A Fuzzy–Confidence Engine for Chen's Approach to the Goldbach Conjecture

(Working Note)

August 17, 2025

Abstract

We describe a computational "engine" inspired by Chen's theorem, but framed within a philosophy of *fuzzy truth*: every even number n is treated as being in a superposition of Goldbach–valid and Goldbach–invalid until collapsed by explicit observation of a prime decomposition or its semiprime analog. Instead of a hard Boolean answer, each n produces a confidence value in $[0, 1]$ that accumulates globally. This note outlines the philosophy, implementation, and preliminary stress tests of such an engine.

1 Background

Goldbach's Conjecture asserts that every even integer $n > 2$ is the sum of two primes. Chen's theorem (1973) proves that every sufficiently large even integer can be written as

$$n = p + q$$

where p is prime and q is either prime or semiprime. This motivates an experimental framework where semiprimes are taken as partial evidence toward Goldbach validity.

2 Philosophy: Superposition and Fuzzy Truth

Instead of Boolean truth, we use fuzzy values:

- $0 \approx$ false,
- $0.5 \approx$ "Both" (superposition),
- $1 \approx$ true.

No case is ever "neither." When $n - p = q$ is prime, the confidence for n collapses to nearly 1. When q is semiprime, confidence is incremented by a fuzz amount (up to five times). Otherwise, confidence remains 0.5, interpreted as "Both."

3 The Chen–Goldbach Engine

For each even n, we iterate over a fixed list of odd primes (currently $\{3, 5, 7, 11, 13\}$). For each p:

1. Compute $q = n - p$.
2. If q is prime, confidence ≈ 1 and the search halts.
3. If q is semiprime, increment confidence by a fuzzy amount.
4. Otherwise, confidence remains at 0.5.

After a maximum of five semiprime increments or any prime confirmation, the local test for n ends. Global confidence is updated as a weighted average:

$$C_{\text{global}} \leftarrow \frac{W_{\text{GC}} \cdot C_{\text{global}} + W_{\text{Chen}} \cdot C_n}{W_{\text{GC}} + W_{\text{Chen}}},$$

with $W_{\text{GC}} = 1$ and $W_{\text{Chen}} = 10^{-5}$, modulated by growth factors as n increases.

4 Growth Factors

To reflect the intuition that larger n should carry more evidentiary weight, we introduce mild growth factors:

- Semiprime bumps are scaled by $\left(\frac{\log n}{\log N_0}\right)^{\beta}$, capped at $3\times$ the baseline.
- The Chen weight W_{Chen} grows gently with n, capped at $5\times$ the baseline.

5 Implementation Details

- **Primality:** deterministic Miller–Rabin for 64-bit integers (fixed bases).
- **Semiprime detection:** trial division by a dynamic small-prime budget (shrinks with n; minimum budget is 5); if no small split is found, the case remains "Both."
- **p-reordering:** the list $\{3, 5, 7, 11, 13\}$ is reordered at each n to prioritize promising subtractions (avoid q divisible by small primes).
- **Counter dynamics:** whenever a true prime decomposition is found, the step increment increases chaotically (random jump).

6 Preliminary Stress Test

We ran the engine with dynamic jumps, adaptive small-prime budgets, and growth factors enabled. Within a 5-second budget, the engine reached

$$n \approx 2.38 \times 10^{12},$$

with a global confidence of about 0.565 (from an initial 0.5). The run demonstrates both the scalability of the approach and the slow-but-steady accumulation of confidence.

7 Discussion

Historically, many purported "proofs" of Goldbach have truncated arguments or assumed truth beyond a checked bound. Our approach is more philosophically modest: it accumulates evidence with an explicit measure of fuzziness, never claiming a crisp yes/no. This is closer to the way non-mathematicians reason with numbers in everyday life: intuitively, probabilistically, and with tolerance for ambiguity.

8 Rigor Without Bivalence

Thesis. *Rigor is a property of procedures, not a predicate of propositions.* A procedure is rigorous when it (i) declares its informational limits, (ii) preserves neutrality in the absence of decisive observation, and (iii) updates conservatively when positive observation occurs. In this paper we realize that by assigning the neutral value 0.5 ("Both") by default, collapsing toward 1 only on observation, and never assigning "false" merely for lack of evidence.

R1 (Neutrality). Before observation, assign 0.5 (Both) to relevant claims unless strong, context-independent evidence is present.
R2 (Observation). An observation step returns a local verdict $X \in \{0.5\} \cup (0.5, 1)$ where $X = 1$ represents a decisive positive observation (e.g., a witnessed prime q) and $X = 0.5$ represents retained superposition (no decisive observation within budget). Non-observation is not evidence against.
R3 (Update). Let $C \in [0, 1]$ be the global confidence. Update by a conservative rule

$$C^+ \;=\; \frac{W_{\text{prior}}\, C + W_{\text{obs}}(n)\, X}{W_{\text{prior}} + W_{\text{obs}}(n)}, \quad W_{\text{prior}} > 0,\; W_{\text{obs}}(n) \geq 0,$$

with W_{obs} small and mildly scale-aware. This guarantees $C^+ \geq \min\{C, 0.5\}$ and drifts upward whenever $X > C$.
R4 (Paradox and context). When a claim triggers a context clash (e.g., self-reference, indexicals, or conflicting evidence beyond the declared budget), classical, context-free evaluation is a category error. The proper response is to preserve neutrality (0.5) until a valid observation in the correct context is available.
R5 (Non-explosion by design). Contradictory local reports do not license arbitrary conclusions: we update only by the stated rule and never convert lack of observation into denial. Where neutrality never arises, classical consequence is recovered as a special case.

Aphorism.

Rigor $\neq$ Bivalence. Rigor $=$ Disciplined neutrality + auditable updates + paradox containment.

9 Future Work

- Refine growth factor scaling for both semiprime bumps and global weight.
- Explore alternative heuristics for p-reordering.
- Extend stress tests and measure confidence trajectories.

References

[1] J. R. Chen, *On the Representation of a Large Even Integer as the Sum of a Prime and the Product of at Most Two Primes*, Scientia Sinica 16 (1973), 157–176.

[2] *PEACE: Base logic and Neutral Policy.* Truth set $V = \{T, F, B\}$, designated $D = \{T, B\}$, content calculus, and neutral default B. (see "Base Logic: Meta-Dialetheic Core" and "Neutral Policy") [?].

[3] *PEACE: Category Error Detection.* If $\mathsf{CatErr}(\varphi, c)$, classical evaluation is a category error; switch to PEACE meta-evaluation. (see "Category Error Detection") [?].

[4] *PEACE: Theorems.* Non-explosion and classical preservation on B-free fragments. (see "Theorems") [?].

Appendix: Pseudocode for the Engine

Notation

Even input is n. Candidate odd primes are the list $P = \{3, 5, 7, 11, 13\}$. For each $p \in P$ we set $q \leftarrow n - p$. Confidence for a single n starts at 0.5 and evolves as below.

Algorithm 1 SmallPrimeLimitFor

function SmallPrimeLimitFor(n)
  if $n \le 10^6$ **then**
    return 500
  end if
  if $n \le 10^9$ **then**
    return 200
  end if
  if $n \le 10^{12}$ **then**
    return 50
  end if
  if $n \le 4 \cdot 10^{12}$ **then**
    return 20
  end if
  return 5
end function

Algorithm 2 GrowthFactor

function GrowthFactor(n, N_0, β, cap)
  if $n \le N_0$ **then**
    return 1
  end if
  $g \leftarrow \left(\dfrac{\log n}{\log N_0}\right)^{\beta}$
  if $g < 1$ **then**
    $g \leftarrow 1$
  end if
  if $g > cap$ **then**
    $g \leftarrow cap$
  end if
  return g
end function

Algorithm 3 IsPrimeWithBudget (deterministic MR + small trial division)

function IsPrimeWithBudget(q, L)
  if $q < 2$ **then**
    return false
  end if
  / divide by all primes $\le L$; if some divides, return false */*
  / run Miller–Rabin with fixed 64-bit bases; if passes, return true */*
  / else return false */*
end function

Algorithm 4 IsSemiprimeQuick (budgeted small split)

function IsSemiprimeQuick(q, n)
    $L \leftarrow$ SmallPrimeLimitFor(n)
    if IsPrimeWithBudget(q, L) **then**
        return false
    end if
    for each prime $p \leq L$ **do**
        if $p \cdot p > q$ **then**
            break
        end if
        if $q \bmod p = 0$ **then**
            return IsPrimeWithBudget($q/p, L$)
        end if
    end for
    return false ▷ no small split found; treat as Both upstream
end function

Algorithm 5 ReorderPList (prefer promising q first)

function ReorderPList(n, P)
    for each $p \in P$ **do**
        $q \leftarrow n - p$
        score(p) $\leftarrow$ count of $s \in \{3, 5, 7, 11\}$ with $s \mid q$ and $q \neq s$
    end for
    return P sorted by ascending *score*(p)
end function

Algorithm 6 ChenTestForEven

function ChenTestForEven(n)
  $C \leftarrow 0.5$ ▷ start at Both
  $h \leftarrow 0$ ▷ semiprime hits
  $P \leftarrow \{3, 5, 7, 11, 13\}$
  $P \leftarrow$ ReorderPList(n, P)
  for each p in P **do**
    $q \leftarrow n - p$
    if $q < 2$ **then**
      continue
    end if
    $L \leftarrow$ SmallPrimeLimitFor(n)
    if IsPrimeWithBudget(q, L) **then**
      return $(0.999, \textbf{true}, h, (p, q))$
    else if IsSemiprimeQuick(q, n) **then**
      $h \leftarrow h + 1$
      $\alpha \leftarrow$ GrowthFactor($n, N_0^{\text{semi}}, \beta_{\text{semi}}, cap_{\text{semi}}$)
      $C \leftarrow \min\{0.999,\ 0.5 + h \cdot (FUZZ\_INCREMENT \cdot \alpha)\}$
      if $h \geq 5$ **or** $C \geq 0.999$ **then**
        return $(C, \textbf{false}, h, \textbf{nil})$
      end if
    else
      continue ▷ remains Both
    end if
  end for
  return $(C, \textbf{false}, h, \textbf{nil})$
end function

Algorithm 7 RunEngineWeighted

function RunEngineWeighted(n_0, *steps*)
    $G \leftarrow 0.5$ ▷ global confidence
    $n \leftarrow n_0$;
    $jump \leftarrow 4$
    for $t = 1$ **to** *steps* **do**
        $(C_n, two, h, \pi) \leftarrow$ ChenTestForEven(n)
        if *ENABLE_W_CHEN_GROWTH* **then**
            $\omega \leftarrow$ GrowthFactor($n, N_0^{\mathrm{w}}, \beta_{\mathrm{w}}, cap_{\mathrm{w}}$)
            $W'_{\mathrm{Chen}} \leftarrow W_{\mathrm{Chen}} \cdot \omega$
        else
            $W'_{\mathrm{Chen}} \leftarrow W_{\mathrm{Chen}}$
        end if
        $G \leftarrow \dfrac{W_{\mathrm{GC}} \cdot G + W'_{\mathrm{Chen}} \cdot C_n}{W_{\mathrm{GC}} + W'_{\mathrm{Chen}}}$
        if *two* **then**
            $jump \leftarrow jump + random\_increment$ ▷ chaotic step
        end if
        $n \leftarrow n + jump$
    end for
    return sequence of (n, C_n, G)
end function