

Generating finite transformation semigroups: SgpWin

Donald B. McAlister
(don@math.niu.edu)

Department of Mathematical Sciences
Northern Illinois University
and
C.A.U.L.

September 5, 2006

SgpWin (Semigroup for Windows)

- Written in C++ as standalone program
- Originally written for PC under DOS but rewritten for Windows
- Not part of a major project like GAP

How many semigroups are there?

Number of elements	Number of groups	Number of semigroups
3	1	4
4	2	18
5	1	126
6	2	1160
7	1	15973
8	3	836021
9	2	1833120128

How big are they?

Size of set	Max. size of group	Max. size of semigroup
3	6	27
4	24	64
5	120	2625
6	720	46656
7	5040	823543
8	40340	16777215

- Storing the elements internally quickly becomes impracticable

Solutions?

- Don't store the elements.
 - Instead be satisfied with some structural information
 - Lallement and McFadden "On the determination of Green's relations in finite transformation semigroups", *J. Symbolic Computation* (10) (1990), (5), 481-498.
- Do store the elements
 - Be satisfied with small semigroups
 - Froidure and Pin, "Algorithms for computing finite semigroups", *Foundations of Computational mathematics (Rio de Janeiro)*, (1997), Springer, Berlin 1997.
 - SgpWin

What do we mean by generate?

Let X be a finite set and A a (finite) set of distinct transformations on X . By the universal properties of the free semigroup A^+ on A , there is a unique homomorphism ϕ of A^+ onto $\langle A \rangle$ which makes the diagram

$$\begin{array}{ccc} A & \longrightarrow & A^+ \\ & \searrow & \downarrow \phi \\ & & \langle A \rangle \end{array}$$

commute. (The un-labeled maps are the obvious inclusions.)

To generate $\langle A \rangle$ means here

- 1 to specify a set of representatives of the classes of the congruence $\phi \circ \phi^{-1}$, with the members of A as the representatives of their classes.
- 2 to be able to say how these representatives multiply.

For this it suffices, given a representative x and $a \in A$, to be able to specify the representative of the class of xa .

General strategy

- 1 List, in some way, the elements of A^+ and, for each element x , calculate the corresponding transformation $T(x)$
- 2 If $T(x)$ has already appeared in the list of constructed transformations, discard x and move on to the next member of A^+ . If it has not appeared add x and $T(x)$ to the list and move on to the next member of A^+ .

Problems

- This isn't an algorithm. It has no stopping rule.
- Both calculating the transformations from scratch and searching to see if we have a new transformation require time and effort. searching, in particular - Google notwithstanding - requires a lot of work
- The process doesn't take advantage of work we have already carried out, and information we may have gained.

The short-lex ordering

This begins with a total ordering on A which is then extended to a compatible total ordering on A^+ , as follows

$$u < v \iff \begin{cases} \text{length}(u) < \text{length}(v) \\ \text{or } \text{length}(u) = \text{length}(v) \text{ and } u = xau', v = xbv' \end{cases}$$

where $x, u'v' \in A^*$, $a, b \in A$ and $a < b$.

- This ordering is compatible with multiplication on both sides in A^+ - in fact, it is a compatible well ordering.
- Each $\phi \circ \phi^{-1}$ -class has a least member. We call these the *canonical words* and use them for the class representatives in generating $\langle A \rangle$.

Lemma

The set C of canonical words is factorial in the sense that $w \in C$ implies that each factor of w belongs to C . If Y is a factorial subset of A^+ and $w \in Y$, $a \in A$, then $Y \cup \{wa\}$ is factorial if and only if $wa = bv$ where $v \in Y$, $b \in A$.

The algorithm - Step 1

The algorithm constructs the Cayley graph of $\langle A \rangle$ - actually the Cayley graph of $\langle A \rangle$ with an extra identity adjoined. This guarantees that we have a root node. The set C denotes the set of nodes constructed so far. It is a factorial set. The algorithm proceeds by traversing the sub-tree of A^+ corresponding to the canonical words using the short-lex ordering.

- 1 The generators A are nodes of the Cayley graph. Thus we begin with $C = A$. There is an arrow from the root to a for each generator a .



The algorithm - Step 2

Because the canonical words form a factorial set, a new canonical word of length $n + 1$ must have the form wa where w is a canonical word of length n .

2. Suppose that w is a canonical word (which we have constructed) and that we have dealt with all canonical words smaller than w and all generators smaller than a . This means that the arrows in the Cayley graph starting at nodes smaller than w have been constructed and that the same is true for all arrows starting at w labeled by letters smaller than a .

We need to construct the arrow

$$w \xrightarrow{a} ?$$

in the Cayley graph starting at w with label a .



Step 2 - continued

Test to see if $C \cup \{wa\}$ is a factorial set. By the lemma earlier, this occurs if and only if $wa = bv$ (in A^+) where $b \in A$ and $v \in C$.

- 1 If not then wa cannot be a canonical word, and v also is not a canonical word. The canonical word u such that $u\phi = v\phi$ is smaller than v so $bu < bv = wa$ while bu and wa have the same image under ϕ , and thus the same representative. By our induction hypothesis, we can find this representative z . Draw the arrow

$$w \xrightarrow{a} z$$

- 2 If $C \cup \{wa\}$ is factorial, then wa may or may not be a canonical word. We can't deduce this internally; that is using the part of the Cayley graph we have constructed so far. We need some external help. For this we use an oracle who can answer the question for us.
 - 1 If wa is new, draw an arrow from w to wa with label a and add wa to C .
 - 2 If $wa = v$, as transformations, draw an arrow from w to v with label a .

Step 3 - the next step

- 1 If a is not the last generator, replace a by the next generator and repeat Step 2.
- 2 if a is the last generator but w is not the last member of C on its level, replace w by the next member of C on that level, replace a by the first generator and repeat Step 2.
- 3 If w is the last member of C on this level
 - 1 if C contains an element on the next level, replace w by the first such member and a by the first generator and repeat Step 2.
 - 2 otherwise QUIT. The Cayley graph is complete.



Algorithm - comments

- In the first place, it does not explain, in Step 2, how to determine whether or not $C \cup \{wa\}$ is factorial. Nor does it explain, in case it is not, how to find the canonical word equivalent to wa .
- In Step 2(b), it does not explain how the oracle works.
- The algorithm gives, for free, a presentation for $S = \langle A \rangle$.

There are two ways in which “backward” arrows arise

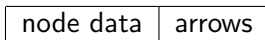
- 1 those which arise from “old” information when $C \cup \{wa\}$ is not factorial;
- 2 those which arise from “new” information provided by the oracle; these give a set $P = \{wa = v\}$ of relations.

Theorem

$S = \langle A | P \rangle$. P is the Knuth-Bendix presentation of S associated with the ordered alphabet A .

Algorithms + data structures = programs (N. Wirth)

- A semigroup element is considered as a data structure with two parts



- The node data consists of things that characterize the elements of the semigroup
- The arrows are pointers, one for each generator, to other elements. They represent the dynamics of the Cayley graph.
- The semigroup itself consists of an array of pointers to elements.
 - We could use a list instead of an array which would allow us, theoretically, to deal with semigroups of arbitrary size.
 - I chose an array when I first started to write the program and never changed this.

At its most basic level, the node data might be considered to consist of

word	action
------	--------

where the word is the canonical word equivalent to the element and the action represents the action of the transformation corresponding to the element.

It is more convenient to include a third item

word	action	structural information
------	--------	------------------------

where the structural information contains data which is useful for semigroup theoretic analysis.

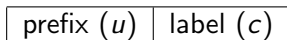
Completing the algorithm

One of the questions not yet fully answered in the algorithm is the following. Given a canonical word w and $a \in A$, with $wa = bv$ how do you know if v is a canonical word?

- 1 Start at the root and travel the path labeled by v in the the Cayley graph to reach the node whose word is v' .
- 2 If $v' = v$ then v is a canonical word and we consult the oracle about wa .
- 3 If not $wa = bv'$ in S and we need to get the canonical word for bv' . Since $bv' < wa$ we can find this by making a second voyage on the Cayley graph to reach the appropriate node.

Avoiding words

- Instead of storing the canonical word $w = uc$, store c and a pointer to u (the prefix of w).



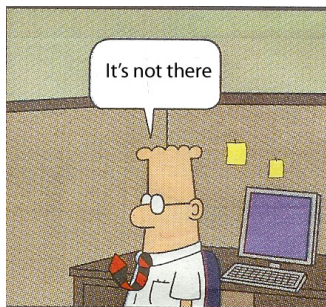
In the graph we have the arrow

$$u \xrightarrow{c} w$$

Then w can be recovered by back-tracking to the root through the predecessors.

- labels, prefixes, and suffixes come for free in the algorithm.
- If we store the suffixes as well as prefixes and labels we can cut down the number of trips on the Cayley graph by one. This results in a substantial increase in speed especially if we still store the canonical words.

The oracle



The oracle here is just a binary tree whose nodes are pointers to the semigroup elements. It uses a binary search to determine whether or not a given action is that of one of its members. If it is, it returns the address of the semigroup element. If it isn't - the semigroup has a new element - it adds the address of the new element to the tree.

Analyzing the structure

- The data structure part of each element consists of

\mathcal{L} -class Rep	\mathcal{R} -class Rep	\mathcal{D} -class Rep	image	rank	type
--------------------------	--------------------------	--------------------------	-------	------	------

All of these are un-signed integers except for “type” which is a character.

- Given that we use an array to contain the elements of the semigroup, it is natural that integers - the indices of the elements - are used to locate the representatives of the various Green’s classes,
- “Type” is one of eight characters

i idempotent
 h non-idempotent group element
 r regular non-group element
 n non-regular element

The same letter capitalized indicates that the element is also a generator.

Analysing the structure - Green's relations

The algorithm used is a hybrid one. It uses

- 1 the fact that it is easy to travel on the Cayley graph to calculate the \mathcal{R} -classes;
- 2 the fact that we are dealing with a transformation semigroup to calculate \mathcal{L} .

The algorithm is traditional in that it first calculates the regular Green's classes - those that contain idempotents - and then uses this information to find the non-regular ones.

Lemma

Let S be a finite transformation semigroup and let e be an idempotent. Then $a \in S$ belongs to the \mathcal{R} -class of e if and only if $\text{rank}(a) = \text{rank}(e)$ and $e \in aS$.

Regular \mathcal{R} -classes

In order to use this lemma, we need to be able to identify the idempotents. To do this we use bitwise operations to obtain a representation of the image of a transformation as an un-signed integer

Set	1	2	3	4	5	6	
Action	3	2	1	4	4	2	
Image	1	1	1	1	0	0	$= 1 + 2 + 4 + 8 = 15$

The rank is just the number of 1's in the representation. It is easy while finding this to determine whether or not the element is an idempotent - or a group element.

Bitwise operations on the image also make it easy to determine, for example, if an idempotent e is a right identity for an element a .

- The \mathcal{R} -class of e consists of all the elements of S , with the same rank as e , from which it is possible to travel to e .
- To find these we construct the adjoint graph Adj by turning the arrows around:
 - there is an arrow in Adj from a to b if and only if $rank(a) = rank(b)$ and there is an arrow in the Cayley graph from b to a .
 - the \mathcal{R} -class of e consists of all elements in Adj which can be reached from e . Finding these is a simple matter.
 - this subgraph of Adj is a familiar object. It is what Buck Stephen calls the Schützenberger graph of e .

Now that we know the regular elements, we are able to use the classical result

Lemma

Let S be a finite transformation semigroup and let a and b be regular elements of S . Then $a\mathcal{L}b$ if and only if $\text{image}(a) = \text{image}(b)$.

to find the regular \mathcal{L} -classes.

- Once we know the regular \mathcal{L} and \mathcal{R} -classes it becomes a standard matter, using Green's relation theory, to identify the regular \mathcal{D} -classes.
- To find the non-regular classes we make use of the fact that the non-trivial non-regular \mathcal{L} and \mathcal{R} -classes are translations of the regular ones. This is the process used by Lallement and McFadden. A detailed description can be found in their paper.

The structural information stored with the semigroup elements, allied with standard semigroup theory allows us to characterize many semigroups efficiently in terms of their structure. For example, a result of FitzGerald shows that a regular semigroup is orthodox if and only if the product of \mathcal{D} -related idempotents is idempotent.

Trace congruences

Let S be a finite inverse semigroup and ρ be a congruence on S . Then the restriction π of ρ to the set E of idempotents of S is a congruence on E with the additional property that

$$(e, f) \in \pi \Rightarrow (a^{-1}ea, a^{-1}fa) \in \pi \text{ for each } a \in S.$$

Such a congruence on E is called a *trace congruence* and $\pi = tr(\rho)$ is called the *trace of ρ* .

The relation on the lattice of congruences on S defined by

$$\rho \sim \tau \Leftrightarrow tr(\rho) = tr(\tau)$$

is easily seen to be a complete lattice congruence.

$$(a, b) \in \pi_{min} \Leftrightarrow (aa^{-1}, bb^{-1}) \in \pi \text{ and } ea = eb$$

for some $e = e^2$ with $(e, aa^{-1}) \in \pi$

$$(a, b) \in \pi_{max} \Leftrightarrow (a^{-1}aea, b^{-1}eb) \in \pi \text{ for all } e \in E.$$

Trace congruences

Suppose that π is a trace congruence. Then, because S is finite, each π class has a minimum member. We call these idempotents the π -minimal idempotents if e is an idempotent then $\varepsilon(e)$ denotes the minimum member of the π -class containing e . In terms of these we can give a simpler description of π_{\min} :

$$(a, b) \in \pi_{\min} \Leftrightarrow \varepsilon(aa^{-1}) = \varepsilon(bb^{-1}) = \varepsilon \text{ and } \varepsilon a = \varepsilon b.$$

Furthermore, the π -minimal idempotents determine the idempotent and Green's relation structure of S/π_{\min} .

For each $a \in S$ denote by $[a]$ the π_{\min} -class of a .

- Let e be a π -minimal idempotent. Then the map $a \rightarrow [a]$ is a groupoid isomorphism of the \mathcal{D} -class of e onto the \mathcal{D} -class of $[e]$ in S/π_{\min} .
- The partially ordered set of \mathcal{D} -classes that contain π -minimal idempotents is isomorphic to the set of \mathcal{D} -classes of S/π_{\min} .
- The semilattice of idempotents of S/π_{\min} is isomorphic to the partially ordered set of π -minimal idempotents.

It follows that if we can efficiently find the π -minimal idempotents we can efficiently read off the structural constants of the quotient semigroup S/π_{\min} .

Indeed, since every quotient of S/ρ , with $\text{tr}(\rho) = \pi$, is an idempotent separating homomorphic image of S/π_{\min} , we can read off the the Green's relation and idempotent structure of S/ρ . Even if, in general, we cannot read off the subgroups.

- If S is a transformation semigroup on a finite semigroup then there is a unique idempotent of lowest rank in each π -class. These elements are precisely the π -minimal idempotents.

Trace congruences - the algorithm

We use what is essentially the sieve of Eristothsenes - the way one finds the primes - twice.

Order the idempotents of S in order of increasing rank, so that the least idempotent z is the first in the list. Set $curr = z$.

- Go through the list, starting at $curr$ and delete any idempotent e which is π -equivalent to $curr$. At the end of the list, move $curr$ to the next remaining member and repeat. If there is no remaining member end. At this stage we have, up to isomorphism, the idempotents of the quotient semigroup.
- We re-sieve this time using \mathcal{D} instead of π . Now, to give the \mathcal{D} -class structure, all we have to do is to print out the \mathcal{D} -class data for each idempotent in the list.

Primitive trace congruences

- Let D be a \mathcal{D} -class of S and define π_D by

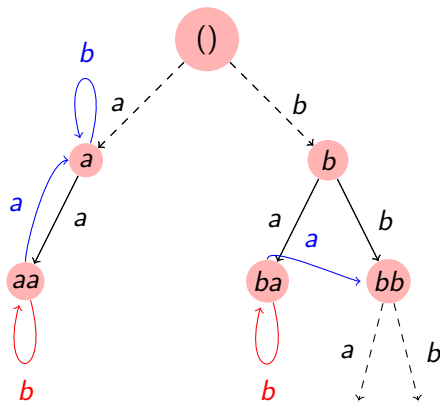
$$(u, v) \in \pi_D \Leftrightarrow \{e = e^2 \in D : e \leq u\} = \{e = e^2 \in D : e \leq v\}$$

- π_D is a trace congruence. We say it is a *primitive trace congruence*.
- Every trace congruence is an intersection of primitive trace congruences.
- The number of trace congruences can be very large in relation to the number of \mathcal{D} -classes. For example, if $S = E$ is a chain with n elements then S has 2^{n-1} (trace)-congruences.

Fim!

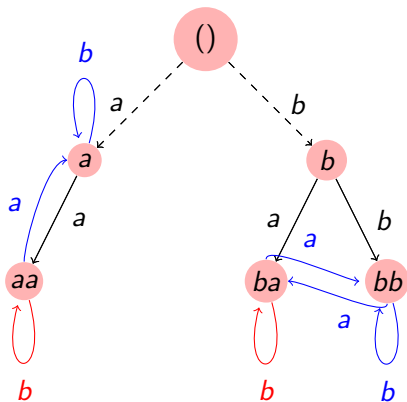
Example

$$a = (2 \ 1 \ 1 \ 2) \text{ and } b = (1 \ 2 \ 2 \ 3)$$

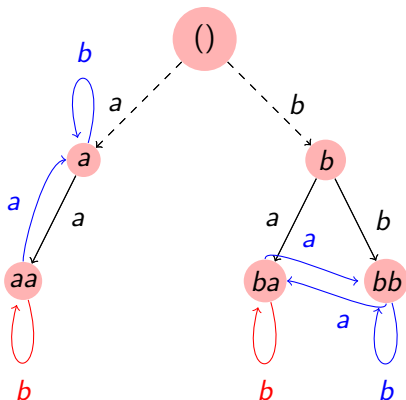


Example

$$a = (2\ 1\ 1\ 2) \text{ and } b = (1\ 2\ 2\ 3)$$



Example - Presentation



Presentation: $ab = a$, $aaa = a$, $baa = bb$, $bba = ba$, $bbb = bb$.

