Formalising Mathematics Coursework 3

In this project I define nonarchimedean norms and prove Hensel's Lemma for complete nonarchimedean fields $\mathbb{K}$: given a polynomial $f \in R[X]$, where $R \coloneqq \{x \in \mathbb{K} \mid |x| \leq 1\}$, and a point $t_0 \in R$ such that $|f(t_0)| < |f'(t_0)|^2$, there exists a unique point $t \in R$ such that $f(t) = 0$ and $|t - t_0| < |f'(t_0)|$; this point t also satisfies $|f'(t)| = |f'(t_0)|$ and $|t - t_0| = |f(t_0)| / |f'(t_0)|$.

1 Nonarchimedean Norms

```
/-- A type with a normed group structure is nonarchimedean if it satisfies `∥x + y∥ ≤ max ∥x∥ ∥y∥`. -/
  class nonarchimedean (K) [normed_group K] :=
  (nonarch : ∀ x y : K, ∥x + y∥ ≤ max (∥x∥) (∥y∥))
```

I defined a normed group $\mathbb{K}$ to be nonarchimedean if it satisfies $\forall\, x, y \in \mathbb{K}$, $|x + y| \leq \max |x|, |y|$, and proved some theorems about general nonarchimedean groups and rings, including that the inequality is an equality when $|x| \neq |y|$ and its contrapositive, both of which are used in the proof of Hensel's Lemma. It is important that a nonarchimedean structure need not be a field, as we will also be working with nonarchimedean rings.

```
/-- The nonarchimedean inequality with addition replaced with subtraction. -/
  theorem nonarchimedean.nonarch_sub (x y : K) : ∥x - y∥ ≤ max (∥x∥) (∥y∥) :=
    (sub_eq_add_neg x y).symm ▷ norm_neg y ▷ nonarchimedean.nonarch x (-y)

/-- The nonarchimedean inequality is equal if the elements being added have different norms. -/
  theorem nonarchimedean.eq_max_of_ne_norm {x y : K} (h : ∥x∥ ≠ ∥y∥) :
    ∥x + y∥ = max (∥x∥) (∥y∥) :=
  begin
    have : ∀ {x y : K}, ∥x∥ > ∥y∥ → ∥x + y∥ = max (∥x∥) (∥y∥),
    { ... },
    cases h.lt_or_lt with h h,
    { rw [add_comm, max_comm], exact this h },
    { exact this h }
  end

/-- If the nonarchimedean inequality is not equal, the elements being added have the same norm. -/
  theorem nonarchimedean.eq_norm_of_ne_max {x y : K} (h : ∥x + y∥ ≠ max (∥x∥) (∥y∥)) :
    ∥x∥ = ∥y∥ := of_not_not (mt nonarchimedean.eq_max_of_ne_norm h)
```

For $\mathbb{K}$ a normed ring we have a very useful theorem: a sequence in $\mathbb{K}$ is Cauchy if and only if the difference of consecutive terms tends to zero, expressed here, for a sequence $(s_n)_{n\in\mathbb{N}}$, in the form $\forall\, \epsilon > 0$, $\exists\, n$, $\forall\, m \geq n$, $|s_{m+1} - s_m| < \epsilon$. This will be used in Hensel's Lemma to show that the root exists as the limit of a Cauchy sequence.

```
/-- A `ℕ`-indexed sequence in a nonarchimedean normed ring is Cauchy iff the difference
    of its consecutive terms tends to `0`. -/
  theorem nonarchimedean.cau {K} [normed_ring K] [nonarchimedean K] {s : ℕ → K} :
    is_cau_seq norm s ⟺ ∀ ϵ > 0, ∃ i, ∀ j ≥ i, ∥s (j + 1) - s j∥ < ϵ :=
  begin
    ...
  end
```

2 The Unit Disc

Although Hensel's Lemma requires a complete nonarchimedean field $\mathbb{K}$, it mainly concerns the closed unit disc in $\mathbb{K}$, i.e. $\{x \in \mathbb{K} \mid |x| \leq 1\}$. This is defined as `disc` $\mathbb{K}$, a subring of $\mathbb{K}$, which is coerced into a `subtype` so that it can be viewed as a nonarchimedean normed ring.

```
/-- The closed unit ball in the nonarchimedean normed field `K`. -/
  def disc (K) [normed_field K] [nonarchimedean K] : subring K := {
    carrier   := {x | ‖x‖ ≤ 1},
    mul_mem'  := λ x y hx hy, (norm_mul_le x y).trans (one_mul (1 : ℝ) ▷
      mul_le_mul hx hy (norm_nonneg y) zero_le_one),
    one_mem'  := norm_one.le,
    add_mem'  := λ x y hx hy, (nonarchimedean.nonarch x y).trans (max_le hx hy),
    zero_mem' := norm_zero.le.trans zero_le_one,
    neg_mem'  := λ x hx, ((norm_neg x).symm ▷ hx : ‖-x‖ ≤ 1)
  }
```

The type `disc` $\mathbb{K}$ inherits the ring structure and norm of $\mathbb{K}$, in particular satisfying the equality $\forall\, x,\, y \in$ `disc` $\mathbb{K}$, $|x \cdot y| = |x| \cdot |y|$, which is only $\leq$ for a general normed ring. Under certain conditions, `disc` $\mathbb{K}$ also inherits division from $\mathbb{K}$: given two elements $x,\, y \in \mathbb{K}$, we have $|x| \leq |y| \implies |x|\,/\,|y| \leq 1$, so division on $\mathbb{K}$ is well-defined for points in `disc` $\mathbb{K}$ with the norm of the numerator less than or equal to the norm of the denominator. This division on `disc` $\mathbb{K}$ inherits properties from division on $\mathbb{K}$, specifically preserving norms and cancellation (for non-zero denominators). This will be used to represent the $f(t_n)/f'(t_n)$ part of the Newton-style sequence used in Hensel's Lemma.

```
/-- `disc K` inherits division from `K`, so long as the denominator has
  at least the numerator's norm. -/
  def divide {x y : disc K} (h : ‖x‖ ≤ ‖y‖) : disc K := ⟨x.1 / y.1,
  begin
    change ‖_‖ ≤ 1,
    rw [norm_div, ←norm_def x, ←norm_def y],
    by_cases hy : ‖y‖ = 0,
    { rw [hy, div_zero], exact zero_le_one },
    { exact (div_le_one (lt_of_le_of_ne (norm_nonneg y) (ne.symm hy))).mpr h }
  end⟩

/-- The norm in `disc K` preserves division. -/
  theorem divide.norm : ‖divide h‖ = ‖x‖ / ‖y‖ :=
    norm_div x.1 y.1

 /-- If the denominator `y` is non-zero, multiplying `divide h` by `y` cancels the division,
   leaving the numerator `x`. -/
   theorem divide.mul_cancel (hy : y ≠ 0) : divide h * y = x :=
     subtype.val_inj.mp (div_mul_cancel x.1 (mt subtype.val_inj.mp hy : y.1 ≠ (0 : disc K).val))
```

In order to use the `cau_seq.is_complete` API with `disc` $\mathbb{K}$, we must show that the norm on `disc` $\mathbb{K}$ is an absolute value. As the norms of normed fields are absolute values, in particular $\mathbb{K}$, `disc` $\mathbb{K}$ inherits the norm properties required to be an absolute value, with the `abv_add` property of $\mathbb{K}$ being used directly and `abv_mul` coming from `disc.norm_mul`, itself a wrapper for $\mathbb{K}$'s `norm_mul`.

```
/-- The norm in `disc K` is an absolute value, thanks to properties inherited
  from the normed field `K`. -/
  instance disc_norm_is_absolute_value : is_absolute_value (norm : disc K → ℝ) := {
    abv_nonneg  := norm_nonneg,
    abv_eq_zero := λ _, norm_eq_zero,
    abv_add     := λ x y, (normed_field.is_absolute_value.abv_add :
      ∀ x y : K, ‖x + y‖ ≤ ‖x‖ + ‖y‖) x y,
    abv_mul     := disc.norm_mul,
  }
```

With `disc_norm_is_absolute_value` defined, we can now show that `disc` $\mathbb{K}$ can inherit completeness from $\mathbb{K}$, with completeness of a space defined as every Cauchy sequence having a limit in that space. This requires showing that the limit of any Cauchy sequence in $\mathbb{K}$ with all terms having norm ≤ 1 must itself have norm ≤ 1, or in other words the inclusion of any Cauchy sequence in `disc` $\mathbb{K}$ into $\mathbb{K}$ converges to a point which is also in the inclusion of `disc` $\mathbb{K}$, and which must be a limit in `disc` $\mathbb{K}$ as well as $\mathbb{K}$ and `disc` $\mathbb{K}$ share the same norm. This is proved by letting $\epsilon = 1$ in the definition of Cauchy sequence $(s_n)_{n\in\mathbb{N}}$ converging to its limit $\hat{s}$, producing an integer n such that $|\hat{s} - s_n| < 1$. This is used, along with the fact that $|s_n| \leq 1$, to show $|\hat{s}| = |\hat{s} - s_n + s_n| \leq \max|\hat{s} - s_n||s_n| \leq 1$, giving us the condition on $|\hat{s}|$ we need to prove completion.

```
/-- `disc K` inherits the completeness of `K`, i.e. if all Cauchy sequences in `K` are
  convergent, then so are all Cauchy sequences in `disc K`. -/
  instance disc_is_complete [cau_seq.is_complete K norm] :
    cau_seq.is_complete (disc K) norm := ⟨λ s,
  begin
    let s' : cau_seq K norm := ⟨λ n, (s n).1, s.2⟩,
    use s'.lim,
    { cases s'.equiv_lim 1 zero_lt_one with n hn,
      rw [←sub_add_cancel s'.lim (s' n)],
      apply le_trans (nonarchimedean.nonarch (s'.lim - s' n) (s' n)),
      have : ‖s' n - cau_seq.const norm s'.lim n‖ = ‖s'.lim - s' n‖,
      { rw [←norm_neg, neg_sub, cau_seq.const_apply] },
      exact max_le (this ▷ le_of_lt (hn n (le_refl n))) (s n).2 },
    { exact s'.equiv_lim }
  end⟩
```

3 Taylor's Theorem

The proof of Hensel's Lemma uses some linear and quadratic Taylor Theorem approximations, those being for any polynomial f and point t_0, there exist polynomials g_1, g_2 such that:

$$f(t) = f(t_0) + (t - t_0) \cdot g_1(t), \quad f(t) = f(t_0) + (t - t_0) \cdot f'(t_0) + (t - t_0)^2 \cdot g_2(t).$$

We define g_2 by taking the full polynomial Taylor decomposition `taylor t_0 f` and removing the first two terms, then lowering the degree of each $(t-t_0)$ by 2. We can then further define g_1 to be $f'(t_0)+(t-t_0)\cdot g_2(t)$, giving us the simpler linear case.

```
/-- Any polynomial `f` can be approximated as a quadratic polynomial centred on a chosen point `t₀`. -/
  theorem taylor₂ (f : polynomial R) (t₀ : R) : ∃ err : polynomial R, ∀ t,
    f.eval t = f.eval t₀ + (t - t₀) * f.derivative.eval t₀ + (t - t₀)^2 * err.eval t :=
  begin
    use (((taylor t₀ f).erase 0).erase 1).sum (λ i a, C a * (X - C t₀) ^ (i - 2)), ...
  end

/-- Any polynomial `f` can be approximated as a linear polynomial centred on a chosen point `t₀`. -/
  theorem taylor₁ (f : polynomial R) (t₀ : R) : ∃ err : polynomial R, ∀ t,
    f.eval t = f.eval t₀ + (t - t₀) * err.eval t :=
  begin
    cases taylor₂ f t₀ with err h,
    use C (f.derivative.eval t₀) + (X - C t₀) * err, intro t,
    rw [h t, eval_add, eval_C, mul_add, ←add_assoc, eval_mul, eval_sub, eval_X, eval_C, ←mul_assoc, sq]
  end
```

4 Continuity

The last tools needed to begin proving Hensel's Lemma are ways to show that continuous functions (specifically polynomials and norms) preserve Cauchy sequences and their limits. This requires moving from the ϵ

definition of Cauchy to the filter definition, for which continuity of polynomials and norms is already defined. The API for moving between these definitions, however, requires that a filter Cauchy sequence be defined over a normed field in order to turn it into a ϵ Cauchy sequence, while only needing a normed ring with an absolute value for the reverse operation. To solve this, I simply copied over the theorem `cauchy_seq.is_cau_seq` and restated it for a normed ring with an absolute value, satisfying all the requirements the proof needs. This allowed me to make tools to help with composing polynomials and norms with Cauchy sequences and comparing the limits of the compositions with the compositions of the limits.

```
/-- A filter-wise Cauchy sequence is an absolute-value-wise Cauchy sequence.
  (This already exists in `topology.metric_space.cau_seq_filter`, but only for normed fields,
  here it is restated for normed rings whose norm is an absolute value). -/
  theorem cauchy_seq.is_cau_seq'  {R} [normed_ring R] [is_absolute_value (norm : R → ℝ)]
    {f : ℕ → R} (hf : cauchy_seq f) : is_cau_seq norm f :=
  begin
    ...
  end

/-- A Cauchy sequence formed by composing a Cauchy sequence with a polynomial. -/
  noncomputable def polynomial_comp (f : polynomial R) (s : cau_seq R norm) : cau_seq R norm :=
    ⟨λ n, f.eval (s n), ((f.continuous.tendsto s.lim).comp s.tendsto_limit).cauchy_seq.is_cau_seq'⟩

/-- The composition of a polynomial with a Cauchy sequence's limit equals the limit of the
  composition of the polynomial with the Cauchy sequence. -/
  theorem polynomial_comp_lim (f : polynomial R) (s : cau_seq R norm) :
    f.eval s.lim = (polynomial_comp f s).lim := tendsto_nhds_unique
      ((f.continuous.tendsto s.lim).comp s.tendsto_limit) (polynomial_comp f s).tendsto_limit

/-- A Cauchy sequence formed by composing a Cauchy sequence with the norm. -/
  noncomputable def norm_comp (s : cau_seq R norm) : cau_seq ℝ norm :=
    ⟨λ n, ‖s n‖, ((continuous_norm.tendsto s.lim).comp s.tendsto_limit).cauchy_seq.is_cau_seq'⟩

/-- The composition of the norm with a Cauchy sequence's limit equals the limit of the
  composition of the norm with the Cauchy sequence. -/
  theorem norm_comp_lim (s : cau_seq R norm) : ‖s.lim‖ = (norm_comp s).lim :=
    tendsto_nhds_unique ((continuous_norm.tendsto s.lim).comp s.tendsto_limit) (norm_comp s).tendsto_limit
```

5 Hensel's Lemma

The proof of Hensel's Lemma uses a Newton-style sequence converging to a root near t_0, i.e. first term t_0, subsequent terms $t_{n+1} = t_n - f(t_n)/f'(t_n)$. The division $f(t_n)/f'(t_n)$ uses `disc.divide`, and so requires a proof of $|f(t_n)| \leq |f'(t_n)|$. To avoid having to prove a fact about t_n in its own definition, we introduce the required inequality with an `if` statement, returning a default value of t_0 if the inequality doesn't hold (but as we will prove later, it always does).

```
  variables {K : Type} [normed_field K] [nonarchimedean K]
  variables (f : polynomial (disc K)) (t₀ : disc K) (ht₀ : ‖f.eval t₀‖ < ‖f.derivative.eval t₀‖ ^ 2)

/-- A Newton-style sequence converging to a root of `f` near `t₀`. -/
  noncomputable def seq : ℕ → disc K
  | 0     := t₀
  | (n+1) :=
    if h : ‖f.eval (seq n)‖ ≤ ‖f.derivative.eval (seq n)‖ then
      seq n - disc.divide h
    else t₀
```

We also define an important constant $\kappa := |f(t_0)| \,/\, |f'(t_0)|^2$. Thanks to the hypothesis $|f(t_0)| < |f'(t_0)|^2$, we must have $0 \leq \kappa < 1$, and so repeated multiplication by κ creates a strictly monotone sequence, which we will use to show, among other things, that $f(t_n)$ tends to 0.

```
/-- The constant `κ` is defined as `‖f(t₀)‖ / ‖f'(t₀)‖^2`. -/
  noncomputable def κ : ℝ := ‖f.eval t₀‖ / ‖f.derivative.eval t₀‖ ^ 2

/-- The constant `κ` is non-negative. -/
  theorem zero_le_κ : 0 ≤ κ f t₀ := div_nonneg (norm_nonneg _) (sq_nonneg _)

/-- Given the inequality `‖f(t₀)‖ < ‖f'(t₀)‖^2`, the constant `κ` is strictly
  less than `1`. -/
  theorem κ_lt_one : κ f t₀ < 1 := (div_lt_one (lt_of_le_of_lt (norm_nonneg _) ht₀)).mpr ht₀
```

There are four properties we need from the Newton-style sequence, all with interdependent proofs, so we prove them all simulatenously in one big round of induction. Here we make use of tools we developed earlier, such as Taylor approximations and most of the `disc` $\mathbb{K}$ theorems, especially those relating to `disc.divide`.

```
/-- For every `n : ℕ`, `seq` satisfies the following properties:
  • `‖f(tₙ₊₁) ‖ ≤ κ * ‖f(tₙ)‖`,
  • `‖f'(tₙ₊₁)‖ = ‖f(t₀)‖`,
  • `‖tₙ₊₁ - tₙ‖ ≤ κ^n * ‖f(t₀)‖/‖f'(t₀)‖`,
  • `‖f(tₙ₊₁)‖ ≤ κ^(n+1) * ‖f(t₀)‖`. -/
  theorem seq_properties (n : ℕ) :
    ‖f.eval (seq f t₀ (n+1))‖ ≤ (κ f t₀) * ‖f.eval (seq f t₀ n)‖
    ‖f.derivative.eval (seq f t₀ (n+1))‖ = ‖f.derivative.eval t₀‖
    ‖seq f t₀ (n+1) - seq f t₀ n‖ ≤ (κ f t₀) ^ n * ‖f.eval t₀‖ / ‖f.derivative.eval t₀‖
    ‖f.eval (seq f t₀ (n+1))‖ ≤ (κ f t₀) ^ (n + 1) * ‖f.eval t₀‖ :=
  begin
    have hf' : f.derivative.eval t₀ ≠ 0, { ... },
    induction n with n ih,
    { have h : ‖f.eval (seq f t₀ 0)‖ ≤ ‖f.derivative.eval (seq f t₀ 0)‖ :=
        le_of_lt (lt_of_lt_of_le ht₀ (sq_le (norm_nonneg _) (disc.norm_le_one _))),
      have h' : ‖f.eval (seq f t₀ 0 - disc.divide h)‖ ≤ κ f t₀ * ‖f.eval (seq f t₀ 0)‖,
      { cases taylor₂ f (seq f t₀ 0) with g hg, ... },
      rw [seq.def_succ h], split,
      { exact h' }, { split, { ... }, { split, { ... }, { ... } } } },
    { rcases ih with ⟨h₁, h₂, h₃, h₄⟩,
      have : ‖f.eval (seq f t₀ (n + 1))‖ ≤ ‖f.eval t₀‖ := ...,
      have h : ‖f.eval (seq f t₀ (n+1))‖ ≤ ‖f.derivative.eval (seq f t₀ (n+1))‖ := ...,
      have hfn' : f.derivative.eval (seq f t₀ (n+1)) ≠ 0, { ... },
      have h' : ‖f.eval (seq f t₀ (n+1) - disc.divide h)‖ ≤ κ f t₀ * ‖f.eval (seq f t₀ (n+1))‖,
      { cases taylor₂ f (seq f t₀ (n+1)) with g hg, ... },
      rw [seq.def_succ h],
      split, { exact h' }, { split, { ... }, { split, { ... }, { ... } } } }
  end
```

The final theorem, `hensels_lemma`, also has many parts, most of which are also needed to prove uniqueness, hence the use of `have`'s and `refine ⟨hzero, hlt, hder, heq, _⟩` than proving each in their own {scope}. Most parts of the theorem consist of passing the Newton-style sequence through norms and polynomials to prove properties of its limit, making extensive use of the `norm_comp_lim` and `polynomial_comp_lim` theorems defined earlier. This is also where the completeness of `disc` $\mathbb{K}$ is used, as well as the proof that in a nonarchimedean ring, a sequence is Cauchy if and only if the difference of consecutive terms tends to zero, to prove that the Newton-style sequence has a limit in `disc` $\mathbb{K}$.

```
/-- Hensel's Lemma: In a complete nonarchimedean field `K`, given any polynomial `f` with coefficients in
  `disc K` and a point `t₀ : disc K` such that `‖f(t₀)‖ < ‖f'(t₀)‖²`, there exists a unique point `t : disc K`
  such that `f(t) = 0` and `‖t - t₀‖ < ‖f'(t₀)‖`. This point `t` also satisfies `‖f'(t)‖ = ‖f'(t₀)‖` and
  `‖t - t₀‖ = ‖f(t₀)‖ / ‖f'(t₀)‖`. -/
theorem hensels_lemma [cau_seq.is_complete K norm] : ∃ t, f.eval t = 0  ‖t - t₀‖ < ‖f.derivative.eval t₀‖
  ‖f.derivative.eval t‖ = ‖f.derivative.eval t₀‖  ‖t - t₀‖ = ‖f.eval t₀‖ / ‖f.derivative.eval t₀‖
  ∀ s, f.eval s = 0 → ‖s - t₀‖ < ‖f.derivative.eval t₀‖ → s = t :=
begin
  have hf' : 0 < ‖f.derivative.eval t₀‖ := lt_of_le_of_lt (norm_nonneg (f.eval t₀))
    (lt_of_lt_of_le ht₀ (sq_le (norm_nonneg _) (disc.norm_le_one _))),
  have hcau : is_cau_seq norm (seq f t₀), { apply nonarchimedean.cau.mpr, ... },
  let t : disc K := cau_seq.lim ⟨seq f t₀, hcau⟩, use t,
  have hzero : f.eval t = 0,
  { rw [polynomial_comp_lim], ... },
  have heq : ‖t - t₀‖ = ‖f.eval t₀‖ / ‖f.derivative.eval t₀‖, { ... },
  have hlt : ‖t - t₀‖ < ‖f.derivative.eval t₀‖,
  { rw [heq], apply (div_lt_iff hf').mpr, rw [←sq], exact ht₀ },
  have hder : ‖f.derivative.eval t‖ = ‖f.derivative.eval t₀‖,
  { rw [polynomial_comp_lim, norm_comp_lim], ... },
  refine ⟨hzero, hlt, hder, heq, _⟩,
  -- Uniqueness
  intros s hs₁ hs₂,
  apply classical.by_contradiction, intro hst,
  have := nonarchimedean.nonarch_sub (s - t₀) (t - t₀),
  rw [sub_sub_sub_cancel_right] at this,
  apply absurd (lt_of_le_of_lt this (max_lt hs₂ hlt)),
  apply not_lt_of_le,
  cases taylor₂ f t with g hg,
  have h := hg s,
  rw [hzero, hs₁, zero_add] at h,
  have h := congr_arg norm (eq_neg_of_add_eq_zero h.symm),
  rw [norm_neg, disc.norm_mul, disc.norm_mul] at h,
  have : ‖(s - t) ^ 2‖ * ‖g.eval s‖ ≤ ‖(s - t) ^ 2‖ :=
    mul_le_of_le_one_right (norm_nonneg _) (disc.norm_le_one _),
  rw [←h, disc.norm_pow, sq, hder] at this,
  exact (mul_le_mul_left (lt_of_le_of_ne (norm_nonneg _)
    (λ h, sub_ne_zero.mpr hst (norm_eq_zero.mp h.symm)))).mp this
end
```

At almost 100 lines, this is the longest theorem in this project, partly because of the number of things being proved but also because it includes a lot of edge cases: if t_0 is itself a root of f, the sequence becomes constant at t_0 and a lot of things become zero (e.g. κ and $|t - t_0|$), requiring a separate case when this interferes with the main branch of the proof. Also, the properties proved in `seq_properties` mostly concern t_{n+1} rather than t_n itself, hence we must often split into $n = 0$ and $n = n' + 1$ cases before invoking `seq_properties`, creating more edge cases. These problems could potentially be alleviated by a separate proof that if $f(t_0) = 0$, then $t_n = t_0$ for all n, making these edge cases easier to deal with, and by defining separate theorems for properties in `seq_properties` that hold for any n rather than just $n + 1$, e.g. the inequality $|f(t_n)| \leq \kappa^n |f(t_0)|$ as opposed to $|f(t_{n+1})| \leq \kappa^{n+1} |f(t_0)|$.