Experiences with Isabelle/HOL: Formalising Real Algebraic Geometry

Artie Khovanov, Michael Nedzelsky, and Wenda Li

2024-02-22

Outline

- Study $\{x \in \mathbb{R}^n \mid p_1(x) \geq 0, \ldots, p_n(x) \geq 0\} \quad (p_i \in \mathbb{Q}[\underline{T}])$.

Example

$$p_1(x) = 1 - x^2 - y^2$$
$$p_2(x) = (x - 0.3)^2 + (y - 0.5)^2 - 0.02$$
$$p_3(x) = (x + 0.3)^2 + (y - 0.5)^2 - 0.02$$
$$p_4(x) = 0.05x^2 + (y + 0.5)^2 - 0.02$$

- Want to formally verify algorithms and their implementations.

Real closed fields

Definition

A **real closed field** (RCF) R is an ordered field in which every positive element has a square root and every odd-degree polynomial has a root.

- RCFs have the same first-order properties as $\mathbb{R}$: many equivalent definitions.
- An RCF R has algebraic closure $R(i)$.

Examples

- $\mathbb{R} \subseteq \mathbb{C}$
- $\mathbb{R}_{\text{alg}} \subseteq \mathbb{C}_{\text{alg}}$

Definition

Let K be a field. The **field of Puiseaux series** over K is

$$K\langle\langle\varepsilon\rangle\rangle = \left\{ \sum_{k=-m}^{\infty} a_k \varepsilon^{k/n} \;\middle|\; a_k \in K, m \in \mathbb{Z}, n \in \mathbb{N} \right\}$$

Example

- $\varepsilon^{-2} + \pi\varepsilon^{-1/2} + \frac{1}{2}\varepsilon^{1/3} + 2\varepsilon^{11/6} + \cdots \in \mathbb{R}\langle\langle\varepsilon\rangle\rangle$.

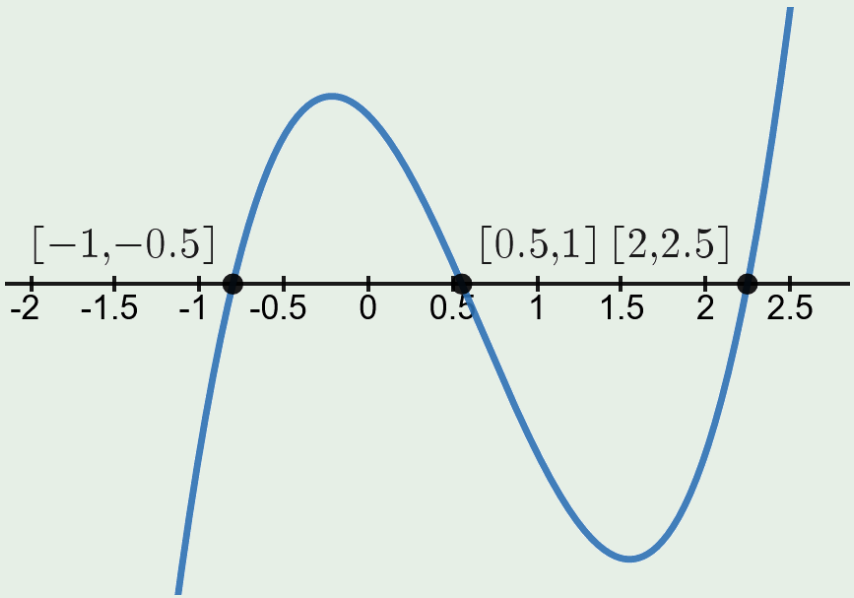
Theorem (Newton-Puiseaux for RCFs)

Let R be an RCF. Then $R\langle\langle\varepsilon\rangle\rangle$ is an RCF.

Representing algebraic numbers

- Root of $p \in \mathbb{Z}[T]$: rational interval isolation.
- Non-Archimedean coefficients: isolation not always possible.

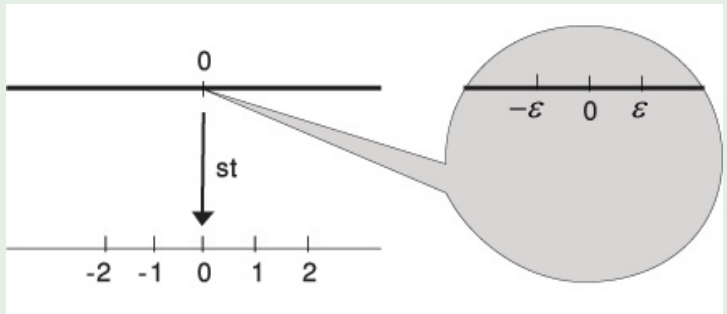
Example



$$p\,(T) = T^3 - 2T^2 - T + 1$$

Example

$p(T) = T(T - \varepsilon)$ has rationally indistinguishable roots $0, \varepsilon$.



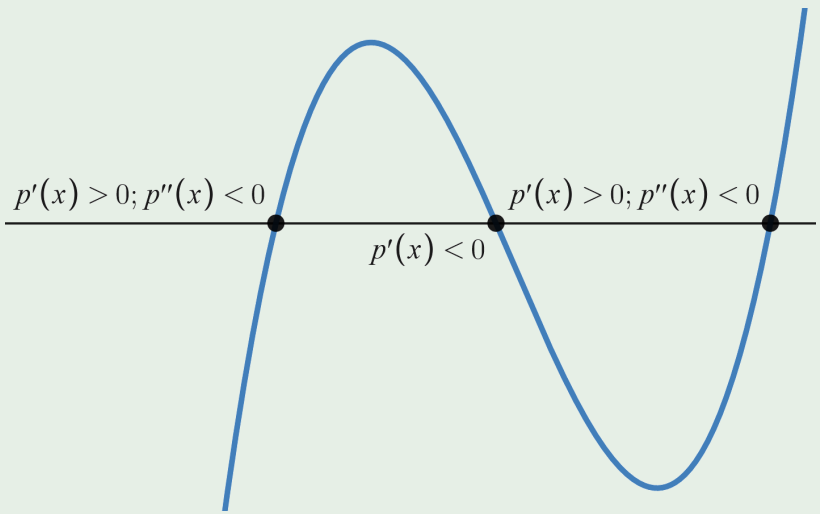
Thom encoding

Theorem (Thom's Lemma)

Let R be an RCF, and let $p \in R[T]$ be a nonzero polynomial. Then the roots of p are distinguished by the signs of

$$p'(T), p''(T), p^{(3)}(T), \ldots, p^{(\deg\ p)}(T).$$

Example



$$p\,(T) = T^3 - 2T^2 - T + 1$$

Example

$$p(T) = T(T - \varepsilon)$$
$$p'(T) = 2T - \varepsilon$$
$$p'(0) = -\varepsilon < 0$$
$$p'(\varepsilon) = \varepsilon > 0$$

Isabelle/HOL

- Generic proof assistant based on higher-order logic.
- Isar language – human-readable proofs.
- Sledgehammer – powerful automated proof-finding.

HOL-Algebra framework

- Type class system: attractive but inflexible for algebra
- Locale system: more flexible but automation is more difficult
- HOL-Algebra: locale-based abstract algebra library with structures defined on a carrier set

Example Proof in Isabelle/HOL

Lemma

Let R be an integral domain, and suppose $p \in R[T]$ is irreducible. If $\deg p > 1$, then p has no roots in R.

Proof.

Suppose $p(x) = 0$ for some $x \in R$. By the factor theorem, $p = (T - x) \cdot q$ for some $q \in R[T]$. Since $\deg p > 1$, p is a product of two non-units in $R[T]$.$_\#$ □

Example Proof in Isabelle/HOL

Lemma

Let R be an integral domain, and suppose $p \in R[T]$ is irreducible. If $\deg p > 1$, then p has no roots in R.

```
lemma (in UP_domain) irred_imp_no_root:
  assumes "f ∈ carrier P" "ring_irreducible⇘P⇙ f" "deg R f > 1"
  shows "∀x ∈ carrier R. to_fun f x ≠ 0"
proof
  fix x assume "x ∈ carrier R"
  show "to_fun f x ≠ 0"
  proof (rule ccontr)
    assume "¬ to_fun f x ≠ 0"
    hence pr: "f = X_poly_minus R x ⊗⇘P⇙ UP_root_div f x"
              "X_poly_minus R x ∈ carrier P" "UP_root_div f x ∈ carrier P"
      using factor_theorem
      by (auto simp add: ‹f ∈ carrier P› ‹x ∈ carrier R› X_minus_closed UP_root_div_closed)
    hence "deg R f = deg R (X_poly_minus R x) + deg R (UP_root_div f x)"
      using integral_iff deg_mult ‹ring_irreducible⇘P⇙ f› ring_irreducibleE(1) by metis
    hence "deg R (X_poly_minus R x) > 0" "deg R (UP_root_div f x) > 0"
      using degree_of_X_minus ‹x ∈ carrier R› ‹deg R f > 1› by auto
    hence "X_poly_minus R x ∉ Units P" "UP_root_div f x ∉ Units P"
      using unit_deg_zero degree_of_X_minus ‹x ∈ carrier R› ‹deg R f > 1› by fastforce+
    thus False using ‹ring_irreducible⇘P⇙ f› ring_irreducibleE(5) pr by force
  qed
qed
```

Successful automation

Lemma

Let R be an integral domain, and suppose $p \in R[T]^\times$.
Then $\deg p = 0$.

```
lemma (in UP_domain) pos_deg_not_unit:
  assumes "f ∈ carrier P" "deg R f > 0"
  shows "f ∉ Units P"
  by (metis P.Units_l_inv_ex add_is_0 assms deg_mult deg_one integral_iff zero_not_one not_gr_zero)

lemma (in UP_domain) unit_deg_zero:
  assumes "f ∈ Units P"
  shows "deg R f = 0"
  using assms pos_deg_not_unit Units_closed by blast
```

Automation difficulties: computation

Lemma

Let K be a field, and let $p(T) = aT + b$ be a degree-1 polynomial. Then $p(-b/a) = 0$.

```
lemma (in UP_field) deg_1_root_explicit:
  assumes "f ∈ carrier P" "deg R f = 1"
  shows "to_fun f (⊖ (f 0 ÷ f 1)) = 𝟬"
proof-
  have nz: "f 1 ≠ 𝟬"
    using assms coeff_simp lcoeff_nonzero_deg P_def by force
  hence cl: "⊖ (f 0 ÷ f 1) ∈ carrier R" using assms(1) cfs_closed by simp
  hence cl2: "f i ⊗ (⊖ (f 0 ÷ f 1)) [^] i ∈ carrier R" for i
    using cl cfs_closed assms(1) by simp
  have "to_fun f (⊖ (f 0 ÷ f 1)) = (⨁ i ∈ {..1}. f i ⊗ (⊖ (f 0 ÷ f 1)) [^] i)"
    using to_fun_formula assms cl by simp
  also have "... = f 1 ⊗ (⊖ (f 0 ÷ f 1)) ⊕ f 0"
    using R.finsum_Suc R.finsum_0 cl cl2 assms(1) cfs_closed by simp
  also have "... = f 0 ⊕ ⊖ (f 0 ⊗ (f 1 ⊗ inv f 1))"
    using assms(1) cfs_closed nz field_inv_closed by algebra
  also have "... = 𝟬"
    using assms(1) cfs_closed nz field_inv_closed field_Units R.r_neg by simp
  ultimately show ?thesis by simp
qed
```

Automation difficulties: complex objects

Lemma

In an ordered field, if $\sum_{i=1}^{n} x_i^2 = 0$, *then* $x_j = 0$ *for all* j.

```
lemma sos_nonvanishing:
  assumes "x: {1..n::nat} → carrier R" "(⨁i∈{1..n}. x i ⊗ x i) = 𝟬" "j ∈ {1..n}"
  shows "x j = 𝟬"
using assms proof (induction n arbitrary: x j)
  case 0
  thus ?case by fastforce
next
  case c: (Suc n)
  have f: "(λi. x i ⊗ x i) ∈ {1..n} → carrier R" using c(2) by auto
  hence sum_cl: "(⨁i∈{1..n}. x i ⊗ x i) ∈ carrier R" using finsum_closed by auto
  have t_cl: "x (Suc n) ∈ carrier R" using c(2) by auto
  hence t_sq_cl: "(x (Suc n) ⊗ x (Suc n)) ∈ carrier R" by auto
  have "insert (Suc n) {1..n} = {1..Suc n}" by auto
  hence "𝟬 = (x (Suc n) ⊗ x (Suc n)) ⊕ (⨁i∈{1..n}. x i ⊗ x i)"
    using finsum_insert[of "{1..n}" "Suc n" "λi. x i ⊗ x i"] c(2) c(3) by auto
  hence eqn: "(x (Suc n) ⊗ x (Suc n)) = ⊖(⨁i∈{1..n}. x i ⊗ x i)"
    using sum_cl t_sq_cl minus_equality by presburger
  hence "(x (Suc n) ⊗ x (Suc n)) ⊑ 𝟬" using sos_notation[of x n] sos_pos sum_cl c(2)
      minus_nonpos_is_nonneg by force
  hence sq_z: "(x (Suc n) ⊗ x (Suc n)) = 𝟬"
    by (metis t_sq_cl zero_closed sq_nonneg t_cl le_antisym)
  have sum_z: "(⨁i∈{1..n}. x i ⊗ x i) = 𝟬" by (metis sq_z add.inv_eq_1_iff sum_cl eqn)
```

Equivalent definitions of RCF

For an ordered field R the following properties are equivalent:

- R is real closed.
- $R[i] = R[T]/(T^2 + 1)$ is an algebraically closed field.
- R has the intermediate value property, i.e. for any $p \in R[T]$ and $a, b \in R$ such that $a < b$ and $p(a)p(b) < 0$, there exists $x \in (a, b)$ such that $p(x) = 0$.
- R has no non-trivial algebraic extensions which can be ordered.

Formalisation of the definitions

```
context ordered_field_with_poly
begin

definition "Prop1_real_closed ≡
  positive_cone = {x ⊗ x |x. x∈carrier R}
  ∧ (∀P. polynomial (carrier R) P ∧ odd (degree P) ⟶ (∃r∈carrier R. eval P r = 0))

definition "Prop2_alg_closed ≡
  algebraically_closed (Rupt (carrier R) [1,0,1])"

definition "Prop3_ivt ≡
  (∀P a b. (P ∈ carrier (poly_ring R) ∧ a ∈ carrier R ∧ b ∈ carrier R ∧ a ⊏ b
          ∧ eval P a ⊏ 0 ∧ 0 ⊏ eval P b)
    ⟶ (∃c. c ∈ carrier R ∧ a ⊏ c ∧ c ⊏ b ∧ eval P c = 0))"

definition "Prop4_closure ≡
  ∀P. P ∈ carrier (poly_ring R)
      ∧ (∃le'. (ordered_field ((Rupt (carrier R) P)) le'))
      ⟶ (degree P = 1)"
```

Formalisation of the equivalent definitions of RCF

Symmetric polynomials

The following lemma is the one of the key lemmas to prove the equivalence of the different definitions of RFC.

Lemma

Let R be a real closed field, and let $L \supseteq R$ be an algebraically closed field. Suppose $i \in L$ satisfies the equation $i^2 + 1 = 0$. Then all non-constant polynomials over R have a root in $R(i)$.

In order to prove this lemma we need to use symmetric polynomials and some results which were proved using types instead of locales.
NB: The formalisation of this lemma is still in progress.

The needed lemma about symmetric polynomials

Lemma

Let K be a field. Fix a polynomial $p \in K[T]$ of degree d, and let $x_1, \ldots, x_d$ be the roots of p (listed with multiplicity) in some algebraically closed field $C \supseteq K$. Let $q \in K[T_1, \ldots, T_d]$ be a symmetric polynomial. Then $q(x_1, \ldots, x_d) \in K$.

An existing (type-based) formalisation

```
lemma symmetric_poly_of_roots_in_subring_monic:
  fixes K :: "'b :: comm_ring_1 set"
    and A :: "nat set"
    and root :: "nat ⇒ 'a :: comm_ring_1"
    and l :: "'a ⇒ 'b"
    and q :: "'b mpoly"
  assumes "ring_closed K" and "∀m. coeff q m ∈ K"
  assumes "ring_homomorphism l"
  assumes "finite A"
  assumes "symmetric_mpoly A q" and "vars q ⊆ A"
  assumes "p = (∏i∈A. [:-root i, 1:])"
  assumes "∀i. l (poly.coeff p i) ∈ K"
  shows   "insertion (λx. l (root x)) q ∈ K"
```

Type-based vs. set-based formalisations

- Type-based:

```
lemma uminus_add_conv_diff:
  fixes a b::"'a :: ab_group_add"
  shows "- a + b = b - a"
```

- Pros: conciseness, better automation. For example, given a concrete type int, we have $-5 + 3 = 3 - 5$ without any assumptions.
- Cons: inflexibility, e.g., cannot reason about sub-groups.

- Set-based:

```
lemma uminus_add_conv_diff':
  fixes S:: "'a set"
    and add minus::"'a ⇒ 'a ⇒ 'a"
    and zero::'a  and uminus::"'a ⇒ 'a"
    and a b::"'a"
  assumes "ab_group_add_on_with
                S add zero minus uminus"
  assumes "a∈S" "b∈S"
  shows "add (uminus a) b = minus b a"
```

- Pros: flexibility.
- Cons: verbosity, more proof obligations. For example, to derive $-5 + 3 = 3 - 5$ we still need to show $3 \in \mathbb{Z}$ and $5 \in \mathbb{Z}$.

Types-to-sets framework

- A framework to convert a type-based formalisation to a set-based one.
- 'Prove easily and still be flexible.'
- Labour in *relativisation*: converting $\varphi[\alpha]$ to $\varphi^{\text{on}}_{\text{with}}\ [\alpha, A, \bar{f}]$.
  - For example, $\varphi[\alpha]$ can be

```
'a :: ab_group_add
```

and $\varphi^{\text{on}}_{\text{with}}\ [\alpha, A, \bar{f}]$ can be

```
ab_group_add_on_with S add zero minus uminus.
```

Types-to-sets: ad-hoc locale to encode type class assumptions over a carrier set

The predicate for type class assumptions

class.comm_semiring :: $('a \Rightarrow' a \Rightarrow' a) \Rightarrow ('a \Rightarrow' a \Rightarrow' a) \Rightarrow \mathrm{bool}$

can be 'internalised' automatically but those properties over a carrier set need to be encoded ad hoc:

comm_semiring_on_with :: 'a set $\Rightarrow ('a \Rightarrow' a \Rightarrow' a)$
$\Rightarrow ('a \Rightarrow' a \Rightarrow' a) \Rightarrow \mathrm{bool}$

And we relate the two encodings via the Transfer package:

```
lemma right_total_comm_semiring_transfer[transfer_rule]:
  includes lifting_syntax
  fixes T::"'a ⇒ 'b ⇒ bool"
  assumes [transfer_rule]: "right_total T" "bi_unique T"
  shows "((T ===> T ===> T)
              ===> (T ===> T ===> T)
                  ===>(=))
      (comm_semiring_on_with ({x. Domainp T x}))
      class.comm_semiring"
```

Types-to-sets: compiling out overloaded definitions

Functions on type classes may contain overloaded definitions (e.g., 1 and *):

```
fun power :: "'a::{one,times} ⇒ nat ⇒ 'a" where
    "power a 0 = 1" |
    "power a (Suc n) = a * (power a n)"
```

In types-to-sets, we may need to manually compile them out:

```
fun power_with::"('a ⇒ 'a ⇒ 'a)
                 ⇒ 'a ⇒ 'a ⇒ nat ⇒ 'a" where
    "power_with times' one' a 0 = one'" |
    "power_with times' one' a (Suc n)
          = times' a (power_with times' one' a n)"
```

A locale-based formulation of the previous result on symmetric polynomials

```
proposition (in field) symmetric_eval_in_ring:
  fixes K:: "'a set" and root p::"nat ⇒ 'a"
    and q::"nat multiset ⇒ 'a" and A::"nat set"
  assumes "algebraic_closure R K"
      and "p ∈ up (R⦇carrier := K⦈)"
      and "p = finprod (UP R)
                  (λx n. if n=0 then ⊖(root x)
                      else if n=1 then 1 else 0) A"
      and "∀i. root i ∈ carrier R"
      and "symmetric_mvar_poly A q"
      and "q ∈ carrier (Pring (R⦇carrier := K⦈) A)"
      and "finite A"
    shows "eval_in_ring R A root q ∈ K"
```

Transferring between similar theorems

Theorem (Newton-Puiseaux for RCFs)

Let R be an RCF. Then $R\langle\langle\varepsilon\rangle\rangle$ is an RCF.

```
instance fpxs :: ("{alg_closed_field, field_char_0, field_gcd}") alg_closed_field
```

Theorem (Newton-Puiseaux)

Let C be an algebraically closed field. Then $C\langle\langle\varepsilon\rangle\rangle$ is an algebraically closed field.

```
locale pxs_rcf = pxs_ordered_field +
  assumes rcf: "real_closed_field R (⊑s)"
begin
sublocale real_closed_field "R⟨X⟩" "(⊑)"
```

Finding a common lemma

Theorem ('Generalised Newton-Puiseux')

1. *Let K be a field of characteristic 0, and let $P(T) = \sum_{i=0}^{n} A_i T^i \in K\langle\langle\varepsilon\rangle\rangle[T]$ be an irreducible polynomial of degree $n > 1$. Then there is an irreducible polynomial $p \in K[T]$ of degree $d > 1$ with $d \mid n$.*
2. *Suppose further that $A_n = 1$ and $A_{n-1} = 0$. Write $n = rd$ and $p^r(T) = \sum_{i=0}^{n} a_k T^k$. Then $a_n = 1$, $a_{n-1} = 0$ and a_j is the leading coefficient of A_j for some $j < n$.*

Finding a common lemma

Theorem ('Generalised Newton-Puiseux')

1. *Let K be a field of characteristic 0, and let $P(T) = \sum_{i=0}^{n} A_i T^i \in K\langle\langle\varepsilon\rangle\rangle[T]$ be an irreducible polynomial of degree $n > 1$. Then there is an irreducible polynomial $p \in K[T]$ of degree $d > 1$ with $d \mid n$.*
2. *Suppose further that $A_n = 1$ and $A_{n-1} = 0$. Write $n = rd$ and $p^r(T) = \sum_{i=0}^{n} a_k T^k$. Then $a_n = 1$, $a_{n-1} = 0$ and a_j is the leading coefficient of A_j for some $j < n$.*

Corollary (of the first part)

Fix $n > 1$, and let K be a field of characteristic 0. Suppose all polynomials over K of degree $d > 1$ with $d \mid n$ are reducible. Then all polynomials over $K\langle\langle\varepsilon\rangle\rangle$ of degree n are reducible.

Transferring the common lemma

```
lemma RCF_pxs_fact2 [fpxs_transfer_field_char_0]:
  fixes p :: "'a::{field_char_0,field_gcd} fpxs poly"
  assumes deg_p: "degree p > 1"
    and lc_p: "lead_coeff p = 1"
    and coeff_deg_minus_1: "coeff p (degree p - 1) = 0"
    and "coeff p 0 ≠ 0" "irreducible p"
  obtains j r k where "irreducible (r :: 'a poly)" "degree r > 1" "degree p = k * degree r"
    "lead_coeff (r ^ k) = 1" "coeff (r ^ k) (degree (r ^ k) - 1) = 0"
    "j < degree (r ^ k)" "coeff p j ≠ 0"
    "coeff (r ^ k) j = fpxs_nth (coeff p j) (fpxs_val (coeff p j))"
```

```
lemma lift_fact:
  assumes "p ∈ carrier R⟪X⟫[T]" "deg R⟪X⟫ p > 1" "up_ring.coeff R⟪X⟫[T] p 0 ≠ 0↘R⟪X⟫↙"
    "up_ring.coeff R⟪X⟫[T] p (deg R⟪X⟫ p) = 1↘R⟪X⟫↙"
    "up_ring.coeff R⟪X⟫[T] p (deg R⟪X⟫ p - 1) = 0↘R⟪X⟫↙" "ring_irreducible↘R⟪X⟫[T]↙ p"
  obtains r j k where "r ∈ carrier R[T]" "ring_irreducible↘R[T]↙ r" "deg R r > 1"
    "deg R⟪X⟫ p = k * deg R r"
    "up_ring.coeff R[T] (r [^]↘R[T]↙ k) (deg R (r [^]↘R[T]↙ k)) = 1↘R↙"
    "up_ring.coeff R[T] (r [^]↘R[T]↙ k) (deg R (r [^]↘R[T]↙ k) - 1) = 0↘R↙"
    "j < deg R (r [^]↘R[T]↙ k)" "up_ring.coeff R⟪X⟫[T] p j ≠ 0↘R⟪X⟫↙"
    "up_ring.coeff R[T] (r [^]↘R[T]↙ k) j = fs_lead_coeff (up_ring.coeff R⟪X⟫[T] p j)"
```

Summary

- Sledgehammer is very effective at closing small (non-computational) gaps in a proof.
- Sledgehammer allows automation of theorems that are simple consequences of existing ones, and finds relevant lemmas.
- In the HOL-Algebra framework, Sledgehammer struggles with computations and complex objects. Closure conditions disrupt automation capabilities.
- It is feasible to transfer results from one framework to another using the types-to-sets framework and the Transfer package.

Project progress

- Newton-Puiseaux for RCFs has been formalised!
- Thom's Lemma has been formalised – up to equivalent definitions of an RCF.

```
theorem Thom_encoding_unique:
  assumes "P ∈ carrier (poly_ring R)" "P ≠ 0ₚ" "x ∈ carrier R" "y ∈ carrier R"
          "Der_sgn_cond x P = Der_sgn_cond y P" "eval P x = 0" "eval P y = 0"
  shows "x = y"
  using assms Thom_encoding_unique' self_in_Reali_Der_sgn_cond[of y] by algebra
```

- Equivalence of different definitions of RCF is still in progress.

Thank you for your attention.
Any questions?

Thom's lemma - informal proof

- Let $\sigma_0, \ldots, \sigma_d$ be a list of signs.
- Claim: $\{x \in R \mid \text{sign } p^{(j)}(x) = \sigma_j\}$ is either empty, a point, or an open interval.