumption
  |H : (Col $?X$ $?Y$ $?Z$) ⊢ Col $?Z$ $?Y$ $?X$ ⇒ apply col_permutation_3; assumption
  |_ : _ ⊢ _ ⇒ idtac
end.

Ltac *bet* :=
match goal with
  |$H0$: Bet $?A$ $?B$ $?C$ ⊢ Bet $?A$ $?B$ $?C$ ⇒ assumption
  |$H0$: Bet $?A$ $?B$ $?C$, $H1$: Bet $?B$ $?C$ $?D$, H : $?B \neq ?C$ ⊢ Bet $?A$ $?B$ $?D$ ⇒ apply (outer_transitivity_between _ _ _ _ $H0$ $H1$ H)
  |$H0$: Bet $?A$ $?B$ $?C$, $H1$: Bet $?B$ $?C$ $?D$, H : $?B \neq ?C$ ⊢ Bet $?A$ $?C$ $?D$ ⇒ apply (outer_transitivity_between2 _ _ _ _ $H0$ $H1$ H)
  |$H0$: Bet $?A$ $?B$ $?D$, $H1$: Bet $?B$ $?C$ $?D$ ⊢ Bet $?A$ $?B$ $?C$ ⇒ apply (between_inner_transitivity _ _ _ _ $H0$ $H1$)
  |$H0$: Bet $?A$ $?B$ $?C$, $H1$: Bet $?A$ $?C$ $?D$ ⊢ Bet $?B$ $?C$ $?D$ ⇒ apply (between_exchange3 _ _ _ _ $H0$ $H1$)
  |$H0$: Bet $?A$ $?B$ $?D$, $H1$: Bet $?B$ $?C$ $?D$ ⊢ Bet $?A$ $?C$ $?D$ ⇒ apply (between_exchange2 _ _ _ _ $H0$ $H1$)
  |$H0$: Bet $?A$ $?B$ $?C$, $H1$: Bet $?A$ $?C$ $?D$ ⊢ Bet $?A$ $?B$ $?D$ ⇒ apply (between_exchange4 _ _ _ _ $H0$ $H1$)

  |$H0$: Bet $?A$ $?B$ $?C$ ⊢ Bet $?A$ $?B$ $?C$ ⇒ assumption
  |$H0$: Bet $?A$ $?B$ $?C$, $H1$: Bet $?B$ $?C$ $?D$, H : $?B \neq ?C$ ⊢ Bet $?D$ $?B$ $?A$ ⇒ apply

between_symmetry; apply (outer_transitivity_between _ _ _ _ $H0$ $H1$ H)
|$H0$: Bet ?A ?B ?C, $H1$: Bet ?B ?C ?D , H : ?$B \neq$?C ⊢ Bet ?D ?C ?A ⇒ apply (outer_transitivity_between2 _ _ _ _ $H0$ $H1$ H)
|$H0$: Bet ?A ?B ?D, $H1$: Bet ?B ?C ?D ⊢ Bet ?C ?B ?A ⇒ apply between_symmetry; apply (between_inner_transitivity _ _ _ _ $H0$ $H1$)
|$H0$: Bet ?A ?B ?C, $H1$: Bet ?A ?C ?D ⊢ Bet ?D ?C ?B ⇒ apply between_symmetry; apply (between_exchange3 _ _ _ _ $H0$ $H1$)
|$H0$: Bet ?A ?B ?D, $H1$: Bet ?B ?C ?D ⊢ Bet ?D ?C ?A ⇒ apply between_symmetry; apply (between_exchange2 _ _ _ _ $H0$ $H1$)
|$H0$: Bet ?A ?B ?C, $H1$: Bet ?A ?C ?D ⊢ Bet ?D ?B ?A ⇒ apply between_symmetry; apply (between_exchange4 _ _ _ _ $H0$ $H1$)
|$H0$: Bet ?A ?B ?C ⊢ Bet ?C ?B ?A ⇒ apply (between_symmetry _ _ _ $H0$)

|$H0$: Bet ?A ?B ?C ⊢ Bet ?C ?B ?A ⇒ apply (between_symmetry _ _ _ $H0$)
|_ : _ ⊢ Bet ?A ?B ?B ⇒ apply between_trivial
|_ : _ ⊢ Bet ?A ?A ?B ⇒ apply between_trivial2
|_ : _ ⊢ _ ⇒ idtac
end.

Ltac *cong* :=
match goal with
|_ : Cong ?A ?B ?C ?BD ⊢ Cong ?A ?B ?C ?D ⇒ assumption
|_ : _ ⊢ Cong ?A ?B ?A ?B ⇒ apply cong_reflexivity
|_ : _ ⊢ Cong ?A ?A ?B ?B ⇒ apply cong_trivial_identity

|$H0$: Cong ?A ?B ?C ?D ⊢ Cong ?A ?B ?C ?C ⇒ assumption
|$H0$: Cong ?A ?B ?C ?D ⊢ Cong ?A ?B ?D ?C ⇒ apply (cong_right_commutativity _ _ _ _ $H0$)
|$H0$: Cong ?A ?B ?C ?D ⊢ Cong ?B ?A ?D ?C ⇒ apply (cong_commutativity _ _ _ _ $H0$)
|$H0$: Cong ?A ?B ?C ?D ⊢ Cong ?B ?A ?C ?D ⇒ apply (cong_left_commutativity _ _ _ _ $H0$)

|$H0$: Cong ?A ?B ?C ?D ⊢ Cong ?C ?D ?A ?B ⇒ apply (cong_symmetry _ _ _ _ $H0$)
|$H0$: Cong ?A ?B ?C ?D ⊢ Cong ?C ?D ?B ?A ⇒ apply (cong_symmetry _ _ _ _ (cong_left_commutativity _ _ _ _ $H0$))
|$H0$: Cong ?A ?B ?C ?D ⊢ Cong ?D ?C ?B ?B ⇒ apply (cong_symmetry _ _ _ _ (cong_commutativity _ _ _ _ $H0$))
|$H0$: Cong ?A ?B ?C ?D ⊢ Cong ?D ?C ?A ?B ⇒ apply (cong_symmetry _ _ _ _ (cong_right_commutativity _ _ _ _ $H0$))
|_ : _ ⊢ _ ⇒ idtac
end.

Section T11.

```
Context `{MT:Tarski_2D}.
Context `{EqDec:EqDecidability Tpoint}.
Definition Conga := fun A B C D E F ⇒
  A ≠ B ∧ C ≠ B ∧ D ≠ E ∧ F ≠ E ∧
  ∃ A', ∃ C', ∃ D', ∃ F',
  Bet B A A' ∧ Cong A A' E D ∧
  Bet B C C' ∧ Cong C C' E F ∧
  Bet E D D' ∧ Cong D D' B A ∧
  Bet E F F' ∧ Cong F F' B C ∧
  Cong A' C' D' F'.
Lemma l11_3 : ∀ A B C D E F,
Conga A B C D E F →
  ∃ A', ∃ C', ∃ D', ∃ F',
  out B A' A ∧ out B C C' ∧ out E D' D ∧ out E F F' ∧
  Cong_3 A' B C' D' E F'.
Lemma cong_preserves_bet : ∀ B A' A0 E D' D0,
   Bet B A' A0 → Cong B A' E D' → Cong B A0 E D0 → out E D' D0 →
   Bet E D' D0.
Lemma l11_aux : ∀ B A A' A0 E D D' D0,
      out B A A' → out E D D' → Cong B A' E D' →
      Bet B A A0 → Bet E D D0 → Cong A A0 E D
      → Cong D D0 B A → Cong B A0 E D0 ∧ Cong A' A0 D' D0.
Lemma l11_3_bis : ∀ A B C D E F,
  (∃ A', ∃ C', ∃ D', ∃ F',
  out B A' A ∧ out B C' C ∧ out E D' D ∧ out E F' F ∧
  Cong_3 A' B C' D' E F') → Conga A B C D E F.
Lemma out_cong_cong : ∀ B A A0 E D D0,
  out B A A0 → out E D D0 →
  Cong B A E D → Cong B A0 E D0 →
  Cong A A0 D D0.
Lemma l11_4_1 : ∀ A B C D E F,
   Conga A B C D E F → A≠B ∧ C≠B ∧ D≠E ∧ F≠E ∧
   (∀ A' C' D' F', out B A' A ∧ out B C' C ∧ out E D' D ∧ out E F' F ∧
   Cong B A' E D' ∧ Cong B C' E F' → Cong A' C' D' F').
Lemma l11_4_2 : ∀ A B C D E F,
   (A≠B ∧ C≠B ∧ D≠E ∧ F≠E ∧
   (∀ A' C' D' F', out B A' A ∧ out B C' C ∧ out E D' D ∧ out E F' F ∧
   Cong B A' E D' ∧ Cong B C' E F' → Cong A' C' D' F')) → Conga A B C D E F.
Lemma conga_refl : ∀ A B C, A ≠ B → C ≠ B → Conga A B C A B C.
Lemma conga_sym : ∀ A B C A' B' C', Conga A B C A' B' C' → Conga A' B' C' A B C.
```

Lemma out_conga :
 $\forall$ A B C A' B' C' $A0$ $C0$ $A1$ $C1$,
 Conga A B C A' B' C' $\to$
 out B A $A0$ $\to$
 out B C $C0$ $\to$
 out B' A' $A1$ $\to$
 out B' C' $C1$ $\to$
 Conga $A0$ B $C0$ $A1$ B' $C1$.

Lemma bet_out : $\forall$ A B C, $B \neq A \to C \neq A \to$ Bet A B C $\to$ out A B C.

Lemma cong3_conga : $\forall$ A B C A' B' C',
 $A \neq B \to C \neq B \to$
 Cong_3 A B C A' B' C' $\to$
 Conga A B C A' B' C'.

Lemma cong3_conga2 : $\forall$ A B C A' B' C' A'' B'' C'',
 Cong_3 A B C A' B' C' $\to$
 Conga A B C A'' B'' C'' $\to$
 Conga A' B' C' A'' B'' C''.

Lemma diff_out : $\forall$ A B, $A \neq B \to$ out B A A.

Lemma conga_diff1 : $\forall$ A B C A' B' C', Conga A B C A' B' C' $\to A \neq B$.

Lemma conga_diff2 : $\forall$ A B C A' B' C', Conga A B C A' B' C' $\to C \neq B$.

Lemma conga_trans : $\forall$ A B C A' B' C' A'' B'' C'',
 Conga A B C A' B' C' $\to$ Conga A' B' C' A'' B'' C'' $\to$
 Conga A B C A'' B'' C''.

Lemma conga_pseudo_refl : $\forall$ A B C,
 $A \neq B \to C \neq B \to$ Conga A B C C B A.

Lemma conga_trivial_1 : $\forall$ A B C D,
 $A \neq B \to C \neq D \to$ Conga A B A C D C.

Lemma l11_10 : $\forall$ A B C D E F A' C' D' F',
 Conga A B C D E F $\to$ out B A' A $\to$ out B C' C $\to$ out E D' D $\to$ out E F' F $\to$
 Conga A' B C' D' E F'.

Lemma l11_13 : $\forall$ A B C D E F A' D',
 Conga A B C D E F $\to$ Bet A B A' $\to A' \neq B \to$ Bet D E D' $\to D' \neq E \to$ Conga A' B C D' E F.

Lemma conga_right_comm : $\forall$ A B C D E F, Conga A B C D E F $\to$ Conga A B C F E D.

Lemma conga_left_comm : $\forall$ A B C D E F, Conga A B C D E F $\to$ Conga C B A D E F.

Lemma conga_comm : $\forall$ A B C D E F, Conga A B C D E F $\to$ Conga C B A F E D.

Definition Distincts := fun A B C : Tpoint $\Rightarrow$ $A \neq B \land A \neq C \land B \neq C$.

Lemma conga_line : $\forall$ A B C A' B' C',

Distincts A B C → Distincts A' B' C' → Bet A B C → Bet A' B' C' →
Conga A B C A' B' C'.

Lemma l11_14 : ∀ A B C A' C',
Bet A B A' → Distincts A B A' → Bet C B C' → Distincts C B C' →
Conga A B C A' B C'.

Lemma l11_16 : ∀ A B C A' B' C',
Per A B C → $A \neq B$ → $C \neq B$ →
Per A' B' C' → $A' \neq B'$ → $C' \neq B'$ →
Conga A B C A' B' C'.

Lemma l11_17 : ∀ A B C A' B' C',
Per A B C → Conga A B C A' B' C' → Per A' B' C'.

Lemma l11_18_1 : ∀ A B C D,
Bet C B D → Distincts B C D → $A \neq B$ → Per A B C → Conga A B C A B D.

Lemma l11_18_2 : ∀ A B C D,
Bet C B D → Distincts B C D → $A \neq B$ → Conga A B C A B D → Per A B C.

Lemma not_bet_out : ∀ A B C,
$A \neq B$ → $C \neq B$ → Col A B C → ¬Bet A B C →
out B A C.

Lemma cong3_preserves_out : ∀ A B C A' B' C',
out A B C → Cong_3 A B C A' B' C' → out A' B' C'.

Lemma l11_21_a : ∀ A B C A' B' C', out B A C → Conga A B C A' B' C' → out B' A' C'.

Lemma l11_21_b : ∀ A B C A' B' C',
out B A C → out B' A' C' → Conga A B C A' B' C'.

Lemma col_two_sides : ∀ A B C P Q,
Col A B C → $A \neq C$ → two_sides A B P Q →
two_sides A C P Q.

Lemma col_one_side : ∀ A B C P Q,
Col A B C → $A \neq C$ → one_side A B P Q → one_side A C P Q.

Lemma l11_22_aux : ∀ A B C C',
Conga A B C A B C' → out B C C' ∨ two_sides A B C C'.

Lemma per_cong_mid : ∀ A B C H,
$B \neq C$ → Bet A B C → Cong A H C H → Per H B C →
is_midpoint B A C.

Lemma cong2_conga_cong : ∀ A B C A' B' C',
Conga A B C A' B' C' → Cong A B A' B' → Cong B C B' C' →
Cong A C A' C'.

Lemma not_two_sides : ∀ A B P, ¬ two_sides A B P P.

Lemma segment_construction_3 : $\forall$ A B X Y, $A \neq B \rightarrow X \neq Y \rightarrow \exists$ C , out A B C $\wedge$ Cong A C X Y.

Lemma ex_per_cong : $\forall$ A B C D X Y,
$A \neq B \rightarrow X \neq Y \rightarrow$ Col A B C $\rightarrow \neg$Col A B D $\rightarrow$
$\exists$ P , Per P C A $\wedge$ Cong P C X Y $\wedge$ one_side A B P D.

Lemma not_out_bet : $\forall$ A B C, Col A B C $\rightarrow \neg$ out B A C $\rightarrow$ Bet A B C.

Lemma angle_construction_1 : $\forall$ A B C A' B' P,
$\neg$Col A B C $\rightarrow \neg$Col A' B' P $\rightarrow$
$\exists$ C' , Conga A B C A' B' C' $\wedge$ one_side A' B' C' P.

Lemma angle_construction_2 : $\forall$ A B C A' B' P,
Distincts A B C $\rightarrow$ $A' \neq B' \rightarrow \neg$Col A' B' P $\rightarrow$
$\exists$ C' , Conga A B C A' B' C' $\wedge$ (one_side A' B' C' P $\vee$ Col A' B' C').

Lemma l11_15 : $\forall$ A B C D E P, $\neg$Col A B C $\rightarrow \neg$Col D E P $\rightarrow$
$\exists$ F , Conga A B C D E F $\wedge$ one_side E D F P $\wedge$
($\forall$ $F1$ $F2$, ((Conga A B C D E $F1$ $\wedge$ one_side E D $F1$ P) $\wedge$
(Conga A B C D E $F2$ $\wedge$ one_side E D $F2$ P)) $\rightarrow$ out E $F1$ $F2$).

Lemma l11_19 : $\forall$ A B $P1$ $P2$,
Per A B $P1$ $\rightarrow$ Per A B $P2$ $\rightarrow$ one_side A B $P1$ $P2$ $\rightarrow$
out B $P1$ $P2$.

Lemma l11_22_bet :
$\forall$ A B C P A' B' C' P',
Bet A B C $\rightarrow$
two_sides P' B' A' C' $\rightarrow$
Conga A B P A' B' P' $\wedge$ Conga P B C P' B' C' $\rightarrow$
Bet A' B' C'.

Lemma not_bet_and_out :
$\forall$ A B C,
$\neg$ (Bet A B C $\wedge$ out B A C).

Lemma out_to_bet :
$\forall$ A B C A' B' C',
Col A' B' C' $\rightarrow$
(out B A C $\leftrightarrow$ out B' A' C') $\rightarrow$
Bet A B C $\rightarrow$
Bet A' B' C'.

Lemma l11_22a :
$\forall$ A B C P A' B' C' P',
two_sides B P A C $\wedge$ two_sides B' P' A' C' $\wedge$
Conga A B P A' B' P' $\wedge$ Conga P B C P' B' C' $\rightarrow$

Conga A B C A' B' C'.

Lemma l11_22b :
∀ A B C P A' B' C' P',
one_side B P A C ∧ one_side B' P' A' C' ∧
Conga A B P A' B' P' ∧ Conga P B C P' B' C' →
Conga A B C A' B' C'.

Lemma l11_22 :
∀ A B C P A' B' C' P',
((two_sides B P A C ∧ two_sides B' P' A' C')\/
(one_side B P A C ∧ one_side B' P' A' C')) ∧
Conga A B P A' B' P' ∧ Conga P B C P' B' C' →
Conga A B C A' B' C'.

Definition InAngle P A B C :=
$A{\neq}B$ ∧ $C{\neq}B$ ∧ $P{\neq}B$ ∧ ∃ X , Bet A X C ∧ (X=B ∨ out B X P).

Lemma l11_24 :
∀ P A B C,
InAngle P A B C →
InAngle P C B A.

Lemma out_in_angle :
∀ A B C P,
out B A C →
out B P A →
InAngle P A B C.

Lemma col_in_angle :
∀ A B C P,
$A \neq B \rightarrow C \neq B \rightarrow P \neq B \rightarrow$
out B A P ∨ out B C P →
InAngle P A B C.

Lemma in_angle_two_sides :
∀ A B C P,
¬ Col B A P → ¬ Col B C P →
InAngle P A B C →
two_sides P B A C.

Lemma in_angle_out :
∀ A B C P,
out B A C →
InAngle P A B C →
out B A P.

Lemma col_in_angle_out :
∀ A B C P,

```
   Col B A P →
   ¬ Bet A B C →
   InAngle P A B C →
   out B A P.

Lemma l11_25_aux : ∀ P A B C A' ,
  InAngle P A B C →
  ¬ Bet A B C →
  out B A' A →
  InAngle P A' B C.

Lemma l11_25 : ∀ P A B C A' C' P',
  InAngle P A B C →
  out B A' A →
  out B C' C →
  out B P' P →
  InAngle P' A' B C'.

Definition lea := fun A B C D E F ⇒
    ∃ P , InAngle P D E F ∧ Conga A B C D E P.

Lemma segment_construction_0 : ∀ A B A', ∃ B' , Cong A' B' A B.

Lemma angle_construction_3 :
  ∀ A B C A' B',
   A ≠ B → C ≠ B → A' ≠ B' →
   ∃ C' , Conga A B C A' B' C'.

Lemma l11_28 : ∀ A B C D A' B' C',
  Cong_3 A B C A' B' C' → Col A C D →
  ∃ D' , Cong A D A' D' ∧ Cong B D B' D' ∧ Cong C D C' D'.

Lemma bet_conga_bet :
  ∀ A B C A' B' C',
   Bet A B C →
   Conga A B C A' B' C' →
   Bet A' B' C'.

Lemma out_in_angle_out :
  ∀ A B C P,
   out B A C →
   InAngle P A B C →
   out B A P.

Lemma two_sides_not_col :
  ∀ A B X Y,
   two_sides A B X Y →
   ¬ Col A B X.

Lemma one_side_not_col :
```

$\forall\ A\ B\ X\ Y,$
one_side $A\ B\ X\ Y \rightarrow$
$\neg$ Col $A\ B\ X.$

Lemma out_out_one_side :
$\forall\ A\ B\ X\ Y\ Z,$
one_side $A\ B\ X\ Y \rightarrow$
out $A\ Y\ Z \rightarrow$
one_side $A\ B\ X\ Z.$

Lemma in_angle_one_side :
$\forall\ A\ B\ C\ P,$
$\neg$ Col $A\ B\ C \rightarrow$
$\neg$ Col $B\ A\ P \rightarrow$
InAngle $P\ A\ B\ C \rightarrow$
one_side $A\ B\ P\ C.$

Lemma or_bet_out : $\forall\ A\ B\ C,\ A \neq B \rightarrow C \neq B \rightarrow$ (Bet $A\ B\ C \vee$ out $B\ A\ C \vee \neg$Col $A\ B\ C$).

Lemma in_angle_trivial_1 : $\forall\ A\ B\ C,\ A \neq B \rightarrow C \neq B \rightarrow$ InAngle $A\ A\ B\ C.$

Lemma in_angle_trivial_2 : $\forall\ A\ B\ C,\ A \neq B \rightarrow C \neq B \rightarrow$ InAngle $C\ A\ B\ C.$

Lemma col_out2_col : $\forall\ A\ B\ C\ AA\ CC,$ Col $A\ B\ C \rightarrow$ out $B\ A\ AA \rightarrow$ out $B\ C\ CC \rightarrow$ Col $AA\ B\ CC.$

Lemma col_conga_col : $\forall\ A\ B\ C\ D\ E\ F,$ Col $A\ B\ C \rightarrow$ Conga $A\ B\ C\ D\ E\ F \rightarrow$ Col $D\ E\ F.$

Lemma ncol_conga_ncol : $\forall\ A\ B\ C\ D\ E\ F,\ \neg$Col $A\ B\ C \rightarrow$ Conga $A\ B\ C\ D\ E\ F \rightarrow \neg$Col $D\ E\ F.$

Lemma l11_29_a : $\forall\ A\ B\ C\ D\ E\ F,$ lea $A\ B\ C\ D\ E\ F \rightarrow \exists\ Q\ ,$ InAngle $C\ A\ B\ Q \wedge$ Conga $A\ B\ Q\ D\ E\ F.$

Lemma in_angle_line : $\forall\ A\ B\ C\ P,\ P \neq B \rightarrow A \neq B \rightarrow C \neq B \rightarrow$ Bet $A\ B\ C \rightarrow$ InAngle $P\ A\ B\ C.$

Lemma l11_29_b : $\forall\ A\ B\ C\ D\ E\ F,\ (\exists\ Q\ ,$ InAngle $C\ A\ B\ Q \wedge$ Conga $A\ B\ Q\ D\ E\ F) \rightarrow$ lea $A\ B\ C\ D\ E\ F.$

Lemma bet_in_angle_bet : $\forall\ A\ B\ C\ P,$ Bet $A\ B\ P \rightarrow$ InAngle $P\ A\ B\ C \rightarrow$ Bet $A\ B\ C.$

Lemma lea_line : $\forall\ A\ B\ C\ P,$ Bet $A\ B\ P \rightarrow$ lea $A\ B\ P\ A\ B\ C \rightarrow$ Bet $A\ B\ C.$

Lemma bet2_out_out : $\forall\ A\ B\ C\ B'\ C',\ B \neq A \rightarrow B' \neq A \rightarrow$ out $A\ C\ C' \rightarrow$ Bet $A\ B\ C \rightarrow$ Bet $A\ B'\ C' \rightarrow$ out $A\ B\ B'.$

Lemma eq_conga_out : $\forall\ A\ B\ D\ E\ F,$ Conga $A\ B\ A\ D\ E\ F \rightarrow$ out $E\ D\ F.$

Lemma out_conga_out : $\forall\ A\ B\ C\ D\ E\ F,$ out $B\ A\ C \rightarrow$ Conga $A\ B\ C\ D\ E\ F \rightarrow$ out $E\ D\ F.$

Lemma out_one_side : $\forall\ A\ B\ X\ Y,\ (\neg$Col $A\ B\ X \vee \neg$ Col $A\ B\ Y) \rightarrow$ out $A\ X\ Y \rightarrow$ one_side $A\ B\ X\ Y.$

Lemma conga_ex_cong3 : ∀ A B C A' B' C',
Conga A B C A' B' C' → ∃ AA, ∃ CC, out B A AA → out B C CC →
Cong_3 AA B CC A' B' C'.

Lemma conga_preserves_in_angle : ∀ A B C I A' B' C' I',
Conga A B C A' B' C' → Conga A B I A' B' I' →
InAngle I A B C → one_side A' B' I' C' →
InAngle I' A' B' C'.

Lemma l11_30 : ∀ A B C D E F A' B' C' D' E' F',
lea A B C D E F →
Conga A B C A' B' C' →
Conga D E F D' E' F' →
lea A' B' C' D' E' F'.

Lemma l11_31_1 : ∀ A B C D E F,
out B A C → $D \neq E$ → $F \neq E$ →
lea A B C D E F.

Lemma l11_31_2 : ∀ A B C D E F,
$A \neq B$ → $C \neq B$ → $D \neq E$ → $F \neq E$ →
Bet D E F →
lea A B C D E F.

Lemma lea_refl : ∀ A B C,
$A \neq B$ → $C \neq B$ → lea A B C A B C.

Lemma in_angle_trans : ∀ A B C D E,
InAngle C A B D → InAngle D A B E → InAngle C A B E.

Lemma lea_trans : ∀ A B C $A1$ $B1$ $C1$ $A2$ $B2$ $C2$,
lea A B C $A1$ $B1$ $C1$ →
lea $A1$ $B1$ $C1$ $A2$ $B2$ $C2$ →
lea A B C $A2$ $B2$ $C2$.

Lemma out2_out_1 : ∀ B C D X,
out B X C → out B X D → out B C D.

Lemma out2_out_2 : ∀ B C D X,
out B C X → out B D X → out B C D.

Lemma out_bet_out_1 : ∀ A B C P,
out P A C → Bet A B C → out P A B.

Lemma out_bet_out_2 : ∀ A B C P,
out P A C → Bet A B C → out P B C.

Lemma in_angle_asym : ∀ A B C D,
InAngle D A B C → InAngle C A B D → Conga A B C A B D.

Lemma lea_asym : ∀ A B C D E F,
lea A B C D E F → lea D E F A B C → Conga A B C D E F.

Lemma two_sides_in_angle : $\forall$ A B C P P',
$B \neq P' \rightarrow$
two_sides B P A C $\rightarrow$
Bet P B P' $\rightarrow$
InAngle P A B C $\vee$ InAngle P' A B C.

Lemma col_one_side_out : $\forall$ A B X Y,
Col A X Y $\rightarrow$
one_side A B X Y $\rightarrow$
out A X Y.

Lemma col_two_sides_bet :
$\forall$ A B X Y,
Col A X Y $\rightarrow$
two_sides A B X Y $\rightarrow$
Bet X A Y.

Lemma col_perp_perp_col :
$\forall$ A B X Y P,
$P \neq A \rightarrow$
Col A B P $\rightarrow$
Perp A B X P $\rightarrow$
Perp P A Y P $\rightarrow$
Col Y X P.

Lemma out_two_sides_two_sides :
$\forall$ A B X Y P PX,
$A \neq PX \rightarrow$
Col A B PX $\rightarrow$
out PX X P $\rightarrow$
two_sides A B P Y $\rightarrow$
two_sides A B X Y.

Lemma not_two_sides_one_side_aux :
$\forall$ A B X Y PX,
$A \neq B \rightarrow PX \neq A \rightarrow$
Perp A B X PX $\rightarrow$
Col A B PX $\rightarrow$
$\neg$ Col X A B $\rightarrow$
$\neg$ Col Y A B $\rightarrow$
$\neg$ two_sides A B X Y $\rightarrow$
one_side A B X Y.

Lemma not_two_sides_one_side :
$\forall$ A B X Y,
$A \neq B \rightarrow$
$\neg$ Col X A B $\rightarrow$

$\neg$ Col Y A B $\rightarrow$
$\neg$ two_sides A B X Y $\rightarrow$
one_side A B X Y.

Lemma l9_31 :
$\forall$ A X Y Z,
one_side A X Y Z $\rightarrow$
one_side A Z Y X $\rightarrow$
two_sides A Y X Z.

Lemma in_angle_reverse :
$\forall$ A B A' C D,
$A \neq B \rightarrow C \neq B \rightarrow D \neq B \rightarrow A' \neq B \rightarrow$
Bet A B A' $\rightarrow$
InAngle C A B D $\rightarrow$
InAngle D A' B C.

Lemma l11_36 : $\forall$ A B C D E F A' D',
$A \neq B \rightarrow A' \neq B \rightarrow D \neq E \rightarrow D' \neq E \rightarrow$
Bet A B A' $\rightarrow$ Bet D E D' $\rightarrow$
(lea A B C D E F $\leftrightarrow$ lea D' E F A' B C).

Definition lta := fun A B C D E F $\Rightarrow$ lea A B C D E F $\wedge$ $\neg$Conga A B C D E F.

Definition gta := fun A B C D E F $\Rightarrow$ lta D E F A B C.

Definition acute := fun A B C $\Rightarrow$ $\exists$ A', $\exists$ B', $\exists$ C', Per A' B' C' $\wedge$ lta A B C A' B' C'.

Definition obtuse := fun A B C $\Rightarrow$ $\exists$ A', $\exists$ B', $\exists$ C', Per A' B' C' $\wedge$ gta A B C A' B' C'.

Lemma l11_41_aux : $\forall$ A B C D,
$\neg$ Col A B C $\rightarrow$
Bet B A D $\rightarrow$
$A \neq D \rightarrow$
lta A C B C A D.

Lemma l11_41 : $\forall$ A B C D,
$\neg$ Col A B C $\rightarrow$
Bet B A D $\rightarrow$
$A \neq D \rightarrow$
lta A C B C A D $\wedge$ lta A B C C A D.

Lemma not_conga : $\forall$ A B C A' B' C' D E F ,
Conga A B C A' B' C' $\rightarrow$
$\neg$ Conga A B C D E F $\rightarrow$
$\neg$ Conga A' B' C' D E F.

Lemma not_conga_sym : $\forall$ A B C D E F,

¬ Conga $A\ B\ C\ D\ E\ F$ →
¬ Conga $D\ E\ F\ A\ B\ C$.

Lemma not_and_lta : ∀ $A\ B\ C\ D\ E\ F$, ~(lta $A\ B\ C\ D\ E\ F$ ∧ lta $D\ E\ F\ A\ B\ C$).

Lemma conga_preserves_lta : ∀ $A\ B\ C\ D\ E\ F$ A' B' C' D' E' F',
Conga $A\ B\ C$ A' B' C' →
Conga $D\ E\ F$ D' E' F' →
lta $A\ B\ C\ D\ E\ F$ →
lta A' B' C' D' E' F'.

Lemma conga_preserves_gta : ∀ $A\ B\ C\ D\ E\ F$ A' B' C' D' E' F' ,
Conga $A\ B\ C$ A' B' C' →
Conga $D\ E\ F$ D' E' F' →
gta $A\ B\ C\ D\ E\ F$ →
gta A' B' C' D' E' F'.

Lemma lta_trans : ∀ $A\ B\ C\ A1\ B1\ C1\ A2\ B2\ C2$,
lta $A\ B\ C\ A1\ B1\ C1$ →
lta $A1\ B1\ C1\ A2\ B2\ C2$ →
lta $A\ B\ C\ A2\ B2\ C2$.

Lemma gta_trans : ∀ $A\ B\ C\ A1\ B1\ C1\ A2\ B2\ C2$,
gta $A\ B\ C\ A1\ B1\ C1$ →
gta $A1\ B1\ C1\ A2\ B2\ C2$ →
gta $A\ B\ C\ A2\ B2\ C2$.

Lemma lea_left_comm : ∀ $A\ B\ C\ D\ E\ F$, lea $A\ B\ C\ D\ E\ F$ → lea $C\ B\ A\ D\ E\ F$.

Lemma lea_right_comm : ∀ $A\ B\ C\ D\ E\ F$, lea $A\ B\ C\ D\ E\ F$ → lea $A\ B\ C\ F\ E\ D$.

Lemma lea_comm : ∀ $A\ B\ C\ D\ E\ F$, lea $A\ B\ C\ D\ E\ F$ → lea $C\ B\ A\ F\ E\ D$.

Lemma lta_left_comm : ∀ $A\ B\ C\ D\ E\ F$, lta $A\ B\ C\ D\ E\ F$ → lta $C\ B\ A\ D\ E\ F$.

Lemma lta_right_comm : ∀ $A\ B\ C\ D\ E\ F$, lta $A\ B\ C\ D\ E\ F$ → lta $A\ B\ C\ F\ E\ D$.

Lemma lta_comm : ∀ $A\ B\ C\ D\ E\ F$, lta $A\ B\ C\ D\ E\ F$ → lta $C\ B\ A\ F\ E\ D$.

Lemma l11_43_aux : ∀ $A\ B\ C$, ¬Col $A\ B\ C$ → (Per $B\ A\ C$ ∨ obtuse $B\ A\ C$) → acute $A\ B\ C$.

Lemma obtuse_sym : ∀ $A\ B\ C$, obtuse $A\ B\ C$ → obtuse $C\ B\ A$.

Lemma acute_sym : ∀ $A\ B\ C$, acute $A\ B\ C$ → acute $C\ B\ A$.

Lemma l11_43 : ∀ $A\ B\ C$, ¬Col $A\ B\ C$ → (Per $B\ A\ C$ ∨ obtuse $B\ A\ C$) → acute $A\ B\ C$ ∧ acute $A\ C\ B$.

Lemma acute_lea_acute : ∀ $A\ B\ C\ D\ E\ F$, acute $D\ E\ F$ → lea $A\ B\ C\ D\ E\ F$ → acute $A\ B\ C$.

`Definition` gea := `fun` $A\ B\ C\ D\ E\ F$ ⇒ lea $D\ E\ F\ A\ B\ C$.

Lemma obtuse_gea_obtuse : ∀ $A\ B\ C\ D\ E\ F$, obtuse $D\ E\ F$ → gea $A\ B\ C\ D\ E\ F$ → obtuse $A\ B\ C$.

Lemma lea_acute_obtuse : $\forall$ A B C D E F, acute A B C $\rightarrow$ obtuse D E F $\rightarrow$ lea A B C D E F.

Lemma l11_44_1_a : $\forall$ A B C, $\neg$Col A B C $\rightarrow$ Cong B A B C $\rightarrow$ Conga B A C B C A.

Lemma l11_44_2_a : $\forall$ A B C, $\neg$Col A B C $\rightarrow$ lt B A B C $\rightarrow$ lta B C A B A C.

Lemma not_lta_and_cong : $\forall$ A B C D E F, ~(lta A B C D E F $\wedge$ Conga A B C D E F).

Lemma not_gta_and_cong : $\forall$ A B C D E F, ~(gta A B C D E F $\wedge$ Conga A B C D E F).

Lemma not_lta_and_gta : $\forall$ A B C D E F, ~(lta A B C D E F $\wedge$ gta A B C D E F).

Lemma conga_sym_equiv : $\forall$ A B C A' B' C', Conga A B C A' B' C' $\leftrightarrow$ Conga A' B' C' A B C.

Lemma Conga_dec :
 $\forall$ A B C D E F,
  Conga A B C D E F $\vee$ $\neg$ Conga A B C D E F.

Lemma or_lt_cong_gt : $\forall$ A B C D, lt A B C D $\vee$ gt A B C D $\vee$ Cong A B C D.

Lemma lta_not_conga : $\forall$ A B C D E F, $A \neq B$ $\rightarrow$ $C \neq B$ $\rightarrow$ $D \neq E$ $\rightarrow$ $F \neq E$ $\rightarrow$ lta A B C D E F $\rightarrow$ $\neg$ Conga A B C D E F.

Lemma l11_44_1_b : $\forall$ A B C, $\neg$Col A B C $\rightarrow$ Conga B A C B C A $\rightarrow$ Cong B A B C.

Lemma l11_44_2_b : $\forall$ A B C, $\neg$Col A B C $\rightarrow$ lta B A C B C A $\rightarrow$ lt B C B A.

Lemma l11_44_1 : $\forall$ A B C, $\neg$Col A B C $\rightarrow$ (Conga B A C B C A $\leftrightarrow$ Cong B A B C).

Lemma l11_44_2 : $\forall$ A B C, $\neg$Col A B C $\rightarrow$ (lta B A C B C A $\leftrightarrow$ lt B C B A).

Lemma between_symmetric : $\forall$ A B X Y P, Bet A P B $\rightarrow$ Cong A P X Y $\rightarrow$ ($\exists$ P', Bet A P' B $\wedge$ Cong B P' X Y).

Lemma le_left_comm : $\forall$ A B C D, le A B C D $\rightarrow$ le B A C D.

Lemma le_right_comm : $\forall$ A B C D, le A B C D $\rightarrow$ le A B D C.

Lemma le_comm : $\forall$ A B C D, le A B C D $\rightarrow$ le B A D C.

Lemma ge_left_comm : $\forall$ A B C D, ge A B C D $\rightarrow$ ge B A C D.

Lemma ge_right_comm : $\forall$ A B C D, ge A B C D $\rightarrow$ ge A B D C.

Lemma ge_comm : $\forall$ A B C D, ge A B C D $\rightarrow$ ge B A D C.

Lemma lt_right_comm : $\forall$ A B C D, lt A B C D $\rightarrow$ lt A B D C.

Lemma lt_left_comm : $\forall$ A B C D, lt A B C D $\rightarrow$ lt B A C D.

Lemma lt_comm : $\forall$ A B C D, lt A B C D $\rightarrow$ lt B A D C.

Lemma gt_left_comm : $\forall$ A B C D, gt A B C D $\rightarrow$ gt B A C D.

Lemma gt_right_comm : $\forall$ A B C D, gt A B C D $\rightarrow$ gt A B D C.

Lemma gt_comm : $\forall$ A B C D, gt A B C D $\rightarrow$ gt B A D C.

Lemma acute_distinct : $\forall$ A B C, acute A B C $\rightarrow$ acute A B C $\wedge$ $A \neq B$ $\wedge$ $C \neq B$.

Lemma lta_diff : ∀ A B C D E F, lta A B C D E F → lta A B C D E F ∧ $A \neq B$ ∧ $C \neq B$ ∧ $D \neq E$ ∧ $F \neq E$.

Lemma l5_12_a : ∀ A B C, Col A B C → Bet A B C → le A B A C ∧ le B C A C.

Lemma l5_12_b : ∀ A B C, Col A B C → (le A B A C ∧ le B C A C) → Bet A B C.

Lemma l11_46 : ∀ A B C, ¬Col A B C → (Per A B C ∨ obtuse A B C) → lt B A A C ∧ lt B C A C.

Lemma l11_47 : ∀ A B C H , Per A C B → Perp_in H C H A B → Bet A H B ∧ Distincts A H B.

Lemma l11_49 : ∀ A B C A' B' C',
Conga A B C A' B' C' → Cong B A B' A' → Cong B C B' C' →
Cong A C A' C' ∧ ($A \neq C$ → Conga B A C B' A' C' ∧ Conga B C A B' C' A').

Lemma l11_50_1 : ∀ A B C A' B' C',
¬Col A B C → Conga B A C B' A' C' → Conga A B C A' B' C' → Cong A B A' B' →
Cong A C A' C' ∧ Cong B C B' C' ∧ Conga A C B A' C' B'.

Lemma l11_50_2 : ∀ A B C A' B' C',
¬Col A B C → Conga B C A B' C' A' → Conga A B C A' B' C' → Cong A B A' B' →
Cong A C A' C' ∧ Cong B C B' C' ∧ Conga C A B C' A' B'.

Lemma l11_51 : ∀ A B C A' B' C',
Distincts A B C → Cong A B A' B' → Cong A C A' C' → Cong B C B' C' →
Conga B A C B' A' C' ∧ Conga A B C A' B' C' ∧ Conga B C A B' C' A'.

Lemma conga_distinct : ∀ A B C D E F, Conga A B C D E F → Conga A B C D E F ∧ $A \neq B$ ∧ $C \neq B$ ∧ $D \neq E$ ∧ $F \neq E$.

Definition cong_preserves_le := l5_6.

Lemma bet_le_eq : ∀ A B C, Bet A B C → le A C B C → $A = B$.

Lemma l11_52 : ∀ A B C A' B' C',
Conga A B C A' B' C' → Cong A C A' C' → Cong B C B' C' → le B C A C →
Cong B A B' A' ∧ Conga B A C B' A' C' ∧ Conga B C A B' C' A'.

Lemma l11_53 : ∀ A B C D,
Per D C B → $C \neq D$ → Distincts A B C → Bet A B C →
lta C A D C B D ∧ lt B D A D.

End T11.

Chapter 15

Library Ch12_parallel

```
Require Export Ch11_angles.

Section T12_1.

Context `{MT:Tarski_2D_euclidean}.
Context `{EqDec:EqDecidability Tpoint}.

Definition coplanar := fun A B C D: Tpoint ⇒ True.

Definition Par_strict := fun A B C D ⇒
  A≠B ∧ C≠D ∧ coplanar A B C D ∧ ¬ ∃ X , Col X A B ∧ Col X C D.

Definition Par := fun A B C D ⇒
  Par_strict A B C D ∨ (A≠B ∧ C≠D ∧ Col A C D ∧ Col B C D).

Lemma par_reflexivity : ∀ A B, A≠B → Par A B A B.

Lemma par_strict_irreflexivity : ∀ A B,
 ¬ Par_strict A B A B.

Lemma not_par_strict_id : ∀ A B C,
 ¬ Par_strict A B A C.

Lemma par_id : ∀ A B C,
 Par A B A C → Col A B C.

Lemma par_strict_not_col_1 : ∀ A B C D,
 Par_strict A B C D → ¬ Col A B C.

Lemma par_strict_not_col_2 : ∀ A B C D,
 Par_strict A B C D → ¬ Col B C D.

Lemma par_strict_not_col_3 : ∀ A B C D,
 Par_strict A B C D → ¬ Col C D A.

Lemma par_strict_not_col_4 : ∀ A B C D,
 Par_strict A B C D → ¬ Col A B D.

Lemma par_id_1 : ∀ A B C,
 Par A B A C → Col B A C.
```

Lemma par_id_2 : $\forall$ A B C,
 Par A B A C $\rightarrow$ Col B C A.

Lemma par_id_3 : $\forall$ A B C,
 Par A B A C $\rightarrow$ Col A C B.

Lemma par_id_4 : $\forall$ A B C,
 Par A B A C $\rightarrow$ Col C B A.

Lemma par_id_5 : $\forall$ A B C,
 Par A B A C $\rightarrow$ Col C A B.

Lemma par_strict_symmetry :$\forall$ A B C D,
 Par_strict A B C D $\rightarrow$ Par_strict C D A B.

Lemma par_symmetry :$\forall$ A B C D,
 Par A B C D $\rightarrow$ Par C D A B.

Lemma par_left_comm : $\forall$ A B C D,
 Par A B C D $\rightarrow$ Par B A C D.

Lemma par_right_comm : $\forall$ A B C D,
 Par A B C D $\rightarrow$ Par A B D C.

Lemma par_comm : $\forall$ A B C D,
 Par A B C D $\rightarrow$ Par B A D C.

Lemma par_strict_left_comm : $\forall$ A B C D,
 Par_strict A B C D $\rightarrow$ Par_strict B A C D.

Lemma par_strict_right_comm : $\forall$ A B C D,
 Par_strict A B C D $\rightarrow$ Par_strict A B D C.

Lemma par_strict_comm : $\forall$ A B C D,
 Par_strict A B C D $\rightarrow$ Par_strict B A D C.

End T12_1.

Hint Resolve
 par_reflexivity par_strict_irreflexivity
 par_strict_symmetry par_strict_comm par_strict_right_comm par_strict_left_comm
 par_symmetry par_comm par_right_comm par_left_comm : *par*.

Ltac *Par* := eauto with *par*.

Section T12_2.

Context ‘{*MT*:**Tarski_2D_euclidean**}.
Context ‘{*EqDec*:**EqDecidability** Tpoint}.

Lemma Par_cases :
 $\forall$ A B C D,
 Par A B C D $\vee$ Par B A C D $\vee$ Par A B D C $\vee$ Par B A D C $\vee$
 Par C D A B $\vee$ Par C D B A $\vee$ Par D C A B $\vee$ Par D C B A $\rightarrow$

Par A B C D.

Lemma Par_perm :
$\forall$ A B C D,
Par A B C D $\rightarrow$
Par A B C D $\wedge$ Par B A C D $\wedge$ Par A B D C $\wedge$ Par B A D C $\wedge$
Par C D A B $\wedge$ Par C D B A $\wedge$ Par D C A B $\wedge$ Par D C B A.

Lemma Par_strict_cases :
$\forall$ A B C D,
Par_strict A B C D $\vee$ Par_strict B A C D $\vee$ Par_strict A B D C $\vee$ Par_strict B A D C $\vee$
Par_strict C D A B $\vee$ Par_strict C D B A $\vee$ Par_strict D C A B $\vee$ Par_strict D C B A
$\rightarrow$
Par_strict A B C D.

Lemma Par_strict_perm :
$\forall$ A B C D,
Par_strict A B C D $\rightarrow$
Par_strict A B C D $\wedge$ Par_strict B A C D $\wedge$ Par_strict A B D C $\wedge$ Par_strict B A D C $\wedge$
Par_strict C D A B $\wedge$ Par_strict C D B A $\wedge$ Par_strict D C A B $\wedge$ Par_strict D C B A.

Lemma l12_6 : $\forall$ A B C D,
Par_strict A B C D $\rightarrow$ one_side A B C D.

Lemma perp_perp_col : $\forall$ X Y Z A B,
Perp X Y A B $\rightarrow$ Perp X Z A B $\rightarrow$ Col X Y Z.

Lemma Perp_dec : $\forall$ A B C D, Perp A B C D $\vee$ $\neg$ Perp A B C D.

Lemma perp_perp_col_col : $\forall$ $X1$ $X2$ $Y1$ $Y2$ A B,
Perp $X1$ $X2$ A B $\rightarrow$ Perp $Y1$ $Y2$ A B $\rightarrow$ Col $X1$ $Y1$ $Y2$ $\rightarrow$ Col $X2$ $Y1$ $Y2$.

Lemma l12_9 : $\forall$ $A1$ $A2$ $B1$ $B2$ $C1$ $C2$,
coplanar $A1$ $A2$ $B1$ $B2$ $\rightarrow$ Perp $A1$ $A2$ $C1$ $C2$ $\rightarrow$ Perp $B1$ $B2$ $C1$ $C2$ $\rightarrow$
Par $A1$ $A2$ $B1$ $B2$.

Lemma parallel_existence : $\forall$ A B P, $A \neq B$ $\rightarrow$ $\exists$ C, $\exists$ D, $C \neq D$ $\wedge$ Par A B C D $\wedge$ Col P C D.

Lemma par_col_par : $\forall$ A B C D D',
$C \neq D'$ $\rightarrow$ Par A B C D $\rightarrow$ Col C D D' $\rightarrow$ Par A B C D'.

Lemma parallel_existence_spec : $\forall$ A B P : Tpoint,
$A \neq B$ $\rightarrow$ $\exists$ C : Tpoint, $C \neq P$ $\wedge$ Par A B P C.

Lemma par_not_col : $\forall$ A B C D X, Par_strict A B C D $\rightarrow$ Col X A B $\rightarrow$ $\neg$Col X C D.

Lemma not_strict_par1 : $\forall$ A B C D X, Par A B C D $\rightarrow$ Col A B X $\rightarrow$ Col C D X $\rightarrow$ Col A B C.

Lemma not_strict_par2 : $\forall$ A B C D X, Par A B C D $\rightarrow$ Col A B X $\rightarrow$ Col C D X $\rightarrow$ Col A B D.

Lemma not_strict_par : $\forall$ A B C D X, Par A B C D $\rightarrow$ Col A B X $\rightarrow$ Col C D X $\rightarrow$ Col A B C $\wedge$ Col A B D.

Lemma not_par_not_col : $\forall$ A B C, $A \neq B$ $\rightarrow$ $A \neq C$ $\rightarrow$ $\neg$Par A B A C $\rightarrow$ $\neg$Col A B C.

Lemma not_par_inter_unicity : $\forall$ A B C D X Y,
$A \neq B$ $\rightarrow$ $C \neq D$ $\rightarrow$ $\neg$Par A B C D $\rightarrow$ Col A B X $\rightarrow$ Col C D X $\rightarrow$ Col A B Y $\rightarrow$ Col C D Y $\rightarrow$
$X = Y$.

Lemma inter_unicity_not_par : $\forall$ A B C D P,
$\neg$Col A B C $\rightarrow$ Col A B P $\rightarrow$ Col C D P $\rightarrow$ $\neg$Par A B C D.

Lemma col_not_col_not_par :
$\forall$ A B C D,
($\exists$ P, Col A B P $\wedge$ Col C D P) $\rightarrow$
($\exists$ Q, Col C D Q $\wedge$ $\neg$Col A B Q) $\rightarrow$ $\neg$Par A B C D.

Lemma par_distincts : $\forall$ A B C D,
Par A B C D $\rightarrow$ (Par A B C D $\wedge$ $A \neq B$ $\wedge$ $C \neq D$).

Lemma par_not_col_strict : $\forall$ A B C D P,
Par A B C D $\rightarrow$ Col C D P $\rightarrow$ $\neg$Col A B P $\rightarrow$ Par_strict A B C D.

Lemma all_one_side_par_strict : $\forall$ A B C D,
$C \neq D$ $\rightarrow$ ($\forall$ P, Col C D P $\rightarrow$ one_side A B C P) $\rightarrow$
Par_strict A B C D.

Lemma par_col_par_2 : $\forall$ A B C D P,
$A \neq P$ $\rightarrow$ Col A B P $\rightarrow$ Par A B C D $\rightarrow$ Par A P C D.

Lemma par_col2_par : $\forall$ A B C D E F,
$E \neq F$ $\rightarrow$ Par A B C D $\rightarrow$ Col C D E $\rightarrow$ Col C D F $\rightarrow$ Par A B E F.

Lemma par_strict_col_par_strict : $\forall$ A B C D E,
$C \neq E$ $\rightarrow$ Par_strict A B C D $\rightarrow$ Col C D E $\rightarrow$
Par_strict A B C E.

Lemma par_strict_col2_par_strict : $\forall$ A B C D E F,
$E \neq F$ $\rightarrow$ Par_strict A B C D $\rightarrow$ Col C D E $\rightarrow$ Col C D F $\rightarrow$
Par_strict A B E F.

Lemma line_dec : $\forall$ $B1$ $B2$ $C1$ $C2$, (Col $C1$ $B1$ $B2$ $\wedge$ Col $C2$ $B1$ $B2$) $\vee$ $\neg$ (Col $C1$ $B1$ $B2$ $\wedge$ Col $C2$ $B1$ $B2$).

Lemma par_distinct : $\forall$ A B C D, Par A B C D $\rightarrow$ $A \neq B$ $\wedge$ $C \neq D$.

Definition inter $A1$ $A2$ $B1$ $B2$ X :=
($\exists$ P, Col P $B1$ $B2$ $\wedge$ $\neg$Col P $A1$ $A2$) $\wedge$
Col $A1$ $A2$ X $\wedge$ Col $B1$ $B2$ X.

Lemma inter_trivial : $\forall$ A B X, $\neg$Col A B X $\rightarrow$ inter A X B X X.

Lemma inter_sym : $\forall$ A B C D X, $C \neq D$ $\rightarrow$ inter A B C D X $\rightarrow$ inter C D A B X.

Lemma inter_left_comm : $\forall\ A\ B\ C\ D\ X$, inter $A\ B\ C\ D\ X \rightarrow$ inter $B\ A\ C\ D\ X$.

Lemma inter_right_comm : $\forall\ A\ B\ C\ D\ X$, inter $A\ B\ C\ D\ X \rightarrow$ inter $A\ B\ D\ C\ X$.

Lemma inter_comm : $\forall\ A\ B\ C\ D\ X$, inter $A\ B\ C\ D\ X \rightarrow$ inter $B\ A\ D\ C\ X$.

Lemma other_point_exists : $\forall\ A$, $\exists\ B$: Tpoint, $A \neq B$.

Lemma l12_17 : $\forall\ A\ B\ C\ D\ P$,
$A \neq B \rightarrow$ is_midpoint $P\ A\ C \rightarrow$ is_midpoint $P\ B\ D \rightarrow$ Par $A\ B\ C\ D$.

Lemma l12_18_a :
$\forall\ A\ B\ C\ D\ P$,
Cong $A\ B\ C\ D \rightarrow$ Cong $B\ C\ D\ A \rightarrow \neg$Col $A\ B\ C \rightarrow$
$B \neq D \rightarrow$ Col $A\ P\ C \rightarrow$ Col $B\ P\ D \rightarrow$
Par $A\ B\ C\ D$.

Lemma l12_18_b :
$\forall\ A\ B\ C\ D\ P$,
Cong $A\ B\ C\ D \rightarrow$ Cong $B\ C\ D\ A \rightarrow \neg$Col $A\ B\ C \rightarrow$
$B \neq D \rightarrow$ Col $A\ P\ C \rightarrow$ Col $B\ P\ D \rightarrow$
Par $B\ C\ D\ A$.

Lemma l12_18_c :
$\forall\ A\ B\ C\ D\ P$,
Cong $A\ B\ C\ D \rightarrow$ Cong $B\ C\ D\ A \rightarrow \neg$Col $A\ B\ C \rightarrow$
$B \neq D \rightarrow$ Col $A\ P\ C \rightarrow$ Col $B\ P\ D \rightarrow$
two_sides $B\ D\ A\ C$.

Lemma l12_18_d :
$\forall\ A\ B\ C\ D\ P$,
Cong $A\ B\ C\ D \rightarrow$ Cong $B\ C\ D\ A \rightarrow \neg$Col $A\ B\ C \rightarrow$
$B \neq D \rightarrow$ Col $A\ P\ C \rightarrow$ Col $B\ P\ D \rightarrow$
two_sides $A\ C\ B\ D$.

Lemma l12_18 :
$\forall\ A\ B\ C\ D\ P$,
Cong $A\ B\ C\ D \rightarrow$ Cong $B\ C\ D\ A \rightarrow \neg$Col $A\ B\ C \rightarrow$
$B \neq D \rightarrow$ Col $A\ P\ C \rightarrow$ Col $B\ P\ D \rightarrow$
Par $A\ B\ C\ D \wedge$ Par $B\ C\ D\ A \wedge$ two_sides $B\ D\ A\ C \wedge$ two_sides $A\ C\ B\ D$.

Lemma par_two_sides_two_sides :
$\forall\ A\ B\ C\ D$,
Par $A\ B\ C\ D \rightarrow$ two_sides $B\ D\ A\ C \rightarrow$
two_sides $A\ C\ B\ D$.

Lemma out_one_side_1 :
$\forall\ A\ B\ C\ D\ X$,
$A \neq B \rightarrow \neg$ Col $A\ B\ C \rightarrow$ Col $A\ B\ X \rightarrow$ out $X\ C\ D \rightarrow$
one_side $A\ B\ C\ D$.

Lemma midpoint_preserves_out :
 ∀ A B C A' B' C' M,
  out A B C →
  is_midpoint M A A' →
  is_midpoint M B B' →
  is_midpoint M C C' →
 out A' B' C'.

Lemma l12_21_b : ∀ A B C D,
 two_sides A C B D →
 (Conga B A C D C A → Par A B C D).

Lemma l6_7_1 : ∀ A B C D,
 out A B C → out A B D →
 out A C D.

Lemma l12_22_aux :
 ∀ A B C D P,
  Distincts P A C → Bet P A C → one_side P A B D →
  Conga B A P D C P →
  Par A B C D.

Lemma l12_22_b :
 ∀ A B C D P,
  out P A C → one_side P A B D → Conga B A P D C P →
  Par A B C D.

Lemma par_strict_par : ∀ A B C D,
 Par_strict A B C D → Par A B C D.

Lemma par_strict_distinct : ∀ A B C D,
 Par_strict A B C D → $A \neq B$ ∧ $C \neq D$.

Lemma col_par : ∀ A B C,
 $A \neq B$ → $B \neq C$ →
 Col A B C → Par A B B C.

End T12_2.

Hint Resolve col_par : *par*.

Section T12_3.

Context `{MT:**Tarski_2D_euclidean**}.
Context `{$EqDec$:**EqDecidability** Tpoint}.

Lemma perp_not_par : ∀ A B X Y, Perp A B X Y → ¬ Par A B X Y.

End T12_3.

Chapter 16

Library Ch12_parallel_inter_dec

Require Export Ch12_parallel.

Section T13.

Context `{MT:**Tarski_2D_euclidean**}.
Context `{$EqDec$:**EqDecidability** Tpoint}.
Context `{$InterDec$:**InterDecidability** Tpoint Col}.

Lemma two_sides_dec :
$\forall$ A B C D,
two_sides A B C D $\vee$ $\neg$ two_sides A B C D.

Lemma not_par_inter_exists : $\forall$ $A1$ $B1$ $A2$ $B2$,
$\neg$Par $A1$ $B1$ $A2$ $B2$ $\rightarrow$ $\exists$ X , Col X $A1$ $B1$ $\wedge$ Col X $A2$ $B2$.

Lemma not_par_two_sides : $\forall$ A B C D,
$A \neq B$ $\rightarrow$ $C \neq D$ $\rightarrow$ $\neg$Par A B C D $\rightarrow$
$\exists$ X , $\exists$ Y , Col C D X $\wedge$ Col C D Y $\wedge$ two_sides A B X Y.

Lemma not_par_other_side : $\forall$ A B C D P,
$A \neq B$ $\rightarrow$ $C \neq D$ $\rightarrow$ $\neg$Par A B C D $\rightarrow$ $\neg$Col A B P $\rightarrow$
$\exists$ Q , Col C D Q $\wedge$ two_sides A B P Q.

Lemma parallel_unicity_aux : $\forall$ $A1$ $A2$ $B1$ $B2$ $C1$ $C2$ P,
$\neg$ Col P $A1$ $A2$ $\rightarrow$
Par $A1$ $A2$ $B1$ $B2$ $\rightarrow$ Col P $B1$ $B2$ $\rightarrow$
Par $A1$ $A2$ $C1$ $C2$ $\rightarrow$ Col P $C1$ $C2$ $\rightarrow$
Col $C1$ $B1$ $B2$ $\wedge$ Col $C2$ $B1$ $B2$.

Lemma parallel_unicity :
$\forall$ $A1$ $A2$ $B1$ $B2$ $C1$ $C2$ P : Tpoint,
Par $A1$ $A2$ $B1$ $B2$ $\rightarrow$ Col P $B1$ $B2$ $\rightarrow$
Par $A1$ $A2$ $C1$ $C2$ $\rightarrow$ Col P $C1$ $C2$ $\rightarrow$
Col $C1$ $B1$ $B2$ $\wedge$ Col $C2$ $B1$ $B2$.

Lemma par_trans : $\forall$ $A1$ $A2$ $B1$ $B2$ $C1$ $C2$, Par $A1\ A2\ B1\ B2 \rightarrow$ Par $B1\ B2\ C1\ C2 \rightarrow$ Par $A1\ A2\ C1\ C2$.

Lemma l12_16 : $\forall\ A1\ A2\ B1\ B2\ C1\ C2\ X$,
Par $A1\ A2\ B1\ B2 \rightarrow$ inter $A1\ A2\ C1\ C2\ X \rightarrow \exists\ Y$, inter $B1\ B2\ C1\ C2\ Y$.

Lemma not_one_side_two_sides :
$\forall\ A\ B\ X\ Y$,
$A \neq B \rightarrow$
$\neg$ Col $X\ A\ B \rightarrow$
$\neg$ Col $Y\ A\ B \rightarrow$
$\neg$ one_side $A\ B\ X\ Y \rightarrow$
two_sides $A\ B\ X\ Y$.

Lemma one_or_two_sides :
$\forall\ A\ B\ X\ Y$,
$\neg$ Col $X\ A\ B \rightarrow$
$\neg$ Col $Y\ A\ B \rightarrow$
two_sides $A\ B\ X\ Y \lor$ one_side $A\ B\ X\ Y$.

Lemma one_side_dec : $\forall\ A\ B\ C\ D$,
one_side $A\ B\ C\ D \lor \neg$ one_side $A\ B\ C\ D$.

Lemma lea_cases :
$\forall\ A\ B\ C\ D\ E\ F$,
$A \neq B \rightarrow C \neq B \rightarrow D \neq E \rightarrow F \neq E \rightarrow$
lea $A\ B\ C\ D\ E\ F \lor$ lea $D\ E\ F\ A\ B\ C$.

Lemma or_lta_conga_gta : $\forall\ A\ B\ C\ D\ E\ F$,
$A \neq B \rightarrow C \neq B \rightarrow D \neq E \rightarrow F \neq E \rightarrow$
lta $A\ B\ C\ D\ E\ F \lor$ gta $A\ B\ C\ D\ E\ F \lor$ Conga $A\ B\ C\ D\ E\ F$.

Lemma not_lta_gea : $\forall\ A\ B\ C\ D\ E\ F$,
$A \neq B \rightarrow C \neq B \rightarrow D \neq E \rightarrow F \neq E \rightarrow$
$\neg$ lta $A\ B\ C\ D\ E\ F \rightarrow$
gea $A\ B\ C\ D\ E\ F$.

Lemma par_strict_one_side : $\forall\ A\ B\ C\ D\ P$,
Par_strict $A\ B\ C\ D \rightarrow$ Col $C\ D\ P \rightarrow$ one_side $A\ B\ C\ P$.

Lemma par_strict_all_one_side : $\forall\ A\ B\ C\ D$,
Par_strict $A\ B\ C\ D \rightarrow (\forall\ P$, Col $C\ D\ P \rightarrow$ one_side $A\ B\ C\ P)$.

Lemma Par_dec : $\forall\ A\ B\ C\ D$, Par $A\ B\ C\ D \lor \neg$ Par $A\ B\ C\ D$.

Lemma par_not_par : $\forall\ A\ B\ C\ D\ P\ Q$, Par $A\ B\ C\ D \rightarrow \neg$Par $A\ B\ P\ Q \rightarrow \neg$Par $C\ D\ P\ Q$.

Lemma par_inter : $\forall\ A\ B\ C\ D\ P\ Q\ X$, Par $A\ B\ C\ D \rightarrow \neg$Par $A\ B\ P\ Q \rightarrow$ Col $P\ Q\ X \rightarrow$ Col $A\ B\ X \rightarrow \exists\ Y$, Col $P\ Q\ Y \land$ Col $C\ D\ Y$.

Lemma l12_19 :
$\forall\ A\ B\ C\ D$,
$\neg$Col $A\ B\ C \rightarrow$ Par $A\ B\ C\ D \rightarrow$ Par $B\ C\ D\ A \rightarrow$

Cong A B C D ∧ Cong B C D A ∧ two_sides B D A C ∧ two_sides A C B D.

Lemma l12_20_bis :
∀ A B C D,
Par A B C D → Cong A B C D → two_sides B D A C →
Par B C D A ∧ Cong B C D A ∧ two_sides A C B D.

Lemma l12_20 :
∀ A B C D,
Par A B C D → Cong A B C D → two_sides A C B D →
Par B C D A ∧ Cong B C D A ∧ two_sides A C B D.

Lemma par_one_or_two_sides :
∀ A B C D,
Par_strict A B C D →
two_sides A C B D ∧ two_sides B D A C ∨ one_side A C B D ∧ one_side B D A C.

Lemma l12_21_a :
∀ A B C D,
two_sides A C B D →
(Par A B C D → Conga B A C D C A).

Lemma l12_21 : ∀ A B C D,
two_sides A C B D →
(Conga B A C D C A ↔ Par A B C D).

Lemma l12_22_a : ∀ A B C D P,
out P A C → one_side P A B D → Par A B C D →
Conga B A P D C P.

Lemma l12_22 :
∀ A B C D P,
out P A C → one_side P A B D →
(Conga B A P D C P ↔ Par A B C D).

Lemma l12_23 :
∀ A B C,
¬Col A B C →
∃ B', ∃ C',
two_sides A C B B' ∧ two_sides A B C C' ∧
Bet B' A C' ∧ Conga A B C B A C' ∧ Conga A C B C A B'.

Lemma parallel_trans : ∀ A B C D E F, Par A B C D → Par C D E F → Par A B E F.

Lemma not_par_strict_inter_exists :
∀ $A1$ $B1$ $A2$ $B2$,
¬Par_strict $A1$ $B1$ $A2$ $B2$ →
∃ X, Col X $A1$ $B1$ ∧ Col X $A2$ $B2$.

Lemma not_par_inter : ∀ A B A' B' X Y, ¬Par A B A' B' → (∃ P, Col P X Y ∧ (Col P A B ∨ Col P A' B')).

Lemma not_par_one_not_par : $\forall$ A B A' B' X Y, $\neg$Par A B A' B' $\rightarrow$ $\neg$Par A B X Y $\vee$ $\neg$Par A' B' X Y.

Lemma col_par_par_col : $\forall$ A B C A' B' C', Col A B C $\rightarrow$ Par A B A' B' $\rightarrow$ Par B C B' C' $\rightarrow$ Col A' B' C'.

Lemma parallel_existence1 : $\forall$ A B P, $A \neq B$ $\rightarrow$ $\exists$ Q, Par A B P Q.

Lemma par_strict_not_col : $\forall$ A B C D, Par_strict A B C D $\rightarrow$ $\forall$ X, Col A B X $\rightarrow$ $\neg$Col C D X.

Lemma perp_inter_exists : $\forall$ A B C D, Perp A B C D $\rightarrow$ $\exists$ P, Col A B P $\wedge$ Col C D P.

Lemma perp_inter_perp_in : $\forall$ A B C D, Perp A B C D $\rightarrow$ $\exists$ P, Col A B P $\wedge$ Col C D P $\wedge$ Perp_in P A B C D.

Lemma cong_conga_perp : $\forall$ A B C P, two_sides B P A C $\rightarrow$ Cong A B C B $\rightarrow$ Conga A B P C B P $\rightarrow$ Perp A C B P.

End T13.

Chapter 17

Library Ch13_1

```
Require Export project.
```

```
Section L13.
```

```
Context `{MT:Tarski_2D_euclidean}.
Context `{EqDec:EqDecidability Tpoint}.
Context `{InterDec:InterDecidability Tpoint Col}.
```

Pappus Desargues

Lemma triangle_mid_par : ∀ A B C P Q, ¬Col A B C → is_midpoint P B C → is_midpoint Q A C → Par_strict A B Q P.

Lemma l13_1 : ∀ A B C P Q R, ¬Col A B C → is_midpoint P B C → is_midpoint Q A C → is_midpoint R A B
→ ∃ X , ∃ Y , Perp_in R X Y A B ∧ Perp X Y P Q.

Lemma per_lt : ∀ A B C, $A \neq B$ → $C \neq B$ → Per A B C → lt A B A C ∧ lt C B A C.

Lemma cong_perp_conga : ∀ A B C P, Cong A B C B → Perp A C B P → Conga A B P C B P ∧ two_sides B P A C.

Lemma os_out_os : ∀ A B C D C' P , Col A B P → one_side A B C D → out P C C' → one_side A B C' D.

Lemma ts_ts_os : ∀ A B C D, two_sides A B C D → two_sides C D A B → one_side A C B D.

Lemma ts_per_per_ts : ∀ A B C D, two_sides A B C D → Per B C A → Per B D A → two_sides C D A B.

Lemma lea_in_angle : ∀ A B C A' B' C' P, lea A' B' C' A B C → Conga A B P A' B' C' → one_side A B C P
→ InAngle P A B C.

Lemma l13_2_1 : ∀ A B C D E, two_sides A B C D → Per B C A → Per B D A → Col C D E

$\rightarrow$ Perp $A\ E\ C\ D \rightarrow$ Conga $C\ A\ B\ D\ A\ B$
$\rightarrow$ Conga $B\ A\ C\ D\ A\ E \wedge$ Conga $B\ A\ D\ C\ A\ E \wedge$ Bet $C\ E\ D$.

Lemma inangle_one_side : $\forall\ A\ B\ C\ P\ Q$, $\neg$ Col $A\ B\ C \rightarrow \neg$ Col $A\ B\ P \rightarrow \neg$ Col $A\ B\ Q$
$\rightarrow$ InAngle $P\ A\ B\ C \rightarrow$ InAngle $Q\ A\ B\ C$

$\rightarrow$ one_side $A\ B\ P\ Q$.

Lemma inangle_one_side2 : $\forall\ A\ B\ C\ P\ Q$, $\neg$ Col $A\ B\ C \rightarrow \neg$ Col $A\ B\ P \rightarrow \neg$ Col $A\ B\ Q$
$\rightarrow \neg$ Col $C\ B\ P \rightarrow \neg$ Col $C\ B\ Q$
$\rightarrow$ InAngle $P\ A\ B\ C \rightarrow$ InAngle $Q\ A\ B\ C$

$\rightarrow$ one_side $A\ B\ P\ Q \wedge$ one_side $C\ B\ P\ Q$.

Lemma l13_2 : $\forall\ A\ B\ C\ D\ E$, two_sides $A\ B\ C\ D \rightarrow$ Per $B\ C\ A \rightarrow$ Per $B\ D\ A \rightarrow$ Col $C\ D\ E \rightarrow$ Perp $A\ E\ C\ D$
$\rightarrow$ Conga $B\ A\ C\ D\ A\ E \wedge$ Conga $B\ A\ D\ C\ A\ E \wedge$ Bet $C\ E\ D$.

Lemma l13_8 : $\forall\ O\ P\ Q\ U\ V$, $U \neq O \rightarrow V \neq O \rightarrow$Col $O\ P\ Q \rightarrow$ Col $O\ U\ V$
$\rightarrow$ Per $P\ U\ O \rightarrow$ Per $Q\ V\ O \rightarrow$ (out $O\ P\ Q \leftrightarrow$ out $O\ U\ V$).

Definition Perp2 := fun $A\ B\ C\ D\ P \Rightarrow \exists\ X$, $\exists\ Y$, Col $P\ X\ Y \wedge$ Perp $X\ Y\ A\ B \wedge$ Perp $X\ Y\ C\ D$.

Lemma perp2_refl : $\forall\ A\ B\ P$, $A \neq B \rightarrow$ Perp2 $A\ B\ A\ B\ P$.

Lemma perp2_sym : $\forall\ A\ B\ C\ D\ P$, Perp2 $A\ B\ C\ D\ P \rightarrow$ Perp2 $C\ D\ A\ B\ P$.

Lemma perp2_left_comm : $\forall\ A\ B\ C\ D\ P$, Perp2 $A\ B\ C\ D\ P \rightarrow$ Perp2 $B\ A\ C\ D\ P$.

Lemma perp2_right_comm : $\forall\ A\ B\ C\ D\ P$, Perp2 $A\ B\ C\ D\ P \rightarrow$ Perp2 $A\ B\ D\ C\ P$.

Lemma perp2_comm : $\forall\ A\ B\ C\ D\ P$, Perp2 $A\ B\ C\ D\ P \rightarrow$ Perp2 $B\ A\ D\ C\ P$.

Lemma perp2_trans : $\forall\ A\ B\ C\ D\ E\ F\ P$, Perp2 $A\ B\ C\ D\ P \rightarrow$ Perp2 $C\ D\ E\ F\ P \rightarrow$ Perp2 $A\ B\ E\ F\ P$.

Lemma perp2_par : $\forall\ A\ B\ C\ D\ O$, Perp2 $A\ B\ C\ D\ O \rightarrow$ Par $A\ B\ C\ D$.

Lemma perp2_preserves_bet : $\forall\ O\ A\ B\ A'\ B'$, Bet $O\ A\ B \rightarrow$ Col $O\ A'\ B' \rightarrow \neg$Col $O\ A\ A'$ $\rightarrow$
Perp2 $A\ A'\ B\ B'\ O \rightarrow$ Bet $O\ A'\ B'$.

Lemma perp2_perp_in : $\forall\ A\ B\ C\ D\ O$, Perp2 $A\ B\ C\ D\ O \rightarrow \neg$Col $O\ A\ B \wedge \neg$Col $O\ C\ D \rightarrow$ $\exists\ P$, $\exists\ Q$, Col $A\ B\ P \wedge$ Col $C\ D\ Q \wedge$ Col $O\ P\ Q \wedge$ Perp_in $P\ O\ P\ A\ B \wedge$ Perp_in $Q\ O\ Q\ C\ D$.

Lemma perp_vector1 : $\forall\ A\ B\ P$, $A \neq B \rightarrow \exists\ Q$, Perp $A\ B\ P\ Q$.

Lemma par_perp2 : $\forall\ A\ B\ C\ D\ P$, Par $A\ B\ C\ D \rightarrow$ Perp2 $A\ B\ C\ D\ P$.

End L13.

Chapter 18

Library Ch13_2_length

```
Require Export Ch13_1.

Section Length_1.

Context `{MT:Tarski_2D_euclidean}.
Context `{EqDec:EqDecidability Tpoint}.
Context `{InterDec:InterDecidability Tpoint Col}.
```

Pappus Desargues
****************** length

```
Definition lg (A : Tpoint → Tpoint → Prop) := ∃ a , ∃ b , ∀ x y, Cong a b x y ↔ A x y.

Definition long A B := fun x y ⇒ Cong A B x y.

Lemma lg_exists : ∀ A B, ∃ l , lg l ∧ l A B.

Lemma lg_cong : ∀ l A B C D, lg l → l A B → l C D → Cong A B C D.

Lemma lg_cong_lg : ∀ l A B C D, lg l → l A B → Cong A B C D → l C D.

Lemma lg_sym : ∀ l A B, lg l → l A B → l B A.

Lemma ex_points_lg : ∀ l, lg l → ∃ A , ∃ B , l A B.

End Length_1.

Ltac lg_instance l A B :=
  assert(tempo_sg:= ex_points_lg l);
  match goal with
    |H: lg l ⊢ _ ⇒ assert(tempo_H:=H); apply tempo_sg in tempo_H; elim tempo_H;
intros A ; intro tempo_HP; clear tempo_H; elim tempo_HP; intro B; intro; clear
tempo_HP
  end;
  clear tempo_sg.

Section Length_2.

Context `{MT:Tarski_2D_euclidean}.
```

```
Context `{EqDec:EqDecidability Tpoint}.
Context `{InterDec:InterDecidability Tpoint Col}.
Definition is_len := fun A B l ⇒ lg l ∧ l A B.
Lemma is_len_cong : ∀ A B C D l, is_len A B l → is_len C D l → Cong A B C D.
Lemma is_len_cong_is_len : ∀ A B C D l, is_len A B l → Cong A B C D → is_len C D l.
Lemma not_cong_is_len : ∀ A B C D l , ~(Cong A B C D) → is_len A B l → ~(l C D).
Lemma not_cong_is_len1 : ∀ A B C D l , ¬Cong A B C D → is_len A B l → ¬is_len C D l.
Definition lg_null := fun l ⇒ lg l ∧ ∃ A , l A A.
Lemma lg_null_instance : ∀ l A, lg_null l → l A A.
Lemma lg_null_trivial : ∀ l A, lg l → l A A → lg_null l.
Lemma lg_null_dec : ∀ l, lg l → lg_null l ∨ ¬lg_null l.
Lemma ex_point_lg : ∀ l A, lg l → ∃ B , l A B.
Lemma ex_point_lg_out : ∀ l A P, A ≠ P → lg l → ¬lg_null l → ∃ B , l A B ∧ out A B P.
Lemma ex_point_lg_bet : ∀ l A M, lg l → ∃ B : Tpoint, l M B ∧ Bet A M B.
End Length_2.
Ltac lg_instance1 l A B :=
  assert(tempo_sg:= ex_point_lg l);
  match goal with
    |H: lg l ⊢ _ ⇒ assert(tempo_H:=H); apply (tempo_sg A) in tempo_H; ex_elim
tempo_H B; ∃ B
  end;
  clear tempo_sg.
Tactic Notation "soit" ident(A) ident(B) "de" "longueur" ident(l) := lg_instance1 l A B.
Ltac lg_instance2 l A P B :=
  assert(tempo_sg:= ex_point_lg_out l);
  match goal with
    |H: A ≠ P ⊢ _ ⇒
                      match goal with
                        |HP : lg l ⊢ _ ⇒
                                           match goal with
                                             |HQ : ¬lg_null l ⊢ _ ⇒ assert(tempo_HQ:=HQ);
                                                                     apply
(tempo_sg A P H HP) in tempo_HQ;
                                                                     ex_and
tempo_HQ B
                                           end
                      end
  end;
```

```
  clear tempo_sg.
Tactic Notation "soit" ident(B) "sur" "la" "demie" "droite" ident(A) ident(P) "/" "longueur"
ident(A) ident(B) "=" ident(l) := lg_instance2 l A P B.

Section Length_3.

Context `{MT:Tarski_2D_euclidean}.
Context `{EqDec:EqDecidability Tpoint}.
Context `{InterDec:InterDecidability Tpoint Col}.

Lemma ex_points_lg_not_col : ∀ l P, lg l → ¬ lg_null l → ∃ A, ∃ B, l A B ∧ ¬Col A B P.

End Length_3.

Ltac lg_instance_not_col l P A B :=
  assert(tempo_sg:= ex_points_lg_not_col l P);
  match goal with
        |HP : lg l ⊢ _ ⇒ match goal with
                                  |HQ : ¬lg_null l ⊢ _ ⇒ assert(tempo_HQ:=HQ);
                                                              apply (tempo_sg HP)
in tempo_HQ;
                                                              elim tempo_HQ;
                                                              intro A;
                                                              intro tempo_HR;
                                                              elim tempo_HR;
                                                              intro B;
                                                              intro;
                                                              spliter;
                                                              clear tempo_HR tempo_HQ
                              end
  end;
  clear tempo_sg.

Tactic Notation "soit" ident(B) "sur" "la" "demie" "droite" ident(A) ident(P) "/" "longueur"
ident(A) ident(B) "=" ident(l) := lg_instance2 l A P B.

Section Length_4.

Context `{MT:Tarski_2D_euclidean}.
Context `{EqDec:EqDecidability Tpoint}.
Context `{InterDec:InterDecidability Tpoint Col}.

Definition eqL := fun l1 l2 ⇒ lg l1 ∧ lg l2 ∧ ∀ A B, l1 A B ↔ l2 A B.

Notation "l1 =l= l2" := (eqL l1 l2) (at level 80, right associativity).

Lemma ex_eql : ∀ l1 l2, (∃ A , ∃ B, is_len A B l1 ∧ is_len A B l2) → eqL l1 l2.

Lemma all_eql : ∀ A B l1 l2, is_len A B l1 → is_len A B l2 → eqL l1 l2.

Lemma null_len : ∀ A B la lb, is_len A A la → is_len B B lb → eqL la lb.
```

Lemma eql_refl : $\forall$ l, lg l $\rightarrow$ eqL l l.

Lemma eql_sym : $\forall$ $l1$ $l2$, lg $l1$ $\rightarrow$ lg $l2$ $\rightarrow$ eqL $l1$ $l2$ $\rightarrow$ eqL $l2$ $l1$.

Lemma eql_trans : $\forall$ $l1$ $l2$ $l3$, lg $l1$ $\rightarrow$ lg $l2$ $\rightarrow$ lg $l3$ $\rightarrow$ eqL $l1$ $l2$ $\rightarrow$ eqL $l2$ $l3$ $\rightarrow$ eqL $l1$ $l3$.

Lemma ex_lg : $\forall$ A B, $\exists$ l , lg l $\wedge$ l A B.

End Length_4.

Chapter 19

Library Ch13_3_angles

```
Require Export Ch13_2_length.

Section Angles_1.

Context `{MT:Tarski_2D_euclidean}.
Context `{EqDec:EqDecidability Tpoint}.
Context `{InterDec:InterDecidability Tpoint Col}.

Definition ang (A : Tpoint → Tpoint → Tpoint → Prop) := ∃ a , ∃ b , ∃ c , a ≠ b ∧ c ≠ b
∧
                                                              ∀ x y z, Conga a b c x
y z ↔ A x y z.

Definition angle A B C := fun x y z ⇒ Conga A B C x y z.

Lemma ang_exists : ∀ A B C, A ≠ B → C ≠ B → ∃ a , ang a ∧ a A B C.

Lemma ex_points_ang : ∀ a , ang a → ∃ A , ∃ B , ∃ C , a A B C.

End Angles_1.

Ltac ang_instance a A B C :=
  assert(tempo_ang:= ex_points_ang a);
  match goal with
    |H: ang a ⊢ _ ⇒ assert(tempo_H:=H); apply tempo_ang in tempo_H;
                    elim tempo_H; intros A ; intro tempo_HP; clear tempo_H;
                    elim tempo_HP; intro B; intro tempo_HQ ; clear tempo_HP
;
                    elim tempo_HQ; intro C; intro; clear tempo_HQ
  end;
  clear tempo_ang.

Section Angles_2.

Context `{MT:Tarski_2D_euclidean}.
Context `{EqDec:EqDecidability Tpoint}.
```

```
Context `{InterDec:InterDecidability Tpoint Col}.

Lemma ang_conga : ∀ a A B C A' B' C', ang a → a A B C → a A' B' C' → Conga A B C A' B' C'.

Definition is_ang := fun A B C a ⇒ ang a ∧ a A B C.

Lemma is_ang_conga : ∀ A B C A' B' C' a, is_ang A B C a → is_ang A' B' C' a → Conga A B C A' B' C'.

Lemma is_ang_conga_is_ang : ∀ A B C A' B' C' a, is_ang A B C a → Conga A B C A' B' C' → is_ang A' B' C' a.

Lemma not_conga_not_ang : ∀ A B C A' B' C' a , ang a → ~(Conga A B C A' B' C') → a A B C → ~(a A' B' C').

Lemma not_conga_is_ang : ∀ A B C A' B' C' a , ~(Conga A B C A' B' C') → is_ang A B C a → ~(a A' B' C').

Lemma not_cong_is_ang1 : ∀ A B C A' B' C' a , ~(Conga A B C A' B' C') → is_ang A B C a → ~(is_ang A' B' C' a).

Definition eqA' := fun a1 a2 ⇒ ang a1 → ang a2 → ∀ A B C, a1 A B C ↔ a2 A B C.
Definition eqA := fun a1 a2 ⇒ ang a1 ∧ ang a2 ∧ ∀ A B C, a1 A B C ↔ a2 A B C.

Lemma ex_eqa : ∀ a1 a2, (∃ A , ∃ B, ∃ C, is_ang A B C a1 ∧ is_ang A B C a2) → eqA a1 a2.

Lemma all_eqa : ∀ A B C a1 a2, is_ang A B C a1 → is_ang A B C a2 → eqA a1 a2.

Lemma is_ang_distinct : ∀ A B C a , is_ang A B C a → A ≠ B ∧ C ≠ B.

Lemma null_ang : ∀ A B C D a1 a2, is_ang A B A a1 → is_ang C D C a2 → eqA a1 a2.

Lemma flat_ang : ∀ A B C A' B' C' a1 a2, Bet A B C → Bet A' B' C' → is_ang A B C a1 → is_ang A' B' C' a2 → eqA a1 a2.

Lemma ang_distinct: ∀ a A B C, ang a → a A B C → A ≠ B ∧ C ≠ B.

Lemma ex_ang : ∀ A B C, B ≠ A → B ≠ C → ∃ a, ang a ∧ a A B C.

Definition anga (A : Tpoint → Tpoint → Tpoint → Prop) := ∃ a, ∃ b, ∃ c, acute a b c ∧
                                                        ∀ x y z, Conga a b c x
y z ↔ A x y z.
Definition anglea A B C := fun x y z ⇒ Conga A B C x y z.

Lemma anga_exists : ∀ A B C, A ≠ B → C ≠ B → acute A B C → ∃ a, anga a ∧ a A B C.

Lemma anga_is_ang : ∀ a, anga a → ang a.

Lemma ex_points_anga : ∀ a , anga a → ∃ A, ∃ B, ∃ C, a A B C.

End Angles_2.

Ltac anga_instance a A B C :=
  assert(tempo_anga:= ex_points_anga a);
  match goal with
```

|H: anga a ⊢ _ ⇒ assert(tempo_H:=H); apply tempo_anga in tempo_H;
elim tempo_H; intros A ; intro tempo_HP; clear tempo_H;
elim tempo_HP; intro B; intro tempo_HQ ; clear tempo_HP
;
elim tempo_HQ; intro C; intro; clear tempo_HQ
end;
clear tempo_anga.

Section Angles_3.

Context `{MT:**Tarski_2D_euclidean**}.
Context `{EqDec:**EqDecidability** Tpoint}.
Context `{InterDec:**InterDecidability** Tpoint Col}.

Lemma anga_conga : ∀ a A B C A' B' C', anga a → a A B C → a A' B' C' → Conga A B C A' B' C'.

Definition is_anga := fun A B C a ⇒ anga a ∧ a A B C.

Lemma is_anga_to_is_ang : ∀ A B C a, is_anga A B C a → is_ang A B C a.

Lemma is_anga_conga : ∀ A B C A' B' C' a, is_anga A B C a → is_anga A' B' C' a → Conga A B C A' B' C'.

Lemma is_anga_conga_is_anga : ∀ A B C A' B' C' a, is_anga A B C a → Conga A B C A' B' C' → is_anga A' B' C' a.

Lemma not_conga_is_anga : ∀ A B C A' B' C' a , ¬ Conga A B C A' B' C' → is_anga A B C a → ~(a A' B' C').

Lemma not_cong_is_anga1 : ∀ A B C A' B' C' a , ¬Conga A B C A' B' C' → is_anga A B C a → ¬ is_anga A' B' C' a.

Lemma ex_eqaa : ∀ a1 a2, (∃ A , ∃ B, ∃ C, is_anga A B C a1 ∧ is_anga A B C a2) → eqA a1 a2.

Lemma all_eqaa : ∀ A B C a1 a2, is_anga A B C a1 → is_anga A B C a2 → eqA a1 a2.

Lemma is_anga_distinct : ∀ A B C a , is_anga A B C a → A ≠ B ∧ C ≠ B.

Lemma null_anga : ∀ A B C D a1 a2, is_anga A B A a1 → is_anga C D C a2 → eqA a1 a2.

Lemma anga_distinct: ∀ a A B C, anga a → a A B C → A ≠ B ∧ C ≠ B.

Lemma out_is_len_eq : ∀ A B C l, out A B C → is_len A B l → is_len A C l → B = C.

Lemma out_len_eq : ∀ A B C l, lg l → out A B C → l A B → l A C → B = C.

Lemma ex_anga : ∀ A B C, acute A B C → ∃ a, anga a ∧ a A B C.

Definition not_null_ang := fun a ⇒ ang a ∧ ∀ A B C, a A B C → ¬out B A C.
Definition not_flat_ang := fun a ⇒ ang a ∧ ∀ A B C, a A B C → ¬Bet A B C.

Definition not_null_ang' := fun a ⇒ ang a ∧ ∃ A, ∃ B, ∃ C, a A B C ∧ ¬out B A C.
Definition not_flat_ang' := fun a ⇒ ang a ∧ ∃ A, ∃ B, ∃ C, a A B C ∧ ¬Bet A B C.

Definition is_null_ang := fun a ⇒ ang a ∧ ∀ A B C, a A B C → out B A C.

```
Definition is_flat_ang := fun a ⇒ ang a ∧ ∀ A B C, a A B C → Bet A B C.

Definition is_null_ang' := fun a ⇒ ang a ∧ ∃ A, ∃ B, ∃ C, a A B C ∧ out B A C.
Definition is_flat_ang' := fun a ⇒ ang a ∧ ∃ A, ∃ B, ∃ C, a A B C ∧ Bet A B C.

Lemma not_null_ang_ang : ∀ a, not_null_ang a → ang a.

Lemma not_null_ang_def_equiv : ∀ a, not_null_ang a ↔ not_null_ang' a.

Lemma not_flat_ang_def_equiv : ∀ a, not_flat_ang a ↔ not_flat_ang' a.

Definition is_null_anga := fun a ⇒ anga a ∧ ∀ A B C, a A B C → out B A C.
Definition is_null_anga' := fun a ⇒ anga a ∧ ∃ A, ∃ B, ∃ C, a A B C ∧ out B A C.

Definition not_null_anga := fun a ⇒ anga a ∧ ∀ A B C, a A B C → ¬out B A C.

Lemma ang_const : ∀ a A B, ang a → A ≠ B → ∃ C, a A B C.

End Angles_3.

Ltac ang_instance1 a A B C :=
        assert(tempo_ang:= ang_const a A B);
        match goal with
            |H: ang a ⊢ _ ⇒ assert(tempo_H:= H);apply tempo_ang in tempo_H; ex_elim
tempo_H C
        end;
        clear tempo_ang.

Section Angles_4.

Context `{MT:Tarski_2D_euclidean}.
Context `{EqDec:EqDecidability Tpoint}.
Context `{InterDec:InterDecidability Tpoint Col}.

Lemma ang_sym : ∀ a A B C, ang a → a A B C → a C B A.

Lemma ang_not_null_lg : ∀ a l A B C, ang a → lg l → a A B C → l A B → ¬lg_null l.

Lemma ang_distincts : ∀ a A B C, ang a → a A B C → A ≠ B ∧ C ≠ B.

Lemma anga_sym : ∀ a A B C, anga a → a A B C → a C B A.

Lemma anga_not_null_lg : ∀ a l A B C, anga a → lg l → a A B C → l A B → ¬lg_null l.

Lemma anga_distincts : ∀ a A B C, anga a → a A B C → A ≠ B ∧ C ≠ B.

Lemma ang_const_o : ∀ a A B P, ¬Col A B P → ang a → not_null_ang a → not_flat_ang a
→ ∃ C, a A B C ∧ one_side A B C P.

Lemma anga_const : ∀ a A B, anga a → A ≠ B → ∃ C, a A B C.

End Angles_4.

Ltac anga_instance1 a A B C :=
        assert(tempo_anga:= anga_const a A B);
        match goal with
            |H: anga a ⊢ _ ⇒ assert(tempo_H:= H); apply tempo_anga in tempo_H;
ex_elim tempo_H C
```

end;
clear *tempo_anga*.

Section Angles_5.

Context `{MT:**Tarski_2D_euclidean**}.
Context `{$EqDec$:**EqDecidability** Tpoint}.
Context `{$InterDec$:**InterDecidability** Tpoint Col}.

Lemma null_anga_null_anga' : $\forall$ a, is_null_anga $a \leftrightarrow$ is_null_anga' a.

Lemma is_null_anga_out : $\forall$ a A B C, anga $a \rightarrow a$ A B $C \rightarrow$ is_null_anga $a \rightarrow$ out B A C.

Lemma acute_not_bet : $\forall$ A B C, acute A B $C \rightarrow \neg$Bet A B C.

Lemma anga_acute : $\forall$ a A B C , anga $a \rightarrow a$ A B $C \rightarrow$ acute A B C.

Lemma acute_col_out : $\forall$ A B C, acute A B $C \rightarrow$ Col A B $C \rightarrow$ out B A C.

Lemma not_null_not_col : $\forall$ a A B C, anga $a \rightarrow \neg$is_null_anga $a \rightarrow a$ A B $C \rightarrow \neg$Col A B C.

Lemma ang_cong_ang : $\forall$ a A B C A' B' C', ang $a \rightarrow a$ A B $C \rightarrow$ Conga A B C A' B' C' $\rightarrow a$ A' B' C'.

Lemma is_null_ang_out : $\forall$ a A B C, ang $a \rightarrow a$ A B $C \rightarrow$ is_null_ang $a \rightarrow$ out B A C.

Lemma out_null_ang : $\forall$ a A B C, ang $a \rightarrow a$ A B $C \rightarrow$ out B A $C \rightarrow$ is_null_ang a.

Lemma bet_flat_ang : $\forall$ a A B C, ang $a \rightarrow a$ A B $C \rightarrow$ Bet A B $C \rightarrow$ is_flat_ang a.

Lemma out_null_anga : $\forall$ a A B C, anga $a \rightarrow a$ A B $C \rightarrow$ out B A $C \rightarrow$ is_null_anga a.

Lemma anga_not_flat : $\forall$ a, anga $a \rightarrow$ not_flat_ang a.

Lemma anga_const_o : $\forall$ a A B P, $\neg$Col A B $P \rightarrow \neg$ is_null_anga $a \rightarrow$ anga $a \rightarrow \exists$ C , a A B $C \wedge$ one_side A B C P.

End Angles_5.

Chapter 20

Library Ch13_4_cos

```
Require Export Ch13_3_angles.

Ltac anga_instance_o a A B P C :=
        assert(tempo_anga:= anga_const_o a A B P);
        match goal with
           |H: anga a ⊢ _ ⇒ assert(tempo_H:= H); apply tempo_anga in tempo_H;
ex_elim tempo_H C
        end;
        clear tempo_anga.

Section Cosinus.

Context `{MT:Tarski_2D_euclidean}.
Context `{EqDec:EqDecidability Tpoint}.
Context `{InterDec:InterDecidability Tpoint Col}.

Definition lcos := fun lb lc a ⇒ lg lb ∧ lg lc ∧ anga a ∧
                                         (∃ A, ∃ B, ∃ C, (Per C B A ∧ lb A B ∧ lc A C ∧
a B A C)).

Lemma l13_6 : ∀ a lc ld l, lcos lc l a → lcos ld l a → eqL lc ld.

Lemma null_lcos_eql : ∀ lp l a, lcos lp l a → is_null_anga a → eqL l lp.

Lemma eql_lcos_null : ∀ l lp a, lcos l lp a → eqL l lp → is_null_anga a.

Lemma lcos_lg_not_null: ∀ l lp a, lcos l lp a → ¬lg_null l ∧ ¬lg_null lp.

Lemma anga_col_out : ∀ a A B C, anga a → a A B C → Col A B C → out B A C.

Lemma perp_acute_out : ∀ A B C C', acute A B C → Perp A B C C' → Col A B C' → out
B A C'.

Lemma perp_out_acute : ∀ A B C C', out B A C' → Perp A B C C' → Col A B C' → acute
A B C.

Lemma perp_out__acute : ∀ A B C C', Perp A B C C' → Col A B C' → (acute A B C ↔
out B A C').
```

Lemma obtuse_not_acute : $\forall$ A B C, obtuse A B C $\rightarrow$ $\neg$ acute A B C.

Lemma acute_not_obtuse : $\forall$ A B C, acute A B C $\rightarrow$ $\neg$ obtuse A B C.

Lemma perp_obtuse_bet : $\forall$ A B C C', Perp A B C C' $\rightarrow$ Col A B C' $\rightarrow$ obtuse A B C $\rightarrow$ Bet A B C'.

Lemma perp_bet_obtuse : $\forall$ A B C C', $B \neq C'$ $\rightarrow$ Perp A B C C' $\rightarrow$ Col A B C' $\rightarrow$ Bet A B C' $\rightarrow$ obtuse A B C.

Lemma anga_conga_anga : $\forall$ a A B C A' B' C' , anga a $\rightarrow$ a A B C $\rightarrow$ Conga A B C A' B' C' $\rightarrow$ a A' B' C'.

Lemma anga_out_anga : $\forall$ a A B C A' C', anga a $\rightarrow$ a A B C $\rightarrow$ out B A A' $\rightarrow$ out B C C' $\rightarrow$ a A' B C'.

Lemma out_out_anga : $\forall$ a A B C A' B' C', anga a $\rightarrow$ out B A C $\rightarrow$ out B' A' C' $\rightarrow$ a A B C $\rightarrow$ a A' B' C'.

Lemma is_null_all : $\forall$ a A B, $A \neq B$ $\rightarrow$ is_null_anga a $\rightarrow$ a A B A.

Lemma lcos_const0 : $\forall$ l lp a, lcos lp l a $\rightarrow$ is_null_anga a $\rightarrow$
$\exists$ A, $\exists$ B, $\exists$ C, l A B $\wedge$ lp B C $\wedge$ a A B C.

Lemma lcos_const1 : $\forall$ l lp a P, lcos lp l a $\rightarrow$ $\neg$ is_null_anga a $\rightarrow$
$\exists$ A, $\exists$ B, $\exists$ C, $\neg$Col A B P $\wedge$ one_side A B C P $\wedge$ l A B $\wedge$ lp B C $\wedge$ a A B C.

Lemma lcos_const : $\forall$ lp l a, lcos lp l a $\rightarrow$
$\exists$ A, $\exists$ B, $\exists$ C, lp A B $\wedge$ l B C $\wedge$ a A B C.

Lemma lcos_lg_distincts : $\forall$ lp l a A B C, lcos lp l a $\rightarrow$ l A B $\rightarrow$ lp B C $\rightarrow$ a A B C $\rightarrow$ $A \neq B$ $\wedge$ $C \neq B$.

Lemma lcos_const_a : $\forall$ lp l a B, lcos lp l a $\rightarrow$ $\exists$ A, $\exists$ C, l A B $\wedge$ lp B C $\wedge$ a A B C.

Lemma lcos_const_ab : $\forall$ lp l a B A, lcos lp l a $\rightarrow$ l A B $\rightarrow$ $\exists$ C, lp B C $\wedge$ a A B C.

Lemma lcos_const_cb : $\forall$ lp l a B C, lcos lp l a $\rightarrow$ lp B C $\rightarrow$ $\exists$ A, l A B $\wedge$ a A B C.

Lemma lcos_lg_anga : $\forall$ l lp a, lcos lp l a $\rightarrow$ lcos lp l a $\wedge$ lg l $\wedge$ lg lp $\wedge$ anga a.

Lemma eql_lg : $\forall$ $l1$ $l2$, eqL $l1$ $l2$ $\rightarrow$ lg $l1$ $\wedge$ lg $l2$.

Lemma lcos_eql_lcos : $\forall$ $lp1$ $l1$ $lp2$ $l2$ a, eqL $lp1$ $lp2$ $\rightarrow$ eqL $l1$ $l2$ $\rightarrow$ lcos $lp1$ $l1$ a $\rightarrow$ lcos $lp2$ $l2$ a.

Lemma ang_not_lg_null : $\forall$ a la lc A B C, lg la $\rightarrow$ lg lc $\rightarrow$ ang a $\rightarrow$
la A B $\rightarrow$ lc C B $\rightarrow$ a A B C $\rightarrow$ $\neg$ lg_null la $\wedge$ $\neg$ lg_null lc.

Lemma anga_not_lg_null : $\forall$ a la lc A B C, lg la $\rightarrow$ lg lc $\rightarrow$
anga a $\rightarrow$ la A B $\rightarrow$ lc C B $\rightarrow$ a A B C $\rightarrow$ $\neg$ lg_null la $\wedge$ $\neg$ lg_null lc.

Lemma lcos_not_lg_null : $\forall$ lp l a, lcos lp l a $\rightarrow$ $\neg$ lg_null lp.

Lemma lcos_const_o : $\forall$ lp l a A B P, $\neg$ Col A B P $\rightarrow$ $\neg$ is_null_anga a $\rightarrow$ lg l $\rightarrow$ lg lp $\rightarrow$
anga a $\rightarrow$ l A B $\rightarrow$ lcos lp l a $\rightarrow$
$\exists$ C, one_side A B C P $\wedge$ a A B C $\wedge$ lp B C.

Lemma anga_col_null : ∀ a A B C, anga a → a A B C → Col A B C → out B A C ∧ is_null_anga a.

Lemma flat_not_acute : ∀ A B C, Bet A B C → ¬ acute A B C.

Lemma acute_comp_not_acute : ∀ A B C D, Bet A B C → acute A B D → ¬ acute C B D.

Lemma lcos_per : ∀ A B C lp l a, anga a → lg l → lg lp →
lcos lp l a → l A C → lp A B → a B A C → Per A B C.

Lemma is_null_anga_dec : ∀ a, anga a → is_null_anga a ∨ ¬is_null_anga a.

Lemma lcos_lg : ∀ a lp l A B C, lcos lp l a → Perp A B B C → a B A C → l A C → lp A B.

Lemma l13_7 : ∀ a b l la lb lab lba, lcos la l a → lcos lb l b →
lcos lab la b → lcos lba lb a → eqL lab lba.

Lemma out_acute : ∀ A B C, out B A C → acute A B C.

Lemma perp_acute : ∀ A B C P, Col A C P → Perp_in P B P A C → acute A B P.

Lemma null_lcos : ∀ l a,lg l → ¬lg_null l → is_null_anga a → lcos l l a.

Lemma perp2_preserves_bet1 : ∀ O A B A' B', Bet A O B → Col O A' B' → ¬Col O A A' →
Perp2 A A' B B' O → Bet A' O B'.

Lemma eqA_preserves_anga : ∀ a b, anga a → ang b → eqA a b → anga b.

End Cosinus.

Chapter 21

Library Ch13_5_Pappus_Pascal

Require Export Ch13_4_cos.

Section Pappus_Pascal.

Context `{MT:**Tarski_2D_euclidean**}.
Context `{EqDec:**EqDecidability** Tpoint}.
Context `{InterDec:**InterDecidability** Tpoint Col}.

Lemma l13_10_aux1 : ∀ O A B P Q la lb lp lq,
Col O A B → Col O P Q → Perp O P P A → Perp O Q Q B →
lg la → lg lb → lg lp → lg lq → la O A → lb O B → lp O P → lq O Q →
∃ a , anga a ∧ lcos lp la a ∧ lcos lq lb a.

Lemma acute_trivial : ∀ A B, $A \neq B$ → acute A B A.

Lemma lcos_eqa_lcos : ∀ lp l a b, lcos lp l a → eqA a b → lcos lp l b.

Lemma l13_10_aux2 : ∀ O A B la lla lb llb,
Col O A B → lg la → lg lla → lg lb → lg llb → la O A → lla O A → lb O B → llb O B
→
$A \neq O$ → $B \neq O$ → ∃ a , anga a ∧ lcos lla la a ∧ lcos llb lb a.

Lemma acute_not_per : ∀ A B C, acute A B C → ¬ Per A B C.

Lemma lcos_exists : ∀ l a, anga a → lg l → ¬lg_null l → ∃ lp , lcos lp l a.

Definition lcos_eq := fun la a lb b ⇒ ∃ lp , lcos lp la a ∧ lcos lp lb b.

Lemma lcos_eq_refl : ∀ la a, lg la → ¬lg_null la → anga a → lcos_eq la a la a.

Lemma lcos_eq_sym : ∀ la a lb b, lcos_eq la a lb b → lcos_eq lb b la a.

Lemma lcos_eq_trans : ∀ la a lb b lc c, lcos_eq la a lb b → lcos_eq lb b lc c → lcos_eq la a lc
c.

Definition lcos2 := fun lp l a b ⇒ ∃ la , lcos la l a ∧ lcos lp la b.

Lemma lcos2_comm : ∀ lp l a b, lcos2 lp l a b → lcos2 lp l b a.

Lemma lcos2_exists : ∀ l a b, lg l → ¬lg_null l → anga a → anga b → ∃ lp , lcos2 lp l a b.

Lemma lcos2_exists' : ∀ l a b, lg l → ¬lg_null l →anga a → anga b →
∃ la , ∃ lab , lcos la l a ∧ lcos lab la b.

Definition lcos2_eq := fun l1 a b l2 c d ⇒ ∃ lp , lcos2 lp l1 a b ∧ lcos2 lp l2 c d.

Lemma lcos2_eq_refl : ∀ l a b, lg l → ¬lg_null l → anga a → anga b → lcos2_eq l a b l a b.

Lemma lcos2_eq_sym : ∀ l1 a b l2 c d, lcos2_eq l1 a b l2 c d → lcos2_eq l2 c d l1 a b.

Lemma lcos2_unicity: ∀ l l1 l2 a b, lcos2 l1 l a b → lcos2 l2 l a b → eqL l1 l2.

Lemma lcos2_eql_lcos2 : ∀ lla llb la lb a b, lcos2 la lla a b → eqL lla llb → eqL la lb → lcos2 lb llb a b.

Lemma lcos2_lg_anga : ∀ lp l a b, lcos2 lp l a b → lcos2 lp l a b ∧ lg lp ∧ lg l ∧ anga a ∧ anga b.

Lemma lcos2_eq_trans : ∀ l1 a b l2 c d l3 e f, lcos2_eq l1 a b l2 c d → lcos2_eq l2 c d l3 e f
→ lcos2_eq l1 a b l3 e f.

Lemma lcos_eq_lcos2_eq : ∀ la lb a b c, anga c → lcos_eq la a lb b → lcos2_eq la a c lb b c.

Lemma lcos2_lg_not_null: ∀ lp l a b, lcos2 lp l a b → ¬lg_null l ∧ ¬lg_null lp.

Definition lcos3 := fun lp l a b c ⇒ ∃ la , ∃ lab , lcos la l a ∧ lcos lab la b ∧ lcos lp lab c.

Lemma lcos3_lcos_1_2 : ∀ lp l a b c, lcos3 lp l a b c ↔ ∃ la , lcos la l a ∧ lcos2 lp la b c.

Lemma lcos3_lcos_2_1 : ∀ lp l a b c, lcos3 lp l a b c ↔ ∃ lab , lcos2 lab l a b ∧ lcos lp lab c.

Lemma lcos3_permut3 : ∀ lp l a b c, lcos3 lp l a b c → lcos3 lp l b a c.

Lemma lcos3_permut1 : ∀ lp l a b c, lcos3 lp l a b c → lcos3 lp l a c b.

Lemma lcos3_permut2 : ∀ lp l a b c, lcos3 lp l a b c → lcos3 lp l c b a.

Lemma lcos3_exists : ∀ l a b c, lg l → ¬lg_null l → anga a → anga b → anga c →
∃ lp , lcos3 lp l a b c.

Definition lcos3_eq := fun l1 a b c l2 d e f ⇒ ∃ lp , lcos3 lp l1 a b c ∧ lcos3 lp l2 d e f.

Lemma lcos3_eq_refl : ∀ l a b c, lg l → ¬lg_null l → anga a → anga b → anga c → lcos3_eq l a b c l a b c.

Lemma lcos3_eq_sym : ∀ l1 a b c l2 d e f, lcos3_eq l1 a b c l2 d e f → lcos3_eq l2 d e f l1 a b c.

Lemma lcos3_unicity: ∀ l l1 l2 a b c, lcos3 l1 l a b c → lcos3 l2 l a b c → eqL l1 l2.

Lemma lcos3_eql_lcos3 : ∀ lla llb la lb a b c, lcos3 la lla a b c → eqL lla llb → eqL la lb → lcos3 lb llb a b c.

Lemma lcos3_lg_anga : ∀ lp l a b c, lcos3 lp l a b c → lcos3 lp l a b c ∧ lg lp ∧ lg l ∧ anga a ∧ anga b ∧ anga c.

Lemma lcos3_lg_not_null: ∀ lp l a b c, lcos3 lp l a b c → ¬lg_null l ∧ ¬lg_null lp.

Lemma lcos3_eq_trans : ∀ l1 a b c l2 d e f l3 g h i,
lcos3_eq l1 a b c l2 d e f → lcos3_eq l2 d e f l3 g h i → lcos3_eq l1 a b c l3 g h i.

Lemma lcos_eq_lcos3_eq : $\forall$ la lb a b c d, anga c $\rightarrow$ anga d $\rightarrow$ lcos_eq la a lb b $\rightarrow$ lcos3_eq la a c d lb b c d.

Lemma lcos2_eq_lcos3_eq : $\forall$ la lb a b c d e, anga e $\rightarrow$ lcos2_eq la a b lb c d $\rightarrow$ lcos3_eq la a b e lb c d e.

Lemma l13_6_bis : $\forall$ lp $l1$ $l2$ a, lcos lp $l1$ a $\rightarrow$ lcos lp $l2$ a $\rightarrow$ eqL $l1$ $l2$.

Lemma lcos3_lcos2 : $\forall$ $l1$ $l2$ a b c d n, lcos3_eq $l1$ a b n $l2$ c d n $\rightarrow$ lcos2_eq $l1$ a b $l2$ c d.

Lemma lcos2_lcos : $\forall$ $l1$ $l2$ a b c, lcos2_eq $l1$ a c $l2$ b c $\rightarrow$ lcos_eq $l1$ a $l2$ b.

Lemma lcos_per_anga : $\forall$ O A P la lp a, lcos lp la a $\rightarrow$ la O A $\rightarrow$ lp O P $\rightarrow$ Per A P O $\rightarrow$ a A O P.

Lemma lcos_lcos_col : $\forall$ la lb lp a b O A B P,
lcos lp la a $\rightarrow$ lcos lp lb b $\rightarrow$ la O A $\rightarrow$ lb O B $\rightarrow$ lp O P $\rightarrow$ a A O P $\rightarrow$ b B O P $\rightarrow$ Col A B P.

Lemma l13_10_aux3 : $\forall$ A B C A' B' C' O,
$\neg$ Col O A A' $\rightarrow$
$B \neq O$ $\rightarrow$ $C \neq O$ $\rightarrow$ Col O A B $\rightarrow$ Col O B C $\rightarrow$
$B' \neq O$ $\rightarrow$ $C' \neq O$ $\rightarrow$
Col O A' B' $\rightarrow$ Col O B' C' $\rightarrow$ Perp2 B C' C B' O $\rightarrow$ Perp2 C A' A C' O $\rightarrow$
Bet A O B $\rightarrow$ Bet A' O B'.

Lemma l13_10_aux4 : $\forall$ A B C A' B' C' O,
$\neg$ Col O A A' $\rightarrow$ $B \neq O$ $\rightarrow$ $C \neq O$ $\rightarrow$ Col O A B $\rightarrow$ Col O B C $\rightarrow$ $B' \neq O$ $\rightarrow$ $C' \neq O$ $\rightarrow$
Col O A' B' $\rightarrow$ Col O B' C' $\rightarrow$ Perp2 B C' C B' O $\rightarrow$ Perp2 C A' A C' O $\rightarrow$ Bet O A B $\rightarrow$
out O A' B'.

Lemma l13_10_aux5 : $\forall$ A B C A' B' C' O,
$\neg$ Col O A A' $\rightarrow$ $B \neq O$ $\rightarrow$ $C \neq O$ $\rightarrow$ Col O A B $\rightarrow$ Col O B C $\rightarrow$
$B' \neq O$ $\rightarrow$ $C' \neq O$ $\rightarrow$ Col O A' B' $\rightarrow$ Col O B' C' $\rightarrow$
Perp2 B C' C B' O $\rightarrow$ Perp2 C A' A C' O $\rightarrow$ out O A B $\rightarrow$
out O A' B'.

Lemma per_per_perp : $\forall$ A B X Y,
$A \neq B$ $\rightarrow$ $X \neq Y$ $\rightarrow$
$(B \neq X \lor B \neq Y)$ $\rightarrow$ Per A B X $\rightarrow$ Per A B Y $\rightarrow$
Perp A B X Y.

Lemma l13_10 : $\forall$ A B C A' B' C' O,
$\neg$ Col O A A' $\rightarrow$ $B \neq O$ $\rightarrow$ $C \neq O$ $\rightarrow$
Col O A B $\rightarrow$ Col O B C $\rightarrow$
$B' \neq O$ $\rightarrow$ $C' \neq O$ $\rightarrow$
Col O A' B' $\rightarrow$ Col O B' C' $\rightarrow$
Perp2 B C' C B' O $\rightarrow$ Perp2 C A' A C' O $\rightarrow$

Perp2 A B' B A' O.

Lemma l13_11 : $\forall$ A B C A' B' C' O,
¬ Col O A A' $\rightarrow$
$B \neq O \rightarrow C \neq O \rightarrow$
Col O A B $\rightarrow$ Col O B C $\rightarrow$
$B' \neq O \rightarrow C' \neq O \rightarrow$
Col O A' B' $\rightarrow$ Col O B' C' $\rightarrow$ Par B C' C B' $\rightarrow$ Par C A' A C' $\rightarrow$
Par A B' B A'.

Lemma l13_14 : $\forall$ O A B C O' A' B' C',
Par_strict O A O' A' $\rightarrow$ Col O A B $\rightarrow$ Col O B C $\rightarrow$ Col O A C $\rightarrow$
Col O' A' B' $\rightarrow$ Col O' B' C' $\rightarrow$ Col O' A' C' $\rightarrow$
Par A C' A' C $\rightarrow$ Par B C' B' C $\rightarrow$ Par A B' A' B.

End Pappus_Pascal.

Chapter 22

Library Ch13_6_Desargues_Hessenberg

```
Require Export Ch13_5_Pappus_Pascal.
Section Desargues_Hessenberg.
Context `{MT:Tarski_2D_euclidean}.
Context `{EqDec:EqDecidability Tpoint}.
Context `{InterDec:InterDecidability Tpoint Col}.
Lemma l13_15_1 : ∀ A B C A' B' C' O ,
  ¬ Col A B C → ¬ Par O B A C →
  Par_strict A B A' B' → Par_strict A C A' C'→
  Col O A A' → Col O B B' → Col O C C' →
  Par B C B' C'.
Lemma l13_15_2_aux : ∀ A B C A' B' C' O , ¬Col A B C
                                         → ¬Par O A B C
                                         → Par O B A C
                                         → Par_strict A B A' B'
                                         → Par_strict A C A' C'
                                         → Col O A A' → Col O B B' → Col O C
C'
                                         → Par B C B' C'.
Lemma l13_15_2 : ∀ A B C A' B' C' O , ¬Col A B C
                                         → Par O B A C
                                         → Par_strict A B A' B'
                                         → Par_strict A C A' C'
                                         → Col O A A' → Col O B B' → Col O C
C'
                                         → Par B C B' C'.
Lemma l13_15 : ∀ A B C A' B' C' O , ¬Col A B C
```

```
→ Par_strict A B A' B'
→ Par_strict A C A' C'
→ Col O A A' → Col O B B' → Col O C
C'
→ Par B C B' C'.
Lemma l13_15_par : ∀ A B C A' B' C', ¬Col A B C
→ Par_strict A B A' B'
→ Par_strict A C A' C'
→ Par A A' B B'
→ Par A A' C C'
→ Par B C B' C'.
Lemma l13_18_2 : ∀ A B C A' B' C' O, ¬Col A B C
→ Par_strict A B A' B'
→ Par_strict A C A' C'
→ (Par_strict B C B' C' ∧ Col O A A' ∧ Col O B B'
→ Col O C C').
Lemma l13_18_3 : ∀ A B C A' B' C', ¬Col A B C
→ Par_strict A B A' B'
→ Par_strict A C A' C'
→ (Par_strict B C B' C' ∧ Par A A' B B')
→ (Par C C' A A' ∧ Par C C' B B').
Lemma l13_18 : ∀ A B C A' B' C' O, ¬Col A B C ∧ Par_strict A B A' B' ∧ Par_strict A
C A' C'
→(Par_strict B C B' C' ∧ Col O A A' ∧ Col
O B B' → Col O C C')
∧ ((Par_strict B C B' C' ∧ Par A A' B B')
→ (Par C C' A A' ∧ Par C C' B B'))
∧ (Par A A' B B' ∧ Par A A' C C' → Par
B C B' C').
Lemma l13_19_aux : ∀ A B C D A' B' C' D' O, ¬Col O A B → A ≠ A' → A ≠ C
→ O ≠ A → O ≠ A' → O ≠ C → O ≠ C'
→ O ≠ B → O ≠ B' → O ≠ D → O ≠ D'
→ Col O A C → Col O A A' → Col O A
C'
→ Col O B D → Col O B B' → Col O B
D'
→ ¬Par A B C D
→ Par A B A' B' → Par A D A' D' → Par
B C B' C'
→ Par C D C' D'.
Lemma l13_19 : ∀ A B C D A' B' C' D' O, ¬Col O A B
```

$\rightarrow O \neq A \rightarrow O \neq A' \rightarrow O \neq C \rightarrow O \neq C'$
$\rightarrow O \neq B \rightarrow O \neq B' \rightarrow O \neq D \rightarrow O \neq D'$
$\rightarrow$ Col $O\ A\ C \rightarrow$ Col $O\ A\ A' \rightarrow$ Col $O\ A$
C'
$\rightarrow$ Col $O\ B\ D \rightarrow$ Col $O\ B\ B' \rightarrow$ Col $O\ B$
D'
$\rightarrow$ Par $A\ B\ A'\ B' \rightarrow$ Par $A\ D\ A'\ D' \rightarrow$ Par
$B\ C\ B'\ C'$
$\rightarrow$ Par $C\ D\ C'\ D'$.

Lemma l13_19_par_aux : $\forall\ A\ B\ C\ D\ A'\ B'\ C'\ D'\ X\ Y$,
$X \neq A \rightarrow X \neq A' \rightarrow X \neq C \rightarrow X \neq$
C'
$\rightarrow Y \neq B \rightarrow Y \neq B' \rightarrow Y \neq D \rightarrow Y \neq$
D'
$\rightarrow$ Col $X\ A\ C \rightarrow$ Col $X\ A\ A' \rightarrow$ Col $X\ A$
C'
$\rightarrow$ Col $Y\ B\ D \rightarrow$ Col $Y\ B\ B' \rightarrow$ Col $Y\ B$
D'
$\rightarrow A \neq C \rightarrow B \neq D \rightarrow A \neq A'$
$\rightarrow$ Par_strict $X\ A\ Y\ B$
$\rightarrow \neg$Par $A\ B\ C\ D$
$\rightarrow$ Par $A\ B\ A'\ B' \rightarrow$ Par $A\ D\ A'\ D' \rightarrow$ Par
$B\ C\ B'\ C'$
$\rightarrow$ Par $C\ D\ C'\ D'$.

Lemma l13_19_par : $\forall\ A\ B\ C\ D\ A'\ B'\ C'\ D'\ X\ Y$,
$X \neq A \rightarrow X \neq A' \rightarrow X \neq C \rightarrow X \neq C' \rightarrow Y \neq B \rightarrow Y \neq B' \rightarrow Y \neq D \rightarrow Y \neq D' \rightarrow$
Col $X\ A\ C \rightarrow$ Col $X\ A\ A' \rightarrow$ Col $X\ A\ C' \rightarrow$ Col $Y\ B\ D \rightarrow$ Col $Y\ B\ B' \rightarrow$ Col $Y\ B\ D' \rightarrow$
Par_strict $X\ A\ Y\ B \rightarrow$ Par $A\ B\ A'\ B' \rightarrow$ Par $A\ D\ A'\ D' \rightarrow$ Par $B\ C\ B'\ C' \rightarrow$
Par $C\ D\ C'\ D'$.

End Desargues_Hessenberg.

Chapter 23

Library quadrilaterals

Require Export Ch12_parallel.

Section Quadrilateral.

Context ‘{MT:**Tarski_2D_euclidean**}.
Context ‘{$EqDec$:**EqDecidability** Tpoint}.

Lemma cong_identity_inv :
 ∀ A B C, $A \neq B$ → ¬ Cong A B C C.

Lemma midpoint_midpoint_col : ∀ A B A' B' M,
 $A \neq B$ →
 is_midpoint M A A' → is_midpoint M B B' →
 Col A B B' →
 $A' \neq B'$ ∧ Col A A' B' ∧ Col B A' B'.

Lemma midpoint_par :
 ∀ A B A' B' M,
 $A \neq B$ →
 is_midpoint M A A' →
 is_midpoint M B B' →
 Par A B A' B'.

Lemma midpoint_par_strict :
 ∀ A B A' B' M,
 $A \neq B$ →
 ¬ Col A B B' →
 is_midpoint M A A' →
 is_midpoint M B B' →
 Par_strict A B A' B'.

Lemma le_left_comm : ∀ A B C D, le A B C D → le B A C D.

Lemma le_right_comm : ∀ A B C D, le A B C D → le A B D C.

Lemma le_comm :

∀ A B C D, le A B C D → le B A D C.

Lemma le_cong_le :
∀ A B C A' B' C',
Bet A B C →
Bet A' B' C' →
le A B A' B' →
Cong B C B' C' →
le A C A' C'.

Lemma cong_le_le :
∀ A B C A' B' C',
Bet A B C →
Bet A' B' C' →
le B C B' C' →
Cong A B A' B' →
le A C A' C'.

Lemma bet3_cong3_bet : ∀ A B C D D', $A \neq B$ → $A \neq C$ → $A \neq D$ → Bet D A C → Bet A C B → Bet D C D' → Cong A B C D → Cong A D B C → Cong D C C D' → Bet C B D'.

Lemma bet_le_le :
∀ A B C A' B' C',
Bet A B C →
Bet A' B' C' →
le A B A' B' →
le B C B' C' →
le A C A' C'.

Lemma bet_double_bet :
∀ A B C B' C',
is_midpoint B' A B →
is_midpoint C' A C →
Bet A B' C' →
Bet A B C.

Lemma bet_half_bet :
∀ A B C B' C',
Bet A B C →
is_midpoint B' A B →
is_midpoint C' A C →
Bet A B' C'.

Lemma midpoint_preserves_bet :
∀ A B C B' C',
is_midpoint B' A B →
is_midpoint C' A C →

(Bet A B C $\leftrightarrow$ Bet A B' C').

Lemma symmetry_preseves_bet1 :
 $\forall$ A B M A' B',
  is_midpoint M A A' $\rightarrow$
  is_midpoint M B B' $\rightarrow$
  Bet M A B $\rightarrow$
  Bet M A' B'.

Lemma symmetry_preseves_bet2 :
 $\forall$ A B M A' B',
  is_midpoint M A A' $\rightarrow$
  is_midpoint M B B' $\rightarrow$
  Bet M A' B' $\rightarrow$
  Bet M A B.

Lemma symmetry_preserves_bet :
 $\forall$ A B M A' B',
  is_midpoint M A A' $\rightarrow$
  is_midpoint M B B' $\rightarrow$
 (Bet M A' B' $\leftrightarrow$ Bet M A B).

Lemma bet_cong_bet :
 $\forall$ A B C D,
  $A \neq B$ $\rightarrow$
  Bet A B C $\rightarrow$
  Bet A B D $\rightarrow$
  Cong A D B C $\rightarrow$
  Bet B D C.

Lemma col_cong_mid :
 $\forall$ A B A' B',
  Par A B A' B' $\rightarrow$
  $\neg$ Par_strict A B A' B' $\rightarrow$
  Cong A B A' B' $\rightarrow$
  $\exists$ M , is_midpoint M A A' $\wedge$ is_midpoint M B B' $\vee$
   is_midpoint M A B' $\wedge$ is_midpoint M B A'.

Lemma mid_par_cong1 :
 $\forall$ A B A' B' M,
  $A \neq B$ $\rightarrow$
  is_midpoint M A A' $\rightarrow$
  is_midpoint M B B' $\rightarrow$
  Cong A B A' B' $\wedge$ Par A B A' B'.

Lemma mid_par_cong2 :
 $\forall$ A B A' B' M,
  $A \neq B$' $\rightarrow$

is_midpoint M A A' $\rightarrow$
is_midpoint M B B' $\rightarrow$
Cong A B' A' B $\wedge$ Par A B' A' B.

Lemma mid_par_cong :
∀ A B A' B' M,
$A \neq B \rightarrow A \neq B' \rightarrow$
is_midpoint M A A' $\rightarrow$
is_midpoint M B B' $\rightarrow$
Cong A B A' B' $\wedge$ Cong A B' A' B $\wedge$ Par A B A' B' $\wedge$ Par A B' A' B.

Parallelogram

Definition Parallelogram_strict := fun A B A' B' $\Rightarrow$ two_sides A A' B B' $\wedge$ Par A B A' B' $\wedge$ Cong A B A' B'.

Definition Parallelogram_flat := fun A B A' B' $\Rightarrow$ Col A B A' $\wedge$ Col A B B'
$\wedge$ Cong A B A' B' $\wedge$ Cong A B'
A' B
$\wedge$ ($A \neq A' \vee B \neq B'$).

Definition Parallelogram := fun A B A' B' $\Rightarrow$ Parallelogram_strict A B A' B' $\vee$ Parallelogram_flat A B A' B'.

Lemma Parallelogram_strict_Parallelogram :
∀ A B C D,
Parallelogram_strict A B C D $\rightarrow$ Parallelogram A B C D.

Lemma plgf_permut :
∀ A B C D,
Parallelogram_flat A B C D $\rightarrow$
Parallelogram_flat B C D A.

Lemma plgf_sym :
∀ A B C D,
Parallelogram_flat A B C D $\rightarrow$
Parallelogram_flat C D A B.

Lemma plgf_irreflexive :
∀ A B,
¬ Parallelogram_flat A B A B.

Lemma plgs_irreflexive :
∀ A B,
¬ Parallelogram_strict A B A B.

Lemma plg_irreflexive :
∀ A B,
¬ Parallelogram A B A B.

Lemma plgf_mid :

$\forall A\ B\ C\ D,$
Parallelogram_flat $A\ B\ C\ D \rightarrow$
$\exists\ M,$ is_midpoint $M\ A\ C \wedge$ is_midpoint $M\ B\ D.$

Lemma mid_plgs :
$\forall A\ B\ C\ D\ M,$
$\neg$ Col $A\ B\ C \rightarrow$
is_midpoint $M\ A\ C \rightarrow$ is_midpoint $M\ B\ D \rightarrow$
Parallelogram_strict $A\ B\ C\ D.$

Lemma mid_plgf_aux :
$\forall A\ B\ C\ D\ M,$
$A \neq C \rightarrow$
Col $A\ B\ C \rightarrow$
is_midpoint $M\ A\ C \rightarrow$ is_midpoint $M\ B\ D \rightarrow$
Parallelogram_flat $A\ B\ C\ D.$

Lemma mid_plgf :
$\forall A\ B\ C\ D\ M,$
$(A \neq C \vee B \neq D) \rightarrow$
Col $A\ B\ C \rightarrow$
is_midpoint $M\ A\ C \rightarrow$ is_midpoint $M\ B\ D \rightarrow$
Parallelogram_flat $A\ B\ C\ D.$

Lemma mid_plg :
$\forall A\ B\ C\ D\ M,$
$(A \neq C \vee B \neq D) \rightarrow$
is_midpoint $M\ A\ C \rightarrow$ is_midpoint $M\ B\ D \rightarrow$
Parallelogram $A\ B\ C\ D.$

Lemma mid_plg_1 :
$\forall A\ B\ C\ D\ M,$
$A \neq C \rightarrow$
is_midpoint $M\ A\ C \rightarrow$ is_midpoint $M\ B\ D \rightarrow$
Parallelogram $A\ B\ C\ D.$

Lemma mid_plg_2 :
$\forall A\ B\ C\ D\ M,$
$B \neq D \rightarrow$
is_midpoint $M\ A\ C \rightarrow$ is_midpoint $M\ B\ D \rightarrow$
Parallelogram $A\ B\ C\ D.$

Lemma midpoint_cong_unicity :
$\forall A\ B\ C\ D\ M,$
Col $A\ B\ C \rightarrow$
is_midpoint $M\ A\ B \wedge$ is_midpoint $M\ C\ D \rightarrow$
Cong $A\ B\ C\ D \rightarrow$
$A = C \wedge B = D \vee A = D \wedge B = C.$

Lemma plgf_not_comm :
 $\forall$ A B C D, $A \neq B \rightarrow$
  Parallelogram_flat A B C $D \rightarrow$
  $\neg$ Parallelogram_flat A B D C $\wedge$ $\neg$ Parallelogram_flat B A C D.

Lemma plgf_cong :
 $\forall$ A B C D,
  Parallelogram_flat A B C $D \rightarrow$
  Cong A B C D $\wedge$ Cong A D B C.

Definition Plg A B C D := $(A \neq C \vee B \neq D) \wedge \exists M$, is_midpoint M A C $\wedge$ is_midpoint M B D.

Lemma plg_to_parallelogram : $\forall$ A B C D, Plg A B C D $\rightarrow$ Parallelogram A B C D.

Lemma plgs_one_side :
 $\forall$ A B C D,
 Parallelogram_strict A B C $D \rightarrow$
 one_side A B C D $\wedge$ one_side C D A B.

Lemma parallelogram_strict_not_col : $\forall$ A B C D,
 Parallelogram_strict A B C $D \rightarrow$
 $\neg$ Col A B C.

Rhombus

Definition Rhombus := fun A B C D $\Rightarrow$ Plg A B C D $\wedge$ Cong A B B C.

Lemma Rhombus_Plg : $\forall$ A B C D, Rhombus A B C D $\rightarrow$ Plg A B C D.

Lemma ex_col3 : $\forall$ A B C, Col A B C $\rightarrow \exists D$, Col A B D $\wedge$ $A \neq D \wedge B \neq D \wedge C \neq D$.

Rectangle

Definition Rectangle := fun A B C D $\Rightarrow$ Plg A B C D $\wedge$ Cong A C B D.

Lemma Rectangle_Plg : $\forall$ A B C D,
  Rectangle A B C $D \rightarrow$
  Plg A B C D.

Lemma Rectangle_Parallelogram : $\forall$ A B C D,
  Rectangle A B C $D \rightarrow$
  Parallelogram A B C D.

Lemma plg_cong_rectangle :
 $\forall$ A B C D,
  Plg A B C $D \rightarrow$
  Cong A C B $D \rightarrow$
  Rectangle A B C D.

Lemma plg_trivial : $\forall$ A B, $A \neq B \rightarrow$ Parallelogram A B B A.

Lemma plg_trivial1 : $\forall$ A B, $A \neq B \rightarrow$ Parallelogram A A B B.

Lemma col_not_plgs : $\forall$ A B C D, Col A B C $\rightarrow$ $\neg$Parallelogram_strict A B C D.

Lemma plg_col_plgf : $\forall$ A B C D, Col A B C $\rightarrow$ Parallelogram A B C D $\rightarrow$ Parallelogram_flat A B C D.

Lemma col_cong_bet : $\forall$ A B C D, Col A B D $\rightarrow$ Cong A B C D $\rightarrow$ Bet A C B $\rightarrow$ Bet C A D $\vee$ Bet C B D.

Lemma col_cong2_bet1 : $\forall$ A B C D, Col A B D $\rightarrow$ Bet A C B $\rightarrow$ Cong A B C D $\rightarrow$ Cong A C B D $\rightarrow$ Bet C B D.

Lemma col_cong2_bet2 : $\forall$ A B C D, Col A B D $\rightarrow$ Bet A C B $\rightarrow$ Cong A B C D $\rightarrow$ Cong A D B C $\rightarrow$ Bet C A D.

Lemma col_cong2_bet3 : $\forall$ A B C D, Col A B D $\rightarrow$ Bet A B C $\rightarrow$ Cong A B C D $\rightarrow$ Cong A C B D $\rightarrow$ Bet B C D.

Lemma col_cong2_bet4 : $\forall$ A B C D, Col A B C $\rightarrow$ Bet A B D $\rightarrow$ Cong A B C D $\rightarrow$ Cong A D B C $\rightarrow$ Bet B D C.

Lemma col_bet2_cong1 : $\forall$ A B C D, Col A B D $\rightarrow$ Bet A C B $\rightarrow$ Cong A B C D $\rightarrow$ Bet C B D $\rightarrow$ Cong A C D B.

Lemma col_bet2_cong2 : $\forall$ A B C D, Col A B D $\rightarrow$ Bet A C B $\rightarrow$ Cong A B C D $\rightarrow$ Bet C A D $\rightarrow$ Cong D A B C.

Lemma plg_bet1 : $\forall$ A B C D, Parallelogram A B C D $\rightarrow$ Bet A C B $\rightarrow$ Bet D A C.

Lemma plgf_trivial1 : $\forall$ A B, $A \neq B$ $\rightarrow$ Parallelogram_flat A B B A.

Lemma plgf_trivial2 : $\forall$ A B, $A \neq B$ $\rightarrow$ Parallelogram_flat A A B B.

Lemma plgf_not_point : $\forall$ A B, Parallelogram_flat A A B B $\rightarrow$ $A \neq B$.

Lemma plgf_trivial_neq : $\forall$ A C D, Parallelogram_flat A A C D $\rightarrow$ $C = D \wedge A \neq C$.

Lemma plgf_trivial_trans : $\forall$ A B C, Parallelogram_flat A A B B $\rightarrow$ Parallelogram_flat B B C C
$\rightarrow$ Parallelogram_flat A A C C $\vee$ $A = C$.

Lemma plgf_trivial : $\forall$ A B, $A \neq B$ $\rightarrow$ Parallelogram_flat A B B A.

Lemma plgf3_mid : $\forall$ A B C, Parallelogram_flat A B A C $\rightarrow$ is_midpoint A B C.

Lemma cong3_id : $\forall$ A B C D, $A \neq B$ $\rightarrow$ Col A B C $\rightarrow$ Col A B D $\rightarrow$ Cong A B C D $\rightarrow$ Cong A D B C $\rightarrow$ Cong A C B D
$\rightarrow$ $A = D \wedge B = C \vee A = C \wedge B = D$.

Lemma col_cong_mid1 : $\forall$ A B C D, $A \neq D$ $\rightarrow$ Col A B C $\rightarrow$ Col A B D $\rightarrow$ Cong A B C D $\rightarrow$ Cong A C B D
$\rightarrow$ $\exists$ M, is_midpoint M A D $\wedge$ is_midpoint M B C.

Lemma col_cong_mid2 : $\forall$ A B C D, $A \neq C$ $\rightarrow$ Col A B C $\rightarrow$ Col A B D $\rightarrow$ Cong A B C D $\rightarrow$ Cong A D B C
$\rightarrow$ $\exists$ M, is_midpoint M A C $\wedge$ is_midpoint M B D.

Lemma plgs_not_col : $\forall$ A B C D, Parallelogram_strict A B C D $\rightarrow$ $\neg$Col A B C $\wedge$ $\neg$Col A B D.

Lemma not_col_sym_not_col : $\forall$ A B B' C , $\neg$Col A B C $\rightarrow$ is_midpoint A B B' $\rightarrow$ $\neg$Col A B' C.

Lemma plg_existence : $\forall$ A B C, $A \neq B$ $\rightarrow$ $\exists$ D, Parallelogram A B C D.

Lemma plgs_diff : $\forall$ A B C D, Parallelogram_strict A B C D $\rightarrow$ Parallelogram_strict A B C D $\wedge$ $A \neq B$ $\wedge$ $B \neq C$ $\wedge$ $C \neq D$ $\wedge$ $D \neq A$ $\wedge$ $A \neq C$ $\wedge$ $B \neq D$.

Lemma sym_par : $\forall$ A B M, $A \neq B$ $\rightarrow$ $\forall$ A' B', is_midpoint M A A' $\rightarrow$ is_midpoint M B B' $\rightarrow$ Par A B A' B'.

Lemma symmetry_preserves_two_sides : $\forall$ A B X Y M A' B', Col X Y M $\rightarrow$ two_sides X Y A B $\rightarrow$ is_midpoint M A A' $\rightarrow$ is_midpoint M B B'
$\rightarrow$ two_sides X Y A' B'.

Lemma symmetry_preserves_one_side : $\forall$ A B X Y M A' B', Col X Y M $\rightarrow$ one_side X Y A B $\rightarrow$ is_midpoint M A A' $\rightarrow$ is_midpoint M B B'
$\rightarrow$ one_side X Y A' B'.

Lemma plgf_bet : $\forall$ A B A' B', Parallelogram_flat A B B' A'
$\rightarrow$ Bet A' B' A $\wedge$ Bet B' A B
$\vee$ Bet A' A B' $\wedge$ Bet A B' B
$\vee$ Bet A A' B $\wedge$ Bet A' B B'
$\vee$ Bet A B A' $\wedge$ Bet B A' B'.

Lemma bet2_cong_bet : $\forall$ A B C D, $A \neq B$ $\rightarrow$ Bet A B C $\rightarrow$ Bet A B D $\rightarrow$ Cong A C B D $\rightarrow$ Bet B C D.

Lemma not_col_exists : $\forall$ A B, $A \neq B$ $\rightarrow$ $\exists$ P, $\neg$Col A B P.

Lemma plgs_existence : $\forall$ A B, $A \neq B$ $\rightarrow$ $\exists$ C, $\exists$ D, Parallelogram_strict A B C D.

Lemma Rectangle_not_triv : $\forall$ A,
$\neg$ Rectangle A A A A.

Lemma Rectangle_triv : $\forall$ A B,
$A \neq B$ $\rightarrow$
Rectangle A A B B.

Lemma Rectangle_not_triv_2 : $\forall$ A B,
$\neg$ Rectangle A B A B.

Square

Definition Square A B C D := Rectangle A B C D $\wedge$ Cong A B B C.

Lemma Square_not_triv : $\forall$ A,
$\neg$ Square A A A A.

Lemma Square_not_triv_2 : $\forall$ A B,
$\neg$ Square A A B B.

```
Lemma Square_not_triv_3 : ∀ A B,
  ¬ Square A B A B.

Lemma Square_Rectangle : ∀ A B C D,
  Square A B C D → Rectangle A B C D.

Lemma Square_Parallelogram : ∀ A B C D,
  Square A B C D → Parallelogram A B C D.

Lemma Rhombus_Rectangle_Square : ∀ A B C D,
  Rhombus A B C D →
  Rectangle A B C D →
  Square A B C D.

Lemma rhombus_cong_square : ∀ A B C D,
  Rhombus A B C D →
  Cong A C B D →
  Square A B C D.
```

Kite

```
Definition Kite A B C D :=
  Cong B C C D ∧ Cong D A A B.

Lemma Kite_comm : ∀ A B C D,
  Kite A B C D → Kite C D A B.

End Quadrilateral.
```

Chapter 24

Library quadrilaterals_inter_dec

```
Require Export Ch12_parallel_inter_dec.
Require Export quadrilaterals.
Require Export Tagged_predicates.

Ltac midpoint M A B :=
 let T:= fresh in assert (T:= midpoint_existence A B);
 ex_and T M.

Tactic Notation "Name" ident(M) "the" "midpoint" "of" ident(A) "and" ident(B) :=
 midpoint M A B.

Ltac image_6 A B P' H P:=
 let T:= fresh in assert (T:= l10_6_existence A B P' H);
 ex_and T P.

Ltac image A B P P':=
 let T:= fresh in assert (T:= l10_2_existence A B P);
 ex_and T P'.

Ltac perp A B C X :=
 match goal with
   | H:(¬Col A B C) ⊢ _ ⇒
     let T:= fresh in assert (T:= l8_18_existence A B C H);
     ex_and T X
 end.

Ltac parallel A B C D P :=
 match goal with
   | H:(A ≠ B) ⊢ _ ⇒
     let T := fresh in assert(T:= parallel_existence A B P H);
     ex_and T C
 end.

Ltac par_strict :=
repeat
```

```
 match goal with
        | H: Par_strict ?A ?B ?C ?D ⊢ _ ⇒
         let T := fresh in not_exist_hyp (Par_strict B A D C); assert (T := par_strict_comm
A B C D H)
        | H: Par_strict ?A ?B ?C ?D ⊢ _ ⇒
         let T := fresh in not_exist_hyp (Par_strict C D A B); assert (T := par_strict_symmetry
A B C D H)
        | H: Par_strict ?A ?B ?C ?D ⊢ _ ⇒
         let T := fresh in not_exist_hyp (Par_strict B A C D); assert (T := par_strict_left_comm
A B C D H)
        | H: Par_strict ?A ?B ?C ?D ⊢ _ ⇒
         let T := fresh in not_exist_hyp (Par_strict A B D C); assert (T := par_strict_right_comm
A B C D H)
 end.

Ltac clean_trivial_hyps :=
  repeat
  match goal with
   | H:(Cong ?X1 ?X1 ?X2 ?X2) ⊢ _ ⇒ clear H
   | H:(Cong ?X1 ?X2 ?X2 ?X1) ⊢ _ ⇒ clear H
   | H:(Cong ?X1 ?X2 ?X1 ?X2) ⊢ _ ⇒ clear H
   | H:(Bet ?X1 ?X1 ?X2) ⊢ _ ⇒ clear H
   | H:(Bet ?X2 ?X1 ?X1) ⊢ _ ⇒ clear H
   | H:(Col ?X1 ?X1 ?X2) ⊢ _ ⇒ clear H
   | H:(Col ?X2 ?X1 ?X1) ⊢ _ ⇒ clear H
   | H:(Col ?X1 ?X2 ?X1) ⊢ _ ⇒ clear H
   | H:(Par ?X1 ?X2 ?X1 ?X2) ⊢ _ ⇒ clear H
   | H:(Par ?X1 ?X2 ?X2 ?X1) ⊢ _ ⇒ clear H
   | H:(Per ?X1 ?X2 ?X2) ⊢ _ ⇒ clear H
   | H:(Per ?X1 ?X1 ?X2) ⊢ _ ⇒ clear H
   | H:(is_midpoint ?X1 ?X1 ?X1) ⊢ _ ⇒ clear H
end.

Ltac show_distinct2 := unfold not;intro;treat_equalities; try (solve [intuition]).

Ltac symmetric A B A' :=
 let T := fresh in assert(T:= symmetric_point_construction A B);
 ex_and T A'.

Tactic Notation "Name" ident(X) "the" "symmetric" "of" ident(A) "wrt" ident(C) :=
 symmetric A C X.

Ltac finish := repeat match goal with
 | ⊢ Bet ?A ?B ?C ⇒ Between
 | ⊢ Col ?A ?B ?C ⇒ Col
 | ⊢ ¬ Col ?A ?B ?C ⇒ Col
```

```
 | ⊢ Par ?A ?B ?C ?D ⇒ Par
 | ⊢ Par_strict ?A ?B ?C ?D ⇒ Par
 | ⊢ Perp ?A ?B ?C ?D ⇒ Perp
 | ⊢ Perp_in ?A ?B ?C ?D ?E ⇒ Perp
 | ⊢ Per ?A ?B ?C ⇒ Perp
 | ⊢ Cong ?A ?B ?C ?D ⇒ Cong
 | ⊢ is_midpoint ?A ?B ?C ⇒ Midpoint
 | ⊢ ?A<>?B ⇒ apply swap_diff;assumption
 | ⊢ _ ⇒ try assumption
end.

Ltac sfinish := repeat match goal with
 | ⊢ Bet ?A ?B ?C ⇒ Between; eBetween
 | ⊢ Col ?A ?B ?C ⇒ Col;ColR
 | ⊢ ¬ Col ?A ?B ?C ⇒ Col
 | ⊢ ¬ Col ?A ?B ?C ⇒ intro;search_contradiction
 | ⊢ Par ?A ?B ?C ?D ⇒ Par
 | ⊢ Par_strict ?A ?B ?C ?D ⇒ Par
 | ⊢ Perp ?A ?B ?C ?D ⇒ Perp
 | ⊢ Perp_in ?A ?B ?C ?D ?E ⇒ Perp
 | ⊢ Per ?A ?B ?C ⇒ Perp
 | ⊢ Cong ?A ?B ?C ?D ⇒ Cong;eCong
 | ⊢ is_midpoint ?A ?B ?C ⇒ Midpoint
 | ⊢ ?A<>?B ⇒ apply swap_diff;assumption
 | ⊢ ?A<>?B ⇒ intro;treat_equalities; solve [search_contradiction]
 | ⊢ ?G1 ∧ ?G2 ⇒ split
 | ⊢ _ ⇒ try assumption
end.

Ltac clean_reap_hyps :=
  repeat
  match goal with
   | H:(Parallelogram ?A ?B ?C ?D), H2 : Parallelogram ?A ?B ?D ?C ⊢ _ ⇒ clear H2
   | H:(Parallelogram ?A ?B ?C ?D), H2 : Parallelogram ?A ?B ?C ?D ⊢ _ ⇒ clear H2
   | H:(Parallelogram ?A ?B ?C ?D), H2 : Parallelogram ?C ?D ?A ?B ⊢ _ ⇒ clear H2
   | H:(Parallelogram ?A ?B ?C ?D), H2 : Parallelogram ?C ?D ?B ?A ⊢ _ ⇒ clear H2
   | H:(Parallelogram ?A ?B ?C ?D), H2 : Parallelogram ?D ?C ?B ?A ⊢ _ ⇒ clear H2
   | H:(Parallelogram ?A ?B ?C ?D), H2 : Parallelogram ?D ?C ?A ?B ⊢ _ ⇒ clear H2
   | H:(Parallelogram ?A ?B ?C ?D), H2 : Parallelogram ?B ?A ?C ?D ⊢ _ ⇒ clear H2
   | H:(Parallelogram ?A ?B ?C ?D), H2 : Parallelogram ?B ?A ?D ?C ⊢ _ ⇒ clear H2
   | H:(Par ?A ?B ?C ?D), H2 : Par ?A ?B ?D ?C ⊢ _ ⇒ clear H2
   | H:(Par ?A ?B ?C ?D), H2 : Par ?A ?B ?C ?D ⊢ _ ⇒ clear H2
   | H:(Par ?A ?B ?C ?D), H2 : Par ?C ?D ?A ?B ⊢ _ ⇒ clear H2
   | H:(Par ?A ?B ?C ?D), H2 : Par ?C ?D ?B ?A ⊢ _ ⇒ clear H2
```

| H:(Par $?A$ $?B$ $?C$ $?D$), $H2$: Par $?D$ $?C$ $?B$ $?A$ ⊢ _ ⇒ clear $H2$
| H:(Par $?A$ $?B$ $?C$ $?D$), $H2$: Par $?D$ $?C$ $?A$ $?B$ ⊢ _ ⇒ clear $H2$
| H:(Par $?A$ $?B$ $?C$ $?D$), $H2$: Par $?B$ $?A$ $?C$ $?D$ ⊢ _ ⇒ clear $H2$
| H:(Par $?A$ $?B$ $?C$ $?D$), $H2$: Par $?B$ $?A$ $?D$ $?C$ ⊢ _ ⇒ clear $H2$
| H:(Par_strict $?A$ $?B$ $?C$ $?D$), $H2$: Par_strict $?A$ $?B$ $?D$ $?C$ ⊢ _ ⇒ clear $H2$
| H:(Par_strict $?A$ $?B$ $?C$ $?D$), $H2$: Par_strict $?A$ $?B$ $?C$ $?D$ ⊢ _ ⇒ clear $H2$
| H:(Par_strict $?A$ $?B$ $?C$ $?D$), $H2$: Par_strict $?C$ $?D$ $?A$ $?B$ ⊢ _ ⇒ clear $H2$
| H:(Par_strict $?A$ $?B$ $?C$ $?D$), $H2$: Par_strict $?C$ $?D$ $?B$ $?A$ ⊢ _ ⇒ clear $H2$
| H:(Par_strict $?A$ $?B$ $?C$ $?D$), $H2$: Par_strict $?D$ $?C$ $?B$ $?A$ ⊢ _ ⇒ clear $H2$
| H:(Par_strict $?A$ $?B$ $?C$ $?D$), $H2$: Par_strict $?D$ $?C$ $?A$ $?B$ ⊢ _ ⇒ clear $H2$
| H:(Par_strict $?A$ $?B$ $?C$ $?D$), $H2$: Par_strict $?B$ $?A$ $?C$ $?D$ ⊢ _ ⇒ clear $H2$
| H:(Par_strict $?A$ $?B$ $?C$ $?D$), $H2$: Par_strict $?B$ $?A$ $?D$ $?C$ ⊢ _ ⇒ clear $H2$
| H:(Perp $?A$ $?B$ $?C$ $?D$), $H2$: Perp $?A$ $?B$ $?D$ $?C$ ⊢ _ ⇒ clear $H2$
| H:(Perp $?A$ $?B$ $?C$ $?D$), $H2$: Perp $?A$ $?B$ $?C$ $?D$ ⊢ _ ⇒ clear $H2$
| H:(Perp $?A$ $?B$ $?C$ $?D$), $H2$: Perp $?C$ $?D$ $?A$ $?B$ ⊢ _ ⇒ clear $H2$
| H:(Perp $?A$ $?B$ $?C$ $?D$), $H2$: Perp $?C$ $?D$ $?B$ $?A$ ⊢ _ ⇒ clear $H2$
| H:(Perp $?A$ $?B$ $?C$ $?D$), $H2$: Perp $?D$ $?C$ $?B$ $?A$ ⊢ _ ⇒ clear $H2$
| H:(Perp $?A$ $?B$ $?C$ $?D$), $H2$: Perp $?D$ $?C$ $?A$ $?B$ ⊢ _ ⇒ clear $H2$
| H:(Perp $?A$ $?B$ $?C$ $?D$), $H2$: Perp $?B$ $?A$ $?C$ $?D$ ⊢ _ ⇒ clear $H2$
| H:(Perp $?A$ $?B$ $?C$ $?D$), $H2$: Perp $?B$ $?A$ $?D$ $?C$ ⊢ _ ⇒ clear $H2$
| H:(Perp_in $?X$ $?A$ $?B$ $?C$ $?D$), $H2$: Perp_in $?X$ $?A$ $?B$ $?D$ $?C$ ⊢ _ ⇒ clear $H2$
| H:(Perp_in $?X$ $?A$ $?B$ $?C$ $?D$), $H2$: Perp_in $?X$ $?A$ $?B$ $?C$ $?D$ ⊢ _ ⇒ clear $H2$
| H:(Perp_in $?X$ $?A$ $?B$ $?C$ $?D$), $H2$: Perp_in $?X$ $?C$ $?D$ $?A$ $?B$ ⊢ _ ⇒ clear $H2$
| H:(Perp_in $?X$ $?A$ $?B$ $?C$ $?D$), $H2$: Perp_in $?X$ $?C$ $?D$ $?B$ $?A$ ⊢ _ ⇒ clear $H2$
| H:(Perp_in $?X$ $?A$ $?B$ $?C$ $?D$), $H2$: Perp_in $?X$ $?D$ $?C$ $?B$ $?A$ ⊢ _ ⇒ clear $H2$
| H:(Perp_in $?X$ $?A$ $?B$ $?C$ $?D$), $H2$: Perp_in $?X$ $?D$ $?C$ $?A$ $?B$ ⊢ _ ⇒ clear $H2$
| H:(Perp_in $?X$ $?A$ $?B$ $?C$ $?D$), $H2$: Perp_in $?X$ $?B$ $?A$ $?C$ $?D$ ⊢ _ ⇒ clear $H2$
| H:(Perp_in $?X$ $?A$ $?B$ $?C$ $?D$), $H2$: Perp_in $?X$ $?B$ $?A$ $?D$ $?C$ ⊢ _ ⇒ clear $H2$
| H:(Per $?A$ $?B$ $?C$), $H2$: Per $?A$ $?B$ $?C$ ⊢ _ ⇒ clear $H2$
| H:(Per $?A$ $?B$ $?C$), $H2$: Per $?A$ $?C$ $?B$ ⊢ _ ⇒ clear $H2$
| H:(Per $?A$ $?B$ $?C$), $H2$: Per $?B$ $?A$ $?C$ ⊢ _ ⇒ clear $H2$
| H:(Per $?A$ $?B$ $?C$), $H2$: Per $?B$ $?C$ $?A$ ⊢ _ ⇒ clear $H2$
| H:(Per $?A$ $?B$ $?C$), $H2$: Per $?C$ $?B$ $?A$ ⊢ _ ⇒ clear $H2$
| H:(Per $?A$ $?B$ $?C$), $H2$: Per $?C$ $?A$ $?B$ ⊢ _ ⇒ clear $H2$
| H:(is_midpoint $?A$ $?B$ $?C$), $H2$: is_midpoint $?A$ $?C$ $?B$ ⊢ _ ⇒ clear $H2$
| H:(is_midpoint $?A$ $?B$ $?C$), $H2$: is_midpoint $?A$ $?B$ $?C$ ⊢ _ ⇒ clear $H2$
| H:(¬Col $?A$ $?B$ $?C$), $H2$: (¬Col $?B$ $?A$ $?C$) ⊢ _ ⇒ clear $H2$
| H:(¬Col $?A$ $?B$ $?C$), $H2$: (¬Col $?B$ $?C$ $?A$) ⊢ _ ⇒ clear $H2$
| H:(¬Col $?A$ $?B$ $?C$), $H2$: (¬Col $?C$ $?B$ $?A$) ⊢ _ ⇒ clear $H2$
| H:(¬Col $?A$ $?B$ $?C$), $H2$: (¬Col $?C$ $?A$ $?B$) ⊢ _ ⇒ clear $H2$
| H:(¬Col $?A$ $?B$ $?C$), $H2$: (¬Col $?A$ $?C$ $?B$) ⊢ _ ⇒ clear $H2$
| H:(¬Col $?A$ $?B$ $?C$), $H2$: (¬Col $?A$ $?B$ $?C$) ⊢ _ ⇒ clear $H2$

```
    | H:(Col ?A ?B ?C), H2 : Col ?A ?B ?C ⊢ _ ⇒ clear H2
    | H:(Col ?A ?B ?C), H2 : Col ?A ?C ?B ⊢ _ ⇒ clear H2
    | H:(Col ?A ?B ?C), H2 : Col ?B ?A ?C ⊢ _ ⇒ clear H2
    | H:(Col ?A ?B ?C), H2 : Col ?B ?C ?A ⊢ _ ⇒ clear H2
    | H:(Col ?A ?B ?C), H2 : Col ?C ?B ?A ⊢ _ ⇒ clear H2
    | H:(Col ?A ?B ?C), H2 : Col ?C ?A ?B ⊢ _ ⇒ clear H2
    | H:(Bet ?A ?B ?C), H2 : Bet ?C ?B ?A ⊢ _ ⇒ clear H2
    | H:(Bet ?A ?B ?C), H2 : Bet ?A ?B ?C ⊢ _ ⇒ clear H2
    | H:(Cong ?A ?B ?C ?D), H2 : Cong ?A ?B ?D ?C ⊢ _ ⇒ clear H2
    | H:(Cong ?A ?B ?C ?D), H2 : Cong ?A ?B ?C ?D ⊢ _ ⇒ clear H2
    | H:(Cong ?A ?B ?C ?D), H2 : Cong ?C ?D ?A ?B ⊢ _ ⇒ clear H2
    | H:(Cong ?A ?B ?C ?D), H2 : Cong ?C ?D ?B ?A ⊢ _ ⇒ clear H2
    | H:(Cong ?A ?B ?C ?D), H2 : Cong ?D ?C ?B ?A ⊢ _ ⇒ clear H2
    | H:(Cong ?A ?B ?C ?D), H2 : Cong ?D ?C ?A ?B ⊢ _ ⇒ clear H2
    | H:(Cong ?A ?B ?C ?D), H2 : Cong ?B ?A ?C ?D ⊢ _ ⇒ clear H2
    | H:(Cong ?A ?B ?C ?D), H2 : Cong ?B ?A ?D ?C ⊢ _ ⇒ clear H2
    | H:(?A<>?B), H2 : (?B<>?A) ⊢ _ ⇒ clear H2
    | H:(?A<>?B), H2 : (?A<>?B) ⊢ _ ⇒ clear H2
end.

Ltac tag_hyps :=
  repeat
  match goal with
     | H : ?A ≠ ?B ⊢ _ ⇒ apply Diff_Diff_tagged in H
     | H : Cong ?A ?B ?C ?D ⊢ _ ⇒ apply Cong_Cong_tagged in H
     | H : Bet ?A ?B ?C ⊢ _ ⇒ apply Bet_Bet_tagged in H
     | H : Col ?A ?B ?C ⊢ _ ⇒ apply Col_Col_tagged in H
     | H : ¬ Col ?A ?B ?C ⊢ _ ⇒ apply NCol_NCol_tagged in H
     | H : is_midpoint ?A ?B ?C ⊢ _ ⇒ apply Mid_Mid_tagged in H
     | H : Per ?A ?B ?C ⊢ _ ⇒ apply Per_Per_tagged in H
     | H : Perp_in ?X ?A ?B ?C ?D ⊢ _ ⇒ apply Perp_in_Perp_in_tagged in H
     | H : Perp ?A ?B ?C ?D ⊢ _ ⇒ apply Perp_Perp_tagged in H
     | H : Par_strict ?A ?B ?C ?D ⊢ _ ⇒ apply Par_strict_Par_strict_tagged in H
     | H : Par ?A ?B ?C ?D ⊢ _ ⇒ apply Par_Par_tagged in H
     | H : Parallelogram ?A ?B ?C ?D ⊢ _ ⇒ apply Plg_Plg_tagged in H
  end.

Ltac permutation_intro_in_goal :=
 match goal with
 | ⊢ Par ?A ?B ?C ?D ⇒ apply Par_cases
 | ⊢ Par_strict ?A ?B ?C ?D ⇒ apply Par_strict_cases
 | ⊢ Perp ?A ?B ?C ?D ⇒ apply Perp_cases
 | ⊢ Perp_in ?X ?A ?B ?C ?D ⇒ apply Perp_in_cases
 | ⊢ Per ?A ?B ?C ⇒ apply Per_cases
```

```
  | ⊢ is_midpoint ?A ?B ?C ⇒ apply Mid_cases
  | ⊢ ¬ Col ?A ?B ?C ⇒ apply NCol_cases
  | ⊢ Col ?A ?B ?C ⇒ apply Col_cases
  | ⊢ Bet ?A ?B ?C ⇒ apply Bet_cases
  | ⊢ Cong ?A ?B ?C ?D ⇒ apply Cong_cases
  end.

Ltac perm_apply t :=
  permutation_intro_in_goal;
  try_or ltac:(apply t;solve [finish]).

Section Quadrilateral_inter_dec_1.

Context `{MT:Tarski_2D_euclidean}.
Context `{EqDec:EqDecidability Tpoint}.
Context `{InterDec:InterDecidability Tpoint Col}.

Lemma par_cong_mid_ts :
  ∀ A B A' B',
    Par_strict A B A' B' →
    Cong A B A' B' →
    two_sides A A' B B' →
    ∃ M , is_midpoint M A A' ∧ is_midpoint M B B'.

Lemma par_cong_mid_os :
    ∀ A B A' B',
      Par_strict A B A' B' →
      Cong A B A' B' →
      one_side A A' B B' →
      ∃ M , is_midpoint M A B' ∧ is_midpoint M B A'.

Lemma par_strict_cong_mid :
  ∀ A B A' B',
    Par_strict A B A' B' →
    Cong A B A' B' →
    ∃ M , is_midpoint M A A' ∧ is_midpoint M B B' ∨
                 is_midpoint M A B' ∧ is_midpoint M B A'.

Lemma par_strict_cong_mid1 :
  ∀ A B A' B',
    Par_strict A B A' B' →
    Cong A B A' B' →
    (two_sides A A' B B' ∧ ∃ M , is_midpoint M A A' ∧ is_midpoint M B B') ∨
    (one_side A A' B B' ∧ ∃ M , is_midpoint M A B' ∧ is_midpoint M B A').

Lemma par_cong_mid :
  ∀ A B A' B',
    Par A B A' B' →
```

Cong A B A' B' $\rightarrow$
$\exists$ M, is_midpoint M A A' $\wedge$ is_midpoint M B B' $\vee$
is_midpoint M A B' $\wedge$ is_midpoint M B A'.

Lemma ts_cong_par_cong_par :
$\forall$ A B A' B',
two_sides A A' B B' $\rightarrow$
Cong A B A' B' $\rightarrow$
Par A B A' B' $\rightarrow$
Cong A B' A' B $\wedge$ Par A B' A' B.

Lemma plgs_cong :
$\forall$ A B C D,
Parallelogram_strict A B C D $\rightarrow$
Cong A B C D $\wedge$ Cong A D B C.

Lemma plg_cong :
$\forall$ A B C D,
Parallelogram A B C D $\rightarrow$
Cong A B C D $\wedge$ Cong A D B C.

Lemma rmb_cong :
$\forall$ A B C D,
Rhombus A B C D $\rightarrow$
Cong A B B C $\wedge$ Cong A B C D $\wedge$ Cong A B D A.

Lemma rmb_per:
$\forall$ A B C D M,
is_midpoint M A C $\rightarrow$
Rhombus A B C D $\rightarrow$
Per A M D.

Lemma per_rmb :
$\forall$ A B C D M,
Plg A B C D $\rightarrow$
is_midpoint M A C $\rightarrow$
Per A M B $\rightarrow$
Rhombus A B C D.

Lemma perp_rmb :
$\forall$ A B C D,
Plg A B C D $\rightarrow$
Perp A C B D $\rightarrow$
Rhombus A B C D.

Lemma plg_conga1 : $\forall$ A B C D, $A \neq B \rightarrow A \neq C \rightarrow$ Plg A B C D $\rightarrow$ Conga B A C D C A.

Lemma os_cong_par_cong_par :

∀ A B A' B',
one_side A A' B B' →
Cong A B A' B' →
Par A B A' B' →
Cong A A' B B' ∧ Par A A' B B'.

Lemma plgs_permut :
∀ A B C D,
Parallelogram_strict A B C D →
Parallelogram_strict B C D A.

Lemma plg_permut :
∀ A B C D,
Parallelogram A B C D →
Parallelogram B C D A.

Lemma plgs_sym :
∀ A B C D,
Parallelogram_strict A B C D →
Parallelogram_strict C D A B.

Lemma plg_sym :
∀ A B C D,
Parallelogram A B C D →
Parallelogram C D A B.

Lemma plgs_mid :
∀ A B C D,
Parallelogram_strict A B C D →
∃ M , is_midpoint M A C ∧ is_midpoint M B D.

Lemma plg_mid :
∀ A B C D,
Parallelogram A B C D →
∃ M , is_midpoint M A C ∧ is_midpoint M B D.

Lemma plgs_not_comm :
∀ A B C D,
Parallelogram_strict A B C D →
¬ Parallelogram_strict A B D C ∧ ¬ Parallelogram_strict B A C D.

Lemma plg_not_comm :
∀ A B C D,
$A \neq B$ →
Parallelogram A B C D →
¬ Parallelogram A B D C ∧ ¬ Parallelogram B A C D.

Lemma parallelogram_to_plg : ∀ A B C D, Parallelogram A B C D → Plg A B C D.

Lemma parallelogram_equiv_plg : ∀ A B C D, Parallelogram A B C D ↔ Plg A B C D.

Lemma plg_conga : ∀ $A\ B\ C\ D$, Distincts $A\ B\ C$ → Parallelogram $A\ B\ C\ D$ → Conga $A\ B\ C$ $C\ D\ A$ ∧ Conga $B\ C\ D\ D\ A\ B$.

Lemma half_plgs :
∀ $A\ B\ C\ D\ P\ Q\ M$,
Parallelogram_strict $A\ B\ C\ D$ →
is_midpoint $P\ A\ B$ →
is_midpoint $Q\ C\ D$ →
is_midpoint $M\ A\ C$ →
Par $P\ Q\ A\ D$ ∧ is_midpoint $M\ P\ Q$ ∧ Cong $A\ D\ P\ Q$.

Lemma plgs_two_sides :
∀ $A\ B\ C\ D$,
Parallelogram_strict $A\ B\ C\ D$ →
two_sides $A\ C\ B\ D$ ∧ two_sides $B\ D\ A\ C$.

Lemma plgs_par_strict :
∀ $A\ B\ C\ D$,
Parallelogram_strict $A\ B\ C\ D$ →
Par_strict $A\ B\ C\ D$ ∧ Par_strict $A\ D\ B\ C$.

Lemma plgs_half_plgs_aux :
∀ $A\ B\ C\ D\ P\ Q$,
Parallelogram_strict $A\ B\ C\ D$ →
is_midpoint $P\ A\ B$ →
is_midpoint $Q\ C\ D$ →
Parallelogram_strict $A\ P\ Q\ D$.

Lemma plgs_comm2 :
∀ $A\ B\ C\ D$,
Parallelogram_strict $A\ B\ C\ D$ →
Parallelogram_strict $B\ A\ D\ C$.

Lemma plgf_comm2 :
∀ $A\ B\ C\ D$,
Parallelogram_flat $A\ B\ C\ D$ →
Parallelogram_flat $B\ A\ D\ C$.

Lemma plg_comm2 :
∀ $A\ B\ C\ D$,
Parallelogram $A\ B\ C\ D$ →
Parallelogram $B\ A\ D\ C$.

Lemma par_preserves_conga_ts :
∀ $A\ B\ C\ D$, Par $A\ B\ C\ D$ → two_sides $B\ D\ A\ C$ → Conga $A\ B\ D\ C\ D\ B$.

Lemma par_preserves_conga_os :
∀ $A\ B\ C\ D\ P$, Par $A\ B\ C\ D$ → Bet $A\ D\ P$ → $D \neq P$ → one_side $A\ D\ B\ C$ → Conga B $A\ P\ C\ D\ P$.

Lemma cong3_par2_par :
$\forall$ A B C A' B' C', $A \neq C \rightarrow$ Cong_3 B A C B' A' $C' \rightarrow$ Par B A B' $A' \rightarrow$ Par B C B' $C' \rightarrow$
Par A C A' C' $\vee$ $\neg$ Par_strict B B' A A' $\vee$ $\neg$Par_strict B B' C C'.

Lemma square_perp_rectangle : $\forall$ A B C D,
Rectangle A B C $D \rightarrow$
Perp A C B $D \rightarrow$
Square A B C D.

Lemma plgs_half_plgs :
$\forall$ A B C D P Q,
Parallelogram_strict A B C $D \rightarrow$
is_midpoint P A $B \rightarrow$ is_midpoint Q C $D \rightarrow$
Parallelogram_strict A P Q D $\wedge$ Parallelogram_strict B P Q C.

Lemma parallel_2_plg :
$\forall$ A B C D,
$\neg$ Col A B $C \rightarrow$
Par A B C $D \rightarrow$
Par A D B $C \rightarrow$
Parallelogram_strict A B C D.

Lemma par_2_plg :
$\forall$ A B C D,
$\neg$ Col A B $C \rightarrow$
Par A B C $D \rightarrow$
Par A D B $C \rightarrow$
Parallelogram A B C D.

Lemma parallelogram_strict_not_col_2 : $\forall$ A B C D,
Parallelogram_strict A B C $D \rightarrow$
$\neg$ Col B C D.

Lemma parallelogram_strict_not_col_3 : $\forall$ A B C D,
Parallelogram_strict A B C $D \rightarrow$
$\neg$ Col C D A.

Lemma parallelogram_strict_not_col_4 : $\forall$ A B C D,
Parallelogram_strict A B C $D \rightarrow$
$\neg$ Col A B D.

Lemma plg_cong_1 : $\forall$ A B C D, Parallelogram A B C $D \rightarrow$ Cong A B C D.

Lemma plg_cong_2 : $\forall$ A B C D, Parallelogram A B C $D \rightarrow$ Cong A D B C.

Lemma plgs_cong_1 : $\forall$ A B C D, Parallelogram_strict A B C $D \rightarrow$ Cong A B C D.

Lemma plgs_cong_2 : $\forall$ A B C D, Parallelogram_strict A B C $D \rightarrow$ Cong A D B C.

Lemma Plg_perm :

$\forall$ A B C D,
Parallelogram A B C D $\rightarrow$
Parallelogram A B C D $\wedge$ Parallelogram B C D A $\wedge$ Parallelogram C D A B $\wedge$Parallelogram D A B C $\wedge$
Parallelogram A D C B $\wedge$ Parallelogram D C B A $\wedge$ Parallelogram C B A D $\wedge$ Parallelogram B A D C.

Lemma plg_not_comm_1 :
$\forall$ A B C D : Tpoint,
$A \neq B$ $\rightarrow$
Parallelogram A B C D $\rightarrow$ $\neg$ Parallelogram A B D C.

Lemma plg_not_comm_2 :
$\forall$ A B C D : Tpoint,
$A \neq B$ $\rightarrow$
Parallelogram A B C D $\rightarrow$ $\neg$ Parallelogram B A C D.

End Quadrilateral_inter_dec_1.

Ltac *permutation_intro_in_hyps_aux* :=
repeat
match goal with
| H : Plg_tagged ?A ?B ?C ?D $\vdash$ _ $\Rightarrow$ apply Plg_tagged_Plg in H; apply Plg_perm in H; *spliter*
| H : Par_tagged ?A ?B ?C ?D $\vdash$ _ $\Rightarrow$ apply Par_tagged_Par in H; apply Par_perm in H; *spliter*
| H : Par_strict_tagged ?A ?B ?C ?D $\vdash$ _ $\Rightarrow$ apply Par_strict_tagged_Par_strict in H; apply Par_strict_perm in H; *spliter*
| H : Perp_tagged ?A ?B ?C ?D $\vdash$ _ $\Rightarrow$ apply Perp_tagged_Perp in H; apply Perp_perm in H; *spliter*
| H : Perp_in_tagged ?X ?A ?B ?C ?D $\vdash$ _ $\Rightarrow$ apply Perp_in_tagged_Perp_in in H; apply Perp_in_perm in H; *spliter*
| H : Per_tagged ?A ?B ?C $\vdash$ _ $\Rightarrow$ apply Per_tagged_Per in H; apply Per_perm in H; *spliter*
| H : Mid_tagged ?A ?B ?C $\vdash$ _ $\Rightarrow$ apply Mid_tagged_Mid in H; apply Mid_perm in H; *spliter*
| H : NCol_tagged ?A ?B ?C $\vdash$ _ $\Rightarrow$ apply NCol_tagged_NCol in H; apply NCol_perm in H; *spliter*
| H : Col_tagged ?A ?B ?C $\vdash$ _ $\Rightarrow$ apply Col_tagged_Col in H; apply Col_perm in H; *spliter*
| H : Bet_tagged ?A ?B ?C $\vdash$ _ $\Rightarrow$ apply Bet_tagged_Bet in H; apply Bet_perm in H; *spliter*
| H : Cong_tagged ?A ?B ?C ?D $\vdash$ _ $\Rightarrow$ apply Cong_tagged_Cong in H; apply Cong_perm in H; *spliter*
| H : Diff_tagged ?A ?B $\vdash$ _ $\Rightarrow$ apply Diff_tagged_Diff in H; apply Diff_perm in H; *spliter*

```
 end.
```

Ltac *permutation_intro_in_hyps* := *clean_reap_hyps*; *clean_trivial_hyps*; *tag_hyps*; *permutation_intro_in_hyps_aux*.

```
Ltac assert_cols_aux :=
repeat
 match goal with
      | H:Bet ?X1 ?X2 ?X3 ⊢ _ ⇒
     not_exist_hyp (Col X1 X2 X3);assert (Col X1 X2 X3) by (apply bet_col;apply H)

      | H:is_midpoint ?X1 ?X2 ?X3 ⊢ _ ⇒
     not_exist_hyp (Col X1 X2 X3);let N := fresh in assert (N := midpoint_col X2 X1
X3 H)

      | H:out ?X1 ?X2 ?X3 ⊢ _ ⇒
     not_exist_hyp (Col X1 X2 X3);let N := fresh in assert (N := out_col X1 X2 X3
H)

      | H:Par ?X1 ?X2 ?X1 ?X3 ⊢ _ ⇒
     not_exist_hyp (Col X1 X2 X3);let N := fresh in assert (N := par_id X1 X2 X3
H)
      | H:Par ?X1 ?X2 ?X1 ?X3 ⊢ _ ⇒
     not_exist_hyp (Col X1 X2 X3);let N := fresh in assert (N := par_id_1 X1 X2 X3
H)
      | H:Par ?X1 ?X2 ?X1 ?X3 ⊢ _ ⇒
     not_exist_hyp (Col X1 X2 X3);let N := fresh in assert (N := par_id_2 X1 X2 X3
H)
      | H:Par ?X1 ?X2 ?X1 ?X3 ⊢ _ ⇒
     not_exist_hyp (Col X1 X2 X3);let N := fresh in assert (N := par_id_3 X1 X2 X3
H)
      | H:Par ?X1 ?X2 ?X1 ?X3 ⊢ _ ⇒
     not_exist_hyp (Col X1 X2 X3);let N := fresh in assert (N := par_id_4 X1 X2 X3
H)
      | H:Par ?X1 ?X2 ?X1 ?X3 ⊢ _ ⇒
     not_exist_hyp (Col X1 X2 X3);let N := fresh in assert (N := par_id_5 X1 X2 X3
H)

      | H:Par ?X1 ?X2 ?X3 ?X4, H2:Col ?X1 ?X2 ?X5, H3:Col ?X3 ?X4 ?X5 ⊢ _ ⇒
     not_exist_hyp (Col X1 X2 X3);let N := fresh in assert (N := not_strict_par1 X1
X2 X3 X4 X5 H H2 H3)
 end.
```

Ltac *assert_cols* := *permutation_intro_in_hyps*; *assert_cols_aux*; *clean_reap_hyps*.

```
Ltac not_exist_hyp_perm_cong A B C D := not_exist_hyp (Cong A B C D); not_exist_hyp
(Cong A B D C);
                                        not_exist_hyp (Cong B A C D); not_exist_hyp
(Cong B A D C);
                                        not_exist_hyp (Cong C D A B); not_exist_hyp
(Cong C D B A);
                                        not_exist_hyp (Cong D C A B); not_exist_hyp
(Cong D C B A).

Ltac assert_congs_1 :=
repeat
 match goal with
      | H:is_midpoint ?X1 ?X2 ?X3 ⊢ _ ⇒
      let h := fresh in
      not_exist_hyp_perm_cong X1 X2 X1 X3;
      assert (h := midpoint_cong X2 X1 X3 H)

      | H1:is_midpoint ?M1 ?A1 ?B1, H2:is_midpoint ?M2 ?A2 ?B2, H3:Cong ?A1 ?B1
?A2 ?B2 ⊢ _ ⇒
      let h := fresh in
      not_exist_hyp_perm_cong A1 M1 A2 M2;
      assert (h := cong_cong_half_1 A1 M1 B1 A2 M2 B2 H1 H2 H3)

      | H:Parallelogram ?X1 ?X2 ?X3 ?X4 ⊢ _ ⇒
      let h := fresh in
      not_exist_hyp_perm_cong X1 X2 X3 X4;
      assert (h := plg_cong_1 X1 X2 X3 X4 H)

      | H:Parallelogram_strict ?X1 ?X2 ?X3 ?X4 ⊢ _ ⇒
      let h := fresh in
      not_exist_hyp_perm_cong X1 X2 X3 X4;
      assert (h := plgs_cong_1 X1 X2 X3 X4 H)
  end.

Ltac assert_congs_2 :=
repeat
 match goal with
      | H1:is_midpoint ?M1 ?A1 ?B1, H2:is_midpoint ?M2 ?A2 ?B2, H3:Cong ?A1 ?B1
?A2 ?B2 ⊢ _ ⇒
      let h := fresh in
      not_exist_hyp_perm_cong A1 M1 A2 M2;
      assert (h := cong_cong_half_2 A1 M1 B1 A2 M2 B2 H1 H2 H3)

      | H:Parallelogram ?X1 ?X2 ?X3 ?X4 ⊢ _ ⇒
```

```
      let h := fresh in
      not_exist_hyp_perm_cong X1 X4 X2 X3;
      assert (h := plg_cong_2 X1 X2 X3 X4 H);clean_reap_hyps

      | H:Parallelogram_strict ?X1 ?X2 ?X3 ?X4 ⊢ _ ⇒
      let h := fresh in
      not_exist_hyp_perm_cong X1 X4 X2 X3;
      assert (h := plgs_cong_2 X1 X2 X3 X4 H);clean_reap_hyps
  end.
```

```
Ltac assert_congs := permutation_intro_in_hyps; assert_congs_1; assert_congs_2; clean_reap_hyps.
```

```
Ltac not_exist_hyp_perm_para A B C D := not_exist_hyp (Parallelogram A B C D); not_exist_hyp
(Parallelogram B C D A);
                                        not_exist_hyp (Parallelogram C D A B); not_exist_hyp
(Parallelogram D A B C);
                                        not_exist_hyp (Parallelogram A D C B); not_exist_hyp
(Parallelogram D C B A);
                                        not_exist_hyp (Parallelogram C B A D); not_exist_hyp
(Parallelogram B A D C).
```

```
Ltac not_exist_hyp_perm_para_s A B C D := not_exist_hyp (Parallelogram_strict A B C
D);
                                          not_exist_hyp (Parallelogram_strict B C D
A);
                                          not_exist_hyp (Parallelogram_strict C D A
B);
                                          not_exist_hyp (Parallelogram_strict D A B
C);
                                          not_exist_hyp (Parallelogram_strict A D C
B);
                                          not_exist_hyp (Parallelogram_strict D C B
A);
                                          not_exist_hyp (Parallelogram_strict C B A
D);
                                          not_exist_hyp (Parallelogram_strict B A D
C).
```

```
Ltac assert_paras_aux :=
repeat
 match goal with
      | H:Parallelogram_strict ?X1 ?X2 ?X3 ?X4 ⊢ _ ⇒
      let h := fresh in
      not_exist_hyp_perm_para X1 X2 X3 X4;
      assert (h := Parallelogram_strict_Parallelogram X1 X2 X3 X4 H)
```

```
    | H:Plg ?X1 ?X2 ?X3 ?X4 ⊢ _ ⇒
    let h := fresh in
    not_exist_hyp_perm_para X1 X2 X3 X4;
    assert (h := plg_to_parallelogram X1 X2 X3 X4 H)

    | H:(¬ Col ?X1 ?X2 ?X3), H2:Par ?X1 ?X2 ?X3 ?X4, H3:Par ?X1 ?X4 ?X2 ?X3 ⊢
_ ⇒
    let h := fresh in
    not_exist_hyp_perm_para_s X1 X2 X3 X4;
    assert (h := parallel_2_plg X1 X2 X3 X4 H H2 H3)
  end.
```

```
Ltac assert_paras := permutation_intro_in_hyps; assert_paras_aux; clean_reap_hyps.
```

```
Ltac not_exist_hyp_perm_npara A B C D := not_exist_hyp (¬Parallelogram A B C D);
not_exist_hyp (¬Parallelogram B C D A);
                                    not_exist_hyp (¬Parallelogram C D A B); not_exist_hyp
(¬Parallelogram D A B C);
                                    not_exist_hyp (¬Parallelogram A D C B); not_exist_hyp
(¬Parallelogram D C B A);
                                    not_exist_hyp (¬Parallelogram C B A D); not_exist_hyp
(¬Parallelogram B A D C).
```

```
Ltac assert_nparas_1 :=
 repeat
 match goal with
    | H:(?X1<>?X2), H2:Parallelogram ?X1 ?X2 ?X3 ?X4 ⊢ _ ⇒
    let h := fresh in
    not_exist_hyp_perm_npara X1 X2 X4 X3;
    assert (h := plg_not_comm_1 X1 X2 X3 X4 H H2)
  end.
```

```
Ltac assert_nparas_2 :=
 repeat
 match goal with
    | H:(?X1<>?X2), H2:Parallelogram ?X1 ?X2 ?X3 ?X4 ⊢ _ ⇒
    let h := fresh in
    not_exist_hyp_perm_npara X2 X1 X3 X4;
    assert (h := plg_not_comm_2 X1 X2 X3 X4 H H2)
  end.
```

```
Ltac assert_nparas := permutation_intro_in_hyps; assert_nparas_1; assert_nparas_2; clean_reap_hyps.
```

```
Ltac diag_plg_intersection M A B C D H :=
 let T := fresh in assert(T:= plg_mid A B C D H);
 ex_and T M.
```

```
Tactic Notation "Name" ident(M) "the" "intersection" "of" "the" "diagonals" "(" ident(A)
ident(C) ")" "and" "(" ident(B) ident(D) ")" "of" "the" "parallelogram" ident(H) :=
 diag_plg_intersection M A B C D H.

Ltac assert_diffs :=
repeat
 match goal with
      | H:(¬Col ?X1 ?X2 ?X3) ⊢ _ ⇒
      let h := fresh in
      not_exist_hyp3 X1 X2 X1 X3 X2 X3;
      assert (h := not_col_distincts X1 X2 X3 H);decompose [and] h;clear h;clean_reap_hyps


      | H:Cong ?A ?B ?C ?D, H2 : ?A ≠ ?B ⊢_ ⇒
      let T:= fresh in (not_exist_hyp_comm C D);
        assert (T:= cong_diff A B C D H2 H);clean_reap_hyps
      | H:Cong ?A ?B ?C ?D, H2 : ?B ≠ ?A ⊢_ ⇒
      let T:= fresh in (not_exist_hyp_comm C D);
        assert (T:= cong_diff_2 A B C D H2 H);clean_reap_hyps
      | H:Cong ?A ?B ?C ?D, H2 : ?C ≠ ?D ⊢_ ⇒
      let T:= fresh in (not_exist_hyp_comm A B);
        assert (T:= cong_diff_3 A B C D H2 H);clean_reap_hyps
      | H:Cong ?A ?B ?C ?D, H2 : ?D ≠ ?C ⊢_ ⇒
      let T:= fresh in (not_exist_hyp_comm A B);
        assert (T:= cong_diff_4 A B C D H2 H);clean_reap_hyps

      | H:(Parallelogram_strict ?X1 ?X2 ?X3 ?X4) ⊢ _ ⇒
      let HN := fresh in
      not_exist_hyp (¬Col X1 X2 X3);
      assert (HN := parallelogram_strict_not_col X1 X2 X3 X4 H)
      | H:(Parallelogram_strict ?X1 ?X2 ?X3 ?X4) ⊢ _ ⇒
      let HN := fresh in
      not_exist_hyp (¬Col X2 X3 X4);
      assert (HN := parallelogram_strict_not_col_2 X1 X2 X3 X4 H)
      | H:(Parallelogram_strict ?X1 ?X2 ?X3 ?X4) ⊢ _ ⇒
      let HN := fresh in
      not_exist_hyp (¬Col X3 X4 X1);
      assert (HN := parallelogram_strict_not_col_3 X1 X2 X3 X4 H)
      | H:(Parallelogram_strict ?X1 ?X2 ?X3 ?X4) ⊢ _ ⇒
      let HN := fresh in
      not_exist_hyp (¬Col X1 X2 X4);
      assert (HN := parallelogram_strict_not_col_4 X1 X2 X3 X4 H)
```

```
| H:is_midpoint ?I ?A ?B, H2 : ?A<>?B |- _ =>
let T:= fresh in (not_exist_hyp2 I B I A);
 assert (T:= midpoint_distinct_1 I A B H2 H);
 decompose [and] T;clear T;clean_reap_hyps
| H:is_midpoint ?I ?A ?B, H2 : ?B<>?A |- _ =>
let T:= fresh in (not_exist_hyp2 I B I A);
 assert (T:= midpoint_distinct_1 I A B (swap_diff B A H2) H);
 decompose [and] T;clear T;clean_reap_hyps

| H:is_midpoint ?I ?A ?B, H2 : ?I<>?A |- _ =>
let T:= fresh in (not_exist_hyp2 I B A B);
 assert (T:= midpoint_distinct_2 I A B H2 H);
 decompose [and] T;clear T;clean_reap_hyps
| H:is_midpoint ?I ?A ?B, H2 : ?A<>?I |- _ =>
let T:= fresh in (not_exist_hyp2 I B A B);
 assert (T:= midpoint_distinct_2 I A B (swap_diff A I H2) H);
 decompose [and] T;clear T;clean_reap_hyps

| H:is_midpoint ?I ?A ?B, H2 : ?I<>?B |- _ =>
let T:= fresh in (not_exist_hyp2 I A A B);
 assert (T:= midpoint_distinct_3 I A B H2 H);
 decompose [and] T;clear T;clean_reap_hyps
| H:is_midpoint ?I ?A ?B, H2 : ?B<>?I |- _ =>
let T:= fresh in (not_exist_hyp2 I A A B);
 assert (T:= midpoint_distinct_3 I A B (swap_diff B I H2) H);
 decompose [and] T;clear T;clean_reap_hyps


| H:(Par_strict ?X1 ?X2 ?X3 ?X4) |- _ =>
 let HN := fresh in
 not_exist_hyp (~Col X1 X2 X3);
 assert (HN := par_strict_not_col_1 X1 X2 X3 X4 H)
 | H:(Par_strict ?X1 ?X2 ?X3 ?X4) |- _ =>
 let HN := fresh in
 not_exist_hyp (~Col X2 X3 X4);
 assert (HN := par_strict_not_col_2 X1 X2 X3 X4 H)
 | H:(Par_strict ?X1 ?X2 ?X3 ?X4) |- _ =>
 let HN := fresh in
 not_exist_hyp (~Col X3 X4 X1);
 assert (HN := par_strict_not_col_3 X1 X2 X3 X4 H)
 | H:(Par_strict ?X1 ?X2 ?X3 ?X4) |- _ =>
 let HN := fresh in
```

```
      not_exist_hyp (¬Col X1 X2 X4);
      assert (HN := par_strict_not_col_4 X1 X2 X3 X4 H)


   | H:Par_strict ?A ?B ?C ?D ⊢ _ ⇒
      let T:= fresh in (not_exist_hyp2 A B C D);
       assert (T:= par_strict_distinct A B C D H);
       decompose [and] T;clear T;clean_reap_hyps
      | H:Par ?A ?B ?C ?D ⊢ _ ⇒
      let T:= fresh in (not_exist_hyp2 A B C D);
       assert (T:= par_distincts A B C D H);decompose [and] T;clear T;clean_reap_hyps
      | H:Perp ?A ?B ?C ?D ⊢ _ ⇒
      let T:= fresh in (not_exist_hyp2 A B C D);
       assert (T:= perp_distinct A B C D H);
       decompose [and] T;clear T;clean_reap_hyps
      | H:Perp_in ?X ?A ?B ?C ?D ⊢ _ ⇒
      let T:= fresh in (not_exist_hyp2 A B C D);
       assert (T:= perp_in_distinct X A B C D H);
       decompose [and] T;clear T;clean_reap_hyps
      | H:out ?A ?B ?C ⊢ _ ⇒
      let T:= fresh in (not_exist_hyp2 A B A C);
       assert (T:= out_distinct A B C H);
       decompose [and] T;clear T;clean_reap_hyps
 end.

Hint Resolve parallelogram_strict_not_col
             parallelogram_strict_not_col_2
             parallelogram_strict_not_col_3
             parallelogram_strict_not_col_4 : Col.

Section Quadrilateral_inter_dec_2.

Context `{MT:Tarski_2D_euclidean}.
Context `{EqDec:EqDecidability Tpoint}.
Context `{InterDec:InterDecidability Tpoint Col}.

Lemma parallelogram_strict_midpoint : ∀ A B C D I,
  Parallelogram_strict A B C D →
  Col I A C →
  Col I B D →
  is_midpoint I A C ∧ is_midpoint I B D.

Lemma rmb_perp :
 ∀ A B C D,
  A ≠ C → B ≠ D →
  Rhombus A B C D →
```

Perp $A\ C\ B\ D$.

Lemma par_perp_perp : ∀ $A\ B\ C\ D\ P\ Q$, Par $A\ B\ C\ D$ → Perp $A\ B\ P\ Q$ → Perp $C\ D\ P\ Q$.

Lemma rect_permut : ∀ $A\ B\ C\ D$, Rectangle $A\ B\ C\ D$ → Rectangle $B\ C\ D\ A$.

Lemma rect_comm2 : ∀ $A\ B\ C\ D$, Rectangle $A\ B\ C\ D$ → Rectangle $B\ A\ D\ C$.

Lemma rect_per1 : ∀ $A\ B\ C\ D$, Rectangle $A\ B\ C\ D$ → Per $B\ A\ D$.

Lemma rect_per2 : ∀ $A\ B\ C\ D$, Rectangle $A\ B\ C\ D$ → Per $A\ B\ C$.

Lemma rect_per3 : ∀ $A\ B\ C\ D$, Rectangle $A\ B\ C\ D$ → Per $B\ C\ D$.

Lemma rect_per4 : ∀ $A\ B\ C\ D$, Rectangle $A\ B\ C\ D$ → Per $A\ D\ C$.

Lemma plg_per_rect1 : ∀ $A\ B\ C\ D$, Plg $A\ B\ C\ D$ → Per $D\ A\ B$ → Rectangle $A\ B\ C\ D$.

Lemma plg_per_rect2 : ∀ $A\ B\ C\ D$, Plg $A\ B\ C\ D$ → Per $C\ B\ A$ → Rectangle $A\ B\ C\ D$.

Lemma plg_per_rect3 : ∀ $A\ B\ C\ D$, Plg $A\ B\ C\ D$ → Per $A\ D\ C$ → Rectangle $A\ B\ C\ D$.

Lemma plg_per_rect4 : ∀ $A\ B\ C\ D$, Plg $A\ B\ C\ D$ → Per $B\ C\ D$ → Rectangle $A\ B\ C\ D$.

Lemma plg_per_rect : ∀ $A\ B\ C\ D$, Plg $A\ B\ C\ D$ → (Per $D\ A\ B$ ∨ Per $C\ B\ A$ ∨ Per $A\ D\ C$ ∨ Per $B\ C\ D$) → Rectangle $A\ B\ C\ D$.

Lemma rect_per : ∀ $A\ B\ C\ D$, Rectangle $A\ B\ C\ D$ → Per $B\ A\ D$ ∧ Per $A\ B\ C$ ∧ Per $B\ C\ D$ ∧ Per $A\ D\ C$.

Lemma plgf_rect_id : ∀ $A\ B\ C\ D$, Parallelogram_flat $A\ B\ C\ D$ → Rectangle $A\ B\ C\ D$ → $A = D \land B = C \lor A = B \land D = C$.

Lemma perp_3_perp :
 ∀ $A\ B\ C\ D$,
  Perp $A\ B\ B\ C$ →
  Perp $B\ C\ C\ D$ →
  Perp $C\ D\ D\ A$ →
  Perp $D\ A\ A\ B$.

Lemma perp_3_rect :
 ∀ $A\ B\ C\ D$,
  ¬ Col $A\ B\ C$ →
  Perp $A\ B\ B\ C$ →
  Perp $B\ C\ C\ D$ →
  Perp $C\ D\ D\ A$ →
  Rectangle $A\ B\ C\ D$.

Lemma conga_to_par_ts : ∀ $A\ B\ C\ D$, two_sides $B\ D\ A\ C$ → Conga $A\ B\ D\ C\ D\ B$ → Par $A\ B\ C\ D$.

Lemma conga_to_par_os : ∀ $A\ B\ C\ D\ P$, Bet $A\ D\ P$ → $D \neq P$ → one_side $A\ D\ B\ C$ → Conga $B\ A\ P\ C\ D\ P$
→ Par $A\ B\ C\ D$.

Lemma plg_par : ∀ $A\ B\ C\ D$, $A \neq B$ → $B \neq C$ → Parallelogram $A\ B\ C\ D$ → Par $A\ B\ C\ D$ ∧ Par $A\ D\ B\ C$.

Lemma plg_par_1 : $\forall$ A B C D, $A \neq B \rightarrow B \neq C \rightarrow$ Parallelogram A B C D $\rightarrow$ Par A B C D.

Lemma plg_par_2 : $\forall$ A B C D, $A \neq B \rightarrow B \neq C \rightarrow$ Parallelogram A B C D $\rightarrow$ Par A D B C.

Lemma plgs_pars_1: $\forall$ A B C D : Tpoint, Parallelogram_strict A B C D $\rightarrow$ Par_strict A B C D.

Lemma plgs_pars_2: $\forall$ A B C D : Tpoint, Parallelogram_strict A B C D $\rightarrow$ Par_strict A D B C.

End Quadrilateral_inter_dec_2.

Ltac *not_exist_hyp_perm_par* A B C D := *not_exist_hyp* (Par A B C D); *not_exist_hyp* (Par A B D C);
not_exist_hyp (Par B A C D); *not_exist_hyp* (Par B A D C);
not_exist_hyp (Par C D A B); *not_exist_hyp* (Par C D B A);
not_exist_hyp (Par D C A B); *not_exist_hyp* (Par D C B A).

Ltac *not_exist_hyp_perm_pars* A B C D := *not_exist_hyp* (Par_strict A B C D); *not_exist_hyp* (Par_strict A B D C);
not_exist_hyp (Par_strict B A C D); *not_exist_hyp* (Par_strict B A D C);
not_exist_hyp (Par_strict C D A B); *not_exist_hyp* (Par_strict C D B A);
not_exist_hyp (Par_strict D C A B); *not_exist_hyp* (Par_strict D C B A).

Ltac *assert_pars_1* :=
repeat
match goal with
| H:Par_strict $?X1$ $?X2$ $?X3$ $?X4$ $\vdash$ _ $\Rightarrow$
let h := fresh in
not_exist_hyp_perm_par $X1$ $X2$ $X3$ $X4$;
assert (h := par_strict_par $X1$ $X2$ $X3$ $X4$ H)

| H:Par $?X1$ $?X2$ $?X3$ $?X4$, $H2$:Col $?X1$ $?X2$ $?X5$, $H3$:($\neg$Col $?X3$ $?X4$ $?X5$) $\vdash$ _ $\Rightarrow$
let h := fresh in
not_exist_hyp_perm_pars $X1$ $X2$ $X3$ $X4$;
assert (h := par_not_col_strict $X1$ $X2$ $X3$ $X4$ $X5$ H $H2$ $H3$)

| H: $?X1 \neq ?X2$, $H2$:$?X2 \neq ?X3$, $H3$:Parallelogram $?X1$ $?X2$ $?X3$ $?X4$ $\vdash$ _ $\Rightarrow$
let h := fresh in
not_exist_hyp_perm_par $X1$ $X2$ $X3$ $X4$;

```
      assert (h := plg_par_1 X1 X2 X3 X4 H H2 H3)

      | H:Parallelogram_strict ?X1 ?X2 ?X3 ?X4 ⊢ _ ⇒
      let h := fresh in
      not_exist_hyp_perm_pars X1 X2 X3 X4;
      assert (h := plgs_pars_1 X1 X2 X3 X4 H)
  end.
```

```
Ltac assert_pars_2 :=
 repeat
 match goal with
      | H: ?X1 ≠ ?X2, H2:?X2 ≠ ?X3, H3:Parallelogram ?X1 ?X2 ?X3 ?X4 ⊢ _ ⇒
      let h := fresh in
      not_exist_hyp_perm_par X1 X4 X2 X3;
      assert (h := plg_par_2 X1 X2 X3 X4 H H2 H3)

      | H:Parallelogram_strict ?X1 ?X2 ?X3 ?X4 ⊢ _ ⇒
      let h := fresh in
      not_exist_hyp_perm_pars X1 X4 X2 X3;
      assert (h := plgs_pars_2 X1 X2 X3 X4 H)
  end.
```

Ltac *assert_pars* := *permutation_intro_in_hyps*; *assert_pars_1*; *assert_pars_2*; *clean_reap_hyps*.

Section Quadrilateral_inter_dec_3.

Context ‘{*MT*:**Tarski_2D_euclidean**}.
Context ‘{*EqDec*:**EqDecidability** Tpoint}.
Context ‘{*InterDec*:**InterDecidability** Tpoint Col}.

Lemma par_cong_cong : ∀ A B C D, Par A B C D → Cong A B C D → Cong A C B D ∨ Cong A D B C.

Lemma col_cong_cong : ∀ A B C D, Col A B C → Col A B D → Cong A B C D → Cong A C B D ∨ Cong A D B C.

Lemma par_cong_plg : ∀ A B C D, Par A B C D → Cong A B C D → Plg A B C D ∨ Plg A B D C.

Lemma par_cong_plg_2 : ∀ A B C D, Par A B C D → Cong A B C D → Parallelogram A B C D ∨ Parallelogram A B D C.

Lemma par_cong3_rect : ∀ A B C D, $A \neq C$ ∨ $B \neq D$ → Par A B C D → Cong A B C D → Cong A D B C → Cong A C B D → Rectangle A B C D ∨ Rectangle A B D C.

Lemma pars_par_pars : ∀ A B C D, Par_strict A B C D → Par A D B C → Par_strict A D B C.

Lemma pars_par_plg : ∀ A B C D, Par_strict A B C D → Par A D B C → Plg A B C D.

Lemma not_par_pars_not_cong : ∀ O A B A' B', out O A B → out O A' B' → Par_strict A A' B B' → ¬Cong A A' B B'.

Lemma plg_unicity : $\forall$ A B C D D', Parallelogram A B C D $\rightarrow$ Parallelogram A B C D' $\rightarrow$ $D = D'$.

Lemma plgs_trans_trivial : $\forall$ A B C D B', Parallelogram_strict A B C D $\rightarrow$ Parallelogram_strict C D A B'
$\rightarrow$ Parallelogram A B B' A.

Lemma par_strict_trans : $\forall$ A B C D E F, Par_strict A B C D $\rightarrow$ Par_strict C D E F $\rightarrow$ Par A B E F.

Lemma plgs_pseudo_trans : $\forall$ A B C D E F, Parallelogram_strict A B C D $\rightarrow$ Parallelogram_strict C D E F $\rightarrow$ Parallelogram A B F E.

Lemma plgf_plgs_trans : $\forall$ A B C D E F, $A \neq B$ $\rightarrow$ Parallelogram_flat A B C D $\rightarrow$ Parallelogram_strict C D E F $\rightarrow$ Parallelogram_strict A B F E.

Lemma plgf_plgf_plgf: $\forall$ A B C D E F, $A \neq B$ $\rightarrow$ Parallelogram_flat A B C D $\rightarrow$ Parallelogram_flat C D E F
$\rightarrow$ Parallelogram_flat A B F E.

Lemma plg_pseudo_trans : $\forall$ A B C D E F, Parallelogram A B C D $\rightarrow$ Parallelogram C D E F $\rightarrow$ Parallelogram A B F E $\lor$ $(A = B \land C = D \land E = F \land A = E)$.

Lemma Square_Rhombus : $\forall$ A B C D,
Square A B C D $\rightarrow$ Rhombus A B C D.

Lemma plgs_in_angle : $\forall$ A B C D, Parallelogram_strict A B C D $\rightarrow$ InAngle D A B C.

End Quadrilateral_inter_dec_3.

Chapter 25

Library triangles

```
Require Import Ch12_parallel.

Section Triangles.

Context `{MT:Tarski_2D_euclidean}.
Context `{EqDec:EqDecidability Tpoint}.

Section ABC.

Variable A B C : Tpoint.

Definition isosceles A B C :=
 Cong A B B C.

Lemma isosceles_sym :
  isosceles A B C →
  isosceles C B A.

Lemma isosceles_conga :
  A≠C → A ≠ B →
  isosceles A B C →
  Conga C A B A C B.

Definition equilateral A B C :=
 Cong A B B C ∧ Cong B C C A.

Definition equilateral_strict A B C :=
 equilateral A B C ∧ A ≠ B.

Lemma equilateral_strict_equilateral :
 equilateral_strict A B C →
 equilateral A B C.

Lemma equilateral_cong:
  equilateral A B C →
  Cong A B B C ∧ Cong B C C A ∧ Cong C A A B.

Lemma equilateral_rot :
```

```
  equilateral A B C ->
  equilateral B C A.
Lemma equilateral_swap :
  equilateral A B C ->
  equilateral B A C.
Lemma equilateral_rot_2 :
  equilateral A B C ->
  equilateral C B A.
Lemma equilateral_swap_2 :
  equilateral A B C ->
  equilateral A C B.
Lemma equilateral_swap_rot :
  equilateral A B C ->
  equilateral C A B.
Hint Resolve equilateral_swap equilateral_swap_2
  equilateral_swap_rot equilateral_rot equilateral_rot_2 : equilateral.
Lemma equilateral_isosceles_1 :
   equilateral A B C ->
   isosceles A B C.
Lemma equilateral_isosceles_2 :
   equilateral A B C ->
   isosceles B C A.
Lemma equilateral_isosceles_3 :
   equilateral A B C ->
   isosceles C A B.
Hint Resolve equilateral_isosceles_1 equilateral_isosceles_2 equilateral_isosceles_3 : equilateral.
Lemma equilateral_strict_neq :
  equilateral_strict A B C ->
  A <> B /\ B <> C /\ A <> C.
Hint Resolve equilateral_strict_neq : equilateral.
Lemma equilateral_strict_swap_1 :
  equilateral_strict A B C ->
  equilateral_strict A C B.
Lemma equilateral_strict_swap_2 :
  equilateral_strict A B C ->
  equilateral_strict B A C.
Lemma equilateral_strict_swap_3 :
  equilateral_strict A B C ->
```

equilateral_strict B C A.

Lemma equilateral_strict_swap_4 :
equilateral_strict A B C $\rightarrow$
equilateral_strict C A B.

Lemma equilateral_strict_swap_5 :
equilateral_strict A B C $\rightarrow$
equilateral_strict C B A.

Hint Resolve equilateral_strict_swap_1 equilateral_strict_swap_2
equilateral_strict_swap_3 equilateral_strict_swap_4 equilateral_strict_swap_5 : *equilateral.*

Lemma equilateral_strict_conga_1 :
equilateral_strict A B C $\rightarrow$
Conga C A B A C B.

End ABC.

Lemma equilateral_strict_conga_2 :
$\forall$ A B C,
equilateral_strict A B C $\rightarrow$
Conga B A C A B C.

Lemma equilateral_strict_conga_3 :
$\forall$ A B C,
equilateral_strict A B C $\rightarrow$
Conga C B A B C A.

End Triangles.

Chapter 26

Library perp_bisect

```
Require Export Ch12_parallel.

Section PerpBisect_1.

Context `{MT:Tarski_2D}.
Context `{EqDec:EqDecidability Tpoint}.
```

PQ is the perpendicular bisector of segment AB

```
Definition perp_bisect P Q A B :=
  ∃ I, Perp_in I P Q A B ∧ is_midpoint I A B.

Lemma perp_bisect_sym_1 :
 ∀ P Q A B,
  perp_bisect P Q A B →
  perp_bisect Q P A B.

Lemma perp_bisect_sym_2 :
 ∀ P Q A B,
  perp_bisect P Q A B →
  perp_bisect P Q B A.

Lemma perp_bisect_sym_3 : ∀ A B C D,
 perp_bisect A B C D →
 perp_bisect B A D C.

Lemma perp_in_per_1 :
 ∀ A B C D X,
  Perp_in X A B C D →
  Per A X C.

Lemma perp_in_per_2 :
 ∀ A B C D X,
  Perp_in X A B C D →
  Per A X D.

Lemma perp_in_per_3 :
```

```
 ∀ A B C D X,
  Perp_in X A B C D →
  Per B X C.

Lemma perp_in_per_4 :
 ∀ A B C D X,
  Perp_in X A B C D →
  Per B X D.

Lemma perp_bisect_perp :
 ∀ P Q A B,
  perp_bisect P Q A B →
  Perp P Q A B.

End PerpBisect_1.

Hint Resolve perp_in_per_1 perp_in_per_2 perp_in_per_3 perp_in_per_4 : perp.

Hint Resolve perp_bisect_perp : perp_bisect.

Section PerpBisect_2.

Context `{MT:Tarski_2D}.
Context `{EqDec:EqDecidability Tpoint}.

Lemma perp_bisect_cong_1 :
 ∀ P Q A B,
 perp_bisect P Q A B →
 Cong A P B P.

Lemma perp_bisect_cong_2 :
 ∀ P Q A B,
 perp_bisect P Q A B →
 Cong A Q B Q.

End PerpBisect_2.

Hint Resolve perp_bisect_cong_1 perp_bisect_cong_2 : perp_bisect.

Section PerpBisect_3.

Context `{MT:Tarski_2D}.
Context `{EqDec:EqDecidability Tpoint}.

Lemma perp_bisect_cong :
 ∀ P Q A B,
 perp_bisect P Q A B →
 Cong A P B P ∧ Cong A Q B Q.

Lemma perp_bisect_conc :
 ∀ A A1 B B1 C C1 O: Tpoint,
 ¬ Col A B C →
 perp_bisect O A1 B C → perp_bisect O B1 A C → perp_bisect O C1 A B →
```

```
  Cong A O B O → Cong B O C O →
  Cong A O C O.

Lemma cong_perp_bisect :
 ∀ P Q A B,
 P ≠ Q → A ≠ B →
 Cong A P B P →
 Cong A Q B Q →
 perp_bisect P Q A B.

Definition is_on_perp_bisect P A B := Cong A P P B.

Lemma perp_bisect_is_on_perp_bisect :
 ∀ A B C P,
  is_on_perp_bisect P A B →
  is_on_perp_bisect P B C →
  is_on_perp_bisect P A C.

Lemma perp_mid_perp_bisect : ∀ A B C D,
 is_midpoint C A B → Perp C D A B →
 perp_bisect C D A B.

End PerpBisect_3.

Section Euclid.

Context `{MT:Tarski_2D_euclidean}.
Context `{EqDec:EqDecidability Tpoint}.

Lemma triangle_circumscription_implies_inter_dec :
  (∀ A B C, ¬ Col A B C → ∃ CC, Cong A CC B CC ∧ Cong A CC C CC) →
  ∀ A B C D, (∃ I, Col I A B ∧ Col I C D) ∨ ¬ (∃ I, Col I A B ∧ Col I C D).
End Euclid.
```

Chapter 27

Library aux

We circumvent a limitation of type class definition by defining a polymorphic type for a triple of elements which will be used to define an angle by instantiating A with Point

Record **Triple** $\{A$:Type$\}$: Type := build_triple $\{$V1 : A ; V : A ; V2 : A ; Pred : V1 $\neq$ V $\wedge$ V2 $\neq$ V$\}$.

Record **Couple** $\{A$:Type$\}$: Type := build_couple $\{$P1: A ; P2 : A ; Cond: P1 $\neq$ P2$\}$.

Chapter 28

Library general_tactics

```
Ltac ex_elim H x := elim H; intros x ; intro; clear H.

Ltac DecompEx H P := elim H;intro P;intro;clear H.

Ltac DecompExAnd H P :=
  elim H;intro P;let id:=fresh in
(intro id;decompose [and] id;clear id;clear H).

Ltac exist_hyp t := match goal with
  | H1:t ⊢ _ ⇒ idtac
 end.

Ltac hyp_of_type t := match goal with
  | H1:t ⊢ _ ⇒ H1
end.

Ltac clean_duplicated_hyps :=
  repeat match goal with
      | H:?X1 ⊢ _ ⇒ clear H; exist_hyp X1
end.

Ltac suppose H := cut H;[intro|idtac].

Ltac not_exist_hyp t := match goal with
  | H1:t ⊢ _ ⇒ fail 2
 end || idtac.

Ltac DecompAndAll :=
 repeat
 match goal with
   | H:(?X1 ∧ ?X2) ⊢ _ ⇒ decompose [and] H;clear H
end.

Ltac assert_if_not_exist H :=
  not_exist_hyp H;assert H.

Ltac absurde :=
```

```
match goal with
   |H : (?X ≠ ?X) ⊢ _ ⇒ apply False_ind; apply H; reflexivity
end.

Ltac spliter := repeat
match goal with
   | H:(?X1 ∧ ?X2) ⊢ _ ⇒ induction H
end.

Ltac ex_and H x := elim H; intro x; intros; clear H;spliter.

Ltac use H := decompose [and] H;clear H.

Ltac try_or T :=
 match goal with
 | ⊢ ?A ∨ ?B ⇒
   (left; try_or T) || (right;try_or T)
 | ⊢ _ ⇒ T
 end.

Tactic Notation "generalizes" hyp(X) :=
  generalize X; clear X.

Ltac sort_tactic :=
  try match goal with H: ?T ⊢ _ ⇒
  match type of T with Prop ⇒
    generalizes H; (try sort_tactic); intro
  end end.

Tactic Notation "sort" :=
  sort_tactic.

Definition ltac_something (P:Type) (e:P) := e.

Notation "'Something'" :=
  (@ltac_something _ _).

Lemma ltac_something_eq : ∀ (e:Type),
  e = (@ltac_something _ e).

Lemma ltac_something_hide : ∀ (e:Type),
  e → (@ltac_something _ e).

Lemma ltac_something_show : ∀ (e:Type),
  (@ltac_something _ e) → e.

Tactic Notation "hide_def" hyp(x) :=
  let x' := constr:(x) in
  let T := eval unfold x in x' in
  change T with (@ltac_something _ T) in x.

Tactic Notation "show_def" hyp(x) :=
```

```
  let x' := constr:(x) in
  let U := eval unfold x in x' in
  match U with @ltac_something _ ?T ⇒
    change U with T in x end.

Tactic Notation "show_def" :=
  unfold ltac_something.

Tactic Notation "show_def" "in" "*" :=
  unfold ltac_something in *.

Tactic Notation "hide_defs" :=
  repeat match goal with H := ?T ⊢ _ ⇒
    match T with
    | @ltac_something _ _ ⇒ fail 1
    | _ ⇒ change T with (@ltac_something _ T) in H
    end
  end.

Tactic Notation "show_defs" :=
  repeat match goal with H := (@ltac_something _ ?T) ⊢ _ ⇒
    change (@ltac_something _ T) with T in H end.

Tactic Notation "show_hyp" hyp(H) :=
  apply ltac_something_show in H.

Tactic Notation "hide_hyp" hyp(H) :=
  apply ltac_something_hide in H.

Tactic Notation "show_hyps" :=
  repeat match goal with
    H: @ltac_something _ _ ⊢ _ ⇒ show_hyp H end.

Tactic Notation "hide_hyps" :=
  repeat match goal with H: ?T ⊢ _ ⇒
    match type of T with
    | Prop ⇒
      match T with
      | @ltac_something _ _ ⇒ fail 2
      | _ ⇒ hide_hyp H
      end
    | _ ⇒ fail 1
    end
  end.

Tactic Notation "hide" hyp(H) :=
  first [hide_def H | hide_hyp H].

Tactic Notation "show" hyp(H) :=
```

```
  first [show_def H | show_hyp H].
Tactic Notation "hide_all" :=
  hide_hyps; hide_defs.
Tactic Notation "show_all" :=
  unfold ltac_something in *.
Tactic Notation "hide_term" constr(E) :=
  change E with (@ltac_something _ E).
Tactic Notation "show_term" constr(E) :=
  change (@ltac_something _ E) with E.
Tactic Notation "show_term" :=
  unfold ltac_something.
Tactic Notation "hide_term" constr(E) "in" hyp(H) :=
  change E with (@ltac_something _ E) in H.
Tactic Notation "show_term" constr(E) "in" hyp(H) :=
  change (@ltac_something _ E) with E in H.
Tactic Notation "show_term" "in" hyp(H) :=
  unfold ltac_something in H.
```

Chapter 29

Library circumcenter

```
Require Export perp_bisect.

Section Circumcenter.

Context `{MT:Tarski_2D_euclidean}.
Context `{EqDec:EqDecidability Tpoint}.

Definition is_circumcenter G A B C := Cong A G B G ∧ Cong B G C G.

Lemma circumcenter_cong : ∀ G A B C,
 is_circumcenter G A B C →
 Cong A G B G ∧ Cong B G C G ∧ Cong C G A G.

Lemma circumcenter_perp : ∀ A B C A' B' C' G,
  A≠B → B≠C → A≠C → G ≠ A' → G ≠ B' → G ≠ C' →
  is_circumcenter G A B C →
  is_midpoint A' B C →
  is_midpoint B' A C →
  is_midpoint C' A B →
  perp_bisect G A' B C ∧ perp_bisect G B' A C ∧ perp_bisect G C' A B.

Lemma circumcenter_intersect : ∀ A B C A' B' C' G,
  A≠B → B≠C → A≠C → G ≠ A' → G ≠ B' → G ≠ C' →
  is_midpoint A' B C →
  is_midpoint B' A C →
  is_midpoint C' A B →
  perp_bisect G A' B C →
  perp_bisect G B' A C →
  Perp G C' A B.

End Circumcenter.
```

Chapter 30

Library orientation

Require Import Ch12_parallel_inter_dec.
Require Import quadrilaterals.

Section Ch12.

Context ‘{MT:**Tarski_2D_euclidean**}.
Context ‘{$EqDec$:**EqDecidability** Tpoint}.
Context ‘{$InterDec$:**InterDecidability** Tpoint Col}.

Definition proj := fun T A B P ⇒ A ≠ B ∧ (¬Col A B T ∧ Perp A B T P ∧ Col A B P ∨ Col A B T ∧ P = T).

Lemma proj_exists : ∀ A B T, A ≠ B → ∃ P, proj T A B P.

Lemma proj_per : ∀ A B T P, A ≠ B → proj T A B P → Per T P A ∧ Per T P B ∧ Col A B P.

Lemma proj_unicity : ∀ A B T P P’, proj T A B P → proj T A B P’ → P = P’.

Lemma proj_col : ∀ T P A B, proj T A B P → Col P A B.

Lemma proj_col_proj : ∀ A B C T P, proj T A B P → A ≠ C → Col A B C → proj T A C P.

Lemma per_proj : ∀ A B T P, A ≠ B → Per T P A → Per T P B → Col A B P → proj T A B P.

Definition eqo := fun A B P $A1$ $B1$ $P1$ ⇒ ¬Col A B P ∧ ¬Col $A1$ $B1$ $P1$ ∧
∀ C $C1$ $B2$ M B’ C’ K,
Perp A B C A → Per P C A → Perp $A1$ $B1$ $C1$ $A1$ → Per $P1$ $C1$ $A1$ →
out $A1$ $B1$ $B2$ → Cong A B $A1$ $B2$ →
is_midpoint M A $A1$ → is_midpoint M $B2$ B’ → is_midpoint M $C1$ C’ → is_midpoint K B B’ →
Bet C A C’ ∨ one_side A K C C’.

Definition eq_o := fun A B P $A1$ $B1$ $P1$ ⇒ ¬Col A B P ∧ ¬Col $A1$ $B1$ $P1$ ∧
∀ C $C1$ $B2$ M B’ C’ K,

Perp A B C A $\rightarrow$ proj P A C C $\rightarrow$ Perp $A1$ $B1$ $C1$ $A1$ $\rightarrow$ proj $P1$ $A1$ $C1$ $C1$ $\rightarrow$
out $A1$ $B1$ $B2$ $\rightarrow$ Cong A B $A1$ $B2$ $\rightarrow$
is_midpoint M A $A1$ $\rightarrow$ is_midpoint M $B2$ B' $\rightarrow$ is_midpoint M $C1$ C' $\rightarrow$ is_midpoint K B B' $\rightarrow$
Bet C A C' $\vee$ one_side A K C C'.

Lemma eqo_eq_o : $\forall$ A B P $A1$ $B1$ $P1$, eqo A B P $A1$ $B1$ $P1$ $\rightarrow$ eq_o A B P $A1$ $B1$ $P1$.

Lemma eq_o_eqo : $\forall$ A B P $A1$ $B1$ $P1$, eq_o A B P $A1$ $B1$ $P1$ $\rightarrow$ eqo A B P $A1$ $B1$ $P1$.

Lemma eq_o_one_side : $\forall$ A B X Y, eq_o A B X A B Y $\rightarrow$ one_side A B X Y.

Lemma eqo_one_side : $\forall$ A B X Y, eqo A B X A B Y $\rightarrow$ one_side A B X Y.

Lemma eq_o_refl : $\forall$ A B P, $\neg$Col A B P $\rightarrow$ eq_o A B P A B P.

Lemma eqo_refl : $\forall$ A B P, $\neg$Col A B P $\rightarrow$ eqo A B P A B P.

Lemma per_id : $\forall$ A B B' C, $A \neq B$ $\rightarrow$ $B \neq C$ $\rightarrow$ $B' \neq C$ $\rightarrow$ Per A B C $\rightarrow$ Per A B' C $\rightarrow$ Col C B B' $\rightarrow$ $B = B'$.

Lemma proj_one_side : $\forall$ A B A' B' P Q, $A \neq A'$ $\rightarrow$ proj A P Q A' $\rightarrow$ proj B P Q B' $\rightarrow$ Col B A A' $\vee$ one_side A A' B B'.

Lemma proj_eq_col : $\forall$ A B P Q C, proj A P Q C $\rightarrow$ proj B P Q C $\rightarrow$ Col A B C.

Lemma proj_par : $\forall$ A B A' B' P Q, $A \neq A'$ $\rightarrow$ $B \neq B'$ $\rightarrow$ proj A P Q A' $\rightarrow$ proj B P Q B' $\rightarrow$ Par A A' B B'.

Lemma proj_not_col : $\forall$ A A' P Q, $A \neq A'$ $\rightarrow$ proj A P Q A' $\rightarrow$ $\neg$Col P Q A.

Lemma proj_comm : $\forall$ A B P Q, proj A P Q B $\rightarrow$ proj A Q P B.

Lemma proj_not_eq : $\forall$ A B A' B' P Q, $A' \neq B'$ $\rightarrow$ proj A P Q A' $\rightarrow$ proj B P Q B' $\rightarrow$ $A \neq B$.

Lemma proj_not_eq_not_col : $\forall$ A B A' B' P Q, $A' \neq B'$ $\rightarrow$ $A \neq A'$ $\rightarrow$ proj A P Q A' $\rightarrow$ proj B P Q B' $\rightarrow$ $\neg$Col A A' B'.

Lemma proj_par_strict : $\forall$ A B A' B' P Q, $A \neq A'$ $\rightarrow$ $B \neq B'$ $\rightarrow$ $A' \neq B'$ $\rightarrow$ proj A P Q A' $\rightarrow$ proj B P Q B' $\rightarrow$ Par_strict A A' B B'.

Lemma col_proj_col : $\forall$ A B A' B' P Q, $A \neq A'$ $\rightarrow$ Col A B A' $\rightarrow$ proj A P Q A' $\rightarrow$ proj B P Q B' $\rightarrow$ Col A B B'.

Lemma col_proj_proj : $\forall$ A B A' P Q, $A \neq A'$ $\rightarrow$ Col A B A' $\rightarrow$ proj A P Q A' $\rightarrow$ proj B P Q A'.

Lemma proj_id : $\forall$ A B A' B' P Q, $A \neq A'$ $\rightarrow$ Col A B A' $\rightarrow$ proj A P Q A' $\rightarrow$ proj B P Q B' $\rightarrow$ $A' = B'$.

Lemma proj_diff : $\forall$ A P Q A' , proj A P Q A' $\rightarrow$ $P \neq Q$.

Lemma proj3_col : $\forall$ A B C A' B' C' P Q , proj A P Q A' $\rightarrow$ proj B P Q B' $\rightarrow$ proj C P Q C' $\rightarrow$ Col A' B' C'.

Lemma proj3_id : ∀ A B C C' P Q, $A \neq B$ → Col A B C → proj A P Q A → proj B P Q B → proj C P Q C' → $C = C'$.

Lemma proj_inv_exists : ∀ P Q A', $P \neq Q$ → Col P Q A' → ∃ A, $A \neq A'$ ∧ proj A P Q A'.

Lemma proj_perp_id : ∀ A B C A' B' P Q, $A \neq C$ → Col A B C →
proj A P Q A' → proj B P Q B' → proj C P Q A' →
$A' = B'$.

Lemma proj_diff_not_col : ∀ A B A' B' P Q, $A \neq A'$ → proj A P Q A' → proj B P Q B' → ($A' \neq B'$ ↔ ¬Col A B A').

Lemma proj_diff_not_col_inv : ∀ A B A' B' P Q, $A \neq A'$ → proj A P Q A' → proj B P Q B' → ($A' = B'$ ↔ Col A B A').

Lemma proj_preserves_bet1 : ∀ A B C B' C' P Q, Bet A B C →
proj A P Q A → proj B P Q B' → proj C P Q C' →
Bet A B' C'.

Lemma proj_preserves_bet : ∀ A B C A' B' C' P Q, Bet A B C →
proj A P Q A' → proj B P Q B' → proj C P Q C' →
Bet A' B' C'.

Lemma one_side_eq_o : ∀ A B C D, $A \neq B$ → one_side A B C D → eq_o A B C A B D.

Lemma out_preserves_eq_o : ∀ A B B' P, ¬Col A B P → out A B B' → eq_o A B P A B' P.

Lemma cong_identity_inv : ∀ A B C, $A \neq B$ → ¬Cong A B C C.

Lemma midpoint_col : ∀ A B A' B' M, $A \neq B$ → is_midpoint M A A' → is_midpoint M B B' → Col A B B' → $A' \neq B'$ ∧ Col A A' B' ∧ Col B A' B'.

Lemma midpoint_par : ∀ A B A' B' M, $A \neq B$ → is_midpoint M A A' → is_midpoint M B B' → Par A B A' B'.

Lemma midpoint_par_strict : ∀ A B A' B' M, $A \neq B$ → ¬Col A B B' → is_midpoint M A A' → is_midpoint M B B' → Par_strict A B A' B'.

Lemma le_left_comm : ∀ A B C D, le A B C D → le B A C D.

Lemma le_right_comm : ∀ A B C D, le A B C D → le A B D C.

Lemma le_comm : ∀ A B C D, le A B C D → le B A D C.

Lemma le_cong_le : ∀ A B C A' B' C', Bet A B C → Bet A' B' C' → le A B A' B' → Cong B C B' C' → le A C A' C'.

Lemma cong_le_le : ∀ A B C A' B' C', Bet A B C → Bet A' B' C' → le B C B' C' → Cong A B A' B' → le A C A' C'.

Lemma bet_le_le : ∀ A B C A' B' C', Bet A B C → Bet A' B' C' → le A B A' B' → le B C B' C' → le A C A' C'.

Lemma bet_double_bet : $\forall$ A B C B' C', is_midpoint B' A B $\rightarrow$ is_midpoint C' A C $\rightarrow$ Bet A B' C' $\rightarrow$ Bet A B C.

Lemma bet_half_bet : $\forall$ A B C B' C', Bet A B C $\rightarrow$ is_midpoint B' A B $\rightarrow$ is_midpoint C' A C $\rightarrow$ Bet A B' C'.

Lemma midpoint_preserves_bet : $\forall$ A B C B' C', is_midpoint B' A B $\rightarrow$ is_midpoint C' A C $\rightarrow$ (Bet A B C $\leftrightarrow$ Bet A B' C').

Lemma symmetry_preseves_bet1 : $\forall$ A B M A' B', is_midpoint M A A' $\rightarrow$ is_midpoint M B B' $\rightarrow$ Bet M A B $\rightarrow$ Bet M A' B'.

Lemma symmetry_preseves_bet2 : $\forall$ A B M A' B', is_midpoint M A A' $\rightarrow$ is_midpoint M B B' $\rightarrow$ Bet M A' B' $\rightarrow$ Bet M A B.

Lemma symmetry_preserves_bet : $\forall$ A B M A' B', is_midpoint M A A' $\rightarrow$ is_midpoint M B B' $\rightarrow$ (Bet M A' B' $\leftrightarrow$ Bet M A B).

Lemma par_cong_mid : $\forall$ A B A' B', Par A B A' B' $\rightarrow$ Cong A B A' B' $\rightarrow$ $\exists$ M , is_midpoint M A A' $\wedge$ is_midpoint M B B' $\vee$ is_midpoint M A B' $\wedge$ is_midpoint M B A'.

Lemma per_preserves_bet_aux1 : $\forall$ P Q A B C B' C', $P \neq Q$ $\rightarrow$ Bet A B C $\rightarrow$
Col P Q A $\rightarrow$
Per B B' P $\rightarrow$ Col P Q B' $\rightarrow$
Per C C' P $\rightarrow$ Col P Q C' $\rightarrow$
$P \neq A \rightarrow P \neq B' \rightarrow P \neq C' \rightarrow$
Bet A B' C'.

Lemma perp_not_eq_3 : $\forall$ A B C, Perp A B B C $\rightarrow$ $A \neq C$.

Lemma per_preserves_bet_aux2 : $\forall$ P Q A B C A' C', $P \neq Q$ $\rightarrow$ Bet A B C $\rightarrow$
Per A A' P $\rightarrow$ Col P Q A' $\rightarrow$
Col P Q B $\rightarrow$
Per C C' P $\rightarrow$ Col P Q C' $\rightarrow$
$P \neq A' \rightarrow P \neq B \rightarrow P \neq C' \rightarrow$
Bet A' B C'.

Lemma par_col : $\forall$ A B C, Par A B A C $\rightarrow$ Col A B C.

Lemma per_diff : $\forall$ A B A' B' P, $A \neq B$ $\rightarrow$ $\neg$ Col A B A' $\rightarrow$
Per A A' P $\rightarrow$ Per B B' P $\rightarrow$
$A' \neq P \rightarrow B' \neq P \rightarrow A' \neq B'$.

Lemma per_preserves_bet : $\forall$ P Q A B C A' B' C', $P \neq Q$ $\rightarrow$ Bet A B C $\rightarrow$
Per A A' P $\rightarrow$ Col P Q A' $\rightarrow$
Per B B' P $\rightarrow$ Col P Q B' $\rightarrow$
Per C C' P $\rightarrow$ Col P Q C' $\rightarrow$
$P \neq A' \rightarrow P \neq B' \rightarrow P \neq C' \rightarrow$
Bet A' B' C'.

Lemma ex_col : ∀ A B C, Distincts A B C → Col A B C → ∃ D, Col A B D ∧ $A \neq D$ ∧ $B \neq D$ ∧ $C \neq D$.

Lemma out_preserves_eqo1 : ∀ A B P B', ¬Col A B P → out A B B' → eqo A B P A B' P.

Lemma out_preserves_eqo : ∀ A B P B' P', ¬Col A B P → out A B B' → out A P P' → eqo A B P A B' P'.

Lemma per_one_side : ∀ A B P Q C C', $A \neq P$ → $C' \neq P$ → ¬Col A B C → Col P Q A → Col P Q C' → Perp A B P Q → Per C C' P → one_side A B C C'.

Lemma one_side_eqo : ∀ A B X Y, one_side A B X Y → eqo A B X A B Y.

Lemma ex_col1 : ∀ A B C, $A \neq B$ → Col A B C → ∃ D, Col A B D ∧ $A \neq D$ ∧ $B \neq D$ ∧ $C \neq D$.

End Ch12.

Chapter 31

Library project

Require Export quadrilaterals_inter_dec.
Require Export vectors.

Section Projections.

Context `{MT:**Tarski_2D_euclidean**}.
Context `{$EqDec$:**EqDecidability** Tpoint}.
Context `{$InterDec$:**InterDecidability** Tpoint Col}.

Projections

Definition project P P' A B X Y :=
$A \neq B$ ∧ $X \neq Y$ ∧ ¬Par A B X Y ∧ Col A B P' ∧ (Par P P' X Y ∨ P = P').

Lemma project_id : ∀ A B X Y P P', project P P' A B X Y → Col A B P → P = P'.

Lemma project_not_id : ∀ A B X Y P P', project P P' A B X Y → ¬Col A B P → $P \neq P'$.

Lemma project_col : ∀ A B X Y P , project P P A B X Y → Col A B P.

Lemma project_not_col : ∀ A B X Y P P', project P P' A B X Y → $P \neq P'$ → ¬Col A B P.

Lemma project_par : ∀ A B X Y P Q P' Q', project P P' A B X Y → project Q Q' A B X Y → Par P Q X Y → P' = Q'.

Lemma ker_col : ∀ P Q P' A B X Y, project P P' A B X Y → project Q P' A B X Y → Col P Q P'.

Lemma ker_par : ∀ P Q P' A B X Y, $P \neq Q$ → project P P' A B X Y → project Q P' A B X Y → Par P Q X Y.

Lemma project_unicity : ∀ P P' Q' A B X Y, project P P' A B X Y → project P Q' A B X Y → P' = Q'.

Lemma project_existence : ∀ P A B X Y,
$X \neq Y$ → $A \neq B$ →
¬ Par X Y A B →
∃! P', project P P' A B X Y.

```
Lemma project_col_eq : ∀ P Q P' Q' A B X Y,
 P ≠ P' →
 Col P Q P' →
 project P P' A B X Y →
 project Q Q' A B X Y →
 P' = Q'.

Lemma par_col_project :
 ∀ P P' A B X Y ,
  A ≠ B →
  ¬ Par A B X Y →
  Par P P' X Y →
  Col A B P' →
 project P P' A B X Y.

Lemma project_preserves_bet :
 ∀ A B X Y P Q R P' Q' R',
  Bet P Q R →
  project P P' A B X Y →
  project Q Q' A B X Y →
  project R R' A B X Y →
  Bet P' Q' R'.

Lemma symmetry_preserves_conga :
 ∀ A B C A' B' C' M, A ≠ B → C ≠ B →
  is_midpoint M A A' →
  is_midpoint M B B' →
  is_midpoint M C C' →
  Conga A B C A' B' C'.

Lemma triangle_par :
 ∀ A B C A' B' C',
 ¬ Col A B C →
  Par A B A' B' →
  Par B C B' C' →
  Par A C A' C' →
 Conga A B C A' B' C'.

Definition Conga_3 := fun A B C A' B' C' ⇒ Conga A B C A' B' C' ∧ Conga B C A
B' C' A' ∧ Conga C A B C' A' B'.

Lemma par3_conga3 :
 ∀ A B C A' B' C',
 ¬ Col A B C →
 Par A B A' B' →
 Par B C B' C' →
 Par A C A' C' →
```

Conga_3 A B C A' B' C'.

Lemma cong_conga3_cong3 :
∀ A B C A' B' C',
¬ Col A B C →
Cong A B A' B' →
Conga_3 A B C A' B' C' →
Cong_3 A B C A' B' C'.

Lemma project_par_eqv :
∀ P P' Q Q' A B X Y,
project P P' A B X Y →
project Q Q' A B X Y →
Par P Q A B →
eqV P Q P' Q'.

Lemma eqv_project_eq_eq :
∀ P Q R S P' Q' S' A B X Y,
eqV P Q R S →
project P P' A B X Y →
project Q Q' A B X Y →
project R P' A B X Y →
project S S' A B X Y →
Q' = S'.

Lemma eqv_eq_project :
∀ P Q R S P' Q' A B X Y,
eqV P Q R S →
project P P' A B X Y →
project Q Q' A B X Y →
project R P' A B X Y →
project S Q' A B X Y.

Lemma project_par_dir : ∀ P P' A B X Y, P ≠ P' → project P P' A B X Y → Par P P' X Y.

Lemma project_idem : ∀ P P' A B X Y, project P P' A B X Y → project P' P' A B X Y.

Lemma eqv_cong : ∀ A B C D, eqV A B C D → Cong A B C D.

Lemma project_preserves_eqv :
∀ P Q R S P' Q' R' S' A B X Y, eqV P Q R S →
project P P' A B X Y →
project Q Q' A B X Y →
project R R' A B X Y →
project S S' A B X Y →
eqV P' Q' R' S'.

Lemma perp_vector : ∀ A B, A ≠ B → (∃ X , ∃ Y , Perp A B X Y).

Definition projp := fun P P' A B ⇒ $A \neq B$ ∧ ((Col A B P' ∧ Perp A B P P') ∨ (Col A B P ∧ $P = P'$)).

Lemma perp_projp : ∀ P P' A B, Perp_in P' A B P P' → projp P P' A B.

Lemma proj_distinct : ∀ P P' A B, projp P P' A B → $P' \neq A$ ∨ $P' \neq B$.

Lemma projp_is_project :
 ∀ P P' A B,
  projp P P' A B →
  ∃ X, ∃ Y, project P P' A B X Y.

Lemma projp_is_project_perp :
 ∀ P P' A B,
 projp P P' A B →
 ∃ X, ∃ Y, project P P' A B X Y ∧ Perp A B X Y.

Lemma projp_to_project :
 ∀ P P' A B X Y,
  Perp A B X Y →
  projp P P' A B →
  project P P' A B X Y.

Lemma project_to_projp :
 ∀ P P' A B X Y,
  project P P' A B X Y →
  Perp A B X Y →
  projp P P' A B.

Lemma projp_project_to_perp :
 ∀ P P' A B X Y,
 $P \neq P'$ →
 projp P P' A B →
 project P P' A B X Y →
 Perp A B X Y.

Lemma project_par_project :
 ∀ P P' A B X Y X' Y',
  project P P' A B X Y →
  Par X Y X' Y' →
  project P P' A B X' Y'.

Lemma project_project_par :
 ∀ P P' A B X Y X' Y',
 $P \neq P'$ →
 project P P' A B X Y →
 project P P' A B X' Y' →
 Par X Y X' Y'.

Lemma projp_id : ∀ P P' Q' A B, projp P P' A B → projp P Q' A B → $P' = Q'$.

Lemma projp_preserves_bet :
$\forall$ A B C A' B' C' X Y,
Bet A B C $\rightarrow$
projp A A' X Y $\rightarrow$
projp B B' X Y $\rightarrow$
projp C C' X Y $\rightarrow$
Bet A' B' C'.

Lemma projp_preserves_eqv :
$\forall$ A B C A' B' C' D D' X Y,
eqV A B C D $\rightarrow$
projp A A' X Y $\rightarrow$
projp B B' X Y $\rightarrow$
projp C C' X Y $\rightarrow$
projp D D' X Y $\rightarrow$
eqV A' B' C' D'.

Lemma projp_idem : $\forall$ P P' A B,
projp P P' A B $\rightarrow$
projp P' P' A B.

End Projections.

Chapter 32

Library vectors

```
Require Import quadrilaterals_inter_dec.
Section Vectors.
Context `{MT:Tarski_2D_euclidean}.
Context `{EqDec:EqDecidability Tpoint}.
Context `{InterDec:InterDecidability Tpoint Col}.
```

Vertor

```
Definition eqV := fun A B C D ⇒ Parallelogram A B D C ∨ A = B ∧ C = D.

Lemma eqv_refl : ∀ A B, eqV A B A B.

Lemma eqv_sym : ∀ A B C D, eqV A B C D → eqV C D A B.

Lemma eqv_trans : ∀ A B C D E F, eqV A B C D → eqV C D E F → eqV A B E F.

Lemma eqv_comm : ∀ A B C D, eqV A B C D → eqV B A D C.

Lemma vector_construction : ∀ A B C, ∃ D, eqV A B C D.

Lemma vector_construction_unicity :
 ∀ A B C D D',
 eqV A B C D →
 eqV A B C D' →
 D = D'.

Lemma null_vector : ∀ A B C, eqV A A B C → B = C.

Lemma vector_unicity : ∀ A B C, eqV A B A C → B = C.

Lemma eqv_trivial : ∀ A B , eqV A A B B.

Lemma eqv_permut :
  ∀ A B C D,
  eqV A B C D →
  eqV A C B D.

Lemma eqv_par :
```

```
 ∀ A B C D,
  A ≠ B →
  eqV A B C D →
  Par A B C D.
Lemma eqv_opp_null :
  ∀ A B,
  eqV A B B A →
  A = B.
Lemma eqv_sum :
  ∀ A B C A' B' C',
  eqV A B A' B' →
  eqV B C B' C' →
  eqV A C A' C'.
Definition is_sum := fun A B C D E F ⇒ ∀ D', eqV C D B D' → eqV A D' E F.
Lemma null_sum :
 ∀ A B C,
  is_sum A B B A C C.
Lemma chasles :
 ∀ A B C,
  is_sum A B B C A C.
Lemma eqv_mid :
 ∀ A B C,
  eqV A B B C →
  is_midpoint B A C.
Lemma mid_eqv :
  ∀ A B C, is_midpoint A B C →
  eqV B A A C.
Lemma sum_sym :
  ∀ A B C D E F,
  is_sum A B C D E F →
  is_sum C D A B E F.
Lemma opposite_sum :
  ∀ A B C D E F,
  is_sum A B C D E F →
  is_sum B A D C F E.
Lemma null_sum_eq :
  ∀ A B C D,
  is_sum A B B C D D →
  A = C.
Definition is_sum_exists := fun A B C D E F ⇒ ∃ D', eqV B D' C D ∧ eqV A D' E F.
```

Lemma is_to_ise :
  ∀ A B C D E F,
  is_sum A B C D E F →
  is_sum_exists A B C D E F.

Lemma ise_to_is :
 ∀ A B C D E F,
  is_sum_exists A B C D E F →
  is_sum A B C D E F.

Lemma sum_exists :
 ∀ A B C D, ∃ E , ∃ F , is_sum A B C D E F.

Lemma sum_unicity :
 ∀ A B C D E F E' F',
 is_sum A B C D E F →
 is_sum A B C D E' F' →
 eqV E F E' F'.

Definition same_dir := fun A B C D ⇒ A = B ∧ C = D ∨ ∃ D' , out C D D' ∧ eqV A B C D'.

Lemma same_dir_refl : ∀ A B, same_dir A B A B.

Lemma same_dir_ts :
  ∀ A B C D,
  same_dir A B C D →
  ∃ M , Bet A M D ∧ Bet B M C.

Lemma one_side_col_out :
 ∀ A B X Y,
  Col A X Y →
  one_side A B X Y →
  out A X Y.

Lemma par_ts_same_dir :
 ∀ A B C D, Par_strict A B C D →
 (∃ M , Bet A M D ∧ Bet B M C) →
 same_dir A B C D.

Lemma same_dir_out : ∀ A B C, same_dir A B A C → out A B C ∨ A = B ∧ A = C.

Lemma same_dir_out1 : ∀ A B C, same_dir A B B C → out A B C ∨ A = B ∧ A = C.

Lemma same_dir_null : ∀ A B C, same_dir A A B C → B = C.

Lemma plgs_out_plgs :
 ∀ A B C D B' C',
 Parallelogram_strict A B C D →
 out A B B' →
 out D C C' →

Cong A B' D C' →
Parallelogram_strict A B' C' D.

Lemma plgs_plgs_bet :
∀ A B C D B' C',
Parallelogram_strict A B C D →
Bet A B B' →
Parallelogram_strict A B' C' D →
Bet D C C'.

Lemma plgf_plgf_bet :
∀ A B C D B' C',
Parallelogram_flat A B C D →
Bet A B B' →
Parallelogram_flat A B' C' D →
Bet D C C'.

Lemma plg_plg_bet :
∀ A B C D B' C',
Parallelogram A B C D →
Bet A B B' →
Parallelogram A B' C' D →
Bet D C C'.

Lemma plgf_out_plgf :
∀ A B C D B' C',
Parallelogram_flat A B C D →
out A B B' →
out D C C' →
Cong A B' D C' →
Parallelogram_flat A B' C' D.

Lemma plg_out_plg :
∀ A B C D B' C',
Parallelogram A B C D →
out A B B' →
out D C C' →
Cong A B' D C' →
Parallelogram A B' C' D.

Lemma same_dir_sym : ∀ A B C D, same_dir A B C D → same_dir C D A B.

Lemma same_dir_trans : ∀ A B C D E F, same_dir A B C D → same_dir C D E F → same_dir A B E F.

Lemma same_dir_comm : ∀ A B C D, same_dir A B C D → same_dir B A D C.

Lemma bet_same_dir1 : ∀ A B C, $A \neq B$ → $B \neq C$ → Bet A B C → same_dir A B A C.

Lemma bet_same_dir2 : ∀ A B C, $A \neq B$ → $B \neq C$ → Bet A B C → same_dir A B B C.

Definition opp_dir := fun A B C D ⇒ same_dir A B D C.

Lemma plg_opp_dir : ∀ A B C D, Parallelogram A B C D → same_dir A B D C.

Lemma same_dir_dec : ∀ A B C D,
  same_dir A B C D ∨ ¬ same_dir A B C D.

Lemma same_or_opp_dir : ∀ A B C D, Par A B C D → same_dir A B C D ∨ opp_dir A B C D.

Lemma same_dir_id : ∀ A B, same_dir A B B A → A = B.

Lemma opp_dir_id : ∀ A B, opp_dir A B A B → A = B.

Lemma same_dir_to_null : ∀ A B C D, same_dir A B C D → same_dir A B D C → A = B ∧ C = D.

Lemma opp_dir_to_null : ∀ A B C D, opp_dir A B C D → opp_dir A B D C → A = B ∧ C = D.

Lemma same_not_opp_dir : ∀ A B C D, A ≠ B → same_dir A B C D → ¬opp_dir A B C D.

Lemma opp_not_same_dir : ∀ A B C D, A ≠ B → opp_dir A B C D → ¬same_dir A B C D.

Lemma vector_same_dir_cong : ∀ A B C D, A ≠ B → C ≠ D → ∃ X , ∃ Y , same_dir A B X Y ∧ Cong X Y C D.

End Vectors.

Chapter 33

Library orthocenter

```
Require Import circumcenter.
Require Import quadrilaterals_inter_dec.
Section Orthocenter.
Context `{MT:Tarski_2D_euclidean}.
Context `{EqDec:EqDecidability Tpoint}.
Context `{InterDec:InterDecidability Tpoint Col}.
```

Orthocenter

```
Definition is_orthocenter H A B C :=
 ¬ Col A B C ∧
 ∃ A1 , ∃ B1 , Perp A A1 B C ∧ Perp B B1 A C ∧ Col H A A1 ∧ Col H B B1.
Lemma construct_intersection : ∀ A B C X1 X2 X3,
 ¬ Col A B C →
 Par A C B X1 → Par A B C X2 → Par B C A X3 →
 ∃ E , Col E A X3 ∧ Col E B X1.
Lemma not_col_par_col2_diff : ∀ A B C D E F,
  ¬ Col A B C → Par A B C D → Col C D E → Col A E F → A ≠ E.
Lemma construct_triangle : ∀ A B C,
  ¬ Col A B C → ∃ D , ∃ E , ∃ F ,
  Col B D F ∧ Col A E F ∧ Col C D E ∧
  Par A B C D ∧ Par A C B D ∧ Par B C A E ∧
  Par A B C E ∧ Par A C B F ∧ Par B C A F ∧
  D ≠ E ∧ D ≠ F ∧ E ≠ F.
Lemma diff_not_col_col_par4_mid: ∀ A B C D E,
  D ≠ E → ¬ Col A B C → Col C D E → Par A B C D →
  Par A B C E → Par A E B C → Par A C B D → is_midpoint C D E.
Lemma altitude_is_perp_bisect : ∀ A B C O A1 E F,
  A ≠ O → E ≠ F → Perp A A1 B C → Col O A1 A → Col A E F → Par B C A E →
is_midpoint A E F →
```

```
  perp_bisect A O E F.

Lemma altitude_intersect:
 ∀ A A1 B B1 C C1 O: Tpoint,
 ¬ Col A B C →
 Perp A A1 B C → Perp B B1 A C → Perp C C1 A B →
 Col O A A1 → Col O B B1 →
 Col O C C1.

End Orthocenter.
```

Chapter 34

Library gravityCenter

```
Require Import triangle_midpoints_theorems.
Require Export perp_bisect.

Section GravityCenter.

Context `{MT:Tarski_2D_euclidean}.
Context `{EqDec:EqDecidability Tpoint}.
Context `{InterDec:InterDecidability Tpoint Col}.
```

Center of gravity

```
Lemma intersection_two_medians_exist :
∀ A B C I J,
 ¬Col A B C →
 is_midpoint I B C → is_midpoint J A C →
 ∃ G, Col G A I ∧ Col G B J.

Lemma intersection_two_medians_exist_unique :
∀ A B C I J,
 ¬Col A B C →
 is_midpoint I B C → is_midpoint J A C →
 ∃! G, Col G A I ∧ Col G B J.

Definition is_gravity_center G A B C :=
 ¬ Col A B C ∧
 ∃ I, ∃ J, is_midpoint I B C ∧ is_midpoint J A C ∧ Col G A I ∧ Col G B J.

Lemma is_gravity_center_exist_unique : ∀ A B C,
  ¬ Col A B C →
  ∃! G, is_gravity_center G A B C.

Ltac intersection_medians G A B C I J H1 H2 H3 :=
 let T := fresh in assert(T:= intersection_two_medians_exist A B C I J H1 H2 H3);
 ex_and T G.

Tactic Notation "Name" ident(G) "the" "intersection" "of" "the" "medians" "(" ident(A)
ident(I) ")" "which" "is" "a" "median" "since" ident(H2) "and" "(" ident(B) ident(J) ")"
```

"which" "is" "a" "median" "since" *ident*(*H3*) "of" "the" "non-flat" "triangle" *ident*(*A*) *ident*(*B*) *ident*(*C*) *ident*(*H1*) :=
intersection_medians G A B C I J H1 H2 H3.

```
Lemma three_medians_intersect:
 ∀ A B C I J K,
 ¬Col A B C →
 is_midpoint I B C →
 is_midpoint J A C →
 is_midpoint K A B →
 ∃ G , Col G A I ∧ Col G B J ∧Col G C K.
```

End GravityCenter.

Chapter 35

Library construction_functions

Require Import Epsilon.
Require Import gravityCenter.

Section T.

Context `{MT:**Tarski_2D_euclidean**}.
Context `{$EqDec$:**EqDecidability** Tpoint}.
Context `{$InterDec$:**InterDecidability** Tpoint Col}.

Lemma symmetric_point_ex_unicity :
 ∀ I A, ∃! B , is_midpoint I A B.

Definition symmetric_point I A := select_the (symmetric_point_ex_unicity I A).

Definition gravity_center A B C (H:¬Col A B C) := select_the (is_gravity_center_exist_unique A B C H).

End T.

Chapter 36

Library Epsilon

```
Require Import ClassicalEpsilon.

Set Implicit Arguments.

Lemma ex_unique_introduction :
  ∀ (A:Type)(P:A→Prop),
   (∃ x , P x) →
   (∀ a b, P a → P b → a = b)->
   ∃! x , P x.

Ltac ex_unique_intro :=
    match goal with [ ⊢ ex (unique ?P) ] ⇒
       apply ex_unique_introduction end.

Theorem epsilon_ind :
  ∀ (A:Type)(i : inhabited A)(P:A→Prop)
         (Q:A→Prop), (∃ x , P x)->
                      (∀ x : A, P x → Q x) →
                      Q (epsilon i P).

Theorem epsilon_equiv : ∀ (A:Type)(i : inhabited A)(P:A→Prop),
  (∃ x , P x)<-> P (epsilon i P).

Definition tau (A:Type)(i:inhabited A)(P:A→Prop) :=
     epsilon i (fun y ⇒ ¬ (P y)).

Lemma forall_def : (∀ P, P ∨ ¬ P) →
                    ∀ (A:Type)(i:inhabited A)(P:A→Prop),
                    (∀ y, P y) ↔ P (tau i P).

Definition iota (A:Type)(i : inhabited A)(P: A→ Prop) : A :=
epsilon i (unique P).

Lemma iota_e : ∀ (A:Type) (i: inhabited A) (P : A → Prop),
```

```
                              ex (unique P) →
                              unique P (iota i P).

Lemma iota_e_weak : ∀ (A:Type) (i: inhabited A) (P : A → Prop),
                              ex (unique P) →
                               P (iota i P).

Theorem iota_ind :
   ∀ (A:Type)(i: inhabited A)(P:A→Prop)
         (Q:A→Prop),
                              (∀ b : A, unique P b → Q b) →
                               ex (unique P) →
                               Q (iota i P).

Theorem iota_ind_weak : ∀ (A:Type)(i: inhabited A)(P:A→Prop)
         (Q:A→Prop),
             (∀ b : A, P b → Q b) →
             ex (unique P) →
             Q (iota i P).

Ltac epsilon_elim_aux :=
  match goal with [⊢ (?P (epsilon (A:=?X) ?a ?Q))] ⇒
           apply epsilon_ind; auto
  end.

Ltac epsilon_elim := epsilon_elim_aux ||
  match goal with
  [⊢ (?P (?k ?d))] ⇒
   (let v := eval cbv zeta beta delta [k] in (k d) in
     (match v with (epsilon ?w ?d) ⇒ change (P v); epsilon_elim_aux end))
  | [⊢ (?P (?k ?arg ?arg1))] ⇒
   (let v := eval cbv zeta beta delta [k] in (k arg arg1) in
     (match v with (epsilon ?w ?d) ⇒ change (P v); epsilon_elim_aux end))
  | [⊢ (?P ?k)] ⇒
   (let v := eval cbv zeta beta delta [k] in k in
     (match v with (epsilon ?w ?d) ⇒ change (P v); epsilon_elim_aux end))
  end.

Ltac iota_elim_aux :=
  match goal with [⊢ (?P (iota (A:=?X) ?i ?Q))] ⇒
    apply iota_ind; auto
  end.

Ltac iota_elim := iota_elim_aux ||
  match goal with
  [⊢ (?P (?k ?arg))] ⇒
   (let v := eval cbv zeta beta delta [k] in (k arg) in
```

```
      (match v with (iota ?w ?d) ⇒ change (P v); iota_elim_aux end))
 | [ ⊢ (?P (?k ?arg ?arg1))] ⇒
    (let v := eval cbv zeta beta delta [k] in (k arg arg1) in
      (match v with (iota ?w ?d) ⇒ change (P v); iota_elim_aux end))
 | [ ⊢ (?P ?k)] ⇒
    (let v := eval cbv zeta beta delta [k] in k in
      (match v with (iota ?w ?d) ⇒ change (P v); iota_elim_aux end))
  end.
```

```
Lemma iota_parameter_rw : ∀ (A:Type)(a b:inhabited A)(P:A→ Prop),
  (ex (unique P)) →
    iota a P = iota b P.
```

Structures for descriptions and specifications

The operators "some" and "the" correspond to epsilon and iota applied to the proof they expect

```
Definition select(A:Type)(P:A→Prop)(pi:∃ x , P x)
  : A
  := proj1_sig (constructive_indefinite_description P pi).

Implicit Arguments select [A P].

Definition select_the (A:Type)(P:A→Prop)
                              (pi:∃! x , P x)
               : A
               := proj1_sig (constructive_indefinite_description (unique P) pi).

Implicit Arguments select_the [A P].

Lemma select_e : ∀ (A:Type)(P:A→Prop)(pi:∃ x , P x),
                  P (select pi).

Implicit Arguments select_e [A P].

Theorem select_ind :∀ (A:Type)(P Q:A→Prop)
                  (pi:∃ x , P x),
                  (∀ a, P a → Q a) → Q (select pi).

Theorem select_the_e : ∀ (A:Type)(P:A→Prop)
                                  (pi:∃! x , P x),
  unique P (select_the pi).

Implicit Arguments select_the_e [A P].

Theorem select_the_ind : ∀ (A:Type)(P Q:A→Prop)
                                  (pi:ex (unique P)),
  (∀ b, unique P b → Q b) → Q (select_the pi).

Implicit Arguments select_the_ind [A P].

Theorem select_the_rw : ∀ (A:Type)(P:A→Prop)
```

```
                (pi:∃! x, P x) b,
          P b → b= select_the pi.

Implicit Arguments select_the_rw [A P].

Ltac select_the_elim_aux :=
  match goal with [ ⊢ (?Q (select_the ?d))] ⇒
    apply select_the_ind
  end.

Ltac select_the_elim := select_the_elim_aux ||
  match goal with [ ⊢ (?P ?k)] ⇒
   (let v := eval cbv beta zeta delta [k] in k in
     (match v with (select_the ?d) ⇒ change (P v); select_the_elim_aux end))
  end.

Ltac select_elim_aux :=
  match goal with [ ⊢ (?P (select ?d))] ⇒
    generalize (select d) (select_e d); simpl
  end.

Ltac select_elim := select_elim_aux ||
  match goal with [ ⊢ (?P ?k)] ⇒
   (let v := eval cbv beta zeta delta [k] in k in
     (match v with (select ?d) ⇒ change (P v); select_elim_aux end))
  end.

Ltac hilbert_e :=
 select_the_elim || select_elim || iota_elim || epsilon_elim .

Tactic Notation "epsilon_e" constr(T) "at" integer(n) :=
       pattern T at n; hilbert_e.

Tactic Notation "epsilon_e" constr(T) :=
       pattern T ; hilbert_e.

Ltac select_i P :=
   refine (select (P:=P) _).

Ltac select_the_i P :=
   refine (select_the (P:=P) _).
```

Building partial functions

```
Definition functional(A B : Type) (R: A → B → Prop ) :=
  ∀ x y , R x y → ∀ z, R x z → y = z.

Implicit Arguments functional [A B].

Section AB_fixed.
 Variables A B: Type.
```

description of some partial function by a Domain and a binary relation

```
Definition choice_fun (i : inhabited B)(DA : A→Prop)(R:A→B→Prop)(a:A):=
    (epsilon i (fun (b':B) ⇒ DA a ∧ R a b')).

Definition iota_fun (b : inhabited B)(DA : A→Prop)(R:A→B→Prop)(a:A):=
    (iota b (fun (b':B) ⇒ DA a ∧ R a b')).

Lemma choice_fun_e : ∀ i : inhabited B,
  ∀ (DA : A→Prop)(R:A→B→Prop)(a:A),
   (∃ d : B, R a d) →
   DA a → R a (choice_fun i DA R a).

Lemma choice_fun_ind : ∀ i : inhabited B,
  ∀ (P : A→Prop)(Q R:A→B→Prop)(a x:A),
   (∃ d : B, R a d) → a = x →
   P a → (∀ b, P a → R a b → Q a b) →
   Q x (choice_fun i P R a).
```

a partial function equipped with a proof of its partial correctness

```
Definition the_fun (i : inhabited B)(D : A→Prop)(R:A→B→Prop)
     (pi : ∀ a, D a → ∃! b, R a b): A → B :=
    fun a:A ⇒ iota i (fun (b:B) ⇒ D a ∧ R a b).
```

same stuff but without unicity of the result

```
Definition some_fun (i : inhabited B)(D : A→Prop)(R:A→B→Prop)
     (pi : ∀ a, D a → ∃ b: B, R a b) : A → B :=
    fun a:A ⇒ epsilon i (fun (b:B) ⇒ D a ∧ R a b).

Lemma iota_fun_e : ∀ b : inhabited B,
  ∀ (DA : A→Prop)(R:A→B→Prop)(a:A),
   (ex (unique (R a))) →
   DA a → unique (R a) (iota_fun b DA R a).

Lemma iota_fun_ind : ∀ b : inhabited B,
  ∀ (P : A→Prop)(Q R:A→B→Prop)(a x:A),
   a = x → (ex (unique (R a))) →
   P a → (∀ b, P a → unique (R a) b → Q a b) →
   Q x (iota_fun b P R a).

Lemma iota_fun_rw : ∀ b: inhabited B,
  ∀ (P : A→Prop)(R:A→B→Prop)(a:A)(b':B),
   (ex (unique ( R a))) →
   P a →
   R a b' → b'= (iota_fun b P R a).

Theorem the_fun_e : ∀ (b :inhabited B)(P: A→ Prop)(R:A→B→Prop)
(pi : ∀ a, P a → ex (unique (R a))) (a:A),
    P a → R a (the_fun b P R pi a) .

Theorem some_fun_e : ∀ (i : inhabited B)(P: A→ Prop)(R:A→B→Prop)
```

```
(pi : ∀ a, P a → ∃ b0: B , R a b0) (a:A),
        P a → R a (some_fun i P R pi a) .
Theorem the_fun_rw : ∀ (b : inhabited B)(P: A→ Prop)(R:A→B→Prop)
(pi : ∀ a, P a → ex (unique ( R a))) (a:A)(b0:B),
        P a → R a b0 → b0= (the_fun b P R pi a) .
Theorem the_fun_ind : ∀ (b : inhabited B)(P: A→ Prop)(Q R:A→B→Prop)
(pi : ∀ a, P a → ex (unique (R a))) (a x:A),
        P a → a = x →
(∀ b0, P a → unique (R a) b0 → Q a b0) →
  Q x (the_fun b P R pi a) .
Theorem some_fun_ind : ∀ (i : inhabited B)(P: A→ Prop)(Q R:A→B→Prop)
(pi : ∀ a, P a → ∃ b: B , R a b) (a x:A),
        a = x → P a → (∀ b0, P a → R a b0 → Q a b0) →
        Q x (some_fun i P R pi a) .
End AB_fixed.
```

tactic for building a partial function

```
Opaque the_fun some_fun.
Ltac the_fun_i P R:=
   refine (the_fun _ P R _).
Ltac some_fun_i P R:=
   refine (some_fun _ P R _).
Ltac the_fun_elim :=
match goal with [ ⊢ (?Q ?a (?f ?t)) ] ⇒
 (let v := eval cbv beta zeta delta [f] in f in
        (match v with (the_fun ?i ?P ?R ?d) ⇒ change (Q a (v t));
                  apply the_fun_ind ; auto end)) end.
Ltac some_fun_elim :=
match goal with [ ⊢ (?Q ?a (?f ?t)) ] ⇒
 (let v := eval cbv beta zeta delta [f] in f in
        (match v with (some_fun ?i ?P ?R ?d) ⇒ change (Q a (v t));
                  apply some_fun_ind ; auto end)) end.
Ltac iota_fun_elim :=
match goal with [ ⊢ (?Q ?a (?f ?t)) ] ⇒
 (let v := eval cbv beta zeta delta [f] in f in
        (match v with (iota_fun ?i ?P ?R ) ⇒ change (Q a (v t));
                  apply iota_fun_ind ; auto end)) end.
Ltac choice_fun_elim :=
match goal with [ ⊢ (?Q ?a (?f ?t)) ] ⇒
 (let v := eval cbv beta zeta delta [f] in f in
```

```
      (match v with (choice_fun ?i ?P ?R ) ⇒ change (Q a (v t));
                 apply choice_fun_ind ; auto end)) end.

Ltac pfun_e := choice_fun_elim || iota_fun_elim || some_fun_elim ||
               the_fun_elim.

Ltac partial_fun_e arg result := pattern arg, result;pfun_e.

Ltac iota_fun_rewrite :=
 match goal with [ ⊢ (?f ?x = ?y) ] ⇒
    (let v := eval cbv beta zeta delta [f] in f in
          (match v with (iota_fun ?i ?P ?R ) ⇒
                     change (v x = y);symmetry; apply iota_fun_rw; auto
                      end)) end.

Ltac the_fun_rewrite :=
 match goal with [ ⊢ (?f ?x = ?y) ] ⇒
    (let v := eval cbv beta zeta delta [f] in f in
          (match v with (the_fun ?i ?P ?R ?d ) ⇒
                     change (v x = y);symmetry; apply the_fun_rw; auto
                      end)) end.

Section definite_description.

Lemma dd'' : ∀ (A:Type)(B:A→Type)
                          (R:∀ x:A, B x → Prop),
      (∀ x: A, inhabited (B x)) →
      (sigT (fun f : ∀ x:A, B x ⇒
                     (∀ x:A, (∃ y:B x, R x y) → R x (f x)))).

Lemma dd' : ∀ (A:Type) (B:A → Type)
                              (R:∀ x:A, B x → Prop),
      (∀ x, inhabited (B x)) →
      (∀ x:A, ∃ y : B x , R x y ) →
       sigT (fun f : ∀ x:A, B x ⇒ (∀ x:A, R x (f x))).

End definite_description.

Definition epsilon_extensionality :=
  ∀ (A:Type)(i: inhabited A)(P Q:A→Prop),
  (∀ a, P a ↔ Q a) →
  epsilon i P = epsilon i Q.

Definition restricted_epsilon_extensionality :=
  ∀ (A:Type)(i: inhabited A)(P Q:A→Prop),
  (∀ a, P a ↔ Q a) →
  ex P →
  epsilon i P = epsilon i Q.
```

Lemma inhabited_bool : **inhabited bool**.

Lemma or_to_sumbool : ∀ P Q : Prop, $P \lor Q \to$ {P}+{Q}.

Lemma iota_rw : ∀ (A:Type)(a: **inhabited** A)(P:A→Prop)(x:A),
unique P x →
iota a P = x.

Opaque epsilon.

Definition unspec (A:Type)(i:**inhabited** A) := epsilon i (fun x ⇒ **True**).

Lemma quasi_classic : ∀ (A:Type), A + (A→**False**).

Chapter 37

Library triangle_midpoints_theorems

```
Require Export quadrilaterals_inter_dec.

Ltac assert_all := treat_equalities; assert_cols; assert_diffs; assert_congs.

Section TriangleMidpointsTheorems.

Context `{MT:Tarski_2D_euclidean}.
Context `{EqDec:EqDecidability Tpoint}.
Context `{InterDec:InterDecidability Tpoint Col}.

Lemma triangle_mid_par_strict_cong_aux : ∀ A B C P Q R,
 ¬Col A B C →
 is_midpoint P B C →
 is_midpoint Q A C →
 is_midpoint R A B →
 Par_strict A B Q P ∧ Cong A R P Q ∧ Cong B R P Q .

Lemma triangle_mid_par_strict_cong_1 : ∀ A B C P Q R,
 ¬Col A B C →
 is_midpoint P B C →
 is_midpoint Q A C →
 is_midpoint R A B →
 Par_strict A B Q P ∧ Cong A R P Q.

Lemma triangle_mid_par_strict_cong_2 : ∀ A B C P Q R,
 ¬Col A B C →
 is_midpoint P B C →
 is_midpoint Q A C →
 is_midpoint R A B →
 Par_strict A B Q P ∧ Cong B R P Q.

Lemma triangle_mid_par_strict_cong_simp :
 ∀ A B C P Q,
 ¬ Col A B C →
 is_midpoint P B C →
```

is_midpoint Q A C $\rightarrow$
Par_strict A B Q P.

Lemma triangle_mid_par_strict_cong : $\forall$ A B C P Q R,
$\neg$Col A B C $\rightarrow$
is_midpoint P B C $\rightarrow$
is_midpoint Q A C $\rightarrow$
is_midpoint R A B $\rightarrow$
Par_strict A B Q P $\wedge$ Par_strict A C P R $\wedge$ Par_strict B C Q R $\wedge$
Cong A R P Q $\wedge$ Cong B R P Q $\wedge$ Cong A Q P R $\wedge$ Cong C Q P R $\wedge$ Cong B P Q R $\wedge$
Cong C P Q R.

Lemma triangle_mid_par_strict : $\forall$ A B C P Q,
$\neg$Col A B C $\rightarrow$
is_midpoint P B C $\rightarrow$
is_midpoint Q A C $\rightarrow$
Par_strict A B Q P.

Lemma triangle_mid_par_flat_cong_aux : $\forall$ A B C P Q R,
$B \neq A \rightarrow$
Col A B C $\rightarrow$
is_midpoint P B C $\rightarrow$
is_midpoint Q A C $\rightarrow$
is_midpoint R A B $\rightarrow$
Par A B Q P $\wedge$ Cong A R P Q $\wedge$ Cong B R P Q.

Lemma triangle_mid_par_flat_cong_1 : $\forall$ A B C P Q R,
$B \neq A \rightarrow$
Col A B C $\rightarrow$
is_midpoint P B C $\rightarrow$
is_midpoint Q A C $\rightarrow$
is_midpoint R A B $\rightarrow$
Par A B Q P $\wedge$ Cong A R P Q.

Lemma triangle_mid_par_flat_cong_2 : $\forall$ A B C P Q R,
$B \neq A \rightarrow$
Col A B C $\rightarrow$
is_midpoint P B C $\rightarrow$
is_midpoint Q A C $\rightarrow$
is_midpoint R A B $\rightarrow$
Par A B Q P $\wedge$ Cong B R P Q.

Lemma triangle_mid_par_flat_cong : $\forall$ A B C P Q R,
$B \neq A \rightarrow$
$C \neq A \rightarrow$
$C \neq B \rightarrow$
Col A B C $\rightarrow$

is_midpoint P B C $\rightarrow$
is_midpoint Q A C $\rightarrow$
is_midpoint R A B $\rightarrow$
Par A B Q P $\wedge$ Par A C P R $\wedge$ Par B C Q R $\wedge$
Cong A R P Q $\wedge$ Cong B R P Q $\wedge$ Cong A Q P R $\wedge$ Cong C Q P R $\wedge$ Cong B P Q R $\wedge$ Cong C P Q R.

Lemma triangle_mid_par_flat : $\forall$ A B C P Q,
$B \neq A \rightarrow$
Col A B C $\rightarrow$
is_midpoint P B C $\rightarrow$
is_midpoint Q A C $\rightarrow$
Par A B Q P.

Lemma triangle_mid_par : $\forall$ A B C P Q,
$A \neq B \rightarrow$
is_midpoint P B C $\rightarrow$
is_midpoint Q A C $\rightarrow$
Par A B Q P.

Lemma triangle_mid_par_cong : $\forall$ A B C P Q R,
$B \neq A \rightarrow$
$C \neq A \rightarrow$
$C \neq B \rightarrow$
is_midpoint P B C $\rightarrow$
is_midpoint Q A C $\rightarrow$
is_midpoint R A B $\rightarrow$
Par A B Q P $\wedge$ Par A C P R $\wedge$ Par B C Q R $\wedge$
Cong A R P Q $\wedge$ Cong B R P Q $\wedge$ Cong A Q P R $\wedge$ Cong C Q P R $\wedge$ Cong B P Q R $\wedge$ Cong C P Q R.

Lemma triangle_par_mid : $\forall$ A B C P Q,
$\neg$Col A B C $\rightarrow$
is_midpoint P B C $\rightarrow$
Par A B Q P $\rightarrow$
Col Q A C $\rightarrow$
is_midpoint Q A C.

End TriangleMidpointsTheorems.

Chapter 38

Library tarski_to_hilbert

```
Require Import Ch12_parallel_inter_dec.
Require Import Morphisms.
Require Import hilbert_axioms.

Section T.

Context `{MT:Tarski_2D_euclidean}.
Context `{EqDec:EqDecidability Tpoint}.
Context `{InterDec:InterDecidability Tpoint Col}.
```

We need a notion of line.

```
Definition Line := @Couple Tpoint.
Definition Lin := build_couple Tpoint.

Definition Incident (A : Tpoint) (l : Line) := Col A (P1 l) (P2 l).
```

38.1 Group I Combination

For every pair of distinct points there is a line containing them.

```
Lemma axiom_line_existence : ∀ A B, A≠B → ∃ l, Incident A l ∧ Incident B l.
```

We need a notion of equality over lines.

```
Definition Eq : relation Line := fun l m ⇒ ∀ X, Incident X l ↔ Incident X m.

Infix "=l=" := Eq (at level 70):type_scope.

Lemma incident_eq : ∀ A B l, ∀ H : A≠B,
 Incident A l → Incident B l →
 (Lin A B H) =l= l.
```

Our equality is an equivalence relation.

```
Lemma eq_transitivity : ∀ l m n, l =l= m → m =l= n → l =l= n.

Lemma eq_reflexivity : ∀ l, l =l= l.
```

Lemma eq_symmetry : ∀ l m, l =l= m → m =l= l.

Instance Eq_Equiv : Equivalence Eq.

The equality is compatible with Incident

Lemma eq_incident : ∀ A l m, l =l= m →
(Incident A l ↔ Incident A m).

Instance incident_Proper (A:Tpoint) :
Proper (Eq ==>iff) (Incident A).
Qed.

There is only one line going through two points.

Lemma axiom_line_unicity : ∀ A B l m, $A \neq B$ →
(Incident A l) → (Incident B l) → (Incident A m) → (Incident B m) →
l =l= m.

Every line contains at least two points.

Lemma axiom_two_points_on_line : ∀ l,
∃ A , ∃ B , Incident B l ∧ Incident A l ∧ $A \neq B$.

Definition of the collinearity predicate. We say that three points are collinear if they belongs to the same line.

Definition Col_H := fun A B C ⇒
∃ l , Incident A l ∧ Incident B l ∧ Incident C l.

We show that the notion of collinearity we just defined is equivalent to the notion of collinearity of Tarski.

Lemma cols_coincide_1 : ∀ A B C, Col_H A B C → Col A B C.

Lemma cols_coincide_2 : ∀ A B C, Col A B C → Col_H A B C.

There exists three non collinear points.

Lemma axiom_plan :
∃ A , ∃ B , ∃ C , ¬ Col_H A B C.

38.2 Group II Order

Definition of the Between predicate of Hilbert. Note that it is different from the Between of Tarski. The Between of Hilbert is strict.

Definition Between_H A B C :=
Bet A B C ∧ $A \neq B$ ∧ $B \neq C$ ∧ $A \neq C$.

Lemma axiom_between_col :
∀ A B C, Between_H A B C → Col_H A B C.

If B is between A and C, it is also between C and A.

Lemma axiom_between_comm : ∀ A B C, Between_H A B C → Between_H C B A.

Lemma axiom_between_out :
∀ A B, A ≠ B → ∃ C , Between_H A B C.

Lemma axiom_between_only_one :
∀ A B C,
Between_H A B C → ¬ Between_H B C A ∧ ¬ Between_H B A C.

Lemma between_one : ∀ A B C,
A≠B → A≠C → B≠C → Col A B C →
Between_H A B C ∨ Between_H B C A ∨ Between_H B A C.

Lemma axiom_between_one : ∀ A B C,
A≠B → A≠C → B≠C → Col_H A B C →
Between_H A B C ∨ Between_H B C A ∨ Between_H B A C.

Axiom of Pasch, (Hilbert version).
First we define a predicate which means that the line l intersects the segment AB.

Definition cut := fun l A B ⇒ ¬Incident A l ∧ ¬Incident B l ∧ ∃ I , Incident I l ∧ Between_H A I B.

We show that this definition is equivalent to the predicate two_sides of Tarski.

Lemma cut_two_sides : ∀ l A B, cut l A B ↔ two_sides (P1 l) (P2 l) A B.

Lemma axiom_pasch : ∀ A B C l,
¬ Col_H A B C → ¬ Incident C l →
cut l A B → cut l A C ∨ cut l B C.

Lemma Incid_line :
∀ P A B l, A≠B →
Incident A l → Incident B l → Col P A B → Incident P l.

38.3 Group III Parallels

We use a definition of parallel which is valid only in 2D:

Definition Para l m := ¬ ∃ X , Incident X l ∧ Incident X m.

Lemma axiom_euclid_existence :
∀ l P, ¬ Incident P l → ∃ m , Para l m.

Lemma Para_Par : ∀ A B C D, ∀ HAB: A≠B, ∀ HCD: C≠D,
Para (Lin A B HAB) (Lin C D HCD) → Par A B C D.

Lemma axiom_euclid_unicity :
∀ l P m1 m2,
¬ Incident P l →
Para l m1 → Incident P m1 →
Para l m2 → Incident P m2 →
m1 =l= m2.

38.4 Goup IV Congruence

The cong predicate of Hilbert is the same as the one of Tarski:

Definition Hcong:=Cong.

Lemma axiom_hcong_1_existence :
 ∀ A B l M,
 $A \neq B$ → Incident M l →
 ∃ A', ∃ B',
   Incident A' l ∧ Incident B' l ∧
   Between_H A' M B' ∧ Hcong M A' A B ∧ Hcong M B' A B.

Lemma axiom_hcong_1_unicity :
 ∀ A B l M A' B' A'' B'', $A \neq B$ → Incident M l →
  Incident A' l → Incident B' l →
  Incident A'' l → Incident B'' l →
  Between_H A' M B' → Hcong M A' A B →
  Hcong M B' A B → Between_H A'' M B'' →
  Hcong M A'' A B → Hcong M B'' A B →
  (A' = A'' ∧ B' = B'') ∨ (A' = B'' ∧ B' = A'').

As a remark we also prove another version of this axiom as formalized in Isabelle by Phil Scott.

Definition same_side_scott E A B := $E \neq A$ ∧ $E \neq B$ ∧ Col_H E A B ∧ ¬ Between_H A E B.

Remark axiom_hcong_scott:
 ∀ P Q A C, $A \neq C$ → $P \neq Q$ →
  ∃ B, same_side_scott A B C ∧ Hcong P Q A B.

Transivity of congruence.

Lemma axiom_hcong_trans : ∀ A B C D E F, Hcong A B C D → Hcong A B E F → Hcong C D E F.

Reflexivity of congruence.

Lemma axiom_hcong_refl : ∀ A B , Hcong A B A B.

We define when two segments do not intersect.

Definition disjoint := fun A B C D ⇒ ¬ ∃ P, Between_H A P B ∧ Between_H C P D.

Note that two disjoint segments may share one of their extremities.

Lemma col_disjoint_bet : ∀ A B C, Col_H A B C → disjoint A B B C → Bet A B C.

Lemma axiom_hcong_3 : ∀ A B C A' B' C',
   Col_H A B C → Col_H A' B' C' →
  disjoint A B B C → disjoint A' B' B' C' →
  Hcong A B A' B' → Hcong B C B' C' → Hcong A C A' C'.

We define the notion of half ray.

Definition HLine := @**Couple** Tpoint.
Definition hlin := build_couple Tpoint.

We define incidence with an half ray

Definition IncidentH (*A* : Tpoint) (*l* : HLine) :=
A = (P1 *l*) ∨ *A* = (P2 *l*) ∨ Between_H (P1 *l*) *A* (P2 *l*) ∨ Between_H (P1 *l*) (P2 *l*) *A*.

Definition of half ray equality.

Definition hEq : relation HLine := fun *h1* *h2* ⇒ (P1 *h1*) = (P1 *h2*) ∧
((P2 *h1*) = (P2 *h2*) ∨ Between_H (P1 *h1*) (P2 *h2*) (P2 *h1*) ∨
Between_H (P1 *h1*) (P2 *h1*) (P2 *h2*)).

Infix "=h=" := hEq (at level 70):*type_scope*.

This is an equivalence relation.

Lemma hEq_refl : ∀ *h*, *h* =h= *h*.

Lemma hEq_sym : ∀ *h1* *h2*, *h1* =h= *h2* → *h2* =h= *h1*.

Lemma hEq_trans : ∀ *h1* *h2* *h3*, *h1* =h= *h2* → *h2* =h= *h3* → *h1* =h= *h3*.

Instance hEq_Equiv : **Equivalence** hEq.

We define the concept of angle.

Definition Angle : Type := @**Triple** Tpoint.
Definition angle := build_triple Tpoint.

The congruence of angles of Hilbert is the same as the congruence of angles of Tarski.

Definition Hconga : relation Angle := fun *A1* *A2* ⇒ Conga (V1 *A1*) (V *A1*) (V2 *A1*) (V1 *A2*) (V *A2*) (V2 *A2*).

This is an equivalence relation.

Lemma hconga_refl : ∀ *a*, Hconga *a* *a*.

Lemma hconga_sym : ∀ *a* *b*, Hconga *a* *b* → Hconga *b* *a*.

Lemma hconga_trans : ∀ *a* *b* *c*, Hconga *a* *b* → Hconga *b* *c* → Hconga *a* *c*.

Instance hconga_Equiv : **Equivalence** Hconga.
Qed.

Lemma exists_not_incident : ∀ *A* *B* : Tpoint, ∀ *HH* : *A* ≠ *B* , ∃ *C* , ¬ Incident *C* (Lin *A* *B* *HH*).

Definition line_of_hline := fun *hl* ⇒ Lin (P1 *hl*) (P2 *hl*) (Cond *hl*).

Definition same_side := fun *A* *B* *l* ⇒ ∃ *P* , cut *l* *A* *P* ∧ cut *l* *B* *P*.

Same side predicate corresponds to one_side of Tarski.

Lemma same_side_one_side : ∀ A B l, same_side A B l → one_side (P1 l) (P2 l) A B.

Lemma one_side_same_side : ∀ A B l, one_side (P1 l) (P2 l) A B → same_side A B l.

Definition outH := fun P A B ⇒ Between_H P A B ∨ Between_H P B A ∨ (P ≠ A ∧ A = B).

This is equivalent to the out predicate of Tarski.

Lemma outH_out : ∀ P A B, outH P A B → out P A B.

Lemma out_outH : ∀ P A B, out P A B → outH P A B.

Definition of a point inside an angle.

Definition InAngleH a P :=
 (∃ M , Between_H (V1 a) M (V2 a) ∧ ((outH (V a) M P) ∨ M = (V a))) ∨
 outH (V a) (V1 a) P ∨ outH (V a) (V2 a) P.

This is (almost) equivalent to the same notion in Tarski's.

Lemma in_angle_equiv : ∀ a P, (P ≠ (V a) ∧ InAngleH a P) ↔ InAngle P (V1 a) (V a) (V2 a).

Lemma in_angleH_in_angle : ∀ a P, (P ≠ (V a) ∧ InAngleH a P) → InAngle P (V1 a) (V a) (V2 a).

The 2D version of the fourth congruence axiom

Lemma outH_in_angleH_colH : ∀ a P, outH (V a) (V1 a) (V2 a) → InAngleH a P → Col_H (V a) (V1 a) P.

Lemma incident_col : ∀ M l, Incident M l → Col M (P1 l)(P2 l).

Lemma col_incident : ∀ M l, Col M (P1 l)(P2 l) → Incident M l.

Lemma Bet_Between_H : ∀ A B C,
 Bet A B C → A≠B → B≠C → Between_H A B C.

Lemma aux : ∀ (h h1 : HLine),
 P1 h = P1 h1 →
 P2 h1 ≠ P1 h.

Lemma axiom_hcong_4_existence :
 ∀ a h P,
 ¬Incident P (line_of_hline h) → ¬Between_H (V1 a)(V a)(V2 a) →
 ∃ h1 , (P1 h) = (P1 h1) ∧
 (∀ CondAux : P1 h = P1 h1,
 Hconga a (angle (P2 h) (P1 h) (P2 h1) (conj (sym_not_equal (Cond h)) (aux h h1 CondAux)))
 ∧ (∀ M, ¬Incident M (line_of_hline h) ∧ InAngleH (angle (P2 h) (P1 h) (P2 h1) (conj (sym_not_equal (Cond h)) (aux h h1 CondAux))) M
 → same_side P M (line_of_hline h))).

general case

```
Definition hline_construction a h P hc H :=
 (P1 h) = (P1 hc) ∧
 Hconga a (angle (P2 h) (P1 h) (P2 hc) (conj (sym_not_equal (Cond h)) H)) ∧
  (∀ M, InAngleH (angle (P2 h) (P1 h) (P2 hc) (conj (sym_not_equal (Cond h)) H)) M →
   same_side P M (line_of_hline h)).

Lemma same_side_trans :
 ∀ A B C l,
  same_side A B l → same_side B C l → same_side A C l.

Lemma same_side_sym :
 ∀ A B l,
  same_side A B l → same_side B A l.

Lemma in_angleH_trivial :
 ∀ A B C H,
  InAngleH (angle A B C H) A ∧ InAngleH (angle A B C H) C.

Lemma axiom_hcong_4_unicity :
  ∀ a h P h1 h2 HH1 HH2,
  ¬Incident P (line_of_hline h) → ¬Between_H (V1 a)(V a)(V2 a) →
  hline_construction a h P h1 HH1 → hline_construction a h P h2 HH2 →
  h1 =h= h2.

Lemma axiom_cong_5':
 ∀ (A B C A' B' C' : Tpoint) (H1 : B ≠ A ∧ C ≠ A)
            (H2 : B' ≠ A' ∧ C' ≠ A') ,
 ∀ H3 : (A≠B ∧ C≠B), ∀ H4: A' ≠ B' ∧ C' ≠ B',
  Hcong A B A' B' →
  Hcong A C A' C' →
  Hconga {| V1 := B ; V := A ; V2:= C ; Pred := H1 |}
         {| V1 := B'; V := A'; V2:= C'; Pred := H2 |} →
  Hconga {| V1 := A ; V := B ; V2:= C ; Pred := H3 |}
         {| V1 := A'; V := B'; V2:= C'; Pred := H4 |}.

End T.

Section Hilbert_to_Tarski.

Context `{T:Tarski_2D_euclidean}.
Context `{EqDec:EqDecidability Tpoint}.
Context `{InterDec:InterDecidability Tpoint Col}.

Instance Hilbert_follow_from_Tarski : Hilbert.

End Hilbert_to_Tarski.
```

Chapter 39

Library tarski_to_makarios

```
Require Export tarski_axioms.
```

In this file we formalize the result given in T. J. M. Makarios: A further simplification of Tarski's axioms of geometry, June 2013.

```
Section Tarski83_to_Makarios_variant.

Context `{M:Tarski_neutral_dimensionless}.

Lemma cong_reflexivity : ∀ A B,
 Cong A B A B.

Lemma cong_symmetry : ∀ A B C D : Tpoint,
 Cong A B C D → Cong C D A B.

Lemma cong_left_commutativity : ∀ A B C D,
 Cong A B C D → Cong B A C D.

Lemma five_segments' : ∀ A A' B B' C C' D D' : Tpoint,
    Cong A B A' B' →
    Cong B C B' C' →
    Cong A D A' D' →
    Cong B D B' D' →
    Bet A B C → Bet A' B' C' → A ≠ B → Cong D C C' D'.

Instance Makarios_Variant_follows_from_Tarski : Tarski_neutral_dimensionless_variant.

End Tarski83_to_Makarios_variant.

Section Makarios_variant_to_Tarski83.

Context `{M:Tarski_neutral_dimensionless_variant}.
Context `{EqDec:EqDecidability MTpoint}.

Ltac prolong A B x C D :=
 assert (sg:= Msegment_construction A B C D);
 ex_and sg x.

Lemma Mcong_reflexivity : ∀ A B,
```

CongM A B A B.

Lemma Mcong_symmetry : $\forall$ A B C D ,
CongM A B C D $\rightarrow$ CongM C D A B.

Lemma between_trivial : $\forall$ A B : MTpoint, BetM A B B.

Lemma between_symmetry : $\forall$ A B C : MTpoint, BetM A B C $\rightarrow$ BetM C B A.

Lemma cong_pseudo_reflexivity : $\forall$ A B : MTpoint, CongM A B B A.

Lemma Mcong_left_commutativity : $\forall$ A B C D,
CongM A B C D $\rightarrow$ CongM B A C D.

Lemma five_segments : $\forall$ A A' B B' C C' D D' : MTpoint,
CongM A B A' B' $\rightarrow$
CongM B C B' C' $\rightarrow$
CongM A D A' D' $\rightarrow$
CongM B D B' D' $\rightarrow$
BetM A B C $\rightarrow$ BetM A' B' C' $\rightarrow$ $A \neq B$ $\rightarrow$ CongM C D C' D'.

Instance Tarski_follows_from_Makarios_Variant : **Tarski_neutral_dimensionless**.

End Makarios_variant_to_Tarski83.

Chapter 40

Library tarski_to_beeson

Require Export Ch08_orthogonality.

In this file we formalize the result given in T. J. M. Makarios: A further simplification of Tarski's axioms of geometry, June 2013.

Section Tarski_to_intuitionistic_Tarski.

Context `{M:**Tarski_neutral_dimensionless**}.
Context `{$EqDec$:**EqDecidability** Tpoint}.

Lemma cong_stability : ∀ A B C D, ¬ ¬ Cong A B C D → Cong A B C D.

Definition BetH A B C : Prop := Bet A B C ∧ $A \neq B$ ∧ $B \neq C$.

Lemma bet_stability : ∀ A B C, ¬ ¬ BetH A B C → BetH A B C.

Definition T A B C : Prop := ¬ (A≠B ∧ B≠C ∧ ¬ BetH A B C).

Definition ColB A B C := A≠B ∧ ¬ (~ T C A B ∧ ¬ T A C B ∧ ¬ T A B C).

Lemma between_identity_B : ∀ A B, ¬ BetH A B A.

Lemma Bet_T : ∀ A B C, Bet A B C → T A B C.

Lemma BetH_Bet : ∀ A B C, BetH A B C → Bet A B C.

Lemma T_Bet : ∀ A B C, T A B C → Bet A B C.

Lemma NT_NBet : ∀ A B C, ¬ T A B C → ¬ Bet A B C.

Lemma T_dec : ∀ A B C, T A B C ∨ ¬ T A B C.

Lemma between_inner_transitivity_B : ∀ A B C D : Tpoint, BetH A B D → BetH B C D → BetH A B C.

Lemma ColB_Col : ∀ A B C, ColB A B C → Col A B C.

Lemma Diff_Col_ColB : ∀ A B C, $A \neq B$ ∧ Col A B C → ColB A B C.

Lemma NColB_NDiffCol : ∀ A B C, ¬ ColB A B C → ¬ ($A \neq B$ ∧ Col A B C).

Lemma NColB_NColOrEq : ∀ A B C, ¬ ColB A B C → ¬ Col A B C ∨ A = B.

Lemma inner_pasch_B : ∀ A B C P Q,

```
  BetH A P C → BetH B Q C → ¬ ColB A B C →
  ∃ x, BetH P x B ∧ BetH Q x A.
Lemma between_symmetry_B : ∀ A B C, BetH A B C → BetH C B A.
Lemma five_segments_B : ∀ A A' B B' C C' D D' : Tpoint,
     Cong A B A' B' →
     Cong B C B' C' →
     Cong A D A' D' →
     Cong B D B' D' →
     ¬ (A ≠ B ∧ B ≠ C ∧ ¬ BetH A B C) →
     ¬ (A' ≠ B' ∧ B' ≠ C' ∧ ¬ BetH A' B' C') →
     A ≠ B → Cong C D C' D'.
Lemma segment_construction_B : ∀ A B C D, A≠B → ∃ E, T A B E ∧ Cong B E C D.
Lemma lower_dim_B : ∃ A, ∃ B, ∃ C, ¬ T C A B ∧ ¬ T A C B ∧ ¬ T A B C.
Instance Beeson_follows_from_Tarski : intuitionistic_Tarski_neutral_dimensionless.
End Tarski_to_intuitionistic_Tarski.
Section Proof_of_eq_stability_in_IT.
Context `{MIT:intuitionistic_Tarski_neutral_dimensionless}.
Lemma cong_stability_eq_stability : ∀ A B : ITpoint, ¬ A ≠ B → A = B.
End Proof_of_eq_stability_in_IT.
Require Import Classical.
Section Intuitionistic_Tarski_to_Tarski.
Context `{MIT:intuitionistic_Tarski_neutral_dimensionless}.
Lemma Col_dec : ∀ A B C, ICol A B C ∨ ¬ ICol A B C.
Lemma eq_dec : ∀ A B :ITpoint, A=B ∨ A≠B.
Definition BetT A B C := IBet A B C ∨ A=B ∨ B=C.
Lemma bet_id : ∀ A B : ITpoint, BetT A B A → A = B.
Lemma IT_chara : ∀ A B C,
 IT A B C ↔ A=B ∨ B=C ∨ IBet A B C.
Lemma BetT_symmetry : ∀ A B C, BetT A B C → BetT C B A.
Lemma BetT_id : ∀ A B, BetT A B A → A=B.

Lemma pasch_col_case : ∀ A B C P Q : ITpoint,
        BetT A P C →
        BetT B Q C → ICol A B C → ∃ x : ITpoint, BetT P x B ∧ BetT Q x A.
Lemma pasch : ∀ A B C P Q : ITpoint,
        BetT A P C →
        BetT B Q C → ∃ x : ITpoint, BetT P x B ∧ BetT Q x A.
```

Lemma five_segments:
 ∀ A A' B B' C C' D D' : ITpoint,
  ICong A B A' B' →
  ICong B C B' C' →
  ICong A D A' D' →
  ICong B D B' D' →
  BetT A B C → BetT A' B' C' → $A \neq B$ → ICong C D C' D'.

Lemma IT_trivial : ∀ A B, IT A A B.

Lemma another_point : ∀ A, ∃ B:ITpoint, $A \neq B$.

Lemma segment_construction :
 ∀ A B C D : ITpoint,
  ∃ E : ITpoint, BetT A B E ∧ ICong B E C D.

Lemma lower_dim :
 ∃ A B C : ITpoint, ¬ (BetT A B C ∨ BetT B C A ∨ BetT C A B).

Instance IT_to_T : **Tarski_neutral_dimensionless**.
Qed.

End Intuitionistic_Tarski_to_Tarski.

Chapter 41

Library unit_tests

Require Import quadrilaterals_inter_dec.

Section UnitTests.

Context `{MT:**Tarski_2D_euclidean**}.
Context `{$EqDec$:**EqDecidability** Tpoint}.
Context `{$InterDec$:**InterDecidability** Tpoint Col}.

Goal ∀ A B I, $A \neq B$ → is_midpoint I A B → $I \neq A \wedge I \neq B$.

Goal ∀ A B I, $B \neq A$ → is_midpoint I A B → $I \neq A \wedge I \neq B$.

Goal ∀ A B I, $I{\neq}A$ → is_midpoint I A B → $I \neq B \wedge A \neq B$.

Goal ∀ A B I, $I{\neq}B$ → is_midpoint I A B → $I \neq A \wedge A \neq B$.

Goal ∀ A B I, $A{\neq}I$ → is_midpoint I A B → $I \neq B \wedge A \neq B$.

Goal ∀ A B I, $B{\neq}I$ → is_midpoint I A B → $I \neq A \wedge A \neq B$.

Goal ∀ A B I, $A{\neq}B$ → is_midpoint I A B → $A \neq I \wedge I \neq B$.

Goal ∀ A B:Tpoint, $A{\neq}B$ → $B{\neq}A$ → **True**.

Goal ∀ A B C Q,
 $B \neq A$ → Col A B C → is_midpoint Q A C →
 $A \neq C$ → $B \neq C$ → is_midpoint A B C →
 $Q \neq C$.

Goal ∀ A B C D, Perp A B C D → $A{\neq}B \wedge C{\neq}D$.

Goal ∀ A B C D, $A{\neq}B$ → Perp A B C D → $A{\neq}B \wedge C{\neq}D$.

Goal ∀ A B C D, $A{\neq}B$ → Perp B A C D → $A{\neq}B \wedge C{\neq}D$.

Goal ∀ A B C D, $A{\neq}B$ → Perp B A D C → $A{\neq}B \wedge C{\neq}D$.

Goal ∀ A B C D, $A{\neq}B$ → $C{\neq}D$ → Perp B A D C → $A{\neq}B \wedge C{\neq}D$.

Goal ∀ A B C D, $D{\neq}C$ → Perp B A D C → $A{\neq}B \wedge C{\neq}D$.

Goal ∀ X A B C D, Perp_in X A B C D → $A{\neq}B \wedge C{\neq}D$.

Goal $\forall$ A B C D, Par A B C D $\rightarrow$ $A{\neq}B$ $\wedge$ $C{\neq}D$.

Goal $\forall$ A B C D, Par_strict A B C D $\rightarrow$ $A{\neq}B$ $\wedge$ $C{\neq}D$.

Goal $\forall$ A B C, out A B C $\rightarrow$ $B{\neq}A$ $\wedge$ $C{\neq}A$.

Goal $\forall$ A B C, out A B C $\rightarrow$ Col B A C.

Goal $\forall$ A B C D, $\neg$ Col A B C $\rightarrow$ $\neg$ Col A B D $\rightarrow$
$A{\neq}$ D.

Goal $\forall$ A B C D I,
is_midpoint I A B $\rightarrow$ Par A B C D $\rightarrow$ $I{\neq}A$.

Goal $\forall$ A B C D I,
is_midpoint I A B $\rightarrow$ Par A I C D $\rightarrow$ $B{\neq}A$.

Goal $\forall$ A B C D,
Cong A B C D $\rightarrow$ $C{\neq}D$ $\rightarrow$ $A{\neq}B$.

Goal $\forall$ A B C D,
Cong A B C D $\rightarrow$ $D{\neq}C$ $\rightarrow$ $A{\neq}B$.

Goal $\forall$ A B C D,
Cong A B C D $\rightarrow$ $A{\neq}B$ $\rightarrow$ $C{\neq}D$.

Goal $\forall$ A B C D,
Cong A B C D $\rightarrow$ $B{\neq}A$ $\rightarrow$ $C{\neq}D$.

Goal $\forall$ A B C D E,
$\neg$ Col A B C $\rightarrow$
$\neg$ Col B D E $\rightarrow$ $A{\neq}B$ $\rightarrow$
Col A B D $\rightarrow$ Col A B E $\rightarrow$ Col A B C.

Goal $\forall$ A B C D E,
$\neg$ Col A B C $\rightarrow$
$\neg$ Col B D E $\rightarrow$ $A{\neq}B$ $\rightarrow$
Col A B D $\rightarrow$ Col A B E $\rightarrow$ $C{\neq}D$.

Goal $\forall$ A B C D,
Par_strict A B C D $\rightarrow$
$\neg$ Col A B C.

Goal $\forall$ A B C, ($A{\neq}B$ $\rightarrow$ $B{\neq}C$ $\rightarrow$ $A{\neq}C$ $\rightarrow$ Col A B C) $\rightarrow$ Col A B C.

Goal $\forall$ A B C D I, $I \neq A$ $\rightarrow$ $I \neq B$ $\rightarrow$ $I \neq C$ $\rightarrow$ $I \neq D$ $\rightarrow$ Col I A B $\rightarrow$ Col I C D $\rightarrow$ $\neg$ Col A B C $\rightarrow$ $\neg$ Col A B D.

Goal $\forall$ A B C D I, $I \neq A$ $\rightarrow$ $I \neq B$ $\rightarrow$ $I \neq C$ $\rightarrow$ $I \neq D$ $\rightarrow$ Col I A B $\rightarrow$ Col I C D $\rightarrow$ $\neg$ Col A B C $\rightarrow$ $A \neq D$.

Goal $\forall$ A B C D, Parallelogram A B C D $\rightarrow$ Cong A B C D.

Goal $\forall$ A B C, is_midpoint A B C $\rightarrow$ Cong A B C A.

End UnitTests.

Chapter 42

Library arity

```
Require Import Arith.
Require Import List.
Require Import Sorting.
Require Import Coq.Program.Equality.
Lemma minus_n_0 : ∀ n, n-0 = n.
Lemma plus_0_n : ∀ n, 0+n = n.
Lemma plus_n_0 : ∀ n, n+0 = n.
Lemma plus_n_1 : ∀ n, n+1 = S n.
Lemma minus_n1_n2_0 : ∀ n1 n2, n1+n2-0 = n1+n2.
Fixpoint arity (T:Type) (n:nat) :=
 match n with
 | 0 ⇒ Prop
 | S p ⇒ T → arity T p
 end.
Fixpoint cartesianPowerAux (T:Type) (n:nat) :=
 match n with
 | 0 ⇒ T
 | S p ⇒ (T × cartesianPowerAux T p)%type
 end.
Definition cartesianPower T n := cartesianPowerAux T (n-1).
Definition headCP {T:Type} {n:nat} (cp : cartesianPower T (S n)) : T.
Definition tailCP {T:Type} {n:nat} (cp : cartesianPower T (S (S n))) : (cartesianPower T (S n)).
Definition tailDefaultCP {T:Type} {n:nat} (cp : cartesianPower T (S n)) (Default : cartesianPower T n) : (cartesianPower T n).
Definition allButLastCP {T:Type} {n:nat} (cp : cartesianPower T (S (S n))) : (cartesianPower T (S n)).
```

```
Lemma allButLastCPTl {T:Type} {n:nat} :
  ∀ (cp : cartesianPower T (S (S (S n)))),
  allButLastCP (tailCP cp) = tailCP (allButLastCP cp).

Definition lastCP {T:Type} {n:nat} (cp : cartesianPower T (S n)) : T.

Lemma lastCPTl {T:Type} {n:nat} : ∀ (cp : cartesianPower T (S (S n))), lastCP cp = lastCP
(tailCP cp).

Lemma CP_ind {T:Type} {n : nat} : ∀ (cp cp' : cartesianPower T (S (S n))),
  headCP cp = headCP cp' → tailCP cp = tailCP cp' → cp = cp'.

Definition CPPair {T : Type} :
  ∀ (cp : cartesianPower T 2),
  cp = (fst cp, snd cp).

Definition tailCPbis {T:Type} {n:nat} m1 m2 (cp : cartesianPower T m1) :
  (S (S n)) = m1 → (S n) = m2 → (cartesianPower T m2).

Definition consHeadCP {T:Type} {n:nat} (t : T) (cp : cartesianPower T n) : (cartesian-
Power T (S n)).

Lemma consHeadCPHd {T:Type} {n:nat} :
  ∀ (cp : cartesianPower T n) t,
  headCP (consHeadCP t cp) = t.

Lemma consHeadCPTl {T:Type} {n:nat} :
  ∀ (cp : cartesianPower T (S n)) t,
  tailCP (consHeadCP t cp) = cp.

Lemma consHeadCPOK {T:Type} {n:nat} :
  ∀ (cp : cartesianPower T (S (S n))),
  cp = consHeadCP (headCP cp) (tailCP cp).

Definition consTailCP {T:Type} {n:nat} (cp : cartesianPower T n) (t : T) : (cartesianPower
T (S n)).

Lemma consTailCPTl {T:Type} {n:nat} :
  ∀ (cp : cartesianPower T (S (S n))) t,
  tailCP (consTailCP cp t) = consTailCP (tailCP cp) t.

Lemma consTailCPOK {T:Type} {n:nat} :
  ∀ (cp : cartesianPower T (S (S n))),
  cp = consTailCP (allButLastCP cp) (lastCP cp).

Lemma consTailCPAbl {T:Type} {n:nat} :
  ∀ (cp : cartesianPower T (S n)) t,
  allButLastCP (consTailCP cp t) = cp.

Lemma consTailCPTlD {T:Type} {n:nat} :
  ∀ (cp : cartesianPower T n) t,
  tailDefaultCP (consHeadCP t cp) cp = cp.
```

Lemma consHdTlTlHd {T:Type} {n:nat} :
  ∀ (F L : T) (X : cartesianPower T n),
  consHeadCP F (consTailCP X L) = consTailCP (consHeadCP F X) L.

Lemma consTlHdHdTl {T:Type} {n:nat} :
  ∀ (A B C : T) (X : cartesianPower T n),
  consHeadCP A (consHeadCP B (consTailCP X C)) = consTailCP (consHeadCP A (consHeadCP B X)) C.

Definition CPToList {T:Type} {n:nat} (cp : cartesianPower T n) : list T.

Definition InCP {T:Type} {n:nat} p (cp : cartesianPower T n) := In p (CPToList cp).

Lemma InCPOK {T:Type} {n:nat} : ∀ p (cp : cartesianPower T (S (S n))),
  InCP p cp ↔ ((p = headCP cp) ∨ InCP p (tailCP cp)).

Lemma lastCPIn {T:Type} {n:nat} : ∀ (cp : cartesianPower T (S n)), InCP (lastCP cp) cp.

Definition nthCP {T:Type} {m:nat} (n : nat) (cp : cartesianPower T m) ($Default$: T) := nth (n-1) (CPToList cp) $Default$.

Lemma CPToListOK {T:Type} {n:nat} : ∀ (cp : cartesianPower T (S (S n))), CPToList cp = cons (headCP cp) (CPToList (tailCP cp)).

Lemma CPLHdTlOK {T:Type} {n:nat} : ∀ (cp : cartesianPower T (S (S n))),
  CPToList cp = ((headCP cp) :: nil) ++ CPToList (tailCP cp).

Lemma consTailOK {T:Type} {n:nat} : ∀ (cp : cartesianPower T (S n)) t,
  CPToList (consTailCP cp t) = CPToList cp ++ t :: nil.

Lemma InNth {T:Type} {n:nat} :
  ∀ (cp : cartesianPower T n) (t $Default$: T),
  InCP t cp → (∃ id, $id \geq 1$ ∧ $id \leq n$ ∧ t = nthCP id cp $Default$).

Lemma nthFirst {T:Type} {n:nat} :
  ∀ (cp : cartesianPower T (S n)) (t $Default$: T),
  t = nthCP 1 cp $Default$ → t = headCP cp.

Lemma lengthOfCPToList {T:Type} {n:nat} : ∀ (cp : cartesianPower T n), n = length (CPToList cp).

Lemma lastTailOK {T:Type} {n:nat} :
  ∀ (cp : cartesianPower T (S (S n))),
  lastCP cp = lastCP (tailCP cp).

Lemma consTailCPLast {T:Type} {n:nat} :
  ∀ (cp : cartesianPower T (S n)) t,
  lastCP (consTailCP cp t) = t.

Lemma nthLast {T:Type} {n:nat} :
  ∀ (cp : cartesianPower T (S n)) ($Default$: T),
  lastCP cp = nthCP (S n) cp $Default$.

Lemma nthCircPerm1 {T:Type} {n:nat} :

```
  ∀ (cp : cartesianPower T (S (S n))) (t Default : T),
  t = nthCP 1 cp Default → t = nthCP (S (S n)) (consTailCP (tailCP cp) (headCP cp))
Default.
Lemma nthCircPerm1Eq {T:Type} {n:nat} :
  ∀ (cp : cartesianPower T (S (S n))) (Default : T),
  nthCP 1 cp Default = nthCP (S (S n)) (consTailCP (tailCP cp) (headCP cp)) Default.
Lemma nthCircPerm2 {T:Type} {n:nat} :
  ∀ (cp : cartesianPower T (S (S n))) (t Default : T) id,
  t = nthCP (S (S id)) cp Default → id ≤ n →
  t = nthCP (S id) (consTailCP (tailCP cp) (headCP cp)) Default.
Lemma nthCircPerm2Eq {T:Type} {n:nat} :
  ∀ (cp : cartesianPower T (S (S n))) (Default : T) id,
  id ≤ n → nthCP (S (S id)) cp Default = nthCP (S id) (consTailCP (tailCP cp) (headCP
cp)) Default.
Lemma nthCPTlOK {T:Type} {m:nat} :
  ∀ (cp : cartesianPower T (S (S m))) (Default : T) n,
  nthCP (S n) (tailCP cp) Default = nthCP (S (S n)) cp Default.
Lemma nthEqOK {T:Type} {m:nat} :
  ∀ (cp1 cp2 : cartesianPower T (S m)) (Default : T),
  (∀ n, nthCP n cp1 Default = nthCP n cp2 Default) → cp1 = cp2.
Lemma consTailPerm {T:Type} {n:nat} :
  ∀ (cp : cartesianPower T (S (S n))),
  Permutation.Permutation (CPToList cp) (CPToList (consTailCP (tailCP cp) (headCP
cp))).
Definition ListToCP {T : Type} (l : list T) (Default : T) : cartesianPower T (length l).
Fixpoint circPermNCP {T:Type} {m:nat} (n : nat) (cp : cartesianPower T (S (S m))) :=
  match n with
  | 0 ⇒ cp
  | S p ⇒ circPermNCP p (consTailCP (tailCP cp) (headCP cp))
  end.
Lemma circPermNCP0 {T:Type} {n:nat} :
  ∀ (cp : cartesianPower T (S (S n))),
  cp = circPermNCP 0 cp.
Lemma circPermNCPOK {T:Type} {m:nat} :
  ∀ (n : nat) (cp : cartesianPower T (S (S m))),
  circPermNCP (S n) cp = circPermNCP n (consTailCP (tailCP cp) (headCP cp)).
Lemma nthCircPermNAny {T:Type} {m:nat} :
  ∀ (cp : cartesianPower T (S (S m))) (Default : T) id n,
  id + n ≤ S m → nthCP (S id + n) cp Default = nthCP (S id) (circPermNCP n cp) Default.
```

Lemma circPermNIdFirst {T:Type} {n:**nat**} :
∀ (cp : cartesianPower T (S (S n))) ($Default$: T),
nthCP 1 cp $Default$ = nthCP 1 (circPermNCP (S (S n)) cp) $Default$.

Lemma circPermNConsTlOK {T:Type} {m:**nat**} :
∀ (n : **nat**) (cp : cartesianPower T (S (S m))),
consTailCP (tailCP (circPermNCP n cp)) (headCP (circPermNCP n cp)) = circPermNCP n (consTailCP (tailCP cp) (headCP cp)).

Lemma circPermPerm {T:Type} {m:**nat**} :
∀ (n : **nat**) (cp : cartesianPower T (S (S m))),
circPermNCP (S (S (S n))) cp = circPermNCP 1 (circPermNCP (S (S n)) cp).

Lemma nthCP01 {T:Type} {n:**nat**} :
∀ (cp : cartesianPower T (S n)) $Default$,
nthCP 0 cp $Default$ = nthCP 1 cp $Default$.

Lemma circPermNIdAux {T:Type} {n:**nat**} :
∀ (cp : cartesianPower T (S (S n))) ($Default$: T),
cp = circPermNCP (S (S n)) cp.

Lemma circPermNId {T:Type} {n:**nat**} :
∀ (cp : cartesianPower T (S (S n))),
cp = circPermNCP (S (S n)) cp.

Lemma circPermNConsOK {T:Type} {n:**nat**} :
∀ (cp : cartesianPower T (S n)) ($t1$ $t2$: T),
circPermNCP (S n) (consTailCP (consTailCP cp $t1$) $t2$) = consHeadCP $t1$ (consHeadCP $t2$ cp).

Lemma listInd {T : Type} : ∀ n (l l' : **list** T) $Default$,
length l = (S n) →
length l' = (S n) →
hd $Default$ l = hd $Default$ l' →
tl l = tl l' →
l = l'.

Lemma CPLHd {T : Type} :
∀ (a : T) l $Default$,
hd $Default$ (CPToList (ListToCP (a :: l) $Default$)) = a.

Lemma ListToCPTl {T : Type} :
∀ (a $a0$: T) l ($Ha0l$: (S (length l)) = length ($a0$ :: l)) $Haa0l$ $Default$,
tailCPbis (length (a :: $a0$ :: l)) (length (a :: l)) (ListToCP (a :: $a0$ :: l) $Default$) $Haa0l$ $Ha0l$ =
ListToCP ($a0$ :: l) $Default$.

Lemma CPToListTl1 {T : Type} :
∀ (a $a0$: T) l (cp : cartesianPower T (length (a :: $a0$ :: l))), tl (CPToList cp) = CPToList (tailCP cp).

Lemma CPToListTl2 $\{T : \texttt{Type}\}$ $\{n : \textbf{nat}\}$:
  $\forall$ (cp : cartesianPower T (S (S n))), tl (CPToList cp) = CPToList (tailCP cp).

Lemma CPCPL $\{T : \texttt{Type}\}$:
  $\forall$ (a : T) l ($cp1$: cartesianPower T (length (a :: l)))
  ($cp2$: cartesianPower T (S(length l))),
  $cp1$ = $cp2$ $\rightarrow$ CPToList $cp1$ = CPToList $cp2$.

Lemma CPLCP $\{T : \texttt{Type}\}$ $\{n : \textbf{nat}\}$:
  $\forall$ ($cp1$ $cp2$: cartesianPower T (S n)),
  CPToList $cp1$ = CPToList $cp2$ $\rightarrow$ $cp1$ = $cp2$.

Lemma CPLRec $\{T : \texttt{Type}\}$:
  $\forall$ (a : T) l $Default$,
  (a :: (CPToList (ListToCP l $Default$))) = CPToList (ListToCP (a :: l) $Default$).

Lemma CPLOK $\{T : \texttt{Type}\}$: $\forall$ (l : **list** T) $Default$,
  CPToList (ListToCP l $Default$) = l.

Definition fixLastCP $\{T$:Type$\}$ $\{n$:**nat**$\}$ ($appPred$: cartesianPower T (S (S n)) $\rightarrow$ Prop) (t : T) : cartesianPower T (S n) $\rightarrow$ Prop.

Lemma fixLastCPOK $\{T$:Type$\}$ $\{n$:**nat**$\}$:
  $\forall$ ($appPred$: cartesianPower T (S (S n)) $\rightarrow$ Prop) (cp : cartesianPower T (S n)) (t : T),
  $appPred$ (consTailCP cp t) = (fixLastCP $appPred$ t) cp.

Definition app $\{T$:Type$\}$ $\{n$:**nat**$\}$ ($pred$: arity T n) (cp : cartesianPower T n) : Prop.

Definition app_n_1 $\{T$:Type$\}$ $\{n$:**nat**$\}$ ($pred$: arity T (S n)) (cp : cartesianPower T n) (x : T) : Prop.

Lemma app_n_1_app $\{T$:Type$\}$ $\{n$:**nat**$\}$:
  $\forall$ ($pred$: arity T (S (S n))) (x : T) (cpp : cartesianPower T (S n)) (cpt : cartesianPower T (S (S n))),
  app_n_1 $pred$ cpp x $\rightarrow$ allButLastCP cpt = cpp $\rightarrow$ lastCP cpt = x $\rightarrow$
  app $pred$ cpt.

Lemma app_app_n_1 $\{T$:Type$\}$ $\{n$:**nat**$\}$:
  $\forall$ ($pred$: arity T (S (S n))) (x : T) (cpp : cartesianPower T (S n)) (cpt : cartesianPower T (S (S n))),
  app $pred$ cpt $\rightarrow$ allButLastCP cpt = cpp $\rightarrow$ lastCP cpt = x $\rightarrow$
  app_n_1 $pred$ cpp x.

Lemma app_n_1_app_eq $\{T$:Type$\}$ $\{n$:**nat**$\}$:
  $\forall$ ($pred$: arity T (S (S n))) (x : T) (cpp : cartesianPower T (S n)) (cpt : cartesianPower T (S (S n))),
  allButLastCP cpt = cpp $\rightarrow$ lastCP cpt = x $\rightarrow$
  (app $pred$ cpt $\leftrightarrow$ app_n_1 $pred$ cpp x).

Definition app_1_n $\{T$:Type$\}$ $\{n$:**nat**$\}$ ($pred$: arity T (S n)) (x : T) (cp : cartesianPower T n) : Prop.

Lemma app_1_n_app {T:Type} {n:nat} :
∀ (pred : arity T (S (S n))) (x : T) (cpp : cartesianPower T (S n)) (cpt : cartesianPower T (S (S n))),
app_1_n pred x cpp → headCP cpt = x → tailCP cpt = cpp →
app pred cpt.

Lemma app_app_1_n {T:Type} {n:nat} :
∀ (pred : arity T (S (S n))) (x : T) (cpp : cartesianPower T (S n)) (cpt : cartesianPower T (S (S n))),
app pred cpt → headCP cpt = x → tailCP cpt = cpp →
app_1_n pred x cpp.

Lemma app_1_n_app_eq {T:Type} {n:nat} :
∀ (pred : arity T (S (S n))) (x : T) (cpp : cartesianPower T (S n)) (cpt : cartesianPower T (S (S n))),
headCP cpt = x → tailCP cpt = cpp →
(app pred cpt ↔ app_1_n pred x cpp).

Definition app_2_n {T:Type} {n:nat} (pred : arity T (S (S n))) (x1 x2 : T) (cp : cartesianPower T n) : Prop.

Lemma app_2_n_app {T:Type} {n:nat} :
∀ (pred : arity T (S (S (S n)))) (x1 x2 : T)
(cpp : cartesianPower T (S n)) (cpt : cartesianPower T (S (S (S n)))),
app_2_n pred x1 x2 cpp → headCP cpt = x1 → headCP (tailCP cpt) = x2 → tailCP (tailCP cpt) = cpp →
app pred cpt.

Lemma app_2_n_app_default {T:Type} {n:nat} :
∀ (pred : arity T (S (S n))) (x1 x2 : T)
(cpp Default : cartesianPower T n) (cpt : cartesianPower T (S (S n))),
app_2_n pred x1 x2 cpp → headCP cpt = x1 →
headCP (tailCP cpt) = x2 →
tailDefaultCP (tailCP cpt) Default = cpp →
app pred cpt.

Lemma app_app_2_n {T:Type} {n:nat} :
∀ (pred : arity T (S (S (S n)))) (x1 x2 : T)
(cpp : cartesianPower T (S n)) (cpt : cartesianPower T (S (S (S n)))),
app pred cpt → headCP cpt = x1 → headCP (tailCP cpt) = x2 → tailCP (tailCP cpt) = cpp →
app_2_n pred x1 x2 cpp.

Lemma app_app_2_n_default {T:Type} {n:nat} :
∀ (pred : arity T (S (S n))) (x1 x2 : T)
(cpp Default : cartesianPower T n) (cpt : cartesianPower T (S (S n))),
app pred cpt → headCP cpt = x1 → headCP (tailCP cpt) = x2 → tailDefaultCP (tailCP cpt) Default = cpp →

app_2_n *pred x1 x2 cpp*.

Lemma app_2_n_app_eq {*T*:Type} {*n*:**nat**} :
∀ (*pred* : arity *T* (S (S (S *n*)))) (*x1 x2* : *T*)
(*cpp* : cartesianPower *T* (S *n*)) (*cpt* : cartesianPower *T* (S (S (S *n*)))),
headCP *cpt* = *x1* → headCP (tailCP *cpt*) = *x2* → tailCP (tailCP *cpt*) = *cpp* →
(app *pred cpt* ↔ app_2_n *pred x1 x2 cpp*).

Lemma PermOKAux {*T* : Type} {*m* : **nat**} :
∀ (*appPred* : (cartesianPower *T* (S (S *m*))) → Prop) *n*,
(∀ (*A* : *T*) (*X* : cartesianPower *T* (S *m*)), *appPred* (consHeadCP *A X*) → *appPred* (consTailCP *X A*)) →
(∀ (*X* : cartesianPower *T* (S (S *m*))),
appPred X → *appPred* (circPermNCP *n X*)).

Lemma PermOK {*T* : Type} {*n* : **nat**} :
∀ (*cp1 cp2* : cartesianPower *T* (S (S *n*))) (*appPred* : (cartesianPower *T* (S (S *n*))) → Prop),
(∀ (*A* : *T*) (*X* : cartesianPower *T* (S *n*)),
appPred (consHeadCP *A X*) → *appPred* (consTailCP *X A*)) →
(∀ (*A B* : *T*) (*X* : cartesianPower *T n*),
appPred (consHeadCP *A* (consHeadCP *B X*)) → *appPred* (consHeadCP *B* (consHeadCP *A X*))) →
appPred cp1 →
Permutation.Permutation (CPToList *cp1*) (CPToList *cp2*) →
appPred cp2.

Chapter 43

Library tactics_axioms

```
Require Export arity.
```

Minimal set of lemmas needed to use the ColR tactic.

```
Class Col_theory (COLTpoint : Type) (CTCol: COLTpoint → COLTpoint → COLTpoint → Prop) :=
{
  CTcol_trivial : ∀ A B : COLTpoint, CTCol A A B;
  CTcol_permutation_1 : ∀ A B C : COLTpoint, CTCol A B C → CTCol B C A;
  CTcol_permutation_2 : ∀ A B C : COLTpoint, CTCol A B C → CTCol A C B;
  CTcol3 : ∀ X Y A B C : COLTpoint,
                X ≠ Y → CTCol X Y A → CTCol X Y B → CTCol X Y C → CTCol A B C
}.
Class Arity :=
{
  COATpoint : Type;
  n : nat
}.
Class Coapp_predicates (Ar : Arity) :=
{
  wd : arity COATpoint (S (S n));
  coapp : arity COATpoint (S (S (S n)))
}.
```

Minimal set of lemmas needed to use the Coapp tactic.

```
Class Coapp_theory (Ar : Arity) (COP : Coapp_predicates Ar) :=
{
  wd_perm_1 : ∀ A : COATpoint,
              ∀ X : cartesianPower COATpoint (S n),
                app_1_n wd A X → app_n_1 wd X A;
  wd_perm_2 : ∀ A B : COATpoint,
              ∀ X : cartesianPower COATpoint n,
```

```
                    app_2_n wd A B X → app_2_n wd B A X;
  coapp_perm_1 : ∀ A : COATpoint,
                     ∀ X : cartesianPower COATpoint (S (S n)),
                       app_1_n coapp A X → app_n_1 coapp X A;
  coapp_perm_2 : ∀ A B : COATpoint,
                     ∀ X : cartesianPower COATpoint (S n),
                       app_2_n coapp A B X → app_2_n coapp B A X;
  link_betwenn_wd_and_coapp : ∀ A : COATpoint,
                                  ∀ X : cartesianPower COATpoint (S (S n)),
                                    app_1_n coapp A X ∨ app wd X;
  pseudo_trans_of_coapp : ∀ A B C : COATpoint,
                              ∀ X : cartesianPower COATpoint (S n),
                                app_1_n wd A X →
                                app_2_n coapp B A X →
                                app_2_n coapp C A X →
                                app_2_n coapp B C X
}.
```

Chapter 44

Library Permutations

```
Require Import arity.
Require Import tactics_axioms.

Section Permutations.

Context `{COT : Coapp_theory}.

Lemma PermWdOK :
  ∀ (cp1 cp2 : cartesianPower COATpoint (S (S n))),
  app wd cp1 →
  Permutation.Permutation (CPToList cp1) (CPToList cp2) →
  app wd cp2.

Lemma PermCoappOK :
  ∀ (cp1 cp2 : cartesianPower COATpoint (S (S (S n)))),
  app coapp cp1 →
  Permutation.Permutation (CPToList cp1) (CPToList cp2) →
  app coapp cp2.

End Permutations.
```

Chapter 45

Library sets

```
Require Export MSets.
Require Import Arith.
Require Import NArith.
Require Import Notations.
Require Import Sorting.
Require Import Coq.Program.Equality.
Require Export tactics_axioms.
Module S := MSetList.Make PositiveOrderedTypeBits.
Module SWP := WPropertiesOn PositiveOrderedTypeBits S.
Module SetOfSetsOfPositiveOrderedType <: OrderedType.
  Definition t := S.t.
  Definition eq := S.Equal.
  Include IsEq.
  Definition eqb := S.equal.
  Definition eqb_eq := S.equal_spec.
  Include HasEqBool2Dec.
  Definition lt := S.lt.
  Instance lt_compat : Proper (eq==>eq==>iff) lt.
  Instance lt_strorder : StrictOrder lt.
  Definition compare := S.compare.
  Definition compare_spec := S.compare_spec.
End SetOfSetsOfPositiveOrderedType.
Module SS := MSetList.Make SetOfSetsOfPositiveOrderedType.
Definition fstpp (pair : (positive × positive)) :=
  match pair with
```

```
    |(a , b) ⇒ Pos.min a b
  end.

Definition sndpp (pair : (positive × positive)) :=
  match pair with
    |(a , b) ⇒ Pos.max a b
  end.

Module SetOfPairsOfPositiveOrderedType <: OrderedType.

  Definition t:= (positive × positive).

  Definition eq (t1 t2 : t) :=
    Pos.eq (fstpp(t1)) (fstpp(t2)) ∧ Pos.eq (sndpp(t1)) (sndpp(t2)).

  Include IsEq.

  Definition eqb (t1 t2 : t) :=
    Pos.eqb (fstpp(t1)) (fstpp(t2)) && Pos.eqb (sndpp(t1)) (sndpp(t2)).

  Lemma eqb_eq : ∀ t1 t2, eqb t1 t2 = true ↔ eq t1 t2.

  Include HasEqBool2Dec.

  Definition lt (t1 t2 : t) :=
    let ft1 := fstpp(t1) in
    let ft2 := fstpp(t2) in
    let st1 := sndpp(t1) in
    let st2 := sndpp(t2) in
    if Pos.eqb ft1 ft2 then Pos.lt st1 st2
                       else Pos.lt ft1 ft2.

  Lemma lt_irrefl : Irreflexive lt.

  Lemma lt_antiref : ∀ x, ¬ lt x x.

  Lemma lt_trans : Transitive lt.

  Instance lt_compat : Proper (eq==>eq==>iff) lt.

  Instance lt_strorder : StrictOrder lt.

  Definition compare t1 t2 :=
    let ft1 := fstpp(t1) in
    let ft2 := fstpp(t2) in
    let st1 := sndpp(t1) in
    let st2 := sndpp(t2) in
    match (Pos.compare ft1 ft2) with
      | Lt ⇒ Lt
      | Eq ⇒ Pos.compare st1 st2
      | Gt ⇒ Gt
    end.

  Lemma compare_spec : ∀ t1 t2, CompSpec eq lt t1 t2 (compare t1 t2).
```

```
End SetOfPairsOfPositiveOrderedType.
Module SP := MSetList.Make SetOfPairsOfPositiveOrderedType.
Module PosOrder <: TotalLeBool.
  Definition t := positive.
  Definition leb := Pos.leb.
  Lemma leb_total : ∀ p1 p2,
    leb p1 p2 = true ∨ leb p2 p1 = true.
  Lemma leb_dec : ∀ p1 p2,
    leb p1 p2 = true ∨ leb p1 p2 = false.
End PosOrder.
Module Import PosSort := Sort PosOrder.
Definition OCPAux {n : nat} (cp : cartesianPower positive (S (S n))) := (PosSort.sort
(CPToList cp)).
Lemma OCPALengthOK {n : nat} : ∀ (cp : cartesianPower positive (S (S n))), (length
(OCPAux cp)) = (S (S n)).
Lemma OCPSortedTl :
  ∀ (l : list positive),
  StronglySorted (fun x x0 : positive ⇒ is_true (x <=? x0)%positive) l →
  StronglySorted (fun x x0 : positive ⇒ is_true (x <=? x0)%positive) (tl l).
Lemma PermSorted : ∀ (l l' : list positive),
  Permutation.Permutation l l' →
  StronglySorted (fun x x0 : positive ⇒ is_true (x <=? x0)%positive) l →
  StronglySorted (fun x x0 : positive ⇒ is_true (x <=? x0)%positive) l' →
  l = l'.
Definition OCP {n : nat} (cp : cartesianPower positive (S (S n))) : cartesianPower positive
(S (S n)).
Lemma OCPSortedAux {n : nat} :
  ∀ (cp : cartesianPower positive (S (S n))),
  StronglySorted (fun x x0 : positive ⇒ is_true (x <=? x0)%positive) (CPToList (OCP
cp)).
Lemma OCPPerm {n : nat} :
  ∀ (cp : cartesianPower positive (S (S n))),
  Permutation.Permutation (CPToList cp) (CPToList (OCP cp)).
Lemma CPLOCPTlOK {n : nat} :
  ∀ (cp : cartesianPower positive (S (S (S n)))),
  headCP cp = headCP (OCP cp) →
  CPToList (OCP (tailCP cp)) = CPToList (tailCP (OCP cp)).
Lemma OCPTlOK {n : nat} :
```

```
  ∀ (cp : cartesianPower positive (S (S (S n)))),
  headCP cp = headCP (OCP cp) →
  OCP (tailCP cp) = tailCP (OCP cp).
Lemma InCPOCP {n : nat} : ∀ p (cp : cartesianPower positive (S (S n))),
  InCP p cp ↔ InCP p (OCP cp).
Lemma OCPOK {n : nat} :
  ∀ (cp : cartesianPower positive (S (S n))),
  StronglySorted (fun x x0 : positive ⇒ is_true (x <=? x0)%positive) (CPToList cp) →
  cp = OCP cp.
Lemma OCPIdempotent {n : nat} :
  ∀ (cp : cartesianPower positive (S (S n))),
  OCP cp = OCP (OCP cp).
Context {Ar : Arity}.
Module SetOfTuplesOfPositiveOrderedType <: OrderedType.
  Definition t := cartesianPower positive (S (S n)).
  Fixpoint eqList (l1 l2 : list positive) :=
    match l1, l2 with
    | nil, nil ⇒ True
    | (hd1 :: tl1), (hd2 :: tl2) ⇒ (Pos.eq hd1 hd2) ∧ (eqList tl1 tl2)
    | _, _ ⇒ False
    end.
  Lemma eqListRefl : ∀ l, eqList l l.
  Lemma eqListSym : ∀ l l', eqList l l' → eqList l' l.
  Lemma eqListTrans : ∀ l1 l2 l3, eqList l1 l2 → eqList l2 l3 → eqList l1 l3.
  Definition eq (cp1 cp2 : t) :=
    eqList (PosSort.sort (CPToList cp1)) (PosSort.sort (CPToList cp2)).
  Lemma eqListSortOCP : ∀ (cp : t), eqList (CPToList (OCP cp)) (PosSort.sort (CPToList cp)).
  Fixpoint eqbList (l1 l2 : list positive) :=
    match l1, l2 with
    | nil, nil ⇒ true
    | (hd1 :: tl1), (hd2 :: tl2) ⇒ (Pos.eqb hd1 hd2) && (eqbList tl1 tl2)
    | _, _ ⇒ false
    end.
  Definition eqb (cp1 cp2 : t) := eqbList (PosSort.sort (CPToList cp1)) (PosSort.sort (CP-
ToList cp2)).
  Include IsEq.
  Lemma eqbListEqList : ∀ l1 l2, eqbList l1 l2 = true ↔ eqList l1 l2.
  Lemma eqb_eq : ∀ cp1 cp2, eqb cp1 cp2 = true ↔ eq cp1 cp2.
```

```
Include HasEqBool2Dec.

Fixpoint ltList (l1 l2 : list positive) :=
  match l1, l2 with
  | nil, nil => False
  | (hd1 :: tl1), (hd2 :: tl2) => if (Pos.ltb hd1 hd2) then True
                                   else if (Pos.ltb hd2 hd1) then False
                                        else (ltList tl1 tl2)
  | nil, _ => True
  | _, nil => False
  end.

Definition lt (cp1 cp2 : t) :=
  ltList (PosSort.sort (CPToList cp1)) (PosSort.sort (CPToList cp2)).

Lemma ltListIrrefl : ∀ l,
  ltList l l → False.

Lemma ltIrrefl : Irreflexive lt.

Lemma ltAntiref : ∀ x, ¬ lt x x.

Lemma lengthOne : ∀ (l : list positive),
  length l = 1 → ∃ a , l = a :: nil.

Lemma lengthAtLeastOne : ∀ (l : list positive) n,
  length l = (S n) → ∃ a0 l0 , l = a0 :: l0.

Lemma ltListTrans : ∀ m x y z,
  length x = (S m) →
  length y = (S m) →
  length z = (S m) →
  ltList x y → ltList y z → ltList x z.

Lemma sortOK : ∀ m l, length l = m → length (sort l) = m.

Lemma ltTrans : Transitive lt.

Instance lt_strorder : StrictOrder lt.

Lemma eqListOK : ∀ l1 l2, eqList l1 l2 → l1 = l2.

Instance lt_compat : Proper (eq==>eq==>iff) lt.

Fixpoint compareList (l1 l2 : list positive) :=
  match l1, l2 with
  | nil, nil => Eq
  | (hd1 :: tl1), (hd2 :: tl2) => match Pos.compare hd1 hd2 with
                                  | Lt => Lt
                                  | Eq => compareList tl1 tl2
                                  | Gt => Gt
                                  end
```

```
    | nil, _ ⇒ Lt
    | _, nil ⇒ Gt
    end.
  Definition compare (cp1 cp2 : t) :=
    compareList (PosSort.sort (CPToList cp1)) (PosSort.sort (CPToList cp2)).
  Lemma compareListSpec : ∀ l1 l2,
    CompSpec eqList ltList l1 l2 (compareList l1 l2).
  Lemma compare_spec : ∀ cp1 cp2,
    CompSpec eq lt cp1 cp2 (compare cp1 cp2).
End SetOfTuplesOfPositiveOrderedType.
Module ST := MSetList.Make SetOfTuplesOfPositiveOrderedType.
```

Chapter 46

Library ColR

```
Require Import Recdef.
Require Import NArith.
Require Import sets.
Require Import tarski_to_col_theory.

Module SSWP := WPropertiesOn SetOfSetsOfPositiveOrderedType SS.

Module SSWEqP := WEqPropertiesOn SetOfSetsOfPositiveOrderedType SS.

Module SPWEqP := WEqPropertiesOn SetOfPairsOfPositiveOrderedType SP.

Definition have_pair_distinct_points (s : SS.elt) (sp : SP.t) :=
  SP.exists_ (fun p ⇒ (S.mem (fstpp p) s) && (S.mem (sndpp p) s)) sp.

Lemma proper_00 : ∀ s,
  Proper
  ((fun t1 t2 : SetOfPairsOfPositiveOrderedType.t ⇒
    Pos.eq (fstpp t1) (fstpp t2) ∧ Pos.eq (sndpp t1) (sndpp t2)) ==> eq)
  (fun p : SP.elt ⇒
   S.mem (fstpp p) s && S.mem (sndpp p) s).

Lemma proper_0 :
  Proper (S.Equal ==> eq ==> eq)
   have_pair_distinct_points .

Lemma proper_1 : ∀ s1 sp,
  Proper (S.Equal ==> eq)
  (fun s2 : SS.elt ⇒ have_pair_distinct_points (S.inter s1 s2) sp).

Definition exists_witness (f : SS.elt → bool) (s : SS.t) : option SS.elt :=
  SS.choose (SS.filter f s).

Lemma exists_witness_ok : ∀ e f s,
  Proper (S.Equal ==> eq) f →
  exists_witness f s = Some e → SS.In e s.

Definition get_suitable_pair_of_sets_aux (s1 : SS.elt)
```

```
                                                  (ss : SS.t)
                                                  (sp : SP.t)
                                                  : (option (SS.elt × SS.elt)) :=
  match ((exists_witness (fun s2 ⇒ let i := S.inter s1 s2 in
                                                have_pair_distinct_points i sp)) ss) with
    | None ⇒ None
    | Some s2 ⇒ Some(s1 , s2)
  end.

Definition get_suitable_pair_of_sets (ss : SS.t) (sp : SP.t)
                                            : (option (SS.elt × SS.elt)) :=
  match (exists_witness (fun s ⇒
                                       match (get_suitable_pair_of_sets_aux s (SS.remove s ss) sp)
with
                                         | None ⇒ false
                                         | _ ⇒ true
                                       end) ss) with
    | None ⇒ None
    | Some s1 ⇒ get_suitable_pair_of_sets_aux s1 (SS.remove s1 ss) sp
  end.

Definition eqop (p1 p2 : option SS.elt) :=
  match p1,p2 with
    | None, None ⇒ True
    | Some s1, Some s2 ⇒ True
    | _, _ ⇒ False
  end.

Lemma proper_2 : ∀ (f1 f2 : SS.elt → bool) (s1 s2 : SS.t),
  Proper (S.Equal ==> eq) f1 →
  Proper (S.Equal ==> eq) f2 →
  (∀ x, f1 x = f2 x) →
  SS.Equal s1 s2 →
  eqop (exists_witness f1 s1) (exists_witness f2 s2).

Definition eqopp (p1 p2 : option (SS.elt × SS.elt)) :=
  match p1,p2 with
    | None, None ⇒ True
    | Some s1, Some s2 ⇒ True
    | _, _ ⇒ False
  end.

Lemma proper_3 : Proper (S.Equal ==> SS.Equal ==> eq ==> eqopp) get_suitable_pair_of_sets_aux.

Lemma get_suitable_pair_of_sets_ok_1 : ∀ s1 s2 ss sp,
  get_suitable_pair_of_sets ss sp = Some(s1 , s2) →
  SS.In s1 ss.
```

```
Lemma get_suitable_pair_of_sets_ok_2 : ∀ s1 s2 ss sp,
  get_suitable_pair_of_sets ss sp = Some(s1 ,s2) →
  SS.In s2 (SS.remove s1 ss).

Function compute_new_set_of_sets_of_collinear_points (ss : SS.t)
                                                      (sp : SP.t)
                                                      {measure SS.cardinal ss}
                                                      : SS.t :=
  let suitablepairofsets := get_suitable_pair_of_sets ss sp in
    match suitablepairofsets with
      |None ⇒ ss
      |Some (s1 ,s2) ⇒ let auxsetofsets := SS.remove s2 (SS.remove s1 ss) in
                        let auxset := S.union s1 s2 in
                        let newss := SS.add auxset auxsetofsets in
                          compute_new_set_of_sets_of_collinear_points newss sp
    end.

Definition test_col (ss : SS.t) (sp : SP.t) p1 p2 p3 : bool :=
  let newss := compute_new_set_of_sets_of_collinear_points ss sp in
    SS.exists_ (fun s ⇒ S.mem p1 s && S.mem p2 s && S.mem p3 s) newss.

Section Col_refl.

Context `{CT:Col_theory}.

Lemma CTcol_permutation_5 : ∀ A B C : COLTpoint, CTCol A B C → CTCol A C B.

Lemma CTcol_permutation_2 : ∀ A B C : COLTpoint, CTCol A B C → CTCol C A B.

Lemma CTcol_permutation_3 : ∀ A B C : COLTpoint, CTCol A B C → CTCol C B A.

Lemma CTcol_permutation_4 : ∀ A B C : COLTpoint, CTCol A B C → CTCol B A C.

Lemma CTcol_trivial_1 : ∀ A B : COLTpoint, CTCol A A B.

Lemma CTcol_trivial_2 : ∀ A B : COLTpoint, CTCol A B B.

Definition ss_ok (ss : SS.t) (interp: positive → COLTpoint) :=
  ∀ s, SS.mem s ss = true →
  ∀ p1 p2 p3, S.mem p1 s && S.mem p2 s && S.mem p3 s = true →
    CTCol (interp p1) (interp p2) (interp p3).

Definition sp_ok (sp : SP.t) (interp: positive → COLTpoint) :=
  ∀ p, SP.mem p sp = true → interp (fstpp p) ≠ interp (sndpp p).

Lemma compute_new_set_of_sets_of_collinear_points_ok : ∀ ss sp interp,
  ss_ok ss interp → sp_ok sp interp →
  ss_ok (compute_new_set_of_sets_of_collinear_points ss sp) interp.

Lemma test_col_ok : ∀ ss sp interp p1 p2 p3,
  ss_ok ss interp → sp_ok sp interp →
  test_col ss sp p1 p2 p3 = true →
```

```
  CTCol (interp p1) (interp p2) (interp p3).
Lemma ss_ok_empty : ∀ interp,
  ss_ok SS.empty interp.
Lemma sp_ok_empty : ∀ interp,
  sp_ok SP.empty interp.
Lemma collect_cols :
  ∀ (A B C : COLTpoint) (HCol : CTCol A B C) pa pb pc ss (interp : positive →
COLTpoint),
  interp pa = A →
  interp pb = B →
  interp pc = C →
  ss_ok ss interp → ss_ok (SS.add (S.add pa (S.add pb (S.add pc S.empty))) ss) interp.
Lemma collect_diffs :
  ∀ (A B : COLTpoint) (H : A ≠ B) pa pb sp (interp : positive → COLTpoint),
  interp pa = A →
  interp pb = B →
  sp_ok sp interp → sp_ok (SP.add (pa , pb) sp) interp.
Definition list_assoc_inv :=
  (fix list_assoc_inv_rec (A:Type) (B:Set)
                          (eq_dec:∀ e1 e2:B, {e1 = e2} + {e1 ≠ e2})
                          (lst : list (prodT A B)) {struct lst} : B → A → A :=
  fun (key:B) (default:A) ⇒
    match lst with
      | nil ⇒ default
      | cons (pairT v e) l ⇒
        match eq_dec e key with
          | left _ ⇒ v
          | right _ ⇒ list_assoc_inv_rec A B eq_dec l key default
        end
    end).
Lemma positive_dec : ∀ (p1 p2:positive), {p1=p2}+{¬p1=p2}.
Definition interp (lvar : list (COLTpoint × positive)) (Default : COLTpoint) : positive
→ COLTpoint :=
  fun p ⇒ list_assoc_inv COLTpoint positive positive_dec lvar p Default.
Definition Col_tagged A B C := CTCol A B C.
Lemma Col_Col_tagged : ∀ A B C, CTCol A B C → Col_tagged A B C.
Lemma Col_tagged_Col : ∀ A B C, Col_tagged A B C → CTCol A B C.
Definition Diff_tagged (A B: COLTpoint) := A ≠ B.
Lemma Diff_Diff_tagged : ∀ A B , A ≠ B → Diff_tagged A B.
```

```
Lemma Diff_tagged_Diff : ∀ A B , Diff_tagged A B → A ≠ B.

Definition eq_tagged (lvar : list (COLTpoint × positive)) := lvar = lvar.

Lemma eq_eq_tagged : ∀ lvar, lvar = lvar → eq_tagged lvar.

Definition partition_ss e ss :=
  SS.partition (fun s ⇒ S.mem e s) ss.

Definition fst_ss (pair : SS.t × SS.t) :=
 match pair with
    |(a,b) ⇒ a
  end.

Definition snd_ss (pair : SS.t × SS.t) :=
 match pair with
    |(a,b) ⇒ b
  end.

Definition subst_in_set p1 p2 s := S.add p1 (S.remove p2 s).

Definition subst_in_ss_aux p1 p2 := (fun s ss ⇒ SS.add (subst_in_set p1 p2 s) ss).

Definition subst_in_ss p1 p2 ss :=
  let pair := partition_ss p2 ss in
  let fss := fst_ss(pair) in
  let sss := snd_ss(pair) in
  let newfss := SS.fold (subst_in_ss_aux p1 p2) fss SS.empty in
  SS.union newfss sss.

Lemma proper_4 : ∀ p, Proper (S.Equal ==> eq) (fun s : SS.elt ⇒ S.mem p s).

Lemma proper_5 : ∀ p, Proper (S.Equal ==> eq) (fun s : SS.elt ⇒ negb (S.mem p s)).

Lemma subst_ss_ok :
  ∀ (A B : COLTpoint) (H : A = B) pa pb ss (interp : positive → COLTpoint),
  interp pa = A →
  interp pb = B →
  ss_ok ss interp → ss_ok (subst_in_ss pa pb ss) interp.

Definition partition_sp_1 p sp :=
  SP.partition (fun e ⇒ Pos.eqb (fstpp e) p || Pos.eqb (sndpp e) p) sp.

Definition partition_sp_2 p sp :=
  SP.partition (fun e ⇒ Pos.eqb (fstpp e) p) sp.

Definition fst_sp (pair : SP.t × SP.t) :=
 match pair with
    |(a,b) ⇒ a
  end.

Definition snd_sp (pair : SP.t × SP.t) :=
 match pair with
```

```
    |(a , b) ⇒ b
  end.
Definition new_pair_1 pair (pos : positive) := (pos , sndpp(pair)).
Definition new_pair_2 pair (pos : positive) := (fstpp(pair) , pos).
Definition subst_in_sp_aux_1 := (fun pos pair sp ⇒ SP.add (new_pair_1 pair pos) sp).
Definition subst_in_sp_aux_2 := (fun pos pair sp ⇒ SP.add (new_pair_2 pair pos) sp).
Definition subst_in_sp p1 p2 sp :=
  let pair_1 := partition_sp_1 p2 sp in
  let sp_to_modify := fst_sp(pair_1) in
  let sp_to_keep := snd_sp(pair_1) in
  let pair_2 := partition_sp_2 p2 sp_to_modify in
  let sp_to_modify_f := fst_sp(pair_2) in
  let sp_to_modify_s := snd_sp(pair_2) in
  let newsp_to_modify_f := SP.fold (subst_in_sp_aux_1 p1) sp_to_modify_f SP.empty in
  let newsp_to_modify_s := SP.fold (subst_in_sp_aux_2 p1) sp_to_modify_s SP.empty in
  SP.union (SP.union newsp_to_modify_f newsp_to_modify_s) sp_to_keep.
Lemma proper_6 : ∀ p, Proper ((fun t1 t2 : SetOfPairsOfPositiveOrderedType.t ⇒
                                  Pos.eq (fstpp t1) (fstpp t2) ∧ Pos.eq (sndpp t1) (sndpp
t2)) ==> eq)
                                  (fun e : SP.elt ⇒ (fstpp e =? p)%positive || (sndpp
e =? p)%positive).
Lemma proper_7 : ∀ p, Proper ((fun t1 t2 : SetOfPairsOfPositiveOrderedType.t ⇒
                                  Pos.eq (fstpp t1) (fstpp t2) ∧ Pos.eq (sndpp t1) (sndpp
t2)) ==> eq)
                                  (fun x : SP.elt ⇒ negb ((fstpp x =? p)%positive ||
(sndpp x =? p)%positive)).
Lemma proper_8 : ∀ p, Proper ((fun t1 t2 : SetOfPairsOfPositiveOrderedType.t ⇒
                                  Pos.eq (fstpp t1) (fstpp t2) ∧ Pos.eq (sndpp t1) (sndpp
t2)) ==> eq)
                                  (fun e : SP.elt ⇒ (fstpp e =? p)%positive).
Lemma proper_9 : ∀ p, Proper ((fun t1 t2 : SetOfPairsOfPositiveOrderedType.t ⇒
                                  Pos.eq (fstpp t1) (fstpp t2) ∧ Pos.eq (sndpp t1) (sndpp
t2)) ==> eq)
                                  (fun x : SP.elt ⇒ negb (fstpp x =? p)%positive).
Lemma subst_sp_ok :
  ∀ (A B : COLTpoint) (H : A = B) pa pb sp (interp : positive → COLTpoint),
  interp pa = A →
  interp pb = B →
  sp_ok sp interp → sp_ok (subst_in_sp pa pb sp) interp.
End Col_refl.
```

```
Ltac add_to_distinct_list x xs :=
  match xs with
    | nil ⇒ constr:(x::xs)
    | x::_ ⇒ fail 1
    | ?y::?ys ⇒ let zs := add_to_distinct_list x ys in constr:(y::zs)
  end.

Ltac collect_points_list Tpoint xs :=
  match goal with
    | N : Tpoint ⊢ _ ⇒ let ys := add_to_distinct_list N xs in
                                collect_points_list Tpoint ys
    | _ ⇒ xs
  end.

Ltac collect_points Tpoint := collect_points_list Tpoint (@nil Tpoint).

Ltac number_aux Tpoint lvar cpt :=
  match constr:lvar with
    | nil ⇒ constr:(@nil (prodT Tpoint positive))
    | cons ?H ?T ⇒ let scpt := eval vm_compute in (Pos.succ cpt) in
                      let lvar2 := number_aux Tpoint T scpt in
                        constr:(cons (@pairT Tpoint positive H cpt) lvar2)
  end.

Ltac number Tpoint lvar := number_aux Tpoint lvar (1%positive).

Ltac build_numbered_points_list Tpoint := let lvar := collect_points Tpoint in number
Tpoint lvar.

Ltac List_assoc Tpoint elt lst :=
  match constr:lst with
    | nil ⇒ fail
    | (cons (@pairT Tpoint positive ?X1 ?X2) ?X3) ⇒
      match constr:(elt = X1) with
        | (?X1 = ?X1) ⇒ constr:X2
        | _ ⇒ List_assoc Tpoint elt X3
      end
  end.

Ltac assert_ss_ok Tpoint Col lvar :=
  repeat
  match goal with
    | HCol : Col ?A ?B ?C, HOK : ss_ok ?SS ?Interp ⊢ _ ⇒
        let pa := List_assoc Tpoint A lvar in
        let pb := List_assoc Tpoint B lvar in
        let pc := List_assoc Tpoint C lvar in
         apply (@Col_Col_tagged Tpoint Col) in HCol;
         apply (collect_cols A B C HCol pa pb pc SS Interp) in HOK; try reflexivity
```

```
  end.
Ltac assert_sp_ok Tpoint Col lvar :=
  repeat
  match goal with
    | HDiff : ?A ≠ ?B, HOK : sp_ok ?SP ?Interp ⊢ _ ⇒
        let pa := List_assoc Tpoint A lvar in
        let pb := List_assoc Tpoint B lvar in
          apply (@Diff_Diff_tagged Tpoint) in HDiff;
          apply (collect_diffs A B HDiff pa pb SP Interp) in HOK; try reflexivity
  end.
Ltac subst_in_cols Tpoint Col :=
  repeat
  match goal with
    | HOKSS : ss_ok ?SS ?Interp, HOKSP : sp_ok ?SP ?Interp, HL : eq_tagged ?Lvar, HEQ
: ?A = ?B ⊢ _ ⇒
      let pa := List_assoc Tpoint A Lvar in
      let pb := List_assoc Tpoint B Lvar in
        apply (subst_ss_ok A B HEQ pa pb SS Interp) in HOKSS; try reflexivity;
        apply (subst_sp_ok A B HEQ pa pb SP Interp) in HOKSP; try reflexivity;
        subst B
  end.
Ltac clear_cols_aux Tpoint Col :=
  repeat
  match goal with
    | HOKSS : ss_ok ?SS ?Interp, HOKSP : sp_ok ?SP ?Interp, HL : eq_tagged ?Lvar ⊢ _
⇒
      clear HOKSS; clear HOKSP; clear HL
  end.
Ltac tag_hyps_gen Tpoint Col :=
  repeat
  match goal with
    | HDiff : ?A ≠ ?B ⊢ _ ⇒ apply (@Diff_Diff_tagged Tpoint) in HDiff
    | HCol : Col ?A ?B ?C ⊢ _ ⇒ apply (@Col_Col_tagged Tpoint Col) in HCol
  end.
Ltac untag_hyps_gen Tpoint Col :=
  repeat
  match goal with
    | HDiff : Diff_tagged ?A ?B ⊢ _ ⇒ apply (@Diff_tagged_Diff Tpoint) in HDiff
    | HCol : Col_tagged ?A ?B ?C ⊢ _ ⇒ apply (@Col_tagged_Col Tpoint Col) in HCol
  end.

Ltac show_all :=
```

```
  repeat
  match goal with
    | Hhidden : Something ⊢ _ ⇒ show Hhidden
  end.

Ltac clear_cols_gen Tpoint Col := show_all; clear_cols_aux Tpoint Col.

Ltac Col_refl Tpoint Col :=
  match goal with
    | Default : Tpoint ⊢ Col ?A ?B ?C ⇒
          let lvar := build_numbered_points_list Tpoint in
          let pa := List_assoc Tpoint A lvar in
          let pb := List_assoc Tpoint B lvar in
          let pc := List_assoc Tpoint C lvar in
          let c := ((vm_compute;reflexivity) || fail 2 "Can not be deduced") in
          let HSS := fresh in
            assert (HSS := @ss_ok_empty Tpoint Col (interp lvar Default)); assert_ss_ok
Tpoint Col lvar;
          let HSP := fresh in
            assert (HSP := @sp_ok_empty Tpoint (interp lvar Default)); assert_sp_ok Tpoint
Col lvar;
            match goal with
              | HOKSS : ss_ok ?SS ?Interp, HOKSP : sp_ok ?SP ?Interp ⊢ _ ⇒
                apply (test_col_ok SS SP (interp lvar Default) pa pb pc ); [assumption|assumption|c]
            end
  end.

Ltac deduce_cols_hide_aux Tpoint Col :=
  match goal with
    | Default : Tpoint ⊢ _ ⇒
          let lvar := build_numbered_points_list Tpoint in
          let HSS := fresh in
            assert (HSS := @ss_ok_empty Tpoint Col (interp lvar Default)); assert_ss_ok
Tpoint Col lvar;
          let HSP := fresh in
            assert (HSP := @sp_ok_empty Tpoint (interp lvar Default)); assert_sp_ok Tpoint
Col lvar;
          let HL := fresh in
            assert (HL : lvar = lvar) by reflexivity;
            apply (@eq_eq_tagged Tpoint) in HL;
            hide HSS; hide HSP; hide HL
  end.

Ltac deduce_cols_hide_gen Tpoint Col := deduce_cols_hide_aux Tpoint Col.

Ltac update_cols_aux Tpoint Col :=
```

```
  match goal with
    | HOKSS : ss_ok ?SS ?Interp, HOKSP : sp_ok ?SP ?Interp, HEQ : eq_tagged ?Lvar ⊢
_ ⇒
        assert_ss_ok Tpoint Col Lvar; assert_sp_ok Tpoint Col Lvar; subst_in_cols Tpoint
Col; hide HOKSS; hide HOKSP; hide HEQ
  end.

Ltac update_cols_gen Tpoint Col := show_all; update_cols_aux Tpoint Col.

Ltac cols_aux Tpoint Col :=
  match goal with
    | HOKSS : ss_ok ?SS ?Interp, HOKSP : sp_ok ?SP ?Interp, HL : eq_tagged ?Lvar ⊢
Col ?A ?B ?C ⇒
        let pa := List_assoc Tpoint A Lvar in
        let pb := List_assoc Tpoint B Lvar in
        let pc := List_assoc Tpoint C Lvar in
        let c := ((vm_compute;reflexivity) || fail 1 "Can not be deduced") in
          apply (test_col_ok SS SP Interp pa pb pc ); [assumption|assumption|c];
          hide HOKSS; hide HOKSP; hide HL
  end.

Ltac cols_gen Tpoint Col := show_all; cols_aux Tpoint Col.

Ltac Col_refl_test Tpoint Col := deduce_cols_hide_gen Tpoint Col; cols_gen Tpoint Col.
```

Chapter 47

Library tarski_to_coapp_theory

Require Export Ch06_out_lines.
Require Import tactics_axioms.

Section Tarski_is_a_Coapp_theory.

Context `{MT:**Tarski_neutral_dimensionless**}.
Context `{$EqDec$:**EqDecidability** Tpoint}.

Definition diff : arity Tpoint 2 := fun A B : Tpoint ⇒ $A \neq B$.

Lemma diff_perm_1 : ∀ A B, app_1_n diff A B → app_n_1 diff B A.

Lemma diff_perm_2 : ∀ A B (X : cartesianPower Tpoint 0), app_2_n diff A B X → app_2_n diff B A X.

Definition col : arity Tpoint 3 := Col.

Lemma col_perm_1 : ∀ A (X : cartesianPower Tpoint 2), app_1_n col A X → app_n_1 col X A.

Lemma col_perm_2 : ∀ A B (X : cartesianPower Tpoint 1), app_2_n col A B X → app_2_n col B A X.

Lemma diff_or_col : ∀ A (X : cartesianPower Tpoint 2), app_1_n col A X ∨ app diff X.

Lemma pseudo_trans : ∀ A B C (X : cartesianPower Tpoint 1),
  app_1_n diff A X → app_2_n col B A X → app_2_n col C A X → app_2_n col B C X.

Global Instance Tarski_is_a_Coapp_theory : (**Coapp_theory** (Build_Arity Tpoint 0) (Build_Coapp_predicate (Build_Arity Tpoint 0) diff Col)).

End Tarski_is_a_Coapp_theory.

Chapter 48

Library tarski_to_col_theory

```
Require Export Ch06_out_lines.
Require Import tactics_axioms.
```

In this file we prove that Tarski neutral dimensionless is a Col_theory.

```
Section Tarski_is_a_Col_theory.

Context `{MT:Tarski_neutral_dimensionless}.
Context `{EqDec:EqDecidability Tpoint}.

Global Instance Tarski_is_a_Col_theory : (Col_theory Tpoint Col).

End Tarski_is_a_Col_theory.
```

Chapter 49

Library Tagged_predicates

Require Export quadrilaterals.

Section Tagged_predicates.

Context `{MT:**Tarski_2D_euclidean**}.
Context `{$EqDec$:**EqDecidability** Tpoint}.

Definition Diff_tagged (A B: Tpoint) := $A \neq B$.

Lemma Diff_Diff_tagged : ∀ A B , $A \neq B$ → Diff_tagged A B.

Lemma Diff_tagged_Diff : ∀ A B , Diff_tagged A B → $A \neq B$.

Lemma Diff_perm :
  ∀ (A B: Tpoint),
  $A \neq B$ →
  $A \neq B \wedge B \neq A$.

Definition Cong_tagged A B C D := Cong A B C D.

Lemma Cong_Cong_tagged : ∀ A B C D, Cong A B C D → Cong_tagged A B C D.

Lemma Cong_tagged_Cong : ∀ A B C D, Cong_tagged A B C D → Cong A B C D.

Definition Bet_tagged A B C := Bet A B C.

Lemma Bet_Bet_tagged : ∀ A B C, Bet A B C → Bet_tagged A B C.

Lemma Bet_tagged_Bet : ∀ A B C, Bet_tagged A B C → Bet A B C.

Definition Col_tagged A B C := Col A B C.

Lemma Col_Col_tagged : ∀ A B C, Col A B C → Col_tagged A B C.

Lemma Col_tagged_Col : ∀ A B C, Col_tagged A B C → Col A B C.

Definition NCol_tagged A B C := ¬ Col A B C.

Lemma NCol_NCol_tagged : ∀ A B C, ¬ Col A B C → NCol_tagged A B C.

Lemma NCol_tagged_NCol : ∀ A B C, NCol_tagged A B C → ¬ Col A B C.

Definition Mid_tagged A B C := is_midpoint A B C.

Lemma Mid_Mid_tagged : ∀ A B C, is_midpoint A B C → Mid_tagged A B C.

Lemma Mid_tagged_Mid : ∀ A B C, Mid_tagged A B C → is_midpoint A B C.

Definition Per_tagged A B C := Per A B C.

Lemma Per_Per_tagged : ∀ A B C, Per A B C → Per_tagged A B C.

Lemma Per_tagged_Per : ∀ A B C, Per_tagged A B C → Per A B C.

Definition Perp_in_tagged X A B C D := Perp_in X A B C D.

Lemma Perp_in_Perp_in_tagged : ∀ X A B C D, Perp_in X A B C D → Perp_in_tagged X A B C D.

Lemma Perp_in_tagged_Perp_in : ∀ X A B C D, Perp_in_tagged X A B C D → Perp_in X A B C D.

Definition Perp_tagged A B C D := Perp A B C D.

Lemma Perp_Perp_tagged : ∀ A B C D, Perp A B C D → Perp_tagged A B C D.

Lemma Perp_tagged_Perp : ∀ A B C D, Perp_tagged A B C D → Perp A B C D.

Definition Par_strict_tagged A B C D := Par_strict A B C D.

Lemma Par_strict_Par_strict_tagged : ∀ A B C D, Par_strict A B C D → Par_strict_tagged A B C D.

Lemma Par_strict_tagged_Par_strict : ∀ A B C D, Par_strict_tagged A B C D → Par_strict A B C D.

Definition Par_tagged A B C D := Par A B C D.

Lemma Par_Par_tagged : ∀ A B C D, Par A B C D → Par_tagged A B C D.

Lemma Par_tagged_Par : ∀ A B C D, Par_tagged A B C D → Par A B C D.

Definition Plg_tagged A B C D := Parallelogram A B C D.

Lemma Plg_Plg_tagged : ∀ A B C D, Parallelogram A B C D → Plg_tagged A B C D.

Lemma Plg_tagged_Plg : ∀ A B C D, Plg_tagged A B C D → Parallelogram A B C D.

End Tagged_predicates.

Chapter 50

Library exercises

```
Require Import triangle_midpoints_theorems.

Section Exercises.

Context `{MT:Tarski_2D_euclidean}.
Context `{EqDec:EqDecidability Tpoint}.
Context `{InterDec:InterDecidability Tpoint Col}.

Lemma Per_mid_rectangle : ∀ A B C I J K,
  A ≠ B →
  B ≠ C →
  Per B A C →
  is_midpoint I B C →
  is_midpoint J A C →
  is_midpoint K A B →
  Rectangle A J I K.

End Exercises.
```

Chapter 51

Library midpoint_thales

```
Require Import triangle_midpoints_theorems.

Section T_42.

Context `{MT:Tarski_2D_euclidean}.
Context `{EqDec:EqDecidability Tpoint}.
Context `{InterDec:InterDecidability Tpoint Col}.

Lemma midpoint_thales : ∀ o a b c : Tpoint,
   ¬ Col a b c →
   is_midpoint o a b →
   Cong o a o c →
   Per a c b.

Lemma midpoint_thales_reci :
  ∀ a b c o: Tpoint,
   Per a c b →
   ¬Col a b c →
   is_midpoint o a b →
   Cong o a o b ∧ Cong o b o c.

End T_42.
```