Mathematical Definition and Implementation of the Pratt-Conway Cellular Automaton

Documentation of PCCA Systems

January 17, 2026

1 Introduction

The Pratt-Conway Cellular Automaton (PCCA) is an arithmetic dynamical system that combines the number-theoretic structure of Pratt Trees with the computational logic of FRACTRAN. Unlike traditional cellular automata where cells are passive states, the PCCA is built on one fundamental pillar:

1. **Dual Role of States**: Every cell in the grid contains a natural number n, which serves simultaneously as an input value and as the generator for a canonical FRACTRAN program F_n.

2 Pratt Trees and FRACTRAN Programs

Definition 1 (Pratt Tree)**.** *For a prime number p, the Pratt Tree T_p is defined recursively:*

- *The root of the tree is p.*
- *The children of p are the Pratt Trees of the prime factors of $p-1$.*
- *The recursion terminates at the prime 2.*

Definition 2 (Bounded FRACTRAN Evaluation)**.** *Let $F = (f_1, \ldots, f_k)$ be a FRACTRAN program and let $a \in \mathbb{N}$. Define the bounded evaluation*

$$\mathrm{FracRun}_{t_{\max}}(F, a)$$

as follows:

1. *Set $N_0 = a$.*
2. *For $t = 0, 1, \ldots, t_{\max} - 1$:*
   - *Find the first fraction f_i such that $N_t \cdot f_i \in \mathbb{N}$.*
   - *If such f_i exists, set $N_{t+1} = N_t \cdot f_i$.*
   - *Otherwise halt and return N_t.*
3. *If no halt occurs within $t_{\max}$ steps, return $\perp$.*

We define the observable output

$$\Phi_{t_{\max}}(F, a) \;=\; \begin{cases} r, & \text{if } \mathrm{FracRun}_{t_{\max}}(F, a) = r \in \mathbb{N}, \\ 1, & \text{if } \mathrm{FracRun}_{t_{\max}}(F, a) = \perp. \end{cases}$$

Definition 3 (Pratt Forest and Multiplicity)**.** *For any natural number $n > 1$, the Pratt Forest is the collection of Pratt Trees for all prime factors of n. We denote $m_p(n)$ as the total count of the prime p appearing as a node within the entire forest of n.*

Definition 4 (Canonical FRACTRAN Mapping)**.** *To every $n \in \mathbb{N}$, we assign a canonical FRACTRAN program F_n. This program consists of a sequence of fractions:*

$$F_n = \left(\left(\frac{p-1}{p} \right)^{m_p(n)} \right) \text{ for all } p \leq n \text{ where } m_p(n) > 0$$

The fractions are ordered by the magnitude of the prime p.

3 Why canonical?: $F_n(n) = 1$

We record a basic but crucial normalization fact: each canonical program F_n sends its own index n to 1.

Theorem 1 (Self-normalization)**.** *For every integer $n \geq 1$, the canonical FRACTRAN program F_n halts on input n and returns 1. Equivalently,*

$$F_n(n) = 1 \quad \text{for all } n \geq 1.$$

Proof. If $n = 1$, then F_1 is the empty program and hence halts immediately with output 1.

Assume $n > 1$. For each prime p with $m_p(n) > 0$, the program F_n contains the fraction

$$f_p \;=\; \left(\frac{p-1}{p} \right)^{m_p(n)} \;=\; \frac{(p-1)^{m_p(n)}}{p^{m_p(n)}}.$$

Since $\gcd(p-1, p) = 1$, we have

$$\gcd\left((p-1)^{m_p(n)},\, p^{m_p(n)} \right) = 1.$$

Therefore, the FRACTRAN applicability condition for f_p on a positive integer N simplifies to

$$N \cdot f_p \in \mathbb{N} \quad \Longleftrightarrow \quad p^{m_p(n)} \mid N.$$

If applicable, the update is

$$N \;\mapsto\; N' \;=\; \frac{N}{p^{m_p(n)}}\,(p-1)^{m_p(n)}.$$

In particular, $N' < N$ because $0 < (p-1)^{m_p(n)} < p^{m_p(n)}$ for all primes p and all $m_p(n) \geq 1$. Hence every successful FRACTRAN step strictly decreases the current integer. This implies that the run must halt after finitely many steps.

It remains to show that the halting value cannot exceed 1. Let N be any value that occurs during the run (starting with $N = n$), and let p be the largest prime dividing N. Consider the situation when all primes $> p$ have already been eliminated (so p is currently maximal). By construction of the canonical mapping via Pratt forests, the exponent of p present at that stage is exactly $m_p(n)$: intuitively, each occurrence of p as a node in the Pratt forest corresponds to one required copy of p that must be "discharged" by the rule f_p, and all such copies are generated only from primes $\geq p$ (and once primes $> p$ are removed, no further sources of p remain). Consequently,

$$p^{m_p(n)} \mid N,$$

so f_p is applicable at that stage.

Moreover, applying any rule f_q with $q < p$ cannot create new prime factors $\geq p$ because the multiplier $(q-1)^{m_q(n)}$ only contains primes $< q < p$. Thus the maximal prime factor decreases whenever the rule for the current maximal prime is eventually applied. Since the run is strictly decreasing in N and cannot avoid applying the maximal-prime rule forever, the maximal prime factor must drop stepwise until only the prime 2 remains.

Finally, once N is a power of 2, the rule $f_2 = (1/2)^{m_2(n)}$ (which is present whenever $m_2(n) > 0$) removes blocks of 2-power without introducing any new primes, and strict decrease forces the terminal value to be 1. Therefore the computation halts with output 1, i.e. $F_n(n) = 1$. □

Remark. The key simplification above is that each fraction in F_n is reduced and has denominator $p^{m_p(n)}$ with $\gcd\big((p-1)^{m_p(n)}, p^{m_p(n)}\big) = 1$, so applicability depends only on divisibility by $p^{m_p(n)}$.

4 The Pratt-Conway Cellular Automaton Rules

The PCCA operates on a 2D grid $\mathbb{Z}_W \times \mathbb{Z}_H$ using a Moore neighborhood but it can also be implemented in van Neumann neighborhood.

Rule 1 (Local Update Rule)**.** *Let $n_t(x, y)$ be the value of cell (x, y) at time t. Let $F_{n_t(x,y)}$ be its associated canonical FRACTRAN program.*

For each neighbor $(x', y') \in \mathcal{N}(x, y)$ define

$$v_{x',y'} \;=\; \Phi_{t_{\max}}\big(F_{n_t(x,y)},\; n_t(x', y')\big)\,.$$

Then the next state of the cell is given by

$$n_{t+1}(x, y) \;=\; \max_{(x',y')\in\mathcal{N}(x,y)} v_{x',y'} \;+\; 2.$$

Interpretation. Each cell queries its neighbors for arithmetic information by running its own FRACTRAN program on the neighbors' values. Only the largest successfully computed result influences the update, and the constant offset +2 guarantees strict growth even in quiescent configurations.

The timeout fallback value 1 acts as a neutral element and prevents non-halting programs from dominating the dynamics.

- The automaton is deterministic and synchronous.
- Information flow is purely local; there is no directional memory.
- Canonical normalization ($F_n(n) = 1$) implies that homogeneous regions are dynamically stable up to the additive constant.