```
From Coq Require Import Arith Lia Reals ZArith Ring.

(* ================================================================ *)
(*  This file formalizes a reading of Dedenko's manuscript where *)
(*  the parameters m,p live over the reals and a global           *)
(*  "normalization" multiplier o>1 is introduced so that          *)
(*            o^n = 2·n                                           *)
(*  captures the entire family of exponents under consideration.  *)
(*  Choosing the full-coverage normalization o = 2 collapses the  *)
(*  search for natural solutions of Fermat's equation to n∈{1,2}.  *)
(* ================================================================ *)

(* ---------- Real-parameter identities (m,p ∈ R) ---------- *)
Local Open Scope R_scope.

(* Algebraic consequences of introducing parameters m and p in the reals. *)
Lemma sum_diff_from_parameters_R
      (n : nat) (m p : R) :
  let z := pow m n + pow p n in
  let x := pow m n - pow p n in
  z + x = 2 * pow m n /\
  z - x = 2 * pow p n.
Proof.
  intros z x; unfold z, x; split; ring.
Qed.

Close Scope R_scope.

(* ---------- Integer-parameter specialization (m,p ∈ Z) ---------- *)
Local Open Scope Z_scope.

(* Integer specialization used to reason about parity. *)
Lemma sum_diff_from_parameters_Z
      (n : nat) (m p : Z) :
  let z := m ^ Z.of_nat n + p ^ Z.of_nat n in
  let x := m ^ Z.of_nat n - p ^ Z.of_nat n in
  z + x = 2 * m ^ Z.of_nat n /\
  z - x = 2 * p ^ Z.of_nat n.
Proof.
  intros z x; unfold z, x; split; nia.
Qed.

Corollary parity_condition_Z
          (n : nat) (m p : Z) :
  let z := m ^ Z.of_nat n + p ^ Z.of_nat n in
  let x := m ^ Z.of_nat n - p ^ Z.of_nat n in
  Z.even (z + x) = true /\
  Z.even (z - x) = true.
Proof.
  intros z x.
  destruct (sum_diff_from_parameters_Z n m p) as [Hzx Hzx'].
  split.
  - replace (z + x) with (2 * m ^ Z.of_nat n) by exact Hzx.
    rewrite Z.even_mul; simpl; reflexivity.
  - replace (z - x) with (2 * p ^ Z.of_nat n) by exact Hzx'.
    rewrite Z.even_mul; simpl; reflexivity.
Qed.

(* If the observed parity of (z±x) contradicts the necessary evenness
   implied by the parametrization, then no such integers m,p can exist. *)
Lemma no_parameters_if_parity_violation (n : nat) (z x : Z) :
```

```
  Z.even (z + x) = false \/ Z.even (z - x) = false ->
  ~ (exists m p : Z,
        z = m ^ Z.of_nat n + p ^ Z.of_nat n /\
        x = m ^ Z.of_nat n - p ^ Z.of_nat n).
Proof.
  intros Hpar [m [p [Hz Hx]]].
  destruct (sum_diff_from_parameters_Z n m p) as [Hsum Hdiff].
  destruct Hpar as [H1|H2].
  - rewrite Hz, Hx, Hsum in H1.
    rewrite Z.even_mul in H1; simpl in H1. discriminate.
  - rewrite Hz, Hx, Hdiff in H2.
    rewrite Z.even_mul in H2; simpl in H2. discriminate.
Qed.

(* A concrete obstruction (special case of the lemma above). *)
Lemma no_parameters_for_example :
  ~ (exists m p : Z,
        2%Z = m ^ Z.of_nat 3 + p ^ Z.of_nat 3 /\
        1%Z = m ^ Z.of_nat 3 - p ^ Z.of_nat 3).
Proof.
  apply (no_parameters_if_parity_violation 3 2 1).
  now left.
Qed.

Close Scope Z_scope.
Local Open Scope nat_scope.

(* ---------- Elementary growth facts on naturals ---------- *)

(* Exponential growth compared to linear growth for powers of 2. *)
Lemma pow2_gt_linear_shift (k : nat) :
  2 ^ (k + 3) > 2 * (k + 3).
Proof.
  induction k as [|k IH]; simpl.
  - lia.
  - replace (S k + 3) with (k + 4) by lia.
    replace (2 ^ (S k + 3)) with (2 * 2 ^ (k + 3)) by
         (replace (S k + 3) with (S (k + 3)) by lia;
          rewrite Nat.pow_succ_r; lia).
    assert (Htmp : 2 * 2 ^ (k + 3) > 4 * (k + 3)) by nia.
    apply Nat.le_lt_trans with (m := 4 * (k + 3)).
    + lia.
    + exact Htmp.
Qed.

Lemma pow2_gt_linear (n : nat) :
  3 <= n -> 2 ^ n > 2 * n.
Proof.
  intros Hn.
  destruct (Nat.le_exists_sub 3 n Hn) as [k [Hk _]].
  rewrite Hk.
  replace (3 + k) with (k + 3) by lia.
  apply pow2_gt_linear_shift.
Qed.

Lemma pow_eq_linear_cases (n : nat) :
  2 ^ n = 2 * n -> n = 0 \/ n = 1 \/ n = 2.
Proof.
  destruct n as [|n].
  - simpl. intro H. now left.
  - destruct n as [|n].
```

```
    + simpl. intro H. right; left; lia.
    + destruct n as [|n].
      * simpl. intro H. right; right; lia.
      * intro H.
        assert (3 <= S (S (S n))) by lia.
        specialize (pow2_gt_linear _ H0) as Hgt.
        rewrite H in Hgt. lia.
Qed.

Lemma pow_eq_linear_positive (n : nat) :
  2 ^ n = 2 * n -> n = 1 \/ n = 2.
Proof.
  intro H.
  destruct (pow_eq_linear_cases n H) as [H0 | [H1 | H2]].
  - subst n. discriminate.
  - now left.
  - now right.
Qed.

(* Exponential growth compared to linear growth for powers of 3. *)
Lemma pow3_gt_linear_shift (k : nat) :
  3 ^ (k + 1) > 2 * (k + 1).
Proof.
  induction k as [|k IH]; simpl.
  - lia.
  - replace (S k + 1) with (k + 2) by lia.
    replace (3 ^ (S k + 1)) with (3 * 3 ^ (k + 1)) by
        (replace (S k + 1) with (S (k + 1)) by lia;
         rewrite Nat.pow_succ_r; lia).
    assert (Htmp : 3 * 3 ^ (k + 1) > 3 * (2 * (k + 1))) by nia.
    apply Nat.le_lt_trans with (m := 3 * (2 * (k + 1))).
    + lia.
    + exact Htmp.
Qed.

Lemma pow3_gt_linear (n : nat) :
  1 <= n -> 3 ^ n > 2 * n.
Proof.
  intros Hn.
  destruct (Nat.le_exists_sub 1 n Hn) as [k [Hk _]].
  rewrite Hk.
  replace (1 + k) with (k + 1) by lia.
  apply pow3_gt_linear_shift.
Qed.

(* The equation o^n = 2n with integer o > 1 forces o = 2 and n in {1,2}. *)
Lemma integer_solution_o (o n : nat) :
  1 < o -> 1 <= n -> o ^ n = 2 * n -> o = 2 /\ (n = 1 \/ n = 2).
Proof.
  intros Ho Hn HoEq.
  destruct o as [|o]; [lia|].
  destruct o as [|o]; [lia|].
  destruct o as [|o].
  - (* o = 2 *)
    simpl in HoEq.
    split; [reflexivity|].
    apply pow_eq_linear_positive in HoEq.
    assumption.
  - (* o >= 3 leads to contradiction *)
    assert (Hcomp : 3 ^ n <= (S (S (S o))) ^ n).
    { apply Nat.pow_le_mono_l; lia. }
```

```
    specialize (pow3_gt_linear n Hn) as Hgt.
    rewrite HoEq in Hcomp.
    lia.
Qed.

(* -------------------------------------------------------------- *)
(*  Normalization parameter and the conditional derivation of FLT *)
(* -------------------------------------------------------------- *)
Section Normalization_Parameter.

(* The manuscript introduces a single multiplier o>1 so that o^n = 2·n
   serves as a "normalization" capturing all exponents at once.  We keep
   o abstract and only assume it satisfies the manuscript's equation for
   every putative Fermat counterexample. *)
Variable o : nat.
Hypothesis normalization_gt1 : 1 < o.

Hypothesis normalization_equation :
  forall (n x y z : nat),
    2 < n ->
    x ^ n + y ^ n = z ^ n ->
    o ^ n = 2 * n.  (* "2·n" is product, not a power *)

Theorem fermat_last_theorem_from_normalization :
  forall (n x y z : nat),
    2 < n ->
    x ^ n + y ^ n = z ^ n -> False.
Proof.
  intros n x y z Hn Heq.
  specialize (normalization_equation n x y z Hn Heq) as HoEq.
  destruct (integer_solution_o o n) as [Ho2 Hcases].
  - exact normalization_gt1.
  - lia.
  - exact HoEq.
  - destruct Hcases as [Hn1 | Hn2]; lia.
Qed.

End Normalization_Parameter.

(* By picking the "full coverage" normalization o = 2 (as justified in the
   manuscript's discussion of f(n) = (2n)^(1/n)), we obtain the classical
   contradiction: the resulting equality 2^n = 2·n forces n ∈ {1,2}. *)
Corollary fermat_last_theorem_with_o_two :
  (forall (n x y z : nat),
      2 < n ->
      x ^ n + y ^ n = z ^ n ->
      2 ^ n = 2 * n) ->
  forall (n x y z : nat),
    2 < n ->
    x ^ n + y ^ n = z ^ n -> False.
Proof.
  intros Hnorm n x y z Hn Heq.
  eapply (fermat_last_theorem_from_normalization 2).
  - lia.
  - apply Hnorm; assumption.
  - exact Hn.
  - exact Heq.
Qed.

(* Under the normalization-based reading of the manuscript, Fermat's equation
   has no natural number solutions for exponents above 2. *)
```

```
FLT on  main [?] …
➜ coqc FLT.v

FLT on  main [?] took 3.1s …
➜
```

Code available at:
https://github.com/Gendalf71/FLT-Coq