CDM

Algebra and FSM

Klaus Sutner

Carnegie Mellon University

Fall 2025

1 **Kleene Algebras**

2 **Left Quotients**

Hermann Weyl

Our mathematics of the last decades has wallowed in generalizations and formalizations. But one misunderstands this tendency if one thinks that generality was sought merely for generality's sake. The real aim is simplicity: natural generalization simplifies since it reduces the assumptions that have to be taken into account.

Axiomatic versus Constructive Procedures in Mathematics, 1953

Algebra to the Rescue

Algebra is the Big Simplifier:

If a problem can be reasonably expressed by algebra, at least some of the answers will come for free.

Fortunately, algebra is the key to several useful generalizations in the realm of finite state machines. This is very different from other models of computation.

Math versus CS

One issue we have to contend with when applying algebra to CS is that the structures relevant there are often weaker (more general) than their classical counterparts.

Example: groups vs semigroups, rings vs semirings.

There is really no way around this: if we try to explain finite state machines algebraically, the theory has to accommodate the application. Needless to say, this may cause some additional pain.

Semirings

Definition

A semiring $\mathcal{S}$ is a structure

$$\langle S; \oplus, \otimes, 0, 1\rangle$$

of signature $(2, 2, 0, 0)$ where

- $\langle S; \oplus, 0\rangle$ is a commutative monoid
- $\langle S; \otimes, 1\rangle$ is a monoid
- $\otimes$ distributes over $\oplus$ on the left and right:
  $x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z)$ and
  $(y \oplus z) \otimes x = (y \otimes x) \oplus (z \otimes x)$.
- 0 is a null with respect to $\otimes$: $x \otimes 0 = 0 \otimes x = 0$.

A semiring is commutative if $x \otimes y = y \otimes x$.

It is idempotent if $x \oplus x = x$.

Examples

All the standard examples (integers, rationals, reals, complexes) are semirings. This pretty lame since they are actually rings ($\mathbb{Z}$) and fields (the rest).

The Boolean semiring has the form

$$\mathbb{B} = \langle \mathbf{2}; \vee, \wedge, \mathsf{ff}, \mathsf{tt} \rangle$$

where the operations are logical 'or' and 'and'.

The relation semiring looks like

$$\mathcal{R}_A = \langle \mathsf{Rel}_A; \cup, \circ, \emptyset, I_A \rangle$$

and has carrier set all binary relations over some set A; addition is set union, multiplication is relational composition, 0 is the empty relation, and 1 is the identity relation.

Tropical Semiring

The tropical semiring or min-plus semiring is defined by

$$\mathsf{TS} = \langle \mathbb{N}_\infty; \min, +, \infty, 0 \rangle$$

Here $\mathbb{N}_\infty$ is $\mathbb{N}$ with an “infinitely large” element ∞ adjoined that behaves properly with respect to $\min$ and $+$, in particular $\min(x, \infty) = x$ and $x + \infty = \infty$.

Quoi?

This may look strange, but shortest path algorithms naturally use this structure.

Matrix Semirings

Apart from examples, it is important to have ways to construct more complicated semirings from simpler ones.

Suppose we have a semiring $\mathcal{S} = \langle S; \oplus, \otimes, 0, 1 \rangle$.

The $n \times n$ matrix semiring over $\mathcal{S}$ has the form

$$\mathcal{S}^{n,n} = \langle S^{n,n}; \oplus, \otimes, \mathbf{0}, \mathbf{1} \rangle$$

where $\oplus$ and $\otimes$ are the matrix operations inherited obtained from $\mathcal{S}$; $\mathbf{0}$ and $\mathbf{1}$ are the appropriate null and identity matrices over S.

Claim: $\mathcal{S}^{n,n}$ is again a semiring.

Language Semiring

Here is the killer app for our purposes.

The language semiring over some alphabet Σ has the form[†]

$$\mathcal{L}(\Sigma) = \langle \mathfrak{P}(\Sigma^\star); \cup, \cdot, \emptyset, \varepsilon \rangle$$

The full language semiring is uncountable and just too big, but it has nice subsemirings such as the recognizable languages or context free languages.

One can even introduce a metric on $\mathcal{L}(\Sigma)$ by setting $\mathrm{dist}(L, K) := 2^{-n}$ where n is minimal such that $L \cap \Sigma^n \neq K \cap \Sigma^n$ for $L \neq K$; $\mathrm{dist}(L, L) = 0$.

This turns $\mathcal{L}(\Sigma)$ into a complete metric space.

[†]Yes, ε should be $\{\varepsilon\}$, but no one does that.

Regular Language Semiring

We are mostly interested in the semiring of recognizable languages:

$$\mathsf{Rec}_\Sigma = \text{ all recognizable } L \subseteq \Sigma^\star$$

We have seen closure under union and concatenation which is covered by the semiring operations.

But how about intersection, complement and Kleene star?

Here is a strange fact, discovered by Kleene: once we add a star operation, we can get intersection and complement for free.

Semirings with Star

We would like to find a way to add some sort of Kleene star operation $x^\star$ to a suitable semiring, something along the lines of

$$x^\star = \sum_{i \geq 0} x^i = x^0 + x^1 + x^2 + \ldots + x^n + \ldots$$

Of course, the infinite sum makes no sense a priori, it's just wishful thinking.

We should expect some difficulties since algebraic operation usually are finitary.

Compelling Examples

Clearly, we know how to handle Kleene star for the language semiring:

$$x^\star = \bigcup_{i \geq 0} x^i = x^0 \cup x^1 \cup x^2 \cup \ldots \cup x^n \cup \ldots$$

Here are two more good examples of semirings where star makes sense:

- Binary relations: Kleene star is reflexive transitive closure.
- Boolean semiring: Kleene star is easy: $x^\star = 1$.

Definition

Definition

A Kleene algebra is a structure

$$\langle A; +, \cdot, ^*, 0, 1 \rangle$$

of signature $(2, 2, 1, 0, 0)$ where $\langle A; +, \cdot, 0, 1 \rangle$ is an idempotent semiring. Moreover, we have

- sumstar identity: $(x + y)^\star = (x^\star \cdot y)^\star \cdot x^\star$
- prodstar identity: $(x \cdot y)^\star = 1 + x \cdot (y \cdot x)^\star \cdot y$
- starstar identity: $(x^\star)^\star = x^\star$
- powerstar identity: $(x^n)^\star x^{<n} = x^\star$

For the powerstar axiom let $x^{<n} = 1 + x + x^2 + \ldots + x^{n-1}$.

This holds for all $n \geq 1$.

Handling Sums

The major difference between Kleene algebras and more familiar structures such as groups, fields or semirings is that Kleene star is an infinitary operation:

$$x^\star = 1 + x + x^2 + \ldots + x^n + \ldots$$

First off, finite sums are easy, we can define a summation operation Σ:

$$\begin{aligned} \Sigma_\emptyset &= 0 \\ \Sigma_{\{x\}} &= x \\ \Sigma_{\{x_1,\ldots,x_k\}} &= x_1 + \Sigma_{\{x_2,\ldots,x_k\}} \qquad k \geq 2 \end{aligned}$$

This is the right associative version; since $\langle A; +, 0\rangle$ is associative, any other definition would produce the same result.

Going Infinite

Burning Question: What about $\Sigma_{\{x_0,x_1,x_2,\ldots\}}$?

We are not doing analysis here, convergence, limits and the like won't help. Instead we explain our Σ operator in terms of index sets:

- $\Sigma_I \left(\Sigma_{J_i}\, x_j\right) = \Sigma_J\, x_j$ $\qquad (J_i)_{i\in I}$ any partition of J
- $\left(\Sigma_I\, x_i\right)\left(\Sigma_J\, y_j\right) = \Sigma_{I\times J}\, x_i y_j$

Roughly, a sum of a sum is a sum, and we have distributivity. We now define

$$x^\star = \Sigma_{\mathbb{N}}\, x^n$$

Order

The following partial order is surprisingly useful:

$$x \leq y \iff x + y = y$$

For language semirings this is just ordinary set inclusion.

But then we can talk about the least solution of an equation.
For example, the language equation

$$X = u\,X + v$$

has least solution

$$X_0 = u^\star v = (uu^\star + 1)v = u(u^\star v) + v$$

Moreover, the solution is unique when $\varepsilon \notin u$ (this is known as Arden's Lemma).

Atoms

In any Kleene algebra, define x to be an atom if $x \neq 0$ but

$$y \leq x \quad \text{implies} \quad y = 0 \text{ or } y = x$$

For example, 1 is an atom in the Boolean algebra.

In $\mathbb{B}^{n,n}$ the matrices with a single entry 1 are the atoms.

In the language semiring, atoms are exactly the singletons $\{w\}$.

Apart from legibility, this is one of the reasons why in language theory one often confuses w and $\{w\}$.

Matrix Kleene Algebras

Write $\mathcal{K}^{n \times n}$ for the set of all $n \times n$ matrices over some Kleene algebra $\mathcal{K}$.

We can add and multiply matrices in the usual way.

To define $M^\star$ we use the infinite sum $\sum_{\mathbb{N}} M^n$.

Claim: $\mathcal{K}^{n \times n}$ is again a Kleene algebra.

Claim: If we can compute in $\mathcal{K}$, then we can also compute in $\mathcal{K}^{n \times n}$.

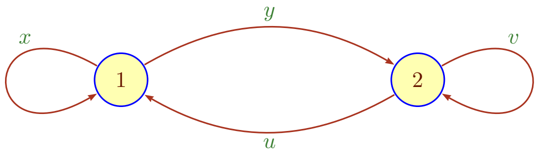
Clearly addition and multiplication carry over to $\mathcal{K}^{n \times n}$.
For Kleene star we have to work a bit harder.

Divide and Conquer

We can compute $M^\star$ by divide-and-conquer. For simplicity let $n = 2$.

$$\begin{pmatrix} x & y \\ u & v \end{pmatrix}^* = \begin{pmatrix} (x + yv^\star u)^\star & x^\star y(v + ux^\star y)^\star \\ v^\star u(x + yv^\star u)^\star & (v + ux^\star y)^\star \end{pmatrix}$$

This looks messy, but it all comes down to diagram-chasing in



FSMs and Matrices

To model a FSM $\mathcal{A}$ we can use square matrices that live in $\mathbb{B}^{Q \times Q}$.

For a letter $a \in \Sigma$, define a $Q \times Q$ matrix $f(a)$ by setting

$$f(a)(p,q) = \begin{cases} 1 & \text{if } \tau(p,a,q) \\ 0 & \text{otherwise.} \end{cases}$$

This map f naturally extends to a monoid homomorphism

$$f : \Sigma^\star \to \mathbb{B}^{Q \times Q}$$

where concatenation goes to matrix product. We have

$$x \in \mathcal{L}(\mathcal{A}) \iff I^{\rightarrow} \cdot f(x) \cdot F^{\downarrow} = 1$$

where $I^{\rightarrow}$ and $F^{\downarrow}$ are Boolean vectors indicating the initial and final states and the product is matrix-vector multiplication.

Regular Expressions

Consider the language Kleene algebra $\mathcal{L}(\Sigma)$.

Definition

A regular expression (regex) is a ground term of the Kleene algebra where we admit constants $\emptyset$, ε and a for each $a \in \Sigma$.

We write $\mathcal{L}(\alpha)$ for the language associated with regex α.

Here are some examples for alphabet $\Sigma = \{a, b\}$:

- All words containing bab: $(a+b)^* bab (a+b)^*$.
- All words containing 3 a's: $b^\star a b^\star a b^\star a b^\star$
- All words not containing aaa: $(1 + a + aa)(b + ba + baa)^\star$

A Language Matrix

Let $\mathcal{A}$ be a finite state machine on Q and switch to $\mathrm{Rec}_\Sigma^{Q\times Q}$.

This time, define the transition matrix T as follows:

$$T(p,q) = \sum\big(\, a \mid p \overset{a}{\to} q\big)$$

Proposition

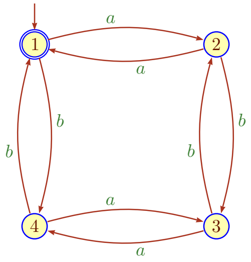
$\mathcal{L}(\mathcal{A}) = I \cdot T^\star \cdot F.$

Here I and F be 0/1 vectors indicating the initial and final states in $\mathcal{A}$, with components $\emptyset$ and ε.

Note: We can actually compute $T^\star$ in this setting: we use regex instead of handling the languages directly.

Example: Even/Even

Let EE be the language of even/even strings over $\Sigma = \{a, b\}$.
The natural DFA looks like



with transition matrix

$$T = \begin{pmatrix} 0 & a & b & 0 \\ a & 0 & 0 & b \\ b & 0 & 0 & a \\ 0 & b & a & 0 \end{pmatrix}$$

There are only two 2×2 submatrices and their stars are fairly simple.

$$A = \begin{pmatrix} 0 & a \\ a & 0 \end{pmatrix}^* = \begin{pmatrix} (aa)^\star & a(aa)^\star \\ a(aa)^\star & (aa)^\star \end{pmatrix}$$

$$B = \begin{pmatrix} b & 0 \\ 0 & b \end{pmatrix}^* = \begin{pmatrix} b^\star & 0 \\ 0 & b^\star \end{pmatrix}$$

The top-left 2×2 submatrix in $T^\star$ is then

$$(A + BA^\star B)^\star = \begin{pmatrix} b(aa)^\star b & a + ba(aa)^\star b \\ a + ba(aa)^\star b & b(aa)^\star b \end{pmatrix}^*$$

Its top-left entry is

$$\alpha = \Big(b(aa)^\star b + \big(a + ba(aa)^\star b\big)\big(b(aa)^\star b\big)^\star\big(a + ba(aa)^\star b\big) \Big)^*$$

WTF?

It is a character building exercise to show that the expression α really denotes the even/even language. Try.

Incidentally, there are much better expressions for EE:

$$\big(aa + bb + (ab + ba)(aa + bb)^{\star}(ab + ba)\big)^{*}$$

It is brutally hard to use algebraic transformations to show that the two expressions are equivalent.

Counting

For any language L, define its census or growth function by

$$\gamma_L(n) = |L \cap \Sigma^n|$$

We can express the growth function nicely in terms of a transition matrix over $\mathbb{N}$. Suppose we have a DFA $\mathcal{A}$ for L. Letting

$$T(p,q) = \text{ number of transitions } p \xrightarrow{a} q$$

we have

$$\gamma_L(n) = I \cdot T^n \cdot F$$

Here the initial/final vectors are over $\mathbb{N}$.

The computation requires only $\log n$ matrix multiplications.

And Star?

We can turn this into a problem about the generating function

$$g(z) = \sum \gamma_L(n)\, z^n \in \mathbb{N}[[z]]$$

With a bit more algebra one can show that

$$g(z) = I \cdot \Big(\mathsf{Id} - z\,T\Big)^{-1} \cdot F = \frac{p(z)}{|\mathsf{Id} - z\,T|}$$

where Id is the identity matrix and $p(z)$ some polynomial.

In essence, this requires solving one linear system of polynomial equations.

Example: Even/Even

$$T = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \in \mathbb{N}^{4,4}$$

$$\begin{aligned} g(z) &= \frac{1-2z^2}{1-4z^2} \\ &= 1 + 2z^2 + 8z^4 + 32z^6 + 128z^8 + 512z^{10} + \dots \end{aligned}$$

Kleene's Theorem

Here is key conclusion of all of this.

Theorem (Kleene 1956)

Every recognizable language over Σ can be constructed from $\emptyset$ and $\{a\}$, $a \in \Sigma$, using only the operations union, concatenation and Kleene star.

This is a bit of a surprise, since there are other closure properties, in particular intersection and complement. Hence, given a regex α for some language L there is another regex β for the complement of L. It is far from clear how to construct β.

On the other hand . . .

Some closure properties are very easy to establish given the theorem.

- union, concatenation, Kleene star
- reversal
- homomorphisms

Application

So we can denote any recognizable language by some suitable regex.

This is critical for applications such as `grep`, otherwise we would have to input some finite state machine. In practice, this would essentially be impossible.

Our algorithm to construct a regex from a FSM is quite clumsy, there are better methods—alas, none of them are efficient.

Fortunately, in the opposite direction regex to machine there are excellent methods.

Kleene's Proof

The original proof is a bit more hands-on and uses dynamic programming.

Suppose we have an NFA that accepts some regular language L.
For p, q in Q define

$$L_{p,q} = \mathcal{L}\big(\langle Q, \Sigma, \delta; \{p\}, \{q\}\rangle\big)$$

Then $L = \bigcup_{p \in I, q \in F} L_{p,q}$ and it suffices to construct regex for all $L_{p,q}$.

We would like to use some sort of induction argument, but a generic finite state machine has no inductive structure. It looks like a ball of yarn and cannot be disentangled.

Forcing Induction

We may safely assume $Q = [n]$.

Define a run from state p to state q to be k-bounded if all *intermediate* states are no greater than k. Note that p and q themselves are not required to be bounded by k.

Now consider the approximation languages:

$$L_{p,q,k} = \{\, x \in \Sigma^\star \mid \text{ there is a } k\text{-bounded run } p \xrightarrow{x} q \,\}.$$

In essence, we have erased all states $> k$, except for p and q.

Note that $L_{p,q,n} = L_{p,q}$.

Proof Sketch, Cont.

One can build expressions for $L_{p,q,k}$ by induction on k.

For $k = 0$ the expressions are easy:

$$L_{p,q,0} = \begin{cases} \sum_{\tau(p,a,q)} a & \text{if } q \neq p, \\ \varepsilon + \sum_{\tau(p,a,q)} a & \text{otherwise.} \end{cases}$$

So suppose $k > 0$. The key idea is to exploit the identity

$$L_{p,q,k} = L_{p,q,k-1} + L_{p,k,k-1} \cdot (L_{k,k,k-1})^* \cdot L_{k,q,k-1}$$

Done by induction hypothesis. $\square$

Feasibility

The critical line is

$$L_{p,q,k} = L_{p,q,k-1} + L_{p,k,k-1} \cdot (L_{k,k,k-1})^* \cdot L_{k,q,k-1}$$

In the actual algorithm, these are all regex.

But the expression on the right is about 4 times bigger than its components, so we get exponential blowup.

It might be tempting to try to simplify the expressions to keep them reasonably small. Alas, finding the shortest equivalent regular expression is PSPACE-hard. In reality, even just a few basic simplifications are quite hard to manage.

Dèjá Vu, All Over Again

Kleene's algorithm should look eminently familiar: logically, Floyd-Warshall's all-pairs shortest path algorithm from 1962 is essentially the same.

The underlying algebra is the min-plus semiring and we don't need to bother with loops. The recursion looks like this:

$$d_{p,q,k} = \min(d_{p,q,k-1}, d_{p,k,k-1} + d_{k,q,k-1})$$

And, of course, we are calculating with rational numbers here, not with formal expressions. There is no danger of expressions blowing up.

1 Kleene Algebras

2 Left Quotients

Missing Operations

From now on, we will focus on the language semiring

$$\mathcal{L}(\Sigma) = \langle \mathfrak{P}(\Sigma^\star); \cup, \cdot, {}^\star, \emptyset, \varepsilon \rangle$$

We have addition and multiplication, and a strange "iteration," but subtraction and division are missing.

Subtraction is hopeless since it requires an additive cancellation monoid:
$x + y = x + z$ implies $y = z$. This is hopelessly false in our setting:
$x + x = x = x + 0$.

But we can fake a sort of inverse of multiplication.

Quotients

Definition

Let $L \subseteq \Sigma^\star$ be a language and $x \in \Sigma^\star$. The left quotient of L by x is

$$x^{-1} L = \{ y \in \Sigma^\star \mid xy \in L \}.$$

So we are simply removing a prefix x from all words in the language that start with this prefix. If there is no such prefix we get an empty quotient.

We have $x(x^{-1} L) \subseteq L$, but we cannot expect $x(x^{-1} L) = L$ in general.

$x^{-1} (x L) = L$ works, though.

Algebra of Quotients

Lemma

Let $a \in \Sigma$, $x, y \in \Sigma^\star$ and $L, K \subseteq \Sigma^\star$. Then the following hold:

- $(xy)^{-1}L = y^{-1}x^{-1}L$
- $x^{-1}(L \odot K) = x^{-1}L \odot x^{-1}K$ *where* $\odot$ *is one of* $\cup$, $\cap$ *or* $-$
- $a^{-1}(LK) = (a^{-1}L)K + \chi_L\, a^{-1}K$
- $a^{-1}L^\star = (a^{-1}L)\, L^\star$

Here we have used the abbreviation χ_L to simplify notation:

$$\chi_L = \begin{cases} \varepsilon & \text{if } \varepsilon \in L, \\ \emptyset & \text{otherwise.} \end{cases}$$

So χ_L is either 0 or 1 in the language semiring and simulates an if-then-else.

Comments

Note that

$$(xy)^{-1}L = y^{-1}(x^{-1}L)$$

and **not** $x^{-1}y^{-1}L$. We will explain this later in our discussion of actions.

Quotients coexist peacefully with Boolean operations, we can just push the quotients inside.

But for concatenation and Kleene star things are a bit more involved; the lemma makes no claims about the general case where we divide by a word rather than a single letter.

Exercise

Prove the last lemma.

Exercise

Generalize the rules for concatenation and Kleene star to words.

Killer App

The ultimate reason we are interested in quotients is that they provide an elegant tool to construct the minimal automaton for a regular language, the smallest possible DFA. And the associated algorithms can be made very efficient.

Right now, let us focus on the algebra. We write

$$\mathcal{Q}(L) = \{\, x^{-1} L \mid x \in \Sigma^\star \,\}$$

for the set of all quotients of a language.

Question: How would we go about computing $\mathcal{Q}(L)$?

In general this will be difficult since languages are infinitary objects, but for languages given by regex we can use the lemma from above. There is a little glitch, though.

More Closure

Abstractly, this is yet another closure problem: we need to compute the least set $\mathcal{X} \subseteq \mathcal{L}(\Sigma)$ such that

- $L \in \mathcal{X}$ and
- $\mathcal{X}$ is closed under a^{-1} for all $a \in \Sigma$.

If you are fond of fixed points, consider the monotonic operation $F : \mathfrak{P}(\mathcal{L}(\Sigma)) \to \mathfrak{P}(\mathcal{L}(\Sigma))$ mapping families of languages to families of languages

$$F(\mathcal{X}) = \mathcal{X} \cup \{\, a^{-1}X \mid X \in \mathcal{X}, a \in \Sigma \,\}$$

So we are looking for the least fixed point of $\{L\}$ under F.

Quotients Example 1

Using the lemma, we can compute the quotients of a^*b.

$$\begin{aligned} a^{-1}\, a^*b &= a^*b \\ b^{-1}\, a^*b &= \varepsilon \\ a^{-1}\, \varepsilon &= \emptyset \\ b^{-1}\, \varepsilon &= \emptyset \\ a^{-1}\, \emptyset &= \emptyset \\ b^{-1}\, \emptyset &= \emptyset \end{aligned}$$

Thus $\mathcal{Q}(a^*b)$ consists of: a^*b, ε and $\emptyset$.

Quotients Example 1, Cont.

$$\begin{array}{cc}
a^{-1}\, a^*b = a^*b & a^*b \xrightarrow{a} a^*b \\
b^{-1}\, a^*b = \varepsilon & a^*b \xrightarrow{b} \varepsilon \\
a^{-1}\, \varepsilon = \emptyset & \varepsilon \xrightarrow{a} \emptyset \\
b^{-1}\, \varepsilon = \emptyset & \varepsilon \xrightarrow{b} \emptyset \\
a^{-1}\, \emptyset = \emptyset & \emptyset \xrightarrow{a} \emptyset \\
b^{-1}\, \emptyset = \emptyset & \emptyset \xrightarrow{b} \emptyset
\end{array}$$

The equations really determine the transitions in a finite state machine.

Quotients Example 2

For finite languages the quotient computation is particularly simple and easy to implement by brute force string operations.

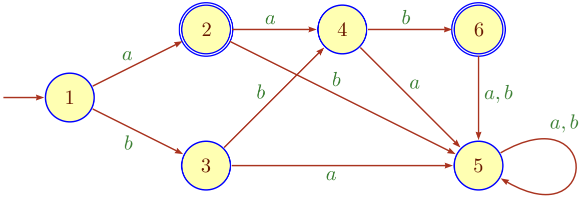
Say, let $L_1 = L = \{a, aab, bbb\}$.

$a^{-1}L_1$	$\{\varepsilon, ab\}$	L_2
$b^{-1}L_1$	$\{bb\}$	L_3
$a^{-1}L_2$	$\{b\}$	L_4
$b^{-1}L_2$	$\emptyset$	L_5
$a^{-1}L_3$	L_5	
$b^{-1}L_3$	L_4	
$a^{-1}L_4$	L_5	
$b^{-1}L_4$	ε	L_6
$s^{-1}L_5$	L_5	
$s^{-1}L_6$	L_5	

So $\mathcal{Q}(L)$ has size 6.

Quotients Example 2.5

Moreover, there happens to be a “natural” DFA for L that has six states.



Could this be coincidence? Nah, more later ...

Quotients Example 3

A slightly larger example, $L = L_1 = a^*b^* + bab$.

$a^{-1}L_1$	a^*b^*	L_2
$b^{-1}L_1$	$b^* + ab$	L_3
$a^{-1}L_2$	L_2	
$b^{-1}L_2$	b^*	L_4
$a^{-1}L_3$	b	L_5
$b^{-1}L_3$	L_4	
$a^{-1}L_4$	$\emptyset$	L_6
$b^{-1}L_4$	L_4	
$a^{-1}L_5$	L_6	
$b^{-1}L_5$	ε	L_7
$a^{-1}L_{6/7}$	L_6	
$b^{-1}L_{6/7}$	L_6	

Quotients Example 4

An even larger example, $L = L_1 = a^*ba^* + b^*ab^*$.

$a^{-1}L_1$	$a^*ba^* + b^*$	L_2	$b^{-1}L_5$	b^*	L_8
$b^{-1}L_1$	$b^*ab^* + a^*$	L_3	$a^{-1}L_6$	b^*	
$a^{-1}L_2$	a^*ba^*	L_4	$b^{-1}L_6$	b^*ab^*	
$b^{-1}L_2$	$a^* + b^*$	L_5	$a^{-1}L_7$	b^*	
$a^{-1}L_3$	$a^* + b^*$		$b^{-1}L_7$	$\emptyset$	L_9
$b^{-1}L_3$	b^*ab^*	L_6	$a^{-1}L_8$	$\emptyset$	
$a^{-1}L_4$	a^*ba^*		$b^{-1}L_8$	b^*	
$b^{-1}L_4$	a^*	L_7	$a^{-1}L_9$	$\emptyset$	
$a^{-1}L_5$	a^*		$b^{-1}L_9$	$\emptyset$	

Quotients Example 5

Here is a very different scenario, the counting language:

$$L = \{\, a^i b^i \mid i \geq 0 \,\} = \{\varepsilon, ab, aabb, aaabbb, \ldots\}$$

This time there are infinitely many quotients.

$$\begin{aligned}
(a^k)^{-1}L &= \{\, a^i b^{i+k} \mid i \geq 0 \,\} & & \\
(a^k b^l)^{-1}L &= \{b^{k-l}\} & & 1 \leq l \leq k \\
(a^k b^l)^{-1}L &= \emptyset & & l > k
\end{aligned}$$

This is no coincidence: the language L is context free but fails to be regular.

An Algorithm?

The calculations from above suggest that for we can actually compute the quotients in a purely algebraic manner, starting from an regex.

Is this really true?

Yes and no. In order for this to work, we need to be able to test whether two regex are equivalent, whether they denote the same language.

This turns out to be decidable, but it is quite difficult: the problem is PSPACE-complete in general.

Fortunately, there is a much better algorithms based on deterministic finite state machines.

Quotients and Machines

Suppose $\mathcal{A}$ is a DFA for a regular language L. Define the behavior of a state p to be

$$[\![p]\!] = \mathcal{L}(\mathcal{A}(p, F))$$

In other words, move the initial state to p but leave the automaton unchanged otherwise. In particular $[\![q_0]\!] = L$.

Proposition

For any word w, $[\![\delta(q_0, w)]\!] = w^{-1} L$.

It follows immediately that every regular language has only finitely many quotients. In fact, the size of any DFA for the language is a bound on this number. The next result establishes the opposite direction: finitely many quotients implies regular.

The Quotient Machine

Suppose L is some language with a finite set of quotients. We can exploit $\mathcal{Q} = \mathcal{Q}(L)$ as the state set of a DFA for L.

$$\mathfrak{Q}_L = \langle \mathcal{Q}, \Sigma, \delta; q_0, F \rangle$$

where

$$\begin{aligned} \delta(K, a) &= a^{-1}\, K \\ q_0 &= L \\ F &= \{\, K \in Q \mid \varepsilon \in K \,\} \end{aligned}$$

Induction shows that $\delta(q_0, w) = w^{-1}L$, so this works as advertised. Since every quotient occurs only once in $\mathfrak{Q}_L$ there cannot be a smaller DFA for L.