Intersecting Chords Theorem

Lukas Bulwahn

December 17, 2016

Abstract

This entry provides a geometric proof of the intersecting chords theorem. The theorem states that when two chords intersect each other inside a circle, the products of their segments are equal.

After a short review of existing proofs in the literature [1, 3–5], I decided to use a proof approach that employs reasoning about lengths of line segments, the orthogonality of two lines and Pythagoras Law. Hence, one can understand the formalized proof easily with the knowledge of a few general geometric facts that are commonly taught in high-school.

This theorem is the 55th theorem of the Top 100 Theorems list.

Contents

1 Introduction

The intersecting chords theorem states:

> When two chords intersect each other inside a circle, the products of their segments are equal.

To prove this theorem in Isabelle, I reviewed existing formalizations in theorem provers and proofs in the literature [1, 3–5]. At the time of this AFP submission, the formalization of geometry in Isabelle is limited to only a few concepts and theorems. Hence, I selected to formalize the proof approach that fitted best to the already existing geometry formalizations.

The proof in HOL Light [3] simply unfolds the involved geometric predicates and then proves the theorem using only algebraic computations on real numbers. By a quick and shallow inspection of the proof script without executing the proof script step by step in HOL Light, I could not understand the proof script well enough to re-write the proof in Isabelle. As running the script in HOL Light seemed too involved to me, I ignored HOL Light's proof approach and considered the other approaches in the literature.

The first proof approach [5] that I found in the literature employs similarity of triangles, the inscribed angle theorem, and basic reasoning with angles. The intersecting chords theorem only consists of two reasoning steps after stating the geometric observations about angles. However, the proof requires to formalize the concept of similarity of triangles, extend the existing formalization of angles, and prove the inscribed angle theorem. So, I abandoned this proof approach and considered the second proof approach.

The second proof approach [5] needs only basic geometric reasoning about lengths of line segments, the orthogonality of two lines and Pythagoras Law. More specifically, one must prove that the line that goes through the apex and the midpoint of the base in an isosceles triangle is orthogonal to the base. This is easily derived from the property of an isosceles triangle using the congruence properties of triangles, which is already formalized in AFP's Triangle entry [2]. Furthermore, Pythagoras Law is a special case of the Law of Cosines, which is already formalized in AFP's Triangle entry.

Ultimately, I decided to use this second proof approach, which I sketch in more detail in the next subsection.

1.1 Informal Proof Sketch

The proof of the intersecting chords theorem relies on the following observation which is depicted in Figure 1.

Instead of considering *two* arbitrary chords intersecting, consider *one* arbitrary chord with endpoints S and T on a circle with center C and one arbitrary point X on the chord ST. This point X on the chord creates two line segments on this chord, the left part SX, and the right part XT. Without loss of generality, we can assume that SX is longer that XT, as shown in Figure 1.

The key lemma for the intersecting chords theorem provides a closed expression for the length of these two line segment using the distances of the chord endpoints and the point to the center C, i.e., the lemma states:

$$\overline{SX} \cdot \overline{XT} = \overline{SC}^2 - \overline{XC}^2.$$

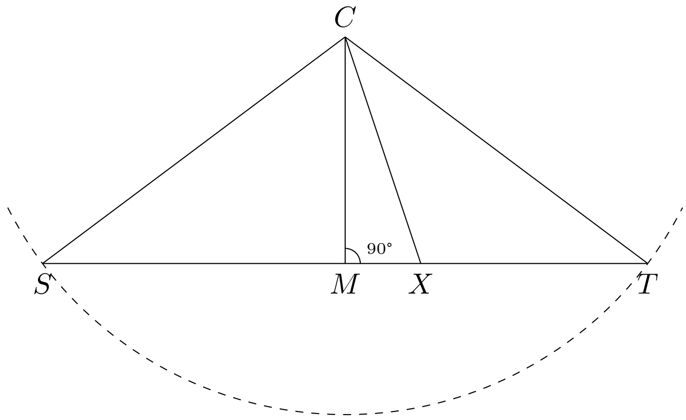

Figure 1: Key Lemma states $\overline{SX} \cdot \overline{XT} = \overline{SC}^2 - \overline{XC}^2$

To prove this fact, we consider the midpoint M of the chord ST. First, as M is the midpoint, $\overline{SM}$ and $\overline{TM}$ are equal. Second, we observe that the lengths of the line segments SX and XT are:

$$\overline{SX} = \overline{SM} + \overline{MX} \text{ and } \overline{XT} = \overline{TM} - \overline{MX} = \overline{SM} - \overline{MX}.$$

Third, the Pythagoras law for the triangles SMC and XMC states:

$$\overline{SM}^2 + \overline{MC}^2 = \overline{SC}^2 \text{ and } \overline{XM}^2 + \overline{MC}^2 = \overline{XC}^2.$$

Finally, the product can be expressed as:

$$\overline{SX} \cdot \overline{XT} = (\overline{SM} + \overline{MX}) \cdot (\overline{TM} - \overline{MX}) = \overline{SM}^2 - \overline{MX}^2 = (\overline{SC}^2 - \overline{MC}^2) - (\overline{XC}^2 - \overline{MC}^2) = \overline{SC}^2 - \overline{XC}^2.$$

The intersecting chord theorem now follows directly from this lemma: as the distances SC and XC for two arbitrary chords intersecting at X are equal, also the products of the chord segments are equal.

2 Intersecting Chord Theorem

theory *Chord-Segments*
imports
  ~~/src/HOL/Analysis/Analysis
  ../Triangle/Triangle
begin

2.1 Preliminaries

2.1.1 Addition to Fields

lemma *divide-eq-minus-1-iff*:

fixes *a b* :: *'a* :: *field*
shows $(a\ /\ b = -\ 1) \longleftrightarrow b \neq 0 \wedge a = -\ b$
using *divide-eq-1-iff* **by** *fastforce*

2.1.2 Addition to Power

lemma *power2-eq-iff-nonneg*:
fixes *x y* :: *'a* :: *linordered-semidom*
assumes $0 \leq x$ $0 \leq y$
shows $(x\ \hat{}\ 2 = y\ \hat{}\ 2) \longleftrightarrow x = y$
using *assms power2-eq-imp-eq* **by** *blast*

2.1.3 Addition to Transcendental

lemma *arccos-minus-abs*:
assumes $|x| \leq 1$
shows $arccos\ (-\ x) = pi\ -\ arccos\ x$
using *assms* **by** (*simp add*: *arccos-minus*)

2.1.4 Additions to Convex Euclidean Space

lemma *between-same*:
assumes *between* (*a*, *a*) *x*
shows $x = a$
using *assms* **unfolding** *between-mem-segment* **by** *simp*

lemma *betweenI*:
assumes $0 \leq u$ $u \leq 1$ $x = (1\ -\ u) *_R a + u *_R b$
shows *between* (*a*, *b*) *x*
using *assms* **unfolding** *between-def closed-segment-def* **by** *auto*

lemma *betweenE*:
assumes *between* (*a*, *b*) *x*
obtains *u* **where** $0 \leq u$ $u \leq 1$ $x = (1\ -\ u) *_R a + u *_R b$
using *assms* **unfolding** *between-def closed-segment-def* **by** *auto*

lemma *betweenI'*:
assumes $0 \leq u$ $u \leq 1$ $x = u *_R a + (1\ -\ u) *_R b$
shows *between* (*a*, *b*) *x*
proof −
from *assms* **have** *between* (*b*, *a*) *x* **by** (*auto intro*: *betweenI*)
from *this* **show** *between* (*a*, *b*) *x* **by** (*simp add*: *between-commute*)
qed

lemma *betweenE'*:
assumes *between* (*a*, *b*) *x*
obtains *u* **where** $0 \leq u$ $u \leq 1$ $x = u *_R a + (1\ -\ u) *_R b$
proof −
from *assms* **have** *between* (*b*, *a*) *x* **by** (*simp add*: *between-commute*)
from *this* **obtain** *u* **where** $0 \leq u$ $u \leq 1$ $x = u *_R a + (1\ -\ u) *_R b$

by (*auto elim*: *betweenE*)
from *this that* **show** *thesis* **by** *blast*
qed

lemma *between-implies-scaled-diff*:
assumes *between* (S, T) X *between* (S, T) Y $S \neq Y$
obtains c **where** $(X - Y) = c *_R (S - Y)$
proof −
from ‹*between* (S, T) X› **obtain** u_X **where** X: $X = u_X *_R S + (1 - u_X) *_R T$
by (*auto elim*: *betweenE'*)
from ‹*between* (S, T) Y› **obtain** u_Y **where** Y: $Y = u_Y *_R S + (1 - u_Y) *_R T$
by (*auto elim*: *betweenE'*)
have $X - Y = (u_X - u_Y) *_R (S - T)$
proof −
from X Y **have** $X - Y = u_X *_R S - u_Y *_R S + ((1 - u_X) *_R T - (1 - u_Y) *_R T)$ **by** *simp*
also have $\ldots = (u_X - u_Y) *_R S - (u_X - u_Y) *_R T$ **by** (*simp add*: *scaleR-left.diff*)
finally show *?thesis* **by** (*simp add*: *real-vector.scale-right-diff-distrib*)
qed
moreover from Y **have** $S - Y = (1 - u_Y) *_R (S - T)$
by (*simp add*: *real-vector.scale-left-diff-distrib real-vector.scale-right-diff-distrib*)
moreover note ‹$S \neq Y$›
ultimately have $(X - Y) = ((u_X - u_Y) / (1 - u_Y)) *_R (S - Y)$ **by** *auto*
from *this that* **show** *thesis* **by** *blast*
qed

lemma *between-swap*:
fixes A B X Y :: *'a::euclidean-space*
assumes *between* (A, B) X
assumes *between* (A, B) Y
shows *between* (X, B) $Y \longleftrightarrow$ *between* (A, Y) X
using *assms* **by** (*auto simp add*: *between*)

lemma *betweenE-if-dist-leq*:
fixes A B X :: *'a::euclidean-space*
assumes *between* (A, B) X
assumes *dist* A $X \leq$ *dist* B X
obtains u **where** $1 / 2 \leq u$ $u \leq 1$ **and** $X = u *_R A + (1 - u) *_R B$
proof (*cases* $A = B$)
assume $A \neq B$
from ‹*between* (A, B) X› **obtain** u **where** u: $u \geq 0$ $u \leq 1$ **and** X: $X = u *_R A + (1 - u) *_R B$
by (*auto elim*: *betweenE'*)
from X **have** $X = B + u *_R (A - B)$ **and** $X = A + (u - 1) *_R (A - B)$
by (*simp add*: *scaleR-diff-left real-vector.scale-right-diff-distrib*)+
from ‹$X = B + u *_R (A - B)$› **have** *dist-B*: *dist* B X = *norm* $(u *_R (A - B))$

```
    by (auto simp add: dist-norm)
  from ‹X = A + (u − 1) *R (A − B)› have dist-A: dist A X = norm ((u − 1)
*R (A − B))
    by (auto simp add: dist-norm)
  from ‹A ≠ B› have norm (A − B) > 0 by auto
  from this ‹dist A X ≤ dist B X› have u ≥ 1 / 2
    using dist-A dist-B by simp
  from this ‹u ≤ 1› X that show thesis by blast
next
  assume A = B
  define u :: real where u = 1
  from ‹between (A, B) X› ‹A = B› have 1 / 2 ≤ u u ≤ 1 X = u *R A + (1 −
u) *R B
    unfolding u-def by (auto simp add: between-same)
  from this that show thesis by blast
qed

lemma dist-geq-iff-midpoint-in-between:
  fixes A B X :: 'a::euclidean-space
  assumes between (A, B) X
  shows dist A X ≤ dist B X ⟷ between (X, B) (midpoint A B)
proof
  assume dist A X ≤ dist B X
  from ‹between (A, B) X› this obtain u
    where u: 1 / 2 ≤ u u ≤ 1 and X: X = u *R A + (1 − u) *R B
    using betweenE-if-dist-leq by blast
  have M: midpoint A B = (1 / 2) *R A + (1 / 2) *R B
    unfolding midpoint-def by (simp add: scaleR-add-right)
  from ‹1 / 2 ≤ u› have 1: midpoint A B = (1 / (2 * u)) *R X + (1 − (1 / (2
* u))) *R B
  proof −
    have (2 − u * 2) / (2 * u) = 1 / u − u / u
      using u(1) by (simp add: diff-divide-distrib)
    also have ... = 1 / u − 1 using u(1) by auto
    finally have (2 − u * 2) / (2 * u) = 1 / u − 1 .
    from ‹1 / 2 ≤ u› this show ?thesis
      using X M by (simp add: scaleR-add-right scaleR-add-left[symmetric])
  qed
  moreover from u have 2: (1 / (2 * u)) ≥ 0 (1 / (2 * u)) ≤ 1 by auto
  ultimately show between (X, B) (midpoint A B)
    using betweenI' by blast
next
  assume between (X, B) (midpoint A B)
  from this have between (A, midpoint A B) X
    using ‹between (A, B) X› between-midpoint(1) between-swap by blast
  from this have dist A X ≤ dist A (midpoint A B)
    using between zero-le-dist by force
  also have dist A (midpoint A B) ≤ dist B (midpoint A B)
    by (simp add: dist-midpoint)
```

also from ⟨*between* (X, B) (*midpoint* A B)⟩ **have** *dist* B (*midpoint* A B) $\leq$ *dist* B X
using *between zero-le-dist* **by** (*metis add.commute dist-commute le-add-same-cancel1*)
finally show *dist* A X $\leq$ *dist* B X **.**
qed

2.1.5 Additions to Angles

lemma *vangle-scales*:
assumes $0 < c$
shows *vangle* $(c *_R v_1)$ v_2 = *vangle* v_1 v_2
using *assms* **unfolding** *vangle-def* **by** *auto*

lemma *vangle-inverse*:
vangle $(- v_1)$ v_2 = *pi* $-$ *vangle* v_1 v_2
proof $-$
have $|v_1 \cdot v_2$ / (*norm* v_1 $*$ *norm* $v_2)| \leq 1$
proof *cases*
assume $v_1 \neq 0 \wedge v_2 \neq 0$
from *this* **show** *?thesis* **by** (*simp add*: *Cauchy-Schwarz-ineq2*)
next
assume $\neg (v_1 \neq 0 \wedge v_2 \neq 0)$
from *this* **show** *?thesis* **by** *auto*
qed
from *this* **show** *?thesis*
unfolding *vangle-def* **by** (*simp add*: *arccos-minus-abs*)
qed

lemma *orthogonal-iff-angle*:
shows *orthogonal* $(A - B)$ $(C - B)$ $\longleftrightarrow$ *angle* A B C = *pi* / *2*
unfolding *angle-def* **by** (*auto simp only*: *orthogonal-iff-vangle*)

lemma *angle-inverse*:
assumes *between* (A, C) B
assumes $A \neq B$ $B \neq C$
shows *angle* A B D = *pi* $-$ *angle* C B D
proof $-$
from ⟨*between* (A, C) B⟩ **obtain** u **where** u: $u \geq 0$ $u \leq 1$
and X: $B = u *_R A + (1 - u) *_R C$ **by** (*auto elim*: *betweenE'*)
from ⟨$A \neq B$⟩ ⟨$B \neq C$⟩ X **have** $u \neq 0$ $u \neq 1$ **by** *auto*
have $0 < ((1 - u) / u)$
using ⟨$u \neq 0$⟩ ⟨$u \neq 1$⟩ ⟨$u \geq 0$⟩ ⟨$u \leq 1$⟩ **by** *simp*
from X **have** $A - B = - (1 - u) *_R (C - A)$
by (*simp add*: *real-vector.scale-right-diff-distrib real-vector.scale-left-diff-distrib*)
moreover from X **have** $C - B = u *_R (C - A)$
by (*simp add*: *scaleR-diff-left real-vector.scale-right-diff-distrib*)
ultimately have $A - B = - (((1 - u) / u) *_R (C - B))$
using ⟨$u \neq 0$⟩ **by** *simp* (*metis minus-diff-eq real-vector.scale-minus-left*)
from *this* **have** *vangle* $(A - B)$ $(D - B)$ = *pi* $-$ *vangle* $(C - B)$ $(D - B)$

using ‹$0 < (1 - u) / u$› **by** (*simp add*: *vangle-inverse vangle-scales*)
from *this* **show** *?thesis*
unfolding *angle-def* **by** *simp*
qed

lemma *strictly-between-implies-angle-eq-pi*:
assumes *between* (A, C) B
assumes $A \neq B$ $B \neq C$
shows *angle* A B C = *pi*
proof –
from ‹*between* (A, C) B› **obtain** u **where** u: $u \geq 0$ $u \leq 1$
and X: $B = u *_R A + (1 - u) *_R C$ **by** (*auto elim*: *betweenE'*)
from ‹$A \neq B$› ‹$B \neq C$› X **have** $u \neq 0$ $u \neq 1$ **by** *auto*
from ‹$A \neq B$› ‹$B \neq C$› ‹*between* (A, C) B› **have** $A \neq C$
using *between-same* **by** *blast*
from X **have** $A - B = - (1 - u) *_R (C - A)$
by (*simp add*: *real-vector.scale-right-diff-distrib real-vector.scale-left-diff-distrib*)
moreover from *this* **have** *dist* A B = *norm* $((1 - u) *_R (C - A))$
using ‹$u \geq 0$› ‹$u \leq 1$› **by** (*simp add*: *dist-norm*)
moreover from X **have** $C - B = u *_R (C - A)$
by (*simp add*: *scaleR-diff-left real-vector.scale-right-diff-distrib*)
moreover from *this* **have** *dist* C B = *norm* $(u *_R (C - A))$
by (*simp add*: *dist-norm*)
ultimately have $(A - B) \cdot (C - B)$ / (*dist* A B $*$ *dist* C B) $= u * (u - 1) / (|1 - u| * |u|)$
using ‹$A \neq C$› **by** (*simp add*: *dot-square-norm power2-eq-square*)
also have $\ldots = - 1$
using ‹$u \neq 0$› ‹$u \neq 1$› ‹$u \geq 0$› ‹$u \leq 1$› **by** (*simp add*: *divide-eq-minus-1-iff*)
finally show *?thesis*
unfolding *angle-altdef* **by** *simp*
qed

2.1.6 Additions to Triangles

lemma *pythagoras*:
fixes A B C :: *'a* :: *euclidean-space*
assumes *orthogonal* $(A - C)$ $(B - C)$
shows (*dist* B C) ^ 2 + (*dist* C A) ^ 2 = (*dist* A B) ^ 2
proof –
from *assms* **have** *cos* (*angle* A C B) = 0
by (*metis orthogonal-iff-angle cos-pi-half*)
from *this* **show** *?thesis*
by (*simp add*: *cosine-law-triangle*[*of* A B C] *dist-commute*)
qed

lemma *isosceles-triangle-orthogonal-on-midpoint*:
fixes A B C :: *'a::euclidean-space*
assumes *dist* C A = *dist* C B
shows *orthogonal* (C − *midpoint* A B) (A − *midpoint* A B)

proof (*cases* $A = B$)
  assume $A \neq B$
  let *?M* = *midpoint A B*
  have *angle A ?M C* = *pi* − *angle B ?M C*
    using ‹$A \neq B$› *angle-inverse between-midpoint*(*1*) *midpoint-eq-endpoint* **by** *metis*
  moreover have *angle A ?M C* = *angle C ?M B*
  proof −
    have *congruence*: *congruent-triangle C A ?M C B ?M*
    proof (*rule congruent-triangleI-sss*)
      show *dist C A* = *dist C B* **using** *assms* **.**
      show *dist A ?M* = *dist B ?M* **by** (*simp add*: *dist-midpoint*)
      show *dist C* (*midpoint A B*) = *dist C* (*midpoint A B*) **..**
    qed
    from *this* **show** *?thesis* **by** (*simp add*: *congruent-triangle.angles*(*6*))
  qed
  ultimately have *angle A ?M C* = *pi* / *2* **by** (*simp add*: *angle-commute*)
  from *this* **show** *?thesis*
    by (*simp add*: *orthogonal-iff-angle orthogonal-commute*)
next
  assume $A = B$
  from *this* **show** *?thesis*
    by (*simp add*: *orthogonal-clauses*(*1*))
qed

2.2 Properties of Chord Segments

lemma *chord-property*:
  fixes *S C* :: *'a* :: *euclidean-space*
  assumes *dist C S* = *dist C T*
  assumes *between* (*S*, *T*) *X*
  shows *dist S X* ∗ *dist X T* = (*dist C S*) ˆ *2* − (*dist C X*) ˆ *2*
proof −
  define *M* **where** *M* = *midpoint S T*
  have *between* (*S*, *T*) *M*
    unfolding *M-def* **by** (*simp add*: *between-midpoint*(*1*))
  have *dist T M* = *dist S M*
    unfolding *M-def* **by** (*simp add*: *dist-midpoint*)

  have *distances*: *max* (*dist S X*) (*dist X T*) = (*dist S M*) + (*dist X M*) ∧
    min (*dist S X*) (*dist X T*) = (*dist S M*) − (*dist X M*)
  proof *cases*
    assume *dist S X* ≤ *dist X T*
    from *this* **have** *between* (*X*, *T*) *M*
      using ‹*between* (*S*, *T*) *X*› *M-def*
      by (*simp add*: *dist-geq-iff-midpoint-in-between dist-commute*)
    from *this* **have** *between* (*S*, *M*) *X*
      using ‹*between* (*S*, *T*) *X*› ‹*between* (*S*, *T*) *M*› *between-swap* **by** *blast*
    from ‹*between* (*X*, *T*) *M*› **have** *dist X T* = *dist X M* + *dist M T*

using *between* **by** *auto*
moreover from ‹*between* (S, M) X› **have** *dist* S X = *dist* S M − *dist* M X
using *between dist-commute* **by** *force*
ultimately show *?thesis*
using ‹*dist* S $X \leq$ *dist* X T› ‹*dist* T M = *dist* S M›
by (*simp add*: *add.commute dist-commute max-def min-def*)
next
assume ¬ (*dist* S $X \leq$ *dist* X T)
from *this* **have** *dist* T $X \leq$ *dist* S X **by** (*simp add*: *dist-commute*)
from *this* **have** *between* (S, X) M
using ‹*between* (S, T) X› *M-def*
by (*simp add*: *dist-geq-iff-midpoint-in-between midpoint-sym between-commute*)
from *this* **have** *between* (T, M) X
using ‹*between* (S, T) X› ‹*between* (S, T) M› *between-swap between-commute*
by *metis*
from ‹*between* (S, X) M› **have** *dist* S X = *dist* S M + *dist* M X
using *between* **by** *auto*
moreover from ‹*between* (T, M) X› **have** *dist* T X = *dist* T M − *dist* M X
using *between dist-commute* **by** *force*
ultimately show *?thesis*
using ‹¬ *dist* S $X \leq$ *dist* X T› ‹*dist* T M = *dist* S M›
by (*metis dist-commute max-def min-def*)
qed

have *orthogonal* $(C - M)$ $(S - M)$
using ‹*dist* C S = *dist* C T› *M-def*
by (*auto simp add*: *isosceles-triangle-orthogonal-on-midpoint*)
have *orthogonal* $(C - M)$ $(X - M)$
proof −
have *between* (S, T) M
using *M-def between-midpoint*(*1*) **by** *blast*
obtain c **where** $(X - M) = c *_R (S - M)$
proof (*cases* $S = M$)
assume $S \neq M$
from ‹*between* (S, T) X› ‹*between* (S, T) M› *this* **obtain** c **where** $(X - M) = c *_R (S - M)$
using *between-implies-scaled-diff* **by** *blast*
from *this that* **show** *thesis* **by** *blast*
next
assume $S = M$
from *this* ‹*between* (S, T) X› **have** $X = M$
unfolding *M-def* **by** (*metis between-same midpoint-eq-endpoint*(*1*))
from ‹$X = M$› ‹$S = M$› **have** $(X - M) = 0 *_R (S - M)$ **by** *simp*
from *this that* **show** *thesis* **by** *blast*
qed
from *this* ‹*orthogonal* $(C - M)$ $(S - M)$› **show** *?thesis*
by (*auto intro*: *orthogonal-clauses*(*2*))
qed
from ‹*orthogonal* $(C - M)$ $(S - M)$› ‹*orthogonal* $(C - M)$ $(X - M)$› **have**

(*dist S M*) ^ *2* + (*dist M C*) ^ *2* = (*dist C S*) ^ *2*
(*dist X M*) ^ *2* + (*dist M C*) ^ *2* = (*dist C X*) ^ *2*
by (*auto simp only*: *pythagoras*)
from *this* **have** *geometric-observation*:
(*dist S M*) ^ *2* = (*dist C S*) ^ *2* − (*dist M C*) ^ *2*
(*dist X M*) ^ *2* = (*dist C X*) ^ *2* − (*dist M C*) ^ *2*
by *auto*

have *dist S X* ∗ *dist X T* = *max* (*dist S X*) (*dist X T*) ∗ *min* (*dist S X*) (*dist X T*)
by (*auto split*: *split-max*)
also have . . . = ((*dist S M*) + (*dist X M*)) ∗ ((*dist S M*) − (*dist X M*))
using *distances* **by** *simp*
also have . . . = (*dist S M*) ^ *2* − (*dist X M*) ^ *2*
by (*simp add*: *field-simps power2-eq-square*)
also have . . . = ((*dist C S*) ^ *2* − (*dist M C*) ^ *2*) − ((*dist C X*) ^ *2* − (*dist M C*) ^ *2*)
using *geometric-observation* **by** *simp*
also have . . . = (*dist C S*) ^ *2* − (*dist C X*) ^ *2* **by** *simp*
finally show *?thesis* **.**
qed

theorem *product-of-chord-segments*:
fixes S_1 T_1 S_2 T_2 *X C* :: *'a* :: *euclidean-space*
assumes *between* (S_1, T_1) *X between* (S_2, T_2) *X*
assumes *dist C* S_1 = *r dist C* T_1 = *r*
assumes *dist C* S_2 = *r dist C* T_2 = *r*
shows *dist* S_1 *X* ∗ *dist X* T_1 = *dist* S_2 *X* ∗ *dist X* T_2
proof −
from ‹*dist C* S_1 = *r*› ‹*dist C* T_1 = *r*› ‹*between* (S_1, T_1) *X*›
have *dist* S_1 *X* ∗ *dist X* T_1 = *r* ^ *2* − (*dist C X*) ^ *2*
by (*subst chord-property*) *auto*
also from ‹*dist C* S_2 = *r*› ‹*dist C* T_2 = *r*› ‹*between* (S_2, T_2) *X*›
have . . . = *dist* S_2 *X* ∗ *dist X* T_2
by (*subst chord-property*) *auto*
finally show *?thesis* **.**
qed

end

References

[1] G. D. Birkhoff and R. Beatley. *Basic Geometry*, page 150. AMS Chelsea Publishing, Providence, Rhode Island, third edition, 1959.

[2] M. Eberl. Basic geometric properties of triangles. *Archive of Formal Proofs*, Dec. 2015. http://isa-afp.org/entries/Triangle.shtml, Formal proof development.

[3] J. Harrison. Theorem on product of segments of chords (in HOL light), 2010. https://github.com/jrh13/hol-light/blob/master/100/chords.ml.

[4] W. R. Knorr. *Textual Studies in Ancient and Medieval Geometry*, pages 34–36. Birkhäuser, Boston, first edition, 1989.

[5] Proofwiki. Intersecting chord theorem, 2014. https://proofwiki.org/wiki/Intersecting_Chord_Theorem [Online; accessed = 10-October-2016].