CDM
More on Groups

Klaus Sutner
Carnegie Mellon University

22-groups 2019/10/20 16:43

1 Group Representations

- Congruences
- Generators and Cayley Graphs
- Products
- The Word Problem

Other Representations

Needless to say, a table of size n^2 is only useful for rather small values of n.

Sometimes we can express the group operation effectively by using standard operations on some numerical or combinatorial data structure.

One very important example of this technique is the use of matrices to represent groups: matrices over any ring come equipped with a natural operation: matrix multiplication.

As it turns out, by choosing suitable matrices to represent group elements one can often translated the group multiplication into matrix multiplication.

Note that this representation is potentially much smaller than the full Cayley table: we just have to store the translation from group element to matrix.

Permutation Matrices

Here is a first step in this direction: we express permutations as matrices.

Definition

A permutation matrix is a 0/1 matrix that has exactly one 1 in each row and each column.

It is easy to associate a permutation f on $[n]$ with an $n \times n$ permutation matrix M_f:

$$M_f(i,j) = \begin{cases} 1 & \text{if } f(i) = j, \\ 0 & \text{otherwise.} \end{cases}$$

Proposition

The map $\Phi : f \mapsto M_f$ is a group monomorphism.

This is easy to check. Note, however, that it is important how composition of permutations is defined here: $(f \circ g)(x) = g(f(x))$. The group operation on the matrices is ordinary matrix multiplication.

Alternative Composition

What if we use the other form of composition on permutations:

$$(f * g)(x) = f(g(x))?$$

Then $f \mapsto M_f$ is an anti-homomorphism.

Anti-homomorphisms are slightly less useful than homomorphisms, but this problem can be fixed by considering the transpose of M_f instead: $f \mapsto M_f^T$ is a group homomorphism from $\langle \mathfrak{S}_n, * \rangle$ to $\langle \mathrm{Mat}, \cdot \rangle$.

Why would anyone care about these two different representations?

They come in very handy in connection with group actions.

Exercise

Verify that Φ is a homomorphism and anti-homomorphism, respectively, given the two group structures on the set of permutations.

A Matrix Group

Here is another example. Consider the set G of all integer matrices of the form

$$\begin{pmatrix} e & k \\ 0 & 1 \end{pmatrix}$$

where $e \in \{-1, +1\}$ and $k \in \mathbb{Z}$. Since the determinant of these matrices is e they are in fact invertible – though it is far from clear that the inverse is in G.

Closure of G under matrix multiplication is easy to see, though:

$$\begin{pmatrix} e & k \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} e' & k' \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} ee' & k + ek' \\ 0 & 1 \end{pmatrix}$$

Exercise

Give a careful proof that G really forms a group. What else can you say about G?

Modeling D_n

There is a fairly simple representation of dihedral groups as matrix groups:

$$\begin{pmatrix} e & k \\ 0 & 1 \end{pmatrix}$$

where $e \in \{-1, +1\}$ and $k \in \mathbb{Z}_n$.

The basic rotation and reflection are represented by

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$$

This is nice, but not quite what we are looking for.

Representing Dihedral Groups

For simplicity, here is the special case D_4.

- 1: identity
- α, α^2, $\alpha^3 = \alpha^{-1}$: rotations
- β, $\alpha^2\beta$: reflections along axes
- $\alpha\beta$, $\alpha^3\beta$: reflection along diagonals

You have to check that this is geometrically correct.

This is clear for the rotations α^i, but the reflections require a little check. In particular it is not clear that there is a single reflection β such that all others can be expressed in terms of β and rotations.

Exercise

Check the details.

Simplification Rules

Now for the important part: again we don't need a full multiplication table, we can just "multiply" these words over the alphabet $\{\alpha, \beta\}$.

$$\alpha^2 \cdot \beta\alpha = \alpha^5\beta = \alpha\beta$$

Here we have used the following "simplification" rules:

- $\alpha^4 = 1$,
- $\beta^2 = 1$,
- $\beta\alpha = \alpha^3\beta = \alpha^{-1}\beta$.

As is customary in this context, we have written 1 to express the empty word.

Note that this group is not commutative.

The General Case

What is the structure of a dihedral group D_n in general?

Exactly the same, we only need to replace 4 by n. No extra work is needed.

- $\alpha^n = 1$,
- $\beta^2 = 1$,
- $\beta\alpha = \alpha^{n-1}\beta = \alpha^{-1}\beta$.

Very elegant.

Exercise

Check that this description holds in general for all n. Make sure to distinguish between even and odd n.

Normal Form

It follows that every element can be written in the form

$$\alpha^i \quad \text{or} \quad \alpha^i\beta$$

where $0 \leq i < n$.

- The α^i are all the pure rotations.
- The $\beta\alpha^i$ are all the combined rotations cum reflection along a particular axis, which produces arbitrary reflections.

To multiply two elements, we concatenate the words and then apply the simplification rules until we are back in normal form.

Simplification in D_4

The simplification process can always be organized into two phases: move all the β's to the back, than cancel α^4 and β^2:

$$\begin{array}{l} x \\ \alpha^k\beta^\ell \\ \alpha^i\beta^j \end{array}$$

where $i = k \bmod 4$ and $j = \ell \bmod 2$.

But note that it is not clear that application of the rules in a different order might not produce a different result. One can prove that there is no problem, one always gets to the same normal form (confluent rewrite system).

Finitely Presented Groups

Hence we can describe the dihedral group D_n very compactly by

$$\langle \alpha, \beta \mid \alpha^n = 1, \beta^2 = 1, \alpha\beta = \beta\alpha^{-1} \rangle$$

This means: there are two special elements α and β, and all the other elements can be obtained from these two by arbitrary combinations subject to the rules indicated.

Again, there is no need for a multiplication table of size $(2n)^2$.

We can just form products of α's and β's and apply the simplification rules given.

A Definition

More generally we can obtain groups over some set X of generators as follows:

- Select a set $R \subseteq F(X)$, the set of relators.
- Define the normal closure $\overline{R}$ of R as the subgroup of $F(X)$ generated by $\{\, grg^{-1} \mid r \in R, g \in F(G) \,\}$.
- Then the group $G = F(X)/\overline{R}$ is represented by X and R.

One usually simply writes $\langle X \mid R \rangle$ for the group so defined.

Of course, the most interesting case is when X and R are both finite. The group is then said to be finitely presented.

Finitely Presented

It is very hard to find examples of groups that are finitely generated but not finitely presented.

For example, one can show that there are $2^{\aleph_0}$ non-isomorphic groups with 2 generators. Since there can be only countably many finitely presented such groups most of them are not.

Cardinality arguments are usually frustrating. Concretely, in this case one can show that a wreath product (see below) of two infinite cyclic groups is not finitely presented, but the argument is difficult.

Relators versus Relations

The idea behind this construction is that $\langle X \mid R \rangle$ is the group generated by X that has only the properties described in R (and whatever follows from there via the group axioms), but nothing else.

By property we mean that $r \in R$ is identified with the unit element.

For example, for the dihedral group D_n above we could choose

$$R = \{\alpha^n, \beta^2, (\alpha\beta)^2\}$$

The relator notation can be a little terse, it may be better to write $x = y$ rather than xy^{-1}.

Von Dyck's Lemma

It is intuitively clear that for relators $R \subseteq S \subseteq F(X)$, the group $\langle X \mid S \rangle$ is more specialized than $\langle X \mid R \rangle$. This is made explicit in the next lemma.

Lemma

Let $G = \langle X \mid R \rangle$ and $H = \langle X \mid S \rangle$. For $R \subseteq S \subseteq F(X)$ there exists an epimorphism $f : G \to H$ that is the identity on X and has kernel $\overline{S - R}$.
On the other hand, every factor of G has a presentation $\langle X \mid S \rangle$ where $R \subseteq S$.

An Example

Consider the Fibonacci group on three generators:

$$F = \langle \alpha, \beta, \gamma \mid \alpha\beta = \gamma, \beta\gamma = \alpha, \gamma\alpha = \beta \rangle$$

Here is the famous quaternion group:

$$Q = \langle a, b, c, d \mid a^2 = 1, b^2 = c^2 = d^2 = bcd = a \rangle$$

BTW, this group is not commutative, yet all its proper subgroups are normal.

Exercise

Show that the two groups are isomorphic.

Another Example

Here is just one other example of a finitely presented group. There are $n-1$ generators $g_1, g_2, \ldots, g_{n-1}$ and the relations are

$$\begin{aligned} g_1^2 &= 1 \\ (g_i g_{i+1})^3 &= 1 \qquad \text{for } i < n-1 \\ (g_i g_j)^2 &= 1 \qquad \text{for } j < i-1 \end{aligned}$$

Note how it is entirely unclear what the size of this group is, or what its properties might be. It requires a bit of effort to establish the following characterization.

Lemma

The group defined by these rules is (isomorphic to) the symmetric group on n points. Hint: figure out what the g_i are.

A Question

Is it possible to write down a contradictory set of relators?

For example, suppose we write $a = b^2$, $a = b^5$, $a^2 = b^{-3}$.

Does this still describe a group?

- Group Representations

2 Congruences

- Generators and Cayley Graphs
- Products
- The Word Problem

Subgroups and Partitions

To explain more carefully how the last construction works we need to consider good equivalence relations on groups. This will also be crucial for applications to counting.

Definition

Suppose G is a finite group. Fix a subgroup H of G, let $a \in G$. Define

$$x \sim_H y \iff x^{-1} y \in H$$

$$a \cdot H = \{\, a \cdot b \mid b \in H \,\}$$

$\sim_H$ is the equivalence induced by H. The sets $a \cdot H$ are the (left) cosets of H in G, and the number of such cosets is the index of H in G, written $[G : H]$.

One can define right cosets in an analogous way.

Example 1

Let G be the integers under addition and $H = m\mathbb{Z}$. Then

$$\begin{aligned} x \sim y &\iff y - x \in m\mathbb{Z} \\ &\iff x = y \pmod{m} \end{aligned}$$

H is the kernel of the epimorphism $x \mapsto x \bmod m$.

Example 2

Let G be the group of all permutations on $[n]$. Define

$$f(x) = \begin{cases} 0 & \text{if } x \text{ is even,} \\ 1 & \text{otherwise.} \end{cases}$$

Then f is homomorphism from G to the additive group $\mathbb{Z}_2$.

The kernel of f is the subgroup

$$H = \{\, x \in G \mid x \text{ even} \,\}$$

Note that $|H| = |G|/2 = n!/2$.

Example 3

Consider the multiplicative group

$$G = \mathbb{Z}_{13}^{*} = \{1, 2, \ldots, 12\}$$

We one can check that $H = \{1, 3, 9\}$ is a subgroup with cosets

$$H = \{1, 3, 9\}, 2H = \{2, 5, 6\}, 4H = \{4, 10, 12\}, 7H = \{7, 8, 11\}$$

The multiplication table for G/H written with canonical representatives is

1	2	4	7
2	4	7	1
4	7	1	2
7	1	2	4

and is isomorphic to the additive group $\mathbb{Z}_4$.

Compare to Plain Sets

This should look rather familiar by now: we have used plain functions

$$f : A \to A$$

to describe equivalence relations on sets (canonical selector function).

Now we use subgroups to describe special equivalence relations on (the carrier sets of the) groups. But these subgroups are all kernels of appropriate homomorphisms.

Of course, not all equivalence relations can be obtained via homomorphisms, but those that are so obtained have particularly good properties.

Exercise

Show that for H the kernel of a homomorphism our definition of $\sim_H$ is the same as in the set case.

Partition Size

Lemma

$\sim_H$ is an equivalence relation on G, and the equivalence classes of $\sim_H$ all have the same size $|H|$.

Proof.

Reflexivity follows from $1 \in H$.

Symmetry since $x^{-1}y \in H$ implies $(x^{-1}y)^{-1} = y^{-1}x \in H$,

Transitivity since $x^{-1}y, y^{-1}z \in H$ implies $x^{-1}z \in H$.

For the second claim note that $[x]_\sim = xH$.

But $z \mapsto xz$ is a bijection from H to xH. □

Lagrange's theorem

Theorem (Lagrange)

Let G be a finite group, and H any subgroup of G. Then $|G| = |H| \cdot [G : H]$.

In particular, $|H|$ divides $|G|$.

Note how algebra produces a stronger result here: if we look at arbitrary functions $f : A \to B$ then any equivalence relation arises as a kernel relation.

But if we consider groups and homomorphisms we get only very special equivalence relations.

This restriction will turn out to be very helpful to answer certain counting problems.

Application

As a special case the order of any group element divides the order (cardinality) of the whole group.

Hence for $n = |G|$, $a \in G$ we have $a^n = 1$.

This provides a simple proof for the famous Euler-Fermat theorem.

Recall that $\mathbb{Z}_m^{\star}$ is the group of elements in $\mathbb{Z}_m$ that have multiplicative inverses.

Also, $\varphi(m)$ is Euler's totient function: $\varphi(m) = |\mathbb{Z}_m^{\star}|$.

Theorem (Euler-Fermat)

The order of $a \in \mathbb{Z}_m^{\star}$ divides $\varphi(m)$.

Congruences

Definition

Suppose G is a group and $\sim$ an equivalence relation on G. Then $\sim$ is a congruence if for all $x, y, u, v \in G$:

$$x \sim y, u \sim v \quad \text{implies} \quad xu \sim yv.$$

Congruences are very important since they make it possible to define a group structure on the quotient set $G/\sim$:

$$[x] \cdot [y] = [x \cdot y]$$

Why? Suppose $[a] = [a']$ and $[b] = [b']$. Then our definition makes sense only if $[ab] = [a'b']$.

But that's exactly what the congruence property guarantees.

Congruences and Homomorphisms

Unfortunately, the equivalence relations $\sim_H$ are not congruences in general.
We need another condition to get a congruence.

Definition

Let H be a subgroup of G. H is a normal subgroup if for all $a \in G$: $aH = Ha$.

Note that in a commutative group any subgroup is normal.

Proposition

If H is a normal subgroup then $\sim_H$ is a congruence.

Proposition

H is the kernel of a homomorphism $f : G \to G'$ iff H is normal.

Exercise

Prove these propositions.

Example: Chinese Remainder

You know this already. E.g., let p and q be two distinct primes.

$$f : \mathbb{Z} \to \mathbb{Z}_p \times \mathbb{Z}_q$$
$$f(x) = (x \bmod p, x \bmod q)$$

Then $H = pq\,\mathbb{Z}$ and the quotient is $\mathbb{Z}/(pq\mathbb{Z}) = \mathbb{Z}_{pq}$.

One can show that f is an epimorphism (this requires a little argument).
Hence $\mathbb{Z}_{pq}$ is isomorphic to $\mathbb{Z}_p \times \mathbb{Z}_q$.

So?

Hence we can either compute

- with one number modulo pq, or
- with two numbers, one modulo p and the other modulo q.

$\mathbb{Z}_{pq}$ and $\mathbb{Z}_p \times \mathbb{Z}_q$ are isomorphic, but computationally there is a difference.
This can be exploited sometimes to fake high-precision computations with small word sizes.

Also note that the correctness proof for RSA more or less requires the product representation.

Word Processing in Groups

Recall our clever description for dihedral groups:

$$\langle \alpha, \beta \mid \alpha^n = 1, \beta^2 = 1, \beta\alpha = \alpha^{-1}\beta \rangle$$

We can also think of α and β as pure letters, say, a and b for clarity. Then there is a homomorphism from the monoid of all words over $\{a, b\}$ to the group

$$\eta : \{a, b\}^\star \to G$$

which is given by $f(a) = \alpha$ and $f(b) = \beta$. The word equations

$$a^n = 1, b^2 = 1, ba = a^{n-1}b$$

describe the kernel K of f.

Quotients, Again

In other words, for any $w \in \{a, b\}^\star$ we have $f(w) = 1$ iff w can be reduced to 1 (if you prefer: the empty word) by repeated application of the rewrite rules.

The quotient of the monoid $\{a, b\}^\star$ by K is isomorphic to the group G.

Algebraically, there is no difference between G and the quotient structure: exactly the same equations hold in both.

The great advantage of this representation is that we can easily manipulate words, but may not have a direct handle on the group elements (even for dihedral groups where we can handle the group elements the word representation is computationally easier).

Automatic Groups

In fact, we can use finite state machines to perform operations on the words representing group elements.

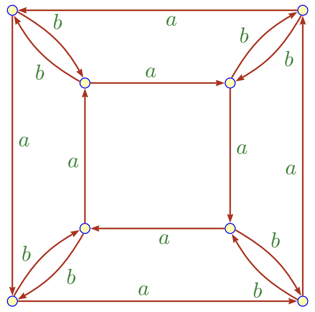
For example, we might try to find a FSM M_1 that, given a word w over $\{a, b\}$ determines whether $f(w) = 1$.

Likewise, we can have machines for multiplication. This is technically a little tricky since we need two words as input (and cannot simply read first u and then v; rather, we must read both words simultaneously using a product alphabet).

Groups that can be described by finite state machines have a particularly good structure and are well understood.

Example: D_4

The minimal DFA that checks identity in D_4.



Interpretation:
inner states: α^i
outer states: $\alpha^i \beta$

Any state can be chosen as initial and final state.

Digression: Counting Identities

How many words over $\{a, b\}^\star$ of length n are there that denote the identity in D_4?

This may seem like a hard problem, but given the DFA one can easily experiment. Here is a census for words up to length 12:

n	0	1	2	3	4	5	6	7	8	9	10	11	12
#	1	0	1	0	4	0	16	0	64	0	256	0	1024

The conjecture is now obvious: for length $2n > 0$ there are 2^{2n-2} words that denote the identity.

Exercise
Prove this conjecture.

Constructing Groups

We started from a concrete group, and found a nice description for it. But we can also go backwards and start with a description (assuming that we really describe a group).

For example, what is the group described by

$$G = \langle a \mid a^6 \rangle$$

Here the reduced elements are of the form $\{1, a, aa, aaa, aaaa, aaaaa\}$.

So we get the none other than the cyclic group $\mathbb{Z}_6$.

Exercise
Give a representation for $\mathbb{Z}_n \times \mathbb{Z}_m$.

Exercise
Give a representation for the full permutation group on n points.

- Group Representations
- Congruences
- 3 Generators and Cayley Graphs
- Products
- The Word Problem

Back To Subgroups: The Closure Perspective

Our characterization of subgroup shows that subgroups are related to closures: we want subsets $B \subseteq G$ that are "closed under the group operations":

- $1 \in B$
- $x \in B$ implies $x^{-1} \in B$
- $x, y \in B$ implies $xy \in B$

In the finite case the second condition is superfluous.

This point of view makes it natural to start from any set $A \subseteq G$ and then add elements until we have closure, i.e., a subgroup: we want the least subgroup that contains A.

Generating Subgroups

As it turns out, there are many useful subgroups defined by conditions such as the following:

> Let H be the least subgroup of G containing all the elements $g_1, \ldots, g_k \in G$.

Definition
Let G be a group and $A \subseteq G$. The group generated by A is the least subgroup of G that contains A. A is a set of generators for this subgroup.
In symbols: $\langle A \rangle$.

Here are some examples.

Some Subgroups

Example

- Swap: The permutation "swap 1 and 2" generates a subgroup of size 2: $\{I, (1,2)\}$.
- The permutation "rotate left" generates a subgroup of size n: $\{\, (1,2,\ldots,n)^i \mid i < n \,\}$.
- The same subgroup is generated by "rotate left".
- The alternating group $\mathbb{A}_n$.
- But rotate together with swap generate the whole symmetric group.
- Let G be the full permutation group on X. Let $X_0 \subseteq X$ and set

  $$H = \{\, f \in G \mid \forall x \in X_0 \, f(x) = x \,\}.$$

  Then H is a subgroup of G.

Existence versus Construction

Abstractly we know that $\langle A \rangle$ always exists since

$$\langle A \rangle = \bigcap \{\, H \subseteq G \mid A \subseteq H \text{ subgroup} \,\}$$

But computationally this is useless: we already have to know all subgroups (containing A) to compute the intersection.

Even if G is finite and relatively small, there is no hope to build an algorithm on this characterization.

Also note that $\langle A \rangle$ is one of the H's on the right hand side (impredicative definition).

So how do we actually compute $\langle A \rangle$ from A?

One Generator

If there is only one generator g, and G is finite, then all we get is

$$H = \{\, g^i \mid 0 \leq i < m \,\}$$

where m is the order of g in G.

Note that in this case H is Abelian.

In fact, H is isomorphic to $\mathbb{Z}_m$ via $i \mapsto g^i$.

If the order of g is infinite, then $\langle g \rangle$ is isomorphic to $\mathbb{Z}$.

Two Generators

A small set of generators provides a good, compact description of a group provided one can easily compute the group operation. Alas, for multiple generators, things soon become very complicated.

Let $a = (1,2)$ and $b = (1,2,\ldots,n)$ in $\mathfrak{S}_n$.

Proposition

$\langle a, b \rangle = \mathfrak{S}_n$.

Exercise

Show that a single generator cannot suffice to produce $\mathfrak{S}_n$.

Exercise

Find some other set of generators for $\mathfrak{S}_n$.

Exercise

Characterize all two-element sets of generators for $\mathfrak{S}_n$.

Cayley Graphs

Generators also build a bridge between groups and labeled digraphs.

Definition

Let G be a group and A a set of generators of G. The (right) Cayley graph of G (wrt. A) is the directed labeled graph $\Gamma_r(G) = \langle G, E \rangle$ with edges given by

$$x \xrightarrow{a} x \cdot a \qquad \text{where } x \in G, a \in A$$

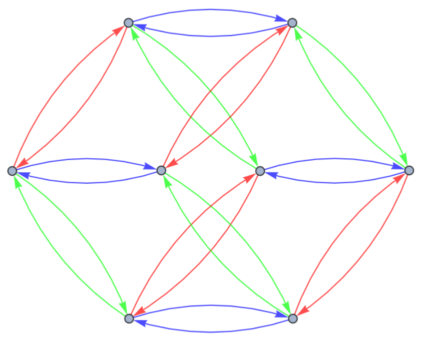
Likewise we can define a left Cayley graph $\Gamma_l(G)$.

Lemma

$\Gamma_r(G)$ *is strongly connected and regular.*

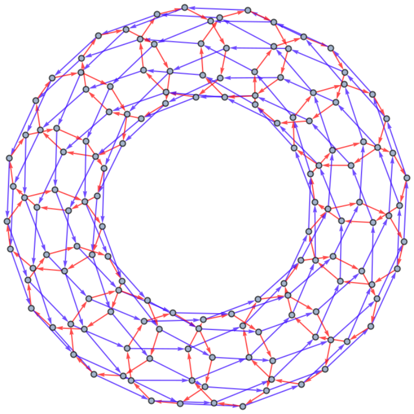
This follows from A being a set of generators.

Example



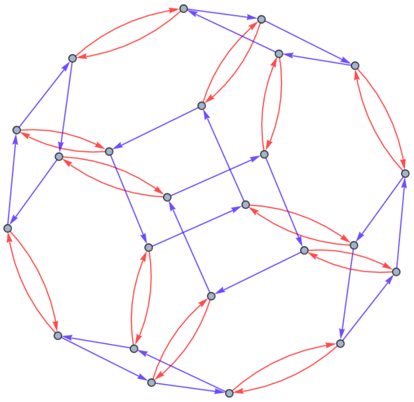
The Cayley graph of the Abelian group $\mathbb{Z}_2 \times \mathbb{Z}_2 \times \mathbb{Z}_2$.

Example



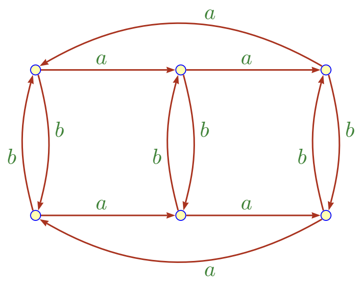
The Cayley graph of the Abelian group $\mathbb{Z}_9 \times \mathbb{Z}_{16}$.

Example



The Cayley graph of the symmetric group of four letters.

Example



Exercise

Determine the group given by this Cayley graph.

Generating Subgroups

So suppose we have a set of generators $g_1, \ldots, g_k$ in some finite group G. How do we compute the generated subgroup $\langle g_1, \ldots, g_k \rangle$?

- We form all possible compositions of the generators: $g_1 \cdot g_1$, $g_1 \cdot g_2$, $g_2 \cdot g_1$, $g_1 \cdot g_2 \cdot g_1$, $g_2 \cdot g_1 \cdot g_1$, and so on.
- Stop when no new elements appear.

This is a classical closure operation.

Note that if we were operating in an infinite group we also would need to deal with inverses (just try a single generator).

Fixed Point Perspective

Since we are dealing with a closure operation it is natural to express the construction using fixed points.

Define F to be the operation "multiply by all the generators and keep the old stuff":

$$F : \mathfrak{P}(G) \to \mathfrak{P}(G)$$
$$F(X) = Xg_1 \cup Xg_2 \cup \ldots \cup Xg_k \cup X$$

Then at least for finite groups G we have $\langle g_1, \ldots, g_k \rangle = \mathsf{FP}(F, \{1\})$.

In any implementation bookkeeping will be crucial to avoid recomputation.

Fixed Point Algorithm

Here is an implementation of the fixed point idea. The program below avoids recomputation by keeping a set of "active" elements (which may lead to new group elements when multiplied by a generator).

```
act = H = {1};
while( act != {} )
{
        H   = Union[ H, act ];
        act = act * gen;          // form all possible products
        act = Complement[ act, H ];
}
```

Exercise

Figure out how to implement this algorithm. For example, how should one handle subgroups of $\mathfrak{S}_n$ generated in this fashion?

Example

Let $a = T(2,1,3,4)$ and $b = T(2,3,4,1)$ in $\mathfrak{S}_4$. The table shows the elements added at each step. We omit the T for clarity.

level	new elements
0	$(1,2,3,4)$
1	$(2,1,3,4),(2,3,4,1)$
2	$(1,3,4,2),(3,2,4,1),(3,4,1,2)$
3	$(2,4,1,3),(3,1,4,2),(3,4,2,1),(4,1,2,3),(4,3,1,2)$
4	$(1,4,2,3),(3,1,2,4),(4,1,3,2),(4,2,1,3),(4,3,2,1)$
5	$(1,2,4,3),(1,3,2,4),(1,4,3,2),(3,2,1,4),(4,2,3,1)$
6	$(2,1,4,3),(2,3,1,4),(2,4,3,1)$

Spanning Tree Perspective

The fixed point algorithm is perfectly usable, but sometimes a more detailed construction is preferable.

For an alternative approach we can use Cayley graphs. Suppose we had the Cayley graph $\Gamma_r(H)$ of H whose edges are labeled by the given generators A.

We can think of generating a subgroup as a graph exploration problem: starting at vertex 1 we are trying to produce all points reachable via path segments labeled by a sequence $g_1, \ldots, g_k$ of generators.

Needless to say, this is just a Gedankenexperiment: all we have in reality is the generators and some group operation in G.

Building a Spanning Tree

At any rate, we can build a spanning tree T of $\Gamma_r(H)$ as follows:

- Initialize $T = \{1\}$.
- For each leaf x of T compute $y = x \cdot g$ for all generators g.
- If y is not in T add y as a new node, and add a tree edge (x, g, y).
- Stop when no new nodes appear.

This is really just breadth-first-search in a virtual graph: it suffices to have one point and the ability to generate all out-edges of a point.

It is even an advantage not to have a standard data structure (such as an adjacency list): this approach also works when G is huge (even infinite) as long as H is not too large.

Witnesses

Note that every point in the spanning tree (and thus every element of H) is associated with a unique path from the root to the point.

Reading off the labels of that path we obtain a "word" whose letters are generators that represents the subgroup element: multiplying out this product of generators we obtain the subgroup element.

This word is called a witness for the corresponding element.

Witnesses are occasionally useful, in particular in connection with finite state machine constructions. More later.

Example 1

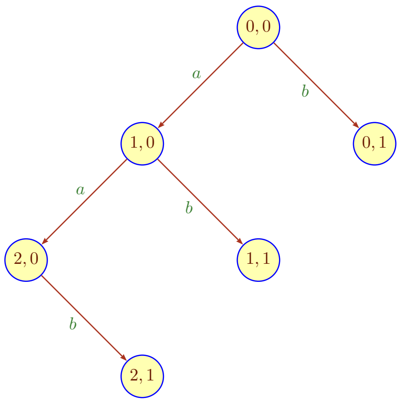
Consider the two generators $a = (1,0)$ and $b = (0,1)$ in $\mathbb{Z}_3 \times \mathbb{Z}_2$, the group from the Cayley graph example.

witness	element
ε	$(0,0)$
a	$(1,0)$
b	$(0,1)$
aa	$(2,0)$
ab	$(1,1)$
aab	$(2,1)$

There is no witness ba since $ba = (1,1)$ which already has a length-lex shorter witness $ab < ba$.

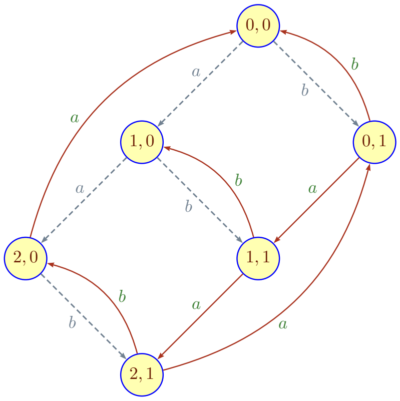
The Tree

Here is the spanning tree constructed by the algorithm.



The Full Graph

Adding the back-edges encountered during the construction we get the full Cayley graph.

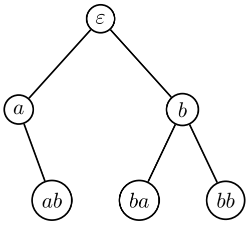


Example 2

Consider the two generators $a = T(2,1,3,4)$ and $b = T(2,3,1,4)$ in $\mathfrak{S}_4$.

ε	$T(1,2,3,4)$
a	$T(2,1,3,4)$
b	$T(2,3,1,4)$
ab	$T(3,2,1,4)$
ba	$T(1,3,2,4)$
bb	$T(3,1,2,4)$

Tree:



Example 3

Consider the two generators $a = T(2,1,3,4)$ and $b = T(2,3,4,1)$. We use one-line notation for clarity.

ε	$(1,2,3,4)$	$abba$	$(4,3,2,1)$
a	$(2,1,3,4)$	$abbb$	$(1,4,2,3)$
b	$(2,3,4,1)$	$babb$	$(3,1,2,4)$
ab	$(3,2,4,1)$	$bbab$	$(4,1,3,2)$
ba	$(1,3,4,2)$	$ababb$	$(1,3,2,4)$
bb	$(3,4,1,2)$	$abbab$	$(1,4,3,2)$
aba	$(3,1,4,2)$	$babba$	$(3,2,1,4)$
abb	$(4,3,1,2)$	$babbb$	$(4,2,3,1)$
bab	$(2,4,1,3)$	$bbabb$	$(1,2,4,3)$
bba	$(3,4,2,1)$	$ababba$	$(2,3,1,4)$
bbb	$(4,1,2,3)$	$ababbb$	$(2,4,3,1)$
$abab$	$(4,2,1,3)$	$abbabb$	$(2,1,4,3)$

This time the tree has depth 6.

Implementation

As usual, assume that each group element can be stored in $O(1)$ space, that a group operation can be computed in $O(1)$ time, and that we can store and search for elements in $O(1)$ steps.

Theorem

One can compute the subgroup H (and indeed the Cayley graph of H) generated by $g_1, \ldots, g_k$ in expected time linear in the size of H.

Note, though, that the efficiency assumption may be blatantly false in practice. E.g. for permutation and matrix groups the algebraic operations are quite expensive.

The theorem is based on using a hash table as the main container. Computing a good hash function and checking equality may be far from constant.

Pseudocode

Here is pseudocode for the subgroup generating algorithm given a list of generators.

```
enqueue( 1 );
while( queue not empty )
        x = dequeue;
        for each generator g do
                y = x * g;
                if( y is new )
                        enqueue( y );
```

Use a hash table (or some other dictionary) to check if y is new.

Why a Queue?

As far as the set of generated elements is concerned, any container would do. But in some applications a queue is the best choice.

To see why, first let us rephrase the problem slightly. We are really dealing with a structure

$$\mathcal{A} = \langle A, f_1, \ldots, f_k \rangle$$

and, given a point $a \in A$, we are trying to compute the least $B \subseteq A$ such that

- $a \in B$, and
- $x \in B$ implies $f_i(x) \in B$ for all $i = 1, \ldots, k$.

As a set, B is independent of our choice of container in the algorithm.

But Witnesses Are Not

Consider the alphabet $\Sigma_k = \{1, 2, \dots, k\}$. Much as in a DFA we can evaluate a word $w = e_1 e_2 \dots e_n$ by applying the functions to the seed point a in this order:

$$b = f_{e_1} \circ f_{e_2} \circ \dots \circ f_{e_n}(a)$$

So w is a witness (or certificate) for $b \in B$.

But in general there are many witnesses, so one would like to have a natural one. For algorithmic purposes, the best choice is the length-lex minimal witness.

To get the length-lex minimal witness we have to use a queue (and run through the generating functions in the natural order).

Exercise

Prove this assertion about minimal witnesses.

Prototype Algorithm

Here is the spanning tree algorithm as pseudocode.

```
enqueue( a );
while( queue not empty )
      x = dequeue;
      for each function fi do
          y = fi(x);
          if( y is new )
              wit(y) = wit(x).i;
              enqueue( y ); // perhaps other processing
```

By choosing appropriate "other processing" versions of this prototype algorithm can be used in many situations.

- Group Representations
- Congruences
- Generators and Cayley Graphs

4 Products

- The Word Problem

Direct Products

Given two groups A and B we can construct a new group $G = A \times B$ as follows: the carrier set is the Cartesian product $A \times B$ and multiplication is pointwise:

$$(a_1, b_1)(a_2, b_2) = (a_1 a_2, b_1 b_2)$$

The construction naturally generalizes to the direct product of an arbitrary number of factors:

$$\prod_{i \in I} A_i$$

A variant of this is the coproduct:

$$\coprod_{i \in I} A_i = \{\, a \in \prod A_i \mid \mathrm{spt}(a) \text{ finite} \,\}$$

where $\mathrm{spt}(a)$ denotes the support of a: the indices i for which $a_i \neq 1$.

Unique Products

Suppose we have $G = AB$ where A and B are subgroups of G, A normal, so that every element of G can be written as a product ab where $a \in A$, $b \in B$.

This is particularly interesting when this decomposition is unique: $a_1 b_1 = a_2 b_2$ implies $a_1 = a_2$, $b_1 = b_2$.

It is not hard to see that this type of uniqueness requires that $A \cap B = 1$.

But then we can define a map

$$f : G \to B \qquad f(x) = b \text{ where } x = ab.$$

This looks weird in an algebraic context, but f is well-defined by uniqueness.

An Isomorphism

Even better, f is a group homomorphism since A is normal:

$$\begin{aligned}(a_1, b_1)(a_2, b_2) &= a_1 b_1 a_2 b_2 \\ &= a_1 a_2' b_1 b_2 \\ &= (a_1 a_2', b_1 b_2)\end{aligned}$$

where the a_2' is chosen according to $b_1 A = A b_1$: $a_2' b_1 = b_1 a_2$.

Then f is the identity on B and has kernel A; we get an isomorphism

$$B \longrightarrow G \longrightarrow G/A$$

Example: D_4

For the dihedral group D_4, we have a unique decomposition into

$$A = \{1, \alpha, \alpha^2, \alpha^3\}$$
$$B = \{1, \beta\}$$

where α is a rotation by $\pi/2$ and β any reflection.

However, it is emphatically not true that $D_4 \simeq A \times B$:

$$(\alpha, \beta)(\alpha, \beta) = (\alpha^2, 1)$$

which is wrong since (α, β) is a reflection, so $(\alpha, \beta)^2 = 1$.

Question: Is there some other group structure on $A \times B$ that produces an isomorphism?

Semidirect Products

The verificatin that the projection f from above is a homomorphism suggests that we need to apply an automorphism to a_2, and the choice of automorphism depends on b_1 (above the automorphism is $a \mapsto b_1 a b_1^{-1}$).

Definition

Let $\varphi : B \to \mathrm{Aut}(A)$ be a homomorphism. The semidirect product $A \rtimes_\varphi B$ is defined on the Cartesian product of the carrier sets of A and B by

$$(a_1, b_1)(a_2, b_2) = (a_1\, \varphi(b_1)(a_2), b_1 b_2)$$

Note that if φ is the constant map $b \mapsto I$, we get back the ordinary direct product.

Quoi?

It is not even clear that this operation is associative.

$$\begin{aligned}
((a_1, b_1)(a_2, b_2))(a_3, b_3) &= (a_1\, \varphi(b_1)(a_2), b_1 b_2)(a_3, b_3) \\
&= (a_1\, \varphi(b_1)(a_2)\, \varphi(b_1 b_2)(a_3), b_1 b_2 b_3)
\end{aligned}$$

$$\begin{aligned}
(a_1, b_1)((a_2, b_2)(a_3, b_3)) &= (a_1, b_1)(a_2\, \varphi(b_2)(a_3), b_2 b_3) \\
&= (a_1\, \varphi(b_1)(a_2\, \varphi(b_2)(a_3)), b_1 b_2 b_3) \\
&= (a_1\, \varphi(b_1)(a_2)\, \varphi(b_1 b_2)(a_3), b_1 b_2 b_3)
\end{aligned}$$

The neutral element is $(1_A, 1_B)$ and the inverse is

$$(a, b)^{-1} = (\varphi(b^{-1})(a^{-1}), b^{-1})$$

Semidirect Examples

Example

Let $G = AB$ as above and define

$$\varphi(b)(x) = bxb^{-1}$$

Then G is isomorphic to $A \rtimes_\varphi B$.

Example

Let $A = \mathbb{Z}_n$ and $B = \mathbb{Z}_2$, define

$$\varphi(1)(x) = x^{-1}$$

(which is an automorphism since A is commutative).
Then the dihedral group D_n is isomorphic to $A \rtimes_\varphi B$. Rotation corresponds to $(1, 0)$ and reflection corresponds to $(1, 1)$.

Wreath Products

There is a special type of semidirect product that often comes in handy. Given two groups A and B define a left action of B on A^B by

$$(b * \boldsymbol{a})_\beta = \boldsymbol{a}_{b^{-1}\beta}$$

Here we have written the index as β but note that $\beta \in B$. Thus, b simply permutes the elements of vector $\boldsymbol{a}$ which are indexed by B.

Note that the map $\boldsymbol{a} \mapsto b * \boldsymbol{a}$ is an automorphism $\varphi(b)$ of A^B.

The group A^B is called the base group and B the top group.

Definition

The wreath product $A \wr B$ of A and B is the semidirect product $A^B \rtimes_\varphi B$.

Wreath Products II

We have chosen to have the top group operate on the base group on the left, which causes the b^{-1} in the definition.

Note that one writes the top group on the right in $A \wr B$, leading to minor cognitive dissonance.

We could also have B act on the right on the base group:

$$(\boldsymbol{a} * b)_\beta = \boldsymbol{a}_{\beta b}$$

Exercise

Compare the two definitions.

Restricted Wreath Products

In the last construction we used the full product

$$A^B = \prod_B A$$

It is not hard to check that we could also perform our construction with

$$\coprod_B A$$

instead: the sequences now have finite support. Of course, for B finite this distinction is immaterial.

The corresponding group is called the restricted wreath product of A and B.

We will not introduce new notation and just mention if we mean the restricted wreath product.

- Group Representations
- Congruences
- Generators and Cayley Graphs
- Products

5 The Word Problem

Putting Things Together

Here is a slightly more general problem: suppose we want to generate a group H that is given implicitly by some finite representation

$$H = \langle g_1, \ldots, g_k \mid E_1, \ldots, E_l \rangle$$

where the E_i are equations (or relators) using the generators such as $g_1^2 = 1$, $g_2 g_3 = g_3 g_2$ and so on.

In principle, there is a simple algorithm: we start with the identity element, and keep multiplying what we already have by the generators. Multiplication `y = x * g` is done by concatenating words, and then simplifying according to the rewrite rules E_i.

Example

$\langle a \mid a^n = 1 \rangle$ will produce the elements $a, a^2, \ldots, a^{n-1}, 1$, in this order.

Example: Dihedral Group

We already have seen that two generators a and b and the three equations

$$a^4 = 1, b^2 = 1, ba = a^3 b.$$

produce the irreducible elements

$$1, a, aa, aaa, b, ab, aab, aaab$$

In this case, the group is finite and the algorithm duly terminates.

Some Problems

In general, the group H will be infinite, so the algorithm may not terminate. E.g., think about

$$H = \langle a \mid - \rangle$$

There is one generator, but no simplification rules at all.

This should produce (a group isomorphic to) $\mathbb{Z}$, rather than just a semigroup. At present, our algorithm fails in this sense (apart from not terminating).

The solution is to add inverses to the generators:

For every letter g also include another letter $\overline{g}$ and the rules

$$g\overline{g} = \overline{g}g = 1$$

Enlarging the Alphabet

Then for the enlarged alphabet

$$\Sigma = \{g_1, \ldots, g_k, \overline{g}_1, \ldots, \overline{g}_k\}$$

we have a homomorphism (of monoids)

$$\eta : \Sigma^\star \to H$$

that evaluates the words over Σ to the corresponding group elements according to the defining relations. Needless to say

$$\eta(\overline{g}) = \eta(g)^{-1}.$$

Exercise

Explain why it is not necessary to enlarge the alphabet whenever H is finite.

More Word Processing

Another problem is that the simplification may not be so easy: for this algorithm to work we need to be able to check if a newly generated word u represents the same group element as some already encountered word v.

Since witnesses are not unique in general, to check equality we will need some kind of normal form, some canonical way to write u and v in terms of the generators. We could agree that the length-lex minimal word is the right normal form, but how do we compute this particular representation?

Our only hope is to check whether

$$uv^{-1} \text{ somehow simplifies to } 1$$

using the given rewrite rules. Note that it is convenient to have formal inverses for this approach.

The Word Problem

Problem:	**Word Problem**
Instance:	Generators A, rewrite rules R and a word w in $\Sigma^\star$ describing some group element.
Question:	Can w be simplified to 1 via the rules R?

Theorem (Novikov, Boone, Britton)

The Word Problem is undecidable in general.

In fact, the Word Problem remains undecidable for some fixed A and R.

But as we have seen, for some sufficiently simple groups the word problem can even be handled by a finite state machine.

Opening the Flood Gates

The proof ideas for this theorem were then used to show that just about any question about a finitely presented group G turns out to be undecidable.

- Word Problem: Given a word w, does it denote 1?
- Finiteness: Is G finite?
- Commutativity: Is G commutative?
- Cyclicity: Is G cyclic?
- Isomorphism: Given two such groups, are they isomorphic?

Of course, special cases are decidable. But in general finitely presented groups are too complicated to be handled computationally.

A Bizarre Result

Here is a result that shows how atrociously complicated finitely presented groups can be.

Theorem (Novikov)

There is a finitely presented group that contains an isomorphic copy of every finitely presented group.

This sounds utterly impossible at first glance: there are only countably many finitely presented groups and they can surely be wrapped up into a single group. But why should that group be finitely presented itself?

But, Novikov's theorem is a consequence of computability theory and a result in group theory: a finitely generated group is isomorphic to a subgroup of a finitely presented one if, and only if, it is recursively presented (replace finitely many equations with a semi-decidable set of equations).

Practical Computing in Groups

There is still a lot that can be computed effectively and even efficiently in groups, though some of the algorithms are quite sophisticated (and incorporate a lot of group theory knowledge).

An excellent public domain package for computational group theory is GAP, available at

`http://www-gap.dcs.st-and.ac.uk/~gap`

Rather large groups such as the group of motions of Rubik's Cube can easily be handled easily by GAP.

There is even a group at CUNY (Baumslag) that tries to implement an interactive system that can be used to solve undecidable problems in group theory – at least in some cases.