PatternClass
A GAP Package for Permutation Pattern Classes

Ruth Hoffmann

University of St Andrews, School of Computer Science

Permutation Patterns 2012
University of Strathclyde
13th June 2012

University of St Andrews

http://www.gap-system.org/
http://www.cs.st-andrews.ac.uk/~ruthh/

```
gap> LoadPackage("PatternClass");
----------------------------------------------------------------
Loading  Automata 1.12
For help, type: ?Automata:
----------------------------------------------------------------
----------------------------------------------------------------
Loading  PatternClass 1.0
For help, type: ?PatternClass:
----------------------------------------------------------------
true
```

Regular Permutation Pattern Classes

Definition

The *rank encoding* of a permutation $\pi = \pi_1 \dots \pi_n$ is the sequence $E(\pi) = p_1 \dots p_n$ where p_i is the rank of π_i among $\{\pi_i, \pi_{i+1}, \dots, \pi_n\}$.

Definition

Ω_k is the class of permutations which in their rank encoding have highest rank k.

Theorem ([AAR03])

$E(\Omega_k)$ is a regular language.

Definition

If a pattern class $\mathcal{C}$ is a regular subset of Ω_k for some k, we call $\mathcal{C}$ a *regular pattern class*.

Let,

- $\mathcal{C}$ any regular pattern class.
- $\mathcal{B}$ the basis of $\mathcal{C}$.
- $\mathcal{D}$ a transducer that deletes an arbitrary letter in a rank encoded permutation.
- $\mathcal{H}$ a transducer that deletes an arbitrary number of letters in a rank encoded permutation.

Then in [AAR03] it has been found that

$$E(\mathcal{B}) = (E(\mathcal{C}))^C \cap ((E(\mathcal{C}))^C \mathcal{D}^t)^C$$

and also

$$E(\mathcal{C}) = (E(\mathcal{B})\mathcal{H}^t)^C \cap E(\Omega_k).$$

Regular Permutation Pattern Classes

```
gap> b:=BoundedClassAutomaton(3);
< deterministic automaton on 3 letters with 3 states >
gap> AutToRatExp(b);
((cc*(aUb)Ub)(cc*(aUb)Ub)*aUa)*
gap> a:=ClassAutFromBase([[3,1,2]],3);
< deterministic automaton on 3 letters with 4 states >
gap> AutToRatExp(a);
((cc*bUb)(cc*bUb)*aUa)*
gap> ba:=BasisAutomaton(a);
< deterministic automaton on 3 letters with 5 states >
gap> AutToRatExp(ba);
c(aaU@)Ub
```

Token Passing Networks

Definition

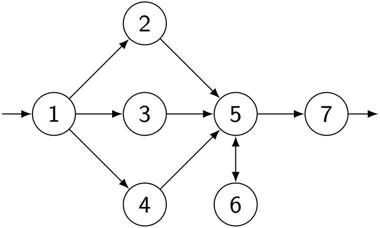
A *token passing network* is a directed graph G with a designated input and a designated output node, where each node of G can hold at most one token.

The output of a token passing network is a set of permutations of the input sequences $1, 2, \ldots, n$ where $n \in \mathbb{N}$.
The set of permutations output by a token passing network is closed under the relation of containment and is thus a permutation pattern class.[ARL04]

Theorem ([ALT97])

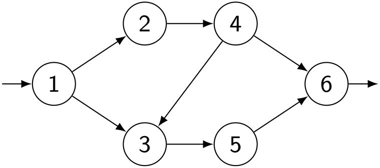
The class $\mathcal{C}(G)$ of permutations output by the token passing network G is a regular pattern class.

Token Passing Networks




```
gap> BufferAndStack(3,2);
[ [ 2 .. 4 ], [ 5 ], [ 5 ], [ 5 ], [ 6, 7 ], [ 5 ], [  ] ]
```

Token Passing Networks




```
gap> hex:=[[2,3],[4],[5],[3,6],[6],[]];
[ [ 2, 3 ], [ 4 ], [ 5 ], [ 3, 6 ], [ 6 ], [  ] ]
```

Token Passing Networks

```
gap> g:=BufferAndStack(3,2);
[ [ 2 .. 4 ], [ 5 ], [ 5 ], [ 5 ], [ 6, 7 ], [ 5 ], [  ] ]
gap> a:=GraphToAut(g,1,7);
< epsilon automaton on 5 letters with 460 states >
gap> a:=MinimalAutomaton(a);
< deterministic automaton on 4 letters with 4 states >
gap> g1:=BufferAndStack(4,3);
[ [ 2 .. 5 ], [ 6 ], [ 6 ], [ 6 ], [ 6 ], [ 7, 9 ], [ 6, 8 ],
[ 7 ], [  ] ]
gap> a1:=GraphToAut(g1,1,9);
< epsilon automaton on 7 letters with 14680 states >
gap> a1:=MinimalAutomaton(a1);
< deterministic automaton on 6 letters with 19 states >
gap> AutToRatExp(a);
(((dd*(aUbUc)Uc)(dd*(aUbUc)Uc)*(aUb)Ub)((dd*(aUbUc)Uc)(dd*\
(aUbUc)Uc)*(aUb)Ub)*aUa)*
```

Some Handy Functions

Definition ([ALR05])

The *spectrum* of a pattern class $\mathcal{C}$ is the sequence $(|\mathcal{C} \cap S_n|)_{n=1}^{\infty}$, where S_n is the set of all permutations of length n.

Some Handy Functions

```
gap> hex:=[[2,3],[4],[5],[3,6],[6],[]];;
gap> hexaut:=MinimalAutomaton(GraphToAut(hex,1,6));;
gap> Spectrum(hexaut);
[ 1, 2, 6, 18, 54, 161, 477, 1408, 4148, 12208, 35912, 105617,
  310585, 913282, 2685462 ]
gap> basisaut:=BasisAutomaton(hexaut);
< deterministic automaton on 3 letters with 11 states >
gap> AutToRatExp(basisaut);
c(b(ca(ca)*cbaUbcba)U@)Ub
gap> Spectrum(basisaut);
[ 2, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1 ]
gap> NumberAcceptedWords(hexaut,10);
12208
gap> AcceptedWords(hexaut,4);
[ [ 1, 1, 1, 1 ], [ 1, 1, 2, 1 ], [ 1, 2, 1, 1 ], [ 1, 2, 2, 1 ],
  [ 1, 3, 1, 1 ], [ 1, 3, 2, 1 ], [ 2, 1, 1, 1 ], [ 2, 1, 2, 1 ],
  [ 2, 2, 1, 1 ], [ 2, 2, 2, 1 ], [ 2, 3, 1, 1 ], [ 2, 3, 2, 1 ],
  [ 3, 1, 1, 1 ], [ 3, 1, 2, 1 ], [ 3, 2, 1, 1 ], [ 3, 2, 2, 1 ],
  [ 3, 3, 1, 1 ], [ 3, 3, 2, 1 ] ]
```

Some Handy Functions

```
gap> RankDecoding([ 2, 3, 1, 1 ]);
[ 2, 4, 1, 3 ]
gap> RankEncoding([ 3, 4, 5, 2, 6, 1, 7, 8 ]);
[ 3, 3, 3, 2, 2, 1, 1, 1 ]
gap> IsRankEncoding([3,2,3,5,2,1]);
false
gap> IsRankEncoding([3,3,3,2,2,1,1,1]);
true
```

Properties of Permutations

Definition

An *interval* of a permutation is a set of contiguous values of consecutive indices of the permutation.

Definition

A permutation of length n is called *simple* if it only contains the intervals of length 0, 1, and n.

Definition

A *block decomposition* of a permutation σ is $\sigma = \pi[\alpha_1, \ldots, \alpha_n]$ where π is of length n.

Properties of Permutations

Definition

A permutation σ is said to be *plus-decomposable* if it can be uniquely written as $\sigma = 12[\alpha_1, \alpha_2]$, where α_1 is not plus-decomposable.

Definition

A permutation σ is said to be *minus-decomposable* if it can be uniquely written as $\sigma = 21[\alpha_1, \alpha_2]$, where α_1 is not minus-decomposable.

Properties of Permutations

```
gap> IsPlusDecomposable([3,3,2,3,2,2,1,1]);
true
gap> IsPlusDecomposable([3,3,3,3,3,3,2,1]);
false
gap> IsMinusDecomposable([3,3,3,3,3,3,2,1]);
true
gap> IsSimplePerm([3,1,2,3,1,3,1,1]);
true
```

Subsets

Plus-decomposable permutations $\pi = \pi(1)\dots\pi(n) = 12[\alpha_1, \alpha_2]$ with $|\alpha_1| = x$, $|\alpha_2| = n - x$, have the following properties,

- $\pi(1)\dots\pi(x) = \alpha_1$ and $\pi(x+1)\dots\pi(n)$ is order isomorphic to α_2.
- Under the rank encoding α_1 and α_2 will be $E(\pi(1)\dots\pi(x)) = E(\alpha_1)$ and $E(\pi(x+1)\dots\pi(n)) = E(\alpha_2)$.

Subsets

```
gap> a:=PlusDecompAut(hexaut);
< deterministic automaton on 3 letters with 14 states >
gap> Spectrum(a);
[ 0, 1, 3, 11, 37, 121, 385, 1200, 3684, 11184, 33672, 100753,
  300089, 890754, 2637334 ]
gap> b:=PlusIndecompAut(hexaut);
< deterministic automaton on 3 letters with 10 states >
gap> Spectrum(b);
[ 1, 1, 3, 7, 17, 40, 92, 208, 464, 1024, 2240, 4864, 10496, 22528,
  48128 ]
gap> Spectrum(hexaut);
[ 1, 2, 6, 18, 54, 161, 477, 1408, 4148, 12208, 35912, 105617,
  310585, 913282, 2685462 ]
```

Subsets

Let $\pi \in \mathcal{M}$ be a k-bounded minus-decomposable permutation of length n. Then $E(\pi)$ consists of $n - d$ letters that are $> d$ followed by d letters $\leq d$, where $d \in \mathbb{N}$.

Subsets

```
gap> a:=MinusDecompAut(hexaut);
< deterministic automaton on 3 letters with 10 states >
gap> Spectrum(a);
[ 0, 1, 3, 5, 9, 16, 27, 43, 65, 94, 131, 177, 233, 300, 379 ]
gap> b:=MinusIndecompAut(hexaut);
< deterministic automaton on 3 letters with 16 states >
gap> Spectrum(b);
[ 1, 1, 3, 13, 45, 145, 450, 1365, 4083, 12114, 35781, 105440,
  310352, 912982, 2685083 ]
```

Further Development

Open Question

Does the subset of simple permutations of a regular pattern class build a regular language under the rank encoding?

Next Functions

- One point deletion in simple permutations
- Direct and skew sum of permutations
- Calculation of the direct sum of classes
- Unique block-decomposition of permutations
- Different encodings, e.g. Insertion Encoding
- Grid Classes

Literature

- M.H. Albert, M.D. Atkinson, and N. Ruškuc, *Regular closed sets of permutations*, Theoretical Computer Science **306** (2003), no. 1–3, 85 – 100.
- Michael H. Albert, Steve Linton, and Nik Ruškuc, *The insertion encoding of permutations*, Electron. J. Combin **12** (2005).
- M. D. Atkinson, M. J. Livesey, and D. Tulley, *Permutations generated by token passing in graphs*, Theoretical Computer Science **178** (1997), no. 1–2, 103 – 118.
- M. H. Albert, N. Ruškuc, and S. Linton, *On the permutational power of token passing networks*, Tech. report, 2004.
- The GAP Group, *GAP – Groups, Algorithms, and Programming, Version 4.4.12*, 2008.
- Takeaki Uno and Mutsunori Yagiura, *Fast algorithms to enumerate all common intervals of two permutations*, Algorithmica **26** (2000), 2000.