Towards intuitive reasoning in axiomatic geometry

Maximilian Doré[1], Krysia Broda[2]

[1]LMU Munich, [2]Imperial College London

Theorem Prover Components for Educational Software
FLoC 2018, Oxford

July 18, 2018

Goal of this work

- Bring formalized mathematics closer to intuitive reasoning
- Make extensive use of ATPs to decrease level of detail in formal proofs

$\longrightarrow$ Harness interactive theorem provers not only for expert users

Outline

Outline

An exemplary proof

Include functions.
Let A,B,C be set.
Let f: $A \to B$.
Let g: $B \to C$.

Lemma: $g \circ f$ is injective implies f is injective.
Proof:
    Assume $g \circ f$ is injective.
    Assume $(f\{x\}) = (f\{x'\})$ and $x \in A$ and $x' \in A$.
    Then $((g \circ f)\{x\}) = ((g \circ f)\{x'\})$.
    Hence $x = x'$.
    Hence f is injective.
qed.

An exemplary proof

Include functions.
Let A,B,C be set.
Let f: A $\rightarrow$ B.
Let g: B $\rightarrow$ C.

Lemma: g$\circ$f is injective implies f is injective.
Proof:
    Assume g$\circ$f is injective.
    Assume (f{x}) = (f{x'}) and x $\in$ A and x' $\in$ A.
    Note ((g$\circ$f){x}) = ((g$\circ$f){x'}):
    ...
    qed.
    Hence x = x'.
    Hence f is injective.
qed.

First-order logic as an internal language

...

Lemma: $\forall set(A), set(B), set(C), function(f, A, B), function(g, B, C).$
$injective(composition(g, f)) \rightarrow injective(f).$

Proof:

Assume $injective(composition(g, f))$.

Assume $funApp(f, x) = funApp(f, x') \wedge in(x, A) \wedge in(x', A)$.

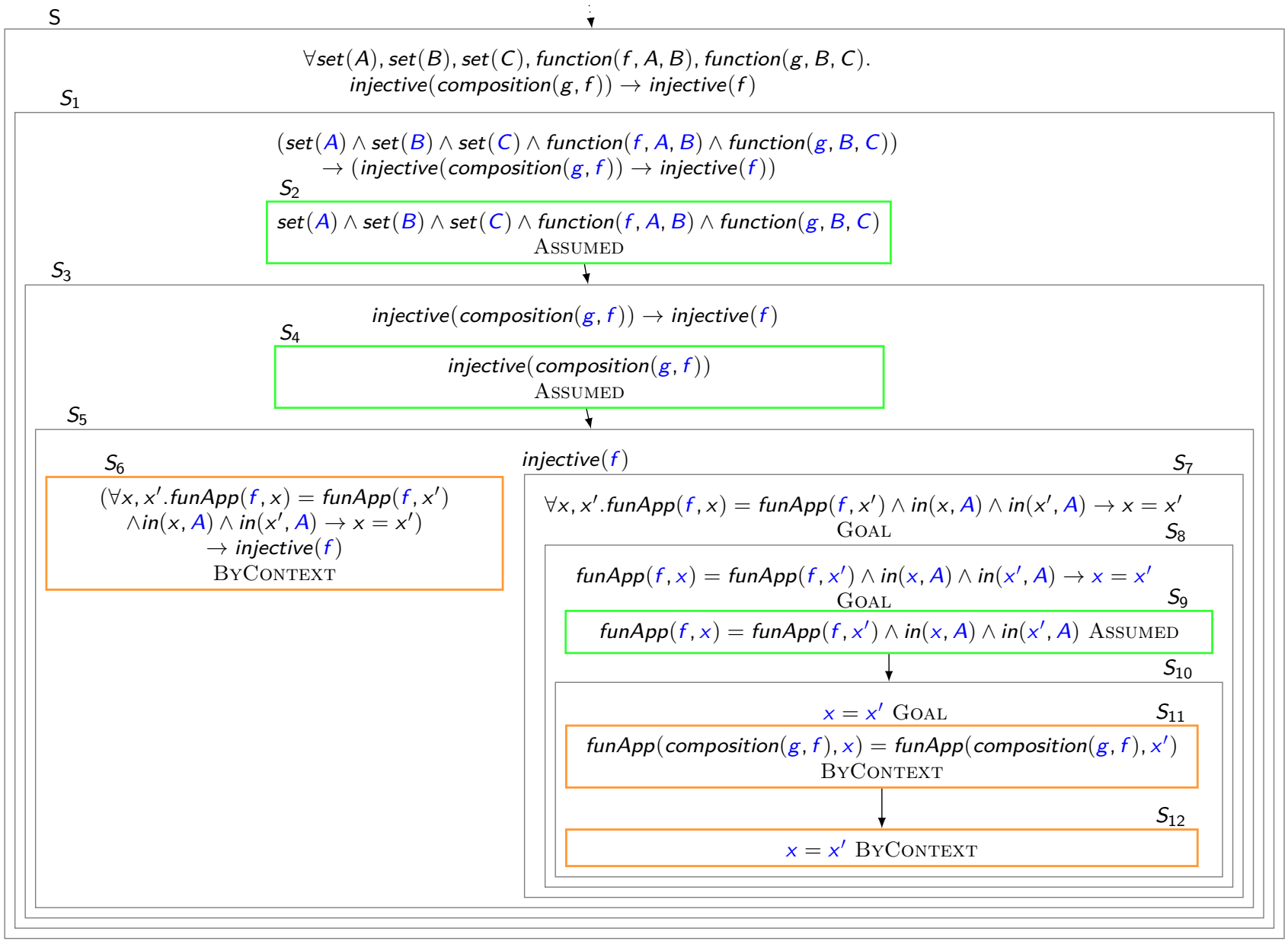
Then $funApp(composition(g, f), x) = funApp(composition(g, f), x')$.

Hence $x = x'$.

Hence $injective(f)$.

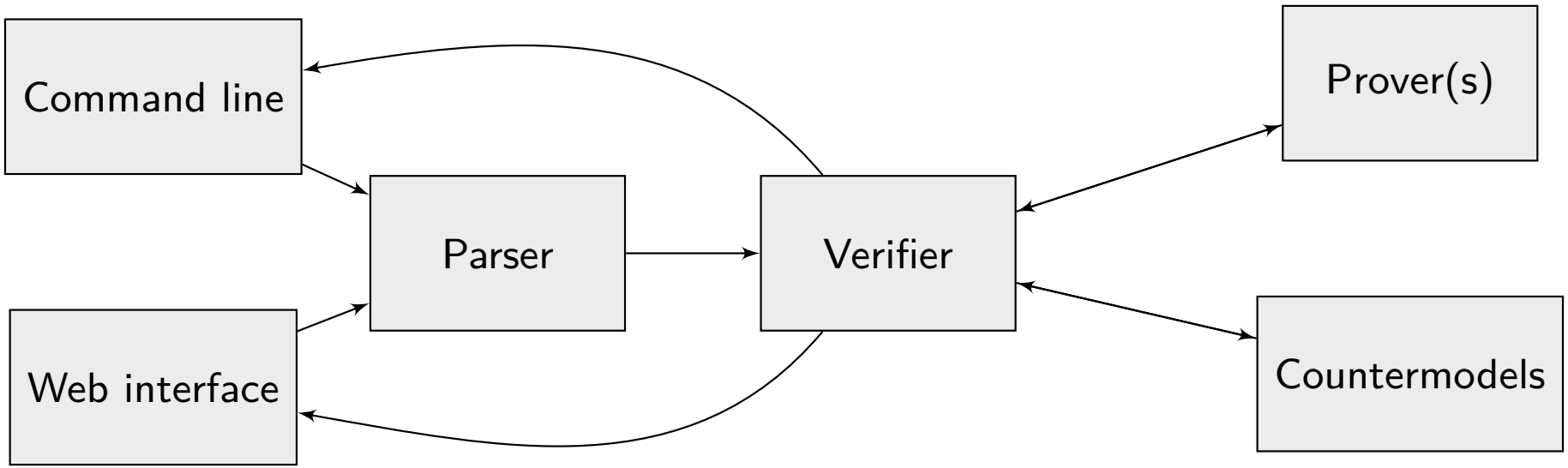
qed.

Capturing the structure of a proof

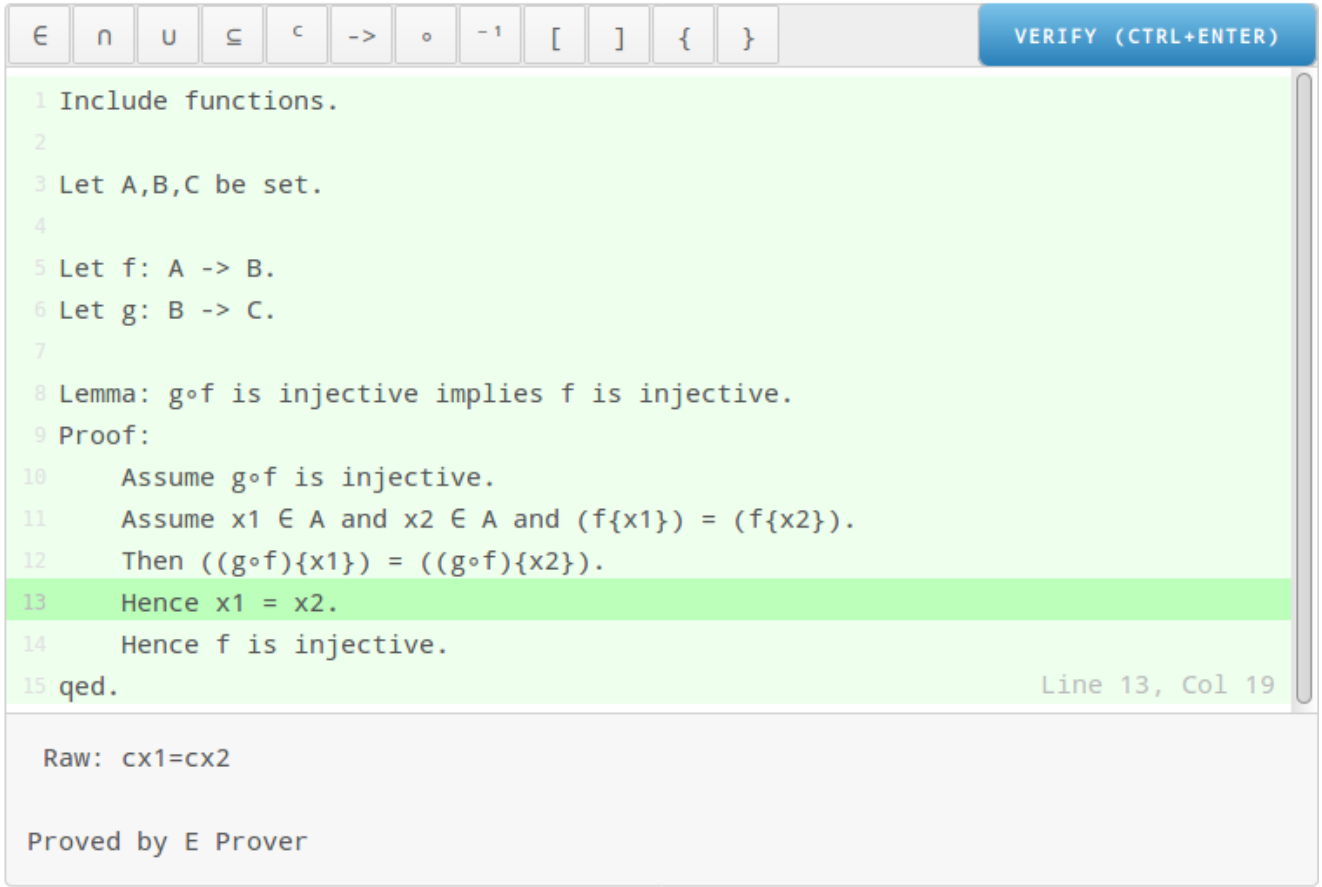


Architecture of Elfe

- Written in Haskell
- Interfaced provers: E Prover, Vampire, Spass and Beagle



Web interface



Available online: `https://elfe-prover.org`

Summary

- ELFE provides a fairly natural input language and can be accessed through an intuitive web interface
- Precursor: SYSTEM FOR AUTOMATED DEDUCTION
- Inner workings:
  - Input language is converted into sequence of first-order formulas
  - This data structure implies proof obligations which are checked by ATP
- Background libraries initially developed for sets, relations and functions. Now extended to geometry!

Outline

Tarskis's axiomatization of geometry in ELFE

Notation between: a-b-c.

Notation equidistant: a-b ≡ c-d.

Axiom CongrRefl: for all a,b. a-b ≡ b-a.

Axiom CongrIdent: for all a,b,c. a-b ≡ c-c implies a = b.

Axiom CongrTrans: for all a,b,p,q,r,s. a-b ≡ p-q and a-b ≡ r-s implies p-q ≡ r-s.

Axiom SegmentConstr: for all a,b,c,d. exists e. b-e ≡ c-d and a-b-e.

Axiom FiveSegment: for all a,b,c,d,a',b',c',d'. (a-b-c and a'-b'-c' and a-b ≡ a'-b' and b-c ≡ b'-c' and a-d ≡ a'-d' and b-d ≡ b'-d' and not a = b) implies c-d ≡ c'-d'.

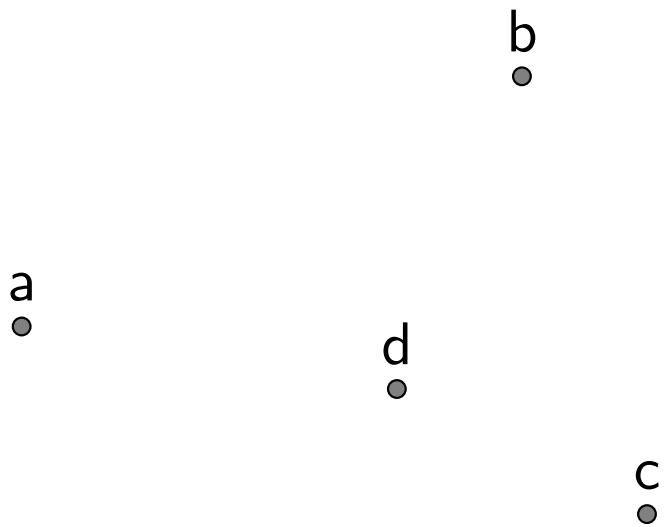
Axiom BetwIdent: for all a,b. a-b-a implies a = b.

Axiom Pasch: for all a,b,c,p,q. a-p-c and b-q-c implies exists x. p-x-b and q-x-a.

Axiom LowerDim: exists a,b,c. not a-b-c and not b-c-a and not c-a-b.

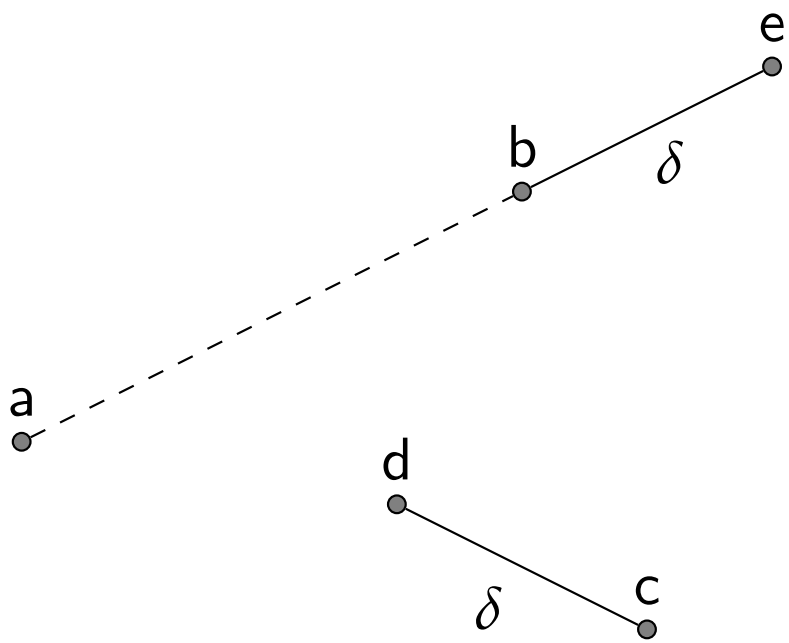
Axiom Euclid: for all a,b,c,d,t. exists x,y.
(a-d-t and b-d-c and not a = d) implies (a-b-x and a-c-y and x-t-y).

Segment construction



Axiom SegmentConstr: for all a,b,c,d. exists e. b-e $\equiv$ c-d and a-b-e.

Segment construction



Axiom SegmentConstr: for all a,b,c,d. exists e. b-e ≡ c-d and a-b-e.

Building up geometry

Definition DefCol: for all a,b,c. col(a,b,c) iff a-b-c or b-c-a or c-a-b.

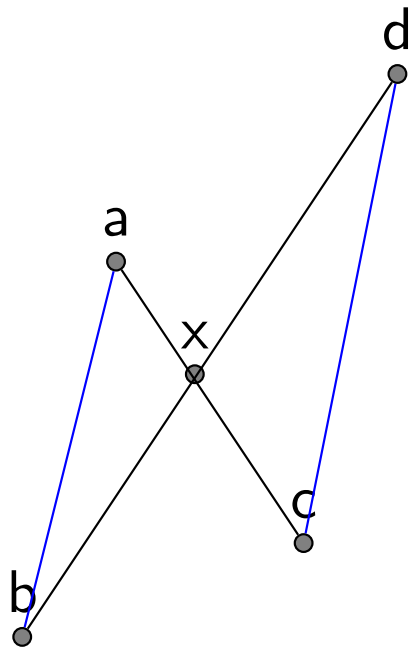
Definition DefMidpoint: for all a,b,m.
midpoint(m,a,b) iff a-m-b and a-m $\equiv$ m-b.

Definition DefCoplanar: for all a,b,c,d. coplanar(a,b,c,d) iff exists x.
(col(a,b,x) and col(c,d,x)) or
(col(a,c,x) and col(b,d,x)) or
(col(a,d,x) and col(b,c,x)).

Notation parstr: a-b|–|c-d.
Definition DefParallelStrict: for all a,b,c,d. a-b|–|c-d iff
((not a = b) and (not c = d) and coplanar(a,b,c,d) and
not exists x. col(x,a,b) and col(x,c,d)).

Notation parallel: a-b||c-d.
Definition DefParallel: for all a,b,c,d. a-b||c-d iff a-b|–|c-d or
((not a = b) and (not c = d) and col(a,c,d) and col(b,c,d)).

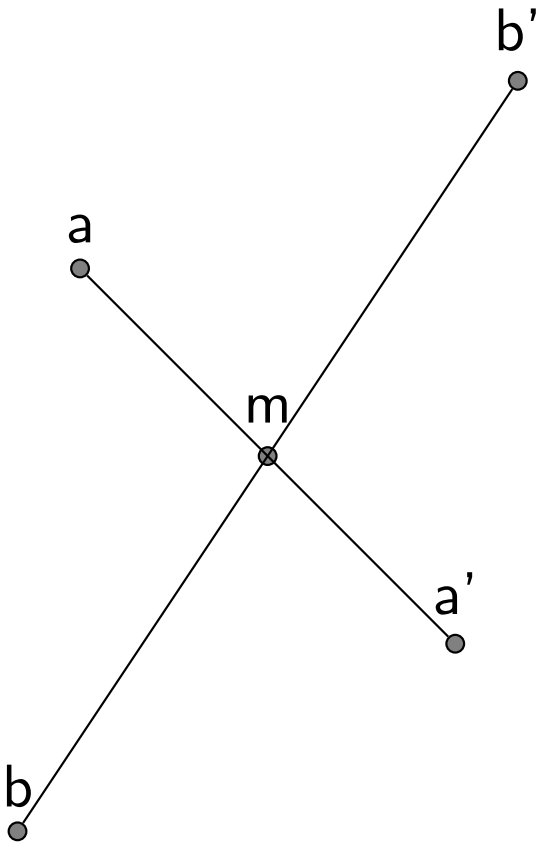
Parallelism



Definition DefParallelStrict: for all a,b,c,d. a-b|–|c-d iff
((not a = b) and (not c = d) and coplanar(a,b,c,d) and
not exists x. col(x,a,b) and col(x,c,d)).

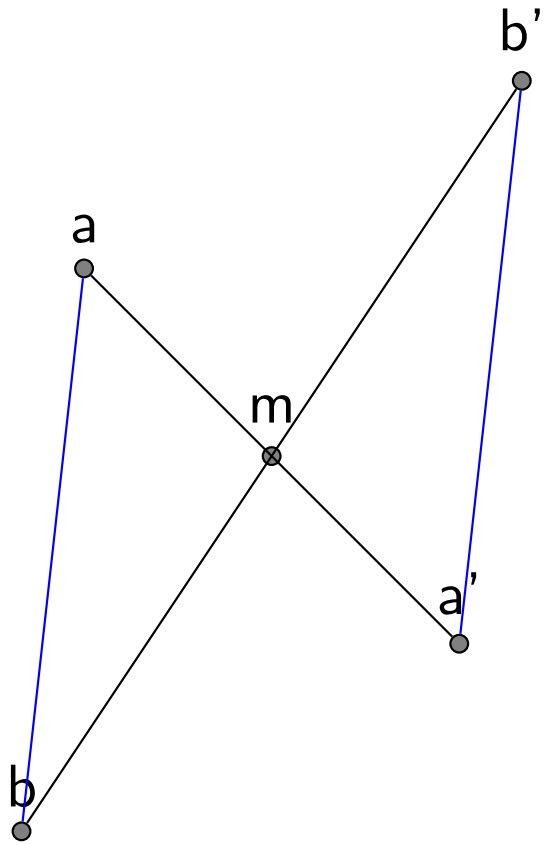
Definition DefParallel: for all a,b,c,d. a-b||c-d iff a-b|–|c-d or
((not a = b) and (not c = d) and col(a,c,d) and col(b,c,d)).

Common midpoint implies parallelity



Lemma: for all a,b,a',b',m.
a $\neq$ b and midpoint(m,a,a') and midpoint(m,b,b') implies a-b||a'-b'.

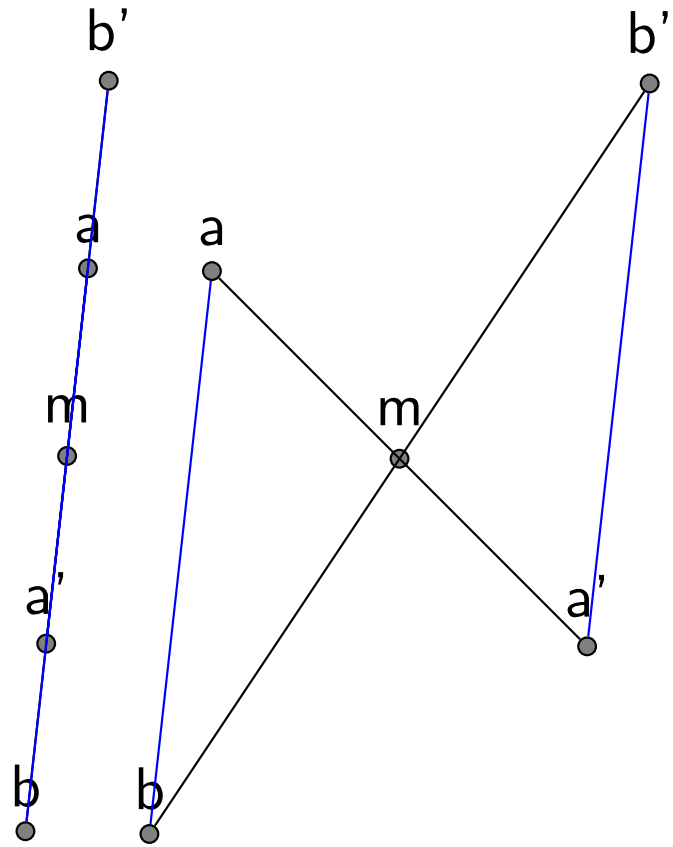
Common midpoint implies parallelity



Lemma: for all a,b,a',b',m.
a $\neq$ b and midpoint(m,a,a') and midpoint(m,b,b') implies a-b||a'-b'.

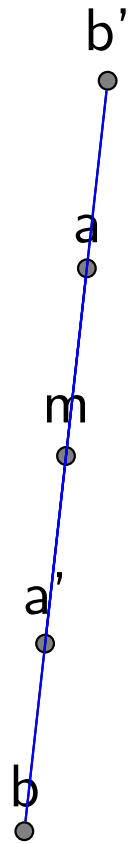
Case distinction

Lemma: for all a,b,a',b',m. a $\neq$ b and midpoint(m,a,a') and midpoint(m,b,b') implies a-b||a'-b'.
Proof:
    Assume a $\neq$ b and midpoint(m,a,a')
        and midpoint(m,b,b').
    Case col(a,b,b'):
        ...
        ...
    qed.
    Case not col(a,b,b'):
        ...
    qed.
    Hence a-b||a'-b'.
qed.



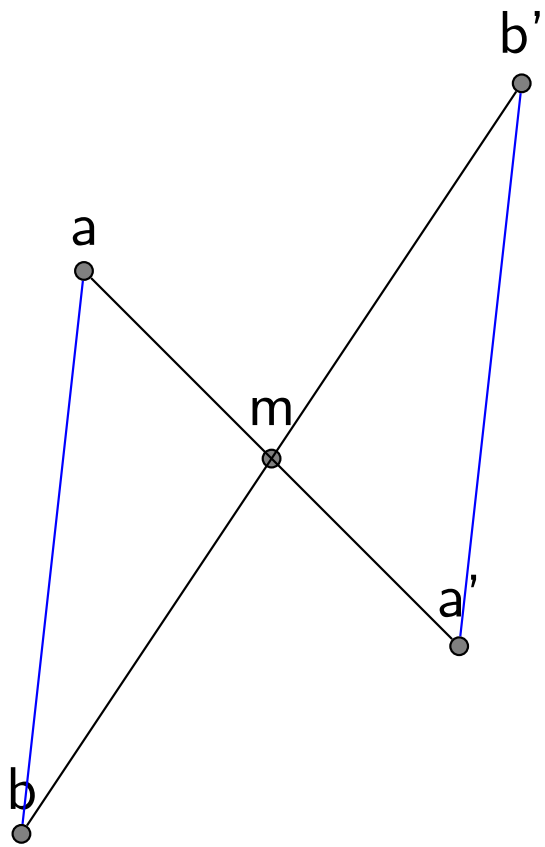
Degenerate case

Lemma: for all a,b,a',b',m. a $\neq$ b and midpoint(m,a,a') and midpoint(m,b,b') implies a-b||a'-b'.
Proof:
    Assume a $\neq$ b and midpoint(m,a,a')
        and midpoint(m,b,b').
    Case col(a,b,b'):
    Then a' $\neq$ b' and col(a,a',b') and col(b,a',b').
    Then a-b||a'-b' by DefParallel.
    qed.
    Case not col(a,b,b'):
        ...
    qed.
    Hence a-b||a'-b'.
qed.



Strict parallelism

```
Case not col(a,b,b'):
    Note a' ≠ b':
        ...
    qed.
    Note coplanar(a,b,a',b'):
        ...
    qed.
    Note not exists x. col(x,a,b) and col(x,a',b'):
        ...
    qed.
    Then (a-b|–|a'-b') by DefParallelStrict.
    Then (a-b||a'-b') by DefParallel.
qed.
```



Proof by contradiction

Note not exists x. col(x,a,b) and col(x,a',b'):

Assume exists x. col(x,a,b) and col(x,a',b').

Take x such that col(x,a,b) and col(x,a',b')

by SegmentConstr, DefCol.

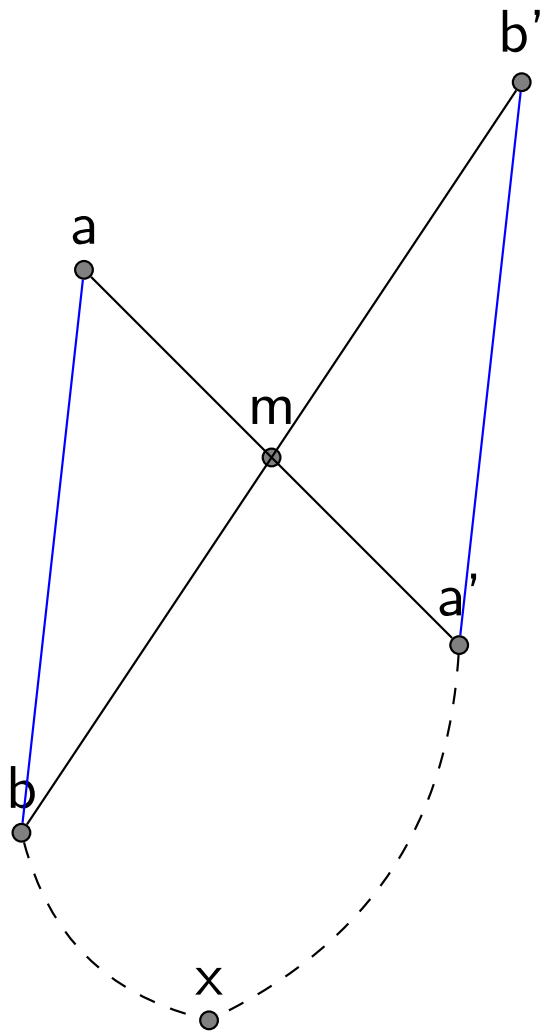
...

...

...

Hence contradiction.

qed.



Constructing points

Note not exists x. col(x,a,b) and col(x,a',b'):

Assume exists x. col(x,a,b) and col(x,a',b').

Take x such that col(x,a,b) and col(x,a',b')

by SegmentConstr, DefCol.

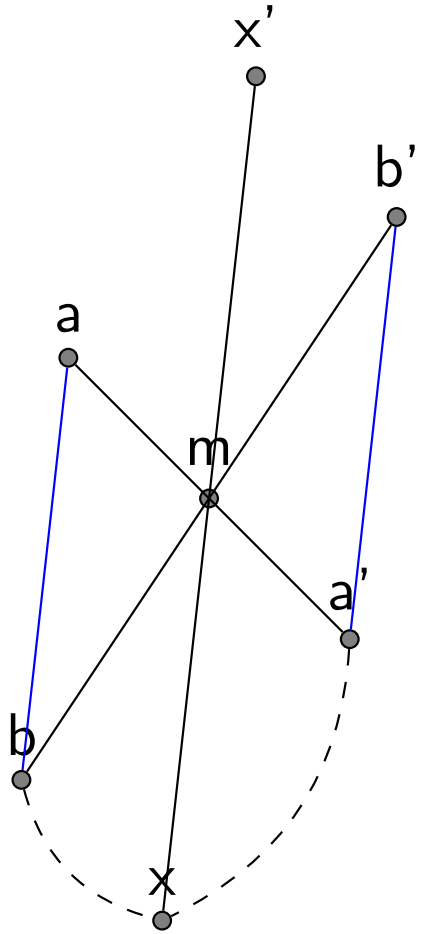
Take x' such that x-m-x' and (m-x' ≡ m-x)

by SegmentConstr.

...

Hence contradiction.

qed.



Deriving a contradiction

Note not exists x. col(x,a,b) and col(x,a',b'):

Assume exists x. col(x,a,b) and col(x,a',b').

Take x such that col(x,a,b) and col(x,a',b')

by SegmentConstr, DefCol.

Take x' such that x-m-x' and (m-x' $\equiv$ m-x)

by SegmentConstr.

Then col(a,b,x') and col(a',b',x').

Then col(b',x,x') since col(b',a',x) and col(b',a',x').

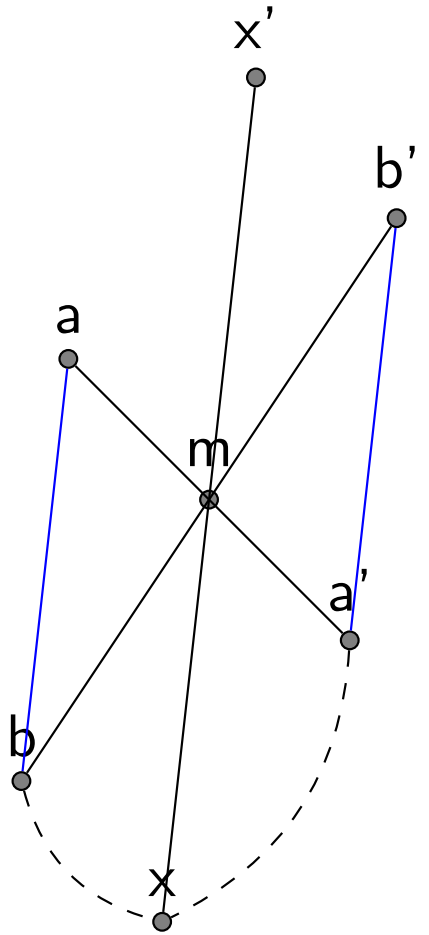
Then col(b,x,x') since col(b,a,x) and col(b,a,x').

Then col(b,x',b') since col(b,x,x') and col(b',x,x').

Then col(a,b,b') since col(b,x,b') and col(b,x,a).

Hence contradiction.

qed.



Deriving a contradiction

Note not exists x. col(x,a,b) and col(x,a',b'):

Assume exists x. col(x,a,b) and col(x,a',b').

Take x such that col(x,a,b) and col(x,a',b')

by SegmentConstr, DefCol.

Take x' such that x-m-x' and (m-x' $\equiv$ m-x)

by SegmentConstr.

Then col(a,b,x') and col(a',b',x').

Then col(b',x,x') since col(b',a',x) and col(b',a',x').

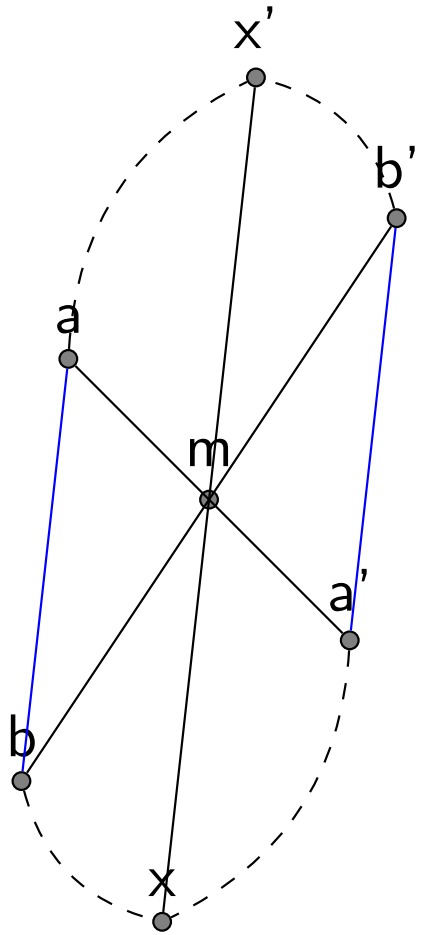
Then col(b,x,x') since col(b,a,x) and col(b,a,x').

Then col(b,x',b') since col(b,x,x') and col(b',x,x').

Then col(a,b,b') since col(b,x,b') and col(b,x,a).

Hence contradiction.

qed.



Deriving a contradiction

Note not exists x. col(x,a,b) and col(x,a',b'):

Assume exists x. col(x,a,b) and col(x,a',b').

Take x such that col(x,a,b) and col(x,a',b')

by SegmentConstr, DefCol.

Take x' such that x-m-x' and (m-x' $\equiv$ m-x)

by SegmentConstr.

Then col(a,b,x') and col(a',b',x').

Then col(b',x,x') since col(b',a',x) and col(b',a',x').

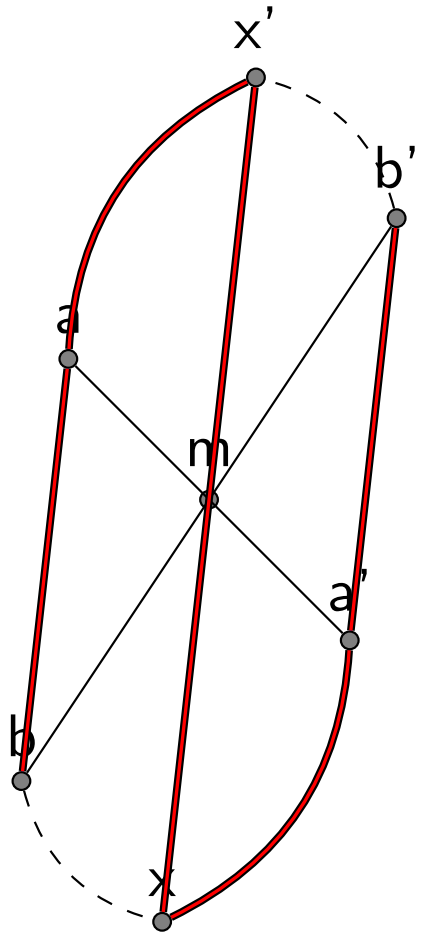
Then col(b,x,x') since col(b,a,x) and col(b,a,x').

Then col(b,x',b') since col(b,x,x') and col(b',x,x').

Then col(a,b,b') since col(b,x,b') and col(b,x,a).

Hence contradiction.

qed.



Deriving a contradiction

Note not exists x. col(x,a,b) and col(x,a',b'):

Assume exists x. col(x,a,b) and col(x,a',b').

Take x such that col(x,a,b) and col(x,a',b')

by SegmentConstr, DefCol.

Take x' such that x-m-x' and (m-x' $\equiv$ m-x)

by SegmentConstr.

Then col(a,b,x') and col(a',b',x').

Then col(b',x,x') since col(b',a',x) and col(b',a',x').

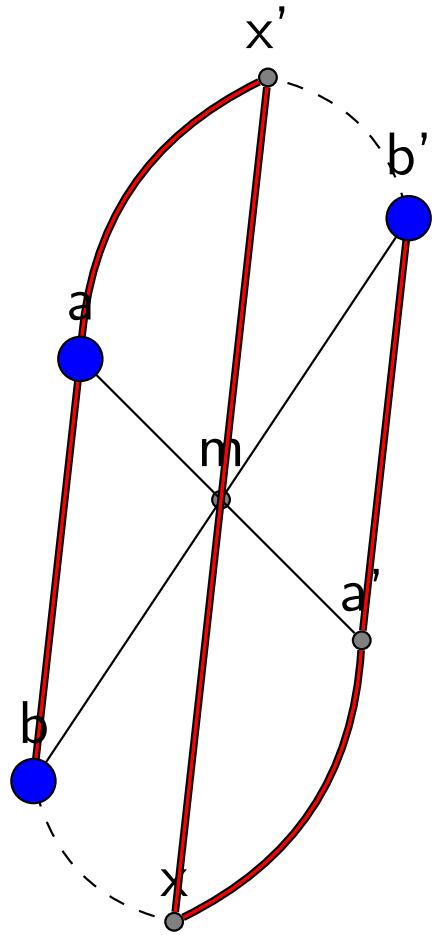
Then col(b,x,x') since col(b,a,x) and col(b,a,x').

Then col(b,x',b') since col(b,x,x') and col(b',x,x').

Then col(a,b,b') since col(b,x,b') and col(b,x,a).

Hence contradiction.

qed.



The complete proof

Lemma: for all a,b,a',b',m. (not a = b) and midpoint(m,a,a') and midpoint(m,b,b') implies a-b||a'-b'.
Proof:
Assume (not a = b) and midpoint(m,a,a') and midpoint(m,b,b').
Case col(a,b,b'):
Then (not a' = b') and col(a,a',b') and col(b,a',b') by midmidcol. Then (a-b||a'-b') by DefParallel.
qed.
Case not col(a,b,b'):
Note not a' = b':
Assume a' = b'. Then a'-b'-m and m-a' ≡ m-b'.
Then m-a-b and m-a ≡ m-b. Then (a = b) by BetweenCong. Hence contradiction.
qed.
Note coplanar(a,b,a',b'):
Then a-m-a' and (b-m-b') by DefMidpoint. Then col(a,a',m) and col(b,b',m) by ColPerm, DefCol.
Then coplanar(a,b,a',b') by DefCoplanar.
qed.
Note not exists x. col(x,a,b) and col(x,a',b'):
Assume exists x. col(x,a,b) and col(x,a',b').
Take x such that col(x,a,b) and col(x,a',b') by SegmentConstr, DefCol.
Take x' such that x-m-x' and (m-x' ≡ m-x) by SegmentConstr.
Then col(a,b,x') and col(a',b',x).
Then col(b',x,x') since col(b',a',x) and col(b',a',x'). Then col(b,x,x') since col(b,a,x) and col(b,a,x').
Then col(b,x,b') since col(b,x,x') and col(b',x,x'). Then col(a,b,b') since col(b,x,b') and col(b,x,a).
Hence contradiction.
qed.
Then (a-b|–|a'-b') by DefParallelStrict.
Then (a-b||a'-b') by DefParallel.
qed.
Hence a-b||a'-b'.
qed.

Outline

Power of background provers

```
Case not col(a,b,b'):
    Note not exists x. col(x,a,b) and col(x,a',b'):
        Assume exists x. col(x,a,b) and col(x,a',b').
        ...
        Then col(a,b,b') since col(b,x,b') and col(b,x,a).
        Then England is winnerOfWorldcup2018.
        Hence contradiction.
    qed.
    ...
qed.
```

- Power of background provers varies from case to case

Using ELFE in education

elfe Theorem Proving

Exercise 2

You will now complete your own first proof about subrelations. The lemma proves that if R is a subrelation of S and S is a subrelation of T, then R must be a subrelation of T as well. Try to complete the proof by replacing "{INSERT-SOLUTION-HERE}"!

∈ ∩ ∪ ⊆ ᶜ -> ∘ ⁻¹ [ ] { VERIFY (CTRL+ENTER)

```
1  Include relations.
2
3  Let R,S,T be relation.
4
5  Lemma: R ⊆ S and S ⊆ T implies R ⊆ T.
6  Proof:
7      Assume R ⊆ S and S ⊆ T.
8      Assume R[x,y].
9      Then {INSERT-SOLUTION-HERE}.
10     Hence T[x,y].
11     Hence R ⊆ T.
12 qed.                                    Line 0, Col 0
```

GO TO NEXT EXERCISE

- System tested with 80 students of the Universidad El Bosque in Colombia
- Students learn syntax of ELFE while practicing mathematical proofs

Discussion and Outlook

- ATP for higher-order logic in active development
  $\longrightarrow$ Use higher-order logic internally in Elfe
- Utilize machines to verify handwavy steps in formal proofs
- Synthetic mathematics gains traction, e.g., development of Homotopy Type Theory

$\longrightarrow$ Use interactive theorem provers early in mathematical education

Axioms

Axiom ColPerm: for all a,b,c. col(a,b,c) implies col(a,c,b) and col(b,a,c) and col(c,a,b) and col(b,c,a) and col(c,b,a).

Axiom midmidcol: for all a,b,a',b',m. (not a = b) and midpoint(m,a,a') and midpoint(m,b,b') and col(a,b,b') implies (not a' = b') and col(a,a',b') and col(b,a',b').

Axiom midpreservescol: for all a,b,c,m,a',b',c'. col(a,b,c) and midpoint(m,a,a') and midpoint(m,b,b') and midpoint(m,c,c') implies col(a',b',c').

Axiom ColTrans: for all a,b,c,d. (not a = b) and col(a,b,c) and col(a,b,d) implies col(a,c,d).

Axiom BetweenCong: for all a,b,c. a-b-c and a-b $\equiv$ a-c implies b = c.