Inverse Semigroups and their applications

Peter M. Hines
Y.C.C.S.A. , Univ. York

York Mathematics – Algebra Seminar
An introduction to inverse semigroup theory

— Elementary definitions & theory from the point of view of applications.

— Where we find these structures in
  - theoretical & practical computer science
  - other areas of mathematics
We will look at **inverse semigroups & monoids**:  
- A branch of abstract algebra / semigroup theory.  
- Introduced simultaneously & independently in 1950’s  
  - Viktor Wagner (U.S.S.R.)  
  - Gordon Preston (U.K.)  
- Theory developed separately, along two different tracks  
  - USSR  
  - U.S. & Europe  

with minimal contact between the two sides.  
- Some degree of re-unification in 1990s
A couple of references:


[Image of a book cover titled "Inverse Semigroups: The Theory of Partial Symmetries" by Mark V. Lawson]
Some -very- basic definitions

A **semigroup** is a set \((S, \cdot)\) with an associative binary operation \(\cdot : S \times S \rightarrow S\). usually written as concatenation:
- Given \(a \in S\) and \(b \in S\), then \(ab \in S\).
- \(a(bc) = (ab)c\) for all \(a, b, c \in S\).

A **monoid** is a semigroup with an identity \(1 \in S\) satisfying

\[
1a = a = a1 \quad \forall \ a \in S
\]

A **group** is a monoid where every \(a \in S\) has an inverse \(a^{-1} \in S\)

\[
aa^{-1} = 1 = a^{-1}a \quad \forall \ a \in S
\]
Even more basic definitions

Simplest examples include **free semigroups** and **monoids**.

**The free semigroup on a set** \( X \)

\( X^+ \) is the set of all non-empty strings of symbols of \( X \).
Composition is just concatenation of strings.

The **free monoid** \( X^* \) also allows for the empty string \( \lambda \).

Free semigroups / monoids have the expected universal property ...
A congruence $\sim$ on a semigroup $S$ is a composition-preserving equivalence relation:

$$a \sim b \text{ and } x \sim y \Rightarrow ax \sim by$$

for all $a, b, x, y \in S$.

Equivalence classes form the quotient semigroup $S/\sim$.

Every semigroup (monoid) is a quotient of some free semigroup (monoid).

There is no analogy of “normal subgroup” for monoids!
Inverse monoids / semigroups have a ‘relaxed’ notion of inverses:

Inverse semigroups: the definition

Every element $a \in S$ has a unique \textit{generalised inverse} $a^\dagger \in S$ satisfying

$$aa^\dagger a = a \quad \text{and} \quad a^\dagger aa^\dagger = a^\dagger$$

Axioms introduced independently by Wagner & Preston, based on (different collections of) concrete examples.

Many examples are – even nowadays – not always recognised as being inverse semigroups.
Elementary properties

In an inverse semigroup $S$, the following are almost immediate:

1. $(a^\dagger)^\ddagger = a$, for all $a \in S$.
2. $(ab)^\dagger = b^\dagger a^\dagger$, for all $a, b \in S$.
3. $e^\dagger = e$, for any idempotent $e^2 = e$.
   (Special case: $I^\dagger = I$, when $S$ is a monoid).
4. $aa^\dagger$ and $a^\dagger a$ are both idempotent.
5. All idempotents commute:

   $e^2 = e$ and $f^2 = f \implies ef = fe$
A (well-known) class of examples!

All groups are (trivially) inverse monoids, but not vice versa.

Important

Even in a monoid, the conditions

\[ aa^\dagger a = a \quad \text{and} \quad a^\dagger aa^\dagger = a^\dagger \]

do not imply that \( aa^\dagger \) is the identity.

Instead, \( aa^\dagger \) and \( a^\dagger a \) are both idempotent (i.e. \( e^2 = e \)).

The inverse semigroup axioms are strictly more general than the group axioms.

How should we understand these axioms?
A representation theorem or two

Cayley’s theorem (1854)
Every group has a representation as bijections on a set.

The Wagner-Preston theorem (1954)
Every inverse semigroup has a representation as partial injections on a set.

Inverse semigroup theory is what happens when we combine reversibility with partiality.

Historically, computer scientists have been more comfortable with partiality than mathematicians.
How do partial functions compose?

Given partial functions $f : X \rightarrow Y$ and $g : Y \rightarrow Z$, then $gf(x)$ is defined when

- $x$ is in the domain of $f$.
- $f(x)$ is in the domain of $g$.

### Composing partial reversible functions

<table>
<thead>
<tr>
<th>Partial function $g$</th>
<th>Partial function $f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 ← 6</td>
<td>6 ← 6</td>
</tr>
<tr>
<td>5 ← 5</td>
<td>5 ← 5</td>
</tr>
<tr>
<td>4 ← 4</td>
<td>4 ← 4</td>
</tr>
<tr>
<td>3 ← 3</td>
<td>3 ← 3</td>
</tr>
<tr>
<td>2 ← 2</td>
<td>2 ← 2</td>
</tr>
<tr>
<td>1 ← 1</td>
<td>1 ← 1</td>
</tr>
<tr>
<td>0 ← 0</td>
<td>0 ← 0</td>
</tr>
</tbody>
</table>

peter.hines@york.ac.uk
How do partial functions compose?

Given partial functions $f : X \to Y$ and $g : Y \to Z$, then $gf(x)$ is defined when

- $x$ is in the domain of $f$.
- $f(x)$ is in the domain of $g$.

Composing partial reversible functions

<table>
<thead>
<tr>
<th>Partial function $gf$</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

$g(6) = 6$, $g(5) = 5$, $g(4) = 4$, $g(3) = 3$, $g(2) = 2$, $g(1) = 1$, $g(0) = 0$
It is easy to convince ourselves that, unless domains / images match exactly\(^1\),

1. The composites get progressively ‘less defined’
2. Long composites must tend towards the nowhere-defined partial function \(0\).

Both of these intuitions are \textit{incorrect}.

The reason why is best illustrated by example.

\(^1\)In which case, everything reduces to group theory ...
Some interesting partial injections

Partial injections defined on the **odd** and **even** numbers only:

**Partial injection** \( p \)

\[
p(n) = \begin{cases} 
\frac{n}{2} & \text{n even,} \\
\bot & \text{otherwise.}
\end{cases}
\]

**Partial injection** \( q \)

\[
q(n) = \begin{cases} 
\frac{n-1}{2} & \text{n odd,} \\
\bot & \text{otherwise.}
\end{cases}
\]
Total inverses of partial injections

Their generalised inverses are globally defined injections:

\[ p^\dagger(n) = 2n \]
\[ q^\dagger(n) = 2n + 1 \]
Non-shrinking domains!

Observe:

\[ \text{dom}(p) \subset \text{dom}(p^2) \subset \text{dom}(p^3) \subset \text{dom}(p^4) \subset \text{dom}(p^5) \subset \ldots \]

Nevertheless, these are all countably infinite.

These two functions generate a representation of the (inverse) \textbf{polycyclic monoid} \( P_2 \) of Nivat & Perot (1972). Specified by simple relations:

\[ pp^\dagger = I = qq^\dagger \quad \text{and} \quad pq^\dagger = 0 = qp^\dagger \]

Also known as the logicians’ ‘dynamical algebra’
A significant inverse monoid:

Given any set $X$, the **polycyclic monoid** $P_X$, is the inverse monoid with:

- $X$ as a generating set,
- A zero $0$ and an identity $1$,
- the relations

$$xy^\ddagger = \begin{cases} 
1 & x = y \\
0 & x \neq y 
\end{cases}$$

A ‘one-sided version of the Kronecker delta’.

A relevant property

Provided $|X| > 1$, there are no non-trivial congruences on $P_X$.

Any quotient causes a collapse to a single element.
The question of significance ...

What makes $P_2$ a “significant” monoid ??

A useful criterion

When it is repeatedly re-discovered in different fields:

Logic & $\lambda$ calculus  The dynamical algebra

$C^*$ algebra & mathematical physics  The Cuntz algebra

Automata theory  Syntactic monoids of certain automata

Language theory  The well-formed bracketing language

A range of areas  Linguistics, Ring Theory, Tilings, Category Theory, Foundations of Mathematics, . . .
First seen as the ‘dynamical algebra’, of
“Local and Asynchronous $\beta$-reduction”
– V. Danos & L. Regnier (1992)

Models of untyped $\lambda$ calculus, and hence computational universality.

Later core to logical models — particularly those of J.-Y. Girard.

Let’s look at more ‘elementary’ applications ...
Race Conditions

In parallel or multi-threaded computation:

“The behaviour of a system varies according to the order in which individual operations from distinct threads are processed’.”

The distinct behaviours may be:

*desirable, undesirable, or unimportant.*

A non-judgmental analysis:

The terms ‘desirable’ or ‘undesirable’ are subjective.

We *study* such conditions, without aiming to either cause or eliminate them!
Connecting to the same Starbucks personal account simultaneously from two distinct browsers caused a race condition among multiple asynchronous processes for:

1. Check balance on card 1.
2. If sufficient funds, add funds to card 2.
3. Decrease funds on card 1.

Disclaimer: This bug has since been fixed!
Consider the free monoid over the set \( \{A, C, D\} \)

- \( A \) – add funds to card 2.
- \( C \) – check funds on card 1.
- \( D \) – decrease funds on card 1.

Which particular strings of actions (submonoids of the free monoid \( \{A, C, D\}^* \)) are:

1. Permitted by Starbucks servers?
2. Possible to create, using two distinct connections?
3. Profitable for Egor Homakov??
4. Fair to everyone concerned?

What we wish to find:

Tools to find *intersections* of such monoids, and *transformations* (homomorphisms?) between them.
Interleaving processes as shuffles

The mathematics of **shuffling cards** and **interleaving processes** is of course identical.

Card shuffles are very well-studied in **combinatorics**, **probability**, **representation theory**, **statistics**, &c.

For some applications to C.S., we also need their (inverse) semigroup theory.
Combinatorics answers questions such as:

Given $K$ decks of $N$ cards, how many different ways are there of shuffling them into a single stack of $K \times N$ cards?

The right tools for multi-threaded finite computational tasks such as parallel matrix processing.

We need to consider the infinite setting:

unbounded number of decks Arbitrarily many clients connected to a server.

a never-ending stream of cards Non-terminating processes (internet servers, again).
How to shuffle two (possibly infinite) decks of cards

Riffle Shuffles

- Cards from Deck $A$ and Deck $B$ are merged into a single stack.
- At each step, a single card is taken from the bottom of either $A$ or $B$, and placed on top of the stack.

Some important conventions:

- The ordering of cards is preserved.
- Every card from each deck ends up in the stack.
Everything in order ...

Consider two copies of the natural numbers:

\[ \mathbb{N} \oplus \mathbb{N} \overset{\text{def.}}{=} \mathbb{N} \times \{0, 1\} \]

and give this a \textit{partial order} by

\[(a, i) \leq (b, j) \iff a \leq b \text{ and } i = j\]

We may only compare members of the same copy of \( \mathbb{N} \):

- \((4, 0) \leq (7, 0)\)
- \((3, 1) \leq (8, 1)\)
- \((4, 0)\) and \((8, 1)\) are incomparable.
Shuffles of two infinite decks of cards

≡

order-preserving injections

from $\mathbb{N} \cup \mathbb{N}$ to $\mathbb{N}$.

How may we characterise (not count!) these?

An old result (P.M.H. — M.V. Lawson 1998)

Arbitrary injections from $\mathbb{N} \cup \mathbb{N}$ to $\mathbb{N}$ are in 1:1 correspondence with (effective) representations of $P_2$ on $\mathbb{N}$.

What about the monotone (order-preserving) case?
The infinitary setting

Every shuffle of two infinite decks corresponds to a point of Cantor space $\mathcal{C}$.

Formally, one-sided infinite strings over $\{0, 1\}$,

$$c = 0100101101 \ldots$$

or equivalently, functions from $\mathbb{N}$ to $\{0, 1\}$. 
**Operationally:** Cantor points are *descriptions* of shuffles:

Given a Cantor point $c : \mathbb{N} \to \{0, 1\}$,

At the $n^{th}$ step, a card was taken from:
- The first deck, when $c(n) = 0$
- The second deck, when $c(n) = 1$

*Caution!*

We can also think of Cantor points as *instructions*,
but not all Cantor points arise from valid shuffles.
An illustrative example

The **perfect riffle shuffle**:  

Cards are alternately taken from each deck  

This is modeled by the function $\phi : \mathbb{N} \times \{0, 1\} \rightarrow \mathbb{N}$ given by  

$$\phi(n, i) = 2n + i$$  

The corresponding Cantor point is $a(n) = n \ (mod \ 2)$.

$$a = 0101010101\ldots$$

The **alternating Cantor point**
Which Cantor points are *not* shuffles?

Recall our two conditions:

1. The ordering of cards is preserved,
2. Every card is laid at some point.

Condition 1. is accounted for by monotonicity.

Condition 2. is automatically satisfied, simply because 
\( \phi : \mathbb{N} \times \{0, 1\} \to \mathbb{N} \) is a *globally defined* function.

A consequence is that that the corresponding Cantor point is **balanced**:

\[
\sum_{i=0}^{\infty} c(i) = \infty = \sum_{i=0}^{\infty} (1 - c(i))
\]
The correspondence (mathematically)

Given a shuffle of two infinite decks \( \phi : \mathbb{N} \times \{0, 1\} \to \mathbb{N} \)
consider its (global) inverse \( \phi^{-1} : \mathbb{N} \to \mathbb{N} \times \{0, 1\} \).
For all \( n \in \mathbb{N} \), we have a pair \( \phi^{-1}(n) = (x_n, i_n) \in \mathbb{N} \times \{0, 1\} \).

From shuffles to Cantor points
we define a Cantor point

\[
c_{\phi} = \pi_2 \phi^{-1} : \mathbb{N} \to \{0, 1\} \in \mathcal{C}
\]

by *projecting onto the second component* \( c_{\phi}(n) = i_n \in \mathcal{C} \)

As \( \phi \) is *monotone*, this Cantor point is enough to characterise \( \phi \).
What about that other projection?

Does projecting onto the *first* component also characterise $\phi$?

**Definitely not** There are uncountably many distinct shuffles where such projections are identical.

However, taking two *partial* projections will work!
Given the inverse of a shuffle $\phi^{-1} : \mathbb{N} \to \mathbb{N} \times \{0, 1\}$, let us split the projection onto the first component into two distinct monotone partial injections

$$p_\phi(n) = \begin{cases} \pi_1 \phi^{-1}(n) & \pi_2 \phi^{-1}(n) = 0 \\ \text{undefined} & \text{otherwise.} \end{cases}$$

$$q_\phi(n) = \begin{cases} \pi_1 \phi^{-1}(n) & \pi_2 \phi^{-1}(n) = 1 \\ \text{undefined} & \text{otherwise.} \end{cases}$$

These two partial injections are enough to characterise $\phi$. 
The generalised inverses of $p_\phi$ and $q_\phi$ are monotone injections, and satisfy

\[ p_\phi p_\phi^\dagger = I = q_\phi q_\phi^\dagger \]

\[ q_\phi p_\phi^\dagger = 0 = p_\phi q_\phi^\dagger \]

Giving an effective representation of a two-generator polycyclic monoid.
Given a balanced Cantor point $c : \mathbb{N} \rightarrow \{0, 1\}$, we define partial injections by:

**Counting the number of 0s up to point $n$**

$$p^+_c(n) = \begin{cases} 
\left( \sum_{j=0}^{n} 1 - c(j) \right) - 1 & c(n) = 0, \\
\text{undefined} & \text{otherwise.}
\end{cases}$$

**Counting the number of 1s up to point $n$**

$$q^+_c(n) = \begin{cases} 
\left( \sum_{j=0}^{n} c(j) \right) - 1 & c(n) = 1, \\
\text{undefined} & \text{otherwise.}
\end{cases}$$
Best illustrated by example ...

Consider the alternating Cantor point

\[ a = 01010101010101010 \ldots \]

**partial injection** \( p_a \)

\[
\begin{align*}
7 & \rightarrow 7 \\
6 & \rightarrow 6 \\
5 & \rightarrow 5 \\
4 & \rightarrow 4 \\
3 & \rightarrow 3 \\
2 & \rightarrow 2 \\
1 & \rightarrow 1 \\
0 & \rightarrow 0
\end{align*}
\]

**partial injection** \( q_a \)

\[
\begin{align*}
7 & \leftarrow 7 \\
6 & \leftarrow 6 \\
5 & \leftarrow 5 \\
4 & \leftarrow 4 \\
3 & \leftarrow 3 \\
2 & \leftarrow 2 \\
1 & \leftarrow 1 \\
0 & \leftarrow 0
\end{align*}
\]
We have 1:1 mappings between:

Interleavings of two infinite streams of processes

Balanced points of Cantor space

Monotone effective representations of 2-generator polycyclic monoids

Is there any advantage to treating such things algebraically?
Re-ordering processes

Given a sequence of ‘cards’

\[ A_0 \ A_1 \ A_2 \ A_3 \ A_4 \ A_5 \ldots \]

resulting from some (undesirable) shuffle, specified by
\[ c : \mathbb{N} \rightarrow \{0, 1\}, \]
how can we *re-order it*

\[ A_5 \ A_2 \ A_3 \ A_0 \ A_1 \ A_4 \ldots \]

so it appears to have come from a (desirable) shuffle
\[ d : \mathbb{N} \rightarrow \{0, 1\} \]
How to re-order shuffles

Some well-known semigroup theory

Given partial injections \( f, g \) with:
- disjoint domains
- disjoint images

their set-theoretic union \( f \cup g \) is also a partial injection.

Given shuffles \( c, d : \mathbb{N} \to \{0, 1\} \), the result of \( c \) may be re-arranged into the result of \( d \) by

\[
p_d^\dagger p_c \cup p_d^\dagger q_c : \mathbb{N} \to \mathbb{N}
\]

— a globally defined bijection on \( \mathbb{N} \).
From a practical viewpoint:

Tasks cannot be re-ordered if they are processed the instant they are received!

For re-arrangement to take place, they must first be held in a buffer / queue.

- How big does this need to be — how long a queue is (computationally) acceptable?
- What transformations on this buffer are needed?
- Are there situations where no finite re-arrangement will work?
A traditional approach to (finite) shuffles is via **Young Tableaux**.

We derive infinitary versions from the algebra in a simple manner:

Consider the Cantor point \( c = 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0 \ 0 \ldots \in \mathcal{C} \) and associated partial injections \( p^{\dagger}, q^{\dagger} : \mathbb{N} \rightarrow \mathbb{N} \)

<table>
<thead>
<tr>
<th>( n = )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>\ldots</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p^{\dagger}(n) = )</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>9</td>
<td>\ldots</td>
</tr>
<tr>
<td>( q^{\dagger}(n) = )</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>\ldots</td>
</tr>
</tbody>
</table>

An \((\infty, \infty)\) Young tableau.

The obvious question:

What about **standard** Young tableaux?
On to standard Young tableaux

In **standard** Young tableaux, the cells are well-ordered both *horizontally* and *vertically*.

\[
\begin{array}{cc}
  a & b \\
  c & \end{array}
\]

Horizontal ordering corresponds to monotonicity.

What about the vertical ordering?
Recall the motivation for studying Shuffles, as *ordering of processes*.

- Operations from thread A push data onto a stack.
- Operations from thread B pop data off a stack.

What conditions would prevent us from trying to *read data from an empty stack*?

... or indeed, transfer funds from an empty account?
A (binary) ballot sequence is an element \( w \in \{0, 1\}^* \) where, for every prefix \( u \) of \( w \),

\[
\#1s \text{ in } u \leq \#0s \text{ in } u
\]

Denote the set of all finite ballot sequences by \( B\{0,1\} \) — this forms a submonoid of \( \{0, 1\}^* \).

By contradiction: Consider \( v, w \in B\{0,1\} \) such that \( vw \not\in B\{0,1\} \). Then there exists some prefix \( u \) of \( vw \) satisfying \( \#0s \) in \( u < \#1s \) in \( u \). As \( v \in B\{0,1\} \), \( u \) is not a prefix of \( v \), so \( u = vl \), for some prefix \( l \) of \( w \). However, \( \#0s \) in \( v \geq \#1s \) in \( v \). Therefore, \( \#1s \) in \( l \geq \#0s \) in \( l \), contradicting the assumption that \( w \in B\{0,1\} \).
Ballot sequences are *well-studied* in combinatorics – but also make for interesting monoids!

**Proposition** The monoid of binary ballot sequences is not finitely generated.

**By contradiction:** Assume a finite generating set $G$ for $B_{\{0,1\}} \leq \{0, 1\}^*$. As $G$ is finite, the longest contiguous string of 1s in any member of $G$ is bounded by some finite $K \in \mathbb{N}$. No composite of members of $G$ can account for the ballot sequence $0^{K+1}1^{K+1}$. 
A Cantor point $c \in \mathcal{C}$ is \textit{ballot} when every prefix is a member of the Ballot monoid.

$$
\sum_{j=0}^{N} c(j) \leq \sum_{j=0}^{N} c^\perp(j) \quad \forall \ N \in \mathbb{N}
$$

Denote the ballot Cantor points by $\mathcal{B} \subseteq \mathcal{C}$.

Our claim:

Shuffles described by \textit{ballot} Cantor points are precisely those whose Young tableaux are standard.
Balanced ballot Cantor points

\[ \equiv \]

standard \((\infty, \infty)\) Young tableaux

Let \( c \in \mathcal{B} \) be a balanced ballot Cantor point. This determines a monotone representation of \( P_2 \) as partial injections on \( \mathbb{N} \), and hence an \((\infty, \infty)\) Young tableau:

<table>
<thead>
<tr>
<th>( p^\downarrow(0) )</th>
<th>( p^\downarrow(1) )</th>
<th>( p^\downarrow(2) )</th>
<th>( p^\downarrow(3) )</th>
<th>( p^\downarrow(4) )</th>
<th>\ldots</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q^\downarrow(0) )</td>
<td>( q^\downarrow(1) )</td>
<td>( q^\downarrow(2) )</td>
<td>( q^\downarrow(3) )</td>
<td>( q^\downarrow(4) )</td>
<td>\ldots</td>
</tr>
</tbody>
</table>

By the interpretation of \( p(n) \) and \( q(n) \) as ‘counting the 0s and 1s in a prefix’, \( p^\downarrow(n) \leq q^\downarrow(n) \), so this is standard.
A balanced ballot point \( b \in \mathcal{B} \) satisfies:

\[
\sum_{j=0}^{\infty} b(j) = \sum_{j=0}^{\infty} (1 - b(j))
\]

The total number of 0s and 1s is the same.

\[
\sum_{j=0}^{N} b(j) \leq \sum_{j=0}^{N} (1 - b(j)).
\]

Every prefix has at least as many 0s as 1s.

Imposing such conditions on finite strings results in an uninteresting theory!

Finite balanced Ballot points are simply powers of \((01)\).

The free monoid on a single generator!
The (balanced) Ballot Cantor points form a subset of Cantor space;

We can draw a picture.
Ballot Cantor points – the “fork factory”
There are many different ways of ordering:

- Cantor points in general,
- Ballot points in particular.

The pointwise partial order:

Given Cantor points \( a, b : \mathbb{N} \rightarrow \{0, 1\} \),

we use the pointwise partial ordering:

\[
a \leq b \text{ iff } a(n) \leq b(n) \quad \forall n \in \mathbb{N}
\]

The Ballot points of Cantor space have a particularly neat form.
The Ballot Scott domain

Key properties:

- There is no top element & they are **not** closed under joins
  \[(c \lor d)(n) = \max\{c(n), d(n)\}\.

- They **are** closed under the meet, \[(c \land d)(n) = c(n)d(n)\.

- There is a bottom element \(\bot(n) = 0\), for all \(n \in \mathbb{N}\).

- The supremum of every chain \(c_0 \leq c_1 \leq c_2 \leq \ldots\) is also in \(\mathcal{B}\)
  - chain-completeness \(\Rightarrow\) directed completeness, assuming the axiom of choice (Iwamura’s Lemma).

- There is a notion of **finite support / compactness**: \(c \in \mathcal{B}\) is
  “finitary” iff \(\sum_{j=0}^{\infty} c(j) < \infty\), and every element is the supremum of a chain of such elements.
Scott Domains in Theoretical Computer Science

Scott Domains ...

- Introduced by Dana Scott (early 1970s) to model pure untyped \( \lambda \) calculus
  - and hence computational universality.

- Also used for semantics of functional programming languages, due to the existence of solutions of arbitrary fixed-point equations.

This particular Scott domain is a subset of Cantor space related to standard Young tableaux.
Back to our original motivation ...

Given $C, D \subseteq \mathbb{N}$, the **banker's monoid** $\mathcal{W}_{C,D}$ is a submonoid of the free monoid over $+C \cup -D$.

**Interpretation**

For any $c \in C$, and $d \in D$,

$$+c \text{ is a deposit } \ , \ -d \text{ is a withdrawal.}$$

Elements of $\mathcal{W}_{C,D}$ are **no-credit strings** — those for which *the sum of every prefix is non-negative*.

Taking $C = \{2, 4, 6, 8\}$ and $D = \{1, 3, 5, 7\}$,

$$(+8)(-5)(+4)(-7)(+4)(-3) \quad \text{is a n.-c. string}$$

$$(+6)(-5)(+2)(-5)(+8) \quad \text{is not a n.-c. string}$$
Reducing the complex to the simple

It is relatively straightforward to prove:

1. The Ballot monoid $B_{0,1}$ is isomorphic to $W_{1,1}$.
2. For arbitrary $C, D \subseteq \mathbb{N}$, there is an embedding $W_{C,D} \hookrightarrow B_{0,1}$.

We may use the same structures to study

1. Race conditions for stacks
2. Similar for credits / debits of Starbucks cards ...
Hierarchical Shuffles

We shuffle two (infinite) decks of cards.

Either or both of these are the result of previous shuffles of infinite decks of cards.
We take the obvious step of drawing this as a binary tree:

(As a slight simplification we assume the same shuffle at each step).
What we would like to do:

1. Write down the appropriate bijection:

\[ \mathbb{N} \times \{0, 1, \ldots, k\} \rightarrow \mathbb{N} \]

2. Give the corresponding Young tableaux.

3. Ensure (when appropriate) these are *standard* tableaux.

4. Transform the result of one tree of shuffles into another.
A binary code is a subset $A \subseteq \{0, 1\}^*$ such that the submonoid generated by $A$ is freely generated.

A simple example:
The set

$$L = \{00, 01, 10, 110, 111\} \subseteq \{0, 1\}^*$$

is a binary code.

Operationally: strings of elements of $L$ can be split up, uniquely, into elements of $L$.

$$11110000111001111$$

splits, uniquely, as

$$(111)(10)(00)(01)(110)(01)(111)$$
A maximal prefix code is a subset $A \subseteq \{0, 1\}$ where:

1. Members of $A$ are not prefixes of each other.
2. Every word of $\{0, 1\}^*$ either:
   - is a prefix of some element of $A$,
   - has some element of $A$ as a prefix.

Some elementary T.C.S.

There is a simple and well-known correspondence:

Complete binary trees \quad \equiv \quad Maximal prefix codes
leaf-traversal ≡ lex-ordering

Assuming $0 < 1$, the leaf-traversal, and lexicographic ordering coincide

$\{00 \leq 01 \leq 10 \leq 110 \leq 111\}$
Let’s label our tree branchings by \( \{p^\dagger, q^\dagger\} \) instead.
(Maximal) Prefix codes correspond to *embeddings*: Given a (maximal) prefix code $\mathcal{L}$ over $\{p^\dagger, q^\dagger\}^* \hookrightarrow P_2$, then for all $u^\dagger, v^\dagger \in L$,

$$uv^\dagger = \begin{cases} 
1 & u = v \\
0 & u \neq v
\end{cases}$$

(A one-sided version of the Kronecker delta ...)

Giving us an embedding $P_L \hookrightarrow P_2$. 
Assume a representation of $P_2$ based on the alternating Cantor point $a = 01010101\ldots$
(Equivalently, the perfect riffle shuffle ...)
We have
\[ p^+(n) = 2n \text{ and } q^+(n) = 2n + 1 \]

The max. prefix code
\[ \mathcal{L} = \{ p^+p^+, p^+q^+, q^+p^+, q^+q^+p^+, q^+q^+q^+ \} \]
gives us a $(\infty, \infty, \infty, \infty, \infty)$ Young tableau:
The tableau in question:

<table>
<thead>
<tr>
<th>$n =$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p^p p^p(n) =$</td>
<td>0</td>
<td>4</td>
<td>8</td>
<td>12</td>
<td>16</td>
<td>...</td>
</tr>
<tr>
<td>$p^q q^p(n) =$</td>
<td>2</td>
<td>6</td>
<td>10</td>
<td>14</td>
<td>18</td>
<td>...</td>
</tr>
<tr>
<td>$q^p p^p(n) =$</td>
<td>1</td>
<td>5</td>
<td>9</td>
<td>13</td>
<td>17</td>
<td>...</td>
</tr>
<tr>
<td>$q^q q^p(n) =$</td>
<td>3</td>
<td>11</td>
<td>19</td>
<td>27</td>
<td>31</td>
<td>...</td>
</tr>
<tr>
<td>$q^q q^q(n) =$</td>
<td>7</td>
<td>15</td>
<td>23</td>
<td>31</td>
<td>39</td>
<td>...</td>
</tr>
</tbody>
</table>

This is not a *standard* Young tableau

**What we have:**

1. Every natural number
2. Ordered rows
3. Unordered columns
The tableau in question:

<table>
<thead>
<tr>
<th>$n =$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p^t p^t(n) =$</td>
<td>0</td>
<td>4</td>
<td>8</td>
<td>12</td>
<td>16</td>
<td>...</td>
</tr>
<tr>
<td>$p^t q^t(n) =$</td>
<td>2</td>
<td>6</td>
<td>10</td>
<td>14</td>
<td>18</td>
<td>...</td>
</tr>
<tr>
<td>$q^t p^t(n) =$</td>
<td>1</td>
<td>5</td>
<td>9</td>
<td>13</td>
<td>17</td>
<td>...</td>
</tr>
<tr>
<td>$q^t q^t p^t(n) =$</td>
<td>3</td>
<td>11</td>
<td>19</td>
<td>27</td>
<td>31</td>
<td>...</td>
</tr>
<tr>
<td>$q^t q^t q^t(n) =$</td>
<td>7</td>
<td>15</td>
<td>23</td>
<td>31</td>
<td>39</td>
<td>...</td>
</tr>
</tbody>
</table>

This is not a *standard* tableau

What we have:

1. Every natural number
2. Ordered rows
3. Unordered columns
The question of trees

For any balanced Ballot Cantor point, $b \in \mathcal{B}$, the corresponding shuffle gives a standard $(\infty, \infty)$ Young tableau.

What were we thinking??

There is no reason to expect that an arbitrary hierarchical iteration of such shuffles should give a standard tableau.
The claim:

A necessary & sufficient condition for a hierarchical shuffle to give a *standard* tableaux is that:

the corresponding tree is *right-associated*.

\[
\begin{align*}
\{0, 1\} & \quad \{0, 10, 11\} & \quad \{0, 10, 110, 111\} & \quad \{0, 10, 110, 1110, 1111\}
\end{align*}
\]
Maximal prefix codes $R \subseteq \{0, 1\}$ need not be finite:

$$\begin{align*}
0 & \rightarrow 10 \\
10 & \rightarrow 110 \\
110 & \rightarrow 1110 \\
1110 & \rightarrow 11110 \\
\ldots
\end{align*}$$

$$\{0 \leq 10 \leq 110 \leq 1110 \leq 11110 \leq \ldots\}$$

The unique right-associated, well-ordered, infinite prefix code.
A worked example:

Let's do this for the representation of $P_2$ corresponding to the shuffle determined by the *alternating Cantor point* 

$$a = 0010101010101\ldots \in \mathcal{B}$$

Following the same procedure:

**Mapping prefix codes to polycyclic monoids**

$$0 \leftrightarrow p^\dagger \quad \text{and} \quad 1 \leftrightarrow q^\dagger$$

where

$$p^\dagger(n) = 2n \quad \text{and} \quad q^\dagger(n) = 2n + 1$$
We get our \((\infty, \infty, \infty, \ldots)\) standard Young tableau.

<table>
<thead>
<tr>
<th>(n = )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>\ldots</th>
</tr>
</thead>
<tbody>
<tr>
<td>(p^\dagger(n) = )</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>\ldots</td>
</tr>
<tr>
<td>(q^\dagger p^\dagger(n) = )</td>
<td>1</td>
<td>5</td>
<td>9</td>
<td>13</td>
<td>17</td>
<td>21</td>
<td>\ldots</td>
</tr>
<tr>
<td>((q^2)^\dagger p^\dagger(n) = )</td>
<td>3</td>
<td>11</td>
<td>19</td>
<td>27</td>
<td>35</td>
<td>43</td>
<td>\ldots</td>
</tr>
<tr>
<td>((q^3)^\dagger p^\dagger(n) = )</td>
<td>7</td>
<td>23</td>
<td>39</td>
<td>55</td>
<td>71</td>
<td>87</td>
<td>\ldots</td>
</tr>
<tr>
<td>((q^4)^\dagger p^\dagger(n) = )</td>
<td>15</td>
<td>47</td>
<td>79</td>
<td>111</td>
<td>143</td>
<td>175</td>
<td>\ldots</td>
</tr>
<tr>
<td>((q^5)^\dagger p^\dagger(n) = )</td>
<td>31</td>
<td>94</td>
<td>159</td>
<td>223</td>
<td>287</td>
<td>351</td>
<td>\ldots</td>
</tr>
<tr>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
</tr>
</tbody>
</table>

A (Hilbert-hotel style) bijection from \(\mathbb{N} \times \mathbb{N}\) to \(\mathbb{N}\), that is monotone in both variables:

\[
(r, c) \mapsto 2^c(2r + 1) - 1
\]
This is derived from:

*Alternating shuffles of decks of cards.*

We start by shuffling two Decks $A$ and $B$.

Deck $B$ arose from shuffling Decks $B'$ and $C$.

Deck $C$ arose from shuffling Decks $C'$ and $D$.

Deck $D$ arose from shuffling Decks $D'$ and $E$.

...\[
:\]

**Practically** – is it easy / possible to perform this shuffle?
Deep fractal structure ??

Play a card from the following decks, in order:

0 1 0 2 0 1 0 3 0
1 0 2 0 1 0 4 0 1
0 2 0 1 0 3 0 1 0
2 0 1 0 5 0 1 0 2
0 1 0 3 0 1 0 2 0 ...

Question: How may we characterise this sequence?
Which deck do we play from, at each step?

<table>
<thead>
<tr>
<th>Step</th>
<th>Deck₀</th>
<th>Deck₁</th>
<th>Deck₂</th>
<th>Deck₃</th>
<th>Deck₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td></td>
<td>•</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td>•</td>
<td></td>
<td>•</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td></td>
<td>•</td>
<td></td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>Step 5</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td>•</td>
</tr>
<tr>
<td>Step 6</td>
<td></td>
<td></td>
<td>•</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Step 7</td>
<td>•</td>
<td></td>
<td></td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>Step 8</td>
<td></td>
<td>•</td>
<td></td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>Step 9</td>
<td>•</td>
<td></td>
<td>•</td>
<td></td>
<td>•</td>
</tr>
<tr>
<td>Step 10</td>
<td></td>
<td>•</td>
<td></td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>Step 11</td>
<td>•</td>
<td></td>
<td>•</td>
<td></td>
<td>•</td>
</tr>
<tr>
<td>Step 12</td>
<td></td>
<td>•</td>
<td></td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>Step 13</td>
<td>•</td>
<td>•</td>
<td></td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>Step 14</td>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>Step 15</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Step 16</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
</tbody>
</table>

peter.hines@york.ac.uk  www.peterhines.info
This looks kind of familiar!

<table>
<thead>
<tr>
<th>Step</th>
<th>2^4</th>
<th>2^3</th>
<th>2^2</th>
<th>2^1</th>
<th>2^0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Step 2</td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Step 5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Step 6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Step 7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Step 8</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Step 9</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Step 10</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Step 11</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Step 12</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Step 13</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Step 14</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Step 15</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Step 16</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Performing the perfect infinite riffle

A very simple rule

1. Count in binary ...
2. Which bit has changed from 0 to 1?
3. Play a card from that deck!
We can do the same with any $b \in \mathcal{B}$

Each Balanced Ballot point determines a distinct:

- $(\infty, \infty, \infty, \ldots)$ standard Young tableau.
- shuffle of infinitely many decks of cards, satisfying:
  
  “Number of cards played from $\text{Deck}_i$ is always $\geq$ Number of cards played from $\text{Deck}_{i+1}$”

- bijection $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ monotone in both variables.

Using inverse semigroup theory

It is straightforward to describe the mappings between these:

$$
\begin{array}{ccc}
\mathbb{N} \times \mathbb{N} & \xrightarrow{\phi \text{ monotone}} & \mathbb{N} \\
\downarrow & & \downarrow \\
\mathbb{N} & \xrightarrow{\psi \text{ monotone}} & \mathbb{N}
\end{array}
$$

Exercise : What group do we get??

peter.hines@york.ac.uk  
www.peterhines.info
Every balanced Ballot Cantor point determines an \((\infty, \infty, \infty, \ldots)\) standard Young tableau. These are equivalent to:

*Infinite inclusion-ordered chains of finite standard Young tableaux.*

For the alternating Cantor point:

<table>
<thead>
<tr>
<th>n=0</th>
<th>n=1</th>
<th>n=2</th>
<th>n=3</th>
<th>n=4</th>
<th>n=5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

...just a complicated way of counting in binary!
Let us fix some balanced Cantor point \( c \in \mathcal{C} \)
(We do not assume it is a Ballot point!)

Given trees \( Tree_1, Tree_2 \), both with \( k \) leaves, how can we transform the result of one hierarchical shuffle into the other?

\[
\mathbb{N} \times \{0, \ldots, k - 1\}
\]

\[\text{Tree}_1 \quad \text{Tree}_2\]

\[\mathbb{N} \quad ?? \quad \mathbb{N}\]
Tree_1 and Tree_2 both determine k-element maximal prefix codes over \{p^+, q^+\}. Call these

\[ R = \{r_0^+, \ldots, r_{k-1}^+\} \text{ and } S = \{s_0^+, \ldots, s_{k-1}^+\} \]

The required bijection is simply:

\[ s_0^+ r_0 \cup \ldots s_j^+ r_j \ldots \cup \ldots \cup s_{k-1}^+ r_{k-1} \]

The intuition

Element \(r_j\) maps the \(j^{th}\) row of a Young tableau

to the whole of \(\mathbb{N}\)

Element \(s_j^+\) maps the whole of \(\mathbb{N}\)
to the \(j^{th}\) row of another Young tableau.
For an *arbitrary* effective representation of $P_2$,

The set of *all* such bijections (including varying $k \in \mathbb{N}$) is closed under compositions and inverses.

**Question:** Which group is this?
This has already been shown, as a study in abstract algebra / semigroup-theory, to be Thompson’s group $\mathcal{F}$ in

“The Polycyclic Monoids & The Thompson Groups”

A corollary

Each *balanced Cantor point* uniquely determines a representation of $\mathcal{F}$ as bijections on $\mathbb{N}$.

Does ‘anything special’ happen when we choose a Ballot point?
A particularly ‘significant’ group:

Thompson’s group $\mathcal{F}$

- one of the best-known groups in mathematics
- defined in 1965, as a potential counter-example to a conjecture of von Neumann
- a rich source of conjectures & counterexamples
- has linear-time word problem
- closely connected to both complexity and category theory
- proposed (2004) as a platform for non-commutative cryptography
The group $\mathcal{F}$ was originally defined via representations.

Abstractly, it may be defined as the group with:

- A countably infinite set of generators $\{x_0, x_1, x_2, \ldots\}$
- Relations given by

$$x_k^{-1}x_nx_k = x_{n+1} \text{ for all } k < n$$

(Other presentations are possible, but this is the most intuitive / natural)
Some explicit calculations ...

To which tree re-arrangements do these correspond?

$X_0$ performs:

$$
\begin{array}{c}
0 \\
10 \\
11
\end{array}
\Rightarrow
\begin{array}{c}
00 \\
01 \\
1
\end{array}
$$

$X_1$ performs:

$$
\begin{array}{c}
0 \\
10 \\
110 \\
111
\end{array}
\Rightarrow
\begin{array}{c}
0 \\
100 \\
101 \\
11
\end{array}
$$

$X_2$ performs:

$$
\begin{array}{c}
0 \\
10 \\
110 \\
1110 \\
1111
\end{array}
\Rightarrow
\begin{array}{c}
0 \\
10 \\
1100 \\
1101 \\
111
\end{array}
$$

$X_3$ performs: ...
Let us consider shuffles defined by a balanced \textbf{ballot} point.

For convenience, we again consider the alternating point:

\[ a = 010101010101\ldots \in \mathcal{B} \]

\textbf{Important:}

this is for illustration;

other balanced ballot points will do!

\textit{What is immediately noticeable?}
Consider (the representation of) each generator as ‘mapping the results of one (hierarchical) shuffle into another’.

**Generator** $x_j$ re-arranges the result of $S_j$ into that of $T_j$.

- $S_{j+1}$ is obtained by using the result of $S_j$ as the 2nd deck in an alternating shuffle (and similarly for $T_{j+1}$).
- Each $S_j$ is the **unique** hierarchical shuffle that gives a **standard** $(\infty, \infty, \ldots, \infty)$ Young tableau.
- Each $S_j$ is re-arranged into $T_j$ by a single *rotation* or *associator*

\[
\begin{array}{c}
\bullet \\
\bullet \\
\bullet \\
\end{array} 
\Rightarrow 
\begin{array}{c}
\bullet \\
\bullet \\
\bullet \\
\end{array} 
\]

on its final three leaves.
Generators converge to the identity

Generator \( x_j \) is the *identity* on the first \( j - 1 \) rows of our \((\infty, \infty, \ldots)\) standard Young tableau:

<table>
<thead>
<tr>
<th>( n = )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>( \ldots )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p\uparrow(n) = )</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( q\uparrow p\uparrow(n) = )</td>
<td>1</td>
<td>5</td>
<td>9</td>
<td>13</td>
<td>17</td>
<td>21</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( (q^2)\uparrow p\uparrow(n) = )</td>
<td>3</td>
<td>11</td>
<td>19</td>
<td>27</td>
<td>35</td>
<td>43</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( (q^3)\uparrow p\uparrow(n) = )</td>
<td>7</td>
<td>23</td>
<td>39</td>
<td>55</td>
<td>71</td>
<td>87</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( (q^4)\uparrow p\uparrow(n) = )</td>
<td>15</td>
<td>47</td>
<td>79</td>
<td>111</td>
<td>143</td>
<td>175</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( (q^5)\uparrow p\uparrow(n) = )</td>
<td>31</td>
<td>94</td>
<td>159</td>
<td>223</td>
<td>287</td>
<td>351</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
</tr>
</tbody>
</table>

\[
\lim_{n \to \infty} (x_n) = I_\mathbb{N}
\]

Formally, this is a point-wise limit:

\[
\forall a \in \mathbb{N}, \exists T \in \mathbb{N} \text{ such that } n \geq T \Rightarrow x_n(a) = a
\]
The **NP**-intermediate class

The complexity class **NPI** is the class of problems that are:

- in **NP**,
- not in **P**,
- not **NP**-complete.

**Ladner’s Theorem** (1975)

**NPI** is non-empty $\iff P \neq NP$
Some (possibly) **NPI** problems

We cannot say, for certain, that *any* problem is in **NPI**.

Some ‘promising candidates’

- Prime factorisation.
- Deciding graph isomorphism.
- Finding the ‘rotation distance’ between two trees.
- Computing discrete logarithms, and related problems.

Ladner produced ‘highly artificial’ problems that are guaranteed to be in **NPI**, provided \( P \neq NP \).

No ‘natural’ problems with the same property are known.
The tree rotation distance problem

A rotation of binary trees is a local tree transformation of the form:

\[ \begin{array}{c}
A & B & C \\
\end{array} \leftrightarrow \begin{array}{c}
A & B & C \\
\end{array} \]

where A, B and C may be leaves or subtrees.

Sleator, Tarjan & Thurston (1988)
Any \( n \)-node tree can be transformed into any other \( n \)-node tree using a maximum of \( 2n - 6 \) rotations.

Čulík and Wood’s problem (1982)
Given two trees, what is the shortest sequence of rotations that will transform one into the other?
The notion of rotation

Let us fix some (possibly Ballot) balanced Cantor point

A rotation, applied to a tree $S$, gives another tree $T$.
Together, this pair of trees determines an element of $\mathcal{F}$.
The generators $\{X_0, X_1, \ldots\}$ are of this form.

Can we re-write Čulík and Wood’s problem as a question about words in $\mathcal{F}$?
“On the rotation distance between binary trees”
– P. Dehornoy (2009)

“...introducing a partial action of $\mathcal{F}$ on trees and expressing the rotation distance between two trees as the length of an element of $\mathcal{F}$.

This approach easily leads to a lower bound. However, due to the lack of control on the geometry of $\mathcal{F}$, it seems difficult to obtain higher lower bounds .... ”
Given a rotation:

\[
\begin{array}{c}
A \\
B \\
C
\end{array} \quad \leftrightarrow \quad \begin{array}{c}
A \\
B \\
C
\end{array}
\]

we get the \textit{same} element of $\mathcal{F}$, for \textit{arbitrary} $A$, $B$ and $C$.

\begin{itemize}
  \item Mapping rows between Young tableaux.
  \item Splitting a single row into two.
  \item Merging two rows into one.
\end{itemize}

\textit{We have a more ‘fine-grained’ control, using inverse semigroups instead.}
We can take this further, but at some point, we are forced to interpret as **Category theory:**

**Coherence for associativity**  An entire field based on the study of associators (rotations).

**Higher categorical coherence**  Operads & related structures.

**Symmetries of Polyhedra**  Associahedra, permutahedra, &c. via group and inverse semigroup theory.

More at N.Y.C. category theory seminar, 10\(^{th}\) of Feb., 2021