Proving the Impossibility of Trisecting an Angle and Doubling the Cube

Ralph Romanos and Lawrence Paulson

March 17, 2025

Abstract

Squaring the circle, doubling the cube and trisecting an angle, using a compass and straightedge alone, are classic unsolved problems first posed by the ancient Greeks. All three problems were proved to be impossible in the 19th century. The following document presents the proof of the impossibility of solving the latter two problems using Isabelle/HOL, following a proof by Carrega [Car81]. The proof uses elementary methods: no Galois theory or field extensions. The set of points constructible using a compass and straightedge is defined inductively. Radical expressions, which involve only square roots and arithmetic of rational numbers, are defined, and we find that all constructive points have radical coordinates. Finally, doubling the cube and trisecting certain angles requires solving certain cubic equations that can be proved to have no rational roots. The Isabelle proofs require a great many detailed calculations.

Contents

1 Proving the impossibility of trisecting an angle and doubling the cube

theory *Impossible-Geometry*
imports *Complex-Main*
begin

2 Formal Proof

2.1 Definition of the set of Points

datatype *point* = *Point real real*

definition *points-def*:
  $points = \{M.\ \exists\ x \in \mathbb{R}.\ \exists\ y \in \mathbb{R}.\ (M = Point\ x\ y)\}$

primrec *abscissa* :: *point* => *real*
  where *abscissa*: *abscissa* (*Point x y*) = *x*

primrec *ordinate* :: *point* => *real*
  where *ordinate*: *ordinate* (*Point x y*) = *y*

lemma *point-surj* [*simp*]:
  Point (*abscissa M*) (*ordinate M*) = *M*
  by (*induct M*) *simp*

lemma *point-eqI* [*intro?*]:
  $[\![abscissa\ M = abscissa\ N;\ ordinate\ M = ordinate\ N]\!] \Longrightarrow M = N$
  by (*induct M*, *induct N*) *simp*

lemma *point-eq-iff*:
  $M = N \longleftrightarrow abscissa\ M = abscissa\ N \wedge ordinate\ M = ordinate\ N$
  by (*induct M*, *induct N*) *simp*

2.2 Subtraction

Datatype point has a structure of abelian group

instantiation *point* :: *ab-group-add*
begin

definition *point-zero-def*:

$0 = Point\ 0\ 0$

definition *point-one-def*:
$point\text{-}one = Point\ 1\ 0$

definition *point-add-def*:
$A + B = Point\ (abscissa\ A + abscissa\ B)\ (ordinate\ A + ordinate\ B)$

definition *point-minus-def*:
$-\ A = Point\ (-\ abscissa\ A)\ (-\ ordinate\ A)$

definition *point-diff-def*:
$A - (B{::}point) = A + -\ B$

lemma *Point-eq-0* [*simp*]:
$Point\ xA\ yA = 0 \longleftrightarrow (xA = 0 \land yA = 0)$
by (*simp add*: *point-zero-def*)

lemma *point-abscissa-zero* [*simp*]:
$abscissa\ 0 = 0$
by (*simp add*: *point-zero-def*)

lemma *point-ordinate-zero* [*simp*]:
$ordinate\ 0 = 0$
by (*simp add*: *point-zero-def*)

lemma *point-add* [*simp*]:
$Point\ xA\ yA + Point\ xB\ yB = Point\ (xA + xB)\ (yA + yB)$
by (*simp add*: *point-add-def*)

lemma *point-abscissa-add* [*simp*]:
$abscissa\ (A + B) = abscissa\ A + abscissa\ B$
by (*simp add*: *point-add-def*)

lemma *point-ordinate-add* [*simp*]:
$ordinate\ (A + B) = ordinate\ A + ordinate\ B$
by (*simp add*: *point-add-def*)

lemma *point-minus* [*simp*]:
$-\ (Point\ xA\ yA) = Point\ (-\ xA)\ (-\ yA)$
by (*simp add*: *point-minus-def*)

lemma *point-abscissa-minus* [*simp*]:
$abscissa\ (-\ A) = -\ abscissa\ (A)$
by (*simp add*: *point-minus-def*)

lemma *point-ordinate-minus* [*simp*]:
$ordinate\ (-\ A) = -\ ordinate\ (A)$
by (*simp add*: *point-minus-def*)

lemma *point-diff* [*simp*]:
  Point xA yA $-$ *Point xB yB* $=$ *Point* (*xA* $-$ *xB*) (*yA* $-$ *yB*)
  by (*simp add*: *point-diff-def*)

lemma *point-abscissa-diff* [*simp*]:
  abscissa (*A* $-$ *B*) $=$ *abscissa* (*A*) $-$ *abscissa* (*B*)
  by (*simp add*: *point-diff-def*)

lemma *point-ordinate-diff* [*simp*]:
  ordinate (*A* $-$ *B*) $=$ *ordinate* (*A*) $-$ *ordinate* (*B*)
  by (*simp add*: *point-diff-def*)

instance
  by *intro-classes* (*simp-all add*: *point-add-def point-diff-def*)

end

2.3 Metric Space

We can also define a distance, hence point is also a metric space

instantiation *point* :: *metric-space*
begin

definition *point-dist-def*:
  dist A B $=$ *sqrt* ((*abscissa* (*A* $-$ *B*))^*2* $+$ (*ordinate* (*A* $-$ *B*))^*2*)

definition
  (*uniformity* :: (*point* $\times$ *point*) *filter*) $=$ (*INF e*$\in${*0* <..}. *principal* {(*x*, *y*). *dist x y* $<$ *e*})

definition
  open (*S* :: *point set*) $=$ ($\forall$ *x*$\in$*S*. $\forall_F$ (*x'*, *y*) *in uniformity*. *x'* $=$ *x* $\longrightarrow$ *y* $\in$ *S*)

lemma *point-dist* [*simp*]:
  dist (*Point xA yA*) (*Point xB yB*) $=$ *sqrt* ((*xA* $-$ *xB*)^*2* $+$ (*yA* $-$ *yB*)^*2*)
  unfolding *point-dist-def*
  by *simp*

lemma *real-sqrt-diff-squares-triangle-ineq*:
  fixes *a b c d* :: *real*
  shows *sqrt* ((*a* $-$ *c*)^*2* $+$ (*b* $-$ *d*)^*2*) $\leq$ *sqrt* (*a*^*2* $+$ *b*^*2*) $+$ *sqrt* (*c*^*2* $+$ *d*^*2*)
proof $-$
  have *sqrt* ((*a* $-$ *c*)^*2* $+$ (*b* $-$ *d*)^*2*) $\leq$ *sqrt* (*a*^*2* $+$ *b*^*2*) $+$ *sqrt* (($-$*c*)^*2* $+$ ($-$*d*)^*2*)
    by (*metis diff-conv-add-uminus real-sqrt-sum-squares-triangle-ineq*)
  also have ... $=$ *sqrt* (*a*^*2* $+$ *b*^*2*) $+$ *sqrt* (*c*^*2* $+$ *d*^*2*)
    by *simp*
  finally show *?thesis* **.**

qed

instance
proof
  fix $A\ B\ C$ :: *point* **and** S :: *point set*
  show $(\mathit{dist}\ A\ B = \mathit{0}) = (A = B)$
    by (*induct* A, *induct* B) (*simp add*: *point-dist-def*)
  show $(\mathit{dist}\ A\ B) \leq (\mathit{dist}\ A\ C) + (\mathit{dist}\ B\ C)$
  proof $-$
    have $\mathit{sqrt}\ ((\mathit{abscissa}\ (A - B))\hat{}\mathit{2} + (\mathit{ordinate}\ (A - B))\hat{}\mathit{2}) \leq$
        $\mathit{sqrt}\ ((\mathit{abscissa}\ (A - C))\hat{}\mathit{2} + (\mathit{ordinate}\ (A - C))\hat{}\mathit{2}) +$
        $\mathit{sqrt}\ ((\mathit{abscissa}\ (B - C))\hat{}\mathit{2} + (\mathit{ordinate}\ (B - C))\hat{}\mathit{2})$
      using *real-sqrt-diff-squares-triangle-ineq*
        [*of* $\mathit{abscissa}\ (A) - \mathit{abscissa}\ (C)\ \mathit{abscissa}\ (B) - \mathit{abscissa}\ (C)$
          $\mathit{ordinate}\ (A) - \mathit{ordinate}\ (C)\ \mathit{ordinate}\ (B) - \mathit{ordinate}\ (C)$]
      by (*simp only*: *point-diff-def*) (*simp add*: *algebra-simps*)
    thus *?thesis*
      by (*simp add*: *point-dist-def*)
  qed
qed (*rule uniformity-point-def open-point-def*)+
end

2.4 Geometric Definitions

These geometric definitions will later be used to define constructible points

The distance between two points is defined with the distance of the metric space point

definition *distance-def*:
  $\mathit{distance}\ A\ B = \mathit{dist}\ A\ B$

parallel $A\ B\ C\ D$ is true if the lines AB and CD are parallel. If not it is false.

definition *parallel-def*:
  $\mathit{parallel}\ A\ B\ C\ D = ((\mathit{abscissa}\ A - \mathit{abscissa}\ B) * (\mathit{ordinate}\ C - \mathit{ordinate}\ D) = (\mathit{ordinate}\ A - \mathit{ordinate}\ B) * (\mathit{abscissa}\ C - \mathit{abscissa}\ D))$

Three points $A\ B\ C$ are collinear if and only if the lines AB and AC are parallel

definition *collinear-def*:
  $\mathit{collinear}\ A\ B\ C = \mathit{parallel}\ A\ B\ A\ C$

The point M is the intersection of two lines AB and CD if and only if the points A, M and B are collinear and the points C, M and D are also collinear

definition *is-intersection-def*:
  $\mathit{is\text{-}intersection}\ M\ A\ B\ C\ D = (\mathit{collinear}\ A\ M\ B \wedge \mathit{collinear}\ C\ M\ D)$

2.5 Reals definable with square roots

The inductive set *radical-sqrt* defines the reals that can be defined with square roots. If x is in the following set, then it depends only upon rational expressions and square roots. For example, suppose x is of the form : $x = (\sqrt{a+\sqrt{b}} + \sqrt{c+\sqrt{d*e+f}})/(\sqrt{a}+\sqrt{b}) + (a+\sqrt{b})/\sqrt{g}$, where a, b, c, d, e, f and g are rationals. Then x is in *radical-sqrt* because it is only defined with rationals and square roots of radicals.

inductive-set *radical-sqrt* :: *real set*
  where
    Rat: $x \in \mathbb{Q} \Longrightarrow x \in$ *radical-sqrt*
  | *Neg*: $x \in$ *radical-sqrt* $\Longrightarrow -x \in$ *radical-sqrt*
  | *Inverse*: $x \in$ *radical-sqrt* $\Longrightarrow x \neq 0 \Longrightarrow 1/x \in$ *radical-sqrt*
  | *Plus*: $x \in$ *radical-sqrt* $\Longrightarrow y \in$ *radical-sqrt* $\Longrightarrow x{+}y \in$ *radical-sqrt*
  | *Times*: $x \in$ *radical-sqrt* $\Longrightarrow y \in$ *radical-sqrt* $\Longrightarrow x{*}y \in$ *radical-sqrt*
  | *Sqrt*: $x \in$ *radical-sqrt* $\Longrightarrow x \geq 0 \Longrightarrow$ *sqrt* $x \in$ *radical-sqrt*

Here, we list some rules that will be used to prove that a given real is in *radical-sqrt*.

Given two reals in *radical-sqrt* x and y, the subtraction $x - y$ is also in *radical-sqrt*.

lemma *radical-sqrt-rule-subtraction*:
  $x \in$ *radical-sqrt* $\Longrightarrow y \in$ *radical-sqrt* $\Longrightarrow x{-}y \in$ *radical-sqrt*
  using *radical-sqrt.Neg radical-sqrt.Plus* **by** *fastforce*

Given two reals in *radical-sqrt* x and y, and $y \neq 0$, the division x/y is also in *radical-sqrt*.

lemma *radical-sqrt-rule-division*:
  $[\![x \in$ *radical-sqrt*; $y \in$ *radical-sqrt*; $y \neq 0]\!] \Longrightarrow x/y \in$ *radical-sqrt*
  using *divide-real-def radical-sqrt.Inverse radical-sqrt.Times* **by** *auto*

Given a positive real x in *radical-sqrt*, its square x^2 is also in *radical-sqrt*.

lemma *radical-sqrt-rule-power2*:
  $x \in$ *radical-sqrt* $\Longrightarrow x \geq 0 \Longrightarrow x\hat{}2 \in$ *radical-sqrt*
  by (*simp add*: *power2-eq-square radical-sqrt.Times*)

Given a positive real x in *radical-sqrt*, its cube x^3 is also in *radical-sqrt*.

lemma *radical-sqrt-rule-power3*:
  $x \in$ *radical-sqrt* $\Longrightarrow x \geq 0 \Longrightarrow x\hat{}3 \in$ *radical-sqrt*
  by (*metis power3-eq-cube radical-sqrt.intros*(*5*))

2.6 Introduction of the datatype expr which represents radical expressions

An expression expr is either a rational constant: Const or the negation of an expression or the inverse of an expression or the addition of two expressions or the multiplication of two expressions or the square root of an expression.

datatype *expr* = *Const rat* | *Negation expr* | *Inverse expr* | *Addition expr expr* | *Multiplication expr expr* | *Sqrt expr*

The function *translation* translates a given expression into its equivalent real.

fun *translation* :: *expr* => *real* (‹(2⦃-⦄)›)
  where
  translation (*Const x*) = *of-rat x*|
  translation (*Negation e*) = − *translation e*|
  translation (*Inverse e*) = (*1*::*real*) / *translation e*|
  translation (*Addition e1 e2*) = *translation e1* + *translation e2*|
  translation (*Multiplication e1 e2*) = *translation e1* ∗ *translation e2*|
  translation (*Sqrt e*) = (*if translation e* < *0 then 0 else sqrt* (*translation e*))

Define the set of all the radicals of a given expression. For example, suppose *expr* is of the form : expr = Addition (Sqrt (Addition (Const a) Sqrt (Const b))) (Sqrt (Addition (Const c) (Sqrt (Sqrt (Const d))))), where a, b, c and d are rationals. This can be translated as follows: ⦃*expr*⦄ $= \sqrt{a+\sqrt{b}} + \sqrt{c+\sqrt{\sqrt{d}}}$. Moreover, the set *radicals* of this expression is : ⦃Addition (Const a) (Sqrt (Const b)), Const b, Addition (Const c) (Sqrt (Sqrt (Const d))), Sqrt (Const d), Const d⦄.

fun *radicals* :: *expr* => *expr set*
  where
  radicals (*Const x*) = {}|
  radicals (*Negation e*) = (*radicals e*)|
  radicals (*Inverse e*) = (*radicals e*)|
  radicals (*Addition e1 e2*) = ((*radicals e1*) ∪ (*radicals e2*))|
  radicals (*Multiplication e1 e2*) = ((*radicals e1*) ∪ (*radicals e2*))|
  radicals (*Sqrt e*) = (*if* ⦃*e*⦄ < *0 then radicals e else* {*e*} ∪ (*radicals e*))

If r is in *radicals* of e then the set *radical-sqrt* of r is a subset (strictly speaking) of the set *radicals* of e.

lemma *radicals-expr-subset*: *r* ∈ *radicals e* ⟹ *radicals r* ⊂ *radicals e*
  by (*induct e*, *auto simp*: *if-split-asm*)

If x is in *radical-sqrt* then there exists a radical expression e which translation is x (it is important to notice that this expression is not necessarily unique).

lemma *radical-sqrt-correct-expr*:
  x ∈ *radical-sqrt* ⟹ ∃ *e*. ⦃*e*⦄ = *x*
proof (*induction rule*: *radical-sqrt.induct*)
  case (*Rat x*)
  then show *?case*
    by (*metis Rats-cases translation.simps*(*1*))
next

case (*Sqrt x*)
then show *?case*
by (*meson linorder-not-le translation.simps*(*6*))
qed (*use translation.simps* **in** *blast*)+

The order of an expression is the maximum number of radicals one over another occurring in a given expression. Using the example above, suppose *expr* is of the form : expr = Addition (Sqrt (Addition (Const a) Sqrt (Const b))) (Sqrt (Addition (Const c) (Sqrt (Sqrt (Const d))))), where a, b, c and d are rationals and which can be translated as follows: $\{\!|expr|\!\}$ $= \sqrt{a+\sqrt{b}+\sqrt{c+\sqrt{\sqrt{d}}}}$. The order of *expr* is $max(2,3) = 3$.

fun *order* :: *expr* => *nat*
where
order (*Const x*) = *0*|
order (*Negation e*) = *order e*|
order (*Inverse e*) = *order e*|
order (*Addition e1 e2*) = *max* (*order e1*) (*order e2*)|
order (*Multiplication e1 e2*) = *max* (*order e1*) (*order e2*)|
order (*Sqrt e*) = *1* + *order e*

If an expression s is one of the radicals (or in *radicals*) of the expression r, then its order is smaller (strictly speaking) then the order of r.

lemma *in-radicals-smaller-order*:
$s \in$ *radicals* $r \Longrightarrow$ (*order* s) < (*order* r)
by (*induction r*) (*force split*: *if-splits*)+

The following theorem is the converse of the previous lemma.

lemma *in-radicals-smaller-order-contrap*:
(*order* s) $\geq$ (*order* r) $\Longrightarrow \neg$ ($s \in$ *radicals* r)
by (*metis in-radicals-smaller-order leD*)

An expression r cannot be one of its own radicals.

lemma *not-in-own-radicals*:
$\neg$ ($r \in$ *radicals* r)
by (*metis in-radicals-smaller-order order-less-irrefl*)

If an expression e is a radical expression and it has no radicals then its translation is a rational.

lemma *radicals-empty-rational*: *radicals* $e = \{\} \Longrightarrow \{\!|e|\!\} \in \mathbb{Q}$
by (*induct e*, *auto*)

A finite non-empty set of natural numbers has necessarily a maximum.

lemma *finite-set-has-max*:
finite (s:: *nat set*) $\Longrightarrow s \neq \{\} \Longrightarrow \exists\, k \in s.\ \forall\ p \in s.\ p \leq k$
by (*metis Max-ge Max-in*)

There is a finite number of radicals in an expression.

lemma *finite-radicals*: *finite* (*radicals* e)
  by (*induct* e, *auto*)

We define here a new set corresponding to the orders of each element in the set *radicals* of an expression *expr*. Using the example above, suppose *expr* is of the form : expr = Addition (Sqrt (Addition (Const a) Sqrt (Const b))) (Sqrt (Addition (Const c) (Sqrt (Sqrt (Const d))))), where a, b, c and d are rationals and which can be translated as follows: $\{\!|expr|\!\} = \sqrt{a+\sqrt{b}} + \sqrt{c+\sqrt{\sqrt{d}}}$. The set *radicals* of *expr* is {Addition (Const a) Sqrt (Const b), Const b, Addition (Const c) (Sqrt (Sqrt (Const d))), Sqrt (Const d), Const d}; therefore, the set *order-radicals* of this set is $\{1, 0, 2, 1, 0\}$.

fun *order-radicals*:: *expr set* => *nat set*
  where *order-radicals* $s = \{y.\ \exists\ x \in s.\ y = order\ x\}$

If the set of radicals of an expression e is not empty and is finite then the set *order-radicals* of the set of radicals of e is not empty and is also finite.

The following lemma states that given an expression e, if the set *order-radicals* of the set *radicals* e is not empty and is finite, then there exists a radical r of e which is of highest order among the radicals of e.

lemma *finite-order-radicals-has-max*:
  $[\![$*order-radicals* (*radicals* e) $\neq \{\}$;
    finite (*order-radicals* (*radicals* e))$]\!]$
  $\Longrightarrow \exists\, r.\ r \in radicals\ e \wedge (\forall\, s{\in}radicals\ e.\ order\ s \leq order\ r)$
  using *finite-set-has-max* [*of order-radicals* (*radicals* e)]
    by *auto*

This important lemma states that in an expression that has at least one radical, we can find an upmost radical r which is not radical of any other term of the expression e. It is also important to notice that this upmost radical is not necessarily unique and is not the term of highest order of the expression e. Using the example above, suppose e is of the form : e = Addition (Sqrt (Addition (Const a) Sqrt (Const b))) (Sqrt (Addition (Const c) (Sqrt (Sqrt (Const d))))), where a, b, c and d are rationals and which can be translated as follows: $\{\!|e|\!\} = \sqrt{a+\sqrt{b}} + \sqrt{c+\sqrt{\sqrt{d}}}$. The possible upmost radicals in this expression are Addition (Const a) (Sqrt (Const b)) or Addition (Const c) (Sqrt (Sqrt (Const d))).

lemma *finite-order-radicals*:
  radicals $e \neq \{\} \Longrightarrow$ *finite* (*radicals* e) $\Longrightarrow$
   order-radicals (*radicals* e) $\neq \{\} \wedge$ *finite* (*order-radicals* (*radicals* e))
  by *auto*

lemma *upmost-radical-sqrt2*:

radicals e $\neq$ {} $\Longrightarrow$
  $\exists$ *r* $\in$ *radicals e.* $\forall$ *s* $\in$ *radicals e. r* $\notin$ *radicals s*
 by (*meson finite-order-radicals finite-order-radicals-has-max finite-radicals in-radicals-smaller-order leD*)

The following 7 lemmas are used to prove the main lemma *radical-sqrt-normal-form* which states that if an expression e has at least one radical then it can be written in a normal form. This means that there exist three radical expressions a, b and r such that $\{\!|e|\!\} = \{\!|a|\!\} + \{\!|b|\!\} * \sqrt{\{\!|r|\!\}}$ and the radicals of a are radicals of e but are not r, and the same goes for the radicals of b and r. It is important to notice that a, b and r are not unique and *Sqrt r* is not necessarily the term of highest order.

lemma *eq-sqrt-squared*:
 (*x*::*real*) $\geq$ *0* $\Longrightarrow$ (*sqrt x*) $*$ (*sqrt x*) = *x*
 by (*metis abs-of-nonneg real-sqrt-abs2 real-sqrt-mult*)

lemma *radical-sqrt-normal-form-inverse*:
 assumes *z* $\geq$ *0 x* $\neq$ *y* $*$ *sqrt z*
 shows
  1 / (*x* + *y* $*$ *sqrt z*) =
   x / (*x* $*$ *x* $-$ *y* $*$ *y* $*$ *z*) $-$ (*y* $*$ *sqrt z*) / (*x* $*$ *x* $-$ *y* $*$ *y* $*$ *z*)
proof $-$
 have *1* / (*x* + *y* $*$ *sqrt z*) = ((*x* $-$ *y* $*$ *sqrt z*) / (*x* + *y* $*$ *sqrt z*)) / (*x* $-$ *y* $*$ *sqrt z*)
  by (*auto simp*: *eq-divide-imp assms*)
 also have ... = *x* / (*x* $*$ *x* $-$ *y* $*$ *y* $*$ *z*) $-$ (*y* $*$ *sqrt z*) / (*x* $*$ *x* $-$ *y* $*$ *y* $*$ *z*)
  by (*auto simp*: *algebra-simps eq-sqrt-squared diff-divide-distrib assms*)
 finally show *?thesis* **.**
qed

lemma *radical-sqrt-normal-form-lemma*:
 fixes *e*::*expr*
 assumes *radicals e* $\neq$ {}
 and $\forall$ *s* $\in$ *radicals e. r* $\notin$ *radicals s*
 and *r* $\in$ *radicals e*
 shows $\exists$ *a b. 0* $\leq$ $\{\!|r|\!\}$ $\wedge$ $\{\!|e|\!\}$ = $\{\!|a|\!\}$ + $\{\!|b|\!\}$ $*$ *sqrt* $\{\!|r|\!\}$ &
     radicals a $\cup$ *radicals b* $\cup$ *radicals r* $\subseteq$ *radicals e* &
     r $\notin$ *radicals a* $\cup$ *radicals b*
   (**is** $\exists$ *a b. ?concl e a b*)
 using *assms*
proof (*induct e*)
 case (*Const rat*) **thus** *?case*
  by *auto*
next
 case (*Negation e*)
 obtain *a b*
  where *a2*: *?concl e a b*
  by (*metis Negation radicals.simps*(*2*))

```
  hence {|Negation e|} = {|Negation a|} + {|Negation b|} * sqrt {|r|}
    by simp
  thus ?case using a2
    by (metis radicals.simps(2))
next
  case (Inverse e)
  obtain a b where ab: ?concl e a b
    by (metis Inverse radicals.simps(3))
  show ?case
  proof (cases {|b|} * sqrt {|r|} = {|a|})
    case eq: True
    show ?thesis
    proof (cases {|a|} = 0)
      case True
      with eq show ?thesis
        by (smt (verit) ab radicals.simps(3) translation.simps(3))
    next
      case False
      let ?a = Multiplication (Const 1) (Inverse (Multiplication (Const 2) a))
      let ?b = Const 0
      show ?thesis
        by (rule exI [where x= ?a], rule exI [where x= ?b], (use ab eq in force))
    qed
  next
    case False
       let ?a = Multiplication a (Inverse (Addition (Multiplication a a) (Negation
(Multiplication (Multiplication b b) r))))
       let ?b = Negation (Multiplication b (Inverse (Addition (Multiplication a a)
(Negation (Multiplication (Multiplication b b) r)))))
     show ?thesis
      apply (rule exI [where x= ?a], rule exI [where x= ?b])
      using ab False
    by (simp add: algebra-simps not-in-own-radicals eq-diff-eq' radical-sqrt-normal-form-inverse)
  qed
next
  case (Addition e1 e2)
  hence d1: ∀ s ∈ radicals e1 ∪ radicals e2. r ∉ radicals s
    by (metis radicals.simps(4))
  show ?case
  proof (cases r ∈ radicals e1 ∧ r ∈ radicals e2)
    case True
    obtain a1 b1 a2 b2
      where ab: ?concl e1 a1 b1
        and bb: ?concl e2 a2 b2
      using Addition.hyps
      by (simp add: d1) (metis True empty-iff)
    thus ?thesis
      apply (rule-tac x = Addition a1 a2 in exI)
      apply (rule-tac x = Addition b1 b2 in exI)
```

```
      by (auto simp: comm-semiring-class.distrib)
  next
    case False
    thus ?thesis
    proof (cases r ∈ radicals e1)
      case True
      obtain a1 b1
      where 0 ≤ ⦃r⦄ ?concl e1 a1 b1
        using Addition.hyps
        by (auto simp: d1) (metis True empty-iff)
      with False True show ?thesis
        apply (rule-tac x = Addition a1 e2 in exI)
        apply (rule-tac x = b1 in exI)
        by auto
    next
      case False
      obtain a2 b2
        where 0 ≤ ⦃r⦄ ?concl e2 a2 b2
        using Addition d1
        by (metis False Un-iff empty-iff radicals.simps(4))
      with False show ?thesis
        apply (rule-tac x = Addition a2 e1 in exI)
        apply (rule-tac x = b2 in exI)
        by auto
    qed
  qed
next
  case (Multiplication e1 e2)
  show ?case
  proof (cases r ∈ radicals e1 ∧ r ∈ radicals e2)
    case True
    then obtain a1 b1 a2 b2
      where ?concl e1 a1 b1 ?concl e2 a2 b2
      using Multiplication
      by simp (metis True empty-iff)
    thus ?thesis
     apply (rule-tac x = Addition (Multiplication a1 a2) (Multiplication r (Multiplication
b1 b2)) in exI)
      apply (rule-tac x = Addition (Multiplication a1 b2) (Multiplication a2 b1) in
exI)
      by (auto simp: not-in-own-radicals algebra-simps eq-sqrt-squared)
  next
    case False
    thus ?thesis
    proof (cases r ∈ radicals e1)
      case True
      then obtain a1 b1
      where ?concl e1 a1 b1
        using Multiplication.hyps Multiplication(4)
```

by *auto* (*metis True empty-iff*)
with *False True* **show** *?thesis*
apply (*rule-tac x* = *Multiplication a1 e2* **in** *exI*)
apply (*rule-tac x* = *Multiplication b1 e2* **in** *exI*)
by (*force simp add*: *algebra-simps*)
next
case *False*
then obtain *a2 b2*
where *?concl e2 a2 b2*
using *Multiplication.hyps Multiplication*(*4*) *Multiplication*(*5*)
by *auto blast*
with *False* **show** *?thesis*
apply (*rule-tac x* = *Multiplication a2 e1* **in** *exI*)
apply (*rule-tac x* = *Multiplication b2 e1* **in** *exI*)
by (*force simp add*: *algebra-simps*)
qed
qed
next
case (*Sqrt e*)
show *?case*
proof (*cases* $\{\!|e|\!\} < 0$)
case *True* **with** *Sqrt* **show** *?thesis*
by (*intro exI* [**where** *x* = *Const 0*]) *auto*
next
case *False*
with *Sqrt* **show** *?thesis*
apply (*rule-tac x* = *Const 0* **in** *exI*)
apply (*rule-tac x* = *Const 1* **in** *exI*)
by (*auto simp*: *linorder-not-less*)
qed
qed

This main lemma is essential for the remaining part of the proof.

theorem *radical-sqrt-normal-form*:
radicals e $\neq$ {} $\Longrightarrow$
$\exists$ *r* $\in$ *radicals e*.
$\exists$ *a b*. $\{\!|e|\!\} = \{\!|$*Addition a* (*Multiplication b* (*Sqrt r*))$|\!\} \wedge \{\!|r|\!\} \geq 0 \wedge$
radicals a $\cup$ *radicals b* $\cup$ *radicals r* $\subseteq$ *radicals e* &
r $\notin$ *radicals a* $\cup$ *radicals b* $\cup$ *radicals r*
using *upmost-radical-sqrt2* [*of e*] *radical-sqrt-normal-form-lemma*
by *auto* (*metis all-not-in-conv leD*)

2.7 Important properties of the roots of a cubic equation

The following 7 lemmas are used to prove a main result about the properties of the roots of a cubic equation (*cubic-root-radical-sqrt-rational*) which states that assuming that a b and c are rationals and that x is a radical satisfying $x^3 + ax^2 + bx + c = 0$ then there exists a rational root. This lemma will be used in the proof of the impossibility of trisection an angle and of duplicating a cube.

lemma *cubic-root-radical-sqrt-steplemma*:
  fixes P :: *real set*
  assumes *Nats* [*THEN subsetD*, *intro*]: $Nats \subseteq P$
  and *Neg*: $\forall x \in P.\ -x \in P$
  and *Inv*: $\forall x \in P.\ x \neq 0 \longrightarrow 1/x \in P$
  and *Add*: $\forall x \in P.\ \forall y \in P.\ x+y \in P$
  and *Mult*: $\forall x \in P.\ \forall y \in P.\ x*y \in P$
  and *a*: $a \in P$ **and** *b*: $b \in P$ **and** *c*: $c \in P$
  and *eq0*: $z\hat{}3 + a * z\hat{}2 + b * z + c = 0$
  and *u*: $u \in P$
  and *v*: $v \in P$
  and *s*: $s * s \in P$
  and *z*: $z = u + v * s$
  shows $\exists w \in P.\ w\hat{}3 + a * w\hat{}2 + b * w + c = 0$
proof (*cases* $v * s = 0$)
  case *True*
  thus *?thesis*
    by (*metis eq0 u z add-0-iff*)
next
  case *False*
  hence *sl0*: $v \neq 0$
    by (*metis mult-eq-0-iff*)
  from *Add Neg* **have** *Minus*: $\forall x \in P.\ \forall y \in P.\ x - y \in P$ **by** (*simp only*: *diff-conv-add-uminus*) *blast*
  have *l2*: $(u\hat{}3 + 3 * u * v\hat{}2 * s\hat{}2 + a * u\hat{}2 + a * v\hat{}2 * s\hat{}2 + b * u + c) + (3 * u\hat{}2 * v + v\hat{}3 * s\hat{}2 + 2 * a * u * v + b * v) * s = 0$
    using *eq0 z* **by** *algebra*
  show *?thesis*
  proof (*cases* $3 * u\hat{}2 * v + v\hat{}3 * s\hat{}2 + 2 * a * u * v + b * v \neq 0$)
    case *True*
    hence $s * ((3 * u\hat{}2 * v + v\hat{}3 * s\hat{}2 + 2 * a * u * v + b * v) * (1/ (3 * u\hat{}2 * v + v\hat{}3 * s\hat{}2 + 2 * a * u * v + b * v)))= - (u\hat{}3 + 3 * u * v\hat{}2 * s\hat{}2 + a * u\hat{}2 + a * v\hat{}2 * s\hat{}2 + b * u + c)* (1/ (3 * u\hat{}2 * v + v\hat{}3 * s\hat{}2 + 2 * a * u * v + b * v))$
      using *l2*
      by *algebra*
    hence $s * ((3 * u\hat{}2 * v + v\hat{}3 * s\hat{}2 + 2 * a * u * v + b * v) /$
          $(3 * u\hat{}2 * v + v\hat{}3 * s\hat{}2 + 2 * a * u * v + b * v))$
        $= - (u\hat{}3 + 3 * u * v\hat{}2 * s\hat{}2 + a * u\hat{}2 + a * v\hat{}2 * s\hat{}2 + b * u + c) *$
          $(1 / (3 * u\hat{}2 * v + v\hat{}3 * s\hat{}2 + 2 * a * u * v + b * v))$
      by *auto*
    hence $s = - (u\hat{}3 + 3 * u * v\hat{}2 * s\hat{}2 + a * u\hat{}2 + a * v\hat{}2 * s\hat{}2 + b * u + c) *$
          $(1 /(3 * u\hat{}2 * v + v\hat{}3 * s\hat{}2 + 2 * a * u * v + b * v))$
      by (*metis mult-1-right True divide-self-if*)
    hence $*$: $s = - (u *u *u + 3 * u * v *v * (s*s) + a * u *u + a * v*v * (s$

$*s) + b * u + c) *$
$(1 / (3 * u *u * v + v *v*v * (s *s) + 2 * a * u * v + b * v))$
by (*simp add*: *algebra-simps power2-eq-square power3-eq-cube*)
have $(3*u*u * v + v*v*v * (s *s) + 2 * a * u * v + b * v) \in P$
using *a b u v s Nats Mult Add*
by *auto*
hence $1 / (3 * u *u * v + v *v*v * (s *s) + 2 * a * u * v + b * v) \in P$
using *Inv True* **by** *auto*
moreover
have $-(u*u*u + 3 * u * v *v * (s*s) + a * u *u + a * v*v * (s *s) + b * u + c) \in P$
using *a b c u v s Mult Add Neg Minus Nats*
by *simp*
ultimately have $- (u *u *u + 3 * u * v *v * (s*s) + a * u *u + a * v*v * (s *s) + b * u + c) * (1 /(3 * u *u * v + v *v*v * (s *s) + 2 * a * u * v + b * v)) \in P$
using *Mult* **by** *metis*
hence $s \in P$
using $*$ **by** *auto*
hence $z \in P$
using *z u v Mult Add* **by** *auto*
thus *?thesis*
using *eq0* **by** *auto*
next
case *False*
have $(- a - 2 * u)\hat{}3 + a * (- a - 2 * u)\hat{}2 + b * (- a - 2 * u) + c =$
$(- a - 2 * u)\hat{}3 + a * (- a - 2 * u)\hat{}2 + (- (3 * u\hat{}2 + v\hat{}2 * s\hat{}2 + 2 * a * u)) *$
$(- a - 2 * u) + (- (u\hat{}3) - 3 * u * v\hat{}2 * s\hat{}2 - a * u\hat{}2 - a * v\hat{}2 * s\hat{}2 + 3 * u\hat{}3 + v\hat{}2 * s\hat{}2 * u + 2 * a * u\hat{}2)$
using *l2 False sl0*
by *algebra*
also have $\ldots = 0$
by (*simp add*: *algebra-simps power-def*)
finally show *?thesis*
by (*metis a u Add Neg diff-conv-add-uminus mult-2*)
qed
qed

lemma *cubic-root-radical-sqrt-steplemma-sqrt*:
assumes *Nats* [*THEN subsetD*, *intro*]: $Nats \subseteq P$
and $\forall x \in P.\ -x \in P$
and $\forall x \in P.\ x \neq 0 \longrightarrow 1/x \in P$
and $\forall x \in P.\ \forall y \in P.\ x+y \in P$
and $\forall x \in P.\ \forall y \in P.\ x*y \in P$
and $(a \in P)$ **and** b: $(b \in P)$ **and** c: $(c \in P)$
and $z\hat{}3 + a * z\hat{}2 + b * z + c = 0$
and $u \in P\ v \in P\ s \in P$
and $s \geq 0$

and $z = u + v * sqrt\ s$
shows $\exists w \in P.\ w\hat{}3 + a * w\hat{}2 + b * w + c = 0$
proof−
have $(sqrt\ s) * (sqrt\ s) \in P$
by (*metis eq-sqrt-squared* ‹$s \in P$› ‹$s \geq 0$›)
thus *?thesis*
using *cubic-root-radical-sqrt-steplemma* [*of P a b c z u v sqrt s*] *assms*
by *auto*
qed

lemma *cubic-root-radical-sqrt-lemma*:
fixes *e*::*expr*
assumes *a*: $a \in \mathbb{Q}$ **and** *b*: $b \in \mathbb{Q}$ **and** *c*: $c \in \mathbb{Q}$
and *notEmpty*: *radicals e* $\neq \{\}$
and *eq0*: $\{\!|e|\!\}\hat{}\ 3 + a * \{\!|e|\!\}\hat{}2 + b * \{\!|e|\!\} + c = 0$
shows $\exists$ *e1*. *radicals e1* $\subset$ *radicals e* $\wedge$ $(\{\!|e1|\!\}\hat{}3 + a * \{\!|e1|\!\}\hat{}2 + b * \{\!|e1|\!\} + c = 0)$
proof −
obtain *r u v*
where *hypsruv*: $\{\!|r|\!\} \geq 0$ $r \in$ *radicals e*
$\{\!|e|\!\} = \{\!|$*Addition u* (*Multiplication v* (*Sqrt r*))$|\!\}$
radicals u $\cup$ *radicals v* $\cup$ *radicals r* $\subseteq$ *radicals e*
$r \notin$ *radicals u* $\cup$ *radicals v* $r \notin$ *radicals r*
using *notEmpty radical-sqrt-normal-form* [*of e*]
by *blast*
let *?E* = $\{x.\ \exists\ ex.\ (\{\!|ex|\!\} = x) \wedge (($*radicals ex*$) \subseteq ($*radicals e*$)) \wedge (r \notin ($*radicals ex*$))\}$
have *NatsE*: *Nats* $\subseteq$ *?E*
by (*force elim*: *Nats-cases intro*: *exI*[*of - Const* (*rat-of-nat n*) **for** *n*])
have *negE*: $\forall x \in$ *?E*. $-x \in$ *?E*
using *hypsruv* **by** (*force intro*: *exI*[*of - Negation ex* **for** *ex*])
have *invE*: $\forall x \in$ *?E*. $x \neq 0$ −−> $1/x \in$ *?E*
using *hypsruv* **by** (*force intro*: *exI*[*of - Inverse ex* **for** *ex*])
have *addE*: $\forall x \in$ *?E*. $\forall y \in$ *?E*. $x+y \in$ *?E*
using *hypsruv* **by** (*force intro*: *exI*[*of - Addition ex1 ex2* **for** *ex1 ex2*])
have *multE*: $\forall x \in$ *?E*. $\forall y \in$ *?E*. $x*y \in$ *?E*
using *hypsruv* **by** (*force intro*: *exI*[*of - Multiplication ex1 ex2* **for** *ex1 ex2*])
obtain *ra rb rc*
where *hypsra*: $a =$ *of-rat ra*
and *hypsrb*: $b =$ *of-rat rb*
and *hypsrc*: $c =$ *of-rat rc*
unfolding *Rats-def*
by (*metis Rats-cases a b c*)
have $a \in$ *?E* $\wedge$ $b \in$ *?E* $\wedge$ $c \in$ *?E* $\wedge$ $\{\!|u|\!\} \in$ *?E* $\wedge$ $\{\!|v|\!\} \in$ *?E* $\wedge$ $\{\!|r|\!\} \in$ *?E* $\wedge$ $\{\!|r|\!\} \geq 0 \wedge \{\!|e|\!\} = \{\!|u|\!\} + \{\!|v|\!\} * sqrt\ \{\!|r|\!\}$
using *a b c notEmpty hypsruv hypsra hypsrb hypsrc*
by (*auto intro*: *exI*[*of - Const x* **for** *x*])
with *eq0 hypsruv NatsE negE invE addE multE*
cubic-root-radical-sqrt-steplemma-sqrt [*of ?E a b c*$\{\!|e|\!\}$ $\{\!|u|\!\}$ $\{\!|v|\!\}$ $\{\!|r|\!\}$]

```
    obtain w where w ∈ ?E ∧ (w^3 + a ∗ w^2 + b ∗ w + c = 0)
      by auto
    then obtain e2
      where {|e2|} = w radicals e2 ⊆ radicals e r ∉ radicals e2
            {|e2|}^3 + a ∗ {|e2|}^2 + b ∗ {|e2|} + c = 0
      by auto
    with hypsruv show ?thesis
      by (metis subset-iff-psubset-eq)
qed

lemma cubic-root-radical-sqrt:
  assumes abc: a ∈ ℚ b ∈ ℚ c ∈ ℚ
  shows card (radicals e) = n ⟹ {|e|}^3 + a ∗ {|e|}^2 + b ∗ {|e|} + c = 0 ⟹
        ∃ x ∈ ℚ. x^3 + a ∗ x^2 + b ∗ x + c = 0
proof (induct n arbitrary: e rule: less-induct)
  case (less n)
  thus ?case
  proof cases
    assume n: n = 0
    thus ?thesis
      using less.prems radicals-empty-rational [of e] finite-radicals [of e]
      by (auto simp: card-eq-0-iff n)
  next
    assume n ≠ 0
    hence card (radicals e) ≠ 0
      using less.prems by auto
    hence radicals e ≠ {}
      by (metis card.empty)
    hence ∃ e1. radicals e1 ⊂ radicals e ∧ ({|e1|}^3 + a ∗ {|e1|}^2 + b ∗ {|e1|} +
c = 0)
      using abc less.prems cubic-root-radical-sqrt-lemma [of a b c e]
      by auto
    then obtain e1
     where hypse1: radicals e1 ⊂ radicals e ∧ ({|e1|}^3 + a ∗ {|e1|}^2 + b ∗ {|e1|}
+ c = 0)
      by auto
    hence card (radicals e1) < card (radicals e)
      by (metis finite-radicals psubset-card-mono)
    hence card (radicals e1) < n ∧ a : ℚ ∧ b : ℚ ∧ c : ℚ ∧ {|e1|}^3 + a ∗ {|e1|}^2
+ b ∗ {|e1|} + c = 0
      using hypse1 less.prems abc
      by auto
    thus ?thesis using less.hyps [of - e1]
      by auto
  qed
qed
```

Now we can prove the final result about the properties of the roots of a cubic equation.

theorem *cubic-root-radical-sqrt-rational*:
assumes *a*: $a \in \mathbb{Q}$ **and** *b*: $b \in \mathbb{Q}$ **and** *c*: $c \in \mathbb{Q}$
and *x*: $x \in$ *radical-sqrt*
and *x-eqn*: $x\hat{}3 + a * x\hat{}2 + b * x + c = 0$
shows *c*: $\exists x \in \mathbb{Q}.\ x\hat{}3 + a * x\hat{}2 + b * x + c = 0$
proof−
obtain *e n*
where $\{\!|e|\!\} = x \wedge (\{\!|e|\!\}\hat{}\ 3 + a * \{\!|e|\!\}\hat{}2 + b * \{\!|e|\!\} + c = 0)$ $n = card$ (*radicals e*)
using *x x-eqn radical-sqrt-correct-expr* [*of x*] **by** *auto*
thus *?thesis*
using *cubic-root-radical-sqrt* [*OF a b c*] **by** *auto*
qed

2.8 Important properties of radicals

lemma *sqrt-roots*:
$y\hat{}2{=}x \Longrightarrow x{\geq}0 \wedge (sqrt\ (x) = y \mid sqrt\ (x) = -y)$
by *auto*

lemma *radical-sqrt-linear-equation*:
assumes $a \in$ *radical-sqrt* $b \in$ *radical-sqrt*
and $\neg\ (a = 0 \wedge b = 0)$
and $a * x + b = 0$
shows $x \in$ *radical-sqrt*
proof (*cases a=0*)
case *True*
with *assms* **show** *?thesis*
by *auto*
next
case *False*
hence $x = -\ b\ /a$
using *assms* **by** (*simp add*: *field-simps*)
also have $\ldots \in$ *radical-sqrt*
by (*simp add*: *False assms radical-sqrt.Neg radical-sqrt-rule-division*)
finally show *?thesis* **.**
qed

lemma *radical-sqrt-simultaneous-linear-equation*:
assumes $a \in$ *radical-sqrt*
and $b \in$ *radical-sqrt*
and $c \in$ *radical-sqrt*
and $d \in$ *radical-sqrt*
and $e \in$ *radical-sqrt*
and $f \in$ *radical-sqrt*
and *NotNull*: $\neg\ (a*e - b*d = 0 \wedge a*f - c*d = 0 \wedge e*c = b*f)$
and *eq*: $a*x + b*y = c$ $d*x + e*y = f$
shows $x \in$ *radical-sqrt* $\wedge$ $y \in$ *radical-sqrt*

```
proof (cases a∗e − b∗d =0)
  case False
  hence (a∗e−b∗d) ∗ x = (e∗c−b∗f) using eq
    by algebra
  hence x: x = (e∗c−b∗f) / (a∗e−b∗d)
    using False by (simp add: field-simps)
  hence (a∗e−b∗d) ∗ y = (a∗f − d∗c) using eq
    by algebra
  hence y: y = (a∗f−d∗c)/(a∗e−b∗d)
    using False by (simp add: field-simps)
  have ae-rad: (a∗e −b∗d) ∈ radical-sqrt
    using assms radical-sqrt.simps
    by (metis radical-sqrt.intros(5) radical-sqrt-rule-subtraction)
   hence ((e∗c−b∗f) / (a∗e−b∗d)) ∈ radical-sqrt  ((a∗f−d∗c) / (a∗e−b∗d)) ∈
radical-sqrt
   using False assms by (auto intro!: radical-sqrt.intros(5) radical-sqrt-rule-division
radical-sqrt-rule-subtraction)
  thus ?thesis
    by (simp add: x y)
next
  case True
  hence (a∗e−b∗d) ∗ x = (e∗c−b∗f) (a∗e−b∗d) ∗ y = (a∗f − d∗c) using eq
    by algebra+
  thus ?thesis using NotNull True
    by simp
qed


lemma radical-sqrt-quadratic-equation:
  assumes a ∈ radical-sqrt
      and b ∈ radical-sqrt
      and c ∈ radical-sqrt
      and eq0: a∗x^2+b∗x+c = 0
      and NotNull: ¬ (a = 0 ∧ b = 0 ∧ c = 0)
  shows x ∈ radical-sqrt
proof (cases a=0)
  case True
  have ¬ (b = 0 ∧ c = 0)
    by (metis True NotNull)
  with assms show ?thesis
    by (metis True add-0 mult-zero-left radical-sqrt-linear-equation)
next
  case False
  hence (2∗a∗x+b)^2 = 4∗a∗(− c)+b^2 using eq0
    by algebra
 hence (b^2 − 4∗a∗c)≥0 ∧ (sqrt ((b^2 − 4∗a∗c)) = (2∗a∗x+b) ∨ sqrt ((b^2 −
4∗a∗c)) = −(2∗a∗x+b))
    using sqrt-roots [of 2∗a∗x+b b^2 − 4∗a∗c]
    by auto
```

hence *quad*: $b^2 - 4*a*c \geq 0 \wedge ((-b + sqrt\ (b^2 - 4*a*c)) / (2*a) = x \vee (-b - sqrt\ (b^2 - 4*a*c)) / (2*a) = x)$
using *False*
by *auto*
have $4*a*c \in$ *radical-sqrt*
using *Rats-number-of assms radical-sqrt.simps* **by** *blast*
hence $b^2 - 4*a*c \in$ *radical-sqrt* **using** *assms*
by (*simp add*: *power2-eq-square radical-sqrt.Times radical-sqrt-rule-subtraction*)
hence *RS1*: $sqrt\ (b^2 - 4*a*c) \in$ *radical-sqrt*
using *quad*
by (*metis radical-sqrt.intros*(*6*))
hence *RS2*: $(-b + sqrt\ (b^2 - 4*a*c)) / (2*a) \in$ *radical-sqrt*
using *assms False*
by (*metis double-zero-sym mult-2 radical-sqrt.Neg radical-sqrt.Plus radical-sqrt-rule-division*)
have $(-b - sqrt\ (b^2 - 4*a*c)) / (2*a) \in$ *radical-sqrt*
using *assms False RS1* **by** (*smt* (*verit*) *radical-sqrt.Neg radical-sqrt.Plus radical-sqrt-rule-division*)
thus *?thesis*
by (*metis quad RS2*)
qed

lemma *radical-sqrt-simultaneous-linear-quadratic*:
assumes $a \in$ *radical-sqrt*
and $b \in$ *radical-sqrt*
and $c \in$ *radical-sqrt*
and $d \in$ *radical-sqrt*
and $e \in$ *radical-sqrt*
and $f \in$ *radical-sqrt*
and *NotNull*: $\neg(d=0 \wedge e=0 \wedge f=0)$
and *eq*: $(x-a)^2 + (y-b)^2 = cd*x+e*y = f$
shows $x \in$ *radical-sqrt* $\wedge$ $y \in$ *radical-sqrt*
proof (*cases* $d=0 \wedge e=0$)
case *True*
with *assms* **show** *?thesis*
by (*metis add-0 mult-zero-left*)
next
case *False*
hence *l10*: $(e^2 + d^2) * x^2 + (2*e*b*d - 2*a*e^2 - 2*d*f)*x + (a^2 * e^2 + f^2 - 2* e *b* f + b^2 * e^2 - e^2 *c) = 0$
using *eq* **by** *algebra*
have *l12*: $\neg (e^2 +d^2 = 0 \wedge 2*e*b*d - 2*a*e^2 - 2*d*f = 0 \wedge a^2 * e^2 + f^2 - 2* e *b* f + b^2 * e^2 - e^2 *c = 0)$
using *False power-def*
by *auto*
have *l13*: $(e^2 +d^2) \in$ *radical-sqrt*
using *assms* **by** (*metis power2-eq-square radical-sqrt.Plus radical-sqrt.Times*)
have *2*: $2 \in$ *radical-sqrt*

```
    by (auto intro: radical-sqrt.intros)
  have (− 2∗d∗f) ∈ radical-sqrt using radical-sqrt.intros
    using 2 assms by presburger
  with assms have ((2∗e∗b∗d) + (− 2∗a∗e^2) + (− 2∗d∗f)) ∈ radical-sqrt
    by (metis Rats-number-of power2-eq-square radical-sqrt.Neg
        radical-sqrt.Plus radical-sqrt.Times radical-sqrt.intros(1))
  hence l14: (2∗e∗b∗d − 2∗a∗e^2 − 2∗d∗f) ∈ radical-sqrt
    by force
  have RS6: (a^2 ∗ e^2 + f^2 + (− 2 ∗ e ∗b∗ f) + b^2 ∗ e^2 + (− c∗ e^2)) ∈
radical-sqrt
    using assms
   by (metis 2 power2-eq-square radical-sqrt.Neg radical-sqrt.Plus radical-sqrt.Times)
  have a^2 ∗ e^2 + f^2 − 2∗ e ∗b∗ f + b^2 ∗ e^2 − e^2 ∗c = a^2 ∗ e^2 + f^2
+ (− 2 ∗ e ∗b∗ f) + b^2 ∗ e^2 + (− c∗ e^2)
    by auto
  hence (a^2 ∗ e^2 + f^2 − 2∗ e ∗b∗ f + b^2 ∗ e^2 − e^2 ∗c) ∈ radical-sqrt
    using RS6
    by metis
  hence x: x ∈ radical-sqrt
     using radical-sqrt-quadratic-equation [of e^2 +d^2 2∗e∗b∗d − 2∗a∗e^2 −
2∗d∗f a^2 ∗ e^2 + f^2 − 2∗ e ∗b∗ f + b^2 ∗ e^2 − e^2 ∗c x] l13 l14 l12 l10
    by auto
  hence y: y ∈ radical-sqrt
  proof (cases e = 0)
    case True
    hence 1 ∗ y^2 + (− 2∗ b) ∗ y + (b^2 + (x − a)^2 − c) = 0
      using eq by algebra
    moreover
    have 1 ∈ radical-sqrt
      by (metis Rats-1 radical-sqrt.intros(1))
    moreover
    have (− 2∗ b) ∈ radical-sqrt
      using assms
     by (metis minus-mult-commute mult-2 radical-sqrt.intros(2) radical-sqrt.intros(4))
    moreover
    have (b^2 + (x − a)^2 − c) ∈ radical-sqrt
      using assms x
        by (auto intro: radical-sqrt.intros radical-sqrt-rule-subtraction simp add:
power2-eq-square)
    ultimately show ?thesis
      using radical-sqrt-quadratic-equation [of 1::real − 2∗ b b^2 + (x − a)^2 −
c y]
      by auto
  next
    case False
    have (d∗x − f) ∈ radical-sqrt
     using assms x by (simp add: radical-sqrt.Times radical-sqrt-rule-subtraction)
    thus ?thesis
      using radical-sqrt-linear-equation [of e d∗x − f y] assms eq False
```

by *auto*
qed
show *?thesis*
by (*metis x y*)
qed

lemma *radical-sqrt-simultaneous-quadratic-quadratic*:
assumes $a \in$ *radical-sqrt*
and $b \in$ *radical-sqrt*
and $c \in$ *radical-sqrt*
and $d \in$ *radical-sqrt*
and $e \in$ *radical-sqrt*
and $f \in$ *radical-sqrt*
and *NotEqual*: $\neg\ (a = d \wedge b = e \wedge c = f)$
and *eq*: $(x - a)\hat{}2 + (y - b)\hat{}2 = c\ (x - d)\hat{}2 + (y - e)\hat{}2 = f$
shows $x \in$ *radical-sqrt* $\wedge\ y \in$ *radical-sqrt*
proof −
have $(x\hat{}2 - 2{*}a{*}x + a\hat{}2 + y\hat{}2 - 2{*}y{*}b + b\hat{}2) - (x\hat{}2 - 2{*}d{*}x + d\hat{}2 + y\hat{}2 - 2{*}y{*}e + e\hat{}2) = (c - f)$
using *eq* **by** (*simp add*: *algebra-simps power-def*)
hence *l4*: $(2{*}d - 2{*}a){*}x + (2{*}e - 2{*}b){*}y + (b\hat{}2 - e\hat{}2 + a\hat{}2 - d\hat{}2 + f - c) = 0$
by *algebra*
hence *l6*: $\neg\ ((2{*}d - 2{*}a) = 0 \wedge (2{*}e - 2{*}b) = 0 \wedge (b\hat{}2 - e\hat{}2) + (a\hat{}2 - d\hat{}2) + (f - c) = 0)$
using *NotEqual* **by** *algebra*
have *l7*: $(2{*}d - 2{*}a) \in$ *radical-sqrt*
using *assms* **by** (*metis mult-2 radical-sqrt.intros*(*4*) *radical-sqrt-rule-subtraction*)
have *l8*: $(2{*}e - 2{*}b) \in$ *radical-sqrt*
using *assms* **by** (*metis mult-2 radical-sqrt.intros*(*4*) *radical-sqrt-rule-subtraction*)
have *be-rad*: $(b\hat{}2 - e\hat{}2) \in$ *radical-sqrt*
using *assms* **by** (*metis power2-eq-square radical-sqrt.intros*(5) *radical-sqrt-rule-subtraction*)
have *ad-rad*: $(a\hat{}2 - d\hat{}2) \in$ *radical-sqrt*
using *assms* **by** (*metis power2-eq-square radical-sqrt.intros*(5) *radical-sqrt-rule-subtraction*)
have $(f - c) \in$ *radical-sqrt*
using *assms* **by** (*metis radical-sqrt-rule-subtraction*)
hence $-((b\hat{}2 - e\hat{}2) + (a\hat{}2 - d\hat{}2) + (f - c)) \in$ *radical-sqrt*
using *radical-sqrt.intros* **by** (*metis be-rad ad-rad*)
thus *?thesis*
using *radical-sqrt-simultaneous-linear-quadratic* [*of a b c* $(2{*}d - 2{*}a)$ $(2{*}e - 2{*}b)$ $-((b\hat{}2 - e\hat{}2) + (a\hat{}2 - d\hat{}2) + (f - c))$ *x y*]
using *assms l7 l8 l6 l4 NotEqual eq*
by *simp*
qed

2.9 Important properties of geometrical points which coordinates are radicals

lemma *radical-sqrt-line-line-intersection*:

assumes *absA*: (*abscissa* (*A*)) ∈ *radical-sqrt*
and *ordA*: (*ordinate A*) ∈ *radical-sqrt*
and *absB*: (*abscissa B*) ∈ *radical-sqrt*
and *ordB*: (*ordinate B*) ∈ *radical-sqrt*
and *absC*: (*abscissa C*) ∈ *radical-sqrt*
and *ordC*: (*ordinate C*) ∈ *radical-sqrt*
and *absD*: (*abscissa D*) ∈ *radical-sqrt*
and *ordD*: (*ordinate D*) ∈ *radical-sqrt*
and *notParallel*: ¬ (*parallel A B C D*)
and *isIntersec*: *is-intersection X A B C D*
shows (*abscissa X*) ∈ *radical-sqrt* ∧ (*ordinate X*) ∈ *radical-sqrt*
proof−
have *l2*: (*abscissa A* − *abscissa X*) ∗ (*ordinate A* − *ordinate B*) = (*ordinate A* − *ordinate X*) ∗ (*abscissa A* − *abscissa B*) ∧ (*abscissa C* − *abscissa X*) ∗ (*ordinate C* − *ordinate D*) = (*ordinate C* − *ordinate X*) ∗ (*abscissa C* − *abscissa D*)
using *isIntersec is-intersection-def collinear-def parallel-def*
by *auto*
hence *l4*: (− (*ordinate A* − *ordinate B*)) ∗ *abscissa X* + (*abscissa A* − *abscissa B*) ∗ *ordinate X* = (− *abscissa A* ∗ (*ordinate A* − *ordinate B*) + *ordinate A* ∗ (*abscissa A* − *abscissa B*))
by (*simp add*: *algebra-simps*)
have *l6*: (− (*ordinate C* − *ordinate D*)) ∗ *abscissa X* + (*abscissa C* − *abscissa D*) ∗ *ordinate X* = (− *abscissa C* ∗ (*ordinate C* − *ordinate D*) + *ordinate C* ∗ (*abscissa C* − *abscissa D*))
using *l2* **by** (*simp add*: *algebra-simps*)
have *RS1*: (− (*ordinate A* − *ordinate B*)) ∈ *radical-sqrt*
by (*metis ordA ordB minus-diff-eq radical-sqrt-rule-subtraction*)
have *RS2*: (*abscissa A* − *abscissa B*) ∈ *radical-sqrt*
by (*metis absA absB radical-sqrt-rule-subtraction*)
have *RS3*: (− *abscissa A* ∗ (*ordinate A* − *ordinate B*) + *ordinate A* ∗ (*abscissa A* − *abscissa B*)) ∈ *radical-sqrt*
using *absA ordA ordB absB radical-sqrt.Times radical-sqrt-rule-subtraction* **by** *force*
have *RS4*: (− (*ordinate C* − *ordinate D*)) ∈ *radical-sqrt*
by (*metis ordC ordD minus-diff-eq radical-sqrt-rule-subtraction*)
have *RS5*: (*abscissa C* − *abscissa D*) ∈ *radical-sqrt*
by (*metis absC absD radical-sqrt-rule-subtraction*)
have *RS6*: (− *abscissa C* ∗ (*ordinate C* − *ordinate D*) + *ordinate C* ∗ (*abscissa C* − *abscissa D*)) ∈ *radical-sqrt*
using *absC ordC absD ordD*
by (*simp add*: *radical-sqrt.Times radical-sqrt-rule-subtraction*)
have (− (*ordinate A* − *ordinate B*)) ∗ (*abscissa C* − *abscissa D*) ≠ (*abscissa A* − *abscissa B*) ∗ (− (*ordinate C* − *ordinate D*))
using *notParallel parallel-def*
by (*simp add*: *algebra-simps*)
thus *?thesis*
using *radical-sqrt-simultaneous-linear-equation* [*of* − (*ordinate A* − *ordinate B*) (*abscissa A* − *abscissa B*)
− *abscissa A* ∗ (*ordinate A* − *ordinate B*) + *ordinate A* ∗ (*abscissa A*

− *abscissa B*) − (*ordinate C* − *ordinate D*)
abscissa C − *abscissa D* − *abscissa C* ∗ (*ordinate C* − *ordinate D*) + *ordinate C* ∗ (*abscissa C* − *abscissa D*)
abscissa X ordinate X]
assms l4 RS1 RS2 RS3 RS4 RS5 RS6 l6
by *simp*
qed

lemma *radical-sqrt-line-circle-intersection*:
assumes *absA*: (*abscissa A*) ∈ *radical-sqrt* **and** *ordA*: (*ordinate A*) ∈ *radical-sqrt*
and *absB*: (*abscissa B*) ∈ *radical-sqrt* **and** *ordB*: (*ordinate B*) ∈ *radical-sqrt*
and *absC*: (*abscissa C*) ∈ *radical-sqrt* **and** *ordC*: (*ordinate C*) ∈ *radical-sqrt*
and *absD*: (*abscissa D*) ∈ *radical-sqrt* **and** *ordD*: (*ordinate D*) ∈ *radical-sqrt*
and *absE*: (*abscissa E*) ∈ *radical-sqrt* **and** *ordE*: (*ordinate E*) ∈ *radical-sqrt*
and *notEqual*: *A* ≠ *B*
and *colin*: *collinear A X B*
and *eqDist*: (*distance C X* = *distance D E*)
shows (*abscissa X*) ∈ *radical-sqrt* ∧ (*ordinate X*) ∈ *radical-sqrt*
proof−
have *RS1*: (− (*ordinate A* − *ordinate B*)) ∈ *radical-sqrt*
by (*metis ordA ordB minus-diff-eq radical-sqrt-rule-subtraction*)
have *RS2*: (*abscissa A* − *abscissa B*) ∈ *radical-sqrt*
by (*metis absA absB radical-sqrt-rule-subtraction*)
have *RS3*: (− *abscissa A* ∗ (*ordinate A* − *ordinate B*) + *ordinate A* ∗ (*abscissa A* − *abscissa B*)) ∈ *radical-sqrt*
by (*simp add*: *absA ordA ordB radical-sqrt.Times radical-sqrt-rule-subtraction RS2*)
have *RS4*: (*abscissa D* − *abscissa E*)ˆ*2* + (*ordinate D* − *ordinate E*)ˆ*2* ∈ *radical-sqrt*
by (*simp add*: *absD absE ordD ordE power2-eq-square radical-sqrt.Plus radical-sqrt.Times radical-sqrt-rule-subtraction*)
have ¬ (− (*ordinate A* − *ordinate B*) = *0* ∧ (*abscissa A* − *abscissa B*) = *0* ∧ (− *abscissa A* ∗ (*ordinate A* − *ordinate B*) + *ordinate A* ∗ (*abscissa A* − *abscissa B*)) = *0*)
using *notEqual point-eqI* **by** *force*
moreover
have (− (*ordinate A* − *ordinate B*)) ∗ *abscissa X* + (*abscissa A* − *abscissa B*) ∗ *ordinate X* = (− *abscissa A* ∗ (*ordinate A* − *ordinate B*) + *ordinate A* ∗ (*abscissa A* − *abscissa B*))
using *colin* **unfolding** *collinear-def parallel-def* **by** *algebra*
moreover
have *sqrt* ((*abscissa X* − *abscissa C*)ˆ*2* + (*ordinate X* − *ordinate C*) ˆ*2*) = *sqrt* ((*abscissa D* − *abscissa E*)ˆ*2* + (*ordinate D* − *ordinate E*) ˆ*2*)
using *eqDist distance-def* **by** (*metis dist-commute point-dist point-surj*)
hence (*abscissa X* − *abscissa C*)ˆ*2* + (*ordinate X* − *ordinate C*) ˆ*2* = (*abscissa D* − *abscissa E*)ˆ*2* + (*ordinate D* − *ordinate E*)ˆ*2*
by *auto*
ultimately show *?thesis*

using *radical-sqrt-simultaneous-linear-quadratic*
[*of abscissa C ordinate C*
(*abscissa D* − *abscissa E*)^2 + (*ordinate D* − *ordinate E*)^2
− (*ordinate A* − *ordinate B*) *abscissa A* − *abscissa B*
− *abscissa A* ∗ (*ordinate A* − *ordinate B*) + *ordinate A* ∗ (*abscissa A* − *abscissa B*)
abscissa X ordinate X]
absC ordC RS1 RS2 RS3 RS4 **by** *fastforce*
qed

lemma *radical-sqrt-circle-circle-intersection*:
assumes *absA*: (*abscissa A*) ∈ *radical-sqrt* **and** *ordA*: (*ordinate A*) ∈ *radical-sqrt*
and *absB*: (*abscissa B*) ∈ *radical-sqrt* **and** *ordB*: (*ordinate B*) ∈ *radical-sqrt*
and *absC*: (*abscissa C*) ∈ *radical-sqrt* **and** *ordC*: (*ordinate C*) ∈ *radical-sqrt*
and *absD*: (*abscissa D*) ∈ *radical-sqrt* **and** *ordD*: (*ordinate D*) ∈ *radical-sqrt*
and *absE*: (*abscissa E*) ∈ *radical-sqrt* **and** *ordE*: (*ordinate E*) ∈ *radical-sqrt*
and *absF*: (*abscissa F*) ∈ *radical-sqrt* **and** *ordF*: (*ordinate F*) ∈ *radical-sqrt*
and *eqDist0*: *distance A X* = *distance B C*
and *eqDist1*: *distance D X* = *distance E F*
and *notEqual*: ¬ (*A* = *D* ∧ *distance B C* = *distance E F*)
shows (*abscissa X*) ∈ *radical-sqrt* ∧ (*ordinate X*) ∈ *radical-sqrt*
proof−
have *sqrt* ((*abscissa X* − *abscissa A*)^2 + (*ordinate X* − *ordinate A*) ^2) = *sqrt* ((*abscissa B* − *abscissa C*)^2 + (*ordinate B* − *ordinate C*) ^2)
by (*metis dist-commute distance-def eqDist0 point-dist point-surj*)
hence (*sqrt* ((*abscissa X* − *abscissa A*)^2 + (*ordinate X* − *ordinate A*) ^2))^2 = (*sqrt* ((*abscissa B* − *abscissa C*)^2 + (*ordinate B* − *ordinate C*)^2)) ^2
by (*auto simp*: *power-def*)
hence *XA*: (*abscissa X* − *abscissa A*)^2 + (*ordinate X* − *ordinate A*) ^2 = (*abscissa B* − *abscissa C*)^2 + (*ordinate B* − *ordinate C*)^2
by *auto*
have *sqrt* ((*abscissa X* − *abscissa D*)^2 + (*ordinate X* − *ordinate D*) ^2) = *sqrt* ((*abscissa E* − *abscissa F*)^2 + (*ordinate E* − *ordinate F*) ^2)
by (*metis dist-commute distance-def eqDist1 point-dist point-surj*)
hence *XD*: (*abscissa X* − *abscissa D*)^2 + (*ordinate X* − *ordinate D*) ^2 = (*abscissa E* − *abscissa F*)^2 + (*ordinate E* − *ordinate F*)^2
by *auto*
have ∗: ¬ (*abscissa A* = *abscissa D* ∧ *ordinate A* = *ordinate D*)
by (*metis point-eq-iff notEqual eqDist0 eqDist1*)
have (*abscissa B* − *abscissa C*) ∈ *radical-sqrt*
by (*metis absB absC radical-sqrt-rule-subtraction*)
hence *RS1*: ((*abscissa B* − *abscissa C*)^2) ∈ *radical-sqrt*
by (*auto intro*: *radical-sqrt.intros simp add*: *power2-eq-square*)
have (*ordinate B* − *ordinate C*) ∈ *radical-sqrt*
by (*metis ordB ordC radical-sqrt-rule-subtraction*)
hence (*ordinate B* − *ordinate C*)^2 ∈ *radical-sqrt*
by (*auto intro*: *radical-sqrt.intros simp add*: *power2-eq-square*)
hence ∗∗: ((*abscissa B* − *abscissa C*)^2 + (*ordinate B* − *ordinate C*)^2) ∈

radical-sqrt
  by (*metis radical-sqrt.intros*(*4*) *RS1*)
 have (*abscissa E* − *abscissa F*) ∈ *radical-sqrt*
  by (*metis absE absF radical-sqrt-rule-subtraction*)
 hence ((*abscissa E* − *abscissa F*)^*2*) ∈ *radical-sqrt*
  by (*auto intro*: *radical-sqrt.intros simp add*: *power2-eq-square*)
 moreover
 have (*ordinate E* − *ordinate F*) ∈ *radical-sqrt*
  by (*metis ordE ordF radical-sqrt-rule-subtraction*)
 hence (*ordinate E* − *ordinate F*)^*2* ∈ *radical-sqrt*
  by (*auto intro*: *radical-sqrt.intros simp add*: *power2-eq-square*)
 ultimately have ((*abscissa E* − *abscissa F*)^*2* + (*ordinate E* − *ordinate F*)^*2*) ∈ *radical-sqrt*
  by (*metis radical-sqrt.intros*(*4*))
 thus *?thesis*
  using *radical-sqrt-simultaneous-quadratic-quadratic*
    [*of abscissa A ordinate A* (*abscissa B* − *abscissa C*)^*2* + (*ordinate B* − *ordinate C*)^*2*
     abscissa D ordinate D (*abscissa E* − *abscissa F*)^*2* + (*ordinate E* − *ordinate F*)^*2*
     abscissa X ordinate X]
   absA ordA absD ordD XA XD ∗ ∗∗
  by *auto*
qed

2.10 Definition of the set of contructible points

inductive-set *constructible* :: *point set*
 where
 (*M* ∈ *points* ∧ (*abscissa M*) ∈ ℚ ∧ (*ordinate M*) ∈ ℚ) ⟹ *M* ∈ *constructible*|
 (*A* ∈ *constructible* ∧ *B* ∈ *constructible* ∧ *C* ∈ *constructible* ∧ *D* ∈ *constructible* ∧ ¬ *parallel A B C D* ∧ *is-intersection M A B C D*) ⟹ *M* ∈ *constructible*|
 (*A* ∈ *constructible* ∧ *B* ∈ *constructible* ∧ *C* ∈ *constructible* ∧ *D* ∈ *constructible* ∧ *E* ∈ *constructible* ∧ ¬ *A* = *B* ∧ *collinear A M B* ∧ *distance C M* = *distance D E*) ⟹ *M* ∈ *constructible*|
 (*A* ∈ *constructible* ∧ *B* ∈ *constructible* ∧ *C* ∈ *constructible* ∧ *D* ∈ *constructible* ∧ *E* ∈ *constructible* ∧ *F* ∈ *constructible* ∧ ¬ (*A* = *D* ∧ *distance B C* = *distance E F*) ∧ *distance A M* = *distance B C* ∧ *distance D M* = *distance E F*) ⟹ *M* ∈ *constructible*

2.11 An important property about constructible points: their coordinates are radicals

lemma *constructible-radical-sqrt*:
 assumes *M* ∈ *constructible*
 shows (*abscissa M*) ∈ *radical-sqrt* ∧ (*ordinate M*) ∈ *radical-sqrt*
 using *assms*
proof (*induction rule*: *constructible.induct*)
 case (*1 M*)

```
  then show ?case by (metis radical-sqrt.intros(1))
next
  case (2 A B C D M)
  then show ?case by (metis radical-sqrt-line-line-intersection)
next
  case (3 A B C D E M)
  then show ?case by (metis radical-sqrt-line-circle-intersection)
next
  case (4 A B C D E F M)
  then show ?case by (metis radical-sqrt-circle-circle-intersection)
qed
```

2.12 Proving the impossibility of duplicating the cube

```
lemma impossibility-of-doubling-the-cube-lemma:
  assumes x: x ∈ radical-sqrt
  and x-eqn: x^3 = 2
  shows False
proof−
  have ∃ y ∈ ℚ. y^3 + 0 ∗ y^2 + 0 ∗ y + (− 2) = (0::real)
    using x x-eqn cubic-root-radical-sqrt-rational [of 0 0 − 2]
    by auto
  then obtain y::real where hypsy: y ∈ ℚ y^3 = 2
    by auto
  then obtain r where hypsr: y = of-rat r
    using Rats-cases by blast
  hence ∃! p. r = Fract (fst p) (snd p) ∧ snd p > 0 ∧ coprime (fst p) (snd p)
    by (metis quotient-of-unique)
  then obtain p q where hypsp: r = Fract p q q > 0 coprime p q
    by auto
  have r^3 = 2
    using hypsr hypsy by (metis of-rat-eq-iff of-rat-numeral-eq of-rat-power)
  moreover have r^3  = Fract (p^3) (q^3)
    using hypsp by (simp add: power3-eq-cube)
  ultimately have Fract (p ^ 3) (q ^ 3) = 2
    by auto
  hence Fract (p ^ 3) (q ^ 3) = Fract 2 1
    by (metis rat-number-expand(3))
  hence l12: p ^ 3 = q ^ 3 ∗ 2 using hypsp
    by (simp add: eq-rat)
  hence even (p ^ 3)
    by (auto intro: dvdI)
  then have even p
    by auto
  then have 8 dvd p ^ 3
    by (auto simp: dvd-def power-def)
  then have 8 dvd q ^ 3 ∗ 2
    using l12 by auto
  then have even (q ^ 3)
```

    by (*auto simp*: *dvd-def*)
  then have *even q*
    by *auto*
  with ‹*even p*› **have** *2 dvd gcd p q*
    by (*rule gcd-greatest*)
  with ‹*coprime p q*› **show** *False* **by** *simp*
qed

theorem *impossibility-of-doubling-the-cube*:
  $x\hat{\ }3 = 2 \Longrightarrow (Point\ x\ 0) \notin constructible$
 by (*metis abscissa.simps constructible-radical-sqrt impossibility-of-doubling-the-cube-lemma*)

2.13 Proving the impossibility of trisecting an angle

lemma *impossibility-of-trisecting-pi-over-3-lemma*:
  assumes *x*: $x \in radical\text{-}sqrt$
  and *x-eqn*: $x\hat{\ }3 - 3 * x - 1 = 0$
  shows *False*
proof−
  have $\exists x \in \mathbb{Q}.\ x\hat{\ }3 + (-\ 3) * x = (1{::}real)$
    using *x-eqn cubic-root-radical-sqrt-rational* [*of 0* − *3* − *1*] *x*
    by *force*
  then obtain *y* :: *real* **where** *hypsy*: $y \in \mathbb{Q} \wedge y\ \hat{}\ 3 - 3 * y = 1$ **by** *auto*
  then obtain *r* **where** *hypsr*: *y* = *of-rat r*
    by (*metis Rats-cases*)
  then obtain *p* **where** *hypsp*: *r* = *Fract* (*fst p*) (*snd p*) ∧ *snd p* > *0* ∧ *coprime*
(*fst p*) (*snd p*)
      using *quotient-of-unique hypsy*
      by *blast*
  have *r3eq*: $r\hat{\ }3 - 3 * r = 1$
    using *hypsy hypsr*
   by (*metis* (*mono-tags*, *opaque-lifting*) *of-rat-diff of-rat-eq-1-iff of-rat-mult of-rat-numeral-eq*
power3-eq-cube)
  have ∗: (*snd p*) ^*3* > *0* ∧ *coprime* ((*fst p*)^*3*) ((*snd p*)^*3*)
    using *hypsp* **by** *simp*
  have *r*^*3* = *Fract* ((*fst p*)^*3*) ((*snd p*)^*3*)
    by (*metis* (*no-types*) *mult-rat power3-eq-cube hypsp*)
  then have *Fract* ((*fst p*)^*3*) ((*snd p*)^*3*) − (*Fract* (*3* ∗ (*fst p*)) (*snd p*)) = *1*
    using *r3eq hypsp*
    by (*simp add*: *Fract-of-int-quotient*)
  then have *l10*: *Fract* ((*fst p*)^*3*) ((*snd p*)^*3*) − *Fract* (*3* ∗ (*fst p*) ∗ (*snd p*)^*2*)
((*snd p*)^*3*) = *1*
    using *hypsp*
    by (*simp add*: *power-def algebra-simps Fract-of-int-quotient*)
  have *Fract* ((*fst p*)^*3* − (*3* ∗ (*fst p*) ∗ (*snd p*)^*2*)) ((*snd p*)^*3*) =
        Fract (((*fst p*)^*3* − (*3* ∗ (*fst p*) ∗ (*snd p*)^*2*))∗(*snd p*)^*3*) (((*snd p*)^*3*) ∗
(*snd p*)^*3*)
    using ∗ *eq-rat* **by** *auto*

```
  also have ... = Fract 1 1
    using One-rat-def int-distrib(3) l10 * by auto
  finally have (fst p)^3 - 3 * (fst p) * (snd p)^2 = (snd p)^3 using hypsp
    by (simp add: eq-rat)
  hence (fst p) * ((fst p)^2 - 3 * (snd p) ^2) = (snd p)^3
      (snd p) * ((snd p)^2 + 3 * (fst p) * (snd p)) = (fst p) ^3
    by (auto simp: power-def algebra-simps)
  hence  fst p ^ 3 = snd p * ((snd p)² + 3 * fst p * snd p)
       snd p ^ 3 = fst p * ((fst p)² - 3 * (snd p)²)
    by auto
  hence (fst p) dvd ((snd p)^3)  (snd p) dvd ((fst p)^3)
    by (auto simp: dvd-def)
  moreover have coprime (fst p) (snd p ^ 3) coprime (fst p ^ 3) (snd p)
    using hypsp by auto
  ultimately have is-unit (fst p) is-unit (snd p)
    using coprime-common-divisor [of fst p snd p ^ 3 fst p]
      coprime-common-divisor [of fst p ^ 3 snd p snd p]
    by auto
  with hypsp have fst p = 1 ∨ fst p = - 1 snd p = 1
    by auto
  with hypsp have r = 1 | r = - 1
    by (auto simp: rat-number-collapse)
  with r3eq show False
    by (auto simp: power-def algebra-simps)
qed


theorem impossibility-of-trisecting-angle-pi-over-3:
  Point (cos (pi / 9)) 0 ∉ constructible
proof-
  have cos (3 *(pi/9)) = 4 * (cos (pi/9))^3 - 3 * cos (pi/9)
    using cos-treble-cos [of pi / 9]
    by auto
  hence 1/2 = 4 * (cos (pi/9))^3 - 3 * cos (pi/9)
    by (simp add: cos-60)
  hence 8 * (cos (pi/9))^3 - 6 * cos (pi/9) - 1 = 0
    by (simp add: algebra-simps)
  hence (2 * cos (pi / 9)) ^3 - 3 * (2 * cos (pi / 9)) - 1 = 0
    by (simp add: algebra-simps power-def)
  hence ¬ (2 * cos (pi / 9)) ∈ radical-sqrt
    by (metis impossibility-of-trisecting-pi-over-3-lemma)
  hence ¬ (cos (pi / 9)) ∈ radical-sqrt
    using radical-sqrt.Plus by fastforce
  thus ?thesis
    by (metis abscissa.simps constructible-radical-sqrt)
qed

end
```

References

[Car81] J. C. Carrega. *Théorie des corps : la règle et le compas*. Hermann, 1981.