Synchronizing automata, de Bruijn graphs, and applications

Peter J. Cameron
University of St Andrews

Progress in Mathematics towards Industrial Applications
Chennai, 29 October 2021

Summary

I am going to tell you about two types of synchronization in finite automata. Both of these have industrial applications: the first especially for putting things in the correct orientation (e.g. parts on an assembly line, or satellites in space); the second to the processing of large quantities of genetic data.

Summary

I am going to tell you about two types of synchronization in finite automata. Both of these have industrial applications: the first especially for putting things in the correct orientation (e.g. parts on an assembly line, or satellites in space); the second to the processing of large quantities of genetic data.
However, I am more concerned with the mathematics than with the applications.

Summary

I am going to tell you about two types of synchronization in finite automata. Both of these have industrial applications: the first especially for putting things in the correct orientation (e.g. parts on an assembly line, or satellites in space); the second to the processing of large quantities of genetic data.
However, I am more concerned with the mathematics than with the applications.
We will touch on a number of different areas of discrete mathematics, including weakly perfect graphs, transformation semigroups and permutation groups, homeomorphisms of Cantor space, and automorphisms of the shift in symbolic dynamics.

Synchronizing automata

An automaton is a machine which has a set Ω of states, and can read symbols from an alphabet A. It is a very simple machine: all it does at a given time step is to read a symbol and change its state.

Synchronizing automata

An automaton is a machine which has a set Ω of states, and can read symbols from an alphabet A. It is a very simple machine: all it does at a given time step is to read a symbol and change its state.
An automaton can read a word or sequence of symbols; each symbol causes a state change.

Synchronizing automata

An automaton is a machine which has a set Ω of states, and can read symbols from an alphabet A. It is a very simple machine: all it does at a given time step is to read a symbol and change its state.

An automaton can read a word or sequence of symbols; each symbol causes a state change.

An automaton is synchronizing if there is a word, called a reset word, such that when the automaton reads this word, it ends up in a fixed state, no matter where it starts.

Synchronizing automata

An automaton is a machine which has a set Ω of states, and can read symbols from an alphabet A. It is a very simple machine: all it does at a given time step is to read a symbol and change its state.

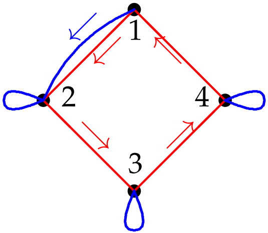
An automaton can read a word or sequence of symbols; each symbol causes a state change.

An automaton is synchronizing if there is a word, called a reset word, such that when the automaton reads this word, it ends up in a fixed state, no matter where it starts.

Reset words are useful to bring a machine into a known state before applying further transformations to it.

An infamous problem

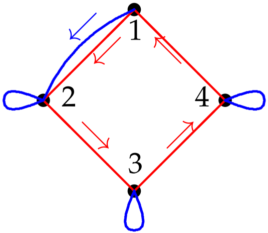
Here is a synchronizing automaton.



It can be verified that BRRRBRRRB is a reset word (and indeed that it is the shortest possible reset word for this automaton).

An infamous problem

Here is a synchronizing automaton.



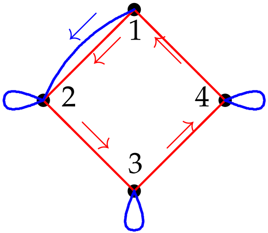
It can be verified that BRRRBRRRB is a reset word (and indeed that it is the shortest possible reset word for this automaton).

Problem

Show that, if an n-state automaton is synchronizing, it has a reset word of length at most $(n-1)^2$.

An infamous problem

Here is a synchronizing automaton.



It can be verified that BRRRBRRRB is a reset word (and indeed that it is the shortest possible reset word for this automaton).

Problem

Show that, if an n-state automaton is synchronizing, it has a reset word of length at most $(n-1)^2$.

This is the Černý conjecture, posed in the 1960s and still open.

Decision is easy

Given a finite automaton, we can decide in polynomial time whether or not it is synchronizing.

Decision is easy

Given a finite automaton, we can decide in polynomial time whether or not it is synchronizing.

This depends on the following observation:

> A finite automaton is synchronizing if and only if, for any two states s and t, there is a word $w = w_{s,t}$ in the input alphabet such that reading w from s or t takes the automaton to the same state.

Decision is easy

Given a finite automaton, we can decide in polynomial time whether or not it is synchronizing.
This depends on the following observation:

> A finite automaton is synchronizing if and only if, for any two states s and t, there is a word $w = w_{s,t}$ in the input alphabet such that reading w from s or t takes the automaton to the same state.

For such a word reduces by (at least) one the number of reachable states. So after at most $n - 1$ such words we arrive at a single state.

Decision is easy

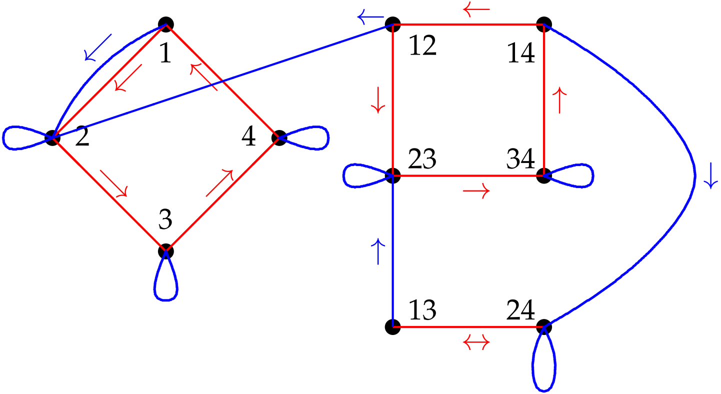
Given a finite automaton, we can decide in polynomial time whether or not it is synchronizing.
This depends on the following observation:

> A finite automaton is synchronizing if and only if, for any two states s and t, there is a word $w = w_{s,t}$ in the input alphabet such that reading w from s or t takes the automaton to the same state.

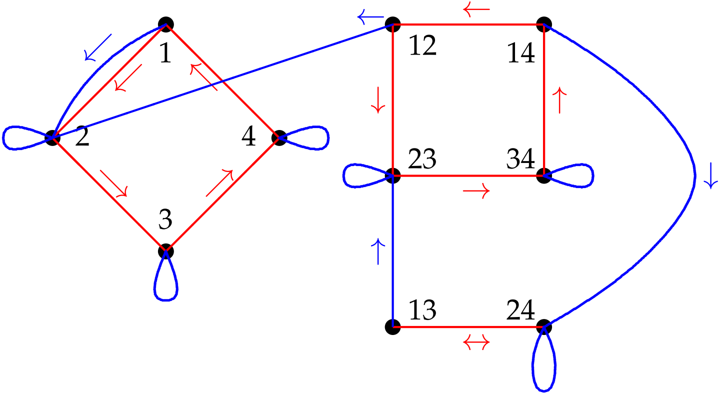
For such a word reduces by (at least) one the number of reachable states. So after at most $n - 1$ such words we arrive at a single state.
Now the next slide shows how this can be tested.

The picture shows the earlier example, with the diagram extended to show all pairs of states.

The picture shows the earlier example, with the diagram extended to show all pairs of states.

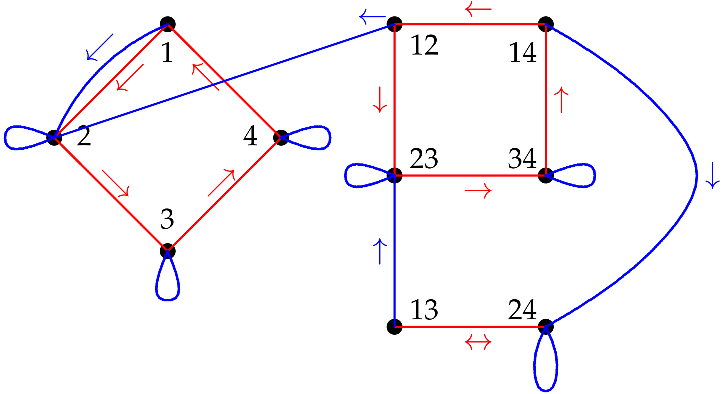


The picture shows the earlier example, with the diagram extended to show all pairs of states.



Now it suffices to check that there is a path from any vertex on the right to some vertex on the left; this can clearly be done in polynomial time.

The picture shows the earlier example, with the diagram extended to show all pairs of states.



Now it suffices to check that there is a path from any vertex on the right to some vertex on the left; this can clearly be done in polynomial time.

The resulting word has length $O(n^2)$, giving an $O(n^3)$ upper bound for the length of a reset word. The constant has been improved, but not the exponent 3.

Transformation monoids

The Černý conjecture seems to have nothing to do with either graphs or algebraic structures; but there are connections, as we will see.

Each letter of the alphabet corresponds to a transition on the set Ω of states. Reading a word corresponds to composing the transitions. So the set of all possible transitions is closed under composition and contains the identity map (corresponding to the empty word): so

> An automaton can be represented as a transformation monoid on the set Ω of states, having a distinguished set of generators. The automaton is synchronizing if and only if the monoid contains an element of rank 1.

Transformation monoids

The Černý conjecture seems to have nothing to do with either graphs or algebraic structures; but there are connections, as we will see.

Each letter of the alphabet corresponds to a transition on the set Ω of states. Reading a word corresponds to composing the transitions. So the set of all possible transitions is closed under composition and contains the identity map (corresponding to the empty word): so

> An automaton can be represented as a transformation monoid on the set Ω of states, having a distinguished set of generators. The automaton is synchronizing if and only if the monoid contains an element of rank 1.

So the Černý conjecture is a question about transformation monoids, and semigroups enter the picture.

Graphs

Graphs in this section will be ordinary simple undirected graphs, with no loops or multiple edges and no colours or directions on the edges.

Graphs

Graphs in this section will be ordinary simple undirected graphs, with no loops or multiple edges and no colours or directions on the edges.

An endomorphism of a graph is a map from the vertex set to itself which carries edges to edges. The action on nonedges is not specified; a nonedge may map to a nonedge, or to an edge, or collapse to a single vertex.

Graphs

Graphs in this section will be ordinary simple undirected graphs, with no loops or multiple edges and no colours or directions on the edges.

An endomorphism of a graph is a map from the vertex set to itself which carries edges to edges. The action on nonedges is not specified; a nonedge may map to a nonedge, or to an edge, or collapse to a single vertex.

The endomorphisms of a graph form a transformation monoid.

Graphs

Graphs in this section will be ordinary simple undirected graphs, with no loops or multiple edges and no colours or directions on the edges.

An endomorphism of a graph is a map from the vertex set to itself which carries edges to edges. The action on nonedges is not specified; a nonedge may map to a nonedge, or to an edge, or collapse to a single vertex.

The endomorphisms of a graph form a transformation monoid. Moreover, as long as the graph has at least one edge, its endomorphism monoid is not synchronizing, since that edge cannot be collapsed by any endomorphism.

Synchronization and endomorphisms

Now we have a pleasant surprise:

Synchronization and endomorphisms

Now we have a pleasant surprise:

Theorem

A transformation monoid M is non-synchronizing if and only if there is a non-trivial graph Γ on the domain such that M is contained in the endomorphism monoid of Γ. Moreover, we can assume that the clique number and chromatic number of Γ are equal.

Synchronization and endomorphisms

Now we have a pleasant surprise:

Theorem

A transformation monoid M is non-synchronizing if and only if there is a non-trivial graph Γ on the domain such that M is contained in the endomorphism monoid of Γ. Moreover, we can assume that the clique number and chromatic number of Γ are equal.

A graph is trivial if it is complete (all possible edges) or null (no edges at all). The clique number is the number of vertices in the largest complete subgraph, while the chromatic number is the number of colours required to colour the vertices so that adjacent vertices get different colours.

Synchronization and endomorphisms

Now we have a pleasant surprise:

Theorem

A transformation monoid M is non-synchronizing if and only if there is a non-trivial graph Γ on the domain such that M is contained in the endomorphism monoid of Γ. Moreover, we can assume that the clique number and chromatic number of Γ are equal.

A graph is trivial if it is complete (all possible edges) or null (no edges at all). The clique number is the number of vertices in the largest complete subgraph, while the chromatic number is the number of colours required to colour the vertices so that adjacent vertices get different colours.
The chromatic number is at least as large as the clique number.
A graph is sometimes called weakly perfect if equality holds.

Sketch proof

Since endomorphisms cannot collapse edges, it is clear that the endomorphism monoid of a non-trivial graph must be non-synchronizing.

Sketch proof

Since endomorphisms cannot collapse edges, it is clear that the endomorphism monoid of a non-trivial graph must be non-synchronizing.

For the converse, let M be a transformation monoid on Ω. We define a graph $\mathrm{Gr}(M)$ as follows: the vertex set is Ω; there is an edge joining s and t if and only if there is no element $m \in M$ with $sm = tm$. Now

Sketch proof

Since endomorphisms cannot collapse edges, it is clear that the endomorphism monoid of a non-trivial graph must be non-synchronizing.

For the converse, let M be a transformation monoid on Ω. We define a graph $\mathrm{Gr}(M)$ as follows: the vertex set is Ω; there is an edge joining s and t if and only if there is no element $m \in M$ with $sm = tm$. Now

- $\mathrm{Gr}(M)$ is non-trivial if and only if M is non-synchronizing;

Sketch proof

Since endomorphisms cannot collapse edges, it is clear that the endomorphism monoid of a non-trivial graph must be non-synchronizing.

For the converse, let M be a transformation monoid on Ω. We define a graph $\mathrm{Gr}(M)$ as follows: the vertex set is Ω; there is an edge joining s and t if and only if there is no element $m \in M$ with $sm = tm$. Now

- $\mathrm{Gr}(M)$ is non-trivial if and only if M is non-synchronizing;
- $M \leq \mathrm{End}(\mathrm{Gr}(M))$;

Sketch proof

Since endomorphisms cannot collapse edges, it is clear that the endomorphism monoid of a non-trivial graph must be non-synchronizing.

For the converse, let M be a transformation monoid on Ω. We define a graph $\mathrm{Gr}(M)$ as follows: the vertex set is Ω; there is an edge joining s and t if and only if there is no element $m \in M$ with $sm = tm$. Now

- $\mathrm{Gr}(M)$ is non-trivial if and only if M is non-synchronizing;
- $M \leq \mathrm{End}(\mathrm{Gr}(M))$;
- $\mathrm{Gr}(M)$ has clique number equal to chromatic number.

Sketch proof

Since endomorphisms cannot collapse edges, it is clear that the endomorphism monoid of a non-trivial graph must be non-synchronizing.

For the converse, let M be a transformation monoid on Ω. We define a graph $\mathrm{Gr}(M)$ as follows: the vertex set is Ω; there is an edge joining s and t if and only if there is no element $m \in M$ with $sm = tm$. Now

- $\mathrm{Gr}(M)$ is non-trivial if and only if M is non-synchronizing;
- $M \leq \mathrm{End}(\mathrm{Gr}(M))$;
- $\mathrm{Gr}(M)$ has clique number equal to chromatic number.

The first point is clear; I will outline the second. If it fails, then some element $m \in M$ maps an edge $\{s, t\}$ to either a single vertex or a non-edge. The first case contradicts the definition; in the second case, there is $m' \in M$ with $(sm)m' = (tm)m'$, so mm' maps s and t to the same place.

Sketch proof

Since endomorphisms cannot collapse edges, it is clear that the endomorphism monoid of a non-trivial graph must be non-synchronizing.

For the converse, let M be a transformation monoid on Ω. We define a graph $\mathrm{Gr}(M)$ as follows: the vertex set is Ω; there is an edge joining s and t if and only if there is no element $m \in M$ with $sm = tm$. Now

- $\mathrm{Gr}(M)$ is non-trivial if and only if M is non-synchronizing;
- $M \leq \mathrm{End}(\mathrm{Gr}(M))$;
- $\mathrm{Gr}(M)$ has clique number equal to chromatic number.

The first point is clear; I will outline the second. If it fails, then some element $m \in M$ maps an edge $\{s, t\}$ to either a single vertex or a non-edge. The first case contradicts the definition; in the second case, there is $m' \in M$ with $(sm)m' = (tm)m'$, so mm' maps s and t to the same place.

For the last point, take an element $m \in M$ of minimal rank; then m is a colouring of the graph and its image is a clique.

Does this help?

We seem to have replaced an easy problem (deciding whether an automaton is synchronizing) by a much harder problem (deciding whether the graph has clique number equal to chromatic number).

Does this help?

We seem to have replaced an easy problem (deciding whether an automaton is synchronizing) by a much harder problem (deciding whether the graph has clique number equal to chromatic number).
However, the advantage is that we can potentially show that whole classes of automata are non-synchronizing, from rather limited knowledge of their transitions.

Does this help?

We seem to have replaced an easy problem (deciding whether an automaton is synchronizing) by a much harder problem (deciding whether the graph has clique number equal to chromatic number).
However, the advantage is that we can potentially show that whole classes of automata are non-synchronizing, from rather limited knowledge of their transitions.

Strong synchronization

For what follows, I require a much stronger condition.

Strong synchronization

For what follows, I require a much stronger condition.
An automaton is strongly synchronizing at level n if, when it reads a word w of length n, the final state depends only on w and not on the initial state.

Strong synchronization

For what follows, I require a much stronger condition.
An automaton is strongly synchronizing at level n if, when it reads a word w of length n, the final state depends only on w and not on the initial state.
In other words, an automaton is strongly synchronizing at level n if every word of length n is a reset word.

Strong synchronization

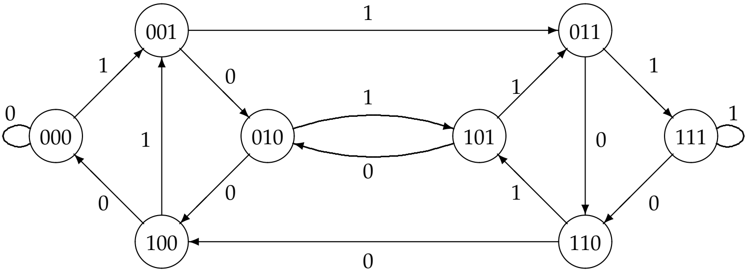
For what follows, I require a much stronger condition.
An automaton is strongly synchronizing at level n if, when it reads a word w of length n, the final state depends only on w and not on the initial state.
In other words, an automaton is strongly synchronizing at level n if every word of length n is a reset word.
This condition, as we will see, is closely connected with automorphisms of the shift map in symbolic dynamics.

De Bruijn graphs

Let n be a positive integer and A a finite alphabet. The de Bruijn graph $G(n, A)$ has vertex set A^n. For $a \in A$, $w \in A^n$, the target of the edge labelled a with source w is obtained by removing the first letter of w and appending a.

De Bruijn graphs

Let n be a positive integer and A a finite alphabet. The de Bruijn graph $G(n, A)$ has vertex set A^n. For $a \in A$, $w \in A^n$, the target of the edge labelled a with source w is obtained by removing the first letter of w and appending a.
Here is $G(3, \{0, 1\})$:



The de Bruijn graph as automaton

Clearly the de Bruijn graph satisfies the condition to be an automaton: there is a unique arc with any given label leaving any vertex.

The de Bruijn graph as automaton

Clearly the de Bruijn graph satisfies the condition to be an automaton: there is a unique arc with any given label leaving any vertex.

Regarded as an automaton, $G(n, A)$ is strongly synchronizing at level n: for if it reads a word $w = a_1 \cdots a_n$ of length n, the letters in the label of the initial state all drop off the front, and the final state is labelled by w.

The de Bruijn graph as automaton

Clearly the de Bruijn graph satisfies the condition to be an automaton: there is a unique arc with any given label leaving any vertex.

Regarded as an automaton, $G(n, A)$ is strongly synchronizing at level n: for if it reads a word $w = a_1 \cdots a_n$ of length n, the letters in the label of the initial state all drop off the front, and the final state is labelled by w.

It seems clear that it is in some sense the "universal" automaton which is strongly synchronizing at level n. We now turn to this.

Foldings

A folding of an automaton is an equivalence relation $\equiv$ on the set of states having the property that, if states s and t are equivalent, and s' and t' are the states resulting from reading a given letter a from these two states, then s' and t' are equivalent.

Foldings

A folding of an automaton is an equivalence relation $\equiv$ on the set of states having the property that, if states s and t are equivalent, and s' and t' are the states resulting from reading a given letter a from these two states, then s' and t' are equivalent. If $\equiv$ is a folding of an automaton $\mathcal{A}$, then there is a folded automaton $\mathcal{A}/\equiv$ whose states are the equivalence classes of states in $\mathcal{A}$, the transition functions defined in the obvious way. The defining condition guarantees that these are well-defined.

Foldings

A folding of an automaton is an equivalence relation $\equiv$ on the set of states having the property that, if states s and t are equivalent, and s' and t' are the states resulting from reading a given letter a from these two states, then s' and t' are equivalent. If $\equiv$ is a folding of an automaton $\mathcal{A}$, then there is a folded automaton $\mathcal{A}/\equiv$ whose states are the equivalence classes of states in $\mathcal{A}$, the transition functions defined in the obvious way. The defining condition guarantees that these are well-defined. The following are now easy to see.

Foldings

A folding of an automaton is an equivalence relation $\equiv$ on the set of states having the property that, if states s and t are equivalent, and s' and t' are the states resulting from reading a given letter a from these two states, then s' and t' are equivalent.

If $\equiv$ is a folding of an automaton $\mathcal{A}$, then there is a folded automaton $\mathcal{A}/\equiv$ whose states are the equivalence classes of states in $\mathcal{A}$, the transition functions defined in the obvious way. The defining condition guarantees that these are well-defined.

The following are now easy to see.

- If $\mathcal{A}$ is strongly synchronizing at level n, then so is any folding of $\mathcal{A}$.

Foldings

A folding of an automaton is an equivalence relation $\equiv$ on the set of states having the property that, if states s and t are equivalent, and s' and t' are the states resulting from reading a given letter a from these two states, then s' and t' are equivalent. If $\equiv$ is a folding of an automaton $\mathcal{A}$, then there is a folded automaton $\mathcal{A}/\equiv$ whose states are the equivalence classes of states in $\mathcal{A}$, the transition functions defined in the obvious way. The defining condition guarantees that these are well-defined. The following are now easy to see.

- If $\mathcal{A}$ is strongly synchronizing at level n, then so is any folding of $\mathcal{A}$.
- Any automaton which is strongly synchronizing at level n over the alphabet A is a folding of $G(n, A)$.

Foldings

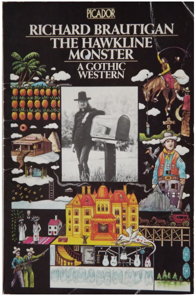
A folding of an automaton is an equivalence relation $\equiv$ on the set of states having the property that, if states s and t are equivalent, and s' and t' are the states resulting from reading a given letter a from these two states, then s' and t' are equivalent. If $\equiv$ is a folding of an automaton $\mathcal{A}$, then there is a folded automaton $\mathcal{A}/\equiv$ whose states are the equivalence classes of states in $\mathcal{A}$, the transition functions defined in the obvious way. The defining condition guarantees that these are well-defined. The following are now easy to see.

- If $\mathcal{A}$ is strongly synchronizing at level n, then so is any folding of $\mathcal{A}$.
- Any automaton which is strongly synchronizing at level n over the alphabet A is a folding of $G(n, A)$.

Problem

If $|A| = k$, how many foldings of $G(n, A)$ are there?

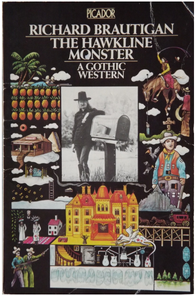
Counting foldings



"I count a lot of things that there's no need to count," Cameron said. "Just because that's 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

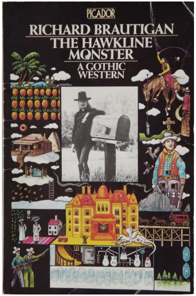


"I count a lot of things that there's no need to count," Cameron said. "Just because that's the way I am. But I count all the things that need to be counted."

Richard Brautigan, *The Hawkline Monster: A Gothic Western*

I believe that if you properly understand objects of some kind, you should be able to count them.

Counting foldings



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

Richard Brautigan, *The Hawkline Monster: A Gothic Western*

I believe that if you properly understand objects of some kind, you should be able to count them.

How many foldings of the de Bruijn graph with word length n over an alphabet of size q are there?

The problem of counting foldings seems to be very difficult. We have solved it only for $n \leq 2$ and a couple of sporadic cases.

The problem of counting foldings seems to be very difficult. We have solved it only for $n \leq 2$ and a couple of sporadic cases. The case $n = 1$ is trivial. The de Bruijn graph $G(1, A)$ has vertex set A, and for every $a \in A$, an edge labelled a from each vertex to the the vertex a. So any partition of A gives rise to a folding.

The problem of counting foldings seems to be very difficult. We have solved it only for $n \leq 2$ and a couple of sporadic cases. The case $n = 1$ is trivial. The de Bruijn graph $G(1, A)$ has vertex set A, and for every $a \in A$, an edge labelled a from each vertex to the the vertex a. So any partition of A gives rise to a folding. So the number of foldings is $B(|A|)$, the Bell number.

The problem of counting foldings seems to be very difficult. We have solved it only for $n \leq 2$ and a couple of sporadic cases.
The case $n = 1$ is trivial. The de Bruijn graph $G(1, A)$ has vertex set A, and for every $a \in A$, an edge labelled a from each vertex to the the vertex a. So any partition of A gives rise to a folding. So the number of foldings is $B(|A|)$, the Bell number.
The formula for $n = 2$ is messy to state, but easy to compute: the numbers of foldings for $|A| = 2, \ldots, 7$ are 5, 192, 78721, 519338423, 82833228599906, 429768478195109381814.

Thompson's groups

Three of the best-studied infinite groups were discovered by Richard Thompson in the 1950s, and are known as F, T and V. Here are brief descriptions.

Thompson's groups

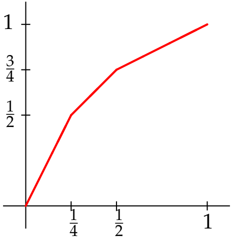
Three of the best-studied infinite groups were discovered by Richard Thompson in the 1950s, and are known as F, T and V. Here are brief descriptions.

The group F consists of piecewise-linear order-preserving permutations of the unit interval, where the slopes are powers of 2 and the points of discontinuity of the slope are dyadic rationals.

Thompson's groups

Three of the best-studied infinite groups were discovered by Richard Thompson in the 1950s, and are known as F, T and V. Here are brief descriptions.

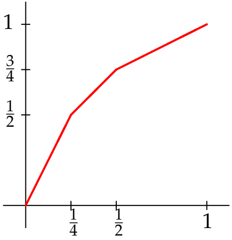
The group F consists of piecewise-linear order-preserving permutations of the unit interval, where the slopes are powers of 2 and the points of discontinuity of the slope are dyadic rationals.



Thompson's groups

Three of the best-studied infinite groups were discovered by Richard Thompson in the 1950s, and are known as F, T and V. Here are brief descriptions.

The group F consists of piecewise-linear order-preserving permutations of the unit interval, where the slopes are powers of 2 and the points of discontinuity of the slope are dyadic rationals.

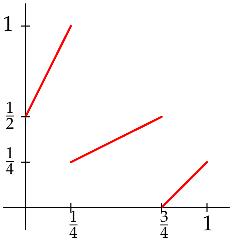


Representing numbers in the unit interval by dyadic rationals, we see that the group acts by prefix replacement: in the above example, $00x \mapsto 0x$, $01x \mapsto 10x$, $1x \mapsto 11x$.

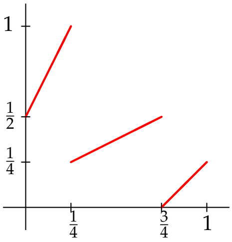
The group T is similar but preserves the circular order of the roots of unity.

The group T is similar but preserves the circular order of the roots of unity.
However our main interest lies in the group V, where the order-preserving assumption is dropped and arbitrary prefix replacement is allowed, provided only that the resulting map is a bijection.

The group T is similar but preserves the circular order of the roots of unity.
However our main interest lies in the group V, where the order-preserving assumption is dropped and arbitrary prefix replacement is allowed, provided only that the resulting map is a bijection.



The group T is similar but preserves the circular order of the roots of unity.
However our main interest lies in the group V, where the order-preserving assumption is dropped and arbitrary prefix replacement is allowed, provided only that the resulting map is a bijection.



In product replacement form this is $00x \mapsto 1x$, $01x \mapsto 010x$, $10x \mapsto 011x$, and $11x \mapsto 00x$.

The Higman–Thompson groups

The group V is a finitely presented infinite simple group, the first known example of such a group.

The Higman–Thompson groups

The group V is a finitely presented infinite simple group, the first known example of such a group.
The construction was generalised by Graham Higman to give a two-parameter family of such groups, denoted by $G_{n,r}$. (Each is finitely presented, and is simple or has a simple subgroup of index 2.) They can be defined by product replacement as above; the alphabet $\{0, 1\}$ is replaced by an alphabet of n symbols, and the parameter r indicates that at the first step we choose one of r initial symbols chosen from a different alphabet.

The Higman–Thompson groups

The group V is a finitely presented infinite simple group, the first known example of such a group.

The construction was generalised by Graham Higman to give a two-parameter family of such groups, denoted by $G_{n,r}$. (Each is finitely presented, and is simple or has a simple subgroup of index 2.) They can be defined by product replacement as above; the alphabet $\{0,1\}$ is replaced by an alphabet of n symbols, and the parameter r indicates that at the first step we choose one of r initial symbols chosen from a different alphabet.

Pardo showed that $G_{n,r} \cong G_{m,s}$ if and only if $m = n$ and $\gcd(r, n-1) = \gcd(s, m-1)$.

Transducers

To relate these groups to the previous discussion, we introduce the notion of a transducer: this is an automaton which has the capacity to write as well as read symbols from an alphabet. In general, a transducer reads a symbol, changes state, and writes a string of symbols from the alphabet (possibly empty).

Transducers

To relate these groups to the previous discussion, we introduce the notion of a transducer: this is an automaton which has the capacity to write as well as read symbols from an alphabet. In general, a transducer reads a symbol, changes state, and writes a string of symbols from the alphabet (possibly empty).
In order to avoid trivial cases, we always assume that *when a transducer reads an infinite string of symbols, it writes out an infinite string*: in other words, if we traverse a cycle in the digraph of the underlying automaton, at least one symbol is written.

Transducers

To relate these groups to the previous discussion, we introduce the notion of a transducer: this is an automaton which has the capacity to write as well as read symbols from an alphabet. In general, a transducer reads a symbol, changes state, and writes a string of symbols from the alphabet (possibly empty).

In order to avoid trivial cases, we always assume that *when a transducer reads an infinite string of symbols, it writes out an infinite string*: in other words, if we traverse a cycle in the digraph of the underlying automaton, at least one symbol is written.

As just hinted, a transducer A with a prescribed starting state s (called an initial transducer) can be regarded as defining a map from the set A^ω of infinite strings over the alphabet A to itself.

We are interested in the case where this map is invertible, and the inverse is also represented by a transducer.

The rational group

The rational group $\mathcal{R}_n$ over an n-letter alphabet A was defined by Grigorchuk, Nekrashevych, and Suschanskiĭ.

The rational group

The rational group $\mathcal{R}_n$ over an n-letter alphabet A was defined by Grigorchuk, Nekrashevych, and Suschanskiĭ.
It is the group of invertible transformations of A^ω induced by initial transducers.

The rational group

The rational group $\mathcal{R}_n$ over an n-letter alphabet A was defined by Grigorchuk, Nekrashevych, and Suschanskiĭ.

It is the group of invertible transformations of A^ω induced by initial transducers.

The maps are composed in the usual way; we can define a composition directly on transducers by using the output of the first transducer as input to the second.

The rational group

The rational group $\mathcal{R}_n$ over an n-letter alphabet A was defined by Grigorchuk, Nekrashevych, and Suschanskiĭ.

It is the group of invertible transformations of A^ω induced by initial transducers.

The maps are composed in the usual way; we can define a composition directly on transducers by using the output of the first transducer as input to the second.

The definition can be extended to the group $\mathcal{R}_{n,r}$, which acts on strings where the first symbol is taken from an auxiliary alphabet of size r.

The automorphism group of $G_{n,r}$

An invertible initial transducer is said to be bisynchronizing if the underlying automaton is strongly synchronizing, and the same holds for the automaton representing its inverse.

Theorem

The automorphism group of $G_{n,r}$ is the group of transformations of A^ω induced by bisynchronizing initial transducers; so it is a subgroup of the rational group $\mathcal{R}_{n,r}$.

The automorphism group of $G_{n,r}$

An invertible initial transducer is said to be bisynchronizing if the underlying automaton is strongly synchronizing, and the same holds for the automaton representing its inverse.

Theorem

The automorphism group of $G_{n,r}$ is the group of transformations of A^ω induced by bisynchronizing initial transducers; so it is a subgroup of the rational group $\mathcal{R}_{n,r}$.

This theorem is proved in the paper of Bleak, Cameron, Maissel, Navas and Olukoya (arXiv 1605.09302).

Consequences

I mention here two consequences of this analysis.

Consequences

I mention here two consequences of this analysis.

Theorem

The outer automorphism group of $G_{n,r}$ has trivial centre and unsolvable order problem.

Consequences

I mention here two consequences of this analysis.

Theorem

The outer automorphism group of $G_{n,r}$ has trivial centre and unsolvable order problem.

The proof involves a connection between $\mathrm{Out}(G_{n,r})$ and the automorphism group of the two-sided shift in symbolic dynamics, allowing known results about the second to be transferred to the first. I turn now to this.

Shift maps

The shift map σ comes in two flavours. It acts on either the set A^ω of infinite strings of symbols from A, or on the set $A^{\mathbb{Z}}$ of two-way infinite strings; it moves each symbol one place to the left. (In the one-way case, the first symbol of the string is lost, so the shift is onto but not one-to-one; in the second case it is a bijection.)

Shift maps

The shift map σ comes in two flavours. It acts on either the set A^ω of infinite strings of symbols from A, or on the set $A^{\mathbb{Z}}$ of two-way infinite strings; it moves each symbol one place to the left. (In the one-way case, the first symbol of the string is lost, so the shift is onto but not one-to-one; in the second case it is a bijection.)

For example, if $A = \{0, 1\}$ and we interpret A^ω as the set of binary decimals representing the unit interval, then the shift map is the function $x \mapsto 2x \pmod 1$.

Shift maps

The shift map σ comes in two flavours. It acts on either the set A^ω of infinite strings of symbols from A, or on the set $A^{\mathbb{Z}}$ of two-way infinite strings; it moves each symbol one place to the left. (In the one-way case, the first symbol of the string is lost, so the shift is onto but not one-to-one; in the second case it is a bijection.)

For example, if $A = \{0, 1\}$ and we interpret A^ω as the set of binary decimals representing the unit interval, then the shift map is the function $x \mapsto 2x \pmod 1$.

The shift map is the central character in *symbolic dynamics*, arising from a discretisation of dynamics of (for example) planetary orbits.

Automorphisms of the shift

An automorphism of the shift is a homeomorphism of X^ω or $X^\mathbb{Z}$ (regarded as Cantor space) which commutes with σ.

Automorphisms of the shift

An automorphism of the shift is a homeomorphism of X^ω or $X^{\mathbb{Z}}$ (regarded as Cantor space) which commutes with σ.
The connection between automata and automorphsims of the shift was pointed out by Grigorchuk *et al.* in 2000.

Automorphisms of the shift

An automorphism of the shift is a homeomorphism of X^ω or $X^{\mathbb{Z}}$ (regarded as Cantor space) which commutes with σ.

The connection between automata and automorphsims of the shift was pointed out by Grigorchuk *et al.* in 2000.

Automorphisms of the one-sided shift are given by transducers; in the case of the two-sided shift, we will see that a little more is required.

Automorphisms of the shift

An automorphism of the shift is a homeomorphism of X^ω or $X^{\mathbb{Z}}$ (regarded as Cantor space) which commutes with σ.

The connection between automata and automorphsims of the shift was pointed out by Grigorchuk *et al.* in 2000.

Automorphisms of the one-sided shift are given by transducers; in the case of the two-sided shift, we will see that a little more is required.

Two recent papers by Bleak, Cameron and Olukoya (arXiv 2004.08478 and 2006.01466) use transducers to study the automorphism groups of the shift maps. Some of the results are new; several give simpler proofs of known results, or versions more suitable to actual computation. Here are some examples.

Automorphisms of the shift

An automorphism of the shift is a homeomorphism of X^ω or $X^\mathbb{Z}$ (regarded as Cantor space) which commutes with σ.

The connection between automata and automorphsims of the shift was pointed out by Grigorchuk *et al.* in 2000.

Automorphisms of the one-sided shift are given by transducers; in the case of the two-sided shift, we will see that a little more is required.

Two recent papers by Bleak, Cameron and Olukoya (arXiv 2004.08478 and 2006.01466) use transducers to study the automorphism groups of the shift maps. Some of the results are new; several give simpler proofs of known results, or versions more suitable to actual computation. Here are some examples.

First, it is noted that the automorphism group of the one-sided shift over an n-letter alphabet embeds into the group of outer automorphisms of $G_{n,r}$: the automorphisms are given by bisynchronizing transducers.

In the one-sided case, the orders of torsion elements of $\mathrm{Aut}(\sigma)$ are orders of automorphism groups of foldings of de Bruijn graphs.

In the one-sided case, the orders of torsion elements of $\mathrm{Aut}(\sigma)$ are orders of automorphism groups of foldings of de Bruijn graphs.
In the two-sided case, $\mathrm{Aut}(\sigma)$ contains the group generated by σ as a central subgroup; the quotient is embeddable in the group of outer automorphisms of $G_{n,r}$.

In the one-sided case, the orders of torsion elements of $\mathrm{Aut}(\sigma)$ are orders of automorphism groups of foldings of de Bruijn graphs.

In the two-sided case, $\mathrm{Aut}(\sigma)$ contains the group generated by σ as a central subgroup; the quotient is embeddable in the group of outer automorphisms of $G_{n,r}$.

In this case, automorphisms are specified by an annotated transducer, where the transducer determines the coset of $\langle\sigma\rangle$, and the annotation determines the element of this coset.

References

J. Araújo, P. J. Cameron and B. Steinberg, Between primitive and 2-transitive: Synchronization and its friends, *Europ. Math. Soc. Surveys* **4** (2017), 101–184.

Collin Bleak, Peter Cameron, Yonah Maissel, Andrés Navas, and Feyishayo Olukoya, The further chameleon groups of Richard Thompson and Graham Higman: Automorphisms via dynamics for the Higman groups $G_{n,r}$, arXiv `1605.09302`.

Collin Bleak, Peter Cameron and Feyishayo Olukoya, Automorphisms of shift spaces and the Higman–Thompson groups: the one-sided case, *Discrete Analysis* **2021:15** (2021), 35pp.; (with James Belk) the two-sided case, arXiv `2006.01466`.

R. I. Grigorchuk, V. V. Nekrashevych, V. I. Sushchanskiĭ, Automata, dynamical systems and groups, *Tr. Mat. Inst. Steklova*, **231** (2000), 134–214; English version *Proc. Steklov Inst. Math.* **231** (2000), 128–203.

… for your attention.