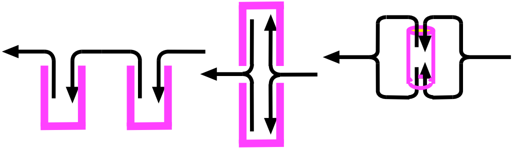
Stacks and Deques

Michael Albert[1] Mike Atkinson[1] Steve Linton[2]

[1]Department of Computer Science, University of Otago

[2]School of Computer Science, University of St Andrews

PP2008, Otago, June 2008



Outline of talk

1. Early pattern class history
2. Upper bounds
3. Lower bounds
4. Conclusions and questions

Three early landmarks

D.E. Knuth:
Fundamental Algorithms, The Art of Computer Programming
Vol. 1 (First Edition), especially §2.2.1
Addison-Wesley, Reading, Mass. (1968).

R.E. Tarjan:
Sorting using networks of queues and stacks,
Journal of the ACM 19 (1972), 341–346.

V.R. Pratt:
Computing permutations with double-ended queues, parallel stacks and parallel queues,
Proc. ACM Symp. Theory of Computing 5 (1973), 268–277.

Data Structures

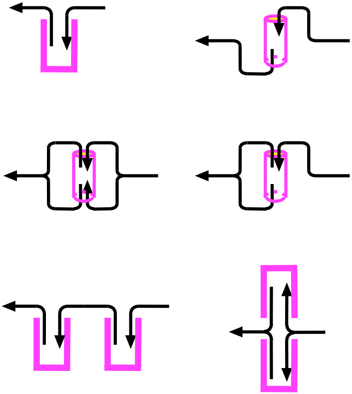
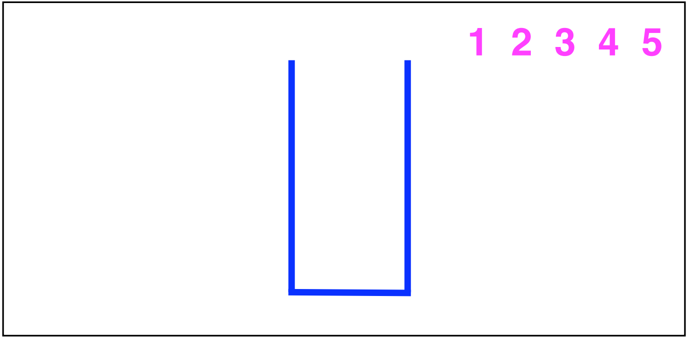
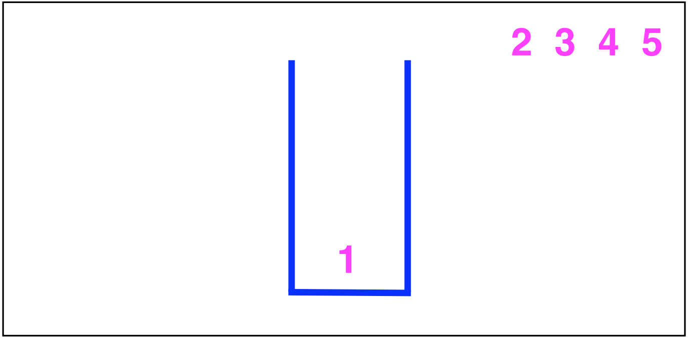


Figure: What permutations can a data structure generate (or sort)?

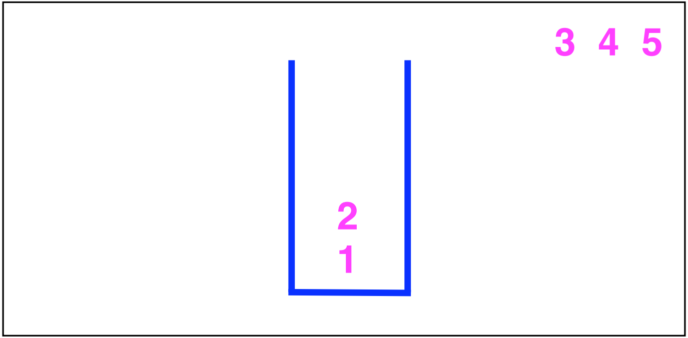
Generating a permutation



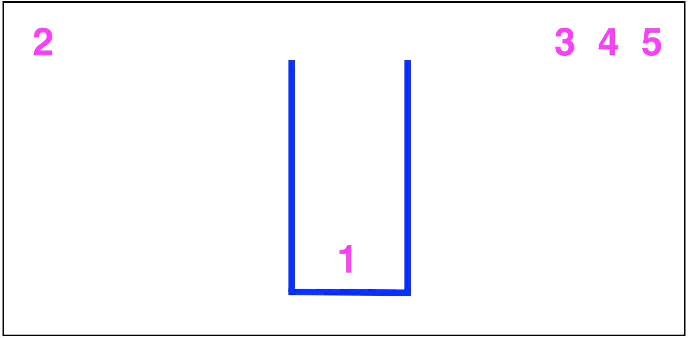
Generating a permutation



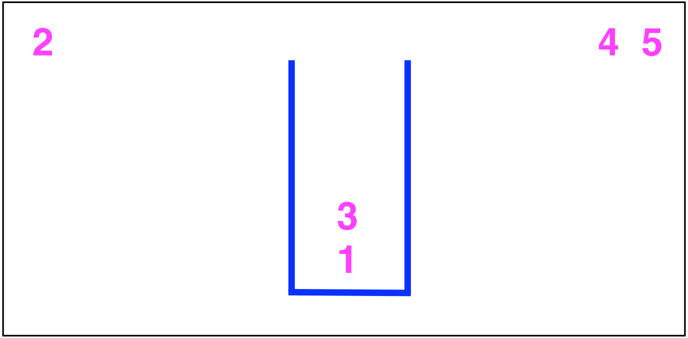
Generating a permutation



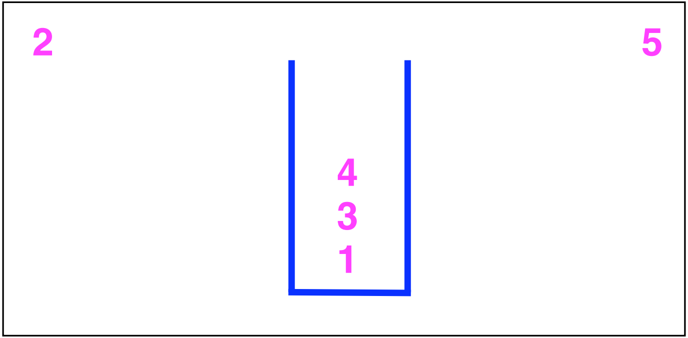
Generating a permutation



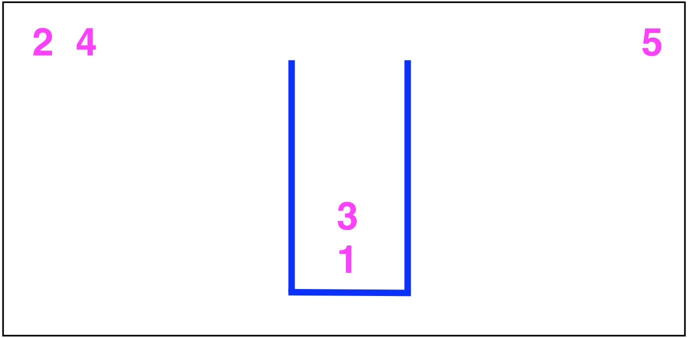
Generating a permutation



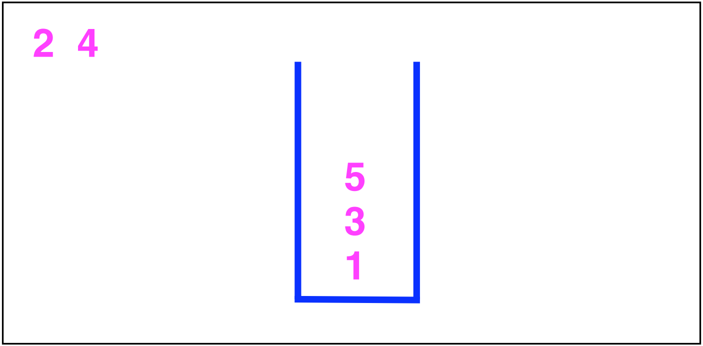
Generating a permutation



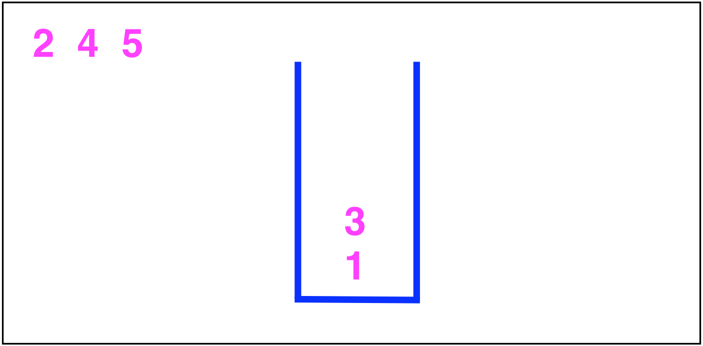
Generating a permutation



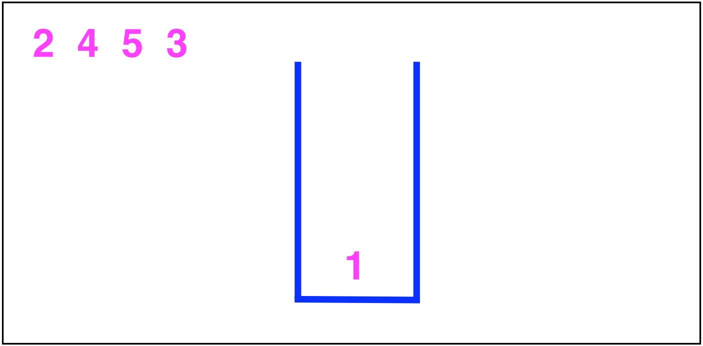
Generating a permutation



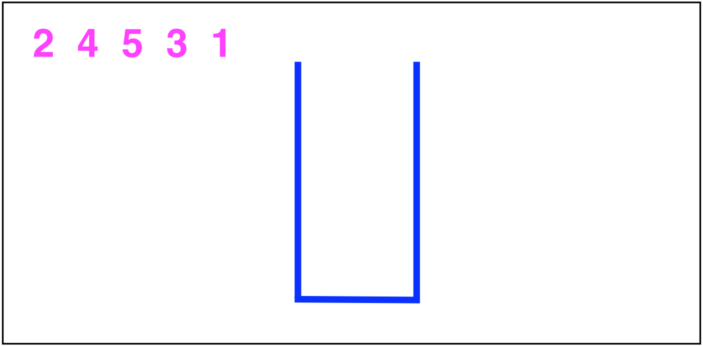
Generating a permutation



Generating a permutation



Generating a permutation



Knuth

- Enumeration of stack permutations and the 312-avoidance criterion

Knuth

- Enumeration of stack permutations and the 312-avoidance criterion
- Enumeration of restricted-input deque permutations, showed they avoid $\{4213, 4231\}$

Knuth

- Enumeration of stack permutations and the 312-avoidance criterion
- Enumeration of restricted-input deque permutations, showed they avoid $\{4213, 4231\}$
- Considered stacks in series

Knuth

- Enumeration of stack permutations and the 312-avoidance criterion
- Enumeration of restricted-input deque permutations, showed they avoid $\{4213, 4231\}$
- Considered stacks in series
- Exercise 2.2.1.13: " *[M48] How many permutations of n elements are obtainable with the use of a general deque?*"

Tarjan

Focused on minimal unsortable permutations for various networks of data structures – nowadays called the basis problem.

- Lemma 6: "*There is an infinite set of permutations, none of which contains another as a pattern, and such that each permutation is unsortable using two parallel stacks*"

Tarjan

Focused on minimal unsortable permutations for various networks of data structures – nowadays called the basis problem.

- Lemma 6: "*There is an infinite set of permutations, none of which contains another as a pattern, and such that each permutation is unsortable using two parallel stacks*"
- Lemma 10: "*Let Y be a series of* 2 *stacks. Then the shortest unsortable sequence in Y is of length* 7."

Tarjan

Focused on minimal unsortable permutations for various networks of data structures – nowadays called the basis problem.

- Lemma 6: "*There is an infinite set of permutations, none of which contains another as a pattern, and such that each permutation is unsortable using two parallel stacks*"
- Lemma 10: "*Let Y be a series of* 2 *stacks. Then the shortest unsortable sequence in Y is of length* 7*.*"
- (End of paper): "*The author has constructed a sequence of length* 41 *which is unsortable using three stacks in series; beyond this ... getting hard*".

Pratt

- Formalised the subpermutation relation: "*. . . the subtask relation on permutations is even more interesting than the networks we were characterizing. This relation seems to be the only partial order on permutations that arises in a simple and natural way, yet it has received no attention to date.*"

Pratt

- Formalised the subpermutation relation: "...*the subtask relation on permutations is even more interesting than the networks we were characterizing. This relation seems to be the only partial order on permutations that arises in a simple and natural way, yet it has received no attention to date.*"
- Proved $\{4213, 4231\}$ is the basis of the restricted-input deque permutations

Pratt

- Formalised the subpermutation relation: "*... the subtask relation on permutations is even more interesting than the networks we were characterizing. This relation seems to be the only partial order on permutations that arises in a simple and natural way, yet it has received no attention to date.*"
- Proved $\{4213, 4231\}$ is the basis of the restricted-input deque permutations
- Found the (infinite) bases of the class of general deque permutations and the basis of the two parallel stacks class

Pratt

- Formalised the subpermutation relation: "*. . . the subtask relation on permutations is even more interesting than the networks we were characterizing. This relation seems to be the only partial order on permutations that arises in a simple and natural way, yet it has received no attention to date.*"
- Proved $\{4213, 4231\}$ is the basis of the restricted-input deque permutations
- Found the (infinite) bases of the class of general deque permutations and the basis of the two parallel stacks class
- Used formal languages in enumeration results

Data Structures

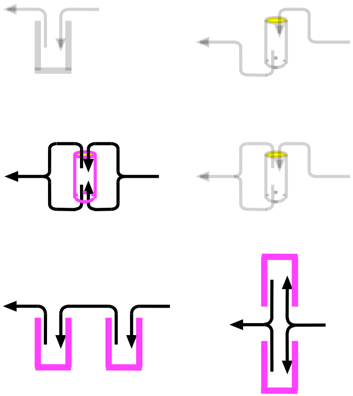


Figure: Data Structures with unknown enumerations

Growth rates

The growth rate of a sequence (c_n) is

$$g = \limsup_{n \to \infty} \sqrt[n]{c_n}$$

(so c_n behaves roughly like g^n)

Question

What is the growth rate for deques, two stacks in parallel, two stacks in series?

It is known that, in all three cases, the growth rate is between 4 and 16. Also the $\limsup$ is a true limit.

Upper bounds on growth rates – Stacks in series

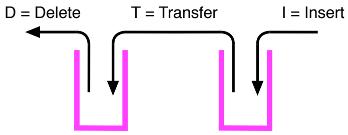


Figure: Two stacks in series

- *IIITITDTDDTD* produces 4231 from input 1234

Upper bounds on growth rates – Stacks in parallel

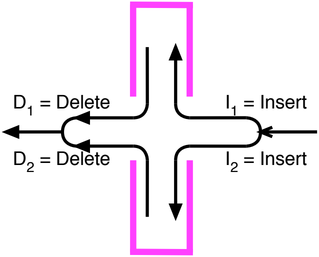


Figure: Two stacks in parallel

- $I_1 I_1 I_2 D_1 I_2 I_1 D_2 I_1 D_2 D_2 D_1 D_1$ produces 24351 from input 12345

Upper bounds on growth rates – Deques

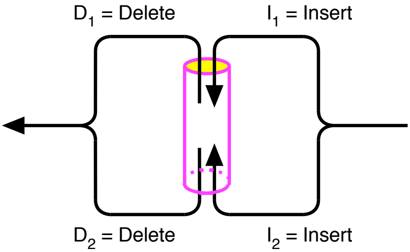


Figure: Deque

- $I_1 I_2 I_1 D_2 I_2 I_2 I_1 D_2 D_1 D_2 D_2 D_2 D_1$ produces 256413

Permutations as words

- Represent permutations by words over a 3 or 4 letter alphabet and count words. This is an overcount since
  1. Not every word represents a permutation, and
  2. Many words represent the same permutation
- The first of these doesn't seem to matter for growth rates. E.g. For two stacks in series there are 27^n words of length $3n$ on $\{I, D, T\}$ but only

$$\frac{12.(3n)!}{(n+2)!(n+1)!n!}$$

  of them represent permutations. This has growth rate 27. The second is much more serious.

Rewriting rules

Definition

If L, R are words then $L \longrightarrow R$ if any permutation which can be generated by a word ULV is also generated by URV.

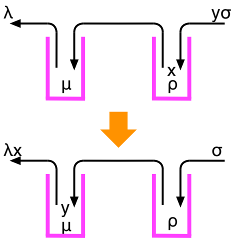


Figure: $TDIT \longrightarrow ITTD$

Systematic search for rewriting rules of length t

- Suppose we know rewriting rules for lengths less than t
- Construct FSA recognising all words not containing any LHS of existing rules (a regular language)
- Run through all words of length t that it accepts, compute their effect on a “generic” state of the data structure, and sort them by their effects
- Create new rules from any duplicate effects
- Check the rules are valid in “non-generic” settings.

Word count

- Having found all rewriting rules of length up to t construct FSA accepting all words not containing any LHS of a rule
- Use state equations to find generating functions and thereby growth rate for the number of words
- This will be an upper bound on the growth rate for the number of permutations

Results – Deque

Length	Number of Rules	Growth Bound
8	51	8.4925
9	85	8.459
10	175	8.428
11	321	8.410
12	756	8.392
13	1480	8.380
14	3806	8.368
15	7734	8.361
16	21029	8.352

Results – Parallel Stacks

Length	Number of Rules	Growth Bound
8	33	8.4606
9	43	8.4474
10	109	8.4087
11	143	8.4031
13	615	8.376
14	2366	8.3597
15	3131	8.3578
16	13263	8.3461

Results – Two Stacks in Series

Length	Number of Rules	Growth Bound
8	23	14.201
9	35	14.048
10	71	13.826
11	106	13.747
13	215	13.623
14	368	13.477
15	1270	13.433
16	2825	13.374

Lower bounds – via bounded capacities

- Consider k-bounded versions of the three structures where the system is constrained to contain at most k elements at a time.

Lower bounds – via bounded capacities

- Consider k-bounded versions of the three structures where the system is constrained to contain at most k elements at a time.
- This corresponds to imposing extra forbidden patterns of the form $(k+1)\alpha$ where α is any permutation of $\{1 \cdots k\}$.

Lower bounds – via bounded capacities

- Consider k-bounded versions of the three structures where the system is constrained to contain at most k elements at a time.
- This corresponds to imposing extra forbidden patterns of the form $(k+1)\alpha$ where α is any permutation of $\{1 \cdots k\}$.
- The system can now be thought of as FSA with states that correspond to the disposition of elements residing in the stacks/deque.

Lower bounds – via bounded capacities

- Consider k-bounded versions of the three structures where the system is constrained to contain at most k elements at a time.
- This corresponds to imposing extra forbidden patterns of the form $(k+1)\alpha$ where α is any permutation of $\{1 \cdots k\}$.
- The system can now be thought of as FSA with states that correspond to the disposition of elements residing in the stacks/deque.
- It outputs rank-encoded permutations: e.g. 4163752 is encoded as 4142321 – and the ranks will be at most k

Bounded deque

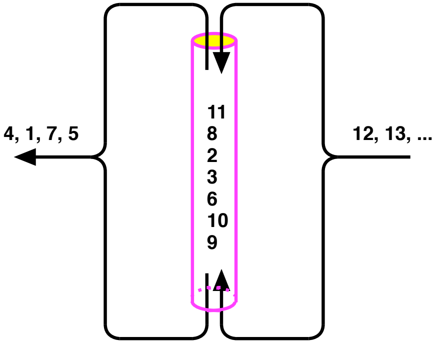


Figure: Bounded deque: snapshot

Bounded deque

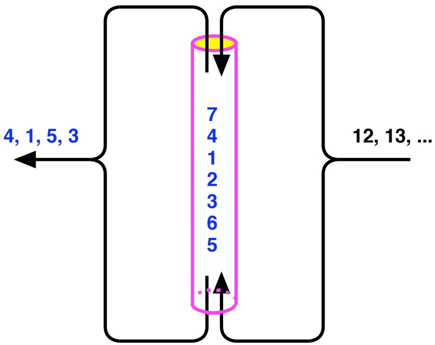


Figure: Bounded deque: encoded snapshot

Bounded deque

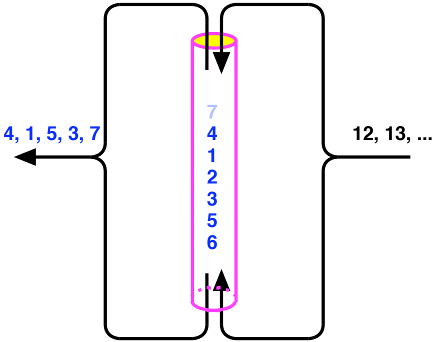


Figure: Bounded deque: encoded snapshot after D_1

Bounded deque

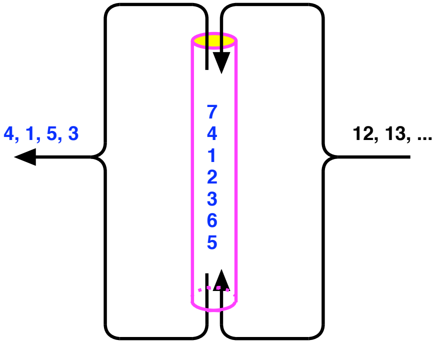


Figure: Bounded deque: encoded snapshot

Bounded deque

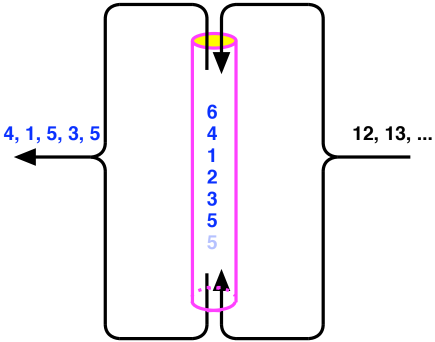


Figure: Bounded deque: encoded snapshot after D_2

Bounded deque

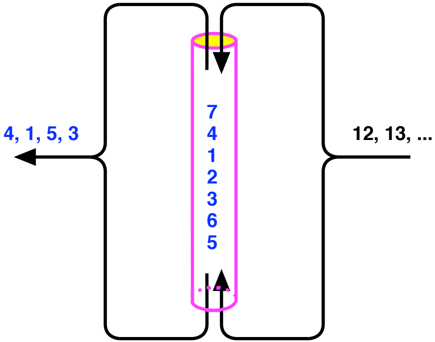


Figure: Bounded deque: encoded snapshot

Bounded deque

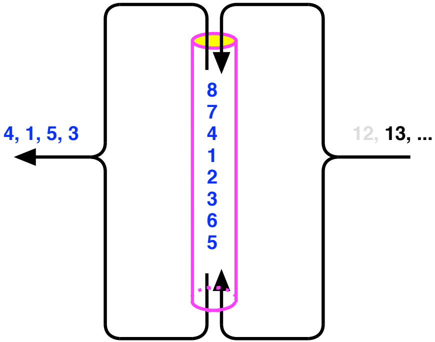


Figure: Bounded deque: encoded snapshot after I_1

Bounded deque

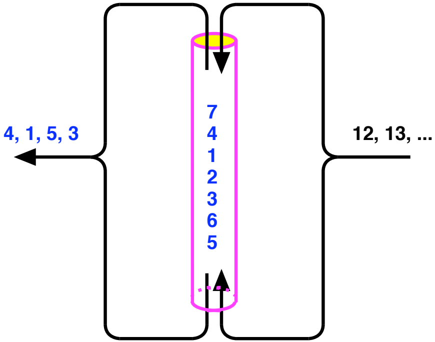


Figure: Bounded deque: encoded snapshot

Bounded deque

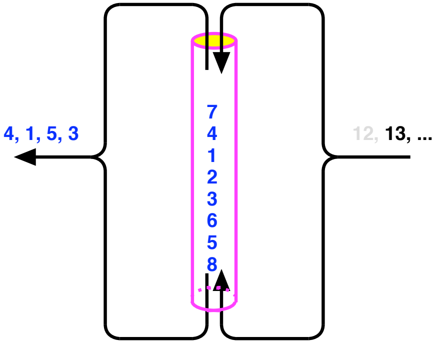


Figure: Bounded deque: encoded snapshot after I_2

Affairs of state

- Compute the non-deterministic FSA for a k-bounded system
- Determinise it and then minimise it
- Read off generating functions from the state transition rules
- Compute the growth rate of the k-bounded system which will be a lower bound for the growth rate of the unrestricted system

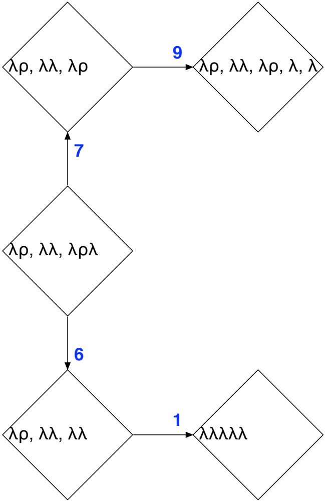
Controlling the state explosion

- The determinisation phase results in a state explosion which the subsequent minimisation phase somewhat controls
- For deques and parallel stacks some structure was observed in the final automaton which allowed us to construct it directly
- Deterministic states represented by sequences of words in letters λ, ρ. Eg for two parallel stacks

$$\lambda\rho, \lambda\lambda, \lambda\rho\lambda$$

represents configurations where the two smallest symbols are in different stacks, the next two smallest symbols are in the same stack, and of the last three symbols the middle one is not in the stack containing the first and third

Example state transitions



Further state reduction

- States fall into equivalence classes - just count numbers of λ's and ρ's in each word - e.g.

$$\lambda\lambda, \lambda\rho, \lambda\rho\lambda \sim \lambda\lambda, \lambda\rho, \lambda\lambda\rho$$

- Pass to the quotient automaton
- Prove it gives the same growth rate
- This allows the computations for deques and for two parallel stacks to be pushed further

Results

	k	Growth bound
Serial stacks	8	7.5535
Parallel stacks	18	7.535
Deques	21	7.890

Bottom line for growth rate γ

- Two stacks in series: $8 \leq \gamma \leq 13.374$
- Two stacks in parallel: $7.535 \leq \gamma \leq 8.3461$
- Deque: $7.890 \leq \gamma \leq 8.352$

Open questions

- What are the true growth rates?
- Do deques and two parallel stacks have the same growth rate?
- Why is two stacks in series more difficult?
- For deques and two parallel stacks we have efficient recognition algorithms; is the recognition problem for two stacks in series NP-complete?
- Can we get the *exact* enumerations for two parallel stacks? For deques?