On Synchronizing Automata

Wolfram Bentz

Centro de Álgebra, Universidade de Lisboa

Joint work with João Araújo (Centro de Álgebra, Universidade de Lisboa)
and
Peter J. Cameron (Queen Mary, University of London)

4th Novi Sad Algebraic Conference

Novi Sad, June 5, 2013

The princess in the dungeon

Imagine the following:

The princess in the dungeon

Imagine the following:

- You are a princess.

The princess in the dungeon

Imagine the following:

- You are a princess.
- You are trapped in a dungeon consisting of identical looking caves.

The princess in the dungeon

Imagine the following:

- You are a princess.
- You are trapped in a dungeon consisting of identical looking caves.
- Each cave has a red, blue, and yellow door.

The princess in the dungeon

Imagine the following:

- You are a princess.
- You are trapped in a dungeon consisting of identical looking caves.
- Each cave has a red, blue, and yellow door.
- Behind the red and blue doors are long winding red and blue colored one-way corridors leading to other caves.

The princess in the dungeon

Imagine the following:

- You are a princess.
- You are trapped in a dungeon consisting of identical looking caves.
- Each cave has a red, blue, and yellow door.
- Behind the red and blue doors are long winding red and blue colored one-way corridors leading to other caves.
- The yellow door in one of the caves leads to freedom, while opening any other yellow door leads to instant death.

The princess in the dungeon

Imagine the following:

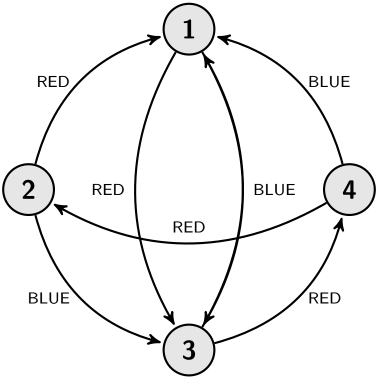
- You are a princess.
- You are trapped in a dungeon consisting of identical looking caves.
- Each cave has a red, blue, and yellow door.
- Behind the red and blue doors are long winding red and blue colored one-way corridors leading to other caves.
- The yellow door in one of the caves leads to freedom, while opening any other yellow door leads to instant death.
- You do not know which cave you are in.

The princess in the dungeon

Imagine the following:

- You are a princess.
- You are trapped in a dungeon consisting of identical looking caves.
- Each cave has a red, blue, and yellow door.
- Behind the red and blue doors are long winding red and blue colored one-way corridors leading to other caves.
- The yellow door in one of the caves leads to freedom, while opening any other yellow door leads to instant death.
- You do not know which cave you are in.
- You have a complete map of the dungeon.

The princess in the dungeon



The princess in the dungeon

The princess in the dungeon

- Assume you also know which cave contains the favorable yellow door.

The princess in the dungeon

- Assume you also know which cave contains the favorable yellow door.
- What is needed is a method that can guarantee that you end up in the same cave, no matter where you start.

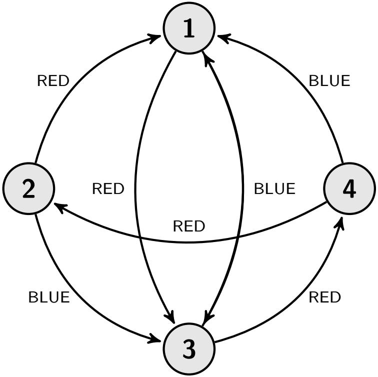
The princess in the dungeon

- Assume you also know which cave contains the favorable yellow door.
- What is needed is a method that can guarantee that you end up in the same cave, no matter where you start.
- So you want to run a route that will always end up in the same spot

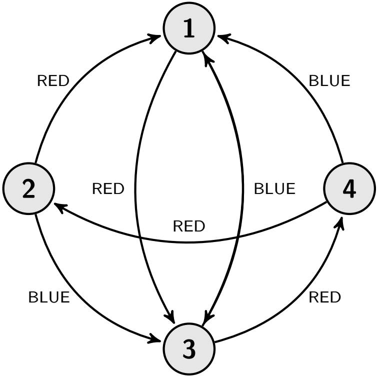
The princess in the dungeon

- Assume you also know which cave contains the favorable yellow door.
- What is needed is a method that can guarantee that you end up in the same cave, no matter where you start.
- So you want to run a route that will always end up in the same spot
- Once you know your location you can go wherever you want (if everything is connected).

The princess in the dungeon



The princess in the dungeon



The magic words are BLUE RED BLUE BLUE.

Error recovery

Error recovery

- Software (or other systems) tend to enter faulty states from time to time due to unforeseen circumstances.

Error recovery

- Software (or other systems) tend to enter faulty states from time to time due to unforeseen circumstances.
- The standard reaction to such errors can be considered versions of *backward recovery*.

Error recovery

- Software (or other systems) tend to enter faulty states from time to time due to unforeseen circumstances.
- The standard reaction to such errors can be considered versions of *backward recovery*.
- POWER OFF, REBOOT, RESTART FROM THE LAST SAVED CHECKPOINT.

Error recovery

- Software (or other systems) tend to enter faulty states from time to time due to unforeseen circumstances.
- The standard reaction to such errors can be considered versions of *backward recovery*.
- POWER OFF, REBOOT, RESTART FROM THE LAST SAVED CHECKPOINT.
- Such an option might not always exist or be ideal.

- Benenson, Paz-Elizur, Adar, Keinen, Livneh and Shapiro: Programmable and autonomous computing machine made of biomolecules. *Nature, 2011*

- Benenson, Paz-Elizur, Adar, Keinen, Livneh and Shapiro: Programmable and autonomous computing machine made of biomolecules. *Nature, 2011*
- No OFF-button!

- Benenson, Paz-Elizur, Adar, Keinen, Livneh and Shapiro: Programmable and autonomous computing machine made of biomolecules. *Nature, 2011*
- No OFF-button!
- Consider a satellite that has reemerged (say from behind Mars) and is in an unknown orientation.

- Benenson, Paz-Elizur, Adar, Keinen, Livneh and Shapiro: Programmable and autonomous computing machine made of biomolecules. *Nature, 2011*
- No OFF-button!
- Consider a satellite that has reemerged (say from behind Mars) and is in an unknown orientation.
- Not practical to go there, unplug, reorient and start-up again.

- Benenson, Paz-Elizur, Adar, Keinen, Livneh and Shapiro: Programmable and autonomous computing machine made of biomolecules. *Nature, 2011*
- No OFF-button!
- Consider a satellite that has reemerged (say from behind Mars) and is in an unknown orientation.
- Not practical to go there, unplug, reorient and start-up again.
- Consider production tools or other products dropped on a conveyor belt in unknown orientation.

- Benenson, Paz-Elizur, Adar, Keinen, Livneh and Shapiro: Programmable and autonomous computing machine made of biomolecules. *Nature, 2011*
- No OFF-button!
- Consider a satellite that has reemerged (say from behind Mars) and is in an unknown orientation.
- Not practical to go there, unplug, reorient and start-up again.
- Consider production tools or other products dropped on a conveyor belt in unknown orientation.
- Before automatic work can be performed on them, the need to be aligned correctly.

Forward Error Recovery

- An alternative to dealing with errors utilizes *forward recovery*.

Forward Error Recovery

- An alternative to dealing with errors utilizes *forward recovery*.
- A forward recovery mechanism is a sequence of procedures or instructions that brings the process to a known state *irrespectively of its current state*.

Forward Error Recovery

- An alternative to dealing with errors utilizes *forward recovery*.
- A forward recovery mechanism is a sequence of procedures or instructions that brings the process to a known state *irrespectively of its current state*.
- Just like the princess in the the dungeon.

Synchronizing automata

Consider an automaton with state set S and instruction set Σ (we do not need initial or terminal states)

Synchronizing automata

Consider an automaton with state set S and instruction set Σ (we do not need initial or terminal states)

A *synchronizing* or *reset word* is a sequence of instructions from Σ such that inputting the sequence into the automaton will put the automaton into the same state irrespectively of its initial state.

Synchronizing automata

Consider an automaton with state set S and instruction set Σ (we do not need initial or terminal states)

A *synchronizing* or *reset word* is a sequence of instructions from Σ such that inputting the sequence into the automaton will put the automaton into the same state irrespectively of its initial state.

An automaton is synchronizing if there is a synchronizing word for it.

Synchronizing automata

Consider an automaton with state set S and instruction set Σ (we do not need initial or terminal states)

A *synchronizing* or *reset word* is a sequence of instructions from Σ such that inputting the sequence into the automaton will put the automaton into the same state irrespectively of its initial state.

An automaton is synchronizing if there is a synchronizing word for it.

Not all automata are synchronizing.

Synchronizing automata

Consider an automaton with state set S and instruction set Σ (we do not need initial or terminal states)

A *synchronizing* or *reset word* is a sequence of instructions from Σ such that inputting the sequence into the automaton will put the automaton into the same state irrespectively of its initial state.

An automaton is synchronizing if there is a synchronizing word for it.

Not all automata are synchronizing.

If an automaton is synchronizing, it has infinitely many synchronizing words.

Questions

- Which automata are synchronizing?

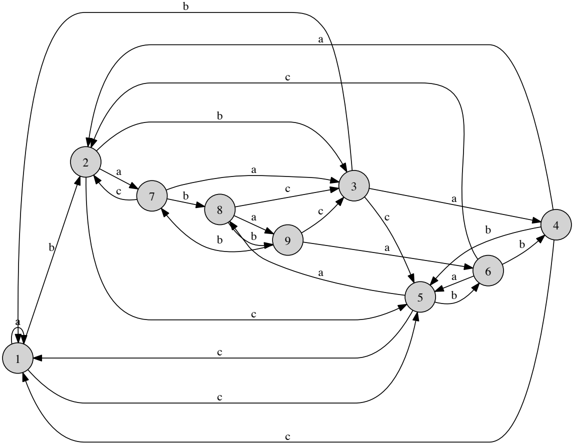
Questions

- Which automata are synchronizing?
- If an automaton is synchronizing, can we get a (good) bound on the length of its reset word?

Questions

- Which automata are synchronizing?
- If an automaton is synchronizing, can we get a (good) bound on the length of its reset word?
- **Černý Conjecture:** If an automaton is synchronizing and has n states, then it has a synchronizing word of at most $(n-1)^2$ letters.

Synchronizing?



Semigroup

- We can associate each label with an element on the full transformation semigroup on the set of states.

Semigroup

- We can associate each label with an element on the full transformation semigroup on the set of states.
- Our questions then translate into: for a given set S of transformations on a finite set,

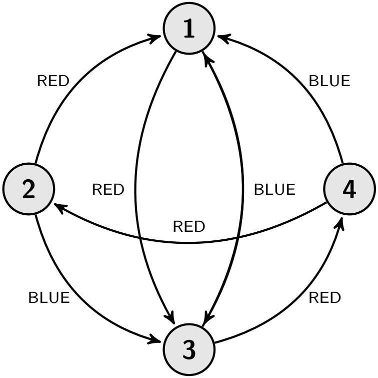
Semigroup

- We can associate each label with an element on the full transformation semigroup on the set of states.
- Our questions then translate into: for a given set S of transformations on a finite set,
  1. does the semigroup generated by S generate a constant map?

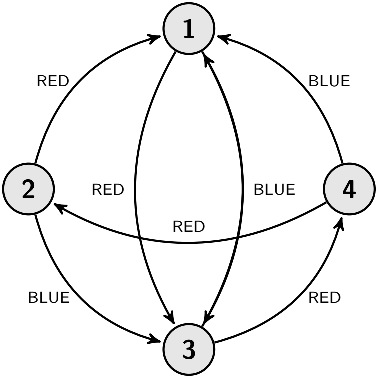
Semigroup

- We can associate each label with an element on the full transformation semigroup on the set of states.
- Our questions then translate into: for a given set S of transformations on a finite set,
  1. does the semigroup generated by S generate a constant map?
  2. if so, can we bound the length of a word, on the generators, that gives a constant map?

The princess in the dungeon



The princess in the dungeon



$$\text{BLUE} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 3 & 1 & 1 \end{pmatrix} \quad \text{RED} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 4 & 2 \end{pmatrix}.$$

The princess in the dungeon

- In our dungeon example there were four states $1, 2, 3, 4$ and we get:

$$\mathrm{BLUE} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 3 & 1 & 1 \end{pmatrix} \quad \mathrm{RED} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 4 & 2 \end{pmatrix}.$$

The princess in the dungeon

- In our dungeon example there were four states $1, 2, 3, 4$ and we get:

$$\text{BLUE} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 3 & 1 & 1 \end{pmatrix} \quad \text{RED} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 4 & 2 \end{pmatrix}.$$

- If we compose those function from left to right in the order BLUE RED BLUE BLUE, we get

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 3 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 4 & 2 \end{pmatrix}$$
$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 3 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 3 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 3 & 3 & 3 \end{pmatrix}.$$

Groups

Definition

Let S_n and T_n are, respectively, the symmetric group and the full transformation monoid on the set $X = \{1, \ldots, n\}$.
We say that a group $G \leq S_n$ *synchronizes* a transformation $t \in T_n \setminus S_n$ if the subsemigroup of T_n generated by $G \cup \{t\}$ contains a constant map.

Synchronizing Groups

Definition

A subgroup G of S_n is a *synchronizing group* if it synchronizes every non-permutation in T_n.

Synchronizing Groups

Definition

A subgroup G of S_n is a *synchronizing group* if it synchronizes every non-permutation in T_n.

"Pure" groups theoretic definition: A permutation group G on X is synchronizing if no (proper, non-trivial) partition of X has a section (or transversal) S that is invariant under G (i.e. for which S^g is also a section for all $g \in G$)

Results on synchronizing groups

- Synchronizing groups must be *primitive* ([Araújo 2006], [Arnold and Steinberg 2006], [Neumann 2009]), that is, there is no (non-trivial and proper) G-invariant partition of X.

Results on synchronizing groups

- Synchronizing groups must be *primitive* ([Araújo 2006], [Arnold and Steinberg 2006], [Neumann 2009]), that is, there is no (non-trivial and proper) G-invariant partition of X.
- If the group G is primitive, but not synchronizing, let t be a transformation not synchronized by G with minimal rank. Then t has *uniform kernel*, i.e. all kernel classes are of the same size ([Neumann 2009]).

Almost Synchronizing Groups

Definition (ABC)

A subgroup G of S_n is an *almost synchronizing group* if it synchronizes every transformation of non-uniform kernel in T_n.

Almost Synchronizing Groups

Definition (ABC)

A subgroup G of S_n is an *almost synchronizing group* if it synchronizes every transformation of non-uniform kernel in T_n.

Conjecture

With at most finitely many exceptions, every primitive group is almost synchronizing.

Almost Synchronizing Groups

Definition (ABC)

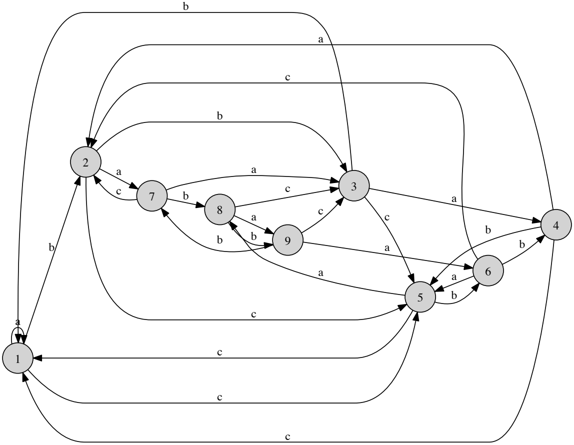
A subgroup G of S_n is an *almost synchronizing group* if it synchronizes every transformation of non-uniform kernel in T_n.

Conjecture

With at most finitely many exceptions, every primitive group is almost synchronizing.

We have developed two different methods to recognize almost synchronizing groups. Both give an infinite list of examples (with little overlap).

Shortly back to our example



- $$\begin{aligned} a &= \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 7 & 4 & 2 & 8 & 5 & 3 & 9 & 6 \end{pmatrix} \\ b &= \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 2 & 3 & 1 & 5 & 6 & 4 & 8 & 9 & 7 \end{pmatrix} \\ c &= \begin{pmatrix} \{1,2,3\} & \{4,5\} & \{6,7\} & \{8,9\} \\ 5 & 1 & 2 & 3 \end{pmatrix} \end{aligned}$$

- $$\begin{aligned} a &= \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 7 & 4 & 2 & 8 & 5 & 3 & 9 & 6 \end{pmatrix} \\ b &= \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 2 & 3 & 1 & 5 & 6 & 4 & 8 & 9 & 7 \end{pmatrix} \\ c &= \begin{pmatrix} \{1,2,3\} & \{4,5\} & \{6,7\} & \{8,9\} \\ 5 & 1 & 2 & 3 \end{pmatrix} \end{aligned}$$
- We have a non-uniform transformation.

- $$\begin{aligned} a &= \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 7 & 4 & 2 & 8 & 5 & 3 & 9 & 6 \end{pmatrix} \\ b &= \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 2 & 3 & 1 & 5 & 6 & 4 & 8 & 9 & 7 \end{pmatrix} \\ c &= \begin{pmatrix} \{1,2,3\} & \{4,5\} & \{6,7\} & \{8,9\} \\ 5 & 1 & 2 & 3 \end{pmatrix} \end{aligned}$$
- We have a non-uniform transformation.
- The two permutations can be checked to generate a primitive group.

- $$
\begin{aligned}
a &= \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 7 & 4 & 2 & 8 & 5 & 3 & 9 & 6 \end{pmatrix} \\
b &= \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 2 & 3 & 1 & 5 & 6 & 4 & 8 & 9 & 7 \end{pmatrix} \\
c &= \begin{pmatrix} \{1,2,3\} & \{4,5\} & \{6,7\} & \{8,9\} \\ 5 & 1 & 2 & 3 \end{pmatrix}
\end{aligned}
$$
- We have a non-uniform transformation.
- The two permutations can be checked to generate a primitive group.
- If our conjecture is right, we have a synchronizing word.

A hierarchy of non-synchronizing groups

Assume that G is a primitive non-synchronizing group acting on X, and that $S \subset X$ with $1 < |S| < |X|$.

A hierarchy of non-synchronizing groups

Assume that G is a primitive non-synchronizing group acting on X, and that $S \subset X$ with $1 < |S| < |X|$.

- Let $\overline{\Gamma}_S$ be the graph on X where $\{x, y\}$ is an edge (for $x \neq y$) if and only if there is no element $g \in G$ with $\{x, y\}g \subseteq S$.

A hierarchy of non-synchronizing groups

Assume that G is a primitive non-synchronizing group acting on X, and that $S \subset X$ with $1 < |S| < |X|$.

- Let $\overline{\Gamma}_S$ be the graph on X where $\{x, y\}$ is an edge (for $x \neq y$) if and only if there is no element $g \in G$ with $\{x, y\}g \subseteq S$.
- The closed neighborhood of x is denoted by $\overline{\Gamma}_S[x] := \{x\} \cup \{y \mid x \sim y\}$ for each $x \in X$, where $x \sim y$ if there is an edge from x to y in $\overline{\Gamma}_S$.

A hierarchy of non-synchronizing groups

Assume that G is a primitive non-synchronizing group acting on X, and that $S \subset X$ with $1 < |S| < |X|$.

- Let $\overline{\Gamma}_S$ be the graph on X where $\{x, y\}$ is an edge (for $x \neq y$) if and only if there is no element $g \in G$ with $\{x, y\}g \subseteq S$.
- The closed neighborhood of x is denoted by $\overline{\Gamma}_S[x] := \{x\} \cup \{y \mid x \sim y\}$ for each $x \in X$, where $x \sim y$ if there is an edge from x to y in $\overline{\Gamma}_S$.
- Let $T \subset X$ be a clique of $\overline{\Gamma}_S$ and define $m(S, T)$ to be the smallest number m such that the intersection of the closed neighborhoods of *any* m points of T in $\overline{\Gamma}_S$ is equal to T.

A hierarchy of non-synchronizing groups

Assume that G is a primitive non-synchronizing group acting on X, and that $S \subset X$ with $1 < |S| < |X|$.

- Let $\overline{\Gamma}_S$ be the graph on X where $\{x, y\}$ is an edge (for $x \neq y$) if and only if there is no element $g \in G$ with $\{x, y\}g \subseteq S$.
- The closed neighborhood of x is denoted by $\overline{\Gamma}_S[x] := \{x\} \cup \{y \mid x \sim y\}$ for each $x \in X$, where $x \sim y$ if there is an edge from x to y in $\overline{\Gamma}_S$.
- Let $T \subset X$ be a clique of $\overline{\Gamma}_S$ and define $m(S, T)$ to be the smallest number m such that the intersection of the closed neighborhoods of *any* m points of T in $\overline{\Gamma}_S$ is equal to T.
- Finally, consider all uniform transformations t that are not synchronized by G. Let $m(G)$ be the maximal $m(S, T)$ where S is the image of t and T a block in the kernel of t.

A hierarchy of non-synchronizing groups

Assume that G is a primitive non-synchronizing group acting on X, and that $S \subset X$ with $1 < |S| < |X|$.

- Let $\overline{\Gamma}_S$ be the graph on X where $\{x, y\}$ is an edge (for $x \neq y$) if and only if there is no element $g \in G$ with $\{x, y\}g \subseteq S$.
- The closed neighborhood of x is denoted by $\overline{\Gamma}_S[x] := \{x\} \cup \{y \mid x \sim y\}$ for each $x \in X$, where $x \sim y$ if there is an edge from x to y in $\overline{\Gamma}_S$.
- Let $T \subset X$ be a clique of $\overline{\Gamma}_S$ and define $m(S, T)$ to be the smallest number m such that the intersection of the closed neighborhoods of *any* m points of T in $\overline{\Gamma}_S$ is equal to T.
- Finally, consider all uniform transformations t that are not synchronized by G. Let $m(G)$ be the maximal $m(S, T)$ where S is the image of t and T a block in the kernel of t.
- $m(G)$ is the *non-synchronizing parameter* of G.

One more parameter

Assume that G is a primitive non-synchronizing group acting on X.

- Consider two transformations t_1, t_2 that are not synchronized by G, of the same rank k, and with distinct kernels P and P'

One more parameter

Assume that G is a primitive non-synchronizing group acting on X.

- Consider two transformations t_1, t_2 that are not synchronized by G, of the same rank k, and with distinct kernels P and P'
- Let $M(P, P') = \frac{|P \cap P'|}{k}$.

One more parameter

Assume that G is a primitive non-synchronizing group acting on X.

- Consider two transformations t_1, t_2 that are not synchronized by G, of the same rank k, and with distinct kernels P and P'
- Let $M(P, P') = \frac{|P \cap P'|}{k}$.
- Set $M(G)$ to be the maximum over all such pairs.

Main Theorem

Theorem

Let $G \leq S_n$ be a non-synchronizing group with $m(G) = 2$ and $M(G) \leq 1/2$. If $t \in T_n$ is a transformation such that $Ker(t)$ is a non-uniform partition, then $\langle G, t \rangle$ contains a constant function; that is, G is almost synchronizing.

Main Theorem

Theorem

Let $G \leq S_n$ be a non-synchronizing group with $m(G) = 2$ and $M(G) \leq 1/2$. If $t \in T_n$ is a transformation such that $Ker(t)$ is a non-uniform partition, then $\langle G, t\rangle$ contains a constant function; that is, G is almost synchronizing.

We do not know any non-synchronizing G with $m(G) = 2$ and $M(G) > 1/2$.

Almost non-synchronizing

$$a = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 7 & 4 & 2 & 8 & 5 & 3 & 9 & 6 \end{pmatrix}$$
$$b = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 2 & 3 & 1 & 5 & 6 & 4 & 8 & 9 & 7 \end{pmatrix}$$

Another method to obtain non-synchronizing groups

Another method to obtain non-synchronizing groups

- A primitive permutation group G with the property that the only G-invariant graphs X with coinciding clique and chromatic numbers are pseudo cores, is almost synchronizing.

Another method to obtain non-synchronizing groups

- A primitive permutation group G with the property that the only G-invariant graphs X with coinciding clique and chromatic numbers are pseudo cores, is almost synchronizing.
- *Pseudo cores*: every endomorphism of X is either an automorphism or a colouring.

Another method to obtain non-synchronizing groups

- A primitive permutation group G with the property that the only G-invariant graphs X with coinciding clique and chromatic numbers are pseudo cores, is almost synchronizing.
- *Pseudo cores*: every endomorphism of X is either an automorphism or a colouring.
- We also obtained infinite families of examples for this method.

Examples of almost-synchronizing groups

- The symmetric group S_m acting on 2-sets for m even, $m \geq 6$.
- G a subgroup of $\mathrm{P\Gamma L}(n, q)$ containing $\mathrm{PSL}(n, q)$ with $n \geq 5$, acting on the lines of the projective space.
- $G$the group $\mathrm{P\Gamma L}(3, q) \cdot 2$ or a subgroup containing $\mathrm{PSL}(3, q) \cdot 2$, where 2 denotes the *inverse transpose automorphism*, acting on the set of incident point-line pairs in the projective plane (where the outer automorphism induces a duality of the plane interchanging points and lines).

- Is it true that $m(G) = 2$ implies $M(G) \leq \frac{1}{2}$?

- Is it true that $m(G) = 2$ implies $M(G) \leq \frac{1}{2}$?
- Is it possible to prove the main Theorem of the first method without any assumption about $M(G)$?

- Is it true that $m(G) = 2$ implies $M(G) \leq \frac{1}{2}$?
- Is it possible to prove the main Theorem of the first method without any assumption about $M(G)$?
- Classify the primitive non-synchronizing groups G such that $m(G) = 2$.

- Is it true that $m(G) = 2$ implies $M(G) \leq \frac{1}{2}$?
- Is it possible to prove the main Theorem of the first method without any assumption about $M(G)$?
- Classify the primitive non-synchronizing groups G such that $m(G) = 2$.
- For each $k \geq 2$, classify the primitive almost synchronizing groups such that $m(G) = k$.

- Is it true that $m(G) = 2$ implies $M(G) \leq \frac{1}{2}$?
- Is it possible to prove the main Theorem of the first method without any assumption about $M(G)$?
- Classify the primitive non-synchronizing groups G such that $m(G) = 2$.
- For each $k \geq 2$, classify the primitive almost synchronizing groups such that $m(G) = k$.
- Is it true that primitive groups are almost synchronizing?

J. Araújo, A group theoretical approach to synchronizing automata and the Černy problem.Unpublished manuscript, 2006.

F. Arnold and B. Steinberg, Synchronizing groups and automata. *Theoret. Comput. Sci.* **359** (2006), no. 1-3, 101–110.

Y. Benenson, T. Paz-Elizur, R. Adar, E. Keinan, Z. Livneh, and E. Shapiro(2001). Programmable and autonomous computing machine made of biomolecules. *Nature 404(6862)*, 430–434.

P. M. Neumann, Primitive permutation groups and their section-regular partitions. *Michigan Math. J.* **58** (2009), 309–322.