Summary for the Lean Project “Fermat’s Theorem on the Sums of Two Squares”

1 Overview of Mathematical Proof

Theorem 1.1. *$p \in \mathbb{N}$ is a prime, then*

$$\exists x, y \in \mathbb{N} \quad s.t. \quad p = x^2 + y^2 \Leftrightarrow p = 2 \;\; or \;\; p \equiv 1[\mathrm{mod}\, 4]$$

Proof.

“$\Rightarrow$”

via “Lemma-1.2”, the question can be divided into 4 cases,

leaving some calculation under mod equality

“$\Leftarrow$”

by cases,

when $p = 2$, it is easy, just setting $x = 1, y = 1$

when $p \equiv 1[\mathrm{mod}\, 4]$, we consider $p \in \mathbb{Z}[i]$

since p is still not a unit in $\mathbb{Z}[i]$, there exists a maximal ideal $\mathfrak{m}$ of $\mathbb{Z}[i]$ by the existence of max ideals

in addition, we have $\mathbb{Z}[i]$ is a PID, so $\mathfrak{m} = (\pi)$ for some $\pi \in \mathbb{Z}[i]$

we can write $\pi = x + yi$, claim that $p = x^2 + y^2$

in fact, this can be derived via analysing norm of p and π

(here, norm means the norm of Gaussian integers, $\|\cdot\|: \mathbb{Z}[i] \to \mathbb{N}, x + yi \mapsto x^2 + y^2$

beginning with $\pi | p$, we deduce $\|\pi\| | \|p\|$. moreover, $\|p\| = p^2$

then it implies that $\|\pi\| = 1, p$ or p^2

here we can prove what we want in the case “$\|\pi\| = p$”, which means we only need to show $\|\pi\| = 1$ and $\|\pi\| = p^2$ are impossible

when $\|\pi\| = 1$, it leads to a contradiction against π generates a max ideal via “Lemma-1.3”

when $\|\pi\| = p^2$, there exists a unit v s.t. $p = v \cdot \pi$, which means $(\pi) = (p)$

since (p) is a max ideal now, $\mathbb{Z}[i] / (p)$ is a field

it is inconvenient to analyse $\mathbb{Z}[i] / (p)$, but it is ring isomorphic to $\mathbb{F}_p[X] / (X^2 + 1)$ via “Lemma-1.4” and $\mathbb{F}_p[X]/(X^2+1)$ is not a field via "Lemma-1.5"

this contradiction completes the proof □

Lemma 1.2. *$x \in \mathbb{N}$, then $x^2 \equiv 0[\mathrm{mod}\, 4]$ or $x^2 \equiv 1[\mathrm{mod}\, 4]$*

Lemma 1.3. *$x \in \mathbb{Z}[i]$, then $\|x\| = 1$ iff x is a unit*

Lemma 1.4. *p is a prime, then $\mathbb{Z}[i]/(p) \cong \mathbb{F}_p[X]/(X^2+1)$*

Proof.

$$\begin{aligned} \mathbb{Z}[i]/(p) &\cong \mathbb{Z}/(p)[X]/(X^2+1) \\ &\cong \mathbb{F}_p[X]/(X^2+1) \end{aligned}$$

□

Lemma 1.5. *p is a prime s.t. $p \equiv 1[\mathrm{mod}\, 4]$, then $\mathbb{F}_p[X]/(X^2+1)$ is not a field*

Proof.

we only need to prove $(X^2+1) \leqslant \mathbb{F}_p[X]$ isn't a prime ideal, which means there exists $x, y \in \mathbb{F}_p[X]$ s.t. $xy \in (X^2+1)$ but neither x nor y is in (X^2+1)

here, we notice that $\mathbb{F}_p^\times$(:=the set of units in $\mathbb{F}_p$) is a cyclic group with cardinality $p-1$, let ξ represent its generator

since $p \equiv 1[\mathrm{mod}\, 4]$, $n = (p-1)/4 \in \mathbb{N}$, we can consider ξ^n

we have the following computation results :

$$\xi^{2n} = -1$$

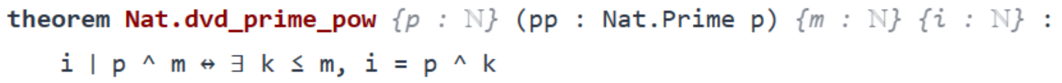
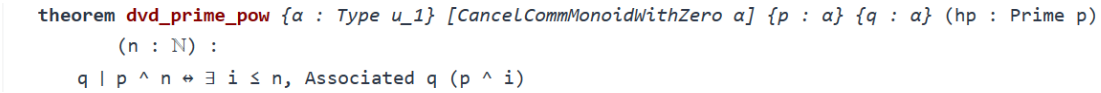
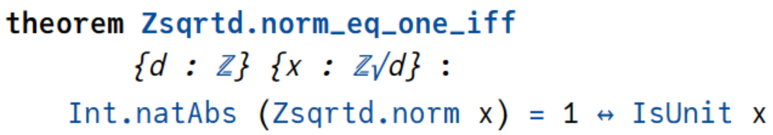
in fact, ξ^{2n} is a solution to $x^2 = 1$ in $\mathbb{F}_p^\times$ but it should not equal to 1 thus $(X+\xi^n)(X-\xi^n)=X^2-\xi^{2n}=X^2+1\in(X^2+1)$

but X^2+1 should not divide $X+\xi^n$ or $X-\xi^n$ since the degree of the former is greater than that of the latter □

2 Explanations for Some Parts in the code

2.1 Explanation 1

Object	to derive $\lVert\pi\rVert \mid p^2$, so as to get $\exists i\leqslant 2$ s.t. $\lVert\pi\rVert=p^i$, which relies on the fact that p is a prime
Act	I apply 'Nat.dvd_prime_pow' and use functions 'Abs.abs' and 'Int.natAbs' converting prerequisites of Type '$\mathbb{Z}[i]$' to Type 'ℕ'

Why?	in fact, to achieve the object, there exist two theorems :
	
	`theorem Nat.dvd_prime_pow {p : ℕ} (pp : Nat.Prime p) {m : ℕ} {i : ℕ} :`
	`i ∣ p ^ m ↔ ∃ k ≤ m, i = p ^ k`
	
	`theorem dvd_prime_pow {α : Type u_1} [CancelCommMonoidWithZero α] {p : α} {q : α} (hp : Prime p)`
	`(n : ℕ) :`
	`q ∣ p ^ n ↔ ∃ i ≤ n, Associated q (p ^ i)`
	reasons why I choosed the former are following:
	1. I suggested it could be inconvenient to cope with 'Associated'
	2. Comparison to conversion from ℤ to ℕ, it is easier to face coercion from ℕ to ℤ.
	3. I found a theorem, which implies that working with 'Zsqrtd.norm' in ℕ is common
	
	`theorem Zsqrtd.norm_eq_one_iff`
	`{d : ℤ} {x : ℤ√d} :`
	`Int.natAbs (Zsqrtd.norm x) = 1 ↔ IsUnit x`

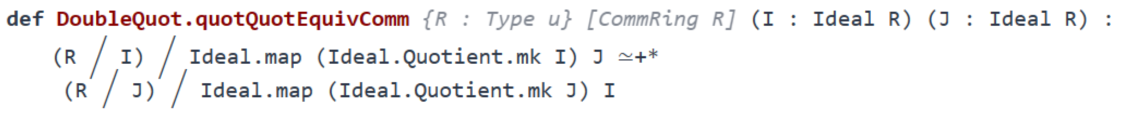
Detail	Prerequisites : Zsqrtd.norm v * Zsqrtd.norm π = ↑p ^ 2 1. Take the absolute value of both sides : '|Zsqrtd.norm v| * |Zsqrtd.norm π| = |↑p ^ 2|' 2. Convert the equation to Type 'ℕ' via 'Int.abs_eq_natAbs (a : ℤ) : |a| = ↑(Int.natAbs a)' and 'norm_cast' : 'Int.natAbs (Zsqrtd.norm v) * Int.natAbs (Zsqrtd.norm π) = p ^ 2'here, 'Int.natAbs' is a function $\mathbb{Z}\to\mathbb{N}$

as this instance shows, in my project, I frequently met requirement of multiple converting items between two types. The reason is that in my proof, some concepts and tools are stated with respect to one type, others are stated with respect to another type.

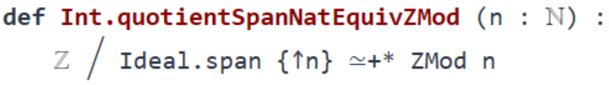
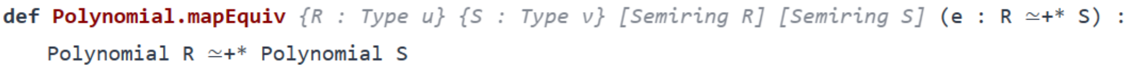
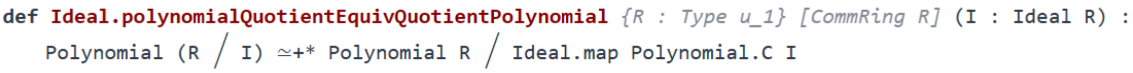
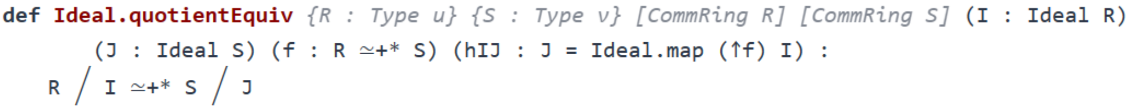
in this case, prime number and the conclusion 'Nat.dvd_prime_pow' are under documents for native numbers, while Gaussian integers, Zsqrtd.norm and calculation of division are defined via integers. If we want to convert a native number to an integer, we can use coercion directly. But it is tricky to work conversely, from a larger type to a smaller one. Thankfully, a function 'Int.natAbs' can be appled here.

2.2 Explanation 2

Object	to derive $\mathbb{F}_p[X]/(X^2+1) \cong \mathbb{Z}[i]/(p)$

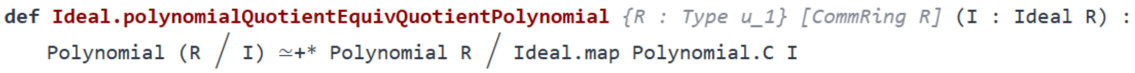
Step 1	First, I find the following theorem :
	
	`def DoubleQuot.quotQuotEquivComm {R : Type u} [CommRing R] (I : Ideal R) (J : Ideal R) :`
	`    (R ⧸ I) ⧸ Ideal.map (Ideal.Quotient.mk I) J ≃+*`
	`     (R ⧸ J) ⧸ Ideal.map (Ideal.Quotient.mk J) I`
	it can be used to deal with $(\mathbb{Z}[X]/(p))/(X^2+1) \cong (\mathbb{Z}[X]/(X^2+1))/(p)$
	thus I plan to divide the main goal into three subgoals :
	• $\mathbb{F}_p[X]/(X^2+1) \cong (\mathbb{Z}[X]/(p))/(X^2+1)$
	• $(\mathbb{Z}[X]/(p))/(X^2+1) \cong (\mathbb{Z}[X]/(X^2+1))/(p)$
	• $(\mathbb{Z}[X]/(X^2+1))/(p) \cong \mathbb{Z}[i]/(p)$

Step 2	to derive $\mathbb{F}_p[X]/(X^2+1) \cong (\mathbb{Z}[X]/(p))/(X^2+1)$

Attempt 1	I had planned to prove this via two moves at the beginning:
	• $\mathbb{F}_p[X] \cong \mathbb{Z}/(p)[X] \cong \mathbb{Z}[X]/(p)$
	• $\mathbb{F}_p[X]/(X^2+1) \cong (\mathbb{Z}[X]/(p))/(X^2+1)$
	the first move is completed easily by applying
	
	`def Int.quotientSpanNatEquivZMod (n : ℕ) :`
	`ℤ ⧸ Ideal.span {↑n} ≃+* ZMod n`
	
	`def Polynomial.mapEquiv {R : Type u} {S : Type v} [Semiring R] [Semiring S] (e : R ≃+* S) :`
	`Polynomial R ≃+* Polynomial S`
	
	`def Ideal.polynomialQuotientEquivQuotientPolynomial {R : Type u_1} [CommRing R] (I : Ideal R) :`
	`Polynomial (R ⧸ I) ≃+* Polynomial R ⧸ Ideal.map Polynomial.C I`
	but next, to finish the second step, I can only find a related theorem :
	
	`def Ideal.quotientEquiv {R : Type u} {S : Type v} [CommRing R] [CommRing S] (I : Ideal R)`
	`(J : Ideal S) (f : R ≃+* S) (hIJ : J = Ideal.map (↑f) I) :`
	`R ⧸ I ≃+* S ⧸ J`
	to prove the image condition, I had to show $X^2+1 \in \mathbb{F}_p[X]$ corresponds to $\overline{X^2+1} \in \mathbb{Z}[X]/(p)$ exactly. I had no ideal to do this since the RingEquiv is implicit.

Attempt 2	I guessed that it should be possible to prove $\mathbb{F}_p[X]/(X^2+1) \cong \mathbb{Z}/(p)[X]/(X^2+1)$ due to the fact that $\mathbb{F}_p[X] \cong \mathbb{Z}/(p)[X]$ as two polynomial rings. I mean the structure of $\mathbb{Z}/(p)[X]$ is more similar to $\mathbb{F}_p[X]$ than $\mathbb{Z}[X]/(p)$ therefore, the plan was changed to two new moves : • $\mathbb{F}_p[X]/(X^2+1) \cong \mathbb{Z}/(p)[X]/(X^2+1)$ • $\mathbb{Z}/(p)[X]/(X^2+1) \cong (\mathbb{Z}[X]/(p))/(X^2+1)$ I jumped to the second move to begin with

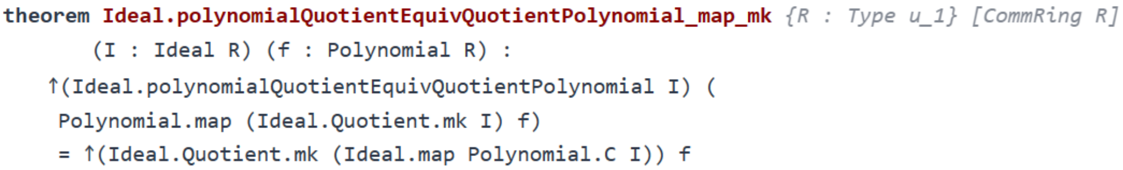
<table>
<tr><td>Step 3</td><td>to derive $\mathbb{Z}/(p)[X]/(X^2+1) \cong \mathbb{Z}[X]/(p)/(X^2+1)$</td></tr>
<tr><td>Act</td><td>
I already proved $\mathbb{Z}/(p)[X] \cong \mathbb{Z}[X]/(p)$ via



<pre>
def Ideal.polynomialQuotientEquivQuotientPolynomial {R : Type u_1} [CommRing R] (I : Ideal R) :
    Polynomial (R / I) ≃+* Polynomial R / Ideal.map Polynomial.C I
</pre>

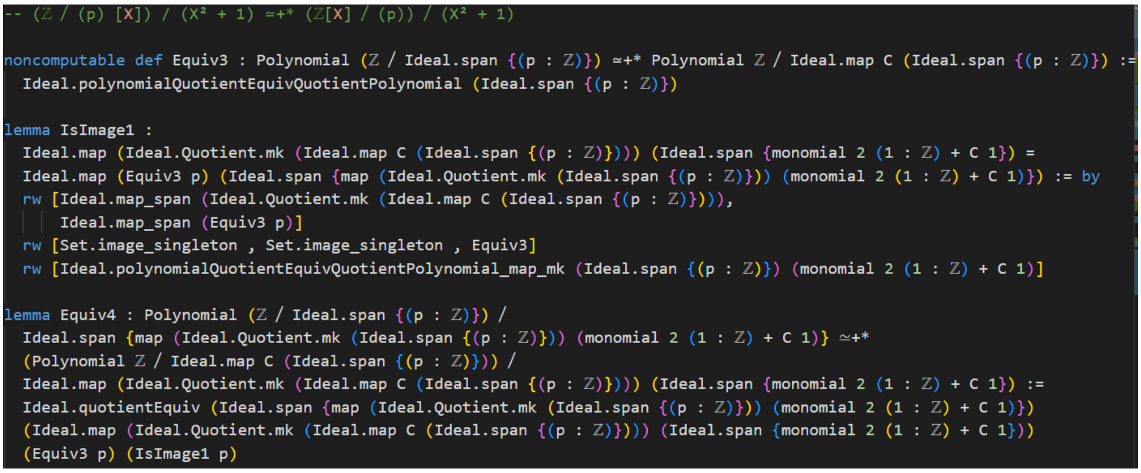
the difficulty to finish this procedure is still the correspondence of ideals in 'Ideal.quotientEquiv'

fortunately, we can apply the following theorem :



<pre>
theorem Ideal.polynomialQuotientEquivQuotientPolynomial_map_mk {R : Type u_1} [CommRing R]
        (I : Ideal R) (f : Polynomial R) :
    ↑(Ideal.polynomialQuotientEquivQuotientPolynomial I) (
     Polynomial.map (Ideal.Quotient.mk I) f)
     = ↑(Ideal.Quotient.mk (Ideal.map Polynomial.C I)) f
</pre>

even it requires me to modify statements enormously



<pre>
-- (ℤ / (p) [X]) / (X² + 1) ≃+* (ℤ[X] / (p)) / (X² + 1)

noncomputable def Equiv3 : Polynomial (ℤ / Ideal.span {(p : ℤ)}) ≃+* Polynomial ℤ / Ideal.map C (Ideal.span {(p : ℤ)}) :=
  Ideal.polynomialQuotientEquivQuotientPolynomial (Ideal.span {(p : ℤ)})

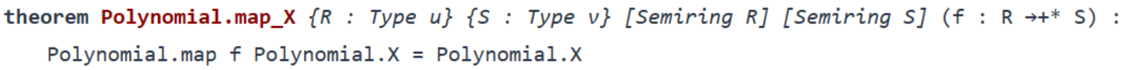
lemma IsImage1 :
  Ideal.map (Ideal.Quotient.mk (Ideal.map C (Ideal.span {(p : ℤ)}))) (Ideal.span {monomial 2 (1 : ℤ) + C 1}) =
  Ideal.map (Equiv3 p) (Ideal.span {map (Ideal.Quotient.mk (Ideal.span {(p : ℤ)})) (monomial 2 (1 : ℤ) + C 1)}) := by
  rw [Ideal.map_span (Ideal.Quotient.mk (Ideal.map C (Ideal.span {(p : ℤ)}))),
      Ideal.map_span (Equiv3 p)]
  rw [Set.image_singleton , Set.image_singleton , Equiv3]
  rw [Ideal.polynomialQuotientEquivQuotientPolynomial_map_mk (Ideal.span {(p : ℤ)}) (monomial 2 (1 : ℤ) + C 1)]

lemma Equiv4 : Polynomial (ℤ / Ideal.span {(p : ℤ)}) /
  Ideal.span {map (Ideal.Quotient.mk (Ideal.span {(p : ℤ)})) (monomial 2 (1 : ℤ) + C 1)} ≃+*
  (Polynomial ℤ / Ideal.map C (Ideal.span {(p : ℤ)})) /
  Ideal.map (Ideal.Quotient.mk (Ideal.map C (Ideal.span {(p : ℤ)}))) (Ideal.span {monomial 2 (1 : ℤ) + C 1}) :=
  Ideal.quotientEquiv (Ideal.span {map (Ideal.Quotient.mk (Ideal.span {(p : ℤ)})) (monomial 2 (1 : ℤ) + C 1)})
  (Ideal.map (Ideal.Quotient.mk (Ideal.map C (Ideal.span {(p : ℤ)}))) (Ideal.span {monomial 2 (1 : ℤ) + C 1}))
  (Equiv3 p) (IsImage1 p)
</pre>
</td></tr>
</table>

<table>
<tr><td>Step 4</td><td>to derive $\mathbb{F}_p[X]/(X^2+1) \cong \mathbb{Z}/(p)[X]/(X^2+1)$</td></tr>
<tr><td>Analysis and Act</td><td>
it is still vital to prove $X^2+1 \in \mathbb{F}_p[X]$ corresponds to $X^2+1 \in \mathbb{Z}/(p)[X]$

'simp' plays an important role here, it reduced the goal to 'X^2 corresponds to X^2'

I was stuck here for a long time. It took me lots of time to find

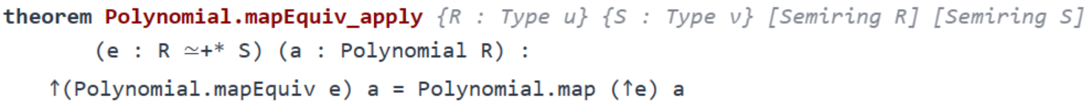


<pre>
theorem Polynomial.map_X {R : Type u} {S : Type v} [Semiring R] [Semiring S] (f : R →+* S) :
    Polynomial.map f Polynomial.X = Polynomial.X
</pre>

but I cannot use it here because f need to be a ring homomorphism, but in Lean, when we make

'RingEquiv' act on an element, it would be coerced to a function

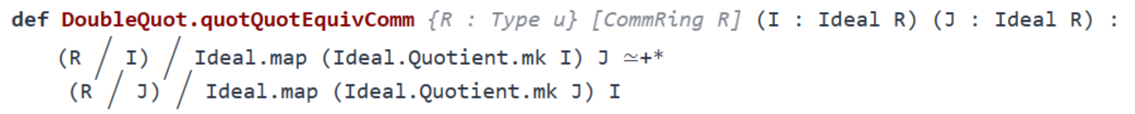
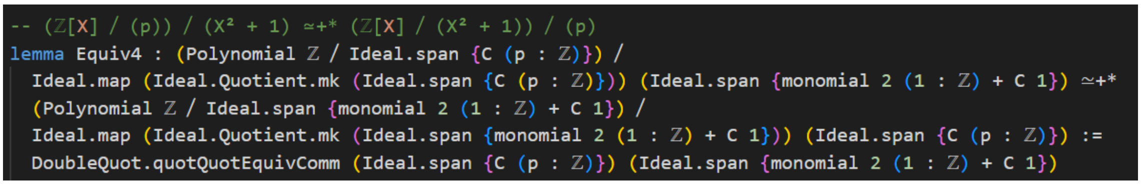
I spent more time seeking in Mathlib, and finally found a proper theorem for cases of 'RingEquiv



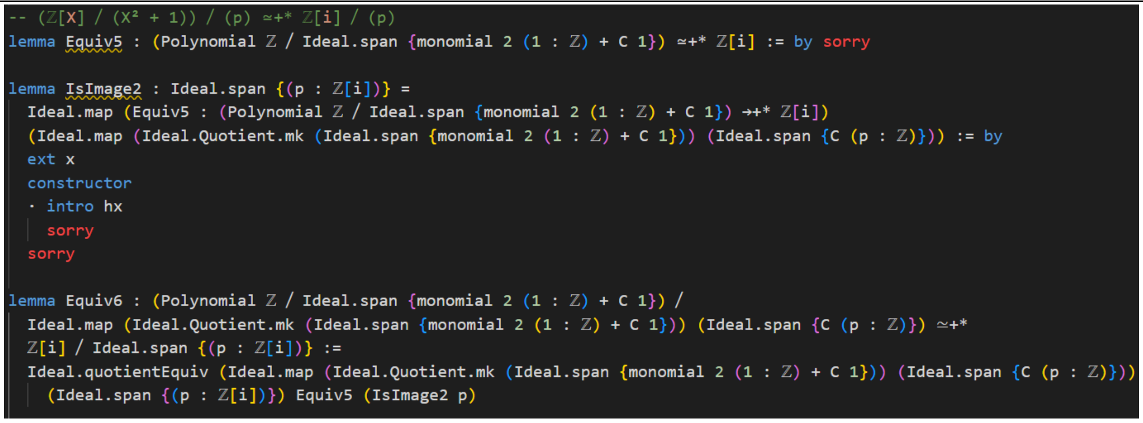
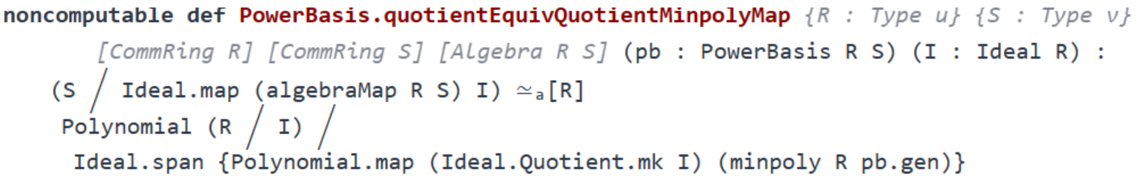
<pre>
theorem Polynomial.mapEquiv_apply {R : Type u} {S : Type v} [Semiring R] [Semiring S]
        (e : R ≃+* S) (a : Polynomial R) :
    ↑(Polynomial.mapEquiv e) a = Polynomial.map (↑e) a
</pre>

'simp' can use this theorem directly, emmmm, when the goal is stated in a proper form.
</td></tr>
</table>

Step 5	to derive $(\mathbb{Z}[X]/(p))/(X^2+1)\cong(\mathbb{Z}[X]/(X^2+1))/(p)$

Act	as I planned, I used the following theorem
	
	`def DoubleQuot.quotQuotEquivComm {R : Type u} [CommRing R] (I : Ideal R) (J : Ideal R) :`
	`    (R ⧸ I) ⧸ Ideal.map (Ideal.Quotient.mk I) J ≃+*`
	`    (R ⧸ J) ⧸ Ideal.map (Ideal.Quotient.mk J) I`
	
	`-- (ℤ[X] / (p)) / (X² + 1) ≃+* (ℤ[X] / (X² + 1)) / (p)`
	`lemma Equiv4 : (Polynomial ℤ ⧸ Ideal.span {C (p : ℤ)}) ⧸`
	`  Ideal.map (Ideal.Quotient.mk (Ideal.span {C (p : ℤ)})) (Ideal.span {monomial 2 (1 : ℤ) + C 1}) ≃+*`
	`  (Polynomial ℤ ⧸ Ideal.span {monomial 2 (1 : ℤ) + C 1}) ⧸`
	`  Ideal.map (Ideal.Quotient.mk (Ideal.span {monomial 2 (1 : ℤ) + C 1})) (Ideal.span {C (p : ℤ)}) :=`
	`  DoubleQuot.quotQuotEquivComm (Ideal.span {C (p : ℤ)}) (Ideal.span {monomial 2 (1 : ℤ) + C 1})`

Step 6	to derive $(\mathbb{Z}[X]/(X^2+1))/(p)\cong\mathbb{Z}[i]/(p)$

Act and Analysis	
	`-- (ℤ[X] / (X² + 1)) / (p) ≃+* ℤ[i] / (p)`
	`lemma Equiv5 : (Polynomial ℤ ⧸ Ideal.span {monomial 2 (1 : ℤ) + C 1}) ≃+* ℤ[i] := by sorry`
	
	`lemma IsImage2 : Ideal.span {(p : ℤ[i])} =`
	`  Ideal.map (Equiv5 : (Polynomial ℤ ⧸ Ideal.span {monomial 2 (1 : ℤ) + C 1}) →+* ℤ[i])`
	`  (Ideal.map (Ideal.Quotient.mk (Ideal.span {monomial 2 (1 : ℤ) + C 1})) (Ideal.span {C (p : ℤ)})) := by`
	`  ext x`
	`  constructor`
	`  · intro hx`
	`    sorry`
	`  sorry`
	
	`lemma Equiv6 : (Polynomial ℤ ⧸ Ideal.span {monomial 2 (1 : ℤ) + C 1}) ⧸`
	`  Ideal.map (Ideal.Quotient.mk (Ideal.span {monomial 2 (1 : ℤ) + C 1})) (Ideal.span {C (p : ℤ)}) ≃+*`
	`  ℤ[i] ⧸ Ideal.span {(p : ℤ[i])} :=`
	`  Ideal.quotientEquiv (Ideal.map (Ideal.Quotient.mk (Ideal.span {monomial 2 (1 : ℤ) + C 1})) (Ideal.span {C (p : ℤ)}))`
	`    (Ideal.span {(p : ℤ[i])}) Equiv5 (IsImage2 p)`
	as showed in the picture of my programming, I planned to do the proof via :
	• prove $\mathbb{Z}[X]/(X^2+1)\cong\mathbb{Z}[i]$ from definition, which means I need to give an explicit expression for the ring isomorphism
	• use the expression to prove that $(p)\subseteq\mathbb{Z}[X]/(X^2+1)$ corresponds to $(p)\subseteq\mathbb{Z}[i]$
	• complete the proof by 'Ideal.quotientEquiv'
	However, when I try to find how to deal with $\mathbb{Z}[i]$, I found the following theorem :
	
	`noncomputable def PowerBasis.quotientEquivQuotientMinpolyMap {R : Type u} {S : Type v}`
	`      [CommRing R] [CommRing S] [Algebra R S] (pb : PowerBasis R S) (I : Ideal R) :`
	`  (S ⧸ Ideal.map (algebraMap R S) I) ≃ₐ[R]`
	`   Polynomial (R ⧸ I) ⧸`
	`    Ideal.span {Polynomial.map (Ideal.Quotient.mk I) (minpoly R pb.gen)}`
	applying this theorem, I can get $\mathbb{Z}/(p)[X]/(X^2+1)\cong\mathbb{Z}[i]/(p)$, which will finish the proof together only with Step 4
	it is meaningless for me to spend two days on other steps !!

3 My Opinions for Lean Programming

a) Formalized proof in Lean is kind of like piecing together a jigsaw. I do enjoy the process of establishing something magnificent step by step.

But the difference is that in Lean, we have to select puzzle pieces from an enormous database — the Mathlib. This leads to two disadvantages : we need to spend much time familiarising ourselves with this database; and when coping with a math question, we need to divert our attention from mathematics, worring whether our argument is easy to achieve in Lean.

As someone who has just been exposed to Lean for a semester and doesn't have intimate knowledge of this language, I suggest that Lean programming can only be an amateur for me. But it only brings unnecassary burden when used to study mathematics.

b) On the other hand, I really appreciate predecessors of Mathlib. I find a number of concepts and conclusions which are not so popular but have been established in Mathlib, like Gaussian integers, the norm of $\mathbb{Z}[i]$ and $\mathbb{F}_p^\times$ is a cyclic group. They considerably reduce my workload.

c) I notice in Mathlib, in order to formalize some abstract concepts, it would use some more complicated definition rather than the common one and sometimes even involve advanced knowledge.

for example, using 'filter' to define 'limit' and applying category theory to establish 'fundamental group'

These make the proof in Lean less comprehensible and lead to more computation and derivation, comparising to normal study.

d) Lean is surprisingly good at some abstract argument while I had expected Lean works as well as Mathlib in some detailed calculation, for instance, symbolics. It is unnecessary to programme all algorithms again. I hope there will be some plug-in calling Mathlib to finish these issues.

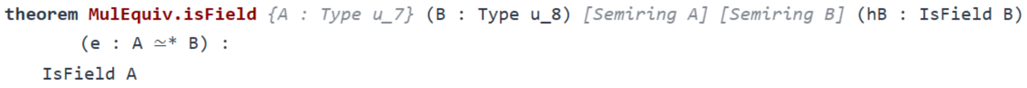
e) however, as to typing math formulas, I see one advantage of Lean, comparising to Latex. It realizing "what you see is what you get".

in Latex, when editing formulas, we program in one window and see results in another window. This separation is inconvenient, especially when we edit a sophisticated formula. We cannot see the result until we finish all programming correclty.

That's why these two years, I replace Latex with Texmacs. I am surprised to see Lean can achieve this function and use simpler codes to express some notations , for example : '\mathbb{Z}' in comparison with '\Z'

f) I also note something I hold in esteem in Mathlib.

Please allow me to demonstrate another example :



```
theorem MulEquiv.isField {A : Type u_7} (B : Type u_8) [Semiring A] [Semiring B] (hB : IsField B)
    (e : A ≃* B) :
  IsField A
```

I want a theorem of 'when two rings are isomorphic, then they are a field at the same time'

But via this theorem, I don't even need a ring isomorphism, I can only show two rings are equivalent via bijection and this bijection preserves multiplication.

I see that the group who build the Mathlib, trying best to weaken prerequisites in theorems. This is what I admire.

However, in this case, condition 'MulEquiv' is not better than 'RingEquiv'. Rings I discussed involve quotient and polynomial, which can not be defined to a 'Mul' over some set. As a consequence, I still need to prove a 'RingEquiv' and let it imply a 'MulEquiv'

g) as for brevity of proof, I admit that Lean does force me to simplify my proof as possible as I can. On the one hand, I don’t want to have too long argument to formalize. On the other hand, if I try to show an implication with a rather stronger prerequisite, Lean often doesn’t accept it without more details even it is totally okay for humanbeing.

4 Suggestions to this Practical Module

a) it might be helpful if lectures focus on knowledge with respect to Lean programming rather than mathematics.

I would expect to be taught ’what is Fact’ , ’the difference between Finite and Fintype’ , strategies of stating and naming a theorem and so on.

After being introduced common sense of Lean programming, we are able to read mathematics in Lean by ourselves.

b) I can see the importance of conventions in Lean programming.

for instant, when we try to state the ideal $(X^2+1)\leqslant(\mathbb{Z}/(p))[X]$, there are two choices

```
Ideal.span {monomial 2 (1 : ℤ / Ideal.span {(p : ℤ)}) + C (1 : ℤ /
Ideal.span {(p : ℤ)})}
```

```
Ideal.span {map (Ideal.Quotient.mk (Ideal.span {(p : ℤ)})) (monomial 2 (1
: ℤ) + C 1)}
```

obviously, the former is more readable. But in Mathlib, most of related theorems are stated as the latter, which means it is more convenient to use it and seek for theorems based on this structure.

I suggest that it is necessary for us to be exposed to these conventions.

c) when evaluating whether a project is proper, it might be important to consider how many Mathlib sections it involve. A project could become too stressful when it is related to a lot of areas.

d) as to control the size of projects and in case some students are unable to finish in time, it might be recommended to complete the proof of main theorems first, leaving some complicated lemmas to be done gradually.