CDM

More on Groups

Klaus Sutner

Carnegie Mellon University

Let's take another look at D_4.
As usual, α denotes rotation and β reflection along an axis.

We know how describe all the group elements in terms of α and β:

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

It is not hard to check that all this makes geometric sense.

We can argue about D_4 by using 3 identities as simplification rules:

$$\alpha^4 \rightsquigarrow 1 \qquad \beta^2 \rightsquigarrow 1 \qquad \beta\alpha \rightsquigarrow \alpha^3\beta$$

The first two are clear, the last is justified by trying to sort the word.

We can now evaluate a product by a chain of rewrites:

$$\beta\alpha \cdot \beta\alpha \rightsquigarrow \beta\alpha^4\beta \rightsquigarrow \beta^2 \rightsquigarrow 1$$

Wild Idea:

Can we think of this purely as an operation on words, not some actual algebraic reasoning?

No Multiplication Table

If this works, we don't need a multiplication table, we can just "multiply" words over the alphabet $\{\alpha, \beta\}$: first just concatenate them, then use the simplification rules to shorten the result.

We have to make sure that the rules produce a unique normal form: for any string x, no matter how we apply the rules, we wind up with a unique normal form $\nu(x)$ that cannot be further simplified, a so-called irreducible word.

For reasons of sanity, we should use different symbols, say a for α and b for β to make clear that we are just pushing letters around.

If this works, then we can handle the general dihedral group D_n similarly.

$$a^n \rightsquigarrow 1 \qquad b^2 \rightsquigarrow 1 \qquad ba \rightsquigarrow a^{n-1}b$$

The irreducible words here should be

$$a^i \qquad \text{or} \qquad a^i b$$

where $0 \leq i < n$.

We will show that every word has such a normal form, but skip over uniqueness (find a suitable invariant).

Lemma

Every word $x \in \{a, b\}^\star$ has a unique normal form wrto the rules for D_n.

Proof. By using the first two rules we can rewrite $x \neq \varepsilon$ as

$$a^{e_1} b^{e'_1} a^{e_2} b^{e'_2} \ldots a^{e_k} b^{e'_k}$$

where $0 < e_i < 4$ and $e'_i = 1$ except that perhaps $e_1 = 0$ and $e'_k = 0$.

If $k > 1$ then we can use rule 3 to get

$$a^{e_1+3e_2} b^{e'_1+e'_2} a^{e_3} \ldots a^{e_k} b^{e'_k}$$

Done by induction on k.

□

We need a bit of machinery to explain how to compute in 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.

Definition

The kernel of a homomorphism $f : G \to H$ is defined as

$$\ker f = \{\, x \in G \mid f(x) = 1 \,\}.$$

Hence

$$f(x) = f(y) \iff y^{-1}x \in \ker f$$

This is slightly different from the kernel relations in combinatorics, but close enough to warrant the same name.

Note that f is injective (a monomorphism) iff the kernel is trivial: $\ker f = 1$.

Proposition

The kernel of a homomorphism is always a subgroup.

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 alternating group

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

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

Example 3

Consider the multiplicative subgroup

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

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

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

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

$$\begin{array}{cccc} 1 & 2 & 4 & 7 \\ 2 & 4 & 7 & 1 \\ 4 & 7 & 1 & 2 \\ 7 & 1 & 2 & 4 \end{array}$$

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

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

$$f : A \to A$$

to describe equivalence relations on sets (canonical choice 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.

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. □

Theorem (Lagrange 1771)

Let G be a finite group, and H any subgroup of G.
Then the order of H divides the order of G.

One defines $[G : H] = |G|/|H|$ to be the index of H in 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 (and useful) equivalence relations.

As a special case, the order of any group element divides the order of the whole group.

Hence, $a^n = 1$ for any $a \in G$ where $n = |G|$.

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

Theorem (Euler-Fermat)

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

Definition

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

$$x \sim x', y \sim y' \quad \text{implies} \quad xy \sim x'y'.$$

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 $[x] = [x']$ and $[y] = [y']$. Then our definition makes sense only if $[xy] = [x'y']$.

But that's exactly what the congruence property guarantees.

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.

Cheesy Example: Chinese Remainder

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

$$\begin{array}{rccl} f: & \mathbb{Z} & \longrightarrow & \mathbb{Z}_p \times \mathbb{Z}_q \\ & x & \longmapsto & (x \bmod p, x \bmod q) \end{array}$$

f is an epimorphism and has kernel $H = n\,\mathbb{Z}$; with quotient $\mathbb{Z}/(n\mathbb{Z}) = \mathbb{Z}_n$.
We get the necessary isomorphism via

$$\begin{array}{rccl} \bar{f}: & \mathbb{Z}_n & \longrightarrow & \mathbb{Z}_p \times \mathbb{Z}_q \\ & [x] & \longmapsto & (x \bmod p, x \bmod q) \end{array}$$

Hence we can either compute

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

The isomorphism transports results back and forth between $\mathbb{Z}_n$ and $\mathbb{Z}_p \times \mathbb{Z}_q$, but computationally there is a difference. E.g., this can be exploited to fake high-precision computations with small word sizes (pick two primes a little smaller than 2^{64}).

This isomorphism is actually a ring isomorphism and is critical for the correctness proof for RSA.

Start with one generator g that produces some cyclic group H.

If H is finite, everything is straightforward.

$$H = \{ g^i \mid 0 \leq i < \mathsf{ord}(g) \}$$

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

If H is infinite, then H is isomorphic to $\mathbb{Z}$.

Somewhat surpassingly, even just 2 generators can produce much more complicated groups.

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

Proposition

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

All conjugates cac^{-1}, cbc^{-1} also work.

An old CDM homework assignment once asked for a characterization of all 2-element generators for the symmetric group. Oh well, ...

How Bad Can It Be?

How complicated could $\langle a, b\rangle \subseteq \mathfrak{S}_n$ possibly be?

Here are four pairs of generators G_k over $[k]$ for $k = 11, 12, 22, 23$.

$$G_{11} = (2\,10)(4\,11)(5\,7)(8\,9),\ (1\,4\,3\,8)(2\,5\,6\,9)$$

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

$$\begin{aligned} G_{22} = {} & (1\,13)(2\,8)(3\,16)(4\,12)(6\,22)(7\,17)(9\,10)(11\,14), \\ & (1\,22\,3\,21)(2\,18\,4\,13)(5\,12)(6\,11\,7\,15)(8\,14\,20\,10)(17\,19) \end{aligned}$$

$$\begin{aligned} G_{23} = {} & (1\,2)(3\,4)(7\,8)(9\,10)(13\,14)(15\,16)(19\,20)(21\,22), \\ & (1\,16\,11\,3)(2\,9\,21\,12)(4\,5\,8\,23)(6\,22\,14\,18)(13\,20)(15\,17) \end{aligned}$$

The groups $\langle G_k\rangle$ are called Mathieu groups and they have sizes

$$7920 \qquad 95040 \qquad 443520 \qquad 10200960$$

They play a role in the classification of finite simple groups.

Theorem (Dixon 1969)

Two random elements of the symmetric group on n points generate $\mathfrak{S}_n$ or $\mathfrak{A}_n$ with probability tending to 1 as $n \to \infty$.

So the generators for the Mathieu groups are indeed very carefully chosen.

Generators also build a bridge between groups and labeled digraphs.

Definition

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

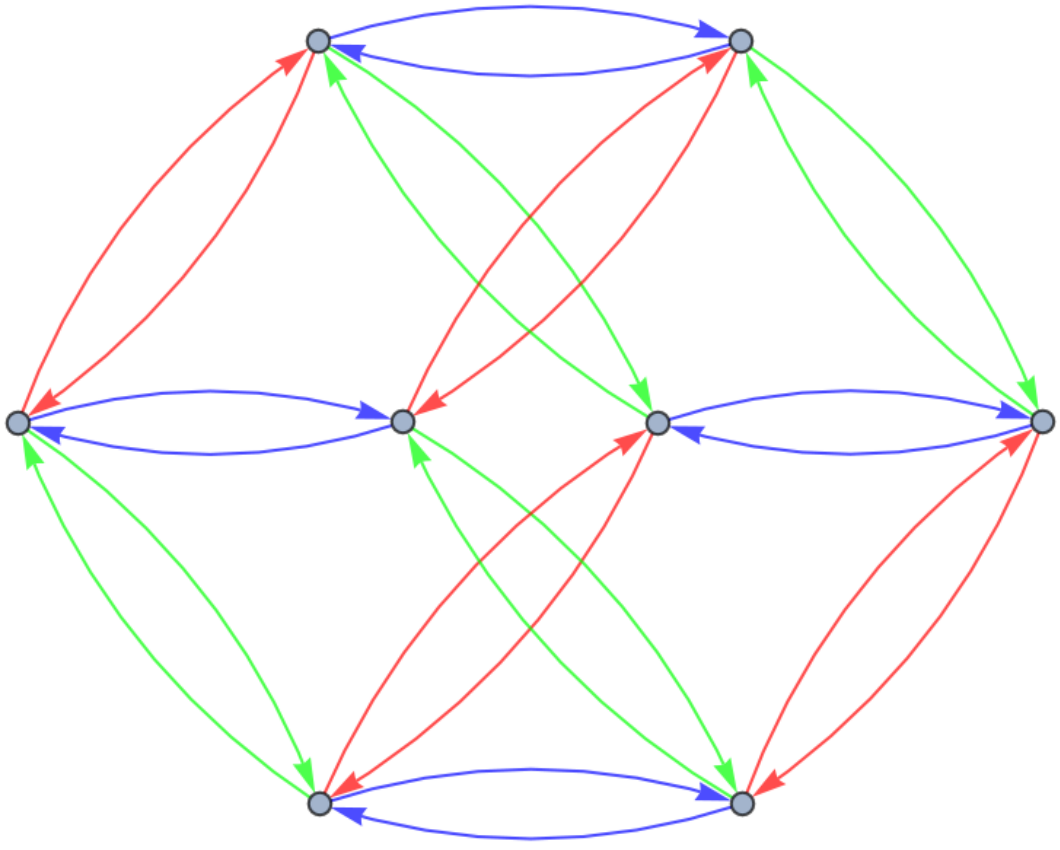
$$x \xrightarrow{a} x \cdot a \qquad \text{where } x \in G, a \in X$$

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

Lemma

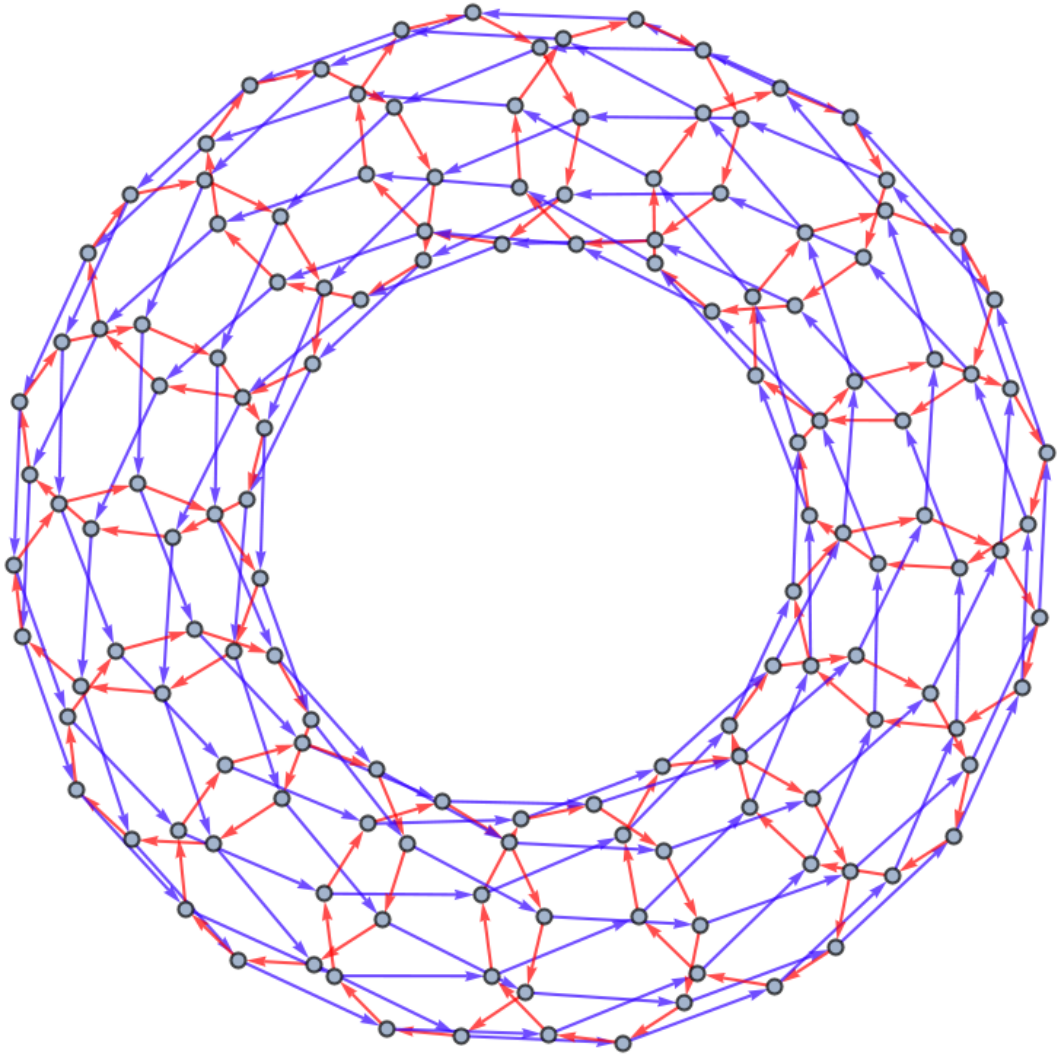
$\Gamma_r(G)$ is strongly connected and regular.

Example

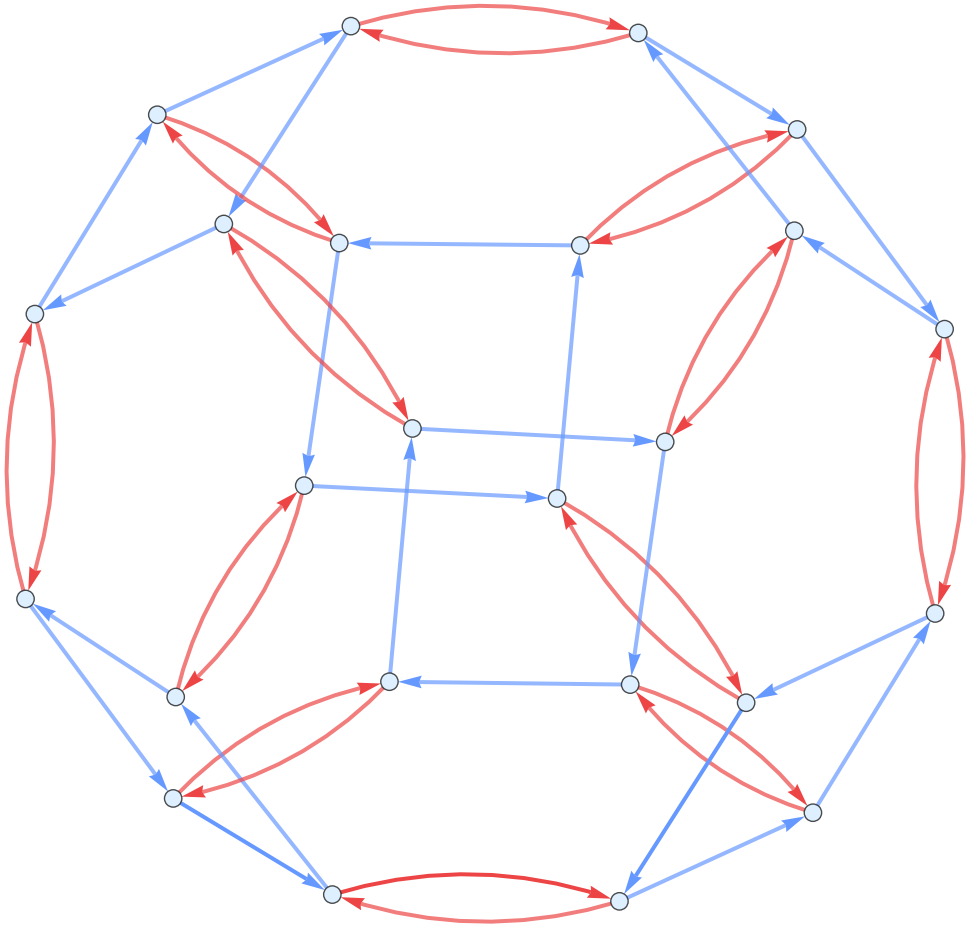


The Cayley graph of the Abelian group $\mathbb{Z}_2 \times \mathbb{Z}_2 \times \mathbb{Z}_2$.

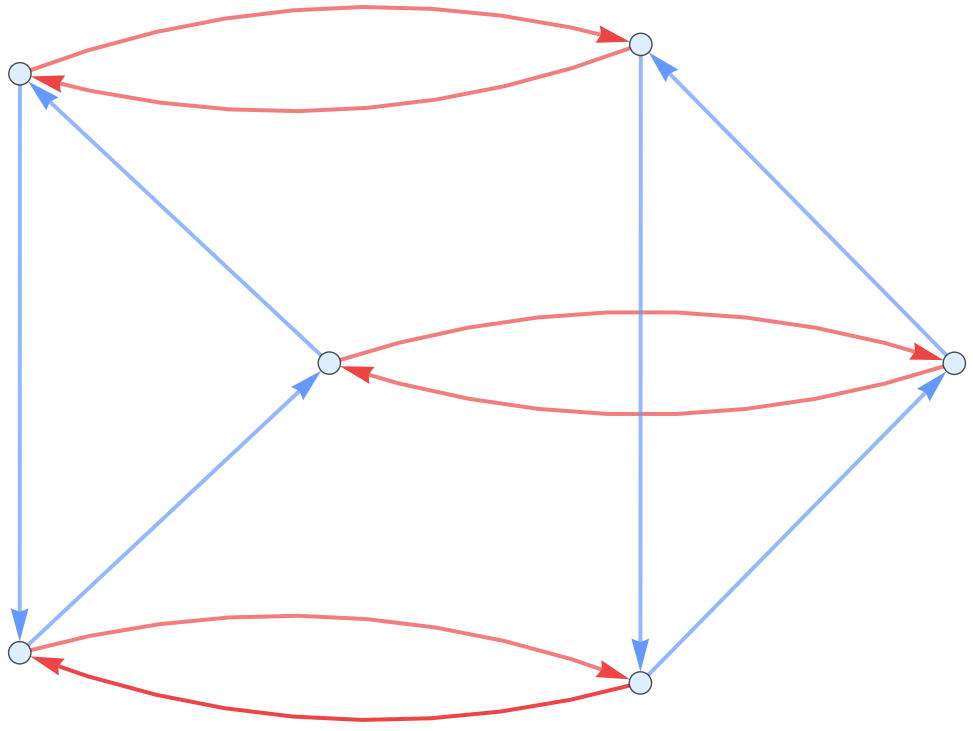
Example



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



The Cayley graph of the symmetric group of four letters.



Exercise

Determine the group given by this Cayley graph.

So suppose we have generators $X = g_1, \ldots, g_k$ in some finite group G. We can compute the generated subgroup $H = \langle X \rangle$ in stages by forming all possible products: in a finite group inverses appear automatically since $a^{-1} = a^{k-1}$ where $k = \mathsf{ord}(a)$.

- $H_0 = X \cup \{1\}$
- $H_{n+1} = H_n \cdot H_n$
- stop when no new elements appear, $H = \bigcup H_i$

As written, this is hopelessly redundant: we are recomputing the same products over and over again.

We need a better algorithm that avoids recomputation as much as possible.

Example

Let $a = [2, 1, 3, 4]$ and $b = [2, 3, 4, 1]$ in $\mathfrak{S}_4$.
The table shows the new elments added at each step.

level	new elements
0	$[1, 2, 3, 4], [2, 1, 3, 4], [2, 3, 4, 1]$
1	$[1, 3, 4, 2], [3, 2, 4, 1], [3, 4, 1, 2]$
2	$[2, 4, 1, 3], [3, 1, 4, 2], [3, 4, 2, 1], [4, 1, 2, 3], [4, 3, 1, 2]$
3	$[1, 4, 2, 3], [3, 1, 2, 4], [4, 1, 3, 2], [4, 2, 1, 3], [4, 3, 2, 1]$
4	$[1, 2, 4, 3], [1, 3, 2, 4], [1, 4, 3, 2], [3, 2, 1, 4], [4, 2, 3, 1]$
5	$[2, 1, 4, 3], [2, 3, 1, 4], [2, 4, 3, 1]$

Cayley graphs help to tackle the recomputation problem.
We have some ambient group G and a set $X \subseteq G$ of generators; we need to compute $H = \langle X \rangle \subseteq G$.

Assume for a moment that we have the Cayley graph $\Gamma_r(H)$ of H.

Finding all points in H is then just a graph exploration problem: starting at vertex 1, we are trying to find all points reachable via paths labeled by a sequence of generators.

To avoid recomputation, we must not epxlore the same vertex repeatedly. But this is exactly what graph exploration algorithms like DFS and BFS do.

Obviously, we emphatically do not have a standard data structure for the Cayley graph of H (never mind G) like an adjacency matrix.

But, all we really need is

- a data structure for group elements, and
- a way to compute the vertex $a \cdot g$ in the ambient group G, given an element a and a generator $g \in X$.

We run BFS in a virtual graph, using a succinct representation.

Building a Spanning Tree

More precisely, we can build a spanning tree T of $\Gamma_r(H)$ by running BFS in our virtual graph:

- 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 $x \xrightarrow{g} y$.
- Stop when no new nodes appear.

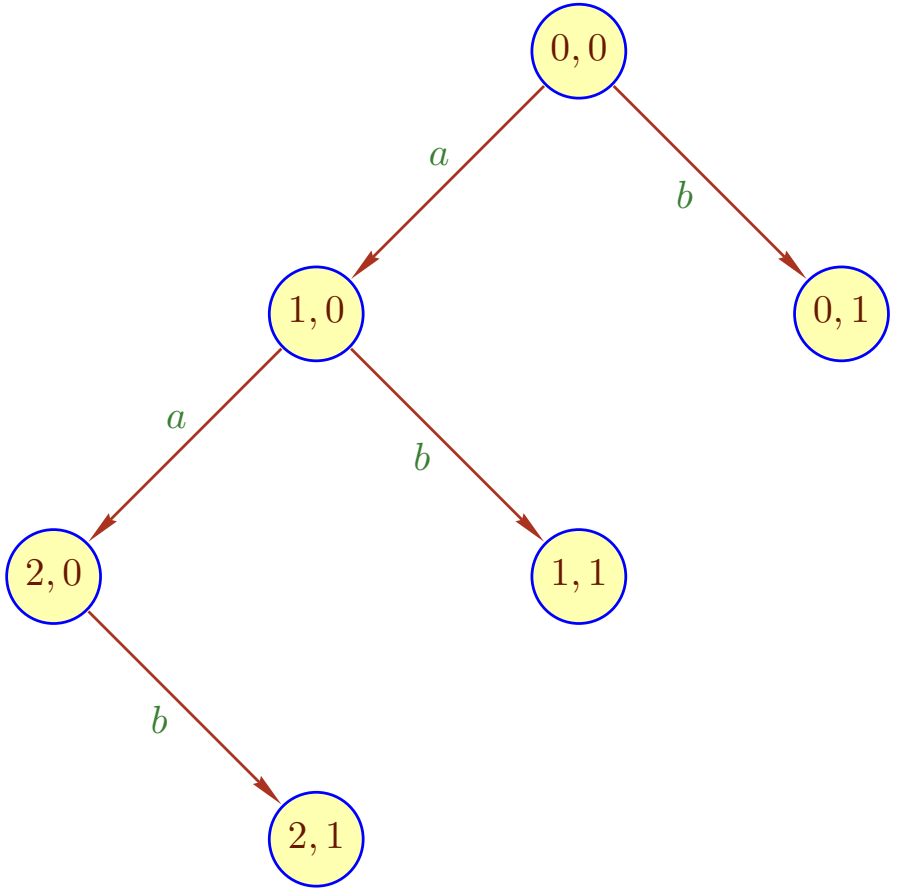
In practice we would also keep track of non-tree edges.

Example 1

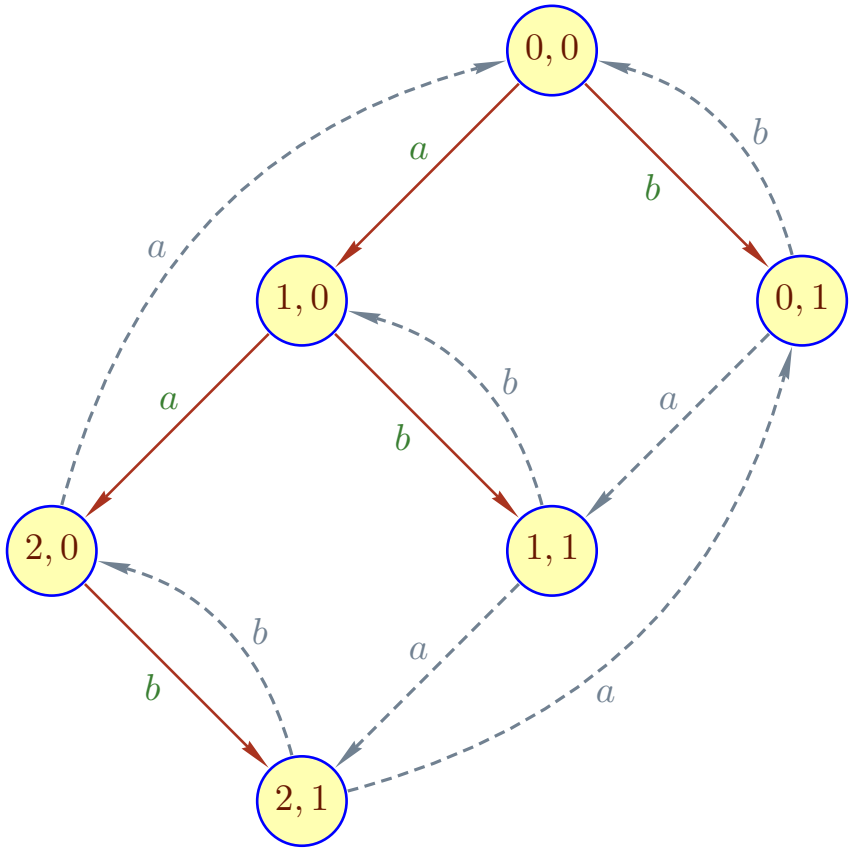
Consider the two generators $a = (1, 0)$ and $b = (0, 1)$ in $\mathbb{Z}_3 \times \mathbb{Z}_2$.

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 spanning tree constructed by the algorithm.

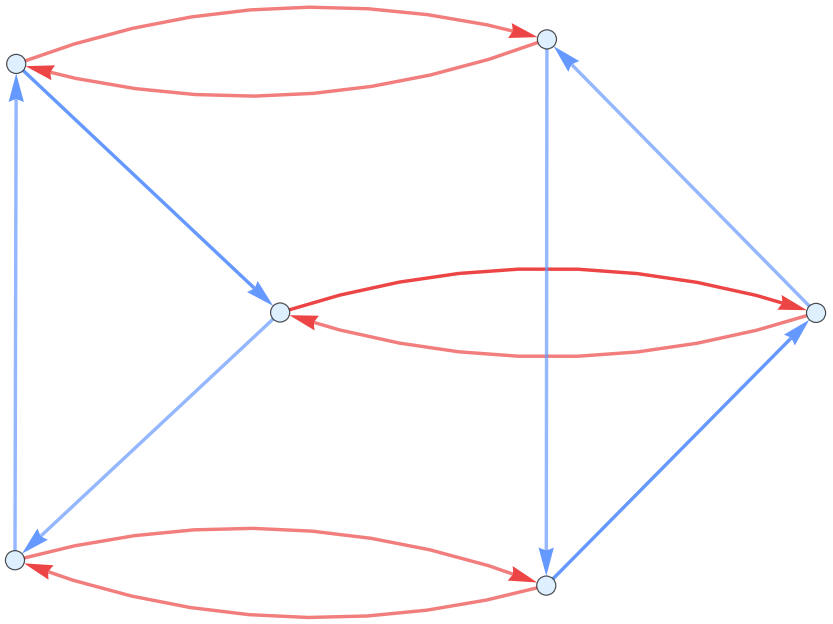


Same with non-tree edges, producing the full Cayley graph.

Example 2

Standard generators $a = (1, 2)$ and $b = (1, 2, 3)$ in $\mathfrak{S}_3$.

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



Example 3

Standard generators $a = (1, 2)$ and $b = (1, 2, 3, 4)$ for $\mathfrak{S}_4$.

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.

Recall: Dihedral Groups

We already have seen how to express the dihedral group D_n with two generators a and b and the three group equations

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

by interpreting the equations as rewrite rules, in particular as simplification rules on strings over $\{a, b\}$.

The next step is to try to use a similar approach to construct the free group $F(X)$ for some set X of generators.

This sounds plausible given the free structures we have already seen:

type	free structure
magma	binary trees
semigroups	non-empty words
monoids	all words
groups	???

If you don't like the trees, replace them by fully parenthesized expressions.

The dihedral example is slightly misleading since the group is finite.
In general, we have to work a bit harder.

Suppose there is only one generator a, we want the 1-generated free group. This should produce the additive group $\mathbb{Z}$, but using words over $\{a\}$ all we get is $\mathbb{N}$, represented by $a^\star$.

To fix this issue we enlarge the alphabet by a special symbol $\overline{a}$ for the inverse of the generator a

$$\Sigma = \{a, \overline{a}\}$$

A word in $\Sigma^\star$ is reduced if it contains no subword $a\overline{a}$ or $\overline{a}a$.
In other words, it is irreducible for the rules $a\overline{a} \rightsquigarrow 1$, $\overline{a}a \rightsquigarrow 1$.

Reduced words then naturally represent $\mathbb{Z}$:

$$R = a^\star \cup \overline{a}^+$$

And the Operation?

To multiply two words in R we

- First concatenate them, and then
- apply the reduction rules till we are back in R.

$$a^7 \cdot \overline{a}^4 \rightsquigarrow a^3$$

More Generally

Suppose we have a list $X = g_1, \ldots, g_k$ of generators.
The symmetrized alphabet or involutive alphabet of X is

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

Words over Σ form the free monoid with generators Σ.

We think of the bar operation as an involution: $\overline{\overline{a}} = a$.
The reduction identities then simply look like

$$g\,\overline{g} = 1 \qquad g \in \Sigma$$

We can also concoct a formal inverse of a whole string $x \in \Sigma^\star$ by

$$\overline{x} = \overline{x}_n \overline{x}_{n-1} \ldots \overline{x}_2 \overline{x}_1$$

Claim:
The reduction identities define a congruence $\equiv$ on $\Sigma^\star$.

Informally, the congruence is given by $x \equiv y$ iff
x can be transformed to y by erasing or inserting adjacent symbols $g\,\overline{g}$.

For example

$$\overline{a}a\overline{a}b\overline{b}b \equiv \overline{a}b\overline{c}c$$

Note that we really get a congruence, not just an equivalence relation.
The language of irreducibles is quite simple, it is regular:

$$\Sigma^\star - \bigcup_{g \in \Sigma} \Sigma^\star\, g\, \overline{g}\, \Sigma^\star$$

Reduction Algorithm

The congruence is really the kernel relation for the normal form map for the rules $g\,\overline{g} = 1$. E.g.

$$\nu(\overline{a}a\overline{a}b\overline{b}b) = \nu(\overline{a}b\overline{c}c) = \overline{a}b$$

Computing normal form is easy with a stack. Let x be any string in $\Sigma^\star$.

$S =$ nil
while $a = x$.next() **do**
    if top$(S) = \overline{a}$
    then pop(S)
    else push(S, a)
return S

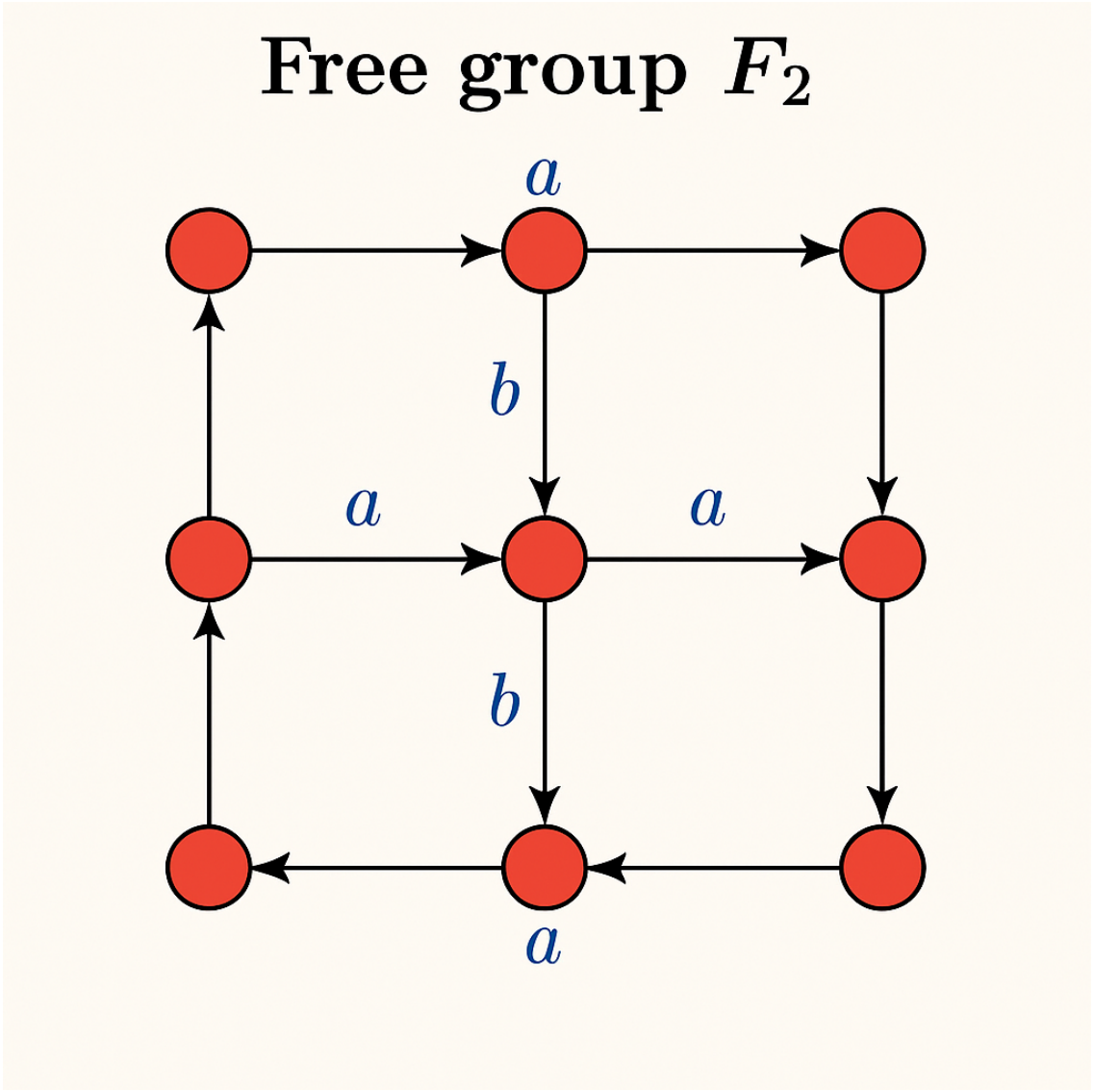
To get the free group all we have to do is take the quotient monoid of $\Sigma^\star$ by the reduction congruence $\equiv$.

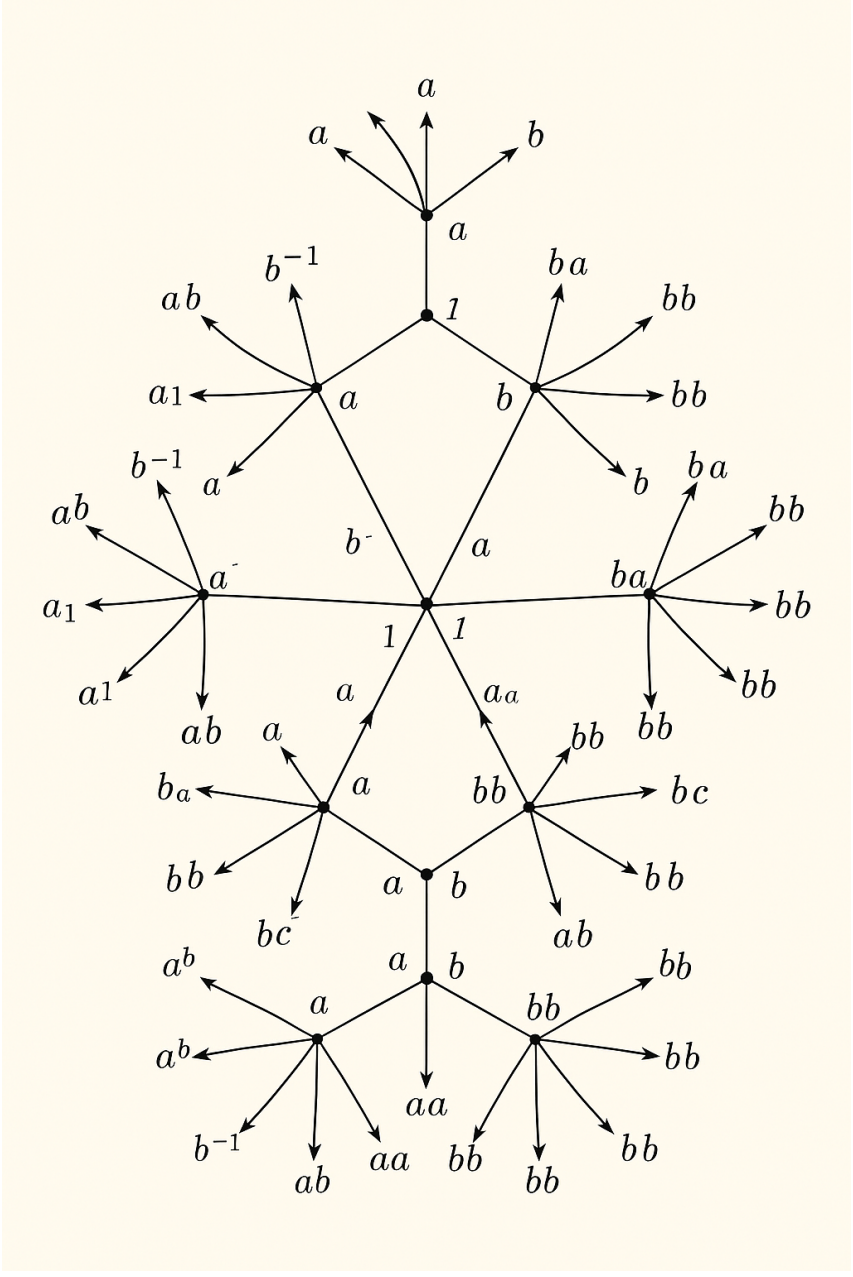
Lemma

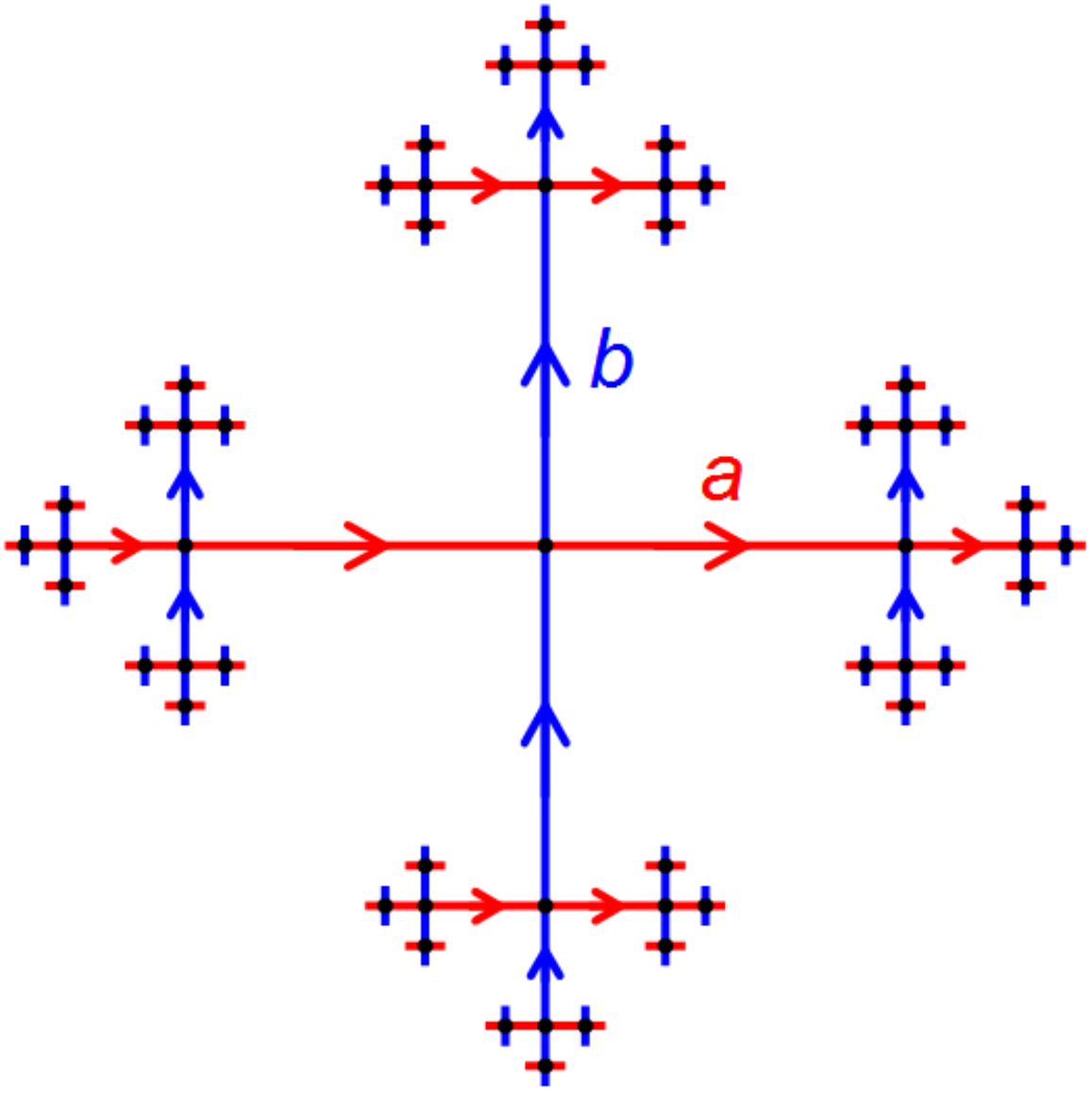
The free group over the set of generators X is the quotient monoid

$$F(X) = \Sigma^\star / \equiv$$

So we have a monoid and a monoid congruence, and we get a group.


Free group F_2
a
b
a
a
b
a


a
a
b
a
b⁻¹
ba
ab
1
bb
a1
a
b
bb
b⁻¹
a
b
ba
ab
bb
b⁻
a
a⁻
ba
a1
bb
1
1
a1
a
aa
ab
a
bb
bb
ba
a
bb
bc
bb
a
b
bb
bc⁻
ab
ab
a
b
bb
a
bb
ab
bb
aa
b⁻¹
ab
aa
bb
bb
bb


b
a

We can obtain groups with special properties by adding more identities

$$x = y \qquad \text{where } x, y \in \Sigma^\star$$

More precisely, given a set of identities $\mathcal{E}$ we can form a group congruence $\equiv_\mathcal{E}$ and then take the quotient $F(X)/\equiv_\mathcal{E}$.

E.g., to construct D_4 this way we use

$$\mathcal{E}: \qquad a^n = 1, b^2 = 1, ba = a^{n-1}b$$

It then follows that

$$\overline{a} \equiv_\mathcal{E} \overline{a}a^n \equiv_\mathcal{E} a^{n-1}$$
$$\overline{b} \equiv_\mathcal{E} \overline{a}b^2 \equiv_\mathcal{E} b$$

so in this case we don't really need the symmetrized alphabet.

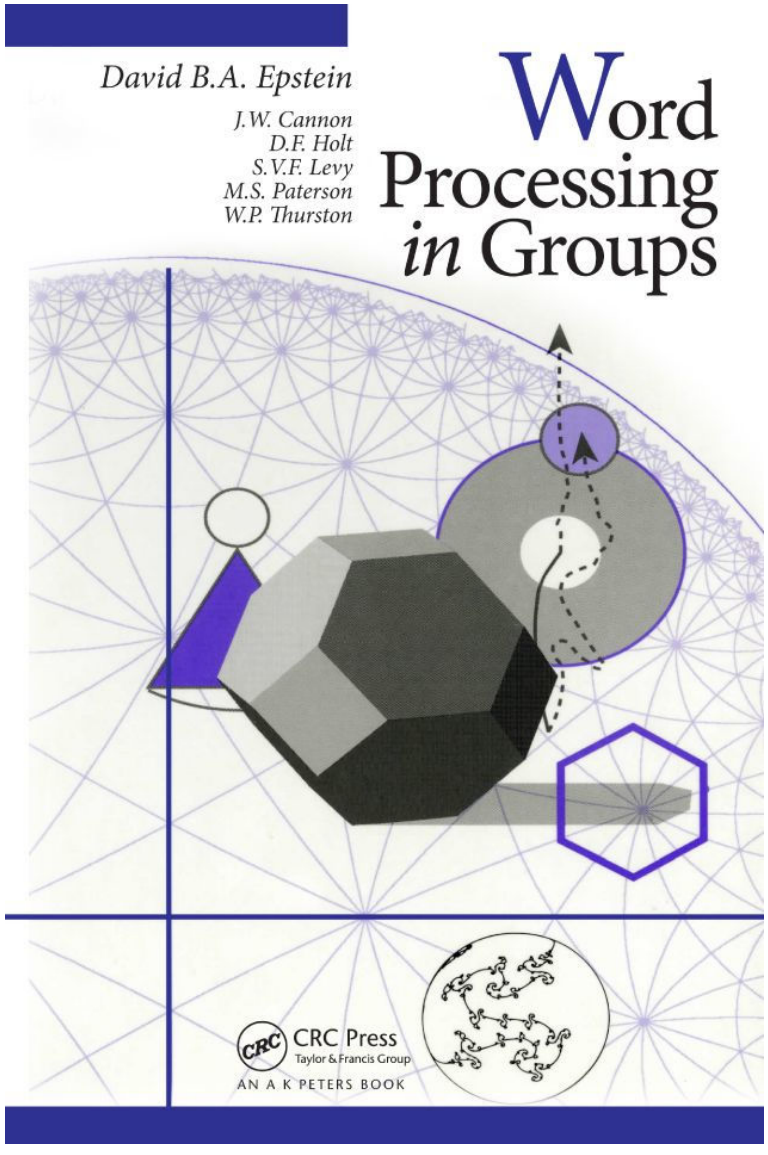
At any rate, $F(a, b)/\equiv_\mathcal{E}$ is isomorphic to D_n.

But Why?

Because we now have a way to work in D_n that only requires strings over Σ and some word processing, in particular the application of rewrite rules.

At no point do we have to worry about the actual algebraic objects.

Hence we can use e.g. FSMs, pushdown automata or other simple word processing operations to do algebra. We have a direct algorithmic handle and one can hope to transfer results or at least methods from automata theory.


David B.A. Epstein
J.W. Cannon
D.F. Holt
S.V.F. Levy
M.S. Paterson
W.P. Thurston
Word Processing in Groups
CRC Press
Taylor & Francis Group
AN A K PETERS BOOK

Recall that we have formal inverse of a string $x \in \Sigma^\star$:

$$\overline{x} = \overline{x}_n \overline{x}_{n-1} \ldots \overline{x}_2 \overline{x}_1$$

Hence we could replace identities by so-called relators $z \in \Sigma^\star$:

$$x = y \qquad \text{translates to} \qquad x\,\overline{y} = 1$$

The last obfuscation step is to just write $x\,\overline{y}$, the "$= 1$" part is understood. For D_n we would then simply write

$$\langle\, a, b \mid a^n, b^2, \overline{a}\overline{b}a^{n-1}b \,\rangle$$

More generally, we can now construct groups over generators X by writing down a list of identities/relators:

$$G = \langle X \mid (E_i)_{i \in I} \rangle$$

Definition

A group is finitely presented iff it is isomorphic to a group of this form where both X and I are finite.

Note that is not exactly clear whether a group described by some other means is finitely generated. But certainly finitely generated groups are perfectly good input for algorithms.

Examples

$\langle\, a \mid a^n \,\rangle$	cyclic group order n
$\langle\, a, b \mid - \,\rangle$	free group on 2 generators
$\langle\, a, b \mid a^n, b^2, abab \,\rangle$	dihedral group order n
$\langle\, a, b, c \mid ab = c, bc = a, ca = b \,\rangle$	Fibonacci group 3 generators

The Fibonacci group on 3 generators is actually isomorphic to the quaternion group.

It generalizes nicely to n generators

$$g_1 g_2 = g_3, \ldots, g_{n-2} g_{n-1} = g_n, g_{n-1} g_n = g_1, g_n g_1 = g_2$$

Wild fact: these groups are finite iff $n = 2, 3, 4, 5, 7$.

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 relators are

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

It is utterly unclear what the size of this group is, never mind its properties. With some effort one can show

Lemma

These identities produce the symmetric group on n points.

We can explain the construction also purely in terms of taking a quotient by a normal subgroup.

- Let $R \subseteq F(X)$ be a (finite) set of relators.
- Let H be the normal subgroup of $F(X)$ generated by

$$\{ x\, r\, x^{-1} \mid r \in R, x \in F(X) \}$$

- Then the group $G = F(X)/H$ is (isomorphic to) $\langle X \mid R \rangle$.

Quotients and congruences produce the same result as our word processing approach, but the latter is directly computational.

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

There is a cardinality argument.
Let $A \subseteq \mathbb{N}$ and define

$$G_A = \langle a, b \mid a^k b = b a^k : k \in A \rangle$$

There are uncountably many, non-isomorphic G_A, but there are only countably many finitely presented such groups.

So most of the G_A are not finitely presented..

More concretely, one can show that a wreath product (more later) of $\mathbb{Z}$ and $\mathbb{Z}$ fails to be finitely presented, but the argument requires work.

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

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

Does this still describe a group?

It does, but a boring one: the trivial group 1.

So how do we multiply x and y in $G = \langle X \mid E \rangle$?

- Concatenate the strings $z = xy$, then
- simplify z using the identities till we get normal form.

In principle, this is straightforward. Alas, there are two problems:

- Termination
  An identity $u = v$ translates into two rules $u \rightsquigarrow v$ and $v \rightsquigarrow u$.
  One has to be careful not to run into loops and infinite chains.
- Confluence
  We need to make sure that the normal form is unique, it must not matter how exactly we apply the rules.

There is a substantial literature on rewrite systems that deals with these issues.

Specifically

Computational difficulties arise e.g. in the following innocent question:

> Given two strings u and v over Σ.
> Do they represent the same group element?

In other words, we need to check whether $\nu(u) = \nu(v)$.

Alternatively, we have to determine whether $u\,\overline{v}$ somehow simplifies to 1.

Assume the sets of generators and identities are finite.

Problem:	**Word Problem for Finitely Presented Groups**
Instance:	Generators X, identities $\mathcal{E}$ and a word w in $\Sigma^\star$.
Question:	Does w simplify to 1 via $\mathcal{E}$?

Theorem (Novikov, Boone, Britton 1955)

The Word Problem is undecidable in general.

In fact, the Word Problem remains undecidable for some fixed X and $\mathcal{E}$.

On the other hand, for some sufficiently simple groups the word problem can even be handled by a finite state machine.

Informally

We want to check whether w simplifies to 1.

If we could compute an upper bound on the length of words needed in a a corresponding chain of substitutions, we could simply try out all possible chains by brute force. This would solve the Word Problem.

Hence there is no computable bound on these word lengths.

Theorem (E. Post, A.A. Markov 1947)

The word problem for finitely presented monoids is undecidable.

The question here is whether a word can be rewritten to another. This theorem is often considered to be the first undecidability result in pure mathematics (as opposed to results inside logic).

A simple example due to G. Makanin looks like this:

$$\mathcal{E}: \quad ccbb = bbcc \qquad bcccbb = cbbbcc \qquad accbb = bba$$
$$abcccbb = cbba \qquad bbccbbbbcc = bbccbbbbcca$$

The Novikov-Boone theorem on groups is significantly harder to prove.
For commutative monoids the word problem is decidable.

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.

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. To be sure, there are only countably many finitely presented groups, so perhaps they could be wrapped up into a single group. But why should that monster 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 iff it is recursively presented: replace finitely many equations with a semidecidable set of equations.

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

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