Dynamical systems on graphs and Hecke-Kiselman monoids

Elena Collina

May 21, 2014

Contents

Introduction

Sequential Dynamical Systems (or SDS) are a discrete structure that has been developed during the nineties in order to provide a mathematical basis for the transportation system developed in the TRANSIMS project in Los Alamos Laboratories ([BR99], [BMR00], [BMR01] and [BMR03]). In an SDS, the state on each vertex of an undirected graph is updated according to fixed rules involving neighboring states. The system state evolution is prescribed by a sequence, often a permutation, of the vertices which induces the order in which to perform the updates. Generally, the evolution varies according to the chosen word, but it has been proved that, under specific assumptions, aspects of the evolution are independent on the chosen permutation ([MM09]). Moreover, when the vertex states are boolean, dynamical aspects of the periodic states can be described by a quotient of a suitable Coxeter group (see [MM10]).

Object of this work is the study of the dynamics of update systems, which are SDS-like structure defined on a directed acyclic graph: here each vertex state is updated based on the states on vertices reached by an arrow pointing out from it (the complete definition is in Chapter 2). As idempotence relations and braid relations are naturally realized in this setting, possible evolutions of the system arise as elements of a monoid which is a quotient of a Hecke-Kiselman monoid, i.e., a monoid admitting a presentation in which the above relations occur. In [KM09], a description of elements in the Kiselman semigroup as words, in a given set of generators, satisfying suitable conditions is given. This description has been used in [CD13], together with a *join* operation defined on states, in order to construct an update system whose dynamics is a faithful image of the Kiselman semigroup.

This work is organized as follows:

- Chapter 1 provides a precise description of SDS, then a short survey of classic results. Particular attention is devoted to dynamical issues of SDS, which are at last expressed in terms of a word problem, i.e., which are the different words that provide the same phase space for an SDS. Coxeter relations occurring in the case of cycle equivalence of SDS were the motivation for our subsequent research work.

- Chapter 2 introduces the definition of update systems. It states the fundamental result that their dynamics monoid is a quotient of an Hecke-Kiselman monoid, then it provides the proof of the isomorphism between the two in the case of the complete graph, via the construction of a particular update system. Finally, some dynamic properties of this update system are proved.

- Chapter 3 tackles the problem on a generic graph, analyzing the issues that did not occur when the base graph was complete. It presents our conjecture for a canonical element of the Hecke-Kiselman monoid and a possible candidate operation for the definition of a function on states, together with a worked example on a 5 vertices graph. It is shown that these two tools are sufficient to prove the conjecture on empty, star and line graph.

1 Preliminaries

A urban traffic simulation system, a social network, an epidemiology contact scheme for people living in the same region, the biological interaction among genes in a DNA sequence: all these are examples of real world structures whose dynamics is of interest. However, rather than on a continuous domain, they should be studied on a discrete one: a (finite) graph.

In applied mathematics models are often continuous, as they work very well whenever a large scale representation describes faithfully enough the object of analysis, like in diffusion processes or fluid flows problems: the continuous model is discretized only when it is implemented in a computer simulation. This provides a powerful tool in many contests, but one inevitably loses the analysis of each agent dynamic.

A discrete model seems natural when the issue involves the study of how different agents interact with each other: for example, trying to model how long would it take for a trip in the traffic, how to implement an effective advertisement campaign in a social network, which is the probability for an individual to become ill in case of the spread of a disease.

The traffic simulation scheme TRANSIMS has been developed during the nineties in Los Alamos Laboratories (see, [NSP$^+$97] for reference[1]): it aimed to provide a rather detailed description of the interaction of people, public transportation and cars in a big transportation network (it has been used to model both Chicago and Dallas areas). Along with the simulation tool, a mathematical structure has been introduced: is that of Sequential Dynamical System (SDS), first introduced by Barret, Mortveit and Reyds in [BR99], [BMR00], [BMR01] and [BMR03].

It is easy to see that the continuous model would not work in this setting. Moreover, unlike in cellular automata, sequential dynamical systems allow the base graph representing the agents to be generic (rather than a regular lattice); finally, the state update of each single element occurs asynchronously, i.e. it considers the updates one after another, thus permitting to obtain the new state in position v at time $t+1$ according to the states of its neighbor vertices $n[v]$ at time t.

This chapter contains no original work of the author: closely following [MR08], we will go through the classical definition of Sequential Dynamical System on a undirected graph and we will outline the main results involving its dynamics. In the section 1.1, we will review the TRANSIMS model, as first and motivating application. In section 1.2, we will introduce the concept of Sequential Dynamical System; sections 1.3 will express the dynamics of an SDS via its phase space, while in section 1.4 we will start look at the possible permutations inducing an SDS. Section 1.5 will go through several classic

[1]the source code is freely available on SourceForge under an established open-source license in the website [Tra14]

results about stability, introducing three types of equivalence among SDS (and their phase spaces): functional equivalence, dynamical equivalence and cycle equivalence. In last sections we follow [MM10] in order to describe the existing analogies between Coxeter groups and SDS. In section 1.6 we state the word problem for an SDS, in analogy with the word problem for a Coxeter group. In section 1.7, we describe w-independent SDS, whose dynamics do not depend on the word inducing the update order. In the sections 1.8 and 1.9 it is shown the connection between Coxeter groups and cycle equivalence, in order to state that, under suitable hypothesis, the periodic points of the phase space of a SDS form a group which is a quotient of a Coxeter group: this has been the starting point for our work, that is described in the next two chapters.

1.1 TRANSIMS

Continuous traffic models can predict jams as shocks in hyperbolic PDEs. However, when modeling a urban traffic network, one is often interested in problems such as choosing the best travel route between two points on the map. The motivation for SDS was to introduce a mathematical tool useful to describe a discrete sequential multi agent interaction, which is the case of traffic model TRANSIMS.

The given data were a given traffic road map, a population, and a schedule with times and location for this person for the duration of the simulation.

Actual simulations were carried out on complicated urban networks (Chicago and Dallas areas, for example), modeling together walkways, public transportation and car traffic. However, for our purposes it is enough to keep in mind a non intersecting road, where cars either move forward, change line or stop.

The simulation was carried out via a TRANSIMS router and a micro-simulator. The router translated each schedule into a travel route. Then micro-simulator would execute all these travel routes, so that each individual activity plan was respected. This simulation typically led to too many agents on main roads, hence travel times were much higher than those expected basing on surveys. Hence, the router would run again, redirecting otherwise part of the individuals whose travel times were too high, and the micro simulator would execute the updated travel routes. This process was repeated until the obtained travel times were acceptable.

The reason behind the need of sequential updates is effectively described when modeling the behavior of cars waiting for traffic lights to turn from red to green. Everyday experience tells us that the front row (with respect to the traffic lights) is expected to move a little earlier than the second, and so on. However,

- a parallel update would show all cars starting to move at the same time;
- a sequential update from front to back would give the same result;
- a sequential update from back to front would represent the expected behavior

This example tells us that parallel and sequential updates are closely connected (as in some cases they can even perform the same results), while the strength of both can be a useful tool in order to correctly represent different situations.

Finally, it is important to stress that sequential updates can be modeled via a synchronous structure. The TRANSIMS micro-simulator was cellular automaton-based: it performed sequentially these four tasks (for a detailed description, we refer to [MR08]):

1. lane change decision,
2. lane change execution,
3. velocity update (acceleration/deceleration),
4. position update (movement).

In each step, the TRANSIMS micro-simulator updates all cells synchronously: however, the constrains for each cell update provided the expected back-to-front sequential update order dynamics, so that overall the implemented model did not work in parallel, but sequentially.

1.2 Preliminary notions and results about SDS

In the following sections, we will introduce and describe properties of SDS according to their classical definition, as given in [MR08]. In particular, we will assume that the base graph is undirect[2].

Let $\Gamma = (V, E)$ be an undirected finite graph, with vertex set $V = \{0, 1, \dots, n-1\}$ of cardinality n and edge set

$$E = \{(i, j) \text{ such that } i, j \in V\}.$$

We will take Γ to be combinatorial, i.e. without loops and multiple edges. The *vertex neighbourhood* of a given vertex $i \in V$ is the subset

$$x[i] = \{i\} \cup \{j \colon (i, j) \in E\}.$$

Fix a graph $\Gamma = (V, E)$, and for each vertex $i \in V$ a finite set of states S_i. The restriction of $\mathbf{s}$ to the vertices in $x[i]$ is referred to as *state neighbourhood* of i, and is written as

$$s[i] = \{s_j \colon j \in x[i]\}.$$

We denote by $S = \prod_{i \in V} S_i$ the family of all the possible *system states*, i.e., n-tuples $\mathbf{s} = (s_1, s_2, \dots, s_n)$, where s_i belongs to S_i for each vertex i.

Moreover, on each vertex $v \in V$ we define a vertex function

$$f_i \colon \prod_{j \in x[i]} S_j \to S_i$$

[2]Our result, described in next chapters, is based on a structure whose base graph is a directed acyclic graph; like in an SDS each vertex is equipped with a state set and an update function, but there is not a word prescribing the order in which to perform the updates. We will call it *update system*.

This function naturally extends to the Γ-*local functions* $\mathbf{F}_i \colon S \to S$, which act as the identity on all vertex states but the i-th, and compute on vertex i the result of $f_i(s[i])$.

Let us consider the alphabet $A = \{a_1, \dots, a_n\}$, where each letter a_i correspond to a vertex $i \in V$. Consider a word $w = a_{j_1}a_{j_2}\dots a_{j_k}$ in the free monoid on $\mathrm{F}(A)$. This induces a Γ-local function

$$\mathbf{F}_w = \mathbf{F}_{a_{j_1}}\mathbf{F}_{a_{j_2}}\dots\mathbf{F}_{a_{j_k}} : S \to S$$

which acts on the system state $\mathbf{s}$ sequentially applying the update functions $\mathbf{F}_{a_{j_i}}$ in the order fixed by the word w.

Definition 1.2.1. A *Sequential Dynamical System* (or SDS) is the triple $(\Gamma, (f_i)_i, w)$.

When w is a permutation on V, i.e., each vertex appears exactly once, we will call it a *permutation*-SDS.

Remark 1.2.2. Fixed Y, S and $(f_i)_i$, a cellular automaton (or CA) is a uniform SDS where

- the base graph Γ is a regular finite or infinite lattice (e.g. $\mathbb{Z}^k$ or Circ_k, for $k \geq 1$) ;
- $f_i = f_j$ and $S_i = S_j$ for any vertices $i, j \in V$;
- all the update functions are performed in parallel (i.e. at the same time).

In the setting of cellular automata, often $S_i = \mathbb{F}^2 = \{0, 1\}$.

The following example was worked out in [MR08]. We choose to refer to it along this introductory chapter on SDS because it holds several interesting properties and it is possible to work it out by hand.

Example 1.2.3. Let us consider the graph Circ_4, with vertex set is $V = \{0, 1, 2, 3\}$ and edge set $E = \{(0, 1), (1, 2), (2, 3), (3, 0)\}$.

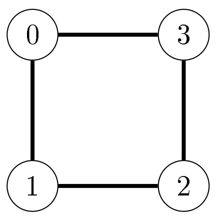


Figure 1.1: Circ_4.

Here, for instance, the vertex neighbourhood of 0 is $\{3, 0, 1\}$.

Let us associate to each vertex a state set $\mathbb{F}_2$ and a vertex function

$$\mathrm{nor} \colon \mathbb{F}_2^3 \to \mathbb{F}_2, \quad \mathrm{nor}(\mathbf{s}) = (1 + s_1)(1 + s_2)(1 + s_3).$$

Hence, the corresponding Γ-local maps are

$$\begin{aligned} F_0(x) &= (\mathrm{nor}(x_3, x_0, x_1), x_1, x_2, x_3), \\ F_1(x) &= (x_0, \mathrm{nor}(x_0, x_1, x_2), x_2, x_3), \\ F_2(x) &= (x_0, x_1, \mathrm{nor}(x_1, x_2, x_3), x_3), \\ F_3(x) &= (x_0, x_1, x_2, \mathrm{nor}(x_2, x_3, x_0)). \end{aligned}$$

Consider the system state $x = (0,0,0,0)$. Updating via the word $w = (0,1,2,3)$ we obtain, for example,

$$\begin{aligned} F_0(0,0,0,0) &= (1,0,0,0), \\ F_1 \circ F_0(0,0,0,0) &= (1,0,0,0), \\ F_2 \circ F_1 \circ F_0(0,0,0,0) &= (1,0,1,0), \\ F_3 \circ F_2 \circ F_1 \circ F_0(0,0,0,0) &= (1,0,1,0), \end{aligned}$$

This is a classical SDS and the standard notation for this type of update functions is

$$[\mathrm{Nor}_{\mathsf{Circ}_4}, (0,1,2,3)](0,0,0,0) = (1,0,1,0).$$

Here comes the point where the *sequential* character of SDS appears: in fact, if we had applied the update function to $(0,0,0,0)$ in parallel (like we would have done in a CA), the result would have been $(1,1,1,1)$.

1.3 Phase spaces

The study of dynamical systems has among its major goals that of identify the dynamics of the system. That is, to decompose the system state family S in

$$S = \mathrm{Per}(S) \cup \mathrm{Transient}(S)$$

where

$$\begin{aligned} \mathrm{Per}(S) &= \{x \in S \colon \exists k \in \mathbb{N}\ ,\ F_w^k(x) = x\}, \\ \mathrm{Transient}(S) &= \{x \in S \colon \nexists k \in \mathbb{N}\ , F_w^k(x) = x\}. \end{aligned}$$

The first set contains the *periodic states* (i.e. states that can be either *fixed* or *cyclic* under the action of $\mathbb{F}_\Gamma$). The states in the second set are called *transient states*. Moreover, for a system state x, it is useful to define its *forward orbit* of x under the $\mathsf{SDS}[F_\Gamma, w]$ as

$$O^+(x) = (x, [F_\Gamma, w](x), [F_\Gamma, w]^2(x), \dots).$$

Definition 1.3.1. The *phase space* of an SDS map $[F_\Gamma, w]$ is the directed graph $\Gamma = \Gamma([F_\Gamma, w])$ whose vertex set is $V = S$ and whose oriented edges are

$$E = \{(x, y) \colon x, y \in S, y = [F_\Gamma, w](x)\}.$$

We will often denote the phase space $\Gamma([F_\Gamma, w])$ simply via its SDS-maps $[F_\Gamma, w]$.

Remark 1.3.2. Even if we fix the base graph Γ and the update functions F_Γ, different words can lead to different SDS.

Example 1.3.3. Let us draw the phase space of some SDS who share the same base graph and update function, and differ only in the chosen word. The base graph is Circ_4, and update functions are those introduced in Example 1.2.3. Just like in the previous case, this example was drawn from [MR08].

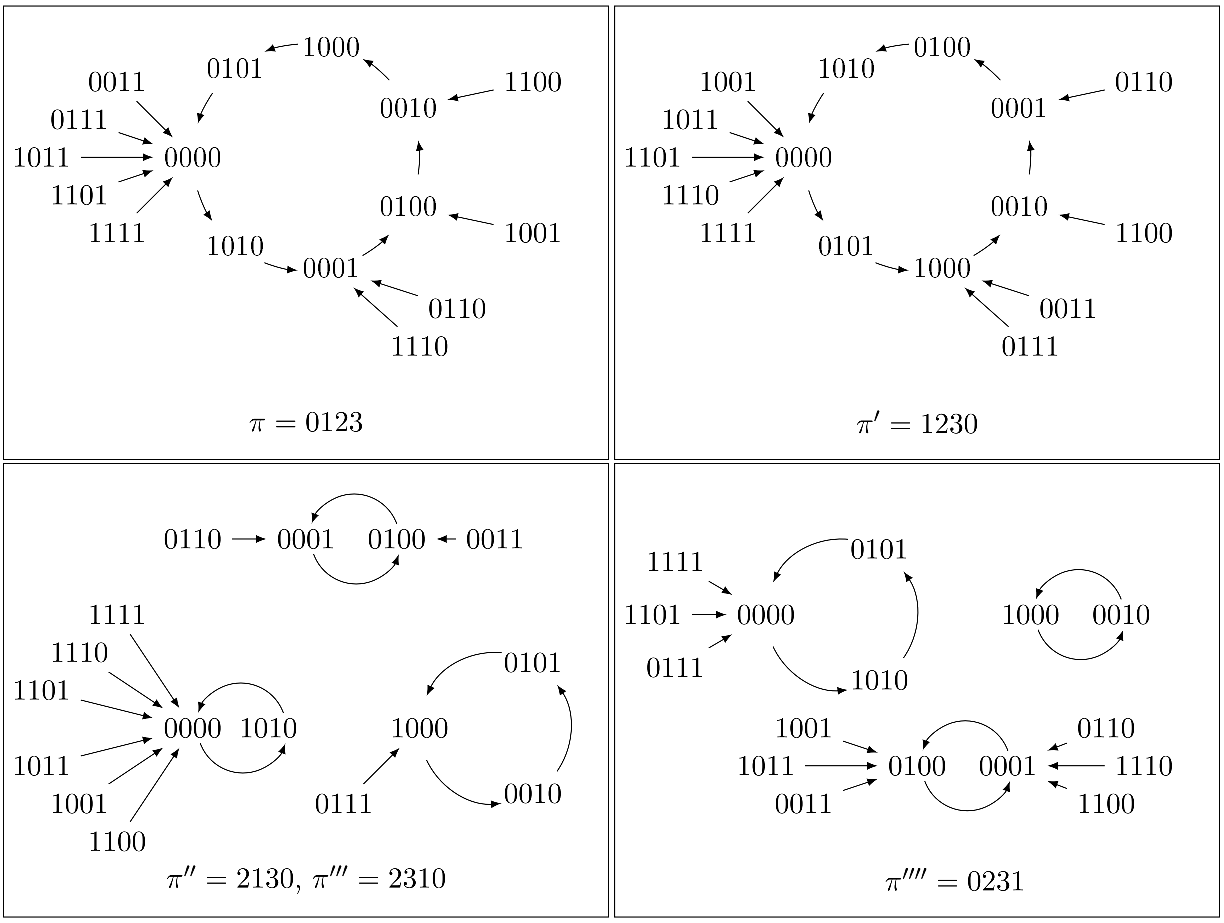


Figure 1.2: Phase spaces.

It is a trivial observation that the phase space of an SDS consists of disjoint cycles, with directed trees (transient states) attached. Periodic states form cycles (eventually of length 1) and transient states form trees.

Moreover, for any SDS, all orbits are finite, as it is the number of possible system states.

Fixing a word (and, hence, a partial order on the vertex set of graph Γ) provide us the dynamics of the SDS, which would otherwise be chaotic, in the generic case.

Each permutation π of the vertices in V induces an orientation $\mathcal{O}_\Gamma(\pi)$ of the graph: for every edge $i, j \in E$, permutation π induces the orientation $i \to j$ if and only if the i appears on the left of j in π, otherwise $j \to i$ is induced.

Therefore, for every combinatorial graph Γ, it is well defined the map

$$\begin{aligned} f'_\Gamma \colon S_\Gamma &\to \mathrm{Acyc}(\Gamma) \\ \pi &\mapsto \mathcal{O}^\pi_\Gamma \end{aligned}$$

which sends each permutation of elements in V to the induced graph orientation.

1.4 Adjacent permutations and the update graph

In this section we introduce some useful definitions and results that provides the simplest example of equivalence among SDS. In particular, an equivalence relation on the symmetric group S_Γ leads to a corresponding one between SDS.

Definition 1.4.1. Let S_Γ be the symmetric group over the vertex set V of the combinatorial graph Γ. An equivalence relation $\sim_\Gamma$ on S_Γ is defined as follows: two permutations $\pi = (u_1, \ldots, u_n)$ and $\pi' = (v_1, \ldots, v_n)$ are *adjacent*, and we will write $\pi \sim_\Gamma \pi'$, if and only if there is an index k such that

1. (u_k, u_{k+1}) is not an edge in Γ,
2. $u_k = v_{k+1}$, $u_{k+1} = v_k$, and
3. $u_i = v_i$ for $i \neq k, k+1$;

This is an equivalence relation on S_Γ, and we will denote the equivalence class of π by

$$[\pi]_\Gamma = \{\pi' \colon \pi' \sim_\Gamma \pi\}$$

and the set of all equivalence classes will be $S_\Gamma / \sim_\Gamma$.

Definition 1.4.2. The *update graph* on Γ is a combinatorial graph whose vertex set is S_Γ and whose edges are π, π' whenever π and π' are adjacent. It will be denoted by $U(\Gamma)$.

Example 1.4.3. The update graph of Circ_4 is the following.

$$\begin{tikzcd}
0123 & 1230 & & 3210 & 2103 \\
2301 & 3012 & & 1032 & 0321 \\
0132 \arrow[r, dash] & 0312 & & 1023 \arrow[r, dash] & 1203 \\
2130 \arrow[r, dash] & 2310 & & 3021 \arrow[r, dash] & 3201 \\
0213 \arrow[r, dash] \arrow[d, dash] & 0231 \arrow[d, dash] & & 1302 \arrow[r, dash] \arrow[d, dash] & 1320 \arrow[d, dash] \\
2013 \arrow[r, dash] & 2031 & & 3102 \arrow[r, dash] & 3120
\end{tikzcd}$$

Figure 1.3: The update graph of Circ_4.

The following result is a consequence of the stated properties of vertex permutations and acyclic orientations.

Proposition 1.4.4 ([MR08])**.** *For any combinatorial graph* Γ *there exists a bijection*

$$f_\Gamma \colon [S_\Gamma / \sim_\Gamma] \quad \to \quad \mathrm{Acyc}(\Gamma)$$

i.e., permutations in the same equivalence class with respect to $\sim_\Gamma$ *induce the same orientation on* Γ*.*

Remark 1.4.5. A consequence of Proposition 1.4.4 is the permutations inducing the same acyclic orientation in Γ form a connected components for the update graph $U(\Gamma)$.

From now on, we will denote the number of acyclic orientations of graph Γ by

$$\alpha(\Gamma) = |\,\mathrm{Acyc}(\Gamma)|$$

It is a Tutte invariant, as $\alpha(\Gamma) = T_\Gamma(2,0)$, hence it satisfies the following recursion formula: fix an edge e in Γ, let Γ'_e be the graph obtained by Γ deleting the edge e, and Γ''_e be the graph obtained via the contraction of e; then, we have

$$\alpha(\Gamma) = \alpha(\Gamma'_e) + \alpha(\Gamma''_e).$$

1.5 Dynamical issues for an SDS

This and the following sections deal with the problem of understanding the role of permutations (and words) in the dynamics of an SDS. On classical (undirect) SDS this has been handled giving different concepts of equivalence, studying the behaviour of different update functions on different graphs. Two main constrain, inherited from cellular automata, are sometimes requested in this setting:

- vertex states were limited to $\mathbb{F} = \{0, 1\}$;
- all vertex function were asked to coincide.

Definition 1.5.1. Let us consider two different sequential dynamical systems with SDS-maps $\phi, \psi \colon S \to S$. Then the SDS are:

1. *functionally equivalent* if their SDS-maps ϕ and ψ coincide;
2. *dynamically equivalent* if $\phi \circ h = h \circ \psi$, for some bijection $h \colon S \to S$;
3. *cycle equivalent* if $\phi|_{\mathrm{Per}(\phi)} \circ h = h \circ \psi|_{\mathrm{Per}(\psi)}$ for some bijection $h : \mathrm{Per}(\phi) \to \mathrm{Per}(\psi)$.

In other words, two SDS are functionally equivalent if they have the same phase space, dynamically equivalent if they coincide as unlabeled graphs, cycle equivalent if their sets of periodic states coincide (regardless the induced dynamics).

Example 1.5.2. Back to Example 1.3.3:

- $[\mathrm{Nor}_{\mathsf{Circ}\ 4}, (2130)]$ and $[\mathrm{Nor}_{\mathsf{Circ}\ 4}, (2310)]$ are functionally equivalent: in fact, their phase spaces coincide.
- $[\mathrm{Nor}_{\mathsf{Circ}\ 4}, (0123)]$ and $[\mathrm{Nor}_{\mathsf{Circ}\ 4}, (1230)]$ provide an example SDS that are dynamically equivalent, as their phase spaces are the same graph with different vertex labels;
- $[\mathrm{Nor}_{\mathsf{Circ}\ 4}, (0231)]$ is cycle equivalent to $[\mathrm{Nor}_{\mathsf{Circ}\ 4}, (2130)]$ (and $[\mathrm{Nor}_{\mathsf{Circ}\ 4}, (2310)]$), although not cycle functionally equivalent, in fact, discarding labels, their periodical orbits coincide.

1.5.1 Functional equivalence

Recalling the notion of adjacent permutations, we have some basic results on functionally equivalent SDS.

Proposition 1.5.3 ([MR08])**.** *Let Γ be a graph, let $(F_i)_i$ be a family of Γ-local maps. Then, adjacent permutations induce the same SDS maps, i.e.,*

$$\pi \sim_\Gamma \pi' \Rightarrow [F_\Gamma, \pi] = [F_\Gamma, \pi'].$$

As a consequence, the number of connected components of the update graph $U(\Gamma)$ is an upper bound for the number of functionally different SDS that can be generated only varying the permutation. Now, recall Proposition 1.4.4 in order to state the following result.

Proposition 1.5.4 ([MR08])**.** *Let Γ be a combinatorial graph, let $(F_i)_i$ be a family of Γ-local functions, then*

$$|\{[F_\Gamma, \pi] \colon \pi \in S_\Gamma\}| \leq |\alpha(\Gamma)|,$$

and the bound is sharp.

It is important to stress that not only the number of acyclic orientations gives an upper bound for the different SDS maps on Γ: for some SDS, any non adjacent permutations correspond to different dynamics. The result has been extended to general word update orders in [Rei06].

Example 1.5.5. The update functions nor is used in the proof of sharpness presented in [MR08]: indeed, what we can see in Examples 1.3.3 and 1.4.3 is that $\pi = 0123$, $\pi' = 1230$, $\pi'' = 2130$ and $\pi'''' = 0231$, belonging to different connected components in $U(\Gamma)$, have different phase spaces.

1.5.2 Dynamic equivalence

As for dynamical equivalence, it can be useful to restate the defining relation in terms of a conjugation. If the update functions are $\mathrm{Aut}(\Gamma)$-invariant, then it is shown in [MR08] that two SDS maps over update sequences in the same orbit are conjugate, and the relation is

$$\gamma \circ [F_v, \pi] \circ \gamma^{-1} = [F_v, \gamma\pi].$$

This induces an equivalence relation on the permutation group S_Γ (and as well on $\mathrm{Acyc}(\Gamma)$), which is denoted by $\sim_{\overline{\alpha}}$. The number of orbits $\overline{\alpha}(\Gamma)$ under the action of $\mathrm{Aut}(\Gamma)$ on $S_\Gamma / \sim_\alpha$

$$\overline{\alpha}(\Gamma) \colon = \alpha(\Gamma)| / \sim_\alpha |$$

has been proved to be an upper bound for conjugation classes of SDS maps, hence of dynamically equivalent SDS-maps.

Burnside's lemma makes it possible to compute $\overline{\alpha}(\Gamma)$ via the number of acyclic orientations of the orbit graph of the cyclic group $\langle\gamma\rangle$.

Proposition 1.5.6. *Let Γ be a combinatorial graph, and let F_Γ be a family of Γ-local functions induced by symmetric functions. Denote by $\langle\gamma\rangle \setminus \Gamma$ the orbit graph of the cyclic group $G = \langle\gamma\rangle$. Then,*

$$\overline{\alpha}(\Gamma) \leq \frac{1}{|\mathrm{Aut}(\Gamma)|} \sum_{\gamma \in \mathrm{Aut}(\Gamma)} \alpha(\langle\gamma\rangle \setminus \Gamma)$$

This bound is proved in [BMR03] to be sharp for some special graphs classes: Circ_n, Wheel_n, Star_n.

Example 1.5.7. Example 1.3.3 pointed out that the two SDS $[\mathrm{Nor}_{\mathsf{Circ}\ 4}, (0123)]$ and $[\mathrm{Nor}_{\mathsf{Circ}\ 4}, (1230)]$ were dynamically equivalent. Indeed, applying the conjugation via element 0,

$$0\pi0 = 0(0123)0 = 001230 = 1230 = \pi'.$$

In [MM11] it is conjectured that the bound introduced in Proposition 1.5.6 is sharp for any combinatorial graph.

Remark 1.5.8. When $\mathrm{Aut}(\Gamma)$ is trivial, then α and $\overline{\alpha}$ coincide, as dynamical equivalence of SDS only measures the eventual symmetries of the base graph.

1.5.3 Cycle equivalence

Let us define an operation which modifies the acyclic orientation of a graph: choose a source (or sink) vertex, and reverse all the orientations of the incident edges. This operation is called *source-to-sink* operation, or a *click*, and was first defined in [BLS91].

It is easy to see that this operation is well defined, as switching a source into a sink (or vice-versa) does not change the number of cycles in an oriented graph.

Definition 1.5.9. Two acyclic orientations $\mathcal{O}_\Gamma$ and $\mathcal{O}'_\Gamma$ will be called *click equivalent* if it is possible to obtain one from another applying some click operations.

This induces an equivalence relation both on the set of acyclic orientations $\alpha(\Gamma)$ and on the group of permutations S_Γ, which is denoted by $\sim_\kappa$ in both cases.

It was shown in [MM09] that if two permutations are *click equivalent*, then the corresponding SDS are cycle equivalent. Therefore, fixed a combinatorial graph Γ and a family of update functions $(F_v)_v$, the number

$$\kappa(\Gamma) \colon = |\operatorname{Acyc}(\Gamma)/\sim_\kappa|$$

is an upper bound for the possible SDS, up to cycle equivalence.

Example 1.5.10. Back to Example 1.5.2, $[\mathrm{Nor}_{\mathsf{Circ}\ 4}, (2130)]$and $[\mathrm{Nor}_{\mathsf{Circ}\ 4}, (0213)]$ are cycle equivalent. In terms of acyclic orientations, we can go from $\mathcal{O}^{(2130)}_{\mathsf{Circ}\ 4}$ to $\mathcal{O}^{(0213)}_{\mathsf{Circ}\ 4}$ simply transforming vertex 0 from a source to a sink. In fact, the two induced orientations are

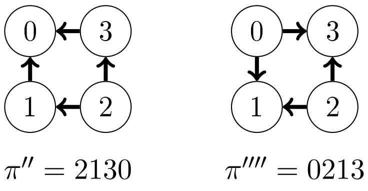


Figure 1.4: The acyclic orientations $\mathcal{O}^{(2130)}_{\mathsf{Circ}\ 4}$ and $\mathcal{O}^{(0213)}_{\mathsf{Circ}\ 4}$.

It was shown that, just like α, also κ is a Tutte invariant.

Remark 1.5.11. As dynamically equivalent maps are in particular cycle equivalent maps, if $\operatorname{Aut}(\Gamma)$ is nontrivial it is possible to define a stronger equivalence relation $\sim_{\overline{\kappa}}$ over $\operatorname{Acyc}(\Gamma)$. In this case, the number

$$\overline{\kappa}(\Gamma) \colon = |\operatorname{Acyc}(\Gamma)/\sim_{\overline{\kappa}}|$$

is a sharper upper bound for the different SDS, up to cycle equivalence.

Example 1.5.12. Back to Example 1.3.3, there are exactly 2 cycle configurations for the graph Circ_4 and update functions Nor: $[\mathrm{Nor}_{\mathsf{Circ}\ 4}, (0123)]$ and $[\mathrm{Nor}_{\mathsf{Circ}\ 4}, (1230)]$ are in the first cycle configuration, $[\mathrm{Nor}_{\mathsf{Circ}\ 4}, (2130)]$, $[\mathrm{Nor}_{\mathsf{Circ}\ 4}, (2310)]$ and $[\mathrm{Nor}_{\mathsf{Circ}\ 4}, (0213)]$ belong to the other.

1.6 The word problem and functional equivalence

Results in this section are proved in [MM09] and [MM10].

In order to appreciate the interesting analogies between SDS theory and Coxeter theory, let us recall some background material.

Definition 1.6.1. A Coxeter system is a pair (W, S), where $S = \{s_1, \dots, s_n\}$ is a finite set of involutions and W is a *Coxeter group* defined via the following presentation

$$W = \langle s_1, \dots, s_n \colon s_i^2 = 1, \quad (s_i s_j)^{m(s_i,s_j)} = 1 \rangle,$$

where $m(s_i, s_j) \geq 2$ for $i \neq j$.

Let $F(S)$ be the free monoid over S. For each integer $m \geq 0$ and distinct generators s, t,

$$\langle s, t \rangle_m \colon = \underbrace{stst\dots}_{m} \in F(S).$$

A relation of the form

$$\langle s, t \rangle_{m(s,t)} = \langle t, s \rangle_{m(s,t)}$$

is called a *braid relation*.

A Coxeter group is an example of finitely presented group, as it has a finite set of generators S and a finite number of defining relations. For this kind of group a classic question arises.

Question 1. (The word problem for Coxeter groups) Given two words $w, w' \in F(S)$, when do they correspond to the same group element?

However not being solvable for a generic group, there is a complete answer for Coxeter groups, given by the following result.

Theorem 1.6.2 (Matsumoto, [Mat64])**.** *Let W be a Coxeter group, then any two reduced expressions for the same element differ by braid relations.*

Remark 1.6.3. This is equivalent as saying that, going from a reduced expression to another one, there is no need to apply conjugations ("add letters") nor of using relations $s^2 = 1$ ("delete letters").

The word problem can be considered in SDS setting, too.

Question 2. (The word problem for SDS) Let Γ be a combinatorial graph, let K be the (finite) state set on each vertex and let $(F_v)_v$ be a list of update rules. Given two update sequences $w, w' \in F(V)$, when do the SDS we obtain have identical phase space? In other words, when two update sequences give rise to functionally equivalent SDS?

As the graph Γ and state sets S_i are finite, this problem can always be solved computationally. It would be interesting to find an equivalent result to Matsumoto Theorem: however, we now state some results from [MM10], where it is shown how conjugation of Coxeter elements correspond to cycle equivalence of SDS maps.

1.7 w-independent SDS

The property of an SDS to be w-independent will be used in the following section, in order to state an analogy with Coxeter groups. We introduce the notion of Asynchronous Cellular Automata (ACA) as it has been proved that many of them are examples of w-independent SDS.

Definition 1.7.1. A word $w \in F(V)$ is said *fair* if and only if each element in V do appear in w at least once.

Definition 1.7.2. An SDS is *w-independent* if there exists a subset $P \subset S$ such that for all fair words w,

$$\mathrm{Per}[F_\Gamma, w] = P.$$

Remark 1.7.3. It is important to stress that, for w-independent words, the phase space is independent as set from the permutation chosen.

The property of being w-independent is more common than it appears. The following two examples are presented in [MR08].

Example 1.7.4. For a combinatorial graph Γ and a fair word w, the SDS $(\Gamma, \mathrm{Nor}_\Gamma, w)$ is w-independent. Its dynamic group $\mathcal{DG}(\mathrm{Nor}_\Gamma)$ acts transitively on $P = \mathrm{Per}[\mathrm{Nor}_\Gamma, w]$.

For the combinatorial graph Circ_n $(n \geq 3)$, there are exactly 11 symmetric Boolean functions that induce w-independent SDS. They are

- nor_3 and nand_3;
- $(\mathrm{nor}_3 + \mathrm{nand}_3)$ and $(1 + \mathrm{nor}_3 + \mathrm{nand}_3)$;
- $_3$, or_3 and $\mathrm{majority}_3$;
- parity_3 and $(1 + \mathrm{parity}_3)$;
- the constant maps 0 and 1.

Example 1.7.5. On function nor_3 is w-independent on Circ_4: in fact, in example 1.3.3, all phase spaces have the same periodic points: 0000, 1010, 0001, 0100, 0010, 1000 and 0101.

In [MMM08] it is introduced the notion on Asynchronous Cellular Automata (or ACA), which is a special case of SDS whose base graph is either a lattice or a graph Circ_n. Its definition coincides with that of a Cellular Automata (or CA) apart from the application of update rules, which in ACA is asynchronous, while in CAit is synchronous.

If the vertex sets are $K = \{0, 1\}$, then each local function f_i calculates the new state on vertex i taking $\{x_{i-1}, x_i, x_{i+1}\}$ as input. Hence, there are eight possible outputs $a_0, a_1, ..., a_7$, one for each of the nine possible inputs $000, 001, ..., 111$. Following a notation introduced in [Wol83], any of the $2^8 = 256$ possible rule can be represented as an integer r smaller than 256, as

$$r = \sum a_i 2^i.$$

The corresponding sequence of local functions is denoted by ECA_r and it is called *Elementary Cellular Automata* or *Wolfram rule r*.

It has been proved in [MMM08] that exactly 104 of the 256 possible update rules give rise to a π-independent SDS. Their dynamics groups are described in [MMM11].

1.8 Coxeter groups and cycle equivalence

Recall that for each Coxeter group it is possible to define a *Coxeter graph* Γ as follows: its vertex set correspond to the set of generators $S = \{s_1, \dots, s_n\}$, and there is an edge (i, j) whenever the corresponding generators s_i and s_j do not commute, its edge label being $m(s_i, s_j)$.

Each Coxeter element $c = s_1 \dots s_n$ induces a partial order on the set of its generators S, hence it induces an acyclic orientation on Γ so that the edge (i, j) assumes the orientation $i \to j$ if and only if s_i appears before s_j in c.

When we conjugate the element $c = s_1 \dots s_n$ by its initial letter s_1, the result is

$$s_1 c s_1 = s_1(s_1 \dots s_n)s_1 = s_2 \dots s_n s_1.$$

The corresponding acyclic orientations coincide in all edges but those including vertex 1: indeed, being the first on the left in element c, 1 was a source, hence all vertices $(1, j)$ were oriented $1 \to j$; now 1 is the rightmost element in $s_2 \dots s_n s_1$, hence it has become a sink, as the same edges are now oriented $j \to i$.

The following result connects the acyclic orientations on graphs and Coxeter elements.

Theorem 1.8.1 ([EE09]). *Let $\sim_\kappa$ be the equivalence relation induced on* $\mathrm{Acyc}(\Gamma)$ *by the source-to-sink conversion Then, given two graph orientations $\mathcal{O}_\Gamma^c$ and $\mathcal{O}_\Gamma^{c'}$, we have*

$$\mathcal{O}_\Gamma^c \sim_\kappa \mathcal{O}_\Gamma^{c'} \textit{ if and only if } c \textit{ and } c' \textit{ are conjugate.}$$

An analogous result involves κ-equivalence of acyclic orientations and cycle equivalence of permutation SDS.

Theorem 1.8.2 ([MM09]). *If two acyclic orientations are cycle equivalent, i.e. $\mathcal{O}_\Gamma^\pi \sim_\kappa \mathcal{O}_\Gamma^\sigma$, then the induced SDS maps $[F_\Gamma, \pi]$ and $[F_\Gamma, \sigma]$ are cycle equivalent.*

1.9 The dynamic group of an SDS

Consider the SDS whose vertex state is $S = \mathbb{F}^2 = \{0, 1\}$, and a sequence of π-independent local functions $\mathbf{F} = (\mathbf{F}_1, \dots, \mathbf{F}_n)$, i.e. such that $\mathrm{Per}(\mathbf{F}_\pi) = \mathrm{Per}(\mathbf{F}_\sigma)$, for all the possible permutations $\pi, \sigma \in S_n$. In particular, each local function permutes the set of periodic points, and nothing is said about the orbits.

Definition 1.9.1. The *dynamics group* of $\mathbf{F}$ is the group generated by the local functions $\mathbf{F}_i$, and it is denoted by $\mathcal{DG}(F)$.

Let $\mathbf{F}_i^\star$ be the restriction of $\mathbf{F}_i$ to the periodic points $\mathrm{Per}(\mathbf{F}_\pi)$. Since $\mathbf{F}_i$ eventually changes only the state of the i-th vertex, and the state set is $S_i = \mathbb{F}^2$, then

- $\mathbf{F}_i^\star \circ \mathbf{F}_i^\star = \mathrm{Id}_{\mathrm{Per}(\mathbf{F}_\pi)}$;
- $(\mathbf{F}_i^\star \circ \mathbf{F}_j^\star)^{m_{ij}} = \mathrm{Id}_{\mathrm{Per}(\mathbf{F}_\pi)}$, where $m_{ij} = |\,\mathbf{F}_i^\star \circ \mathbf{F}_j^\star\,|$.

Hence, it is possible to define a surjective map

$$\langle s_1, \ldots, s_n \colon s_i^2 = 1, (s_i s_j)^{m_{ij}} = 1 \rangle \longrightarrow \mathcal{DG}(F).$$

Here we see that the dynamics group of a generic SDS is naturally a quotient of a Coxeter group, and it leads to the following question.

Question 3. Give a presentation of the dynamics group of the SDS based on the update functions.

Update systems, which we define and study in Chapters 2 and 3, can be seen as an attempt to describe the dynamics of a structure similar to SDS, even if on a directed acyclic graph, without fixing a given order in which to update the states.

2 Update systems on the complete graph

The content of this chapter reflects the preprint [CD13], where the recently defined notion of Hecke-Kiselman monoid finds a natural realization on the combinatorial-computational setting of Sequential Dynamical Systems.

Let Q be a mixed graph, i.e., a simple graph with at most one connection between each pair of distinct vertices; connections can be either oriented (arrows) or non-oriented (edges). In [GM11], Ganyushkin and Mazorchuk associated with Q a semigroup $\mathbf{HK}_Q$, subject to the following relations

- $a_i a_j = a_j a_i$, if i and j are not connected;
- $a_i a_j a_i = a_j a_i a_j$, if $(i \text{ --- } j) \in Q$, i.e., i and j are connected by an edge;
- $a_i a_j = a_i a_j a_i = a_j a_i a_j$, if $(i \rightarrow j) \in Q$, i.e., i and j are connected by an arrow from i to j.

The semigroup $\mathbf{HK}_Q$ is known as the *Hecke-Kiselman monoid* attached to Q.

For the two extremal types of mixed graphs — graphs, where all sides are edges, and oriented graphs, in which all sides are arrows — Hecke-Kiselman monoids are well understood: when Q is an oriented graph with n vertices and no directed cycles, then $\mathbf{HK}_Q$ is isomorphic to a quotient of Kiselman's semigroup K_n [Kis02, KM09], which is known to be finite [KM09].

On the other hand, when Q has only unoriented edges, $\mathbf{HK}_Q$ is finite if and only if Q is a disjoint union of finite simply laced Dynkin diagrams, and the corresponding semigroup is then variously known as *Springer-Richardson*, 0*-Hecke*, or *Coxeter monoid* attached to Q. The problem of characterizing mixed graphs inducing a finite Hecke-Kiselman monoid in the general mixed case seems to be difficult, and only very partial results are known [AD13]. The study of (certain quotients of) Hecke-Kiselman monoids and their representations has also attracted recent interest, see for instance [For12, Gre12, GM13].

The choice of a simple (i.e., without loops or multiple edges) graph Γ is also one of the essential ingredients in the definition of a Sequential Dynamical System (SDS), which are described in the previous chapter. SDS on directed acyclic graphs are related to Hecke-Kiselman monoids in that the so-called Γ-local functions [MR08] satisfy the relations listed in the presentation of $\mathbf{HK}_\Gamma$; in other words, the evaluation morphism mapping each word (or update schedule) in V to the corresponding composition of Γ-local functions factors through $\mathbf{HK}_\Gamma$.

One is naturally led to wonder whether $\mathbf{HK}_\Gamma$ is the smallest quotient through which the above evaluation morphisms must factor, or additional universal relations among Γ-local functions may be found. Henceforth, $\Gamma_n = (V, E)$ will denote the oriented graph where $V = \{1, \ldots, n\}$ and $(i, j) \in E$ if and only if $i < j$. In this chapter, we show that $\mathbf{HK}_\Gamma$ is optimal in the special case $\Gamma = \Gamma_n$, by proving the following statement.

Theorem 2.0.2. *There exists an update scheme $\mathcal{S}_n^\star = (\Gamma_n, S_i, f_i)$ such that the associated evaluation morphism factors through no nontrivial quotient of $\mathbf{HK}_{\Gamma_n} = \mathrm{K}_n$.*

We believe that the same claim holds for every finite directed acyclic graph, and mention Chapter 3 some evidence in support of this conjecture.

This chapter is structured as follows. In Section 2.1 we recall the definition of SDS, define the dynamics monoid of an update system supported on an oriented graph Γ, and show that it is a quotient of the Hecke-Kiselman monoid $\mathbf{HK}_\Gamma$ as soon as Γ has no oriented cycles. In Sections 2.2 and 2.3 we list the results on Kiselman's semigroup K_n that are contained in [KM09], and derive some useful consequences. Section 2.4 introduces the *join operation*, which is the key ingredient in the definition of the update system $\mathcal{S}_n^\star$, which is given in Section 2.5. Sections 2.6 and 2.7 are devoted to the proof of Theorem 2.6.2, which directly implies the above-mentioned result. Finally, Section 2.8 contains some more results on the dynamics of the SDS on the complete graph

2.1 Sequential dynamical systems

An *update system* is a triple $\mathcal{S} = (\Gamma, (S_i)_{i \in V}, (f_i)_{i \in V})$ consisting of

1. a *base graph* $\Gamma = (V, E)$, which is a finite directed graph, with V as vertex set and $E \subseteq V \times V$ as edge set; we will write $i \to j$ for $(i, j) \in E$. The *vertex neighbourhood* of a given vertex $i \in V$ is the subset
$$x[i] = \{j : i \to j\}.$$
2. a collection $S_i, i \in V$ of finite sets of *states* . We denote by $S = \prod_{i \in V} S_i$ the family of all the possible *system states*, i.e., n-tuples $\mathbf{s} = (s_i)_{i \in V}$, where s_i belongs to S_i for each vertex i. The state neighbourhood of $i \in V$ is $S[i] = \prod_{j \in x[i]} S_j$, and the restriction of $\mathbf{s} = (s_j)_{j \in V}$ to $x[i]$ is denoted by
$$\mathbf{s}[i] = (s_j)_{j \in x[i]} \in S[i].$$
3. for every vertex i, a *vertex (update) function*
$$f_i : S[i] \quad \to \quad S_i,$$
computes the new state value on vertex i as a function of its state neighbourhood. In particular, if $x[i]$ is empty, then f_i is a constant $t \in S_i$, and we will write $f_i \equiv t$. Each vertex function f_i is a constant $t \in S_i$ and we will write $f_i \equiv t$. Each vertex function can be incorporated into a Γ-*local function* $\mathbf{F}_i : S \to S$ defined as
$$\mathbf{F}_i(\mathbf{s}) = \mathbf{t} = (t_j)_{j \in V}, \text{ where } t_j = \begin{cases} s_j, & \text{if } i \neq j \\ f_i(\mathbf{s}[i]), & \text{if } i = j \end{cases}$$

An SDS is an update system $\mathcal{S}$ endowed with

4. an *update schedule*, i.e., a word $w = i_1 i_2 \dots i_k$ in the free monoid $\mathrm{F}(V)$ over the alphabet V (just like we did in the first chapter, we will often abuse the notation denoting both the alphabet and the vertex set with the same letter V, and both letters and vertices with the same symbols). The update schedule w induces a dynamical system map (SDS map), or an *evolution* of S, $\mathbf{F}_w \colon S \to S$, defined as

$$\mathbf{F}_w = \mathbf{F}_{i_1} \mathbf{F}_{i_2} \dots \mathbf{F}_{i_k} .$$

Remark 2.1.1. As the graph Γ sets up a dependence relation between nodes under the action of update functions, it makes sense to allow Γ to possess self-loops, and arrows connecting the same vertices but going in opposite directions. However, we exclude the possibility of multiple edges between any two given vertices. Notice, finally, that all SDS of interest, in this and the following chapter, will be supported on directed *acyclic* graphs, thus excluding in particular the possibility of self-loops.

Denote by $\mathrm{End}(S)$ the set of all maps $S \to S$, with the monoid structure given by composition. Then, the Γ-local functions $\mathbf{F}_i$, $i \in V$, generate a submonoid of S which we denote by $D(\mathcal{S})$. The monoid $D(\mathcal{S})$ is the image of the natural homomorphism

$$\begin{array}{rcl}\mathbf{F} \colon \mathrm{F}(V) & \to & \mathrm{End}(S) \\ w & \mapsto & \mathbf{F}_w .\end{array}$$

mapping each update schedule w to the corresponding evolution $\mathbf{F}_w$; in particular, we denote by $\mathbf{F}_\star$ the identity map, induced by the empty word $\star$.

Once an underlying update system has been chosen, our goal is to understand the monoid structure of $D(\mathcal{S})$.

Example 2.1.2. Let $\Gamma = (\{i\}, \emptyset)$ be a Dynkin graph of type A_1. It has only one vertex i, so there is only one vertex function f_i, which is constant, as there are no arrows starting in i. The system dynamics monoid $D(\mathcal{S}) = \{\mathbf{F}_\star, \mathbf{F}_i\}$ contains exactly two elements, as soon as $|S_i| > 1$.

Example 2.1.3. Let Γ be the graph Let us consider $S_i = \{0, 1, 2\}$, $S_j = \{0, 1\}$. Set up

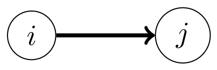


Figure 2.1: Line$_2$.

an update system on Γ by requiring that $f_i \colon S[i] = S_j \to S_i$ acts as $f_i(s) = s + 1$, and $f_j \equiv 1$. Evolutions induced by words $\star$ (the empty word), i, j, ij and ji on $S = S_i \times S_j$ all differ from each other, as they take different values on $(0, 0)$:

$$\begin{array}{rcl}\mathbf{F}_\star(0,0) & = & (0,0) \\ \mathbf{F}_i(0,0) & = & (1,0) \\ \mathbf{F}_j(0,0) & = & (0,1) \\ \mathbf{F}_{ij}(0,0) = \mathbf{F}_i \mathbf{F}_j(0,0) = \mathbf{F}_i(0,1) & = & (2,1) \\ \mathbf{F}_{ji}(0,0) = \mathbf{F}_j \mathbf{F}_i(0,0) = \mathbf{F}_j(1,0) & = & (1,1).\end{array}$$

Both $\mathbf{F}_i$ and $\mathbf{F}_j$ are idempotent. Moreover, it is easy to see that

$$\mathbf{F}_{iji} = \mathbf{F}_{jij} = \mathbf{F}_{ij},$$

as $\mathbf{F}_i\,\mathbf{F}_j\,\mathbf{F}_i = \mathbf{F}_j\,\mathbf{F}_i\,\mathbf{F}_j = \mathbf{F}_i\,\mathbf{F}_j$, so that $\mathbf{F}_i \mapsto a_1$, $\mathbf{F}_j \mapsto a_2$ extends to an isomorphism from $D(\mathcal{S})$ to Kiselman's semigroup K_2,

$$\mathrm{K}_2 = \langle a_1, a_2\colon \quad a_1^2 = a_1, \quad a_2^2 = a_2 \quad a_1a_2a_1 = a_2a_1a_2 = a_1a_2\rangle.$$

These examples are instances of the following statement.

Proposition 2.1.4. *Let $\mathcal{S} = (\Gamma, S_i, f_i)$ be an update system defined on a directed acyclic graph Γ. Then the Γ-local functions $\mathbf{F}_i$ satisfy:*

(*i*) $\mathbf{F}_i^2 = \mathbf{F}_i$;

(*ii*) $\mathbf{F}_i\,\mathbf{F}_j\,\mathbf{F}_i = \mathbf{F}_j\,\mathbf{F}_i\,\mathbf{F}_j = \mathbf{F}_i\,\mathbf{F}_j$, *if* $i \to j$ *(and hence,* $j \nrightarrow i$*);*

(*iii*) $\mathbf{F}_i\,\mathbf{F}_j = \mathbf{F}_j\,\mathbf{F}_i$, *if* i *and* j *are not connected.*

Proof. Every Γ-local function $\mathbf{F}_i$ only affects the vertex state s_i. As $\mathbf{F}_i(\mathbf{s})$ only depends on $\mathbf{s}[i]$, and $i \notin x[i]$, then each $\mathbf{F}_i$ is idempotent. For the same reason, it is enough to check (*ii*) and (*iii*) on a graph with two vertices $i \neq j$. If they are not connected, then both f_i and f_j are constant, hence $\mathbf{F}_i$ and $\mathbf{F}_j$ trivially commute. If there is an arrow $i \to j$, then $\mathbf{F}_i(a_i, a_j) = (f_i(a_j), a_i)$, whereas $\mathbf{F}_j(a_i, a_j) = (a_i, t)$, since $f_j \equiv t$ is a constant. Then it is easy to check that the compositions $\mathbf{F}_i\,\mathbf{F}_j$, $\mathbf{F}_i\,\mathbf{F}_j\,\mathbf{F}_i$,$\mathbf{F}_j\,\mathbf{F}_i\,\mathbf{F}_j$ coincide, as they map every element to $(f_i(t); t)$. □

These relations are remindful of those in the presentation of a Hecke-Kiselman monoid.

Definition 2.1.5. Let $\Gamma = (V, E)$ be a finite directed acyclic graph. The *Hecke-Kiselman monoid* associated with Γ is defined as follows

$$\begin{aligned} \mathbf{HK}_\Gamma = \langle a_i, i \in V\colon \qquad & a_i^2 = a_i, \text{ for every } i \in V; \\ & a_ia_ja_i = a_ja_ia_j = a_ia_j, \text{ for } i \to j; \qquad (2.1.1) \\ & a_ia_j = a_ja_i, \text{ for } i \nrightarrow j, \text{ and } j \nrightarrow i\rangle \qquad (2.1.2) \end{aligned}$$

This structure has been first introduced in [GM11] for a finite mixed graph, i.e., a simple graph (without loops or multiple edges) in which edges can be both oriented and unoriented: there, an unoriented edge (i, j) is used to impose the customary braid relation $a_ia_ja_i = a_ja_ia_j$.

If $\mathcal{S} = (\Gamma, S_i, \mathbf{F}_i)$ is an update system on a finite directed acyclic graph $\Gamma = (V, E)$, then Proposition 2.1.4 amounts to claiming that the evaluation homomorphism $\mathbf{F}\colon \mathrm{F}(V) \to \mathrm{End}(S)$ factors through the Hecke-Kiselman monoid $\mathbf{HK}_\Gamma$.

Our case of interest is when the graph $\Gamma = \Gamma_n$ is the complete graph on n vertices, where the orientation is set so that $i \to j$ if $i < j$. In this case, the semigroup $\mathbf{HK}_{\Gamma_n}$ coincides with Kiselman's semigroup K_n, as defined in [KM09]. The monoid K_n, however, only reflects immediate pairwise interactions between vertex functions. One may, in principle, wonder if K_n is indeed the largest quotient of $\mathrm{F}(V)$ through which evaluation maps $\mathbf{F}$ factor, or additional identities may be imposed that reflect higher order interactions.

In section 2.5 we will exhibit an update system $\mathcal{S}_n^\star$, defined on the graph Γ_n, whose dynamics monoid is isomorphic to K_n. In other words, we will show that $\mathrm{K}_n \to D(\mathcal{S}_n^\star)$ is indeed an isomorphism, once suitable vertex functions have been chosen.

2.2 Combinatorial definitions

Let $\mathrm{F}(A)$ be the free monoid over the alphabet A, and denote by $\star$ the empty word. Recall that, for every subset $B \subseteq A$, the submonoid $\langle B \rangle \subseteq \mathrm{F}(A)$ is identified with the free monoid $\mathrm{F}(B)$.

Definition 2.2.1. Let $w \in \mathrm{F}(A)$. We define

- *subword of* w to be a substring of consecutive letters of w;
- *quasi-subword of* w to be an ordered substring u of not necessarily consecutive letters of w.

We will denote the relation of being a quasi-subword by $\leq$, so that

$$v \leq w$$

if and only if v is a quasi-subword of w.

Obviously, every subword is a quasi-subword. Also notice that $v \leq w$ and $w \leq v$ if and only if $v = w$.

Example 2.2.2. Set $w = acaab \in \mathrm{F}(\{a, b, c\})$; then

- aab is a subword (hence a quasi-subword) of w;
- aaa is a quasi-subword of w which is not a subword;
- abc is neither a subword nor a quasi-subword of w.

Trivial examples of subwords of w are the empty word $\star$, and w itself.

Definition 2.2.3. Let $w \in \mathrm{F}(A)$.Then

- if w is non-empty, the *head* of w, denoted $\mathrm{h}(w) \in A$, is the leftmost letter in w;
- if $a \in A$, then the *a-truncation* $\mathrm{T}_a\, w \in \mathrm{F}(A)$ of w is the longest (non-empty) suffix of w with head a, or the empty word in case a does not occur in w.

Similarly, if $I \subset A$, we denote by $\mathrm{T}_I\, w$ the longest (non-empty) suffix of w whose head lies in I, or the empty word in case no letter from I occurs in w.

The following observations all have trivial proofs.

Remark 2.2.4.

(*i*) If $w \in \mathrm{F}(A)$ does not contain any occurrence of $a \in A$, then

$$\mathrm{T}_a(ww') = \mathrm{T}_a(w'),$$

for every $w' \in \mathrm{F}(A)$.

(*ii*) For every $w \in \mathrm{F}(A)$, and $a \in A$, one may (uniquely) express w as

$$w = w' \, \mathrm{T}_a \, w$$

where $w' \in \langle A \setminus \{a\} \rangle$.

(*iii*) If $w \in \mathrm{F}(A)$ contains some occurrence of $a \in A$, then

$$\mathrm{T}_a(ww') = (\mathrm{T}_a \, w)w',$$

for every $w' \in \mathrm{F}(A)$.

Remark 2.2.5.

(*i*) If $w \in \mathrm{F}(A)$, and $a, b \in A$, then either $\mathrm{T}_a \, w$ is a suffix of $\mathrm{T}_b \, w$, or vice versa.

(*ii*) If $\mathrm{T}_b \, w$ is a suffix of $\mathrm{T}_a \, w$ for all $b \in A$, then $\mathrm{h}(w) = a$, hence $\mathrm{T}_a \, w = w$.

(*iii*) $\mathrm{T}_I \, w = \mathrm{T}_a \, w$ for some $a \in I$, and $\mathrm{T}_b \, w$ is a suffix of $\mathrm{T}_I \, w$ for every $b \in I$.

Definition 2.2.6. Given $I \subseteq A$, the *deletion morphism* is the unique semigroup homomorphism satisfying

$$\begin{array}{rcl} \partial_I \colon \mathrm{F}(A) & \to & \mathrm{F}(A \setminus I) \subseteq \mathrm{F}(A) \\ a_i & \mapsto & \star, \text{ for } i \in I \\ a_j & \mapsto & a_j, \text{ for } j \notin I. \end{array}$$

It associates with any $w \in \mathrm{F}(A)$ the longest quasi-subword of w containing no occurrence of letters from I.

Remark 2.2.7. For every $I, J \subseteq A$,

$$\partial_I \, \partial_J = \partial_{I \cup J}$$

Lemma 2.2.8. *If $a \notin I \subset A$, then*

$$\partial_I \, \mathrm{T}_a \, w = \mathrm{T}_a \, \partial_I w$$

for every $w \in \mathrm{F}(A)$.

Proof. Write $w = w' \,\mathrm{T}_a\, w$. Then, by Remark 2.2.4 (*i*),

$$\begin{aligned}
\mathrm{T}_a\, \partial_I w &= \mathrm{T}_a\, \partial_I \left(w'\, \mathrm{T}_a\, w\right) \\
&= \mathrm{T}_a \left(\partial_I w'\, \partial_I\, \mathrm{T}_a\, w\right) \\
&= \mathrm{T}_a\, \partial_I\, \mathrm{T}_a\, w && (2.2.1) \\
&= \partial_I\, \mathrm{T}_a\, w && (2.2.2)
\end{aligned}$$

In (2.2.1) we use that a does not occur in $\partial_I w'$, as, by definition of w', a does not occur in w'; then (2.2.2) holds because either the head of $\mathrm{T}_a\, w$ is a or $\mathrm{T}_a\, w = \star$, hence the same holds after applying ∂_I, as $a \notin I$. □

2.3 Kiselman's semigroup and canonical words over the complete graph

In this section, we recall results from [GM11], and draw some further consequences.

Henceforth, when $h \leq k$ are natural numbers, we will denote by $[h, k] = \{h, h+1, \ldots, k\}$ the corresponding segment. Choose an alphabet $A = \{a_i, i \in [1, n]\}$; Kiselman's semigroup K_n has the presentation

$$\mathrm{K}_n = \langle a_i \in A\colon \quad a_i^2 = a_i \text{ for every } i, \quad a_i a_j a_i = a_j a_i a_j = a_i a_j \text{ for } i < j\rangle.$$

In accordance with [KM09], let

$$\pi\colon\ \mathrm{F}(A) \to \mathrm{K}_n$$

denote the canonical evaluation epimorphism.

Definition 2.3.1 ([GM11])**.** Let $w \in \mathrm{F}(A)$. A subword of w of the form $a_i u a_i$, where $a_i \in A$ and $u \in \mathrm{F}(A)$, is *special* if u contains both some a_j, with $j > i$, and some a_k, with $k < i$.

Remark 2.3.2. Notice that a subword $a_i u a_i$ cannot be special if $i = 1$ or $i = n$.

Let us recall the following fact.

Theorem 2.3.3 ([KM09])**.** *Let $w \in \mathrm{F}(A)$. The set $\pi^{-1}\pi(w)$ contains a unique element whose only subwords of the form $a_i u a_i$ are special.*

Remark 2.3.4.

(*i*) The unique element described in Theorem 2.3.3 contains at most one occurrence of a_1 and at most one occurrence of a_n.

(*ii*) In order to prove Theorem 2.3.3, the authors of [KM09] define a binary relation $\to$ in $\mathrm{F}(A)$ as follows: $w \to v$ if and only if either

($\xrightarrow{1}$) $w = w_1 a_i a_i w_2$, $v = w_1 a_i w_2$, or

($\xrightarrow{2}$) $w = w_1 a_i u a_i w_2$, $v = w_1 a_i u w_2$ and $u \in \langle a_{i+1}, \dots, a_n \rangle$, or

($\xrightarrow{3}$) $w = w_1 a_i u a_i w_2$, $v = w_1 u a_i w_2$ and $u \in \langle a_1, \dots, a_{i-1} \rangle$.

It is possible to iterate such simplifications, and write (finite) sequences

$$w \to v_1 \to v_2 \cdots \to v_k, \tag{2.3.1}$$

so that each word contains exactly one letter less than the previous, and they all belong to the same fiber with respect to π. Moreover, each v_i is a quasi-subword of w and of v_j, for all $j < i$.

A sequence like (2.3.1) is called *simplifying sequence* if v_k is not simplifiable any further, and each $v_i \to v_{i+1}$ is called a *simplifying step*. It should be stressed that a simplifying step of type 1 can be seen as both of type 2 and of type 3.

(*iii*) According to Theorem 2.3.3, every simplifying sequence for w ends on the same word.

We will refer to any word, whose all subwords of the form $a_i u a_i$ are special, as a *canonical word*, so that the above theorem claims existence of a unique canonical word v in each $\pi^{-1}\pi(w), w \in \mathrm{F}(A)$. Thus, the assignment $w \mapsto v$ is a well defined map $\mathrm{Can} : \mathrm{F}(A) \to \mathrm{F}(A)$ associating with each word its unique *canonical form*. Notice that w is canonical if and only if $w = \mathrm{Can}\, w$.

Remark 2.3.5.

(*i*) If w is canonical then all of its subwords are canonical.

(*ii*) A word $u \in \langle a_i, i \in [h,k] \rangle$ is canonical if and only if, for every $j < h$ or $j > k$, the word $a_j u$ is canonical. Moreover,

$$\mathrm{Can}(a_j u) = a_j \,\mathrm{Can}\, u.$$

Remark 2.3.6. More consequences of Theorem 2.3.3 are that

(*i*) the canonical form $\mathrm{Can}\, w$ is the word of minimal length in $\pi^{-1}\pi(w)$;

(*ii*) due to Remark 2.3.4(*iii*), $\mathrm{Can}\, w$ is the last element in every simplifying sequence starting from any of the words in $\pi^{-1}\pi(w)$;

(*iii*) $\mathrm{Can}\, w$ is a quasi-subword of all the words in any simplifying sequence beginning from w;

(*iv*) if w contains any given letter, so does $\mathrm{Can}\, w$.

Lemma 2.3.7. *For every $u, v \in \mathrm{F}(A)$,*

$$\begin{aligned} \mathrm{Can}(uv) &= \mathrm{Can}\left((\mathrm{Can}\, u) v\right) \\ &= \mathrm{Can}\left(u \,\mathrm{Can}\, v\right) \\ &= \mathrm{Can}\left(\mathrm{Can}\, u \,\mathrm{Can}\, v\right) \end{aligned}$$

Proof. π is a homomorphism, and $\pi(x) = \pi(\operatorname{Can} x)$. □

Definition 2.3.8. If $I \subseteq A$, then

$$\operatorname{Can}_I w = \operatorname{Can} \partial_{A \setminus I} w,$$

where $w \in \mathrm{F}(A)$. In particular $\operatorname{Can}_A w = \operatorname{Can} w$.

Before proceeding further, we need to make an important observation. Say we have a sequence of steps leading from a word y to a word x, and that each step removes a single letter. The same procedure can be applied to every subword of y, and again at each step (at most) a single letter is removed, eventually yielding a subword of x. Every subword of x is obtained in this way from some (possibly non-unique) subword of y. Notice that in the cases we will deal with, some of the steps will be simplifications of type $\xrightarrow{1}$ for which there is an ambiguity on which of the two identical letter is to be removed.

We will say that a subword $a_i u a_i$ of x *originates* from the subword w of y if w is maximal among all subwords of y yielding $a_i u a_i$ under the sequence of simplifications that are of the form $a_i v a_i$. Let us clarify things with an example. In the following sequence of steps, we have highlighted the letter to remove at each step:

$$bdbcd\hat{a}bcdc \to b\hat{d}bcabcdc \to b\hat{b}cabcdc \to bcabcdc.$$

Notice that, in the last step, there is an ambiguity on which letter is being removed. Then the subwords *bdbcdab*, *dbcdab*, *bcdab* of *bdbcdabcdc* all yield *bcab*; however, *bcab* originates only from *bdbcdab* and *bcdab*.

Henceforth, we will shorten the notation and denote by $\partial_i, \mathrm{T}_i, \ldots$ the maps $\partial_{a_i}, \mathrm{T}_{a_i}, \ldots$.

Lemma 2.3.9. *Assume $y = \operatorname{Can} y$ and let $x = \partial_{[1,k-1]} y$, where $1 \le k \le n$. Then any subword $a_i u a_i \le x$ is either special or satisfies $u \in \langle a_i, \ldots, a_n \rangle$.*

Proof. Let $a_i u a_i$ be a subword of x. Then $i \ge k$, as $x = \partial_{[1,k-1]} y$ contains no $a_i, i < k$.

The above subword $a_i u a_i$ originates from a subword $a_i v a_i$ of y. If $a_i u a_i$ is not special, then either $u \in \langle a_j, j \le i \rangle$, or $u \in \langle a_j, j \ge i \rangle$. However, the former case does not occur, otherwise $a_i v a_i$ would fail to be special, as x is obtained from y by only removing letters $a_j, 1 \le j < k$, and $k \le i$. This is a contradiction, as y is canonical. □

Lemma 2.3.10. *Assume $y = \operatorname{Can} y$ and let $x = \partial_{[1,k-1]} y$, where $1 \le k \le n$. Then any simplifying sequence*

$$x = \partial_{[1,k-1]} y \to \cdots \to \operatorname{Can} x = \operatorname{Can}_{[k,n]} y$$

is such that all simplifying steps are of type $\xrightarrow{2}$ (possibly of type $\xrightarrow{1}$).

Proof. By Lemma 2.3.9, any subword $a_i u a_i \le x$ is either special or satisfies $u \in \langle a_j, i \le j \le n \rangle$, hence the first simplifying step is of type $\xrightarrow{2}$ (and possibly of type $\xrightarrow{1}$).

We want to show that, starting from $x = \partial_{[1,k-1]} y$, no simplifications of type $\xrightarrow{2}$ or of type $\xrightarrow{1}$ can lead to a word admitting a subword $a_i u a_i$, on which we may then apply a step of type $\xrightarrow{3}$ (and not of type $\xrightarrow{1}$). Notice that in such a case, $\star \neq u \in \langle a_j, k \leq j < i \rangle$, so that $i > k$.

Assume therefore by contradiction that $a_i u a_i$, where $\star \neq u \in \langle a_j, k \leq j < i \rangle$, occurs as a subword after having performed some simplifying steps of type $\xrightarrow{2}$ on x. The subword $a_i u a_i$ originates from a subword $a_i v a_i$ of x, which is necessarily special, since v cannot lie in $\langle a_j, i \leq j \leq n \rangle$, as it must yields $u \neq \star$ after removing some letters. This shows that letters $a_j, j > i$, must occur in v, whereas they do not occur in u, after only performing steps of type $\xrightarrow{2}$.

However, a simplifying step of type $\xrightarrow{2}$ that removes a letter from between the two occurrences of a_i either occurs completely between them, or begins of the left of the left a_i. In the former case, it does not change the speciality of the subword, whereas in the latter case the simplification cannot be of type $\xrightarrow{2}$, due to the presence of the left a_i. □

Corollary 2.3.11. *If $u a_1 v$ is canonical, then $\mathrm{Can}(uv)$ admits u as a prefix.*

Proof. By Lemma 2.3.10, all simplifying sequences from $uv = \partial_1(u a_1 v)$ to $\mathrm{Can}(uv)$ only contain steps of type $\xrightarrow{2}$, and no such simplifying step does alter u. □

Proposition 2.3.12. *If $u a_1 v$ is canonical, then $u a_1 \mathrm{Can}_{[k,n]} v$ is also canonical.*

Proof. We know that v is canonical. Then Lemma 2.3.10 shows that $u a_1 \partial_{[1,k-1]} v$ can be simplified into $u a_1 \mathrm{Can}_{[k,n]} v$ by only using steps of type $\xrightarrow{2}$ on the right of a_1.

If $u a_1 \mathrm{Can}_{[k,n]} v$ is not canonical, then we may find a subword $a_i x a_i$ that is not special. As both u and $\mathrm{Can}_{[k,n]} v$ are canonical, then a_1 must occur in x. Say that $a_i x a_i$ originates from the subword $a_i y a_i$ of $u a_1 v$. No simplification of type $\xrightarrow{2}$, when performed on the right of a_1, can change the set of letters that appear between the two a_i. This yields a contradiction, as $a_i y a_i$ is special, whereas $a_i x a_i$ is not. □

Corollary 2.3.13. *For every choice of $u, u', v, v' \in \langle a_3, a_4, \ldots, a_n \rangle$,*

(*i*) *if $u a_1 v a_2 v'$ is canonical, then $u a_1 \mathrm{Can}(vv')$ is canonical;*

(*ii*) *if $u a_2 v a_1 v'$ is canonical, then $u a_2 \mathrm{Can}(vv')$ is canonical;*

(*iii*) *if $u a_2 u' a_1 v a_2 v'$ is canonical, then both $u a_2 u' a_1 \mathrm{Can}(vv')$ and $u a_2 \mathrm{Can}(u' v v')$ are canonical.*

Proof. (*i*) follows directly from Proposition 2.3.12, and (*iii*) follows by applying Proposition 2.3.12 and then (*ii*).

However, (*ii*) is equivalent to (*i*), as both $u a_1 v a_2 v'$ and $u a_2 v a_1 v'$ contain single occurrences of a_1 and a_2, and every simplifying sequence for the former can be turned into a simplifying sequence for the latter by switching a_1 with a_2. □

Lemma 2.3.14. *Let $w \in \mathrm{F}(A)$, $i \in [1, n]$. Then*

(*i*) *there is at most one occurrence of a_i in $\mathrm{Can}_{[i,n]} w$;*

(*ii*) *if such an occurrence exists, then*

$$\mathrm{Can}_{[i,n]} \mathrm{T}_i\, w = a_i \,\mathrm{Can}_{[i+1,n]} \mathrm{T}_i\, w.$$

Proof. Without loss of generality, we may assume that $i = 1$.

(*i*) Recall Remark 2.3.4(*i*).

(*ii*) As $\mathrm{Can}\,\mathrm{T}_1\, w$ contains exactly one occurrence of a_1, this must be its head as, by Lemma 2.3.10, every simplifying step removing an a_i must necessarily be of type $\xrightarrow{2}$ or $\xrightarrow{1}$. Recalling Remark 2.3.5(*ii*), and the definition of Can_I,

$$\begin{aligned} \mathrm{Can}\,\mathrm{T}_1\, w &= \mathrm{Can}(a_1 \partial_1 \mathrm{T}_1\, w) \\ &= a_1 \,\mathrm{Can}(\partial_1 \mathrm{T}_1\, w) \\ &= a_1 \,\mathrm{Can}_{[2,n]} \mathrm{T}_1\, w. \end{aligned}$$

□

Lemma 2.3.15. *Let* $w \in \mathrm{F}(A)$, $i \in [1, n]$. *Then*

$$\mathrm{T}_i \,\mathrm{Can}_{[i,n]}\, w = \mathrm{Can}_{[i,n]} \mathrm{T}_i\, w.$$

Proof. Once again, we may assume without loss of generality that $i = 1$. If there are no occurrences of a_1 in w, then both sides equal the empty word, and we are done.

Otherwise, using Remark 2.2.4(*ii*), write $w = u\, \mathrm{T}_1\, w$, and $\mathrm{Can}\,\mathrm{T}_1\, w = a_1 v$. We are asked to show that

$$\mathrm{T}_1 \,\mathrm{Can}(u\, \mathrm{T}_1\, w) = \mathrm{Can}(\mathrm{T}_1\, w),$$

which is equivalent, using Lemma 2.3.7, to

$$\mathrm{T}_1 \,\mathrm{Can}(u a_1 v) = a_1 v.$$

Notice that v is canonical, and $u \in \langle a_2, \ldots, a_n \rangle$. The simplifying steps that can occur on a word of the type $u a_1 v$ may only affect u: indeed, v is canonical, there is an only occurrence of a_1 in $u a_1 v$, and the only special words that begin in u and end in v contain an occurrence of a_1, and thus lead to a $\xrightarrow{3}$.

An easy induction now shows that $\mathrm{Can}(u a_1 v) = u' a_1 v$, where u' is a quasi-subword of u, hence $\mathrm{T}_1 \,\mathrm{Can}(u a_1 v) = a_1 v$. □

Lemma 2.3.16. *For every* $u, v \in \langle a_2, \ldots, a_n \rangle$,

$$\mathrm{Can}(uv) \leq \mathrm{Can}(u a_1 v)$$

Proof. We may assume using Lemma 2.3.7 that v is canonical.

Argue as in the previous proof, to show that $\mathrm{Can}(u a_1 v) = u' a_1 v$, where u' is obtained by a sequence of simplifying steps that either take place on the left of a_1, or are of type $\xrightarrow{3}$. Every such step can be also performed starting with the word uv, as the presence or absence of a_1 has no influence on steps of type $\xrightarrow{3}$ that begin on u and end on v. Thus, $u'v$ can be obtained from uv by a simplifying sequence, hence $\mathrm{Can}(uv) = \mathrm{Can}(u'v)$. Now,

$$\mathrm{Can}(uv) = \mathrm{Can}(u'v) \leq u'v \leq u'a_1v = \mathrm{Can}(ua_1v),$$

and we are done. □

An immediate consequence of the last lemma is that if w contains only one occurrence of a_1, then $\mathrm{Can}_{[2,n]}\, w \leq \mathrm{Can}\, w$. However, this fact holds without the single occurrence assumption.

Lemma 2.3.17. *Let* $w \in \mathrm{F}(A)$. *Then* $\mathrm{Can}_{[2,n]}\, w$ *is a quasi-subword of* $\mathrm{Can}\, w$. *As a consequence,* $\mathrm{Can}_{[k,n]}\, w$ *is a quasi-subword of* $\mathrm{Can}\, w$ *for every* $k \leq n$.

Proof. If $w \in \langle a_2, \dots, a_n \rangle$, then $w = \partial_1 w$, and there is nothing to prove.

Otherwise, using Remark 2.2.4(*ii*), write $w = w'\, \mathrm{T}_1\, w$. According to Lemma 2.3.7 and Lemma 2.3.14(*ii*),

$$\begin{aligned} \mathrm{Can}\, w &= \mathrm{Can}\left(w'\ \mathrm{T}_1\, w\right) \\ &= \mathrm{Can}\left(w'\ \mathrm{Can}\, \mathrm{T}_1\, w\right) \\ &= \mathrm{Can}\left(w'\, a_1\, \mathrm{Can}_{[2,n]}\, \mathrm{T}_1\, w\right). \end{aligned}$$

On the other hand, by Remark 2.2.4(*ii*) and Lemma 2.3.7 (and recalling that here $\partial_1 w' = w'$),

$$\begin{aligned} \mathrm{Can}_{[2,n]}\, w &= \mathrm{Can}\, \partial_1 (w'\, \mathrm{T}_1\, w) \\ &= \mathrm{Can}\left(w'\, \partial_1\, \mathrm{T}_1\, w\right) \\ &= \mathrm{Can}\left(w'\ \mathrm{Can}_{[2,n]}\, \mathrm{T}_1\, w\right). \end{aligned}$$

We can now apply Lemma 2.3.16. The latter statement is taken care of by an easy induction. □

2.4 The join operation

Given an update system over the complete graph Γ_n, Proposition 2.1.4 proves that its dynamics monoid is an epimorphic image of Kiselman's semigroup K_n. In next section, we will exhibit an update system $\mathcal{S}_n^\star$ over Γ_n whose dynamics monoid is isomorphic to K_n; from a dynamical point of view, $\mathcal{S}_n^\star$ serves as a *universal update system*.

We introduce the following operation in order to construct, later, a family of update functions.

Definition 2.4.1. Take $u, v \in \mathrm{F}(A)$. The *join* of u and v, denoted by $[u, v]$, is the shortest word admitting u as quasi-subword and v as suffix. Namely,

$$[u, v] = u^+ v$$

where $u = u^+ u^-$, so that u^- is the longest suffix of u which is a quasi-subword of v.

Notice that the decomposition $u = u^+u^-$ strictly depends on the choice of v.

Example 2.4.2. For instance, consider $u = cbadc$ and $v = abdc$. Then,

$$[u, v] = cbabdc.$$

Remark 2.4.3.

(*i*) The empty word $\star$ is a subword of every $w \in \mathrm{F}(A)$, so that

$$[\star, u] = [u, \star] = u.$$

(*ii*) If u is a quasi-subword of v, then $[u, v] = v$.

(*iii*) If u is not a quasi-subword of v, then

$$[wu, v] = wu^+v = w[u, v],$$

for every $w \in \mathrm{F}(A)$.

(*iv*) If $[u, v] = u^+v$, then

$$[u, wv] = [u^+, w]v$$

Indeed, write $u = u^+u^-$: u^- is the longest suffix of u which is a quasi-subword of v, but there could be a suffix of u^+ which is a quasi-subword of w.

(*v*) If $u, v \in \mathrm{F}(A)$ are canonical, $[u, v]$ may fail to be so. For instance, $[a_1a_2, a_2a_1] = a_1a_2a_1$, which is not canonical.

The following lemma will be used later in the proof of Proposition 2.6.1.

Lemma 2.4.4. *Let $u, x, y \in \mathrm{F}(A)$, if $ux \leq uy$, then $x \leq y$.*

Proof. As ux is a quasi-subword of uy, by Remark 2.4.3(*ii*),

$$[ux, uy] = uy.$$

Assume that x is not a quasi-subword of y and write it as $x = x^+x^-$, where x^- is its longest suffix which is a quasi-subword of y. Using Remark 2.4.3(*iii*),

$$[ux, y] = ux^+y.$$

Finally, using Remark 2.4.3(*iv*),

$$[ux, uy] = [ux^+, u]y$$

However, $[ux^+, u] = u$ can hold only if ux^+ is not longer than u, i.e., only if $x^+ = \star$, hence x is a quasi-subword of y. □

2.5 An update system with universal dynamics

If $U, V \subset \mathrm{F}(A)$, denote by $[U, V] \subset \mathrm{F}(A)$ the subset of all elements $[u, v], u \in U, v \in V$.

Definition 2.5.1. The update system $\mathcal{S}_n^\star$ is the triple (Γ_n, S_i, f_i), where

1. Γ_n is, as before, the complete graph on n vertices, where $i \to j$ if and only if $i < j$.
2. On vertex i, the state set is

$$S_i = \begin{cases} \{\star, a_n\} & \text{if } i = n \\ \{\star, a_{n-1}, a_{n-1}a_n\} & \text{if } i = n-1 \\ \{\star\} \cup a_i[S_n, [\dots, [S_{i+2}, S_{i+1}] \dots]] & \text{if } 1 \le i \le n-2 \end{cases}$$

3. On vertex i, the vertex function is

$$\begin{aligned} f_i \colon \prod_{j=i+1}^{n} S_j &\to S_i \\ (s_{i+1}, \dots, s_n) &\mapsto a_i[s_n, [\dots, [s_{i+2}, s_{i+1}] \dots]] \end{aligned}$$

   if $i \le n-2$, whereas $f_{n-1}(s_n) = a_{n-1}a_n$ and $f_n \equiv a_n$ is a constant.

We will abuse the notation and denote by $\star$ also the system state $(\star, \dots, \star) \in S$.

2.6 Statement of the main result and some technical lemmas

This and next sections will be devoted to the proof of the following

Proposition 2.6.1. *Consider the evaluation morphism $\mathbf{F} \colon \mathrm{F}(A) \to D(\mathcal{S}_n^\star)$, mapping each word $w \in \mathrm{F}(A)$ to the corresponding evolution $\mathbf{F}_w \in D(\mathcal{S}_n^\star)$. If $\mathbf{p} = (p_1, \dots, p_n) = \mathbf{F}_w \star$, then:*

(i) One has $p_i = (\mathbf{F}_w \star)_i = \mathrm{Can}_{[i,n]} \mathrm{T}_i w$.

(ii) For every choice of $k, 1 \le k \le n$, one may find $j \in [1, k]$, so that

$$[p_k, [\dots, [p_2, p_1] \dots]] = \mathrm{T}_j \,\mathrm{Can}\, w,$$

and $\mathrm{T}_i \,\mathrm{Can}\, w$ is a suffix of $\mathrm{T}_j \,\mathrm{Can}\, w$ for all $i \le k$.

Our central result then follows immediately.

Theorem 2.6.2. *With the same hypotheses and notation as in Proposition 2.6.1,*

(i) $\mathrm{Can}\, w = [p_n, [\dots, [p_2, p_1] \dots]]$;

(ii) for every $u, v \in \mathrm{F}(A)$, $u \sim v$ if and only if $\mathrm{Can}\, u = \mathrm{Can}\, v$;

(*iii*) K_n *is isomorphic to* $D(\mathcal{S}_n^\star)$.

Proof.

(*i*) Use Part (*ii*) of Proposition 2.6.1 for $k = n$. Then $[p_n, [\dots, [p_2, p_1]\dots]] = \mathrm{T}_j \operatorname{Can} w$ for a suitably chosen j. However, $\mathrm{T}_i \operatorname{Can} w$ is a suffix of $\mathrm{T}_j \operatorname{Can} w$ for all i, hence, by Remark 2.2.5(*ii*), $\mathrm{T}_j \operatorname{Can} w = \operatorname{Can} w$.

(*ii*) If $u \sim v$, then they compute the same state on $\star$. However, by (*i*), one may recover both $\operatorname{Can} u$ and $\operatorname{Can} v$ from this state, hence $\operatorname{Can} u = \operatorname{Can} v$. The other implication follows trivially, as $\operatorname{Can} u = \operatorname{Can} v$ forces u and v to induce the same element in K_n, hence the same dynamics on $\mathcal{S}_n^\star$.

(*iii*) This is just a restatement of (*ii*): distinct elements in K_n have distinct $\mathcal{S}_n^\star$-actions, since they act in a different way on the system state $\star$.

□

Remark 2.6.3. In order to prove Theorem 2.6.2, we have located $j \in [1, n]$ such that

$$\mathrm{T}_j \operatorname{Can} w = \operatorname{Can} w,$$

whence $\mathrm{h}(\operatorname{Can} w) = a_j$. Now, there is no occurrence of a_j in $p_{j+1}, \dots, p_n$, so that one necessarily have

$$[p_j, [\dots, [p_2, p_1]\dots]] = [p_n, [\dots, [p_2, p_1]\dots]] = \operatorname{Can} w.$$

We will prove Proposition 2.6.1 by induction on the number n of vertices. We will divide the proof in several steps, for all of which we will assume by inductive hypothesis that Proposition 2.6.1, as well as Theorem 2.6.2, hold on $\mathcal{S}_k^\star$, for $k < n$.

Lemma 2.6.4. *For every* k, $1 \le k \le n$, $w \in \mathrm{F}(A)$, *one has*

$$\operatorname{Can}_{[k,n]} \operatorname{Can} w = \operatorname{Can}_{[k,n]} w.$$

Proof. The case $k = 1$ is trivial.

Say $k > 1$. We may use the inductive assumption to argue that the action of two words $u, v \in \mathrm{F}(A)$ on $\star$ coincide on the subgraph formed by vertices indexed by $[k, n]$, if and only if $\operatorname{Can}_{[k,n]} u = \operatorname{Can}_{[k,n]} v$. However, as $u = \operatorname{Can} w$ and $v = w$ have the same action on the whole graph, they certainly coincide on the subgraph $[k, n]$. □

The last technical statement, before proceeding with the proof of Proposition 2.6.1, is the following.

Proposition 2.6.5. *Let* $u, v \in \langle a_{j+1}, a_{j+2}, \dots, a_n\rangle$ *be chosen so that* $u a_j \operatorname{Can} v$ *is a canonical word. Then* $[\operatorname{Can}(uv), v] = uv$. *In particular,* $[\operatorname{Can}(uv), a_j v] = u a_j v$.

Proof. We may assume, without loss of generality, that $j = 1$.

The statement is easily checked case by case when $n = 1$ or 2. Notice that u can have at most one occurrence of a_2, whereas v may have many. Let us therefore distinguish four cases:

1. There is no occurrence of a_2 in either u or v.

   In this case, we can use inductive assumption, after removing the vertex indexed by 2.

2. There is a single occurrence of a_2 in u.

   Write $u = u'a_2u''$. As $u'a_2u''a_1 \operatorname{Can} v$ is canonical, then, using Corollary 2.3.13(*ii*) and Lemma 2.3.7,

$$\begin{aligned} \operatorname{Can}(u'a_2u''v) &= \operatorname{Can}(u'a_2u'' \operatorname{Can} v) \\ &= u'a_2 \operatorname{Can}(u'' \operatorname{Can} v) \\ &= u'a_2 \operatorname{Can}(u''v). \end{aligned}$$

   Thus, we need to compute $[u'a_2 \operatorname{Can}(u''v), v]$. However, applying Case 1 gives

$$[\operatorname{Can}(u''v), v] = u''v,$$

   hence $[u'a_2 \operatorname{Can}(u''v), v] = u'a_2u''v$ follows from Remark 2.4.3(*iii*).

3. a_2 occurs in v, but not in u.

   Write $\operatorname{Can} v = v'a_2v''$. As $ua_1 \operatorname{Can} v = ua_1v'a_2v''$ is canonical, then, using Corollary 2.3.13(*i*), also $ua_1 \operatorname{Can}(v'v'')$ is canonical. However Lemma 2.6.4 informs us that

$$\begin{aligned} \operatorname{Can}(v'v'') &= \operatorname{Can} \partial_2 \operatorname{Can} v \\ &= \operatorname{Can}_{[3,n]} \operatorname{Can} v \\ &= \operatorname{Can}_{[3,n]} v \\ &= \operatorname{Can} \partial_2 v, \end{aligned}$$

   so that $ua_1 \operatorname{Can} \partial_2 v$ is canonical. We may then use Case 1 to argue that

$$[\operatorname{Can}(u\partial_2 v), \partial_2 v] = u\partial_2 v \tag{2.6.1}$$

   Lemma 2.3.17 implies that $\operatorname{Can}(u\partial_2 v) = \operatorname{Can} \partial_2(uv)$ is a quasi-subword of $\operatorname{Can}(uv)$. By Corollary 2.3.11, both $\operatorname{Can}(uv)$ and $\operatorname{Can}(u\partial_2 v)$ admit u as a prefix, so, using Lemma 2.4.4, we can write

$$\operatorname{Can}(uv) = uy, \qquad \operatorname{Can}(u\partial_2 v) = uz, \qquad z \leq y.$$

   We need to show that $[\operatorname{Can}(uv), v] = uv$. Now, applying Corollary 2.3.13 (*ii*),

$$\operatorname{Can}(uv) = \operatorname{Can}(uv')a_2v'' = uxa_2v'',$$

where x is some quasi-subword of v'. As

$$y = xa_2v'' \leq v'a_2v'' = \operatorname{Can} v \leq v,$$

then $[\operatorname{Can}(uv), v] = u^+v$ where $u = u^+u^-$ and $u^-y \leq v$. However, this would force

$$u^-z \leq u^-y \leq v$$

hence also $u^-z \leq \partial_2 v$; and this is only possible if $u^- = \star$, as Equation (2.6.1) shows.

4. a_2 occurs both in u and in v.

   This is similar to the previous case. Write $u = u'a_2u''$, $\operatorname{Can} v = v'a_2v''$. Applying Corollary 2.3.13(*iii*) to the canonical word $ua_1 \operatorname{Can} v = u'a_2u''a_1v'a_2v''$, we obtain that both

   $$u'a_2u''a_1 \operatorname{Can} \partial_2 v = u'a_2u''a_1 \operatorname{Can}(v'v'')$$

   and

   $$u'a_2 \operatorname{Can}(u''\partial_2 v) = u'a_2 \operatorname{Can}(u'' \operatorname{Can} \partial_2 v) = u'a_2 \operatorname{Can}(u''v'v'')$$

   are canonical. Similarly,

   $$\begin{aligned} \operatorname{Can}(uv) &= \operatorname{Can}(u'a_2u''v'a_2v'') \\ &= \operatorname{Can}(u'a_2u''v'v'') \\ &= \operatorname{Can}(u'a_2 \operatorname{Can}(u''v'v'')) \\ &= u'a_2 \operatorname{Can}(u''\partial_2 v). \end{aligned}$$

   Now, as $u''a_1 \operatorname{Can} \partial_2 v$, being a subword of $u'a_2u''a_1 \operatorname{Can} \partial_2 v$, is canonical, we use Case 1 and obtain

   $$[\operatorname{Can}(u''\partial_2 v), \partial_2 v] = u''\partial_2 v. \tag{2.6.2}$$

   As a consequence,

   $$\begin{aligned} [\operatorname{Can}(u'a_2u''v'a_2v''), \partial_2 v] &= [u'a_2 \operatorname{Can}(u''\partial_2 v), \partial_2 v] \\ &= [u'a_2, u'']\partial_2 v \\ &= u'a_2u''\partial_2 v, \end{aligned}$$

   and one may complete the proof as in Case 3.

□

2.7 Proof of Proposition 2.6.1

The basis of induction $n = 1$ being trivial, we assume that Proposition 2.6.1, hence Theorem 2.6.2, hold for a complete graph on less than n vertices.

Proof of Proposition 2.6.1. Let us start by proving Part (i).

Let $w \in \mathrm{F}(A)$. By inductive hypothesis, for a complete graph on $n-1$ vertices $2, \dots, n$, the SDS map $\mathbf{F}_{\partial_1 w}$ constructs on the ith vertex the state

$$p_i = \mathrm{Can}_{[i,n]}\, \mathrm{T}_i\, \partial_1 w.$$

When $i > 1$, we have

$$\partial_1 w \sim_i w$$

and we may use induction to argue that $p_i = \mathrm{Can}_{[i,n]}\, \mathrm{T}_i\, \partial_1 w$. Now, Lemma 2.2.8 allows us to exchange ∂_1 and T_i, so that

$$\begin{array}{rcl} p_i & = & \mathrm{Can}_{[i,n]}\, \mathrm{T}_i\, \partial_1 w \\ & = & \mathrm{Can}_{[i,n]}\, \partial_1\, \mathrm{T}_i\, w \\ & = & \mathrm{Can}\, \partial_{[1,i-1]} \partial_1\, \mathrm{T}_i\, w \\ & = & \mathrm{Can}_{[i,n]}\, \mathrm{T}_i\, w. \end{array}$$

Hence, the statement holds for all vertices $i > 1$.

As far as vertex 1 is concerned, we need to show that $p_1 = \mathrm{Can}\, \mathrm{T}_1\, w$. If w does not contain the letter a_1, then $p_1 = \star = \mathrm{T}_1\, w$, and there is nothing to prove; otherwise, $w \sim_1 \mathrm{T}_1\, w$. We know that the state on vertex 1 depends only on the system state $(p'_2, \dots, p'_n)$, which is computed by $\partial_1\, \mathrm{T}_1\, w$ on the subgraph indexed by $[2, n]$, i.e.,

$$p_1 \quad = \quad a_1 \left[p'_n, [\dots, [p'_3, p'_2] \dots]\right].$$

However, we may apply induction hypothesis, and use Theorem 2.6.2(i), which yields

$$\begin{array}{rcl} p_1 & = & a_1 \left[p'_n, [\dots, [p'_3, p'_2] \dots]\right] \\ & = & a_1\, \mathrm{Can}\, \partial_1\, \mathrm{T}_1\, w \\ & = & \mathrm{Can}\, \mathrm{T}_1\, w \end{array}$$

As for Part (ii) of Proposition 2.6.1, we proceed by induction on k. The basis of induction descends directly from Part (i), and Lemma 2.3.15, as

$$\begin{array}{rcl} p_1 & = & \mathrm{Can}\, \mathrm{T}_1\, w \\ & = & \mathrm{T}_1\, \mathrm{Can}\, w. \end{array}$$

Assume now $k > 1$. By inductive hypothesis, there exists $j < k$ such that

$$[p_{k-1}, [\dots, [p_2, p_1] \dots]] = \mathrm{T}_j\, \mathrm{Can}\, w.$$

Recall that, by Remark 2.2.5(i), either $\mathrm{T}_k\, \mathrm{Can}\, w$ is a suffix of $\mathrm{T}_j\, \mathrm{Can}\, w$ or vice versa. In the former case, we know by Part (i), and Lemma 2.3.15, that

$$p_k = \mathrm{Can}_{[k,n]}\, \mathrm{T}_k\, w = \mathrm{T}_k\, \mathrm{Can}_{[k,n]}\, w. \tag{2.7.1}$$

By Lemma 2.3.17, $\mathrm{Can}_{[k,n]} w$ is a quasi-subword of $\mathrm{Can}\, w$, hence $p_k = \mathrm{T}_k \mathrm{Can}_{[k,n]} w$ is a quasi-subword of $\mathrm{T}_k \mathrm{Can}\, w$, which is a suffix of $\mathrm{T}_j \mathrm{Can}\, w$. Thus,

$$\begin{aligned}[p_k, [p_{k-1} [\dots, [p_2, p_1] \dots]]] &= [p_k, \mathrm{T}_j \mathrm{Can}\, w] \\ &= \mathrm{T}_j \mathrm{Can}\, w.\end{aligned}$$

If, instead, $\mathrm{T}_j \mathrm{Can}\, w$ is a suffix of $\mathrm{T}_k \mathrm{Can}\, w$, we argue as follows. Choose u such that $\mathrm{T}_k \mathrm{Can}\, w = u \mathrm{T}_j \mathrm{Can}\, w$. As $u \in \langle a_i, k \le i \le n \rangle$, then by Part (i)

$$p_k = \mathrm{Can}_{[k,n]}(u \mathrm{T}_j \mathrm{Can}\, w) = \mathrm{Can}(u \partial_{[1,k-1]} \mathrm{T}_j \mathrm{Can}\, w).$$

By Proposition 2.3.12, as the word $\mathrm{T}_k \mathrm{Can}\, w = u \mathrm{T}_j \mathrm{Can}\, w$ is canonical, we have that $u a_j \mathrm{Can}_{[k,n]} \mathrm{T}_j \mathrm{Can}\, w$ is a canonical word. Applying now Proposition 2.6.5 to the word $u a_j \partial_{[1,k-1]} \mathrm{T}_j \mathrm{Can}\, w$, one obtains

$$\begin{aligned}[p_k, a_j \partial_{[1,k-1]} \mathrm{T}_j \mathrm{Can}\, w] &= [\mathrm{Can}(u \partial_{[1,k-1]} \mathrm{T}_j \mathrm{Can}\, w), a_j \partial_{[1,k-1]} \mathrm{T}_j \mathrm{Can}\, w] = \\ &= u a_j \partial_{[1,k-1]} \mathrm{T}_j \mathrm{Can}\, w,\end{aligned}$$

hence a fortiori $[p_k, \mathrm{T}_j \mathrm{Can}\, w] = u \mathrm{T}_j \mathrm{Can}\, w$.

□

2.8 Some more results on the dynamics of the SDS on the complete graph

In order to study the dynamics monoid $D(\mathcal{S}_n^\star)$, it is useful to order its elements according to their action. Let $\star = (\star, \dots, \star) \in S$ be the "initial unaffected system state".

Definition 2.8.1. Let $x, y \in \mathrm{F}(A)$. If $(\mathbf{F}_x \star)_i = (\mathbf{F}_y \star)_i$, in $\mathcal{S}_n^\star$, then we say that x and y *compute the same state on vertex* i, and write

$$x \sim_i y.$$

If $x \sim_i y$ for every $i = 1, \dots, n$, then we say that x and y *compute the same system state*, and write

$$x \sim y.$$

Remark 2.8.2. Given three words $x, y, z \in \mathrm{F}(A)$, if $y \sim z$, then $xy \sim xz$. Indeed, $\mathbf{F}_{xy} \star = \mathbf{F}_x(\mathbf{F}_y \star) = \mathbf{F}_x(\mathbf{F}_z \star) = \mathbf{F}_{xz} \star$.

We will define the following partial order keeping in mind Green relations, as stated in [Gre51].

Definition 2.8.3. For every $x, y \in \mathrm{F}(A)$, we say that x *comes before* y, and we write $x \preceq y$, if and only if there exists $w \in \mathrm{F}(A)$ such that

$$wx \sim y,$$

i.e., such that

$$\mathbf{F}_{wx} \star = \mathbf{F}_y \star.$$

We will also say that y *comes after* x, and write $y \succeq x$.

Remark 2.8.4. Let us recall the left Green relation $\leq_{\mathcal{L}}$ on the dynamics monoid $D(\mathcal{S}_n^\star)$:

$$\mathbf{F}_y \leq_{\mathcal{L}} \mathbf{F}_x \text{ if and only if there exists } \mathbf{F}_u \in D(\mathcal{S}_n^\star) \text{ such that } \mathbf{F}_y = \mathbf{F}_u \, \mathbf{F}_x \,.$$

The relation $\succeq$ is not checked on the whole system state set, but only on the element $\star$. One of the immediate consequences of Theorem 2.6.2 is that two words induce the same action on $\mathcal{S}_n^\star$ if and only if they have the same action on $\star$, so that the relation of "coming before", defined on $\mathrm{F}(A)$, projects to the Green relation $\leq_{\mathcal{L}}$ through the monoid morphism $\mathrm{F}(A) \to D(\mathcal{S}_n^\star)$.

Example 2.8.5. Consider the update system $\mathcal{S}_2^\star$ on two vertices, with initial state $\star = (\star, \star)$. We have that $a_1a_2 \succeq a_2a_1$, as

$$\mathbf{F}_{a_1a_2}(\star,\star) = (a_1a_2, a_2), \quad \mathbf{F}_{a_2a_1}(\star,\star) = (a_1, a_2)$$

and

$$\mathbf{F}_{a_1}(a_1, a_2) = (a_1a_2, a_2).$$

More precisely, the only adjacency relations are $a_1a_2 \succeq a_2a_1 \succeq a_1 \succeq \star$ and $a_1a_2 \succeq a_2 \succeq \star$, so that this relation coincides with $\leq_{\mathcal{L}}$, as mentioned before.

Remark 2.8.6. We will see in Proposition 2.8.12 that

$$x \preceq y \text{ and } y \preceq x \text{ if and only if } x \sim y.$$

This is a claim of irreversibility in the dynamics of the update system $\mathcal{S}_n^\star$.

Lemma 2.8.7. *Let $x, y \in \mathrm{F}(A)$. Then $\mathrm{T}_1\, x \sim_1 \mathrm{T}_1\, y$ if and only if $\mathrm{T}_1\, x \sim_i \mathrm{T}_1\, y$, for all $i \geq 2$.*

Proof. Suppose that $\mathrm{T}_1\, x \sim_1 \mathrm{T}_1\, y$. By inductive assumption, the quasi-subwords $\partial_1 \mathrm{T}_1\, x$ and $\partial_1 \mathrm{T}_1\, y$ compute the vertex states

$$\begin{aligned} p_i &= \mathrm{Can}_{[i,n]}\, \partial_1 \mathrm{T}_1\, x = \mathrm{Can}_{[i,n]}\, \mathrm{T}_1\, x \\ q_i &= \mathrm{Can}_{[i,n]}\, \partial_1 \mathrm{T}_1\, y = \mathrm{Can}_{[i,n]}\, \mathrm{T}_1\, y \end{aligned}$$

where we used the fact that $\mathrm{Can}_{[i,n]}\, \partial_1 = \mathrm{Can}\, \partial_{[1,i-1]} \partial_1 = \mathrm{Can}_{[i,n]}$.

The definition of update function on vertex 1 states that

$$\begin{aligned} (\mathbf{F}_{\mathrm{T}_1 x} \star)_1 &= a_1[p_n, [\ldots, [p_3, p_2] \ldots]] \\ (\mathbf{F}_{\mathrm{T}_1 y} \star)_1 &= a_1[q_n, [\ldots, [q_3, q_2] \ldots]] \end{aligned}$$

Since $\mathrm{T}_1\, x \sim_1 \mathrm{T}_1\, y$, we have $(\mathbf{F}_{\mathrm{T}_1 x} \star)_1 = (\mathbf{F}_{\mathrm{T}_1 y} \star)_1$, thus

$$a_1[p_n, [\ldots, [p_3, p_2] \ldots]] = a_1[q_n, [\ldots, [q_3, q_2] \ldots]],$$

so

$$[p_n, [\ldots, [p_3, p_2] \ldots]] = [q_n, [\ldots, [q_3, q_2] \ldots]].$$

Moreover, none of the p_i nor the q_i contains a_1: hence, it is possible to apply Proposition 2.6.1, which, by inductive hypothesis, holds on the subgraph indexed by $[k, n]$. Thus we have,

$$\begin{aligned}\mathrm{Can}_{[2,n]}\,\mathrm{T}_1\,x &= [p_n, [\ldots, [p_3, p_2]\ldots]]\\ &= [q_n, [\ldots, [q_3, q_2]\ldots]]\\ &= \mathrm{Can}_{[2,n]}\,\mathrm{T}_1\,y,\end{aligned}$$

so that $\mathrm{Can}_{[2,n]}\,\mathrm{T}_1\,x$ and $\mathrm{Can}_{[2,n]}\,\mathrm{T}_1\,y$ compute the same states $p_i = q_i$ on each vertex $i \geq 2$. This means that, in particular,

$$\mathrm{T}_1\,x \sim_i \mathrm{T}_1\,y, \text{ for all } i \geq 2.$$

Vice versa, assume that $\mathrm{T}_1\,x \sim_i \mathrm{T}_1\,y$, for all $i \geq 2$, i.e., that $p_i = q_i, i \geq 2$. Then, by the very definition, both $\mathrm{T}_1\,x$ and $\mathrm{T}_1\,y$ compute, on vertex 1, the state

$$p_1 = a_1[p_n, [\ldots, [p_3, p_2]\ldots]],$$

hence, $(\mathbf{F}_{\mathrm{T}_1\,x}\star)_1 = (\mathbf{F}_{\mathrm{T}_1\,y}\star)_1$, i.e. $\mathrm{T}_1\,x \sim_1 \mathrm{T}_1\,y$.

□

Corollary 2.8.8. $\mathrm{T}_1\,x \sim \mathrm{T}_1\,y$ *if and only if* $\mathrm{T}_1\,x \sim_1 \mathrm{T}_1\,y$ *if and only if* $\mathrm{T}_1\,x \sim_i \mathrm{T}_1\,y$, *for all* $i > 1$.

Lemma 2.8.9. *Let* $x, y \in \mathrm{F}(A)$ *such that* $x \preceq y$. *Then* $\mathrm{T}_1\,x \preceq \mathrm{T}_1\,y$.

Proof. Recall that, since $y \succeq x$, we can find $w \in \mathrm{F}(A)$ such that $y \sim wx$. In particular, $y \sim_1 wx$, hence

$$\mathrm{T}_1\,y \sim_1 \mathrm{T}_1(wx)$$

and, by Corollary 2.8.8, $\mathrm{T}_1\,y \sim \mathrm{T}_1(wx)$. If w contains an occurrence of a_1, then

$$\mathrm{T}_1(wx) = \mathrm{T}_1(w)x \succeq x \succeq \mathrm{T}_1\,x,$$

and the claim follows. Otherwise, $y \sim_1 wx \sim_1 x \sim_1 \mathrm{T}_1\,x$,. Then $\mathrm{T}_1\,y \sim_1 \mathrm{T}_1\,x$, and using Corollary 2.8.8, $\mathrm{T}_1\,y \sim \mathrm{T}_1\,x$, hence $\mathrm{T}_1\,y \succeq \mathrm{T}_1\,x$. □

Corollary 2.8.10. *Let* $x, y \in \mathrm{F}(A)$ *such that* $x \preceq y$. *Then,* $\partial_1\,\mathrm{T}_1\,x \preceq \partial_1\,\mathrm{T}_1\,y$.

Proof. Use Corollary 2.8.8. □

Corollary 2.8.11. *Let* $y \in \mathrm{F}(A)$ *a word containing* a_1, $x \in \langle a_2, \ldots, a_n\rangle$. *If* $x \preceq y$, *then* $x \preceq \mathrm{T}_1\,y$ *and* $x \preceq \partial_1\,\mathrm{T}_1\,y$.

Proof. Take $w \in \mathrm{F}(A)$ such that $y \sim wx$. Since a_1 is the updating of vertex 1 (and recall that there are no arrows pointing to vertex 1), then $y \sim wa_1x$, thus, $y \succeq a_1x$. Then Lemma 2.8.9 gives $\mathrm{T}_1\,y \succeq \mathrm{T}_1(a_1x) = a_1x \succeq x$, and likewise Corollary 2.8.10 gives $\partial_1\,\mathrm{T}_1\,y \succeq x$. □

The following statement shows that the evolution of $\mathcal{S}_n^\star$ is irreversible.

Proposition 2.8.12. *Let $x, y \in \mathrm{F}(A)$. Then $x \sim y$ if and only if $x \preceq y$ and $y \preceq x$.*

Proof. We proceed by induction on $n = |A|$. As for the induction basis, if $n = 1$, there is nothing to prove. Assume that $n > 1$. If $x \preceq y \preceq x$, then by Corollary 2.8.10, $\partial_1 \mathrm{T}_1 x \preceq \partial_1 \mathrm{T}_1 y \preceq \partial_1 \mathrm{T}_1 x$. By induction hypothesis, $\partial_1 \mathrm{T}_1 x \sim \partial_1 \mathrm{T}_1 y$, hence $x \sim_1 y$.

As for the other vertices, from $u \preceq v$ follows trivially $\partial_1 u \preceq \partial_1 v$. Then $\partial_1 x \preceq \partial_1 y \preceq \partial_1 x$, and the induction hypothesis shows once again that $x \sim_i \partial_1 x \sim \partial_1 y \sim_i y$ for all $i \geq 2$. Therefore, the action of x and y coincide on all vertices, and $x \sim y$. The converse implication is trivial. □

One of the ingredients in the proof of Proposition 2.6.1 is the fact that if x comes before y, then their join $[y, x]$ acts as y on the system state $\star$, under the assumption that y is in canonical form. The following slightly stronger statement turns out to be more suitable for a proof by induction.

Proposition 2.8.13. *Choose $x, y, z \in \mathrm{F}(A)$ such that $y = \mathrm{Can}\, y$, and $xz \preceq yz$. Then,*

$$[y, x]z \sim yz.$$

In particular, if y is canonical and $x \preceq y$, then $[y, x] \sim y$.

Proof. We proceed by induction on the cardinality $n = |A|$. If $n = 1$, the verification is trivial. Assume now that $n > 1$. We distinguish four cases:

1. Neither x nor y contain occurrences of a_1.

   If there is not any occurrence of a_1 in z, then we may restrict to the subgraph indexed by $[2, n]$ and use inductive assumption. If, instead, a_1 occurs in z, then $[y, x]z \sim_1 z \sim_1 yz$, whereas we may use inductive assumption on the other vertices, after replacing z by $\partial_1 z$.

2. a_1 occurs in y but not in x.

   As y is canonical, we may write $y = ua_1v$, where a_1 occurs neither in u nor in v. Notice that $[y, x] = ua_1[v, x]$, and that as $yz \succeq xz$, then $vz \succeq xz$, hence $v\partial_1 z \succeq x\partial_1 z$.

   Let us first show that $[y, x]z$ and yz have the same action on vertex 1. Indeed,

$$\begin{aligned} [y, x]z &= ua_1[v, x]z \\ &\sim_1 a_1[v, x]z \\ &\sim_1 a_1[v, x]\partial_1 z, \end{aligned}$$

   whereas

$$\begin{aligned} yz &= ua_1vz \\ &\sim_1 a_1vz \\ &\sim_1 a_1v\partial_1 z. \end{aligned}$$

In order to check $a_1[v,x]\partial_1 z \sim_1 a_1 v\partial_1 z$, by Remark 2.8.2 it is enough to verify that $[v,x]\partial_1 z \sim v\partial_1 z$, which holds by induction hypothesis.

As for the other vertices, we need to compare actions of $u[v,x]\partial_1 z$ and $uv\partial_1 z$, which coincide by induction.

3. a_1 occurs in both x and y.

Notice that, up to replacing z with $\partial_1 z$, we may assume without loss of generality that a_1 does not occur in z. As y is canonical, we may write $y = ua_1 v$, where a_1 occurs neither in u nor in v. We need to compare the actions of $yz = ua_1vz$ and $[y,x]z$. If $[v, \partial_1 \mathrm{T}_1 x] \neq \partial_1 \mathrm{T}_1 x$, then $[y,x]z = ua_1[v,x]z \sim ua_1[v, \partial_1 \mathrm{T}_1 x]z$. However, by Corollary 2.8.10, $yz \succeq xz$ implies

$$\begin{aligned} vz &= \partial_1 \mathrm{T}_1(yz) \\ &\succeq \partial_1 \mathrm{T}_1(xz) \\ &= \partial_1 \mathrm{T}_1(x)z, \end{aligned}$$

and we may use induction to argue that $[v, \partial_1 \mathrm{T}_1 x]z \sim vz$.

If instead $[v, \partial_1 \mathrm{T}_1 x] = \partial_1 \mathrm{T}_1 x$, write $u = u^+u^-$, so that u^- is the longest suffix of u satisfying $[u^-v, \partial_1 \mathrm{T}_1 x] = \partial_1 \mathrm{T}_1 x$. Then $[y,x]z = u^+a_1\partial_1 \mathrm{T}_1(x)z$, whereas one knows $yz = u^+u^-a_1vz$. As $yz \succeq xz$, then using Corollary 2.8.10 $\partial_1 \mathrm{T}_1(yz) \succeq \partial_1 \mathrm{T}_1(xz)$, that is, $vz \succeq \partial_1 \mathrm{T}_1(x)z$; then also

$$\begin{aligned} \mathrm{Can}(u^-v)z &\sim (u^-v)z \\ &\succeq vz \\ &\succeq \partial_1 \mathrm{T}_1(x)z. \end{aligned}$$

We know that $[u^-v, \partial_1 \mathrm{T}_1 x] = \partial_1 \mathrm{T}_1 x$, and as by Remark 2.3.6 $\mathrm{Can}(u^-v)$ is a quasi-subword of u^-v, also $[\mathrm{Can}(u^-v), \partial_1 \mathrm{T}_1 x] = \partial_1 \mathrm{T}_1 x$. As $\mathrm{Can}(u^-v)$ and v are both canonical, we may apply inductive assumption to show

$$\begin{aligned} u^-vz &\sim \mathrm{Can}(u^-v)z \\ &\sim [\mathrm{Can}(u^-v), \partial_1 \mathrm{T}_1 x]z \\ &= \partial_1 \mathrm{T}_1(x)z, \end{aligned}$$

and in the same way, $vz \sim \partial_1 \mathrm{T}_1(x)z$.

We conclude now that as $vz \sim u^-vz$, then $a_1vz \sim u^-a_1vz$, and necessarily

$$\begin{aligned} [y,x]z &= u^+a_1\partial_1 \mathrm{T}_1(x)z \\ &\sim u^+a_1vz \\ &\sim u^+u^-a_1vz \\ &= yz. \end{aligned}$$

4. a_1 occurs in x, but not in y.

   This cannot happen if a_1 does not occur in z, as $yz \succeq xz$ would force at least an occurrence of a_1 in y. Assume then that a_1 occurs in z. As $yz \succeq xz$, then by Corollary 2.8.10,

$$\begin{aligned}
\partial_1 \mathrm{T}_1 z &= \partial_1 \mathrm{T}_1(yz) \\
&\succeq \partial_1 \mathrm{T}_1(xz) \\
&= \partial_1(\mathrm{T}_1(x)z) \\
&\succeq \partial_1 z \\
&\succeq \partial_1 \mathrm{T}_1 z,
\end{aligned}$$

   which must, therefore, all possess the same action.

   As $\partial_1 \mathrm{T}_1(xz) \sim \partial_1 \mathrm{T}_1 z$, then $xz \sim_1 z$, so that $\partial_1(x)z \sim_1 z \sim_1 xz$. However $\partial_1(x)z \sim_i xz$ for all $i \geq 2$, so that $\partial_1(x)z \sim xz$. Also, if $[y,x] = y^+x$, then

$$\begin{aligned}
[y,x]z &= y^+xz \\
&\sim y^+\partial_1(x)z \\
&= [y, \partial_1 x]z.
\end{aligned}$$

   We may then replace x by $\partial_1 x$ in the statement to prove, thus falling in Case 1.

□

3 Update systems on non-complete graphs.

Most of real world applications of SDS occur on directed graphs which are acyclic but not complete. For example, a neural network can be usefully represented by such a graph, as in the brain neural cycles, even if possible, are "rare" and generally involve a very large number of elements. Hence, it has been a natural step that of trying to generalize our result on all such possible graphs.

If Γ is a finite directed acyclic graph, then any update system $\mathcal{S}_\Gamma$ supported on Γ induces a dynamics monoid $D(\mathcal{S}_\Gamma)$ which naturally arises as a quotient of $\mathbf{HK}_\Gamma$. We refer to the smallest quotient of $\mathbf{HK}_\Gamma$ through which all evaluation maps $F\colon \mathrm{F}(V) \to D(\mathcal{S}_\Gamma)$ factor as the universal dynamics monoid $D(\mathcal{S}_\Gamma)$. In the previous chapter, we showed an update system $\mathcal{S}_n^\star$ such that proved that $D(\mathcal{S}_n^\star) \cong \mathbf{HK}_n = \mathrm{K}_n$. Such update system was called *universal.*

The scope of our work has been to extend or (rather) modify our tools in order to construct a universal update system for a generic graph, in order to prove the following conjecture.

Conjecture 4. $D(\mathcal{S}_\Gamma) \cong \mathbf{HK}_\Gamma$ for every finite directed acyclic graph.

We have been able to check it computationally on all graphs up to 5 vertices. Moreover, we came out with some results on stars graph and line graph, whose undirected versions have been studied in SDS theory (see, for reference, [MR08]).

From now on, when not otherwise stated, we will assume that we are dealing with a finite directed acyclic graph.

In section 3.1 we will show why the given definition of canonical form and join operation does not generally provide a universal update system on a directed acyclic graph. In section 3.2, we will slightly modify the definition of canonical form, trying to satisfy our statement for a generic graph. A new definitions of join operation is introduced in 3.3. In section 3.4 is described the content of a program which tests our conjecture on a generic graphs. This program run without errors on all graphs up to 5 vertices, and on many graphs on 6 vertices, plus some special graphs. Immediately after, in section 3.5, we describe an example on a 5 vertices graph, show some calculations and provide the full list of its canonical words. In section 3.6, 3.7 and 3.8 we will examine a few special graphs (empty graph, star graph and line graph). At last, in section 3.9, we outline a possible direction of our future work.

3.1 Why the old tools do not work in the generic case

Let us consider a directed acyclic graph Γ. We proved Theorem 2.6.2 building an update system $\mathcal{S}_n^\star$ such that its dynamics monoid $D(\mathcal{S}_n^\star)$ is isomorphic to K_n. Such update system was called *universal* on Γ_n. Similarly, aiming to prove conjecture 4, we try to construct a *universal update system* $\mathcal{S}^\star$ over a directed acyclic graph Γ, i.e., such that its dynamics monoid $D(\mathcal{S}^\star)$ is isomorphic to $\mathbf{HK}_\Gamma$.

In the previous chapter, we defined update functions and vertex sets on a complete graph which could, in theory, be extended to other directed acyclic graphs. However, generic graphs pose a new issue, which was not there for complete graphs: that of choosing a total order for its vertices.

Definition 3.1.1. Let $\Gamma = (V, E)$ be a directed acyclic graph, let $i \in V$ be a vertex. We call *subgraph starting from* i the subgraph $\Gamma_i = (V_i, E_i)$ such that

- its vertex set $V_i \subset V$ contains all vertices $j \neq i$ such that there is a sequence of edges (i, k_1), (k_2, k_3),..., (k_n, j) from i to j;
- the edge set $E_i \subset E$ contains all edges (j_1, j_2) such that both $j_1, j_2 \in V_i$.

For the case of the complete graph, we assumed that the vertices were labeled according to the orientation of edges, which turned out to be coherent with only one possible vertex order. Moreover, the induced vertex order on the subgraph starting from i was unique for all i.

This was a hidden ingredient in the definition of the update system $\mathcal{S}^\star$, as join operation is not commutative and the update function on i consisted in performing the join of the states of the vertices in the neighbourhood $x[i]$ in the order induced by the graph.

When Γ is a generic directed acyclic graph, there can be more than one total order for its vertices, among those induced by the edges orientation. Moreover, for every vertex i, the subgraph starting in i can have more than one total order, and it is possible that the closure of the induced order relations does not correspond to a total order for the graph.

Summarizing

1. If Γ is a complete graph, the vertex order of the subgraph starting from each vertex is uniquely induced by the edges orientation and induces a total order of the graph.
2. The empty graph Empty_n, which is the graph on n vertices without edges, has a unique vertex order up to isomorphisms, and the subgraph starting from each vertex is empty.

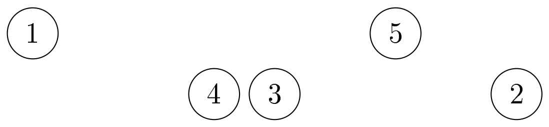


Figure 3.1: Empty_5.

3. The star graph Star_n has a unique sink vertex: just like the empty graphs, there is only one possible vertex order up to isomorphisms (induced by the edge orientation), and the subgraph starting from each vertex is either trivial (only vertex n) or empty (that starting from vertex n). The opposite graph Star_n^{op} (which has a unique sink vertex) has a unique orientation, too: the subgraphs starting from vertex 1 is the empty graph Empty_{n-1}, all the other are trivial.

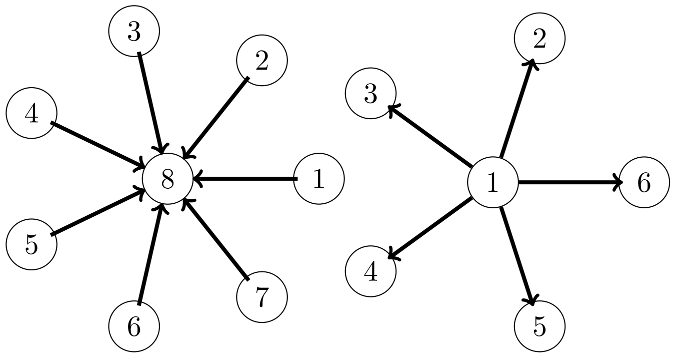


Figure 3.2: Star_8 and Star_6^{op}.

4. In some lucky cases, like the line graph Line_n, the edge orientation induces a unique total order both on the graph and on each connected three vertices subgraph: here, we will see that it is possible to borrow the definition of join operation directly from that of the corresponding complete graph.

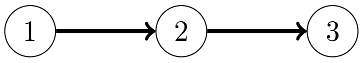


Figure 3.3: Line_3.

5. But, for most graphs, there are several possible vertex orders which respect the edges orientations. In this case, the vertex order on the subgraph starting from vertex 1 is not unique.

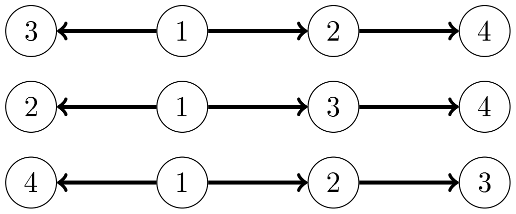


Figure 3.4: A graph G where the order induced on the subgraph starting from 1 is not unique

In the following example we will see that the update system induced by the join operation is not universal if the base graph is generic.

Example 3.1.2. Let us consider an update system whose base graph is the last graph we considered, with vertex states and update functions mutated from those defined for $\mathcal{S}_n^\star$, the universal update system for the complete graph.

In order to make the next example more readable, we will denote by $a, b, c, \dots$ the letters corresponding to vertices $1, 2, 3, \dots$.

We can act on it via the word $\pi = abdc$, which was canonical according to the definition given in the previous chapter. Using the join, we are able to calculate the state on each vertex.

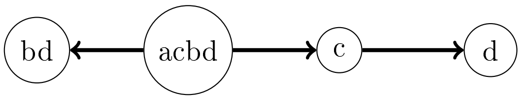


Figure 3.5: The system state $\mathbf{F}_{abdc}\star$ on G.

However, the join of all vertices states does not give the starting word.

$$[d[c,[bd, acbd]]] = [d[c, acbd]] = [d, acbd] = acbd.$$

Moreover, the action of $acbd$ on the update system is

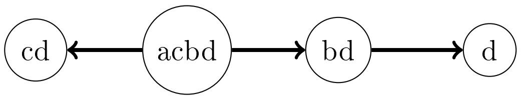


Figure 3.6: The system state $\mathbf{F}_{acbd}\star$ on G.

In this case, the join of these states gives back the word we started from.

$$[d[cd,[bd, acbd]]] = [d[cd, acbd]] = [d, acbd] = acbd.$$

3.2 A new definition of the canonical form

Let us recall the definition of the Hecke-Kiselman monoid $\mathbf{HK}_\Gamma$ whose base graph Γ is an acyclic finite directed graph.

$$\begin{aligned}\mathbf{HK}_\Gamma = \langle a_i, i \in V\colon \quad & a_i^2 = a_i, \text{ for every } i \in V;\\ & a_i a_j a_i = a_j a_i a_j = a_i a_j, \text{ for } i \to j;\\ & a_i a_j = a_j a_i, \text{ for } i \not\twoheadrightarrow j, \text{ and } j \not\twoheadrightarrow i\rangle\end{aligned}$$

Choose an alphabet $A = \{a_i \colon i = 1, \dots, n\}$. Like in the case of the Kiselman monoid, it is possible to consider the following evaluation morphism

$$\pi\colon \mathrm{F}(A) \to \mathbf{HK}_\Gamma\,.$$

We will often abuse the notation and represent elements in $\mathbf{HK}_\Gamma$ via elements in their fiber with respect to π.

The old definition of canonical form, defined in the previous chapter, identifies each element in the Kiselman monoid via a unique element in the fiber $\pi^{-1}\pi(w)$. We aim to provide an analogue for the Hecke-Kiselman monoid $\mathbf{HK}_\Gamma$, for Γ acyclic finite directed graph.

A definition has been introduced in [GM11] in order to identify an analogue of canonical words for the Hecke-Kiselman monoid on the line graph Line_n, via a characterization of their subwords.

Definition 3.2.1. A word $v \in \mathrm{F}(A)$ will be called *strongly special* if all its subwords $a_i u a_i$ are such that u contains both a_{i+1} and a_{i-1}.

Just like in the definition of canonical words, the emphasis is on subwords. Hence, we are going to state what it seems to be a natural extension of this definition for an acyclic finite directed graph Γ.

Definition 3.2.2. Let $w \in \mathrm{F}(A)$. A subword of w of the form $a_i u a_i$, where $a_i \in A$ and $u \in \mathrm{F}(A)$, is called *special* (*with respect to* Γ) if u contains some a_j and some a_k, such that $i \to j$ and $k \to i$ are edges in Γ.

Whenever two vertices i and j are not connected, the subwords $a_i a_j = a_j a_i$ correspond to the same element in $\mathbf{HK}_\Gamma$. As we are willing to find a unique representative in each equivalence class of elements for the Hecke-Kiselman monoid, we introduce another definition, for which we need the graph vertices to be totally ordered, and we require this order to respect the edge orientation, that is, if $i \to j$, then $i < j$.

Definition 3.2.3. A subword $a_i u a_j$ of w is called *untidy* (*with respect to* Γ) if

- $i \geq j$;
- vertex i is not connected to j;
- for each $a_k \in u$, vertex j is not connected to k.

If a subword is not untidy, it is called *tidy* (*with respect to* Γ).

Example 3.2.4.

1. Consider the empty graph Empty_n. Then a subword $a_i u a_j$ is untidy if and only if $i \geq j$ (as all letters commute).
2. Special subwords are always tidy, as the vertex corresponding to the rightmost letter does not commute with all the others.

From now on, when not necessary, we will omit to recall the graph with respect to which a subword is either special, untidy or tidy.

Remark 3.2.5.

1. Any subword $a_i a_i$ is untidy (conventionally, we say that a one letter subword a_i is tidy).

2. A subword $a_i a_j$ is tidy whenever $i < j$.
3. A subword $a_i a_j$ is untidy if and only if i and j are not connected in Γ and $i \geq j$.
4. If $a_i u a_j$ is tidy, u could be untidy (and vice versa).
5. Let $u a_i$ be a untidy subword of w such that, for all $a_h \in u$, both $a_i a_h = a_h a_i$ and $h > i$ hold. Then, $a_i u$ is tidy.
6. If $u a_i$ is an untidy subword, then $a_i u$ comes before $u a_i$ in alphabetical order.

Lemma 3.2.6. *Let u be an untidy word which has no subwords of the form $a_k a_k$. Then, if we move the rightmost letter in the leftmost position a finite number of times, we end up with a tidy word which is in the same fiber of u.*

Proof. Recall that $u = u' a_i$ is untidy only if its rightmost letter commute with all the others, i.e., if $u' a_i = a_i u'$ as elements in $\mathbf{HK}_\Gamma$.

These subwords are finite, hence, after moving the right most letter on the left a finite number of times, we would end up with the starting one. As $i \geq j$ is a total order in Γ, at least one of the words obtained in this way is tidy (the one ending in the letter a_M, where M is the maximal element amongst those in in $u a_i$). □

Remark 3.2.7. Recall that, if $a_h u'' a_k$ is the (first) tidy equivalent of $u = u' a_i$ encountered via the previous lemma, it is not always true that $h < k$: it can be that $h \geq k$, if a_k does not commute with some of the letters in $a_h u''$.

We are ready to state our definition of canonical form for a generic graph Γ.

Definition 3.2.8. A word $w \in \mathrm{F}(A)$, is called *canonical* with respect to the graph Γ if all its subwords are tidy and any subword $a_i u a_i$ is special.

Remark 3.2.9.

1. Notice that a subword $a_i u a_i$ cannot be special if vertex i is a source or a sink.

   Hence, words that are canonical with respect to Γ contain, at most, one occurrence of each letter corresponding to a source or a spring.
2. If $a_i u a_j$ is a subword of a canonical word such that $i > j$, then, in order to be tidy, the subword u must contain a letter a_h such that either $h \to j$ or $j \to h$ is an edge for Γ.

In Remark 2.3.4 we recalled that the authors of [KM09] defined a binary relation in order to prove Theorem 2.3.3. We proceed closely following their idea.

Fix an acyclic finite directed graph Γ. We define the binary relation $\Rightarrow$ in $\mathrm{F}(A)$ as follows: $w \Rightarrow v$ if and only if either

($\overset{1}{\Rightarrow}$) $w = w_1 a_i u a_i w_2$, $v = w_1 a_i u w_2$, and $a_i a_h = a_h a_i$ for all $a_h \in u$, or

($\overset{2}{\Rightarrow}$) $w = w_1 a_i u a_i w_2$, $v = w_1 a_i u w_2$, where u contains a_k such that $i \to k$ and u does not contain any a_h such that $h \to i$, or

($\overset{3}{\Rightarrow}$) $w = w_1 a_i u a_i w_2$, $v = w_1 u a_i w_2$, where u contains a_h such that $h \to i$ and u does not contain any a_k such that $i \to k$, or

($\overset{4}{\Rightarrow}$) $w = w_1 a_i u a_j w_2$, $v = w_1 a_j a_i u w_2$, where for every $a_h \in u$, vertex h is not connected with vertex j in Γ, neither are i and j connected, and $i > j$.

Relations $\overset{1}{\Rightarrow}$, $\overset{2}{\Rightarrow}$ and $\overset{3}{\Rightarrow}$ are analogous to the binary relations $\overset{1}{\to}$, $\overset{2}{\to}$, $\overset{3}{\to}$ defined in [KM09]. Relation $\overset{4}{\Rightarrow}$ is a direct consequence of Lemma 3.2.6. In particular, if $u \overset{4}{\Rightarrow} v$, then v precedes u in alphabetical order and they are equivalent as elements in $\mathbf{HK}_\Gamma$.

Remark 3.2.10. Sequences of the form

$$w \Rightarrow v_1 \Rightarrow v_2 \cdots \Rightarrow v_k, \tag{3.2.1}$$

are finite, since either each word contains one letter less than the previous, or it comes before with respect to the alphabetical order. They all belong to the same fiber with respect to π, i.e., each $v_i \sim w$, for all $i \in [1, k]$.

Extending the notation given in the case of the complete graph Γ_n and the Kiselman monoid K_n, a sequence like (3.2.1) is called *simplifying sequence* (*with respect to* Γ) if v_k is not simplifiable any further, and each $v_i \to v_{i+1}$ is called a *simplifying step*.

Remark 3.2.11. The final element in each simplifying sequence is a canonical word: in fact, if it had a non special subword, we could apply either $\overset{1}{\Rightarrow}$, $\overset{2}{\Rightarrow}$ or $\overset{3}{\Rightarrow}$; if it had an untidy subword, we could apply $\overset{4}{\Rightarrow}$. However, in principle, simplifying sequences starting from the same element w can end in different canonical words.

Lemma 3.2.12. *When $\Gamma = \Gamma_n$ is the complete graph, the new definition of canonical form coincide with the old one.*

Proof. For the complete graph, all words are tidy. □

In the setting of a non complete graph, we do not have a uniqueness result for the canonical form. However, computations show that, for Γ directed acyclic graph without cycles on up to 5 vertices, the universal dynamics monoid for Γ, which, in principle, is a quotient of the Hecke-Kiselman monoid, coincide with the set of all canonical words with respect to Γ.

The following conjecture would provide an answer for the open problem 5.4(v), stated in [GM11].

Conjecture 5. It is possible to represent the elements in $\mathbf{HK}_\Gamma$ as canonical words (with respect to Γ).

Remark 3.2.13. A sketch of the proof could be the following.

1. Prove that $\overset{1}{\Rightarrow}$ can be expressed as a combination of $\overset{4}{\Rightarrow}$ and $\overset{1}{\to}$; as a consequence, if $w \overset{1}{\Rightarrow} u$ and $w \overset{4}{\Rightarrow} v$, then there is x such that $u \overset{\star}{\Rightarrow} x$ and $v \overset{\star}{\Rightarrow} x$, where $\overset{\star}{\Rightarrow}$ is the reflexive-transitive closure of $\overset{i}{\Rightarrow}$, for $i = 1, 2, 3, 4$.

2. Observe that, under the right assumptions,
   - $a_i a_i u a_i \overset{2}{\Rightarrow} a_i a_i u \overset{1}{\to} a_i u$ and $a_i a_i u a_i \overset{1}{\to} a_i u a_i \overset{2}{\Rightarrow} a_i u$;
   - $a_i u a_i a_i \overset{2}{\Rightarrow} a_i u a_i \overset{2}{\Rightarrow} a_i u$ and $a_i u a_i a_i \overset{1}{\to} a_i u a_i \overset{2}{\Rightarrow} a_i u$;
   - $a_i u_1 a_j a_j u_2 a_i \overset{2}{\Rightarrow} a_i u_1 a_j a_j u_2 \overset{1}{\to} a_i u_1 a_j u_2$ and $a_i u_1 a_j a_j u_2 a_i \overset{1}{\to} a_i u_1 a_j u_2 a_i \overset{2}{\Rightarrow} a_i u_1 a_j u_2$,

   and the same holds for $\overset{3}{\Rightarrow}$ and $\overset{1}{\to}$.
3. Prove that, if $w \overset{2}{\Rightarrow} u$ and $w \overset{3}{\Rightarrow} v$, then there is x such that $u \overset{\star}{\Rightarrow} x$ and $v \overset{\star}{\Rightarrow} x$.
4. Prove that if $w \overset{4}{\Rightarrow} u$ and $w \overset{i}{\Rightarrow} v_i$, for $i = 1, 2, 3$, then there is x such that $u \overset{\star}{\Rightarrow} x$ and $v_i \overset{\star}{\Rightarrow} x$.
5. Conclude, applying the Diamond Lemma (see [New42]), that all the simplifying sequences starting from w end in the same element, which we will call $\operatorname{Can} w$.
6. Recalling Remark 3.2.11, the canonical form of w, as defined in 3.2.8, must coincide with $\operatorname{Can} w$ and is unique.

3.3 A generalization of the join operation

When dealing with complete graphs, the unique vertex order induced by the edge orientation is sufficient to describe the dependencies among the states.

The issue we have found with non complete graphs is that a vertex order (which is not necessarily unique) does not fully describe the dependencies among state vertices. In fact, relation $i < j$ holds even if vertex j does not belong to the subgraph starting in i: it only ensures that i does not belong to the subgraph starting in j.

A consequence of Theorem 2.6.2 is that, on canonical words, the join defined in Section 2.4 for the complete graph already takes into account the state dependencies. On a generic directed acyclic graph, however, we need a new definition for the join operation which does not rely only on the vertex order.

Remark 3.3.1. The goal of the new join operation is to replace the old one in the definition of our candidate for the universal update system on a generic graph Γ. We will define the update system on Γ on n vertices assuming that the definition of join is well posed on an alphabet on $n-1$ letters. On the other side, the definition of join is given by induction on the number of letters in the alphabet, and its definition for words in the alphabet on n letters relies on the states calculated on a graph on n vertices.

Let us define the universal update system on a directed acyclic graph Γ.

Definition 3.3.2. The update system $\mathcal{S}^\star$ is the triple (Γ, S_i, f_i), where

1. Γ is a directed acyclic graph on n vertices, where the vertices are ordered according to the edge orientation: $i \to j$ if and only if $i < j$.

2. On vertex i, the state set is

$$S_i = \begin{cases} \{\star, a_n\} & \text{if } i = n \\ \{\star, a_{n-1}, a_{n-1}a_n\} & \text{if } i = n-1 \\ \{\star\} \cup a_i \llbracket S_n, \llbracket \dots, \llbracket S_{i+2}, S_{i+1} \rrbracket \dots \rrbracket \rrbracket & \text{if } 1 \leq i \leq n-2 \end{cases}$$

3. On vertex i, the vertex function is

$$\begin{array}{rcl} f_i \colon \prod_{j \in x[i]} S_j & \to & S_i \\ s[i] = \{s_{j_1}, s_{j_2} \dots s_{j_k} \colon j_1 < \dots < j_k\} & \mapsto & a_i \llbracket s_{j_k}, \llbracket \dots, \llbracket s_{j_2}, s_{j_1} \rrbracket \dots \rrbracket \rrbracket \end{array}$$

   if $i \leq n-2$, whereas $f_{n-1}(s_n) = a_{n-1}a_n$ and $f_n \equiv a_n$ is a constant.

Observe that in the definition just given, the new join operation only involves words in the alphabet $\{a_2, \dots, a_n\}$.

In the following definition, $\sim$ (or $\sim_k$) is the equivalence relation between words that, starting from the state $\star = (\star, \dots, \star)$, calculate the same system state (or the same state on vertex k).

We write $a_0 \dots \hat{a_i} \dots a_n$ if the letter a_i has been deleted from its position in the given word.

Definition 3.3.3. Fix a directed acyclic graph Γ, with vertex set V. Let u, v be words in V, where $u = a_0a_1 \dots a_n$ and $v = b_0b_1 \dots b_m$. Either of the following occurs

1. if $v = \star$, then $\llbracket u, v \rrbracket = u$;
2. if $u = \star$, then $\llbracket u, v \rrbracket = v$;
3. if a_i is the rightmost letter in u such that $u \sim \overline{u}a_i$, where $\overline{u} = a_0 \dots \hat{a_i} \dots a_n$, and $v \sim_k va_i$ for all $k \neq i$:
   - if v does not contain a_i, then
   $$\llbracket u, v \rrbracket = \llbracket \overline{u}, v \rrbracket a_i;$$
   - Let $b_j = a_i$ be the rightmost occurrence of a_i in v, and let $\overline{v} = b_0 \dots \hat{b_j} \dots b_m$. If there is k such that the relation $v \sim_k \overline{v}a_i$ does not hold, then
   $$\llbracket u, v \rrbracket = \llbracket \overline{u}, v \rrbracket a_i;$$
   otherwise, if the relation $v \sim_k \overline{v}a_i$ holds for all k (including $k = i$), then
   $$\llbracket u, v \rrbracket = \llbracket \overline{u}, \overline{v} \rrbracket a_i;$$
4. if b_j is the rightmost letter in v such that $v \sim \tilde{v}b_j$, where $\tilde{v} = b_0 \dots \hat{b_j} \dots b_m$, and $u \sim_k ub_j$ for all $k \neq j$:

- if u does not contain b_j, then

$$\llbracket u, v \rrbracket = \llbracket u, \tilde{v} \rrbracket b_j$$

- Let $a_i = b_j$ be the rightmost occurrence of b_j in u, and let $\tilde{u} = a_0 \ldots \hat{a}_i \ldots a_n$. If there is k such that the relation $u \sim_k ub_j$ does not hold, then

$$\llbracket u, v \rrbracket = \llbracket u, \tilde{v} \rrbracket b_j$$

otherwise, if the relation $u \sim_k ub_j$ holds for all k (including $k = j$), then,

$$\llbracket u, v \rrbracket = \llbracket \tilde{u}, \tilde{v} \rrbracket b_j.$$

If there exist both a_i and b_j satisfying condition 3 and condition 4, we choose according to the total order defined on the vertices on Γ. If $a_i = b_j$, then point 3 and point 4 coincide. If $a_i < b_j$ we will proceed following point 3, otherwise point 4.

If none of the four above conditions is satisfied, then $\llbracket u, v \rrbracket = \star$.

Remark 3.3.4.

1. For all $a_i, a_j \in A$ such that $i \neq j$,

$$\llbracket a_i, a_j \rrbracket = a_{\max(i,j)} a_{\min(i,j)}.$$

2. For all words $u = u'a$, $v = v'a$ in $\mathrm{F}(A)$,

$$\llbracket u, v \rrbracket = \llbracket u', v' \rrbracket a.$$

3. Let i be a source vertex of the graph, a_i being the corresponding letter, if u, v do not end in a source, then

$$\llbracket ua_i, v \rrbracket = \llbracket u, v \rrbracket a_i = \llbracket u, va_i \rrbracket .$$

3.4 Computational results

Let $\mathcal{S}_\Gamma^\star$ be the update system defined through the new join operation on an acyclic directed graph Γ.

Consider the evaluation morphism $\mathbf{F}\colon \mathrm{F}(A) \to D(\mathcal{S}_\Gamma^\star)$, mapping each word $w \in \mathrm{F}(A)$ to the corresponding evolution $\mathbf{F}_w \in D(\mathcal{S}_\Gamma^\star)$. If $\mathbf{p} = (p_1, \ldots, p_n) = \mathbf{F}_w \star$, then, for every vertex i in Γ,

$$\operatorname{Can} w = \operatorname{Can} \llbracket p_n, \llbracket \ldots, \llbracket p_{i+2}, p_{i+1} \rrbracket \ldots \rrbracket \rrbracket .$$

This would mean that $D(\mathcal{S}_\Gamma^\star) = \mathbf{HK}_\Gamma$, i.e., that this update system is universal; moreover, that $\pi^{-1}\pi(w)$ contains a unique canonical word, which is $\operatorname{Can} w$.

The flow chart in next page describes a program we have written in order to test if $\mathcal{S}_\Gamma^\star$ is universal.

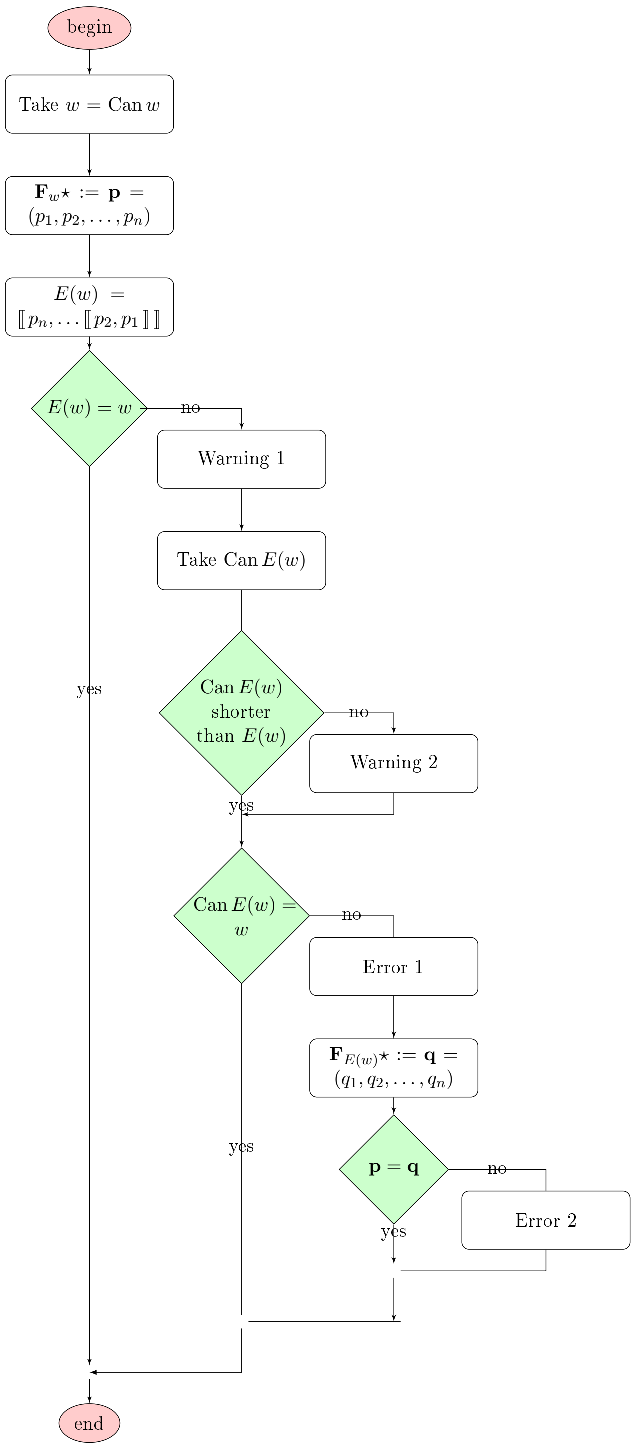

begin
Take $w = \mathrm{Can}\, w$
$\mathbf{F}_{w^\star} := \mathbf{p} = (p_1, p_2, \ldots, p_n)$
$E(w) = [\![p_n, \ldots [\![p_2, p_1]\!]]\!]$
$E(w) = w$
no
Warning 1
Take $\mathrm{Can}\, E(w)$
yes
$\mathrm{Can}\, E(w)$ shorter than $E(w)$
no
Warning 2
yes
$\mathrm{Can}\, E(w) = w$
no
Error 1
$\mathbf{F}_{E(w)^\star} := \mathbf{q} = (q_1, q_2, \ldots, q_n)$
yes
$\mathbf{p} = \mathbf{q}$
no
Error 2
yes
end

Figure 3.7: Flow chart to test $\mathcal{S}^\star_\Gamma$, a possible universal update system.

The idea is the following:

- we recursively compute a complete list of canonical words.
- for each word w in the list, we perform the update functions in the order prescribed by w, hence obtaining the system state

$$\mathbf{F}_w \star \colon = \mathbf{p} = (p_1, \ldots, p_n)$$

- we perform the *join* of the entries $p_1, \ldots, p_n$, thus obtained. This yields

$$E(w) = \llbracket p_n, \llbracket \ldots, \llbracket p_2, p_1 \rrbracket \rrbracket \rrbracket$$

- we print a warning message if either $E(w) \neq w$ or $\operatorname{Can} E(w)$ is shorter then $E(w)$. We print an error message if $\operatorname{Can} E(w) \neq w$.
- we compute

$$\mathbf{F}_{E(w)} \star \colon = \mathbf{q} = (q_1, \ldots, q_n)$$

  If $\mathbf{q} \neq \mathbf{p}$, we issue a error message.

Hence, whenever this program ends on graph Γ without error messages,

- it ensures that, for the defined update system $\mathcal{S}^\star_\Gamma$ on Γ, $D(\mathcal{S}^\star_\Gamma) = \mathbf{HK}_\Gamma$.
- it provides a tool to calculate the canonical form of each word with respect to Γ, and a list of canonical words for $\mathbf{HK}_\Gamma$.

So far, this program runs without errors on all graphs on up to 5 vertices and on many graphs on 6 vertices, plus Empty_n, Star_n, and Line_n, which we will define and analyze in following sections.

3.5 An example of the universal update system on a generic directed acyclic graph

Recall that the adjacency matrix of a directed acyclic graph is an $n \times n$ matrix $M = (m_{ij})$ where the entry m_{ij} is 1 if and only if (i, j) is an edge in E, otherwise it is 0.

Remark 3.5.1. Each adjacency matrix corresponds to a total order of the vertices of one of the directed acyclic graphs on n vertices. We labeled it via the decimal representation of the binary number given by the entries $m_{n-1,n} m_{n-2,n-1} m_{n-2,n} \ldots m_{1,2}$. For example, the graph on 5 vertices in the following picture will be called G_{811}, (as $1100101011_2 = 811_{10}$).

In order to make the next example more readable, we will denote by $a, b, c, \ldots$ the letters corresponding to vertices $1, 2, 3, \ldots$.

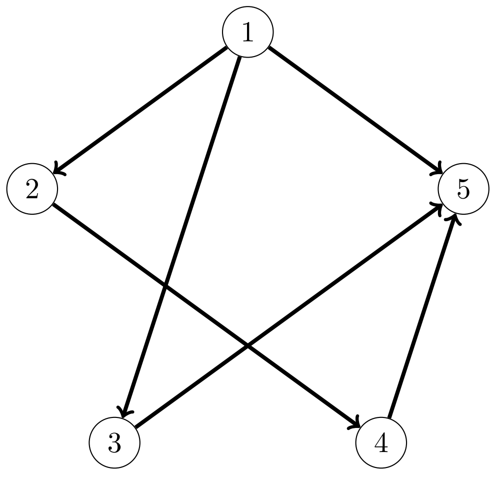


Figure 3.8: Graph G_{811}.

Example 3.5.2. Let us take the word $w = dbacedb$ in the alphabet $\{a, b, c, d, e\}$ of letters corresponding to the vertices $\{1, 2, \dots, 5\}$. Consider the action $F_w = F_d F_b F_a F_c F_e F_d F_b$ on the initial state $\star = (\star, \star, \dots, \star)$.

Recall that each F_i modifies only the state on vertex i according to the states on the children vertices. Hence,

$$\begin{aligned}(F_b\star)_2 &= b\star = b\\ (F_d(\star, b, \star, \star, \star))_4 &= d\star = d\\ (F_e(\star, b, \star, d, \star))_5 &= e.\end{aligned}$$

This is equivalent as writing

$$F_{edb}\star = F_e F_d F_b\star = F_e F_d(\star, b, \star, \star, \star) = F_e(\star, b, \star, d, \star) = (\star, b, \star, d, e).$$

On vertex 3 we simply get $(F_c(\star, b, \star, d, e))_3 = ce$.

At this point, the update of vertex 1 calculates, by definition, $a\llbracket e, \llbracket ce, b\rrbracket\rrbracket$. We have

$$\llbracket ce, b\rrbracket = \llbracket c, b\rrbracket e = bce.$$

Consider the first equality. In theory, both e and b could be in the rightmost position of the join, as

$$\begin{aligned}(F_{ceb}\star)_i = (F_{ce}\star)_i &\quad \text{for } i \neq 2\\ (F_{be}\star)_i = (F_b\star)_i &\quad \text{for } i \neq 5\end{aligned}$$

In this situation, we choose according to the total vertex order and, since $5 > 2$, $\llbracket ce, b\rrbracket = \llbracket c, b\rrbracket e$. As for the second equality, remark 3.3.4(i) assures that $\llbracket c, b\rrbracket = bc$, because vertices 2 and 3 are not connected.

Finally,

$$\llbracket e, \llbracket ce, b\rrbracket\rrbracket = \llbracket e, bce\rrbracket = bce,$$

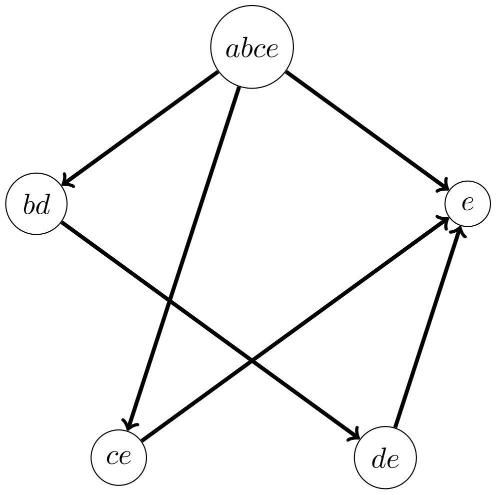
hence $(F_{acedb}\star)_1 = abce$.

The new update of vertex 2 gives the new state bd, as the actual state on vertex 4 is d. Hence, $(F_b(abce, b, ce, d, e))_3 = bd$.

In the same way, the state on vertex 4 is de, as it is calculate on a system state where the state on vertex 5 is e. Hence,

$$F_{dbacedb}\star = F_d(abce, bd, ce, d, e) = (abce, bd, ce, de, e).$$

Here it is the system state reached applying F_w to the empty state $\star$.



$w = dbacedb$

Figure 3.9: The system state $\mathbf{F}_{dbacedb}\star$ on G_{811}.

Given the system state $(abce, bd, ce, de, e)$, we want to use the join operation in order to take back the initial word w. Now, let us join the states according to the vertex order in the graph, from 1 to n.

$$\begin{aligned} [\![bd, abce]\!] &= [\![bd, ace]\!]\, b \\ &= [\![b, ace]\!]\, db \\ &= [\![b, ac]\!]\, edb \\ &= [\![b, a]\!]\, cedb \\ &= [\![b, \star]\!]\, acedb \\ &= bacedb \end{aligned} \tag{3.5.1}$$

$$\begin{aligned} [\![ce, [\![bd, abce]\!]]\!] &= [\![ce, bacedb]\!] \\ &= [\![ce, baced]\!]\, b \\ &= [\![ce, bace]\!]\, db \\ &= bacedb \end{aligned}$$

$$\begin{aligned}
[\![de, [\![ce, [\![bd, abce]\!]]\!]]\!] &= [\![de, bacedb]\!] \\
&= [\![de, baced]\!]\, b \\
&= [\![de, bace]\!]\, db \\
&= [\![d, bac]\!]\, edb \\
&= [\![d, ba]\!]\, cedb \\
&= [\![d, b]\!]\, acedb \\
&= [\![d, \star]\!]\, bacedb \\
&= dbacedb
\end{aligned}$$

$$\begin{aligned}
[\![e, [\![de, [\![ce, [\![bd, abce]\!]]\!]]\!]]\!] &= [\![e, dbacedb]\!] \\
&= [\![e, dbaced]\!]\, b \\
&= [\![e, dbace]\!]\, db \\
&= [\![\star, dbac]\!]\, edb = \\
&= dbacedb.
\end{aligned}$$

We have seen in Definition 3.3.3 that $[\![u, \star]\!] = [\![\star, u]\!] = u$. In all but one the relations above, where both u and v where not $\star$, we wrote $[\![u, v]\!] = [\![u', v']\!]\, l$: either $u = u'l$ and $v' = v$, or $v = v'l$ and $u' = u$, or both $u = u'l$ and $v = v'l$.

The only tricky passage was (3.5.1): there, we had to take b in $abce$, as it was the only one letter (in both words)satisfying our conditions. In fact, the following letters do not satisfy all the requirements

- let us try d in bd: $(F_{abce}\star) \not\sim_i (F_{abced}\star)$ on some $i \neq 4$ (specifically, on $i = 1, 2$);
- let us try b in bd: $(F_{bd}\star) \not\sim_2 (F_{db}\star)$;
- let us try e in $abce$: $(F_{bde}\star) \not\sim_4 (F_{bd}\star)$;
- let us try c in $abce$: $(F_{abce}\star) \not\sim_3 (F_{abec}\star)$;
- let us try a in $abce$: $(F_{abce}\star) \not\sim_1 (F_{bcea}\star)$;

The occurrence of b in the right side word is not in the rightmost position, however $(F_{abce}\star)_i = (F_{aceb}\star)_i$ on all vertices i (including $i = 2$). On the left side word, there is an occurrence of letter b: however, we have $(F_{bd}\star)_i = (F_{bdb}\star)_i \neq (F_{db}\star)_i$. Hence,

$$[\![bd, abce]\!] = [\![bd, ace]\!]\, b.$$

There are 244 canonical words on G_{811}: $\star$, a, b, c, d, e, ba, ca, ad, ea, ab, bc, db, be, ac, cd, ec, bd, ed, ae, ce, de, bca, dba, bea, cad, eca, bad, ead, cea, dea, cab, adb, eab, abc, cdb, bec, edb, abe, bce, dbe, bac, acd, eac, bcd, ecd, aec, dec, abd, bed, aed, ced, bae, cae, ade, ace, cde, bde, $cdba$, $beca$, $edba$, $bcea$, $dbea$, $bcad$, $ecad$, $deca$, $bead$, $cead$, $cdea$, $bdea$, $cadb$, $ecab$, $badb$, $eadb$, $ceab$, $deab$, $acdb$, $eabc$, $ecdb$, $abec$, $dbec$, $aedb$, $cedb$, $cabe$, $adbe$, $abce$, $cdbe$, $dbac$, $beac$, $bacd$, $eacd$, $ceac$, $deac$, $abcd$, $becd$, $aecd$, $baec$, $caec$, $adec$, $bdec$, $cabd$, $eabd$, *abed*, *bced*, *dbed*, *baed*, *caed*, *aced*, *bcae*, *dbae*, *cade*, *bade*, *bace*, *acde*, *bcde*, *abde*, *ecdba*, *dbeca*, *cedba*, *cdbea*, *becad*, *bdeca*, *bcead*, *dbead*, *bcdea*, *bcadb*, *ecadb*, *decab*, *beadb*, *ceadb*, *cdeab*, *bdeab*, *bacdb*, *eacdb*, *ceabc*, *deabc*, *aecdb*, *cabec*, *adbec*, *baedb*, *caedb*, *acedb*, *cadbe*, *badbe*, *acdbe*, *edbac*, *bceac*, *dbeac*, *beacd*, *ceacd*, *cdeac*, *bdeac*, *eabcd*, *abecd*, *dbecd*, *baecd*, *caecd*, *bcaec*, *dbaec*, *cadec*, *badec*, *abdec*, *ecabd*, *ceabd*, *deabd*, *cabed*, *adbed*, *abced*, *cdbed*, *bcaed*, *dbaed*, *baced*, *cdbae*, *bcade*, *dbace*, *bacde*, *abcde*, *cabde*, *dbecad*, *cdbead*, *becadb*, *bdecab*, *bceadb*, *dbeadb*, *bcdeab*, *beacdb*, *ceacdb*, *cdeabc*, *bdeabc*, *baecdb*, *caecdb*, *cadbec*, *badbec*, *bcaedb*, *dbaedb*, *bacedb*, *bcadbe*, *bacdbe*, *cedbac*, *cdbeac*, *bceacd*, *dbeacd*, *bcdeac*, *ceabcd*, *deabcd*, *cabecd*, *adbecd*, *bcaecd*, *dbaecd*, *cdbaec*, *bcadec*, *cabdec*, *decabd*, *cdeabd*, *bdeabd*, *cadbed*, *badbed*, *acdbed*, *cdbaed*, *dbaced*, *dbecadb*, *cdbeadb*, *bceacdb*, *dbeacdb*, *bcdeabc*, *bcaecdb*, *dbaecdb*, *bcadbec*, *cdbaedb*, *dbacedb*, *cdbeacd*, *cdeabcd*, *bdeabcd*, *cadbecd*, *badbecd*, *cdbaecd*, *bdecabd*, *bcdeabd*, *bcadbed*, *bacdbed*, *cdbeacdb*, *cdbaecdb*, *bcdeabcd*, *bcadbecd*.

3.6 An update system on the empty graph

The empty graph, also called discrete graph, is the graph without edges. We will denote it by Empty_n. Let us label its vertices from 1 to n.

A word w cannot admit any special subword with respect to Empty_n. Hence, w is canonical if and only if all its subwords are tidy. Recalling Remark 3.2.5, this means that letters in w are in alphabetical order and each letter occurs at most once: this implies uniqueness for the canonical word on Empty_n.

Definition 3.6.1. The universal update system $S^{\star}_{\mathsf{Empty}_n} = (\mathsf{Empty}_n, S_i, f_i)$ is defined as follows

1. Empty_n is the base graph;
2. on each vertex i, the state set is $S_i = \{\star, a_i\}$;
3. on each vertex i, the update function is the constant $f_i \equiv a_i$.

In this setting, the action of any word w is

$$(F_w \star)_i = \begin{cases} i, & \text{if } a_i \in w \\ \star, & \text{if } a_i \notin w \end{cases}$$

As any two letters commute, the join operation $[u, v]$ puts in alphabetical order the letters occurring either in u or v. Hence, if $w = \operatorname{Can} w$, $\mathbf{F}_w \star = (p_1, p_2 \dots, p_n)$, then

$$[p_n, \dots, [p_2, p_1]] = p_1 p_2 \dots p_n,$$

where, since $S_i = \{\star, i\}$, each p_i is either a_i or the empty word.

This is enough to prove that the dynamics monoid $D(\mathcal{S}^{\star}_{\mathsf{Empty}_n})$ coincide with the Hecke-Kiselman monoid on $\mathbf{HK}_{\mathsf{Empty}_n}$, and their elements correspond uniquely to words in $\mathrm{F}(A)$ whose letter appear (at most once) in alphabetical order. Finally, their cardinality is

$$|D(\mathcal{S}^{\star}_{\mathsf{Empty}_n})| = |\,\mathbf{HK}_{\mathsf{Empty}_n}\,| = 2^n. \tag{3.6.1}$$

as they correspond to the 2^n subsets of $\{a_1, \dots, a_n\}$.

3.7 An update system on the star graph

Consider the star graph Star_n, i.e., the directed acyclic graph where the only edges are of the form $i \to n$, for all vertices $i < n$.

$$\mathsf{Star}_n = \{V = \{1, 2, \ldots, n\}; E = \{(i, n) \text{ for all } i \in [1, n-1]\}\}$$

The induced subgraph on vertices $\{1, 2, \ldots, n-1\}$ coincides with the empty graph Empty_{n-1}, while vertex n is a child of all the others. Hence, the unique constrain for the vertex order is that n must be the maximal element and we can label in any order all the other elements.

Lemma 3.7.1. *Canonical words are either $u \in \mathrm{Can}(\mathsf{Empty}_{n-1})$, or $u_1 a_n u_2$, where $u_1 u_2 \in \mathsf{Empty}_{n-1}$.*

Proof. Just as seen for the empty graph, no word can have special subwords, as no vertex has both a parent and a child vertices: for this reason, canonical words contain at most one occurrence of any letter, as they all commute. □

Remark 3.7.2. In this setting, the join operation is very close to the join on the empty set.

- If u, v do not contain letter a_n, then $[\![u, v]\!]_{\mathsf{Star}_n} = [\![u, v]\!]_{\mathsf{Empty}_{n-1}}$.
- If u do not contain letter a_n, while $v = v_1 a_n v_2$, then

$$\begin{aligned} [\![u, v]\!]_{\mathsf{Star}_n} &= \overline{v_1} a_1 [\![u, v_2]\!]_{\mathsf{Empty}_{n-1}} \\ [\![v, u]\!]_{\mathsf{Star}_n} &= \overline{v_1} a_1 [\![v_2, u]\!]_{\mathsf{Empty}_{n-1}} \end{aligned}$$

  and $\overline{v_1}$ contains the letters in v_1 in alphabetical order.
- If $u = u_1 a_n u_2$ and $v = v_1 a_n v_2$, then

$$[\![u, v]\!]_{\mathsf{Star}_n} = [\![u_1, v_1]\!]_{\mathsf{Empty}_{n-1}} a_1 [\![u_2, v_2]\!]_{\mathsf{Empty}_{n-1}}$$

We are now ready to build an update system on Star_n.

Definition 3.7.3. The update system $\mathsf{Star}_n^\star$ is the triple $(\mathsf{Star}_n, S_i, \mathbf{F}_i)$, where

1. Star_n is, as before, the star graph on n vertices, where $i \to n$, for all $i < n$.
2. The state sets are

$$\begin{aligned} S_i &= \{\star, a_i, a_i a_n\}, \text{ for all } i < n, \\ S_n &= \{\star, a_n\} \end{aligned}$$

3. On vertex i,

$$\begin{array}{rcl} f_i\colon S_n & \to & S_i \\ s_n & \mapsto & a_i s_n. \end{array}$$

On vertex n, the vertex function is $f_n \equiv a_n$.

Lemma 3.7.4. $\mathbf{HK}_{\mathsf{Star}_n}$ *is isomorphic to* $D(\mathsf{Star}_n^\star)$.

Proof. Consider the evaluation morphism $\mathbf{F}\colon \mathrm{F}(A) \to D(\mathcal{S}_n^\star)$, mapping each word $w \in \mathrm{F}(A)$ to the corresponding evolution $\mathbf{F}_w \in D(\mathcal{S}_n^\star)$. Let $w = \operatorname{Can} w$. If $\mathbf{p} = (p_0, \dots, p_{n-1}) = \mathbf{F}_w \star$, then:

- $p_i = a_i$ for all letters a_i that appear in w on the right of a_n;
- $p_j = a_j a_n$ for all letters a_j that appear in w on the left of a_n;
- $p_k = \star$ for all a_k that does not appear in w.

Applying Remark 3.7.2, we have that

$$[\![p_n, \dots [\![p_2, p_1]\!]]\!] = w_1 a_n w_2$$

where $a_i \in w_1$ if and only if $p_i = a_i a_n$, $a_j \in w_2$ if and only if $p_j = a_j$. Hence, $w_1 a_n w_2 = w$, as w_1 and w_2 are in alphabetical order. □

Lemma 3.7.5. *The cardinality of the dynamics monoid of the update system* $(\mathsf{Star}_n, S_i, f_i)$ *is*

$$|D(\mathsf{Star}_n)| = |\,\mathbf{HK}_{Star_n}\,| = 2^{n-1} + 3^{n-1}.$$

Proof. Let us proceed by induction. When $n = 1$, we have $\mathsf{Star}_1 = \mathsf{Empty}_1$ and the two definition for the update system coincide.

The star graph Star_n consist in an empty graph on $n-1$ vertices, and $n-1$ edges from i to n. Recalling Lemma 3.6.1, there are exactly 2^{n-1} canonical words that do not contain the letter a_n. We need to show that there are exactly 3^{n-1} canonical words that do contain the letter a_n.

Assume that the thesis holds for a graph on $n-1$ vertices $\{a_2, \dots, a_n\}$, i.e. that

$$|D(\mathsf{Star}_{n-1})| = |\,\mathbf{HK}_{\mathsf{Star}_{n-1}}\,| = 2^{n-2} + 3^{n-2}.$$

In particular, there are exactly 3^{n-2} canonical words that contain a_n (the sink vertex).

Now, add the new vertex a_1, which is connected to a_n via the edge (a_1, a_n). Any canonical word in Star_{n-1} containing a_n was of the form ua_nv, where the letters in u and v were in anti-alphabetical order. This property holds for canonical words w.r.to Star_n, too. Note that they can contain at most one occurrence of the letter a_{n-1}. Hence, ua_nv gives rise to three canonical words in Star_n: ua_nv, $u'a_nv$ and ua_nv', where u' and v' both contain a_{n-1} and the letters in u and v, respectively. This proves the thesis. □

Finally, what we have seen up to now works not only for Star_n, a graph with a unique "sink" vertex and $n-1$ "source" vertices, but also for $\mathsf{Star}_n^{\mathrm{op}}$, where all the edges are flipped, thanks to the following theorem, which was proved in [GM11].

Theorem 3.7.6. *For any directed acyclic graph* Γ*,*

$$\mathbf{HK}_\Gamma \cong \mathbf{HK}_{\Gamma^{op}},$$

where Γ^{op} *is the graph obtained reversing the orientation of all edges in* Γ*.*

3.8 An update system on the line graph

Let us recall the definition of strongly special words given in 3.2.1. in [GM11] it is stated that equivalence classes of strongly special words correspond exactly to short-braid avoiding permutations in S_{n+1}, which correspond to the number of 321-avoiding permutations.

Definition 3.8.1. The update system $\mathsf{Line}_n^\star$ is the triple $(\mathsf{Line}_n, S_i, \mathbf{F}_i)$, where

1. Line_n is the Dynkin graph A_n, with the given orientation, for all $i \in \{1, 2, \ldots, n-1\}$.
2. The state set on vertex i is
$$S_i \quad = \quad \{\star, a_i, a_i a_{i+1}, \ldots, a_i a_{i+1} \ldots a_n\}.$$
3. On vertex i, the update function is
$$\begin{aligned} f_i \colon S_{i+1} &\to S_i \\ s_{i+1} &\mapsto a_i s_i + 1. \end{aligned}$$
while $f_n \equiv a_n$.

Lemma 3.8.2. *If* s_i *is the state on vertex* i*, then* $(\mathbf{F}_{s_i}\star)_i = s_i$*.*

Proof. Proceed by induction on the number of vertices n. Id $n=1$, the statement is trivial. If the statement is true on Line_{n-1}, then it is true for states of all vertices $i > 1$ on Line_{n-1}. State on vertex i follows from the definition of f_1 and inductive hypothesis on vertex 2. □

The canonical form defined in the complete graph does not suffice anymore, as for any pair of non connected vertices $i < j$, the new relation $a_i a_j = a_j a_i$ is stronger than $a_i a_j a_i = a_j a_i a_j = a_i a_j$, which held in the complete graph, where each pair of vertices is connected.

Definition 3.8.3. For every word w in $F(A)$, $\mathrm{Can}_{\mathsf{Line}_n}(w)$ is the maximal element with respect to the anti-alphabetical vertex order among the words which are equivalent to $\mathrm{Can}_{\Gamma_n}(w)$ modulo the reflexive relations $a_i a_j = a_j a_i$, for all non connected pairs i, j.

This definition is well posed, as the anti-alphabetical order is total.

Remark 3.8.4. For any word w,

$$\mathrm{Can}_{\mathsf{Line}_n} w = \mathrm{Can}_{\Gamma_n}(\mathrm{Can}_{\mathsf{Line}_n} w).$$

The following lemma gives us a more detailed description of canonical words with respect to Line_n.

Lemma 3.8.5.

1. *For any two-letters subword $a_i a_j$ in $w = \mathrm{Can}_{\mathsf{Line}_n} w$ we have that either $j = i + 1$ or $j < i$.*

2. *$w = \mathrm{Can}_{\mathsf{Line}_n} w = u_{[l_1,l_1+m_1]}u_{[l_2,l_2+m_2]}\dots u_{[l_h,l_h+m_h]}$, where*
   - *we used a short notation to write the subwords $u_{[l_i,l_i+m_i]} = a_{l_i}a_{l_{i+1}}\dots a_{l_i+m_i}$ for all i, and*
   - *for two consecutive subwords $u_{[l_i,l_i+m_i]}u_{[l_{i+1},l_{i+1}+m_{i+1}]}$ of w, we have that $l_i > l_{i+1}$ and $l_i + m_i > l_{i+1} + m_{i+1}$.*

Proof.

1. The canonical word cannot contain any subword of the form $a_i a_j$, since they commute and we required the anti alphabetical order.

2. It follows from point 1.

□

Remark 3.8.6.

1. Let w be a word and let us consider its action on $\mathsf{Line}_n^\star$, $F_w\star = \mathbf{s} = (s_1, \dots, s_n)$. For each $a_i \in w$,
   - a_i cannot appear in state s_{i+k}, for $k > 0$.
   - a_i appears in state s_{i-k} only if it appears as well in states s_i,s_{i-1}, …, s_{i-k+1}.

2. If w is canonical, recalling Lemma 3.8.5 we can express it as $w = \mathrm{Can}_{\mathsf{Line}_n} w = u_{[l_1,l_1+m_1]}u_{[l_2,l_2+m_2]}\dots u_{[l_h,l_h+m_h]}$. Then,
   - $s_{l_i} = u_{[l_i,l_i+m_i]} = a_{l_i}a_{l_{i+1}}\dots a_{l_i+m_i}$ for all vertices l_i;
   - $s_k = a_k a_{k+1}\dots a_{m_i}$ if the occurrence of k in subword $u_{[l_i,l_h+m_i]}$ is the leftmost in w.

Lemma 3.8.7. $\mathbf{HK}_{\mathsf{Line}_n}$ *is isomorphic to $D(\mathsf{Line}_n^\star)$.*

Proof. Consider the evaluation morphism $\mathbf{F}\colon \mathrm{F}(A) \to D(\mathsf{Line}_n^\star)$, mapping each word $w \in \mathrm{F}(A)$ to the corresponding evolution $\mathbf{F}_w \in D(\mathsf{Line}_n^\star)$. Take $w = \mathrm{Can}\, w$. If $\mathbf{p} = (p_0, \dots, p_{n-1}) = \mathbf{F}_w \star$, then consider $[p_n, [p_{k+1} \dots [p_2, p_1]]]$. We want to prove that this join is a canonical word, hence, by Remark 3.8.6(2), it will coincide with w.

We proceed by induction on the number of vertices and on the number of states we are joining. On Line_1 there is nothing to prove. If there is more than one state, the state on vertex 1 is a canonical word, as it is either the empty state $\star$ or of the form $p_1 = a_1 \dots a_k$.

As $[p_{k+1} \dots [p_2, p_1]]$ is canonical by induction, let us show that $[p_k, [p_{k+1} \dots [p_2, p_1]]]$ is canonical as well. Recalling the definition of update functions in $\mathsf{Line}_n^\star$, either of the following holds

- $p_k = \star$;
- $p_{k-1} = a_k p_k$;
- $p_{k-1} = a_{k-1} a_k \dots a_{k+h}$ and $p_k = a_k \dots a_{k+h} \dots a_{k+h+h'}$.

If $p_k = \star$ or $p_k = a_k p_{k+1}$, then p_k is a quasi subword of $[p_{k-1} \dots [p_2, p_1]]$, hence

$$[p_k, [p_{k-1} \dots [p_2, p_1]]] = [p_{k-1} \dots [p_2, p_1]]$$

Otherwise, by Remark 3.8.6(1), letter $a_{k+h+h'}$, which is the rightmost letter in p_k, does not appear in any of the states p_{k-1}, ..., p_2, p_1, so

$$[p_k, [p_{k-1} \dots [p_2, p_1]]] = p_k [p_{k-1} \dots [p_2, p_1]].$$

Now, consider $p_k = a_k a_{k+1} \dots a_{k+h} = u_{[l_k, l_{i+k}]}$ and $[p_{k+1} \dots [p_2, p_1]]$ which, being canonical, contains a prefix of the form $u_{[l_i, l_{i+m}]}$. Recalling the definition of join, l_i is among $a_1, a_2, \dots, a_{k-1}$, and $l_{i+m} < a_{k+h}$, so $p_k[p_{k-1} \dots [p_2, p_1]]$ is canonical. $\square$

We conclude recalling some definitions and results from [GM11], which show that the cardinality of $\mathbf{HK}_{\mathsf{Line}_n}$ is the Catalan number C_{n+1}.

Definition 3.8.8. For $i = 1, 2, \dots, n$, we denote by T_i the following transformation of $\{1, 2, \dots, n\}$:

$$\begin{pmatrix} 1 & 2 & \dots & i-1 & i & i+1 & i+2 & \dots & n & n+1 \\ 1 & 2 & \dots & i-1 & i & i & i+2 & \dots & n & n+1 \end{pmatrix}.$$

The semigroup of all order-preserving and order-decreasing total transformations of $\{1, 2, \dots, n, n+1\}$ is denoted by $\mathcal{C}_{n+1}$.

Proposition 3.8.9.

1. *If Γ, as unoriented graph, is a Dynkin diagram of type A_n, then, $|\mathbf{HK}_\Gamma| = C_{n+1}$ if and only if A_n is oriented as Line_n, i.e.,*
2. *The Hecke-Kiselman monoid $\mathbf{HK}_{\mathsf{Line}_n}$ is isomorphic to the monoid C_{n+1} of all order-preserving and order-decreasing total transformations of $\{1, 2, \dots, n, n+1\}$. Their cardinality is the Catalan number C_{n+1}.*

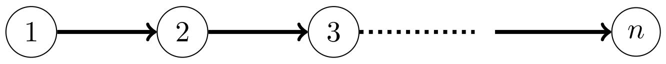


Figure 3.10: Line_n.

3.9 Further developments

A conceptual approach to the problem of determining the universal dynamics of a given graph Γ is by constructing an initial object, if there exists one, in the category of (pointed) update systems based on Γ; here "pointed" means that a preferred system state has been chosen.

Conjecture 6. The pair $(\mathcal{S}_n^\star, \star)$ is an initial object in the category of pointed update systems based on Γ_n.

The dynamics of an initial update system is clearly universal, and its elements are told apart by their action on the marked system state $\star$: this has been the philosophy underneath the proof of Theorem 2.6.2.

Bibliography

[AD13] Riccardo Aragona and Alessandro D'Andrea, *Hecke-Kiselman monoids of small cardinality*, Semigroup Forum **86** (2013), no. 1, 32–40. MR 3016259

[BLS91] Anders Björner, László Lovász, and Peter W. Shor, *Chip-firing games on graphs*, European J. Combin. **12** (1991), no. 4, 283–291. MR 1120415 (92g:90193)

[BMR00] 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), no. 2-3, 121–136. MR 1724827 (2000k:68170)

[BMR01] ———, *Elements of a theory of simulation. III. Equivalence of SDS*, Appl. Math. Comput. **122** (2001), no. 3, 325–340. MR 1842611 (2002e:68141)

[BMR03] ———, *ETS. IV. Sequential dynamical systems: fixed points, invertibility and equivalence*, Appl. Math. Comput. **134** (2003), no. 1, 153–171. MR 1928972 (2003i:68143)

[BR99] 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), no. 2-3, 241–259. MR 1660115 (99h:68181)

[CD13] E. Collina and A. D'Andrea, *Sequential Dynamical Systems on finite acyclic oriented graphs and Hecke-Kiselman monoids*, ArXiv:1311.3460 (2013).

[EE09] Henrik Eriksson and Kimmo Eriksson, *Conjugacy of Coxeter elements*, Electron. J. Combin. **16** (2009), no. 2, Special volume in honor of Anders Bjorner, Research Paper 4, 7. MR 2515767 (2010h:20083)

[For12] Love Forsberg, *Effective representations of hecke-kiselman monoids of type a*, ArXiv:1205.0676 (2012).

[GM11] Olexandr Ganyushkin and Volodymyr Mazorchuk, *On Kiselman quotients of 0-Hecke monoids*, Int. Electron. J. Algebra **10** (2011), 174–191. MR 2821178 (2012g:20121)

[GM13] Anna-Louise Grensing and Volodymyr Mazorchuk, *Monoid algebras of projection functors*, Semigroup Forum (2013).

[Gre51] J. A. Green, *On the structure of semigroups*, Ann. of Math. (2) **54** (1951), 163–172. MR 0042380 (13,100d)

[Gre12] Anna-Louise Grensing, *Monoid algebras of projection functors*, J. Algebra **369** (2012), 16–41. MR 2959784

[Kis02] Christer O. Kiselman, *A semigroup of operators in convexity theory*, Trans. Amer. Math. Soc. **354** (2002), no. 5, 2035–2053. MR 1881029 (2003a:52008)

[KM09] Ganna Kudryavtseva and Volodymyr Mazorchuk, *On Kiselman's semigroup*, Yokohama Math. J. **55** (2009), no. 1, 21–46. MR 2561084 (2010k:20097)

[Mat64] Hideya Matsumoto, *Générateurs et relations des groupes de Weyl généralisés*, C. R. Acad. Sci. Paris **258** (1964), 3419–3422. MR 0183818 (32 #1294)

[MM09] Matthew Macauley and Henning S. Mortveit, *Cycle equivalence of graph dynamical systems*, Nonlinearity **22** (2009), no. 2, 421–436. MR 2475554 (2010b:37042)

[MM10] ———, *Coxeter groups and asynchronous cellular automata*, CoRR **abs/1010.1955** (2010).

[MM11] ———, *Update sequence stability in graph dynamical systems*, Discrete Contin. Dyn. Syst. Ser. S **4** (2011), no. 6, 1533–1541. MR 2754172 (2012b:05275)

[MMM08] Matthew Macauley, Jon McCammond, and Henning S. Mortveit, *Order independence in asynchronous cellular automata*, J. Cell. Autom. **3** (2008), no. 1, 37–56. MR 2394603 (2009d:37022)

[MMM11] ———, *Dynamics groups of asynchronous cellular automata*, J. Algebraic Combin. **33** (2011), no. 1, 11–35. MR 2747798 (2012a:37024)

[MR08] Henning S. Mortveit and Christian M. Reidys, *An introduction to sequential dynamical systems*, Universitext, Springer, New York, 2008. MR 2357144 (2009d:37002)

[New42] M. H. A. Newman, *On theories with a combinatorial definition of "equivalence."*, Ann. of Math. (2) **43** (1942), 223–243. MR 0007372 (4,126c)

[NSP$^+$97] Kai Nagel, Paula Stretz, Martin Pieck, Shannon Leckey, Rick Donnelly, and Christopher L. Barrett, *Transims traffic flow characteristics*, 1997.

[Rei06] C.M. Reidys, *Sequential dynamical systems over words*, Annals of Combinatorics **10** (2006), no. 4, 481–498 (English).

[Tra14] *Transims - an open source transportation modeling and simulation toolbox*, 2014.

[Wol83] Stephen Wolfram, *Statistical mechanics of cellular automata*, Rev. Modern Phys. **55** (1983), no. 3, 601–644. MR 709077 (85d:68057)