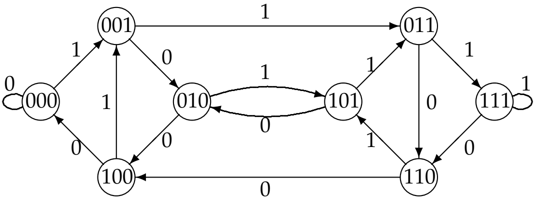
Finding where you are: Automata, graph endomorphisms, and de Bruijn graphs

Peter J. Cameron
University of St Andrews

NBSAN, St Andrews, 24 April 2015



Introduction

I have recently been working on two different problems in which synchronizing automata make an appearance:

Introduction

I have recently been working on two different problems in which synchronizing automata make an appearance:

- the disproof of a conjecture about the relationship between synchronization and primitivity for permutation groups (with João Araújo, Wolfram Bentz, Gordon Royle and Artur Schaefer);

Introduction

I have recently been working on two different problems in which synchronizing automata make an appearance:

- the disproof of a conjecture about the relationship between synchronization and primitivity for permutation groups (with João Araújo, Wolfram Bentz, Gordon Royle and Artur Schaefer);
- the outer automorphism groups of the Higman–Thompson groups, which are also the automorphism groups of full one-sided shifts, and are related to foldings of de Bruijn graphs (with Collin Bleak).

Introduction

I have recently been working on two different problems in which synchronizing automata make an appearance:

- the disproof of a conjecture about the relationship between synchronization and primitivity for permutation groups (with João Araújo, Wolfram Bentz, Gordon Royle and Artur Schaefer);
- the outer automorphism groups of the Higman–Thompson groups, which are also the automorphism groups of full one-sided shifts, and are related to foldings of de Bruijn graphs (with Collin Bleak).

I will give a brief introduction, and then talk about some work on each of these two problems. (Time does not permit a complete survey of either!)

Automata

An automaton is a black box with a finite number of internal states. If a symbol from an alphabet is input, it undergoes a state transition.

Automata

An automaton is a black box with a finite number of internal states. If a symbol from an alphabet is input, it undergoes a state transition.

Our automata are very simple: they don't have accept states, and they don't recognise languages; they don't even have a start state, you can start anywhere; and they are deterministic.

Automata

An automaton is a black box with a finite number of internal states. If a symbol from an alphabet is input, it undergoes a state transition.

Our automata are very simple: they don't have accept states, and they don't recognise languages; they don't even have a start state, you can start anywhere; and they are deterministic.

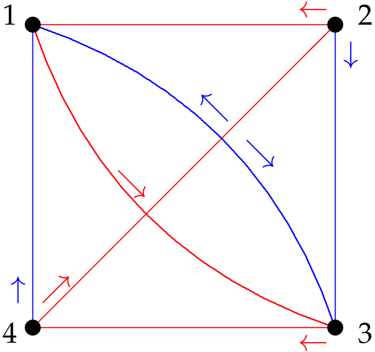
An automaton can be represented combinatorially by a directed graph (whose vertices are the states) with edges labelled by symbols of the alphabet, so that there is exactly one edge with each label *leaving* each vertex.

An example

Here is a 4-state automaton over an alphabet of two symbols, Red and Blue.

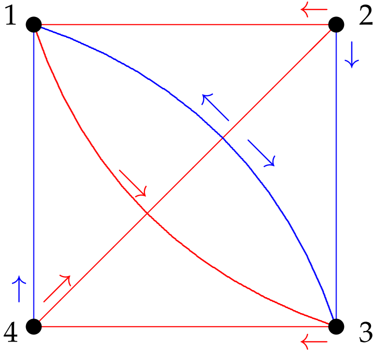
An example

Here is a 4-state automaton over an alphabet of two symbols, Red and Blue.



An example

Here is a 4-state automaton over an alphabet of two symbols, Red and Blue.



You can check that (Blue, Red, Blue) takes you to room 1, no matter where you start. So, if this were the map of a dungeon in which you were lost, you could use the map to find your way to the exit.

Synchronization

An automaton is said to be synchronizing if there is a sequence of inputs which brings it to a known state, regardless of its starting point. Such a sequence is called a reset word.

Synchronization

An automaton is said to be synchronizing if there is a sequence of inputs which brings it to a known state, regardless of its starting point. Such a sequence is called a reset word.

The example on the preceding slide had a reset word of length 3.

Synchronization

An automaton is said to be synchronizing if there is a sequence of inputs which brings it to a known state, regardless of its starting point. Such a sequence is called a reset word.

The example on the preceding slide had a reset word of length 3.

The main open problem about synchronizing automata is the Černý conjecture, which states that, if an n-state automaton is synchronizing, then it has a reset word of length at most $(n-1)^2$. If true, this would be best possible. But I am not going to talk about this, although what I am going to say was motivated by the conjecture.

Complexity of synchronization

If you are lost in a dungeon and have a map, you need to decide whether the automaton is synchronizing. There is good and bad news.

Complexity of synchronization

If you are lost in a dungeon and have a map, you need to decide whether the automaton is synchronizing. There is good and bad news.

The good news is that there is a polynomial-time algorithm to decide the question. This depends on the following fact:

Complexity of synchronization

If you are lost in a dungeon and have a map, you need to decide whether the automaton is synchronizing. There is good and bad news.
The good news is that there is a polynomial-time algorithm to decide the question. This depends on the following fact:

Proposition

An automaton is synchronizing if and only if, for any pair of states, there is a sequence of transitions which maps them to the same place.

Complexity of synchronization

If you are lost in a dungeon and have a map, you need to decide whether the automaton is synchronizing. There is good and bad news.
The good news is that there is a polynomial-time algorithm to decide the question. This depends on the following fact:

Proposition

An automaton is synchronizing if and only if, for any pair of states, there is a sequence of transitions which maps them to the same place.

By collapsing pairs $n-1$ times we reach a synchronizing word. Indeed, this gives us an cubic upper bound for the length of such a word: $n-1$ iterations of a process which takes $O(n^2)$ steps.

Complexity of synchronization

If you are lost in a dungeon and have a map, you need to decide whether the automaton is synchronizing. There is good and bad news.
The good news is that there is a polynomial-time algorithm to decide the question. This depends on the following fact:

Proposition

An automaton is synchronizing if and only if, for any pair of states, there is a sequence of transitions which maps them to the same place.

By collapsing pairs $n-1$ times we reach a synchronizing word.
Indeed, this gives us an cubic upper bound for the length of such a word: $n-1$ iterations of a process which takes $O(n^2)$ steps.
The bad news is that finding the shortest synchronizing word is NP-hard.

Automata and monoids

Algebraically, a transition is a mapping on the set of states; we are allowed to compose transitions; so an automaton is a submonoid of the full transformation monoid on a set of size n *with a distinguished set of generators*.

Automata and monoids

Algebraically, a transition is a mapping on the set of states; we are allowed to compose transitions; so an automaton is a submonoid of the full transformation monoid on a set of size n *with a distinguished set of generators*.
In this language, an automaton is synchronizing if the monoid contains a constant function (a map of rank 1).

Automata and monoids

Algebraically, a transition is a mapping on the set of states; we are allowed to compose transitions; so an automaton is a submonoid of the full transformation monoid on a set of size n *with a distinguished set of generators*.

In this language, an automaton is synchronizing if the monoid contains a constant function (a map of rank 1).

Our next task is to describe the maximal non-synchronizing automata (or monoids).

Graph homorphisms

Graphs in the next part are simple: undirected, without loops or multiple edges.

Graph homomorphisms

Graphs in the next part are simple: undirected, without loops or multiple edges.

A homomorphism from a graph Γ to a graph Δ is a map f from the vertex set of Γ to that of Δ which maps edges to edges. It can map non-edges to non-edges, or to edges, or collapse them to vertices.

Graph homomorphisms

Graphs in the next part are simple: undirected, without loops or multiple edges.

A homomorphism from a graph Γ to a graph Δ is a map f from the vertex set of Γ to that of Δ which maps edges to edges. It can map non-edges to non-edges, or to edges, or collapse them to vertices.

- A homomorphism from the complete graph K_r to Γ is an embedding of K_r in Γ; the largest such r is the clique number of Γ.

Graph homorphisms

Graphs in the next part are simple: undirected, without loops or multiple edges.

A homomorphism from a graph Γ to a graph Δ is a map f from the vertex set of Γ to that of Δ which maps edges to edges. It can map non-edges to non-edges, or to edges, or collapse them to vertices.

- A homomorphism from the complete graph K_r to Γ is an embedding of K_r in Γ; the largest such r is the clique number of Γ.
- A homomorphism from Γ to K_r is a proper vertex-colouring of Γ with r colours; the smallest such r is the chromatic number of Γ.

Graph homorphisms

Graphs in the next part are simple: undirected, without loops or multiple edges.
A homomorphism from a graph Γ to a graph Δ is a map f from the vertex set of Γ to that of Δ which maps edges to edges. It can map non-edges to non-edges, or to edges, or collapse them to vertices.

- A homomorphism from the complete graph K_r to Γ is an embedding of K_r in Γ; the largest such r is the clique number of Γ.
- A homomorphism from Γ to K_r is a proper vertex-colouring of Γ with r colours; the smallest such r is the chromatic number of Γ.

An endomorphism of Γ is a homomorphism from Γ to itself. The endomorphisms form a monoid $\mathrm{End}(\Gamma)$ (the endomorphism monoid of Γ).

Synchronization and graph endomorphisms

Theorem

A transformation monoid M on Ω is non-synchronizing if and only if there is a non-null graph Γ on the vertex set Ω such that $M \leq \mathrm{End}(\Gamma)$. We may assume that the clique number and chromatic number of Γ are equal.

One direction is trivial: if the clique number and chromatic number are equal, then Γ has non-trivial endomorphisms f, which are not synchronized by G.

Synchronization and graph endomorphisms

Theorem

A transformation monoid M on Ω is non-synchronizing if and only if there is a non-null graph Γ on the vertex set Ω such that $M \leq \mathrm{End}(\Gamma)$. We may assume that the clique number and chromatic number of Γ are equal.

One direction is trivial: if the clique number and chromatic number are equal, then Γ has non-trivial endomorphisms f, which are not synchronized by G.

I won't prove the other direction; it is not difficult, but it is the foundation of our study of synchronizing automata, and is the basis of the most efficient test of synchronization.

Synchronizing groups

There are very few examples of synchronizing automata attaining the Černý bound. All of them have the property that one generator is a cyclic permutation, so the monoid contains a transitive permutation group as a subgroup. This motivates looking at monoids generated by a group and one further element.

Synchronizing groups

There are very few examples of synchronizing automata attaining the Černý bound. All of them have the property that one generator is a cyclic permutation, so the monoid contains a transitive permutation group as a subgroup. This motivates looking at monoids generated by a group and one further element.

By abuse of language, we describe a permutation group G acting on Ω as synchronizing if, for every non-permutation f, the monoid $\langle G, f \rangle$ is synchronizing.

Synchronizing groups

There are very few examples of synchronizing automata attaining the Černý bound. All of them have the property that one generator is a cyclic permutation, so the monoid contains a transitive permutation group as a subgroup. This motivates looking at monoids generated by a group and one further element.

By abuse of language, we describe a permutation group G acting on Ω as synchronizing if, for every non-permutation f, the monoid $\langle G, f \rangle$ is synchronizing.

Thus, G is synchronizing if and only if it is not contained in the automorphism group of a graph with clique number equal to chromatic number.

Synchronizing groups

There are very few examples of synchronizing automata attaining the Černý bound. All of them have the property that one generator is a cyclic permutation, so the monoid contains a transitive permutation group as a subgroup. This motivates looking at monoids generated by a group and one further element.

By abuse of language, we describe a permutation group G acting on Ω as synchronizing if, for every non-permutation f, the monoid $\langle G, f \rangle$ is synchronizing.

Thus, G is synchronizing if and only if it is not contained in the automorphism group of a graph with clique number equal to chromatic number.

The hope is that we can use knowledge of permutation groups (acquired since the Classification of Finite Simple Groups) to understand synchronizing groups.

Synchronization and primitivity

A permutation group is primitive if it preserves no non-trivial partition of Ω. (The trivial partitions are the one with a single part and the one with parts of size 1.)

Synchronization and primitivity

A permutation group is primitive if it preserves no non-trivial partition of Ω. (The trivial partitions are the one with a single part and the one with parts of size 1.)

Theorem

A synchronizing group is primitive.

Synchronization and primitivity

A permutation group is primitive if it preserves no non-trivial partition of Ω. (The trivial partitions are the one with a single part and the one with parts of size 1.)

Theorem

A synchronizing group is primitive.

For an imprimitive group preserves two graphs derived from its invariant partition (the disjoint union of complete graphs on the parts, and its complement, the complete multipartite graphs) which have non-trivial endomorphisms.

Synchronization and primitivity

A permutation group is primitive if it preserves no non-trivial partition of Ω. (The trivial partitions are the one with a single part and the one with parts of size 1.)

Theorem

A synchronizing group is primitive.

For an imprimitive group preserves two graphs derived from its invariant partition (the disjoint union of complete graphs on the parts, and its complement, the complete multipartite graphs) which have non-trivial endomorphisms.
The converse is false, but it is thought that the difference between primitivity and synchronization is not all that great.

A conjecture

If G is primitive but not synchronizing, then a minimum-rank map which is not synchronized is a colouring of a G-invariant graph, and so is uniform: all kernel classes have the same size.

A conjecture

If G is primitive but not synchronizing, then a minimum-rank map which is not synchronized is a colouring of a G-invariant graph, and so is uniform: all kernel classes have the same size. João Araújo conjectured that a primitive group synchronizes all non-uniform maps.

A conjecture

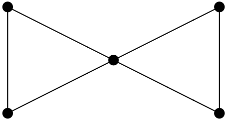
If G is primitive but not synchronizing, then a minimum-rank map which is not synchronized is a colouring of a G-invariant graph, and so is uniform: all kernel classes have the same size.
João Araújo conjectured that a primitive group synchronizes all non-uniform maps.
This conjecture has just been refuted.

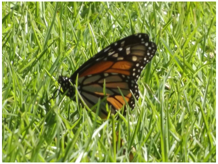
The conjecture bites the dust

Our first example was found in the following way. Suppose we could find a graph with clique number and chromatic number 3, which has a primitive automorphism group, and has a homomorphism onto the butterfly:

The conjecture bites the dust

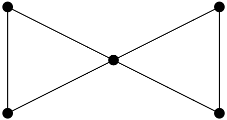
Our first example was found in the following way. Suppose we could find a graph with clique number and chromatic number 3, which has a primitive automorphism group, and has a homomorphism onto the butterfly:

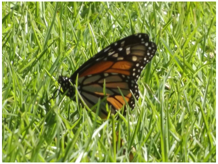




The conjecture bites the dust

Our first example was found in the following way. Suppose we could find a graph with clique number and chromatic number 3, which has a primitive automorphism group, and has a homomorphism onto the butterfly:

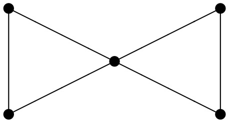


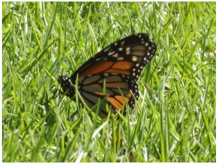


Suppose further that the graph has valency 4 and the closed neighbourhood of a vertex is itself a butterfly. Then it has a non-uniform endomorphism of rank 5.

The conjecture bites the dust

Our first example was found in the following way. Suppose we could find a graph with clique number and chromatic number 3, which has a primitive automorphism group, and has a homomorphism onto the butterfly:





Suppose further that the graph has valency 4 and the closed neighbourhood of a vertex is itself a butterfly. Then it has a non-uniform endomorphism of rank 5.
Fortunately, group theorists have determined such graphs. There are two, with 45 and 153 vertices: the line graphs of the Tutte–Coxeter and Biggs–Smith graphs.

A general construction

These graphs have endomorphisms of ranks 3, 5, and 7.
Another construction is much more prolific.

A general construction

These graphs have endomorphisms of ranks 3, 5, and 7.
Another construction is much more prolific.
This uses the Cartesian product $\Gamma \mathbin{\square} \Delta$ of graphs, whose vertex set is the Cartesian product of the vertex sets, two pairs being joined in the product if they are equal in one coordinate and joined in the other.

A general construction

These graphs have endomorphisms of ranks 3, 5, and 7.

Another construction is much more prolific.

This uses the Cartesian product $\Gamma \mathbin{\Box} \Delta$ of graphs, whose vertex set is the Cartesian product of the vertex sets, two pairs being joined in the product if they are equal in one coordinate and joined in the other.

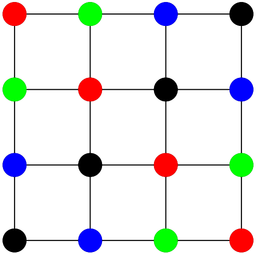
If Γ has primitive automorphism group G, then $\Gamma \mathbin{\Box} \Gamma$ has primitive automorphism group $G \operatorname{wr} C_2$.

An example

The graph $K_k \square K_k$ is the square lattice graph $L_2(k)$, with two vertices joined if and only if they are in the same row or column. It has clique number k (rows and columns are maximum cliques) and chromatic number k (a k-colouring is a Latin square):

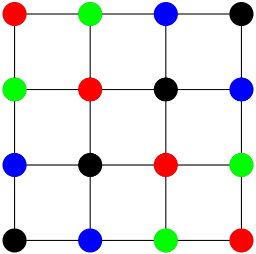
An example

The graph $K_k \,\square\, K_k$ is the square lattice graph $L_2(k)$, with two vertices joined if and only if they are in the same row or column. It has clique number k (rows and columns are maximum cliques) and chromatic number k (a k-colouring is a Latin square):



An example

The graph $K_k \square K_k$ is the square lattice graph $L_2(k)$, with two vertices joined if and only if they are in the same row or column. It has clique number k (rows and columns are maximum cliques) and chromatic number k (a k-colouring is a Latin square):



The complementary graph also has clique number and chromatic number k: a clique is a transversal, and the row numbers give a proper colouring.

A prolific construction

Suppose that Γ is a graph with clique number and chromatic number r. Then there is a homomorphism from Γ to K_k, and hence one from $\Gamma \square \Gamma$ to $L_2(k)$.

A prolific construction

Suppose that Γ is a graph with clique number and chromatic number r. Then there is a homomorphism from Γ to K_k, and hence one from $\Gamma \,\Box\, \Gamma$ to $L_2(k)$.
Suppose that additionally there is a homomorphism from $L_2(k)$ to Γ. Then we can compose

$$\Gamma \,\Box\, \Gamma \to L_2(k) \to \Gamma \to \Gamma \,\Box\, \Gamma.$$

The rank is equal to the rank of the middle map.

A prolific construction

Suppose that Γ is a graph with clique number and chromatic number r. Then there is a homomorphism from Γ to K_k, and hence one from $\Gamma \,\Box\, \Gamma$ to $L_2(k)$.
Suppose that additionally there is a homomorphism from $L_2(k)$ to Γ. Then we can compose

$$\Gamma \,\Box\, \Gamma \to L_2(k) \to \Gamma \to \Gamma \,\Box\, \Gamma.$$

The rank is equal to the rank of the middle map.
For an example, we take Γ to be the complement of $L_2(k)$ (this is the categorical product $K_k \times K_k$). The required homomorphism is a pair (f, g) of functions from $K \times K$ to K, where $K = \{1, \ldots, k\}$.

A prolific construction

Suppose that Γ is a graph with clique number and chromatic number r. Then there is a homomorphism from Γ to K_k, and hence one from $\Gamma \square \Gamma$ to $L_2(k)$.
Suppose that additionally there is a homomorphism from $L_2(k)$ to Γ. Then we can compose

$$\Gamma \square \Gamma \to L_2(k) \to \Gamma \to \Gamma \square \Gamma.$$

The rank is equal to the rank of the middle map.
For an example, we take Γ to be the complement of $L_2(k)$ (this is the categorical product $K_k \times K_k$). The required homomorphism is a pair (f, g) of functions from $K \times K$ to K, where $K = \{1, \ldots, k\}$.
Each function must be a Latin square, since we require that $y \neq z$ implies $f(x, y) \neq f(y, z)$, and similarly for g; but there is no connection between the two squares.

The possible ranks are the numbers of pairs of entries when two Latin squares of order k are superimposed.

The possible ranks are the numbers of pairs of entries when two Latin squares of order k are superimposed.
Colbourn, Zhu and Zhang have showed that, if $k > 6$, then any number between k and k^2 except for $k+1$ and k^2-1 can occur as the number of distinct entries when two Latin squares of order k are superimposed.

The possible ranks are the numbers of pairs of entries when two Latin squares of order k are superimposed.
Colbourn, Zhu and Zhang have showed that, if $k > 6$, then any number between k and k^2 except for $k + 1$ and $k^2 - 1$ can occur as the number of distinct entries when two Latin squares of order k are superimposed.
So we have a primitive group of degree k^4 which fails to synchronize maps of every possible rank between k and k^2 inclusive except for $k + 1$ and $k^2 - 1$.

The possible ranks are the numbers of pairs of entries when two Latin squares of order k are superimposed.
Colbourn, Zhu and Zhang have showed that, if $k > 6$, then any number between k and k^2 except for $k+1$ and k^2-1 can occur as the number of distinct entries when two Latin squares of order k are superimposed.
So we have a primitive group of degree k^4 which fails to synchronize maps of every possible rank between k and k^2 inclusive except for $k+1$ and k^2-1.
Many more examples can no doubt be found …

Another conjecture

We saw that, if G is imprimitive, preserving a relation with m equivalence classes of size k, then G preserves the disjoint union of m copies of K_k, and also its complete bipartite complement. These graphs have endomorphisms whose ranks are all multiples of k (for mK_k) and all integers between m and $mk = n$ (for the complete multipartite graph). This gives $(3/4 - o(1))n$ ranks of maps not synchronized by G.

Another conjecture

We saw that, if G is imprimitive, preserving a relation with m equivalence classes of size k, then G preserves the disjoint union of m copies of K_k, and also its complete bipartite complement. These graphs have endomorphisms whose ranks are all multiples of k (for mK_k) and all integers between m and $mk = n$ (for the complete multipartite graph). This gives $(3/4 - o(1))n$ ranks of maps not synchronized by G.
We conjecture that, if G is primitive, there are only $o(n)$ ranks of maps which it does not synchronize.

Another conjecture

We saw that, if G is imprimitive, preserving a relation with m equivalence classes of size k, then G preserves the disjoint union of m copies of K_k, and also its complete bipartite complement. These graphs have endomorphisms whose ranks are all multiples of k (for mK_k) and all integers between m and $mk = n$ (for the complete multipartite graph). This gives $(3/4 - o(1))n$ ranks of maps not synchronized by G.

We conjecture that, if G is primitive, there are only $o(n)$ ranks of maps which it does not synchronize.

For our example, this number is about $\sqrt{n}$.

Some groups

I now turn to the second part of the talk.

Some groups

I now turn to the second part of the talk.

The Higman–Thompson groups are a family $(G_{n,r})$ of finitely presented infinite simple groups. They have several definitions, in terms of homeomorphisms of Cantor space, or manipulations of trees, or as automorphism groups of free algebras in certain varieties.

Some groups

I now turn to the second part of the talk.

The Higman–Thompson groups are a family $(G_{n,r})$ of finitely presented infinite simple groups. They have several definitions, in terms of homeomorphisms of Cantor space, or manipulations of trees, or as automorphism groups of free algebras in certain varieties.

Bleak, Maissel and Navas investigated the outer automorphism group of $G_{n,r}$. They showed several things:

Some groups

I now turn to the second part of the talk.
The Higman–Thompson groups are a family $(G_{n,r})$ of finitely presented infinite simple groups. They have several definitions, in terms of homeomorphisms of Cantor space, or manipulations of trees, or as automorphism groups of free algebras in certain varieties.
Bleak, Maissel and Navas investigated the outer automorphism group of $G_{n,r}$. They showed several things:

- $\mathrm{Out}(G_{n,r})$ is independent of r;

Some groups

I now turn to the second part of the talk.

The Higman–Thompson groups are a family $(G_{n,r})$ of finitely presented infinite simple groups. They have several definitions, in terms of homeomorphisms of Cantor space, or manipulations of trees, or as automorphism groups of free algebras in certain varieties.

Bleak, Maissel and Navas investigated the outer automorphism group of $G_{n,r}$. They showed several things:

- $\mathrm{Out}(G_{n,r})$ is independent of r;
- this group is a subgroup of the automorphism group of the full (two-sided) shift on an alphabet of size n;

Some groups

I now turn to the second part of the talk.

The Higman–Thompson groups are a family $(G_{n,r})$ of finitely presented infinite simple groups. They have several definitions, in terms of homeomorphisms of Cantor space, or manipulations of trees, or as automorphism groups of free algebras in certain varieties.

Bleak, Maissel and Navas investigated the outer automorphism group of $G_{n,r}$. They showed several things:

- $\mathrm{Out}(G_{n,r})$ is independent of r;
- this group is a subgroup of the automorphism group of the full (two-sided) shift on an alphabet of size n;
- its elements can be realised by transducers acting on strings over an n-letter alphabet.

Some groups

I now turn to the second part of the talk.

The Higman–Thompson groups are a family $(G_{n,r})$ of finitely presented infinite simple groups. They have several definitions, in terms of homeomorphisms of Cantor space, or manipulations of trees, or as automorphism groups of free algebras in certain varieties.

Bleak, Maissel and Navas investigated the outer automorphism group of $G_{n,r}$. They showed several things:

- $\mathrm{Out}(G_{n,r})$ is independent of r;
- this group is a subgroup of the automorphism group of the full (two-sided) shift on an alphabet of size n;
- its elements can be realised by transducers acting on strings over an n-letter alphabet.

A transducer is an automaton which writes as well as reading.

Finitely-determined automata

A transducer representing an invertible transformation must have the property that the “inverse” (reversing the writing and reading operations) is also a transducer. Thus it is constructed from a pair of automata with the same underlying graph.

Finitely-determined automata

A transducer representing an invertible transformation must have the property that the "inverse" (reversing the writing and reading operations) is also a transducer. Thus it is constructed from a pair of automata with the same underlying graph.

We say that an automaton is k-determined if every word of length k is a reset word for it. In other words, when it reads k symbols, the state it is in depends only on the symbols read, and not on the state it was in before reading them.

Finitely-determined automata

A transducer representing an invertible transformation must have the property that the "inverse" (reversing the writing and reading operations) is also a transducer. Thus it is constructed from a pair of automata with the same underlying graph.

We say that an automaton is *k-determined* if every word of length k is a reset word for it. In other words, when it reads k symbols, the state it is in depends only on the symbols read, and not on the state it was in before reading them.

The automata involved in automorphisms of the Higman–Thompson groups turn out to be finitely determined. So we need to examine these further.

De Bruijn graphs

The de Bruijn graph $\mathrm{DB}_{n,k}$ with word length k over an alphabet A of size n is defined as follows.

De Bruijn graphs

The de Bruijn graph $\mathrm{DB}_{n,k}$ with word length k over an alphabet A of size n is defined as follows.

- The vertex set consists of all the words of length k.

De Bruijn graphs

The de Bruijn graph $\mathrm{DB}_{n,k}$ with word length k over an alphabet A of size n is defined as follows.

- The vertex set consists of all the words of length k.
- There is an edge from $x_1x_2\ldots x_k$ to $x_2x_3\ldots x_{k+1}$, whose label is the $(k+1)$-tuple $x_1x_2\ldots x_kx_{k+1}$.

De Bruijn graphs

The de Bruijn graph $\mathrm{DB}_{n,k}$ with word length k over an alphabet A of size n is defined as follows.

- The vertex set consists of all the words of length k.
- There is an edge from $x_1x_2\dots x_k$ to $x_2x_3\dots x_{k+1}$, whose label is the $(k+1)$-tuple $x_1x_2\dots x_kx_{k+1}$.

Since we want single symbols as labels, we will use just x_{k+1} for this edge.

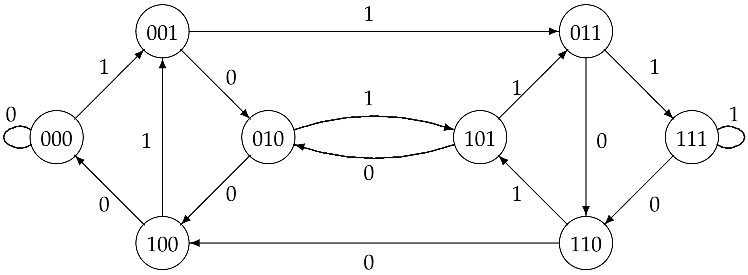
De Bruijn graphs

The de Bruijn graph $\mathrm{DB}_{n,k}$ with word length k over an alphabet A of size n is defined as follows.

- The vertex set consists of all the words of length k.
- There is an edge from $x_1x_2\ldots x_k$ to $x_2x_3\ldots x_{k+1}$, whose label is the $(k+1)$-tuple $x_1x_2\ldots x_kx_{k+1}$.

Since we want single symbols as labels, we will use just x_{k+1} for this edge.

Here is the de Bruijn graph $\mathrm{DB}_{2,3}$.



These graphs were originally used to constuct universal sequences. A de Bruijn sequence over A is a cyclic sequence of length n^{k+1} over A, with the property that each word of length $k+1$ occurs precisely once as a (consecutive) subsequence.

These graphs were originally used to constuct universal sequences. A de Bruijn sequence over A is a cyclic sequence of length n^{k+1} over A, with the property that each word of length $k+1$ occurs precisely once as a (consecutive) subsequence. The existence of de Bruijn sequences is immediate from the fact that the de Bruijn graph is strongly connected and has in-degree and out-degree equal, so is Eulerian; a Eulerian cycle gives the required sequence.

These graphs were originally used to constuct universal sequences. A de Bruijn sequence over A is a cyclic sequence of length n^{k+1} over A, with the property that each word of length $k+1$ occurs precisely once as a (consecutive) subsequence. The existence of de Bruijn sequences is immediate from the fact that the de Bruijn graph is strongly connected and has in-degree and out-degree equal, so is Eulerian; a Eulerian cycle gives the required sequence.
(An Eulerian cycle is a cycle passing once through each directed edge, in the correct direction; a directed graph has an Eulerian cycle if and only if it is strongly connected and has the in-degree of each vertex equal to its out-degree – an obvious necessary condition).

These graphs were originally used to constuct universal sequences. A de Bruijn sequence over A is a cyclic sequence of length n^{k+1} over A, with the property that each word of length $k+1$ occurs precisely once as a (consecutive) subsequence.
The existence of de Bruijn sequences is immediate from the fact that the de Bruijn graph is strongly connected and has in-degree and out-degree equal, so is Eulerian; a Eulerian cycle gives the required sequence.
(An Eulerian cycle is a cycle passing once through each directed edge, in the correct direction; a directed graph has an Eulerian cycle if and only if it is strongly connected and has the in-degree of each vertex equal to its out-degree – an obvious necessary condition).
For us the crucial property (from which the strong connectedness follows) is that, regarded as an automaton over the alphabet A, the de Bruijn graph $\mathrm{DB}_{n,k}$ is k-determined.

Foldings of de Bruijn graphs

A folding of $\mathrm{DB}_{n,k}$ is an equivalence relation $\equiv$ on the vertex set with the property that, if $v \equiv w$, then for any symbol a, the vertices obtained by moving along edges labelled a from v and w are also equivalent.

Foldings of de Bruijn graphs

A folding of $\mathrm{DB}_{n,k}$ is an equivalence relation $\equiv$ on the vertex set with the property that, if $v \equiv w$, then for any symbol a, the vertices obtained by moving along edges labelled a from v and w are also equivalent.

- The quotient of $\mathrm{DB}_{n,k}$ by a folding is a k-determined automaton.

Foldings of de Bruijn graphs

A folding of $\mathrm{DB}_{n,k}$ is an equivalence relation $\equiv$ on the vertex set with the property that, if $v \equiv w$, then for any symbol a, the vertices obtained by moving along edges labelled a from v and w are also equivalent.

- The quotient of $\mathrm{DB}_{n,k}$ by a folding is a k-determined automaton.
- Every k-determined automaton (in which every state is reachable) arises in this way.

Foldings of de Bruijn graphs

A folding of $\mathrm{DB}_{n,k}$ is an equivalence relation $\equiv$ on the vertex set with the property that, if $v \equiv w$, then for any symbol a, the vertices obtained by moving along edges labelled a from v and w are also equivalent.

- The quotient of $\mathrm{DB}_{n,k}$ by a folding is a k-determined automaton.
- Every k-determined automaton (in which every state is reachable) arises in this way.

So in order to study k-determined automata over A, we simply have to study foldings of $\mathrm{DB}_{n,k}$.

Counting foldings

"I count a lot of things that there's no need to count," Cameron said. "Just because thats the way I am. But I count all the things that need to be counted."

Richard Brautigan, The Hawkline Monster: A Gothic Western

Counting foldings

> *"I count a lot of things that there's no need to count," Cameron said. "Just because thats the way I am. But I count all the things that need to be counted."*
>
> *Richard Brautigan, The Hawkline Monster: A Gothic Western*

If we really understand foldings, we should be able to count them. Let $F(n,k)$ be the number of foldings of $\mathrm{DB}_{n,k}$.

Counting foldings

> *"I count a lot of things that there's no need to count," Cameron said. "Just because thats the way I am. But I count all the things that need to be counted."*
>
> *Richard Brautigan, The Hawkline Monster: A Gothic Western*

If we really understand foldings, we should be able to count them. Let $F(n,k)$ be the number of foldings of $\mathrm{DB}_{n,k}$. Trivially, $F(n,1)$ is the Bell number $B(n)$ (the number of partitions of the alphabet).

Counting foldings

> *"I count a lot of things that there's no need to count," Cameron said. "Just because thats the way I am. But I count all the things that need to be counted."*
>
> *Richard Brautigan, The Hawkline Monster: A Gothic Western*

If we really understand foldings, we should be able to count them. Let $F(n,k)$ be the number of foldings of $\mathrm{DB}_{n,k}$. Trivially, $F(n,1)$ is the Bell number $B(n)$ (the number of partitions of the alphabet).

$n \setminus k$	1	2	3	4
2	2	5	30	1247
3	5	192	?	?
4	15	78721	?	?
5	52	519338423	?	?

Word length 2

All but one of the results in the table were found by brute-force computation. However, we have found a formula for $F(n, 2)$:

Word length 2

All but one of the results in the table were found by brute-force computation. However, we have found a formula for $F(n, 2)$:

Theorem

Let

$$R(s,t) = \sum_{\pi} (-1)^{|\pi|-1} (|\pi|-1)! \prod_{i=1}^{|\pi|} B(a_i s),$$

where π runs over all partitions of $\{1, \ldots, t\}$, $|\pi|$ is the number of parts of π, and a_i is the size of the ith part. Then

$$F(n,2) = \sum_{\pi} \prod_{i=1}^{s} R(m, a_i),$$

where π runs over all partitions of the n-letter alphabet, m is the number of parts of π, and a_i is the cardinality of the ith part for $i = 1, \ldots, m$.

It looks complicated, but it allows the computation of $F(20, 2)$, a number of several hundred digits, in a second or so.

It looks complicated, but it allows the computation of $F(20, 2)$, a number of several hundred digits, in a second or so.
If you stare at the formula you will see the technique: Möbius inversion over the lattice of partitions of a set. But we haven't made it work for longer words yet!

It looks complicated, but it allows the computation of $F(20, 2)$, a number of several hundred digits, in a second or so.
If you stare at the formula you will see the technique: Möbius inversion over the lattice of partitions of a set. But we haven't made it work for longer words yet!
Möbius inversion is a general technique for arbitrary partially ordered sets which generalises the Inclusion-Exclusion principle (the case for the lattice of subsets of a set). I have known the form of the Möbius function for the partition lattice for many years, but never before now did I have a chance to use it seriously.

Counting automorphisms?

The obvious next step is to find formulae for the number of foldings of de Bruijn graphs with arbitrary word length.

Counting automorphisms?

The obvious next step is to find formulae for the number of foldings of de Bruijn graphs with arbitrary word length. Even if we could do this, we are still some way off a good description of automorphisms. Among the things we would need to do are

Counting automorphisms?

The obvious next step is to find formulae for the number of foldings of de Bruijn graphs with arbitrary word length. Even if we could do this, we are still some way off a good description of automorphisms. Among the things we would need to do are

- decide when two foldings give rise to isomorphic directed graphs, and count these (since a transducer consists of two automata with isomorphic graphs);

Counting automorphisms?

The obvious next step is to find formulae for the number of foldings of de Bruijn graphs with arbitrary word length. Even if we could do this, we are still some way off a good description of automorphisms. Among the things we would need to do are

- decide when two foldings give rise to isomorphic directed graphs, and count these (since a transducer consists of two automata with isomorphic graphs);
- decide when two transducers have the same action, and so give rise to the same automorphism.

Counting automorphisms?

The obvious next step is to find formulae for the number of foldings of de Bruijn graphs with arbitrary word length.
Even if we could do this, we are still some way off a good description of automorphisms. Among the things we would need to do are

- decide when two foldings give rise to isomorphic directed graphs, and count these (since a transducer consists of two automata with isomorphic graphs);
- decide when two transducers have the same action, and so give rise to the same automorphism.

So we still have plenty more to do!