Faithful local dynamics on every finite acyclic graph

Peter Graham
https://orcid.org/0009-0007-8493-1505

November 3, 2026

Zenodo archive (Lean development in `lean`, `checks`): https://doi.org/10.5281/zenodo.23914494

hexagon:2611.00861v1

Abstract

Give each vertex of a finite directed acyclic graph a finite set of states, and let updating a vertex overwrite its state with a function of the states at its outgoing neighbors. Collina and D'Andrea showed that such updates always satisfy the relations of the graph's Hecke–Kiselman monoid, introduced by Ganyushkin and Mazorchuk, and conjectured that some choice of states and functions satisfies no other relations. They proved this for complete acyclic graphs, and Forsberg's effective representations settle further special classes. We prove the conjecture for every finite directed acyclic graph: some such system represents the monoid faithfully, so two words act equally in every system exactly when they are equal in the monoid. Each update is recorded as a formal expression built from the expressions it reads. Aragona and D'Andrea's cancellation rules reduce every word to one of their normal forms. In a normal form, the expressions produced at each vertex grow strictly and all survive in the final state, so the final state determines the word up to swaps of adjacent commuting letters. Lean checks the full theorem.

This work has not yet been reviewed by experts. The proofs, the Lean formalization and the text were produced by AI models (from OpenAI and Anthropic), working under the author's direction.

MSC 2020: 20M05, 37B15. **Keywords:** Hecke–Kiselman monoid, Kiselman's semigroup, local dynamics, acyclic graph.

1 Introduction

Kiselman's semigroups [4, 5] have a generalization for every directed graph, the Hecke–Kiselman monoids of Ganyushkin and Mazorchuk [6]. Collina and D'Andrea [1] found these monoids in sequential dynamics on a directed acyclic graph: each vertex holds a state, and updating a vertex overwrites its state using only the states at its outgoing neighbors. The updates of every such system satisfy the monoid's relations, and Collina and D'Andrea conjectured that some finite system satisfies no others. We prove this for every finite directed acyclic graph.

Let $G = (V, E)$ be a finite directed acyclic graph, allowing $V = \varnothing$. Write HK_G for the monoid generated by V with relations

$$i^2 = i, \qquad ij = ji \quad (i \neq j \text{ nonadjacent}), \qquad iji = jij = ij \quad (i \to j), \tag{1}$$

where i and j are nonadjacent when there is no arrow between them in either direction. A local system D on G assigns each vertex i a set S_i, always finite and nonempty, and a function $f_i : \prod_{i \to j} S_j \to S_i$. Its update U_i changes coordinate i to $f_i((s_j)_{i \to j})$ and leaves the others fixed. In particular, f_i does not read s_i; for a graph with one arrow $i \to j$, updating the sink j assigns a fixed value and updating i applies a function to the state at j, so updating a vertex twice in a row has the same effect as updating it once. Evaluation $\mathrm{ev}_D : V^* \to \mathrm{End}(\prod_i S_i)$ uses ordinary function composition: the rightmost letter acts first. Here V^* includes the empty word, and $[w]$ denotes its image in HK_G.

Theorem 1. *For every finite directed acyclic graph G, there is a local system D and an injective monoid homomorphism $\rho : \mathrm{HK}_G \to \mathrm{End}(\prod_i S_i)$ with $\rho([i]) = U_i$ for every vertex. Moreover, for all words $a, b \in V^*$,*

$$\mathrm{ev}_A(a) = \mathrm{ev}_A(b) \text{ for every local system } A \text{ on } G \quad \Longleftrightarrow \quad [a] = [b].$$

Collina and D'Andrea call the smallest quotient of HK_G through which every evaluation factors the universal dynamics monoid $D(G)$ [1, Section 8]; the second clause of Theorem 1 says $D(G) \cong \mathrm{HK}_G$. This is their Conjecture 8.1 in the journal version of [1] (Section 8, Conjecture 1 of arXiv:1311.3460v2), first posed in the 2013 preprint (arXiv:1311.3460v1, Section 9, Conjecture 1). Their Proposition 2.4 shows that the updates of every local system satisfy (1), and their Theorem 6.3 implies faithfulness for complete acyclic graphs. Their Conjecture 8.2, that a particular pointed update system on the complete graph is an initial object among pointed update systems, is a different question and is not addressed here. Collina and D'Andrea report a computer check of Conjecture 8.1 on all graphs with at most four vertices and on most graphs with five [1, Section 8]; the results of [1] are part of Collina's thesis [16], according to its acknowledgments. In 2022 D'Andrea reported that a redefined linking operation "works (experimentally)" on all graphs with at most eight vertices [3, slide 21].

The proof uses the elementary cancellations of Aragona and D'Andrea [2, Lemma 2.1]. The words they cannot shorten, their normal forms, are described in [2, Remark 2.5], and [2, Theorem 3.1] shows that two normal forms of the same element of HK_G differ by commutations of nonadjacent letters (see also [2, Corollary 3.2]). The argument below shows directly that two normal forms with the same final state in one particular local system differ in this way, and does not use [2, Theorem 3.1]. It also follows the reconstruction viewpoint of D'Andrea's lecture [3, slides 16, 19].

Earlier faithful models cover special classes of graphs. Kiselman's semigroup is the Hecke–Kiselman monoid of a complete acyclic graph [6, Example 12], and Kudryavtseva and Mazorchuk proved that it has a faithful representation by matrices with nonnegative integer entries [5, Theorem 34]; in their construction each generator differs from the identity matrix in a single column [5, Section 11.1]. Forsberg proved that representations of Hecke–Kiselman monoids "constructed by Ganyushkin and Mazorchuk" are effective, that is, faithful [9, abstract]: for every orientation of a path, the graphs of type A_n [9, Corollary 4.6]; for weighted representations of the graphs Z_n with arrows $a \to v_i \to b$ [9, Theorem 5.1]; and for graphs glued from such pieces at a common source or sink [9, Theorem 4.5]. Passing to the opposite graph [6, Proposition 13], transposing and reducing modulo a suitable prime turns these representations into faithful local systems, so Conjecture 8.1 was already settled for these graphs. The Hecke–Kiselman monoids of type A_n also "naturally appear as monoids of projection functors" [9, Section 1]: Grensing gives defining relations for monoids of projection functors on admissible tree quivers [10, Theorem 3.9], and Grensing and Mazorchuk develop these realizations further [11], showing for admissible tree quivers that "the monoid generated by dual projection functors is the Hecke–Kiselman monoid of the underlying quiver" [12, abstract]. None of these results covers arbitrary acyclic graphs.

Recent work counts the elements of the complete-graph monoids [13, Theorems 2.1 and 3.2] and gives faithful representations of the monoids of directed cycles [14, abstract], which fall outside our hypothesis, and of Catalan monoids [15, abstract]; none of these is used here. In sequential dynamical systems [17, 18, 19, 20, 22, 23], [21, Chapter 7], local maps act on undirected graphs and may read the updated vertex's own state [23, Definition 2.1], so the relations (1) need not hold; these works study the update orders of given systems, and, to our knowledge, none determines the relations satisfied by all local systems. Collina and D'Andrea cite four of them [17, 18, 19, 21] and pose Conjecture 8.1 after them. Ulutaş and Köroğlu [24] study the computational cost of their earlier correspondence [25] between period-3 orbits of sequential dynamical systems on complete graphs and ternary codes, which extends Defant's correspondence for period-2 orbits and binary codes [26].

2 Proof overview

At each vertex we store a formal expression: an initial symbol, or a constructor applied to the current expressions at its outgoing neighbors (Section 3). Acyclicity makes the set of expressions at each vertex finite. Every word equals a normal form, a word in which any two occurrences of a letter i have both an incoming and an outgoing neighbor of i between them (Lemma 2). In a normal form the outgoing neighbor makes successive expressions at i strictly larger, and the incoming neighbor copies each overwritten expression at i into a later expression, where it survives (Lemma 3). The final state therefore determines the expressions produced at each vertex, their order, and, on each arrow, how the updates at its two ends interleave (Lemma 4). The projection lemma for partially commutative words, due to Cori and Perrin and to Duboc [7, 8], then recovers the word up to swaps of adjacent commuting letters (Lemma 5); we prove it below.

3 Constructor terms and recovery

Because the rightmost letter acts first, a word w is executed in the order of its reverse w^{rev}; we call w^{rev} the chronological schedule of w. An *event* is an occurrence of a letter in a chronological schedule.

For each vertex i, let T_i be the set of formal expressions consisting of a symbol X_i and one expression $N_i((t_j)_{i\to j})$ for each tuple of expressions $t_j \in T_j$ at the outgoing neighbors j of i:

$$T_i = \{X_i\} \;\sqcup\; \{N_i((t_j)_{i\to j}) : t_j \in T_j\}.$$

Distinct tuples give distinct expressions, and none of them is X_i. Since G is acyclic, these sets are defined recursively from the sinks upward. At a sink, T_i has exactly two elements, X_i and N_i of the empty tuple; at other vertices T_i is finite because the finitely many child sets are. The *constructor system* is the local system with $S_i = T_i$ and $f_i((t_j)) = N_i((t_j))$: updating i replaces its expression by N_i of the current expressions at its outgoing neighbors. Start at $x = (X_i)_{i\in V}$. The term of an event is the expression it produces. The subterms of an expression are the expression, its entries t_j, their entries, and so on; a subterm $N_j(\dots)$ or X_j belongs to T_j, and we say it is at vertex j. Set $|X_i| = 0$ and $|N_i((t_j))| = 1 + \sum_{i\to j} |t_j|$.

Following Aragona and D'Andrea [2], call a word a *normal form* if between every two occurrences of any letter i it contains an incoming neighbor and an outgoing neighbor of i; by [2, Remark 2.5] these are exactly the words to which none of their elementary cancellations applies. It suffices to check consecutive occurrences, and the reverse of a normal form is a normal form.

Lemma 2. *Every word is equal in* HK_G *to a normal form.*

Proof. Let m be a word and i a vertex. The elementary cancellations [2, Lemma 2.1], whose convention $aba = ab$ for $a \to b$ agrees with (1), state:

$$imi = im \qquad \text{if } m \text{ contains no incoming neighbor of } i,$$
$$imi = mi \qquad \text{if } m \text{ contains no outgoing neighbor of } i.$$

Both follow from (1) by induction on the length of m. For the second, write $m = m_1k$. If $k = i$ or k is nonadjacent to i, use $i^2 = i$ or $ki = ik$ and the induction hypothesis $im_1i = m_1i$. If $k \to i$, then $ki = iki$, so $im_1ki = (im_1i)ki = m_1iki = m_1ki$. The first identity is symmetric. These identities are also proved in the formalization. A shortest word equal to a given word therefore admits neither shortening, so it is a normal form. □

Lemma 3. *Let a chronological schedule be a normal form, executed in the constructor system from x. The sizes of successive event terms at a fixed vertex strictly increase. The noninitial subterms of the final coordinates are exactly the event terms, and those at vertex i are the event terms at i.*

Proof. At every stage, each stored entry t_j of a current expression was a previous value of coordinate j. By induction on time: every update weakly increases the size of its coordinate, and in a normal form strictly increases it. Indeed, an update at i reads the current expressions at the outgoing neighbors, while the expression it replaces is X_i or stores earlier values of the same coordinates, which by induction are no larger. A first update has positive size. Between consecutive events at i, the normal form gives an outgoing neighbor j that fires; by induction that event strictly increases the j coordinate, so the next sum defining the size at i is strictly larger. In a normal form a sink occurs at most once.

Every noninitial subterm was produced at an event: this is an induction on the execution, since an update merely wraps already present terms. Conversely consider an event at i. If its value is never overwritten, it survives in the final i coordinate. Otherwise choose the next event at i. Between them an incoming neighbor $k \to i$ fires, storing the value as an entry. If that later k term is overwritten, repeat the argument. The chosen event moves strictly forward in the finite schedule, so eventually a final coordinate contains the original term. □

Lemma 4. *If two chronological schedules that are normal forms reach the same final state in the constructor system from x, then they have the same letter counts and the same two-letter projection on every edge.*

Proof. By Lemma 3, the final state determines the set of event terms at each vertex. Their strictly increasing sizes order that set, giving the entire event history at that vertex, including its length. Fix an edge $i \to j$. Each i event records either X_j or one term in the ordered j history. These values are all distinct, so the recorded entry identifies the number of j events preceding that i event. These numbers, together with the total j count, determine the $\{i, j\}$ projection: place the required additional j letters before each successive i, then place the remaining j letters at the end. □

Lemma 5 (projection lemma [7, 8])**.** *Two words with equal letter counts and equal two-letter projections for each adjacent pair of distinct vertices differ by swaps of consecutive nonadjacent letters.*

Proof. Induct on the common length. If the first word starts with i, locate the first i in the second word; it exists by the count equality. Every letter before it is distinct from and nonadjacent to i: adjacency would contradict equality of the corresponding two-letter projections. Commute this i to the front and remove the initial i from both words. The remaining counts and projections still agree, so induction applies. The zero-length case is immediate. □

Proof of Theorem 1. Every local system satisfies (1). Idempotence follows because an update does not read its own coordinate. Nonadjacent updates commute. For $i \to j$, the update at j does not read i, and the outer updates overwrite earlier values; direct substitution gives $U_iU_jU_i = U_jU_iU_j = U_iU_j$. Thus evaluation induces a homomorphism from HK_G in every local system.

Use the constructor system. Suppose two words a, b act equally. Choose normal forms a', b' equal to them in HK_G by Lemma 2; their actions still agree. Evaluate at x. The chronological schedules $(a')^{\mathrm{rev}}$ and $(b')^{\mathrm{rev}}$ are normal forms and reach the same final state. Lemmas 4 and 5 relate them by swaps of adjacent nonadjacent letters. Reversing these swaps relates a' and b' by the same commutations. Hence $[a] = [b]$, proving injectivity.

Equality in HK_G implies equal evaluations in every system by the factorization already proved. Conversely, equality in every system implies equality in the constructor system, hence equality in HK_G. For the empty graph the empty product is a singleton and the same argument applies. □

4 Formal verification

Lean checks the theorem `CD13.Faithful.full_endpoint`, whose statement `CD13.FullEndpoint` is defined in the file `CD13/Endpoint.lean` of the accompanying Lean project:

```
def FullEndpoint : Prop :=
  ∀ (V : Type u) [Finite V] (E : V → V → Prop), Acyclic E →
    ∃ (D : LocalSystem.{u, u} E)
      (ρ : HK E →* Function.End (State D)),
      Function.Injective ρ ∧
      (∀ i, ρ (PresentedMonoid.of (HKRel E) i) = update D i) ∧
      (∀ a b : FreeMonoid V,
        (∀ A : LocalSystem.{u, u} E, eval A a = eval A b) ↔
          PresentedMonoid.mk (HKRel E) a = PresentedMonoid.mk (HKRel E) b)
```

Here V is any finite type, possibly empty, and E is the arrow relation. The hypothesis `Acyclic E` says that no vertex is related to itself by the transitive closure of E, that is, E has no directed cycle. `HK E` is the monoid presented by the four relations of (1); a `LocalSystem` gives each vertex a finite nonempty coordinate type and a function reading exactly the coordinates at its outgoing neighbors; `update` and `eval` are U_i and ev_A, with ordinary composition, rightmost letter first. The coordinate types lie in the same universe as V; since every finite set has a copy there, no generality is lost. There is no hypothesis on the number of vertices, degrees, transitivity or word length.

The constructor finiteness, cancellation, event-size and subterm arguments, projection recovery, trace reconstruction and final action are all proved in Lean. The command `#print axioms CD13.Faithful.full_endpoint` reports only Lean's three standard axioms: `propext` (propositional extensionality), `Classical.choice` (choice) and `Quot.sound` (quotient soundness); no unproved mathematical assumption is used. The build also checks the statement's type and elementary boundary cases. Reproduction uses the accompanying Lean project's pinned toolchain and Mathlib dependencies; its README gives the two standard build commands and the resource requirements.

References

[1] E. Collina and A. D'Andrea, *A graph-dynamical interpretation of Kiselman's semigroups*, J. Algebraic Combin. 41 (2015), 1115–1132. doi:10.1007/s10801-014-0569-7. Preprint arXiv:1311.3460v1, 2013, titled *Sequential dynamical systems on finite acyclic oriented graphs and Hecke-Kiselman monoids*; revised as arXiv:1311.3460v2, May 20, 2014, whose numbering is that of the journal version, with Conjectures 8.1 and 8.2 numbered Conjectures 1 and 2 in Section 8.

[2] R. Aragona and A. D'Andrea, *Normal form in Hecke-Kiselman monoids associated with simple oriented graphs*, Algebra Discrete Math. 30 (2020), 161–171, doi:10.12958/adm1571. Numbering cited from arXiv:1904.10745v1, 2019; in the journal version Lemma 2.1, Remark 2.5 and Theorem 3.1 are Lemma 1, Remark 3 and Theorem 1.

[3] A. D'Andrea, *An invitation to Hecke-Kiselman monoids*, ANTIPODE 2 lecture, September 12, 2022, slides 16, 19 and 21. Lecture slides.

[4] C. O. Kiselman, *A semigroup of operators in convexity theory*, Trans. Amer. Math. Soc. 354 (2002), 2035–2053. doi:10.1090/S0002-9947-02-02915-X.

[5] G. Kudryavtseva and V. Mazorchuk, *On Kiselman's semigroup*, Yokohama Math. J. 55 (2009), 21–46. arXiv:math/0511374v1.

[6] O. Ganyushkin and V. Mazorchuk, *On Kiselman quotients of 0-Hecke monoids*, Int. Electron. J. Algebra 10 (2011), 174–191. arXiv:1006.0316v3.

[7] R. Cori and D. Perrin, *Automates et commutations partielles*, RAIRO Inform. Théor. 19 (1985), 21–32. doi:10.1051/ita/1985190100211.

[8] C. Duboc, *On some equations in free partially commutative monoids*, Theoret. Comput. Sci. 46 (1986), 159–174. doi:10.1016/0304-3975(86)90028-9.

[9] L. Forsberg, *Effective representations of Hecke-Kiselman monoids of type A_n*, arXiv:1205.0676v4, February 15, 2017.

[10] A.-L. Grensing, *Monoid algebras of projection functors*, J. Algebra 369 (2012), 16–41. doi:10.1016/j.jalgebra.2012.07.002, arXiv:1203.1943v2.

[11] A.-L. Grensing and V. Mazorchuk, *Categorification of the Catalan monoid*, Semigroup Forum 89 (2014), 155–168. doi:10.1007/s00233-013-9510-y.

[12] A.-L. Grensing and V. Mazorchuk, *Finitary 2-categories associated with dual projection functors*, Commun. Contemp. Math. 19 (2017), 1650016. doi:10.1142/S0219199716500164. arXiv:1501.00095v2.

[13] A. D'Andrea and S. Stella, *Cardinalities of Kiselman monoids and their parity asymptotics*, arXiv:2610.14331v1, posted October 9, 2026.

[14] Z. F. Jin, Y. F. Luo and W. T. Zhang, *Combinatorics and faithful representations of cycle monoids*, Semigroup Forum, October 20, 2026. doi:10.1007/s00233-026-10687-7.

[15] Y. F. Luo, Z. F. Jin and W. T. Zhang, *Finite Gröbner–Shirshov bases and faithful representations for Catalan monoids*, Comm. Algebra 54 (2026), 185–195. doi:10.1080/00927872.2025.2522155.

[16] E. Collina, *Dynamical systems on graphs and Hecke-Kiselman monoids*, PhD thesis, Sapienza Università di Roma, 2014.

[17] C. L. Barrett and C. M. Reidys, *Elements of a theory of computer simulation I: sequential CA over random graphs*, Appl. Math. Comput. 98 (1999), 241–259. doi:10.1016/S0096-3003(97)10166-7.

[18] C. L. Barrett, H. S. Mortveit and C. M. Reidys, *Elements of a theory of simulation II: sequential dynamical systems*, Appl. Math. Comput. 107 (2000), 121–136. doi:10.1016/S0096-3003(98)10114-5.

[19] C. L. Barrett, H. S. Mortveit and C. M. Reidys, *Elements of a theory of simulation III: equivalence of SDS*, Appl. Math. Comput. 122 (2001), 325–340. doi:10.1016/S0096-3003(00)00042-4.

[20] H. S. Mortveit and C. M. Reidys, *Discrete, sequential dynamical systems*, Discrete Math. 226 (2001), 281–295. doi:10.1016/S0012-365X(00)00115-1.

[21] H. S. Mortveit and C. M. Reidys, *An Introduction to Sequential Dynamical Systems*, Universitext, Springer, New York, 2008. doi:10.1007/978-0-387-49879-9.

[22] C. M. Reidys, *Sequential dynamical systems over words*, Ann. Comb. 10 (2006), 481–498. doi:10.1007/s00026-006-0301-y.

[23] L. D. Garcia, A. S. Jarrah and R. Laubenbacher, *Sequential dynamical systems over words*, Appl. Math. Comput. 174 (2006), 500–510. doi:10.1016/j.amc.2005.04.101.

[24] T. Ulutaş and M. E. Köroğlu, *Counting complexity of period-3 orbits in sequential dynamical systems and their relationship to error-correcting codes over ternary fields*, Symmetry 18 (2026), 1658. doi:10.3390/sym18101658.

[25] T. Ulutaş and M. E. Köroğlu, *Period-3 orbits of sequential dynamical systems and their relationship to error-correcting codes over finite fields*, Graphs Combin. 42 (2026), 14. doi:10.1007/s00373-025-03008-2.

[26] C. Defant, *Binary codes and period-2 orbits of sequential dynamical systems*, Discrete Math. Theor. Comput. Sci. 19(3) (2017), 10. doi:10.23638/DMTCS-19-3-10.