Field–Computon Algebra: A Measure–Theoretic and Geometric Framework for Computation as Evolving Fields

Abstract

We introduce *Field–Computon Algebra* (FCA), a mathematical and technological framework in which computation is represented as the evolution of measure fields over a spatial domain. The fundamental state of an FCA system is a time–varying measure μ_t defined on a domain $\Omega \subset \mathbb{R}^n$, interpreted as a distribution of "computational mass" or *computons*. Dynamics are governed by a continuity equation with source terms,

$$\partial_t \rho + \nabla \cdot (\rho v) = S(\rho),$$

where ρ is the density of μ_t with respect to a reference measure, v is a *meaning–velocity field*, and S encodes creation and annihilation of computons. We formalize FCA, specify a family of operators (potential–driven flow, coherence–inducing diffusion, resonant phase dynamics, and nonlocal interaction fields), and derive a discrete lattice implementation suitable for digital hardware and distributed networks. We show how Markov processes, consensus dynamics, message–passing on graphs, and standard neural network layers arise as special cases or discretizations of FCA. We further exhibit a Wasserstein gradient–flow interpretation for a class of FCA systems, connecting them to optimal transport and energy minimization. Finally, we provide a concrete algorithmic scheme—a "computon lattice engine"—that realizes FCA as a local update rule compatible with parallel and edge hardware. The goal is to provide a unified, physically interpretable substrate for computation where symbolic, neural, and field–based models coexist as different regimes of the same measure–theoretic evolution.

Contents

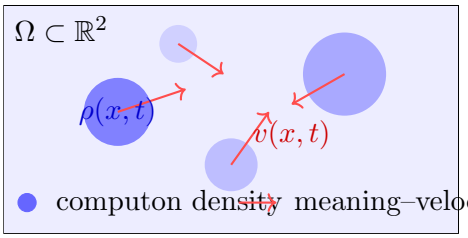


Figure 1: Informal picture of an FCA system: the density field $\rho(x,t)$ represents the distribution of computons over the domain Ω, and the vector field $v(x,t)$ transports this mass according to the continuity equation.

1 Introduction

Conventional models of computation typically adopt one of several paradigms:

- **Discrete symbolic systems** (Turing machines, term–rewriting systems, lambda calculus).
- **Vector–space mappings** (artificial neural networks).
- **Stochastic processes** (Markov chains, probabilistic programs).
- **Field equations** (partial differential equations, diffusion, wave equations).

These frameworks are often combined in practice, but there is no single underlying algebraic and dynamical structure that simultaneously accounts for:

- spatial distribution of state,
- probabilistic semantics,
- continuous or hybrid evolution in time,
- interaction and resonance between heterogeneous components.

Field–Computon Algebra (FCA) proposes such a structure. The core idea is to regard *computational state* as a measure over a domain, whose density we interpret as a field of "computational mass" or *computons*. Computation is then the evolution of this measure under constrained flows and source terms. Intuitively:

> FCA treats computation as the routing, deformation, and resonance of a conserved (or nearly conserved) mass of meaning across space, rather than as the manipulation of isolated symbols or static vectors.

This document formalizes FCA as follows:

(i) Section 2 introduces the measure–theoretic setup and defines computon fields.

(ii) Section sec:dynamics specifies the core continuity dynamics and the notion of an FCA system; we give basic existence and mass balance results.

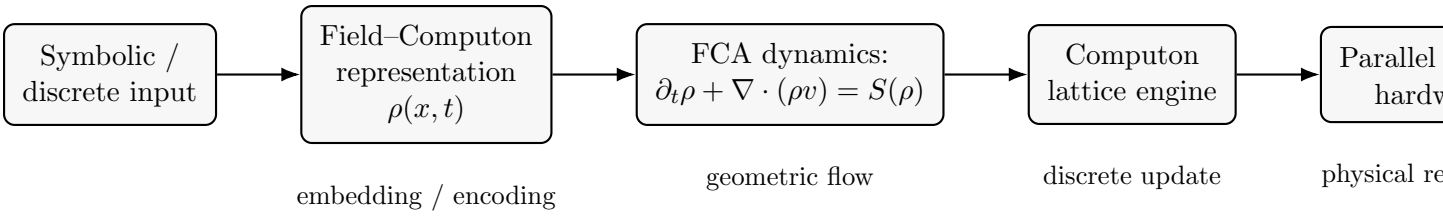


Figure 2: High–level FCA pipeline: symbolic or vector inputs are embedded into computon fields, which evolve under FCA dynamics, are discretized by a lattice engine, and executed on parallel or distributed hardware.

(iii) Section 4 defines several canonical FCA operator families (potential, coherence, resonance, and nonlocal interaction), emphasizing composability.

(iv) Section 5 provides a Wasserstein gradient–flow viewpoint for a subclass of FCA systems.

(v) Section 6 constructs a discrete lattice implementation and proves mass conservation at the discrete level.

(vi) Section 7 shows how Markov chains, consensus, message–passing, and neural layers arise as FCA instances.

(vii) Section 8 discusses expressivity, universality questions, and hybrid symbolic–field computation.

(viii) Section 9 gives an implementation appendix: a computon–lattice update algorithm with pseudo–code and explicit update rules.

(ix) Sections 10–13 add an operator calculus, stability and Lyapunov analysis, a worked micro–example, and a training framework for FCA.

(x) Section 14 summarizes extensions and open problems.

2 Measure–Theoretic Setup

2.1 Domain and base measure

Definition 2.1 (Computational domain)**.** *Let $(\Omega, \mathcal{B}(\Omega))$ be a measurable space, where $\Omega \subset \mathbb{R}^n$ is a bounded Borel set and $\mathcal{B}(\Omega)$ is its Borel σ–algebra. We refer to Ω as the* computational domain*.*

Definition 2.2 (Base measure)**.** *We fix a reference measure λ on $(\Omega, \mathcal{B}(\Omega))$, typically Lebesgue measure on $\mathbb{R}^n$ restricted to Ω:*

$$\lambda(A) = \int_A dx, \quad A \in \mathcal{B}(\Omega).$$

More generally, λ could be any positive Radon measure encoding geometry or density of spatial resources (e.g. a nonuniform medium).

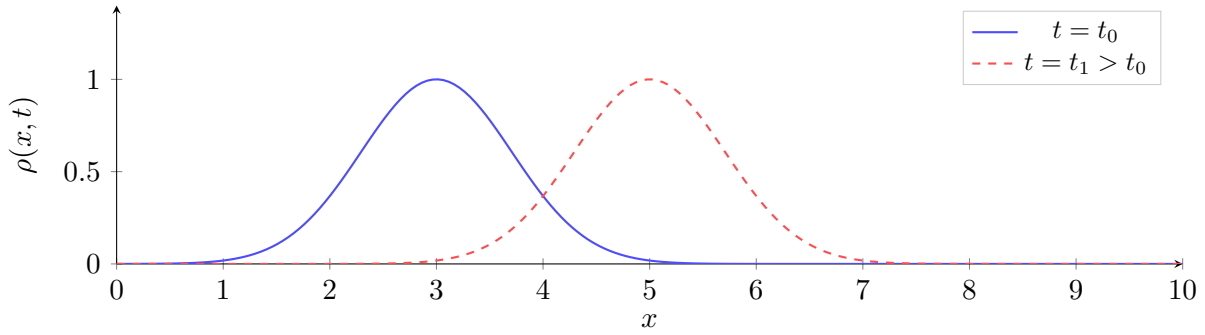


Figure 3: One–dimensional slice of a computon density field $\rho(x,t)$ at two times $t_0 < t_1$. The mass of meaning moves from left to right, representing computation as transport of measure rather than manipulation of isolated scalars.

2.2 Computon measure and density

Definition 2.3 (Computon measure)**.** *A* computon measure *is a family $(\mu_t)_{t\geq 0}$ of finite Borel measures on Ω, each absolutely continuous with respect to λ. That is, for each $t \geq 0$ there exists a density $\rho(\cdot,t) \in L^1(\Omega,\lambda)$ such that*

$$d\mu_t(x) = \rho(x,t)\, d\lambda(x).$$

We call $\rho(\cdot,t)$ the computon density field *at time t.*

Definition 2.4 (Total computon mass)**.** *The total mass at time t is given by*

$$M(t) := \mu_t(\Omega) = \int_\Omega \rho(x,t)\, d\lambda(x).$$

Remark 2.5. *One may normalize so that $M(t) = 1$ for all t, interpreting μ_t as a probability distribution over "where computation is currently located". Alternatively, $M(t)$ can be allowed to vary, representing allocation and depletion of computational resources.*

3 Core Dynamics and FCA Systems

3.1 Continuity equation with source

Definition 3.1 (Meaning–velocity field)**.** *A* meaning–velocity field *is a measurable vector field*

$$v : \Omega \times [0,\infty) \to \mathbb{R}^n,$$

interpreted as the velocity with which computon mass moves through the domain.

Definition 3.2 (Source term)**.** *A* source term *is a function*

$$S : \mathbb{R} \times \Omega \times [0,\infty) \to \mathbb{R}$$

such that $S(\rho,x,t)$ is measurable in (x,t) for each fixed ρ. It encodes local creation ($S > 0$) or annihilation ($S < 0$) of computons.

Definition 3.3 (FCA continuity equation)**.** *An* FCA continuity equation *is a partial differential equation for the density $\rho : \Omega \times [0,\infty) \to \mathbb{R}$ of the form*

$$\partial_t \rho(x,t) + \nabla \cdot \big(\rho(x,t)\, v(x,t)\big) = S\big(\rho(x,t), x, t\big), \tag{1}$$

to be interpreted in the weak (distributional) sense when necessary.

3.2 FCA system definition

Definition 3.4 (Field–Computon Algebra (FCA) system)**.** *An* FCA system *on domain* Ω *consists of:*

(a) a computon measure $(\mu_t)_{t\geq 0}$ *with density* $\rho(x,t)$*,*

(b) a meaning–velocity field $v(x,t)$*,*

(c) a source term $S(\rho,x,t)$*,*

(d) an initial density $\rho_0 \in L^1(\Omega,\lambda)$*,*

such that ρ *satisfies the continuity equation* (1) *with initial condition*

$$\rho(x,0)=\rho_0(x) \quad \text{for a.e. } x\in\Omega,$$

under suitable boundary conditions on $\partial\Omega$ *(e.g. no–flux, periodic, or prescribed inflow/outflow).*

Remark 3.5 (Weak formulation)**.** *For smooth test functions* $\varphi \in C_c^\infty(\Omega)$*, the weak form of* (1) *is*

$$\frac{d}{dt}\int_\Omega \varphi(x)\rho(x,t)\,d\lambda(x)=\int_\Omega \nabla\varphi(x)\cdot\big(\rho(x,t)v(x,t)\big)\,d\lambda(x)+\int_\Omega \varphi(x)S(\rho(x,t),x,t)\,d\lambda(x).$$

3.3 Mass balance and basic results

Lemma 3.6 (Total mass evolution)**.** *Assume* Ω *has boundary conditions such that the flux integral*

$$\int_{\partial\Omega}\rho v\cdot n\,d\sigma=0$$

(e.g. periodic domain or no–flux boundary), where n *is the outward normal and* $d\sigma$ *is surface measure. Then the time derivative of total mass satisfies*

$$\frac{d}{dt}M(t)=\int_\Omega S(\rho(x,t),x,t)\,d\lambda(x). \tag{2}$$

Proof. Integrate (1) over Ω:

$$\frac{d}{dt}\int_\Omega \rho\,d\lambda+\int_\Omega \nabla\cdot(\rho v)\,d\lambda=\int_\Omega S(\rho,x,t)\,d\lambda.$$

By the divergence theorem,

$$\int_\Omega \nabla\cdot(\rho v)\,d\lambda=\int_{\partial\Omega}\rho v\cdot n\,d\sigma=0,$$

so (2) follows. □

Corollary 3.7 (Mass conservation)**.** *If, in addition, the source term is* mass–neutral *in the sense that*

$$\int_\Omega S(\rho(x,t),x,t)\,d\lambda(x)=0$$

for all t*, then the total mass is conserved:*

$$M(t)=M(0)\quad \forall t\geq 0.$$

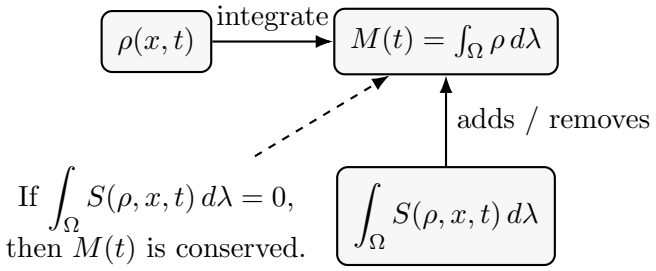


Figure 4: Mass balance in FCA: the density field ρ induces total mass $M(t)$. The integral of S over the domain acts as a global source or sink. If the source term is mass–neutral, total computon mass is conserved.

Assumption 3.8 (Regularity for basic well–posedness)**.** *In many of the examples below, we will implicitly assume:*

- Ω *is bounded with Lipschitz boundary,*
- v *is uniformly Lipschitz in* x *and bounded in* t*,*
- $S(\rho, x, t)$ *is locally Lipschitz in* ρ *and measurable in* (x, t)*.*

Theorem 3.9 (Local existence and uniqueness (informal))**.** *Under Assumption 3.8, for any* $\rho_0 \in L^1(\Omega, \lambda)$ *with* $\rho_0 \geq 0$*, there exists a time* $T > 0$ *and a unique nonnegative weak solution* $\rho \in C([0, T]; L^1(\Omega, \lambda))$ *to the FCA continuity equation* (1) *with initial data* ρ_0 *and no–flux or periodic boundary conditions.*

Remark 3.10. *A rigorous proof would rely on standard theory for transport equations with source, using fixed–point arguments or semigroup methods. In many FCA operator classes (e.g. potential–driven flows with smooth potentials), one can invoke existing well–posedness results for continuity or Fokker–Planck equations.*

4 Canonical FCA Operators

We now define several operator families that specify v and S, each motivated by a different computational intuition. FCA models are typically *compositions* of such operators.

4.1 Potential–driven flow

Definition 4.1 (Potential field)**.** *A* potential field *is a function* $\Phi : \Omega \to \mathbb{R}$*, assumed differentiable a.e. with gradient* $\nabla\Phi$*.*

Definition 4.2 (Potential FCA operator)**.** *Given a potential field* Φ *and a mobility* $\kappa > 0$*, the* potential FCA operator *is defined by choosing*

$$v(x,t) = -\kappa\, \nabla\Phi(x), \tag{3}$$

$$S(\rho, x, t) = 0. \tag{4}$$

The continuity equation becomes

$$\partial_t \rho = \nabla \cdot \big(\kappa\, \rho \nabla\Phi\big). \tag{5}$$

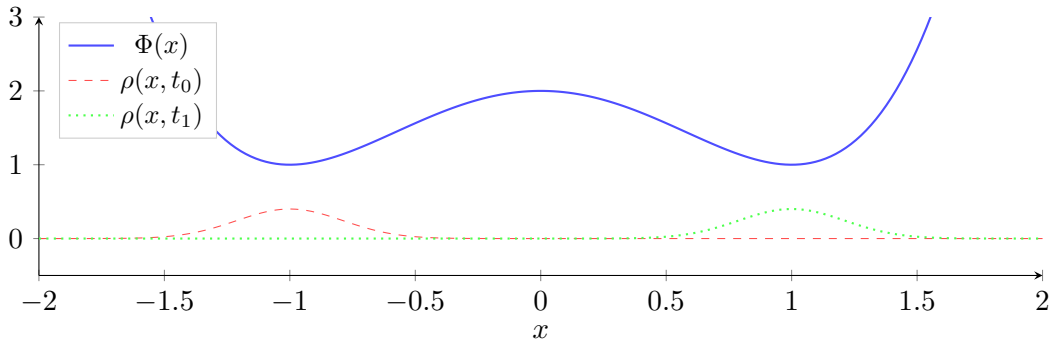


Figure 5: Potential FCA operator in 1D: the potential $\Phi(x)$ defines an energy landscape. Under the dynamics $\partial_t \rho = \nabla \cdot (\kappa \rho \nabla \Phi)$, mass initially concentrated in one well (red) can migrate toward a lower–energy configuration (green), implementing computation as relaxation in the potential.

Remark 4.3 (Energy interpretation)**.** *Define the energy functional*

$$\mathcal{E}_\Phi[\rho] := \int_\Omega \Phi(x)\rho(x)\, d\lambda(x).$$

Intuitively, the dynamics (5) *move* ρ *in the direction that decreases* $\mathcal{E}_\Phi$*, subject to conservation of mass. Section 5 makes this precise via Wasserstein gradient flows.*

4.2 Coherence–inducing diffusion

We introduce a source term that promotes coherence or alignment of the density field.

Definition 4.4 (Coherence FCA operator)**.** *Fix parameters* $\lambda_c > 0$ *and* $\alpha \in \mathbb{R}$*. The* coherence FCA operator *takes*

$$v(x,t) = 0, \tag{6}$$
$$S(\rho, x, t) = \lambda_c \left(\Delta\rho(x,t) - \alpha\, \rho(x,t)^3\right), \tag{7}$$

where Δ *is the Laplacian on* Ω *with appropriate boundary conditions. The FCA equation becomes*

$$\partial_t \rho = \lambda_c \left(\Delta\rho - \alpha\rho^3\right). \tag{8}$$

Remark 4.5 (Relation to reaction–diffusion)**.** *Equation* (8) *resembles a scalar reaction–diffusion equation with cubic reaction term. FCA interprets this as a mechanism that both smooths the computon field and nudges it towards discrete levels, supporting emergent clustering and consensus.*

4.3 Nonlocal interaction fields

To model attraction/repulsion between distant regions of the field, we introduce nonlocal interactions.

Definition 4.6 (Interaction kernel)**.** *An* interaction kernel *is a function* $K : \Omega \times \Omega \to \mathbb{R}^n$ *such that* $K(x, y)$ *is the contribution to the velocity at* x *from a unit mass at* y*.*

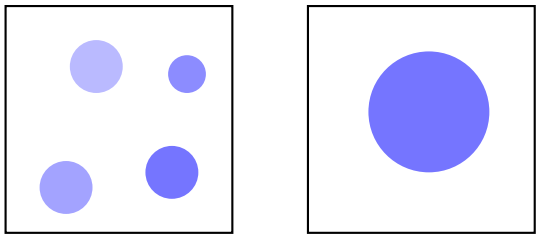


Figure 6: Coherence–inducing FCA dynamics: initially fragmented patches of computon density (left) can merge into a more coherent cluster (right) under diffusion plus nonlinear saturation, implementing a field–based consensus or clustering operation.

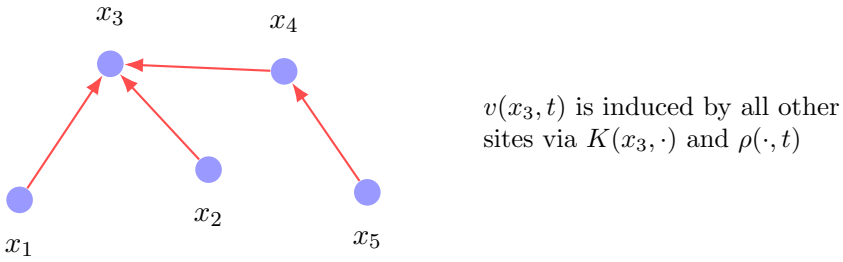


Figure 7: Nonlocal interaction FCA: each location experiences a velocity induced by all other locations via an interaction kernel $K(x, y)$. The figure shows mass at several points x_j inducing a net velocity at x_3.

Definition 4.7 (Nonlocal FCA operator)**.** *Given an interaction kernel K and scalar $\gamma > 0$, define*

$$v(x,t) = \gamma \int_\Omega K(x,y)\rho(y,t)\, d\lambda(y), \tag{9}$$

$$S(\rho, x, t) = 0. \tag{10}$$

The resulting continuity equation

$$\partial_t \rho + \nabla \cdot (\rho v) = 0 \tag{11}$$

describes mass moving under a self–induced velocity field.

4.4 Resonant phase dynamics

To model interference and oscillatory computation, we extend ρ to a complex–valued field with amplitude and phase.

Definition 4.8 (Resonant computon field)**.** *A* resonant computon field *is a complex–valued field*

$$\psi(x,t) = A(x,t)e^{i\theta(x,t)},$$

with real amplitude $A \geq 0$ and phase $\theta \in \mathbb{R}$, such that the computon density is

$$\rho(x,t) = |\psi(x,t)|^2 = A(x,t)^2.$$

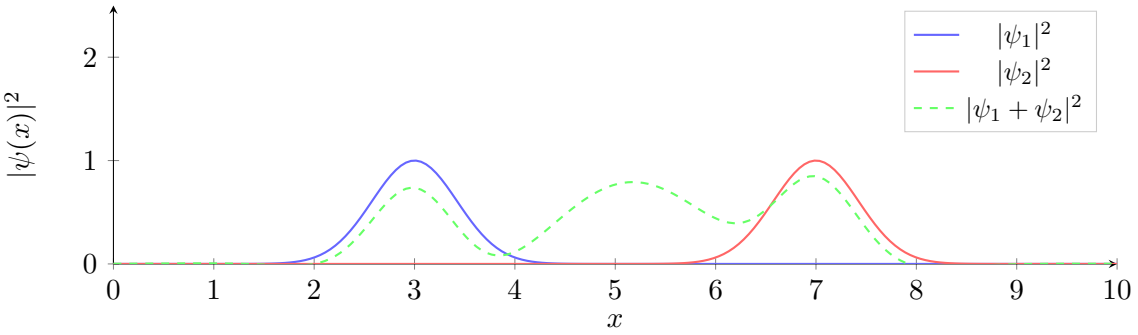


Figure 8: Resonant FCA: two localized wave packets (blue and red) interfere, producing a composite density (green) with oscillatory structure. Interference provides a mechanism for constructive and destructive combination of distributed computational hypotheses.

Definition 4.9 (Resonant FCA operator)**.** *Let $D > 0$, $\beta \in \mathbb{R}$, and $V : \Omega \to \mathbb{R}$ be given. The* resonant FCA operator *evolves ψ via*

$$\partial_t \psi = i\big(D\Delta\psi - V(x)\psi\big) - \beta|\psi|^2\psi, \tag{12}$$

with $\rho = |\psi|^2$ as the computon density.

Remark 4.10 (Current and velocity)**.** *Define the probability current*

$$J(x,t) := \Im\big(\overline{\psi}\,\nabla\psi\big).$$

Then ρ satisfies a continuity equation

$$\partial_t \rho + \nabla \cdot J = 0,$$

so $v = J/\rho$ (where $\rho > 0$) is the induced FCA velocity field. Interference patterns in ψ correspond to constructive/destructive redistribution of computon mass.

4.5 Compositional operators

FCA operators can be composed in time:

- apply potential flow for Δt_1,
- then coherence for Δt_2,
- then nonlocal interactions for Δt_3,
- etc.

This yields rich composite dynamics while maintaining a consistent measure–theoretic semantics.

5 Wasserstein Gradient–Flow View

For a subclass of FCA systems (especially potential–driven flows), the dynamics can be interpreted as gradient flows in the metric space of probability measures equipped with the 2–Wasserstein distance.

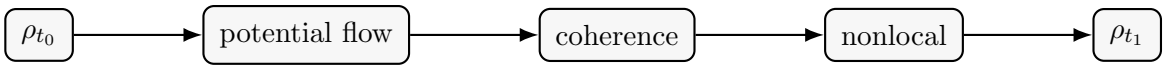


Figure 9: Compositional FCA dynamics: different operator families can be applied in sequence, each contributing a flow or source term.

5.1 Probability measures and Wasserstein distance

Let $\mathcal{P}_2(\Omega)$ denote the set of Borel probability measures on Ω with finite second moment. The 2–Wasserstein distance $W_2(\mu, \nu)$ quantifies the minimal transport cost to move mass from μ to ν.

Remark 5.1. *We assume here that $M(t)$ is constant and equal to 1, so that μ_t is a probability measure. This can be enforced by normalizing or by choosing mass–conserving FCA operators.*

5.2 Energy functionals

Definition 5.2 (Energy functional)**.** *An* energy functional *is a mapping $\mathcal{E} : \mathcal{P}_2(\Omega) \to \mathbb{R} \cup \{+\infty\}$, often of the form*

$$\mathcal{E}[\rho] = \int_\Omega \Phi(x)\rho(x)\, d\lambda(x) + \int_\Omega U(\rho(x))\, d\lambda(x) + \frac{1}{2}\int_{\Omega\times\Omega} W(x,y)\rho(x)\rho(y)\, d\lambda(x)d\lambda(y),$$

where Φ is an external potential, U is an internal energy density, and W encodes pairwise interactions.

5.3 Gradient–flow structure

Informally, a curve $(\mu_t)_{t\geq 0}$ in $\mathcal{P}_2(\Omega)$ is a gradient flow of $\mathcal{E}$ if it evolves in the direction of steepest descent of $\mathcal{E}$ with respect to the Wasserstein metric. For sufficiently regular $\mathcal{E}$, one can show that the corresponding density ρ satisfies a continuity equation of the FCA form.

Theorem 5.3 (FCA as Wasserstein gradient flow (informal))**.** *Consider the energy*

$$\mathcal{E}[\rho] = \int_\Omega \Phi(x)\rho(x)\, d\lambda(x),$$

with smooth Φ. Then under suitable regularity assumptions, the gradient flow of $\mathcal{E}$ in the 2–Wasserstein metric has density ρ solving

$$\partial_t \rho = \nabla \cdot \big(\rho \nabla \Phi\big),$$

which is a special case of the potential FCA equation (5) with $\kappa = 1$ and $S \equiv 0$.

Remark 5.4. *More general energies with internal and interaction terms lead to drift–diffusion and nonlocal FCA operators. This connects FCA to a large body of work on gradient flows in the Wasserstein space, giving access to tools for existence, uniqueness, and convergence to equilibrium.*

6 Discrete FCA: Lattice Implementation

We now construct a discrete version of FCA suitable for implementation on digital hardware or distributed networks.

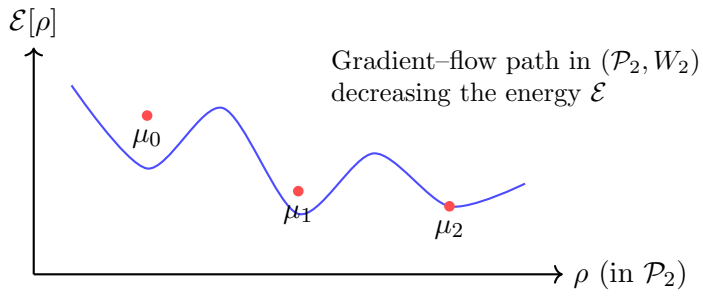


Figure 10: Wasserstein gradient–flow viewpoint: a measure μ_t moves along a path of steepest descent in the space of probability measures with respect to the Wasserstein metric, reducing the energy functional $\mathcal{E}[\rho]$. Potential FCA operators realize such flows for suitable energies.

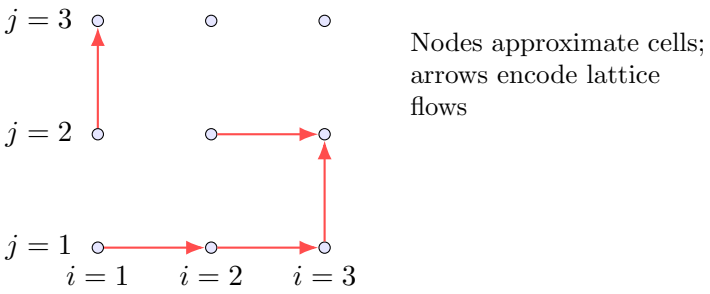


Figure 11: Discrete FCA on a lattice: each node stores a density value ρ_i^k, and directed edges represent possible flow between nodes, encoded in the flow matrix F. Updates are local and parallelizable.

6.1 Lattice and neighborhood

Definition 6.1 (Computon lattice)**.** *Let $G = (V, E)$ be a finite directed or undirected graph with vertex set $V = \{1, \dots, N\}$ and edges $E \subseteq V \times V$. We call G a* computon lattice*. Each node $i \in V$ corresponds to a cell in the continuous domain, and edges represent adjacency or direct communication.*

Definition 6.2 (Discrete computon state)**.** *A* discrete computon state *at time step $k \in \mathbb{N}$ is a vector*

$$\boldsymbol{\rho}^k = (\rho_1^k, \dots, \rho_N^k)^\top \in \mathbb{R}^N_{\geq 0},$$

where ρ_i^k approximates the mass at node i. The total mass at time k is

$$M^k = \sum_{i=1}^{N} \rho_i^k.$$

6.2 Flow matrix and source

Definition 6.3 (Flow matrix)**.** *A* flow matrix *is an $N \times N$ matrix $F = (F_{ij})$ such that:*

(a) $F_{ij} \geq 0$ for $i \neq j$ represents the per–unit–time *fraction of mass flowing from node j to node i;*

(b) diagonal entries are defined by $F_{ii} = -\sum_{j \neq i} F_{ji}$.

Remark 6.4. *The column sums of F satisfy*

$$\sum_i F_{ij} = 0 \quad \forall j,$$

which encodes local mass conservation of the flow component.

Definition 6.5 (Discrete source map)**.** *A* source map *is a function $S : \mathbb{R}^N \to \mathbb{R}^N$, possibly nonlinear, that acts nodewise or with short–range coupling:*

$$S(\boldsymbol{\rho}) = (S_1(\boldsymbol{\rho}), \ldots, S_N(\boldsymbol{\rho}))^\top.$$

6.3 Discrete update rule

Definition 6.6 (Discrete FCA update)**.** *Let F be a flow matrix and $S : \mathbb{R}^N \to \mathbb{R}^N$ a source map. For time step $\Delta t > 0$, the discrete FCA update is*

$$\boldsymbol{\rho}^{k+1} = \boldsymbol{\rho}^k + \Delta t \left(F\boldsymbol{\rho}^k + S(\boldsymbol{\rho}^k) \right). \tag{13}$$

Lemma 6.7 (Mass evolution in the discrete system)**.** *Let $\mathbf{1}$ denote the all–ones vector in $\mathbb{R}^N$, and suppose that the total source is mass–neutral, i.e.*

$$\sum_{i=1}^N S_i(\boldsymbol{\rho}) = 0 \quad \forall \boldsymbol{\rho} \in \mathbb{R}^N.$$

Then the total mass evolves as

$$M^{k+1} = M^k.$$

Proof. Compute

$$M^{k+1} - M^k = \sum_i (\rho_i^{k+1} - \rho_i^k) = \Delta t \sum_i \big((F\boldsymbol{\rho}^k)_i + S_i(\boldsymbol{\rho}^k) \big).$$

Using $\sum_i (F\boldsymbol{\rho}^k)_i = \mathbf{1}^\top F\boldsymbol{\rho}^k$ and column–sum–zero of F,

$$\mathbf{1}^\top F\boldsymbol{\rho}^k = (\mathbf{1}^\top F)\boldsymbol{\rho}^k = 0.$$

By the mass–neutrality of S we also have $\sum_i S_i(\boldsymbol{\rho}^k) = 0$, hence $M^{k+1} = M^k$. □

6.4 Relation to continuous FCA

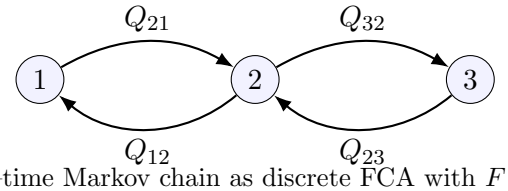
The discrete update (13) is an explicit Euler step for a system of ODEs

$$\frac{d}{dt}\boldsymbol{\rho}(t) = F\boldsymbol{\rho}(t) + S(\boldsymbol{\rho}(t)),$$

which can be derived from the continuous FCA equation by finite–volume or finite–difference discretization on a regular grid or more general tessellations. Each node approximates the density integrated over a cell, and each matrix entry F_{ij} corresponds to flux across a cell interface.

7 Existing Models as FCA Instances

We now illustrate how several familiar computational models emerge as special cases of FCA.



Continuous–time Markov chain as discrete FCA with $F = Q$, $S \equiv 0$.

Figure 12: Markov chains as FCA systems: the generator Q is a flow matrix F whose off–diagonal entries represent transition rates, and diagonal entries enforce column–sum zero.

7.1 Markov chains and diffusion

Example 7.1 (Continuous–time Markov chain)**.** *Let Q be the generator of a continuous–time Markov chain on a finite state space $V = \{1, \dots, N\}$, with off–diagonal entries $Q_{ij} \geq 0$ for $i \neq j$ and $Q_{ii} = -\sum_{j \neq i} Q_{ij}$. Let $\boldsymbol{p}(t)$ be the probability vector evolving under*

$$\frac{d}{dt}\boldsymbol{p}(t) = Q\boldsymbol{p}(t).$$

This is a special case of the discrete FCA system with

$$F = Q, \qquad S \equiv 0.$$

The total mass $M(t) = \sum_i p_i(t)$ is conserved and equals 1 for all t.

Example 7.2 (Discrete diffusion)**.** *On a regular grid graph, let $F = -L$ where L is the graph Laplacian. Then*

$$\frac{d}{dt}\boldsymbol{\rho}(t) = -L\boldsymbol{\rho}(t)$$

is a discrete diffusion equation. This is an FCA system with pure flow and no source.

7.2 Consensus and distributed averaging

Example 7.3 (Consensus dynamics)**.** *On a connected graph G, a standard consensus protocol evolves node values according to*

$$\frac{d}{dt}\boldsymbol{\rho}(t) = -L\boldsymbol{\rho}(t),$$

where $L = D - A$ is the graph Laplacian. This is a mass–conserving FCA system with $F = -L$ and $S \equiv 0$. Under mild connectivity conditions, $\boldsymbol{\rho}(t)$ converges to a consensus where all nodes share the same value, equal to the average of the initial state. This can be viewed as an FCA system whose unique steady state is the globally coherent distribution.

7.3 Graph message passing and GNN layers

Example 7.4 (Linear message passing)**.** *Consider a graph $G = (V, E)$ with adjacency matrix A and degree matrix D. A standard linear message–passing layer for a graph neural network takes the form*

$$\boldsymbol{h}' = \tilde{A}\boldsymbol{h}W,$$

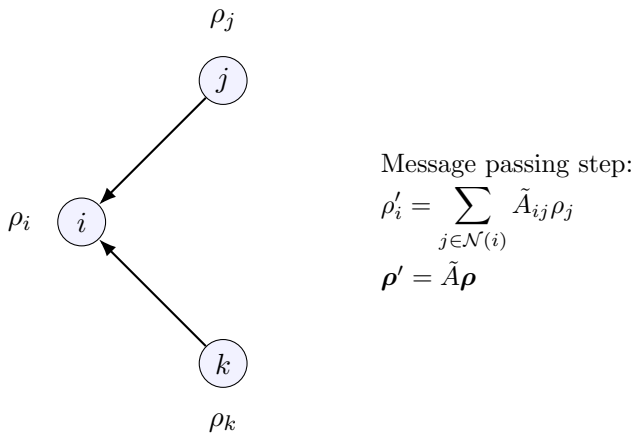


Figure 13: Graph message passing as FCA flow: incoming messages from neighbors correspond to mass flowing along edges into node i, described by a flow matrix derived from the normalized adjacency $\tilde{A}$.

where $\tilde{A}$ is some normalized adjacency (e.g. $D^{-1}A$ or $D^{-1/2}AD^{-1/2}$), $\boldsymbol{h}$ is a matrix of node features, and W is a learnable weight matrix. If we restrict to scalar features (one dimension) and identify ρ_i with the scalar feature at node i, then (ignoring W for a moment) the map

$$\boldsymbol{\rho}' = \tilde{A}\boldsymbol{\rho}$$

can be viewed as the time–Δt map of an FCA system with

$$F = \tilde{A} - I, \qquad S \equiv 0,$$

and $\Delta t = 1$. Nonlinearities such as ReLU or sigmoid can be incorporated into a source term $S(\boldsymbol{\rho})$ that acts nodewise.

Remark 7.5. *Graph neural network layers correspond to discrete FCA steps that combine linear mass redistribution (through $\tilde{A}$) with nodewise creation/annihilation of computon mass via nonlinearity. The FCA formalism makes the conservation or non–conservation of total mass explicit and controllable.*

7.4 Neural network layers as FCA compositions

Example 7.6 (Neural layer as potential flow plus source)**.** *Consider a feedforward layer*

$$\boldsymbol{y} = \sigma(W\boldsymbol{x} + \boldsymbol{b}),$$

where σ is a pointwise nonlinearity. We can interpret $\boldsymbol{x}$ as a discrete density over an input domain, and W as a linear map induced by a potential–driven FCA step on an extended domain, followed by a local source term representing σ. Formally, let $\boldsymbol{\rho}^0 = \boldsymbol{x}$ and define

$$\boldsymbol{\rho}^{1/2} = \boldsymbol{\rho}^0 + \Delta t F \boldsymbol{\rho}^0,$$

with F chosen so that $\boldsymbol{\rho}^{1/2} \approx W\boldsymbol{x}$. Then apply a nodewise source

$$\boldsymbol{\rho}^1 = \boldsymbol{\rho}^{1/2} + \Delta t\, S(\boldsymbol{\rho}^{1/2}),$$

with $S_i(\boldsymbol{\rho}^{1/2}) = \sigma(\rho_i^{1/2}) - \rho_i^{1/2}$. The composite map $\boldsymbol{x} \mapsto \boldsymbol{\rho}^1$ reproduces the neural layer. Thus standard neural layers embed into the FCA class as discrete approximations of potential–driven flow plus source.

7.5 Probabilistic programs as FCA

Example 7.7 (Branching and conditioning)**.** *Probabilistic programs alternately sample and condition on random variables. In FCA, sampling corresponds to introducing new mass in regions consistent with the sampled outcome, while conditioning corresponds to reweighting or renormalizing ρ to a subset of Ω. Both are realized via appropriate source terms $S(\rho, x, t)$, possibly discontinuous in t, coupled with normalization. Composed over time, this yields a field version of probabilistic inference.*

8 Expressivity and Hybrid Computation

8.1 Approximation capabilities

A central question is: what maps can FCA systems approximate?

Problem 8.1 (Expressivity of FCA)**.** *Given a function F between suitable function spaces (e.g. $F : L^1(\Omega) \to L^1(\Omega')$), under what conditions does there exist an FCA system (continuous or discrete, possibly with learned parameters) such that the time–T map $\rho_0 \mapsto \rho_T$ approximates F to within a prescribed error?*

We do not solve this problem fully here, but note:

- Neural networks are already known to be universal approximators of continuous functions on compact domains.
- Neural networks embed into FCA as shown above.
- Therefore FCA inherits at least the expressivity of neural networks.

Proposition 8.2 (FCA is at least as expressive as feedforward networks)**.** *Any function approximable by a finite feedforward neural network with standard activations can also be approximated by a discrete FCA system with suitable flow matrix F, source map S, and finite number of time steps.*

Remark 8.3. *The converse is not obvious: FCA systems can express dynamics with rich spatial and temporal structure (including PDE–like behavior and resonance) that are not naturally captured by shallow networks. A formal comparison of expressive power remains an open research direction.*

8.2 Hybrid symbolic–field architectures

Symbols can be encoded in FCA as concentrated packets of mass in designated regions (or as discrete labels attached to nodes in a lattice). Operations such as reading and writing can be implemented as routing and gating rules for computon mass. This suggests hybrid architectures where:

- high–level, symbolic operations orchestrate the structure of the domain, potentials, and interaction kernels,
- low–level FCA dynamics perform robust, analog–like inference and aggregation in continuous space.

9 Implementation Appendix: Computon Lattice Engine

We now outline a concrete implementation pattern for discrete FCA suitable for software or hardware execution.

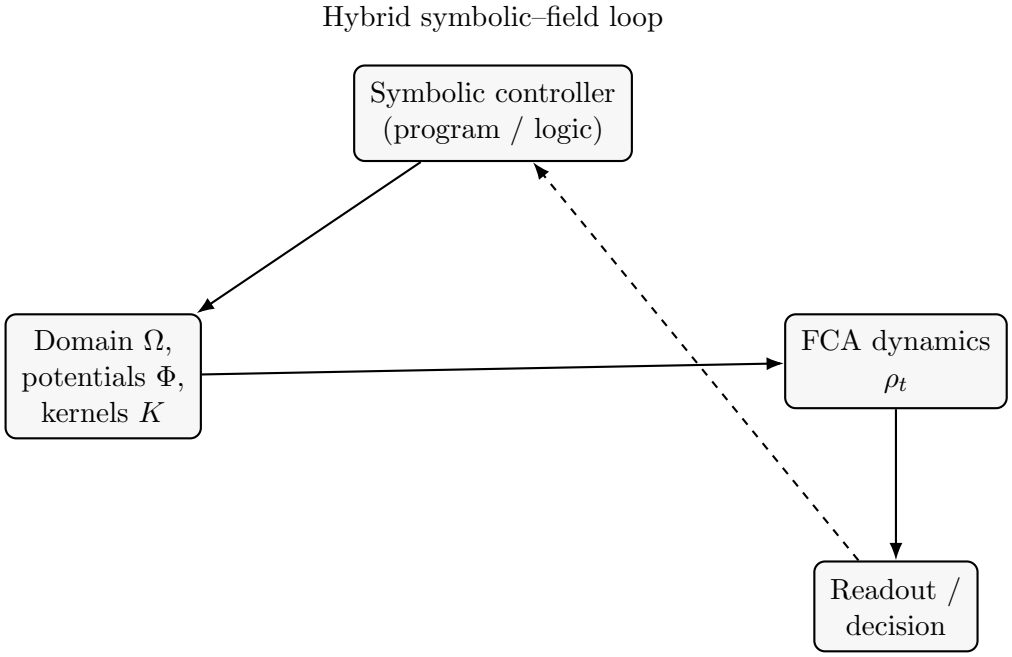


Figure 14: Hybrid architecture: symbolic modules configure the FCA substrate (domain, potentials, interaction kernels), FCA dynamics perform distributed computation in the field space, and readouts feed back to symbolic control.

9.1 Discrete state and operators

Assume:

- A graph $G = (V, E)$ with $|V| = N$.
- Node positions $x_i \in \mathbb{R}^n$ (optional, for geometric operators).
- Node state $\rho_i^k \geq 0$ at time step k.
- A set of operator modules, each specifying a contribution to flow and/or source.

We write:

$$\boldsymbol{\rho}^k = (\rho_1^k, \ldots, \rho_N^k)^\top.$$

9.2 Operator modules

Each module m has:

- a flow contribution $F^{(m)} \in \mathbb{R}^{N \times N}$,
- a source contribution $S^{(m)} : \mathbb{R}^N \to \mathbb{R}^N$.

The total flow and source at step k are

$$F_{\text{tot}} = \sum_m F^{(m)}, \qquad S_{\text{tot}}(\boldsymbol{\rho}^k) = \sum_m S^{(m)}(\boldsymbol{\rho}^k).$$

Algorithm 1 Computon Lattice FCA Update (Single Step)

Require: Current state $\boldsymbol{\rho}^k \in \mathbb{R}^N_{\geq 0}$, flow matrices $\{F^{(m)}\}_m$, source maps $\{S^{(m)}\}_m$, step size $\Delta t > 0$.
Ensure: Next state $\boldsymbol{\rho}^{k+1}$.

1: Compute total flow matrix: $F_{\text{tot}} \leftarrow \sum_m F^{(m)}$.
2: Compute total source: $S_{\text{tot}}(\boldsymbol{\rho}^k) \leftarrow \sum_m S^{(m)}(\boldsymbol{\rho}^k)$.
3: Update:

$$\boldsymbol{\rho}^{k+1} \leftarrow \boldsymbol{\rho}^k + \Delta t \left(F_{\text{tot}} \boldsymbol{\rho}^k + S_{\text{tot}}(\boldsymbol{\rho}^k) \right).$$

4: (Optional) Enforce nonnegativity: $\rho_i^{k+1} \leftarrow \max(0, \rho_i^{k+1})$ for all i.
5: (Optional) Renormalize total mass if required:

$$\boldsymbol{\rho}^{k+1} \leftarrow \frac{M_{\text{target}}}{\sum_i \rho_i^{k+1}} \boldsymbol{\rho}^{k+1}.$$

6: **return** $\boldsymbol{\rho}^{k+1}$.

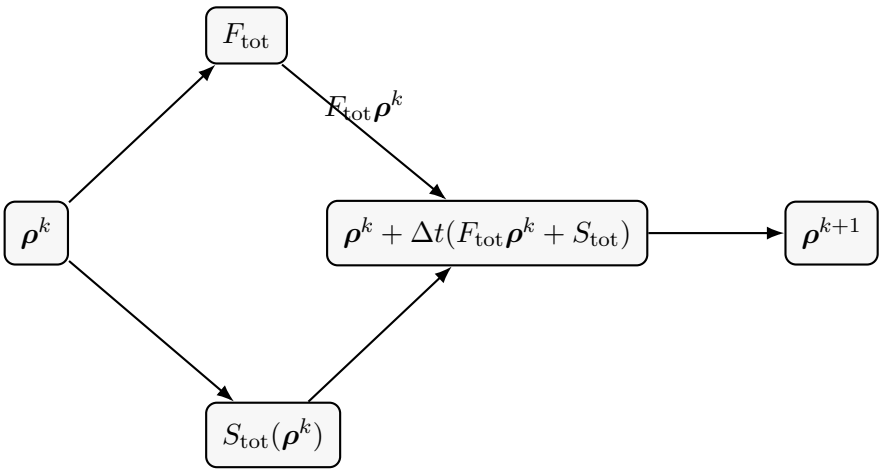


Figure 15: Computation graph for a single FCA lattice update: the current density $\boldsymbol{\rho}^k$ flows through flow and source modules, then combines in an explicit Euler step to produce $\boldsymbol{\rho}^{k+1}$.

9.3 Update scheme

9.4 Example: potential + coherence on a grid

On a 2D grid with spacing h, for node i at position x_i, we can discretize:

- potential–driven flow using a discrete gradient of $\Phi(x)$, mapping to directed flows between neighbors;
- coherence diffusion using the graph Laplacian L of the grid (with suitable weights).

Then:

$$F^{(\text{potential})} \approx \text{skew–symmetric flow derived from } \nabla\Phi,$$

$$F^{(\text{coherence})} = -\lambda_c L, \quad S_i^{(\text{coherence})}(\boldsymbol{\rho}) = -\lambda_c \alpha(\rho_i)^3.$$

The update (13) then gives an explicit scheme approximating the combined PDE.

9.5 Parallel and distributed execution

Because each row of F and each component S_i typically depends only on local neighbors, Algorithm 1:

- parallelizes naturally across nodes on GPUs or multicore CPUs;
- can be distributed across physical devices, each owning a subset of nodes and communicating only with neighbors.

10 Operator Calculus and Semigroup Structure

We now add an operator–theoretic layer on top of FCA, useful for analysis and for designing complex architectures by composition.

10.1 Linear FCA operators

Consider a linear FCA system on a lattice, i.e. $S(\boldsymbol{\rho}) = B\boldsymbol{\rho}$ for some matrix $B \in \mathbb{R}^{N\times N}$, and a linear flow matrix $F \in \mathbb{R}^{N\times N}$. The ODE

$$\frac{d}{dt}\boldsymbol{\rho}(t) = (F + B)\boldsymbol{\rho}(t)$$

has solution

$$\boldsymbol{\rho}(t) = e^{t(F+B)}\boldsymbol{\rho}(0),$$

where $e^{t(F+B)}$ is the matrix exponential.

Definition 10.1 (FCA generator)**.** *For such a linear FCA system, the matrix*

$$L := F + B$$

is called the FCA generator*. The associated semigroup $(T_t)_{t\geq 0}$ is*

$$T_t := e^{tL}.$$

Remark 10.2. *In continuous space, an analogous generator is the (formal) differential operator*

$$\mathcal{L}\rho = -\nabla \cdot (\rho v) + S(\rho),$$

whose semigroup $(\mathcal{T}_t)_{t\geq 0}$ acts on density fields. FCA can be viewed as choosing classes of such generators with geometric meaning and hardware–friendly discretizations.

10.2 Composition and Trotter splitting

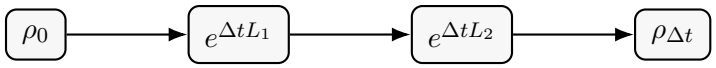
When $L = L_1 + L_2$ can be decomposed into simpler generators (e.g. potential and coherence parts), we can approximate e^{tL} by product formulas.

Proposition 10.3 (First–order Trotter splitting)**.** *Suppose $L = L_1 + L_2$ generates a strongly continuous semigroup. Then, under appropriate domain conditions,*

$$e^{tL} = \lim_{n\to\infty} \left(e^{\frac{t}{n}L_1} e^{\frac{t}{n}L_2}\right)^n,$$

with convergence in the strong operator topology.

In discrete FCA, this corresponds to alternately applying operator modules corresponding to L_1 and L_2 in small time steps.



Approximate $e^{t(L_1+L_2)}$ by repeating this block n times with $\Delta t = t/n$.

Figure 16: Operator–splitting view: complex FCA generators can be approximated by alternating simpler operator blocks $e^{\Delta t L_1}$ and $e^{\Delta t L_2}$.

10.3 Nonlinear FCA operators as flows on state space

For nonlinear FCA operators (e.g. cubic sources), we still think of FCA as defining a flow Φ_t on the state space:

$$\Phi_t : \rho_0 \mapsto \rho_t.$$

These flows compose:

$$\Phi_{t+s} = \Phi_t \circ \Phi_s.$$

Definition 10.4 (FCA operator algebra)**.** *Let $\mathcal{O}$ be a set of FCA operator modules O_i, each defining a flow Φ_t^i. The* FCA operator algebra *is the closure under composition and time–concatenation of these flows:*

$$\mathsf{Alg}(\mathcal{O}) := \left\{ \Phi_{t_k}^{i_k} \circ \cdots \circ \Phi_{t_1}^{i_1} : k \in \mathbb{N},\, i_j \in \mathcal{O},\, t_j \geq 0 \right\}.$$

The design space for FCA architectures is thus an operator algebra generated by a small set of interpretable modules (potential, coherence, nonlocal, resonance, etc.).

11 Stability, Invariants, and Lyapunov Functionals

We briefly analyze stability properties of linear mass–conserving FCA systems and show how energy or entropy–like functionals can serve as Lyapunov functions.

11.1 Linear symmetric flows

Consider a discrete FCA system with $S \equiv 0$ and flow matrix F symmetric negative semidefinite:

$$F = F^\top, \qquad \boldsymbol{x}^\top F \boldsymbol{x} \leq 0 \quad \forall \boldsymbol{x} \in \mathbb{R}^N.$$

This includes, for example, $F = -L$ where L is a symmetric graph Laplacian.

Proposition 11.1 (Decay of quadratic energy)**.** *Let $\boldsymbol{\rho}(t)$ solve $\frac{d}{dt}\boldsymbol{\rho}(t) = F\boldsymbol{\rho}(t)$ with $F = F^\top \preceq 0$. Define*

$$E(t) := \frac{1}{2}\|\boldsymbol{\rho}(t)\|_2^2.$$

Then

$$\frac{d}{dt}E(t) = \boldsymbol{\rho}(t)^\top F \boldsymbol{\rho}(t) \leq 0.$$

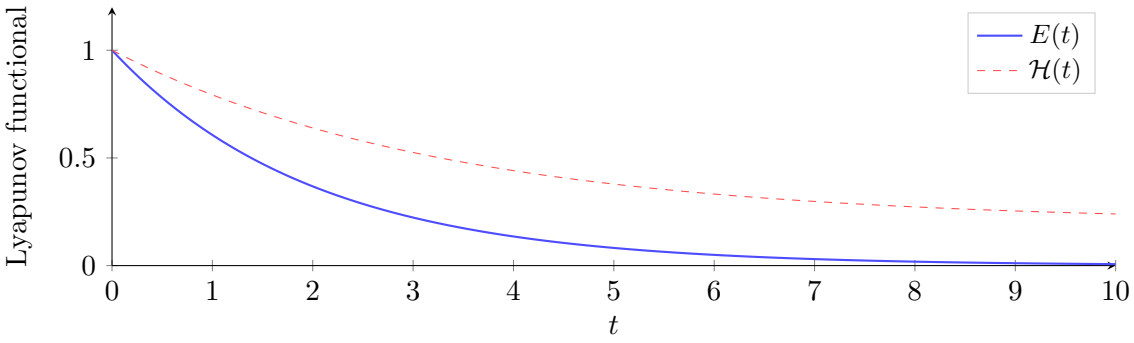


Figure 17: Typical behavior of Lyapunov functionals (quadratic energy $E(t)$ and entropy $\mathcal{H}(t)$) under stable FCA dynamics: both are nonincreasing, constraining long–time behavior and convergence.

Proof. Differentiate:

$$\frac{d}{dt}E(t) = \left\langle \boldsymbol{\rho}(t), \frac{d}{dt}\boldsymbol{\rho}(t) \right\rangle = \langle \boldsymbol{\rho}(t), F\boldsymbol{\rho}(t) \rangle = \boldsymbol{\rho}(t)^\top F \boldsymbol{\rho}(t) \leq 0.$$

□

Thus $E(t)$ is a Lyapunov functional, proving stability of the zero solution and characterizing convergence to the kernel of F (e.g. consensus subspace for Laplacians).

11.2 Mass and entropy invariants

For mass–conserving FCA systems with $S \equiv 0$ and column–sum–zero flow matrices, we already saw $M(t)$ is invariant. For continuous FCA, one can define entropy functionals such as

$$\mathcal{H}[\rho] := \int_\Omega \rho(x) \log \rho(x)\, d\lambda(x),$$

and show, for suitable diffusion–like flows, that $\mathcal{H}$ decreases in time, providing another Lyapunov functional.

12 Worked Micro–Example: Three–Node FCA Computer

To make FCA concrete, we construct a tiny three–node FCA system that computes the average of three inputs and show its exact dynamics.

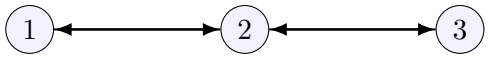
12.1 Setup

Let G be a chain of three nodes $V = \{1, 2, 3\}$ with edges $(1, 2)$ and $(2, 3)$. Let the flow matrix F be the (negative) Laplacian:

$$F = -L = \begin{pmatrix} -1 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -1 \end{pmatrix}.$$

Consider the ODE

$$\frac{d}{dt}\boldsymbol{\rho}(t) = F\boldsymbol{\rho}(t), \qquad \boldsymbol{\rho}(0) = \begin{pmatrix} a \\ b \\ c \end{pmatrix}.$$



$$\rho_1(0) = a \qquad \rho_2(0) = b \qquad \rho_3(0) = c$$

$$\rho_i(t) \to \tfrac{a+b+c}{3} \text{ for } i = 1, 2, 3$$

Figure 18: Three–node FCA system computing an average via consensus dynamics.

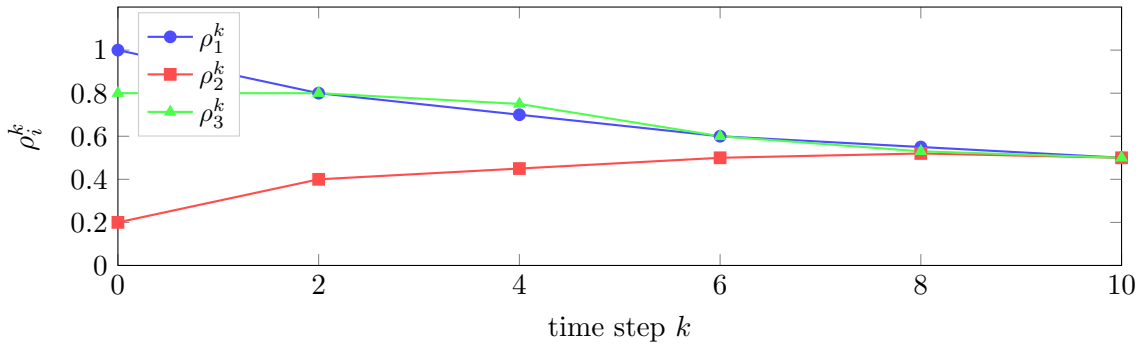


Figure 19: Qualitative discrete evolution of the three–node FCA example: all node densities converge to a common average value.

12.2 Behavior

The matrix F is symmetric negative semidefinite, with a one–dimensional kernel spanned by $\mathbf{1} = (1, 1, 1)^\top$. Thus:

- $M(t) = \rho_1 + \rho_2 + \rho_3$ is conserved;
- the system converges to consensus:

$$\boldsymbol{\rho}(t) \to \bar{\rho}\,\mathbf{1}, \quad \bar{\rho} = \frac{a+b+c}{3}.$$

So this tiny FCA system realizes an averaging computation purely via mass flow.

12.3 Discrete Euler simulation

Using the discrete update

$$\boldsymbol{\rho}^{k+1} = \boldsymbol{\rho}^k + \Delta t\, F \boldsymbol{\rho}^k,$$

with sufficiently small Δt, we approximate the continuous dynamics.

13 Learning FCA Operators from Data

Finally, we sketch how FCA operator modules can be learned from data, analogously to learning weights in a neural network.

13.1 Parametric FCA modules

Let θ denote a vector of parameters controlling:

- potentials $\Phi_\theta(x)$;
- interaction kernels $K_\theta(x, y)$;
- discrete flow matrices F_θ;
- source maps $S_\theta(\boldsymbol{\rho})$.

For fixed θ, the FCA dynamical system defines a mapping

$$\mathcal{F}_\theta : \rho_{\text{in}} \mapsto \rho_{\text{out}}$$

by integrating the dynamics from $t = 0$ to $t = T$ (or running K lattice steps).

Definition 13.1 (FCA model)**.** *Given a dataset of pairs* $(\rho_{in}^{(n)}, \rho_{target}^{(n)})$*, an* FCA model *is a parametric family* $\mathcal{F}_\theta$ *fitted by minimizing a loss*

$$\mathcal{L}(\theta) = \sum_n \ell\big(\mathcal{F}_\theta(\rho_{in}^{(n)}), \rho_{target}^{(n)}\big),$$

where ℓ *measures mismatch (e.g.* L^2 *or KL divergence).*

13.2 Training loop

For differentiable modules, gradients $\nabla_\theta \mathcal{L}$ can be computed via backpropagation through time (discrete case) or adjoint methods (continuous case).

Algorithm 2 Gradient–based Training of an FCA Model (Discrete Case)

Require: Dataset $\{(\boldsymbol{\rho}_{\text{in}}^{(n)}, \boldsymbol{\rho}_{\text{target}}^{(n)})\}_n$, initial parameters θ, learning rate η, number of lattice steps K.

while not converged **do**
  Sample a minibatch $\mathcal{B}$.
  For each $(\boldsymbol{\rho}_{\text{in}}, \boldsymbol{\rho}_{\text{target}}) \in \mathcal{B}$:
    Set $\boldsymbol{\rho}^0 \leftarrow \boldsymbol{\rho}_{\text{in}}$.
    For $k = 0, \ldots, K-1$:
      Compute F_θ, $S_\theta(\boldsymbol{\rho}^k)$.
      Update $\boldsymbol{\rho}^{k+1} \leftarrow \boldsymbol{\rho}^k + \Delta t(F_\theta \boldsymbol{\rho}^k + S_\theta(\boldsymbol{\rho}^k))$.
    Set $\hat{\boldsymbol{\rho}} \leftarrow \boldsymbol{\rho}^K$.
    Accumulate loss $\ell(\hat{\boldsymbol{\rho}}, \boldsymbol{\rho}_{\text{target}})$.
  Compute gradient $\nabla_\theta \mathcal{L}$ via backpropagation through the unrolled updates.
  Update parameters: $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$.
end while

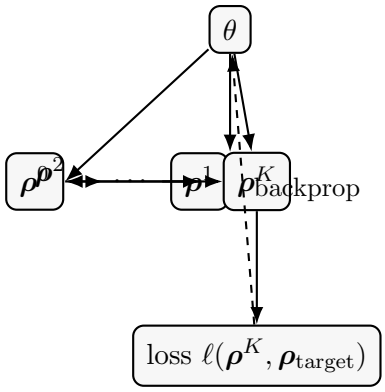


Figure 20: Unrolled FCA lattice over K steps: parameters θ influence all steps. Training proceeds by backpropagation through the unrolled computation.

14 Discussion and Open Directions

Field–Computon Algebra provides a unifying language for computation based on evolving measure fields. Key advantages include:

- a physically interpretable substrate (mass, flow, velocity, source);
- compatibility with PDEs, probabilistic models, and neural networks;
- natural mapping to discrete lattices and distributed hardware;
- a direct connection to optimal transport via Wasserstein gradient flows.

Several directions remain open:

- **Rigorous analysis:** for specific FCA operator classes, establish existence, uniqueness, regularity, and long–time behavior (e.g. convergence to attractors, rates of mixing, metastability).
- **Learning in FCA:** develop training algorithms that learn potentials, interaction kernels, and source terms directly from data, with FCA as the runtime engine, and compare empirically to standard architectures.
- **Hybrid architectures:** integrate symbolic controllers that reshape FCA domains and operators on the fly, yielding programmable field computers with explicit high–level semantics.
- **Hardware realization:** map FCA lattices to neuromorphic or analog substrates where mass and flow are represented by currents, charges, or physical waves, exploring latency, energy, and robustness advantages.
- **Alignment and invariants:** use conserved quantities (mass, energy, entropy) and Lyapunov functionals as guardrails for large FCA systems performing high–stakes computation; study controllability of invariants.
- **Probabilistic FCA:** develop a full measure–theoretic theory of FCA with stochastic perturbations, connecting to Bayesian inference on fields and stochastic partial differential equations.

Conclusion

We have presented an expanded formalization of Field–Computon Algebra: a framework in which computational state is a time–varying measure over a domain, evolving under continuity equations with source terms. Canonical FCA operators—potential, coherence, resonance, and nonlocal interactions—can be composed to build complex dynamics, while discrete lattice implementations turn these ideas into practical algorithms. By embedding Markov chains, consensus processes, graph message passing, and neural layers into the FCA formalism, we have indicated that FCA is at least as expressive as standard models, while offering additional geometric and physical structure. The Wasserstein gradient–flow viewpoint connects FCA to optimal transport, providing tools for analysis and optimization. The operator calculus, stability results, and micro–examples illustrate how FCA can be treated as a genuine dynamical–systems object, with Lyapunov functionals and semigroup structure. Finally, the computon lattice engine and training framework demonstrate how FCA can be realized and tuned in software and hardware with purely local rules. The central vision is that FCA can serve as a common substrate for future computational architectures, where symbolic, neural, and field–based computation coexist and interact within a single, measure–theoretic and geometrically grounded framework.