CDM

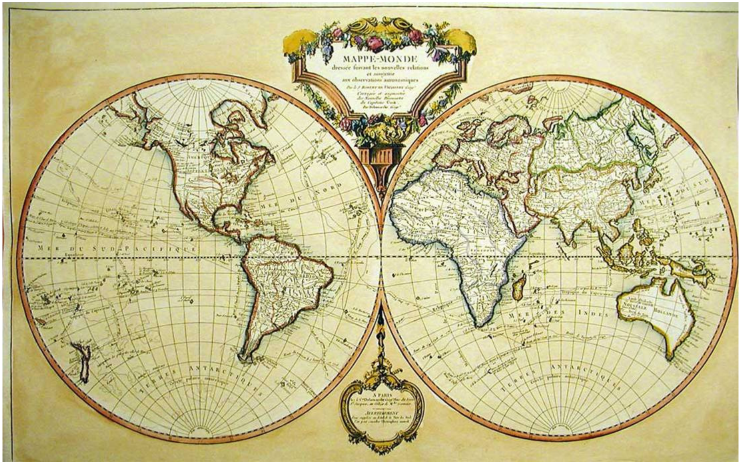
Feedback Shift Registers

Klaus Sutner

Carnegie Mellon University

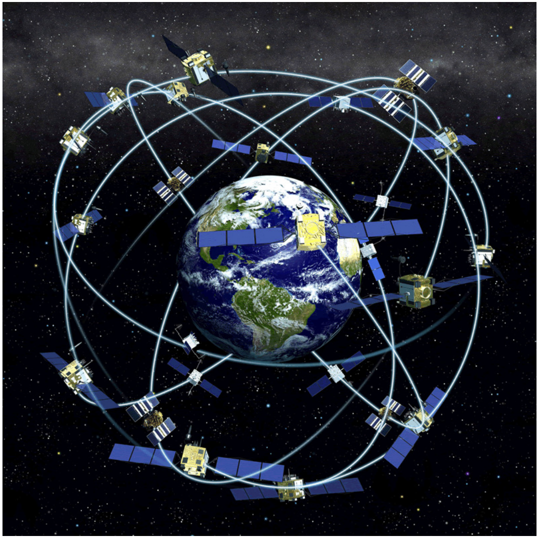
Spring 2024

Getting Around



Athena told Odysseus to “keep the Great Bear on his left”.
Note that it took him 20 years to get back home.

Global Positioning System



Fancy/Cheap System

24 satellites, about 20,000 km above ground, moving at some 4.5 km/sec.

Each has 4 highly accurate atomic clocks, everything tightly controlled by one master and four additional control stations.

The Problem:
At the other end, a GPS receiver must be cheap, small, reliable, zero maintenance.

This is a bit different from standard resource constraints on computation (time, space) but equally interesting.

Data centers now consume twice as much energy as New York City, with huge growth rates.

Einstein's Legacy

Einstein's general theory of relativity is now over 100 years old.

A Question: Who wins?

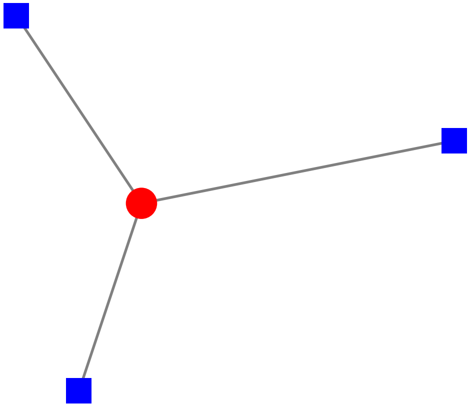
- By general relativity, satellite clocks move faster since they are high up in Earth's gravity field (about 45,850 nanosecs per day). BTW, nowadays one can measure the effect of lifting a super-accurate clock by 2cm.
- By special relativity, satellite clocks undergo relativistic slow-down since they are moving relatively fast (about 7,214 nanosecs per day).

Final result: there is a speed-up of about 38 microsecs per day.

http://physicscentral.com/explore/writers/will.cfm

And the engineers did not believe it.

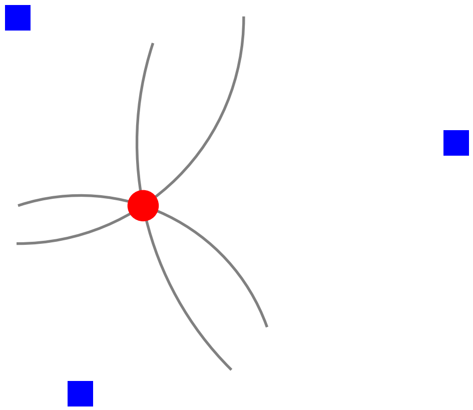
Triangulation



If we can measure the direction of the radio signals, we can triangulate (at least 2, more for better precision).

No way! Our cheapo device cannot handle this.

Trilateration



But we can use trilateration: measure distance by measuring the delay of a signal. Given timing and satellite location data, this suffices to determine location of the receiver.

Measuring Delay

How do we measure a delay of some 0.007 seconds? Cheaply?

An utterly non-workable solution would be to have the satellite send very short radio bursts.

Realistically, the only thing our cheapo receiver can do is to receive a nice, steady stream of bits sent from the satellite. Hence we need to somehow encode the timing information in a bit stream.

Aside: We need to synchronize the cheap local clock, this is done by a little protocol involving at least 3 satellites.

Measuring Delay, on the Cheap

Main Idea:
Send a stream of bits such that k consecutive bits suffice to determine the position in the stream.

0000000010000001100000101000001110000100100001011000011010000111
1000100010011000101010001011100011001000110110001110100011111001
0010100100111001010110010110100101111001100110101001101110011101
1001111010011111101010101110101101101011111011011110111011111111

Whole stream is periodic, the period may be quite long.

Old Specs

Standard Positioning System

C/A signal: period 1023, sent once every millisecond.
Accuracy: 100 m, 340 nanoseconds.

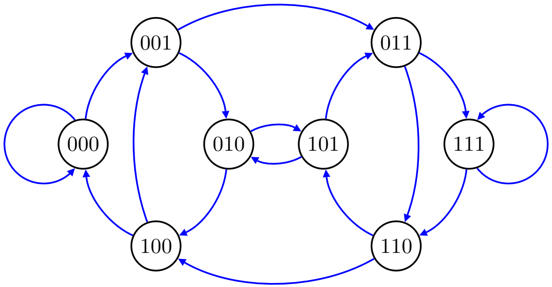
Precision Positioning System

P and Y signals: military use, period of 267 days.
Accuracy: 15 m, 200 nanoseconds.

Current systems are much better than this, under reasonable circumstances accuracy is better than 1 m. Alas, the current design documents are basically illegible.

The Perils of Education

Every CS major will have an immediate knee-jerk response: use a de Bruijn sequence, essentially just a Hamiltonian cycle in a de Bruijn graph ...



This is the de Bruijn graph $\mathbb{B}_3$ of order 3. De Bruijn sequence 00010111.

We want something of much higher order, say, $k = 50$, producing a de Bruijn sequence of length 2^{50}.

What Could Go Wrong?

Finding a Hamiltonian cycle in general is $\mathbb{NP}$-hard, but in this case we can get away with murder.

An Eulerian cycle in a digraph can be constructed in linear time, and de Bruijn graphs are all trivially Eulerian.

Recall: $\mathbb{B}_k$ is the line graph of $\mathbb{B}_{k-1}$. Hence an Eulerian cycle in $\mathbb{B}_{k-1}$ directly translates into a Hamiltonian cycle in $\mathbb{B}_k$.

Done!

Alas . . .

The Evans/Minieka algorithm for Eulerian cycles is linear time, but it is also linear space.

For order $k = 50$ we need $\Omega(2^{50})$ bits of storage. Totally out of the question for our little receiver.

- Are there other ways to generate a Hamiltonian cycle in $\mathbb{B}_k$?
- Or at least a very long cycle, something of length nearly 2^k?
- We would like to generate the ith bit in time and space $O(1)$.

Succinct Representations

To hammer this home: the problem we are trying to solve is trivial in a sense: there is a simple, linear time and linear space algorithm.

Unfortunately, it's linear in 2^{50}, so any explicit data structure is out.

But de Bruijn graphs are highly regular and have a nice succinct representation (virtual graphs). It seems plausible that there might be an algorithm that exploits this succinct representation to find Hamiltonian cycles, or at least very long cycles.

With constant memory.

This is the end of the age of innocence for graph algorithms, huge graphs are much harder to deal with (the web, model checking).

Iteration to the Rescue

Perhaps we could find some easily computable function of the form

$$F : \mathbf{2}^k \to \mathbf{2}^k$$

that can be iterated to trace a (hopefully very long) cycle in the de Bruijn graph:

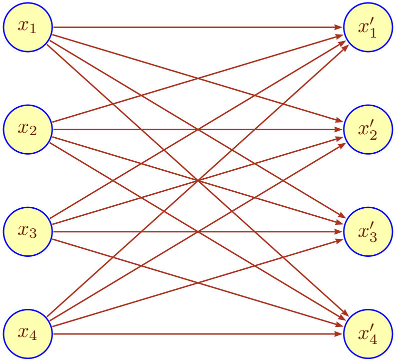
- We want $x \to F(x)$ to be an edge.
- Also, F should be injective, so all orbits are periodic.
- Pick an initial vertex x_0 and iterate away.

In principle, we could simply have F follow a Hamiltonian cycle. Computationally, this is completely useless.

A Circuit

Think of this a circuit design problem: we have k one-bit registers and want to update their contents by some simple circuitry.

In the most general scenario, we could make every new bit depend on every old bit:



Choice Functions

But note that we need $\boldsymbol{x} \mapsto F(\boldsymbol{x})$ to be an edge in $\mathbb{B}_k$. That means for $\boldsymbol{x}' = F(\boldsymbol{x})$:

$$\begin{aligned} x'_i &= x_{i+1} \qquad i < k \\ x'_k &= f(\boldsymbol{x}) \end{aligned}$$

where $f : \mathbf{2}^k \to \mathbf{2}$. We are just computing f plus a shift operation.

Now the only question is: what is the right choice for f?

We still have $2^{2^k} = \infty$ possibilities: recall that $k = 50$. There are no symmetries to exploit, either. We need some clever theory to help find a good function f.

Shifty Choice

$$\begin{tikzcd}
x_1 \arrow[dddr] & x'_1 \\
x_2 \arrow[ur] \arrow[ddr] & x'_2 \\
x_3 \arrow[ur] \arrow[dr] & x'_3 \\
x_4 \arrow[ur] \arrow[r] & x'_4
\end{tikzcd}$$

More Choice

Without the memory constraint we could try something like this. Find a clever, easily computable function

$$C : \mathbf{2}^k \to \mathbf{2}$$

Use C as a choice function that tells us which way to go when we first encounter a node: when we hit node u, go into direction $C(u)$.

Of course, we also must remember that we have already seen u, and then take direction $1 - C(u)$ when we return.

There are some interesting results concerning choice functions in the literature, but for us this won't work: we cannot afford to remember already touched vertices.

Hence we can only preserve a little bit of state, say, k bits worth of state. We need a "device" that can update the state and an "output" operation that reads off the next bit.

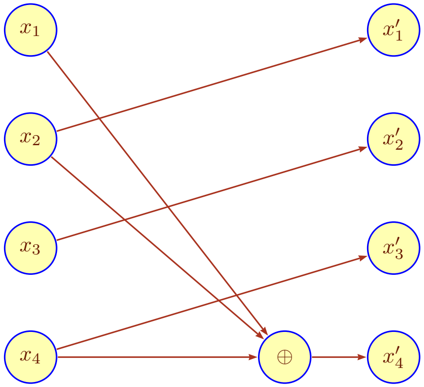
The output operation will be simple, say, $g(b_1, b_2, \ldots, b_k) = b_1$ or $g(b_1, b_2, \ldots, b_k) = b_1 \wedge b_5$ or some such.

Xor Functions

We can take an experimental plunge and restrict our attention to

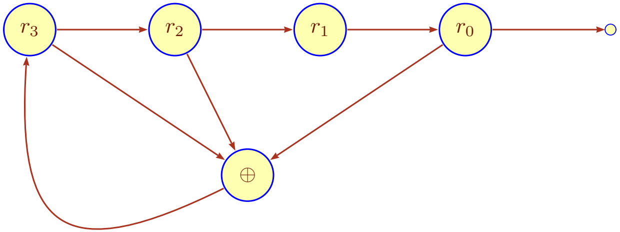
$$f(\boldsymbol{x}) = x_{p_1} + x_{p_2} + \ldots + x_{p_r} \bmod 2$$

The positions p_i are the so-called taps.



Feedback Shift-Registers

The last step is to redraw the picture slightly, and to renumber the registers, here $r_i = x_{k-i}$ for $0 \leq i < k$. k is the span of the FSR. [†]

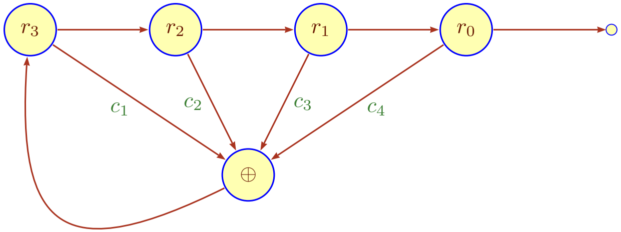


We can think of generating one output bit at register r_0 during each clock cycle.

[†]This renumbering may seem like a plain nuisance, but it actually makes the analysis easier later on.

Rephrase

A better way to think about FSRs: assume there is a tap at every register, but they have weights $c_i \in \mathbf{2}$ which determine whether the tap is on or off.



So the feedback value placed into the last register is the convolution

$$c_1 r_{k-1} + c_2 r_{k-2} + \ldots + c_k r_0$$

Note that we may safely assume that there is a tap at r_0: otherwise we are just inflating the span k and shifting the output bit a number of times before releasing it into the light.

Generating Bit-Sequences

To generate a bit-sequence, we choose k initial values for the registers, say $\boldsymbol{a} = (a_{k-1}, a_{k-2}, \ldots, a_1, a_0)$. We then use the kth order linear recurrence

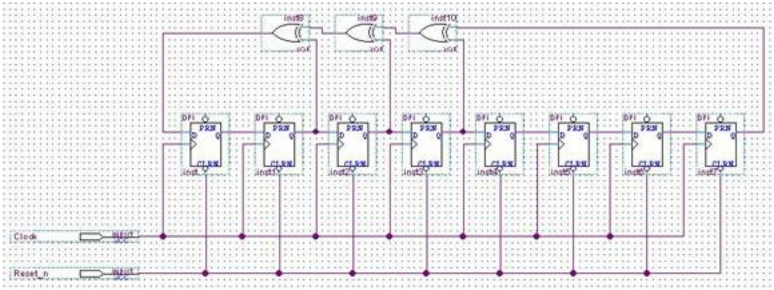
$$a_n = \sum_{i=1}^{n} c_i a_{n-i} = c_1 a_{n-1} + c_2 a_{n-2} + \ldots + c_k a_{n-k}$$

to generate a sequence $(\boldsymbol{a}_i) \in (\mathbf{2}^k)^\omega$.

Clearly, this kind of gadget is very easy to realize in circuitry. Requires only k one-bit registers, some xor circuits, and a clock (the whole shift-register must be synchronized).

Truth in advertising: xor gates usually have two inputs, so we may need to build a little tree to get the desired feedback bit.

Engineering Angle



Engineers love these devices: they are easy to implement and lightning fast.

Exercise

The diagram above is stolen from the interwebs, is it the right one?

Dèjá Vu All Over Again

We can think of xor as addition over the two-element field $\mathbb{F}_2$.

Upcoming attractions: feedback-shift registers are very closely related to multiplication in extension fields $\mathbb{F}_{2^k}$.

Where are We?

By choosing taps we obtain a local function $f : \mathbf{2}^k \to \mathbf{2}$, which gives rise to the global function $F : \mathbf{2}^k \to \mathbf{2}^k$,

$$F(\boldsymbol{x}) = \big(x_2, \ldots, x_k, f(\boldsymbol{x})\big)$$

We need to make sure the following holds:

- The global map is injective, so we get periodic orbits.
- There are good initial conditions $\boldsymbol{a}$ that produce long orbits.

Of course, this all could go completely wrong–but as we will see, things work out nicely.

Wishful Thinking

The good news is that a FSR produces the next bit computed in constant time and space (essentially the span).

Not so good news: $\mathbf{0} \in \mathbf{2}^k$ is always a fixed point of F, so it's useless for long orbits.

But, we can hope for a one long orbit of size $2^k - 1$. In this case, the starting point $\boldsymbol{a} \neq \mathbf{0}$ would not matter.

In general we will see that unit vector $\boldsymbol{e}_1$ is always a good place to start.

Time for some experimentation.

Reversibility

Recall that we always assume a tap at register r_0 so that $c_k = 1$.

Proposition

The global map F based on linear feedback function f is injective.

Proof.

Recall $a_n = c_1 a_{n-1} + c_2 a_{n-2} + \ldots + c_k a_{n-k}$. Hence we can run the recurrence backwards:

$$a_{n-k} = a_n - (c_1 a_{n-1} + c_2 a_{n-2} + \ldots + c_{k-1} a_{n-k+1}).$$

$\square$

Note that we could have written $+$ instead of $-$ since we have characteristic 2. The minus is simply keeping track of the underlying arithmetic.

Additivity

Proposition

The global map F is linear (aka additive):

$$F(\boldsymbol{x}+\boldsymbol{y}) = F(\boldsymbol{x}) + F(\boldsymbol{y})$$

where all addition is mod 2.

Here we think of $\mathbf{2}^k$ as an $\mathbb{F}_2$-vector space.

It follows that we can determine the value of $F(\boldsymbol{x})$ by forming the appropriate linear combination of images of the canonical basis vectors $\boldsymbol{e}_i \in \mathbf{2}^k$ under F:

$$F(\boldsymbol{e}_i) = F(0,\ldots,0,1,0,\ldots,0)$$

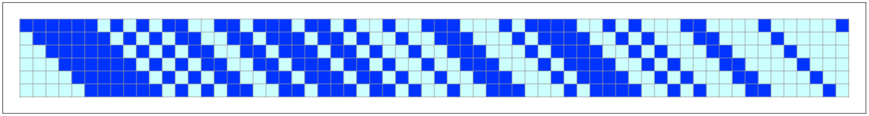
since $F(\sum c_i \boldsymbol{e}_i) = \sum c_i F(\boldsymbol{e}_i)$.

Exercise

Prove that the global map is indeed additive.

An Orbit

Here is the orbit of $\boldsymbol{e}_1$ under the FSR with taps $c_1 = c_k = 1$, and span $k = 6$. We will abbreviate this situaion as FSR $(1, k)$.



Each column corresponds to the 6 registers, the exit register r_0 is at the bottom; time flows left to right.

This orbit has optimal length 63. Of course, this is too good to be true in general ...

Small Spans

k	p	$p/2^k$	k	p	$p/2^k$
3	7	0.875	12	3255	0.794678
4	15	0.9375	13	7905	0.964966
5	21	0.65625	14	11811	0.720886
6	63	0.984375	15	32767	0.999969
7	127	0.992188	16	255	0.003891
8	63	0.246094	17	273	0.002083
9	73	0.142578	18	253921	0.968632
10	889	0.868164	19	413385	0.788469
11	1533	0.748535	20	761763	0.726474

Taps $(1, k)$ work on occasion, but also fail badly. Brief table-staring immediately suggests some conjectures:

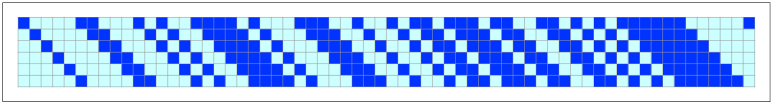
Conjecture

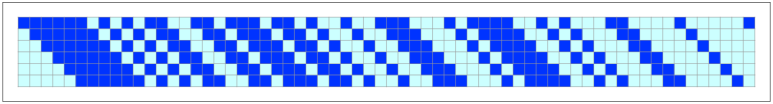
The period is $2^k - 1$ for span $k = 2^\ell - 1$.

A Surprise

If we compute the same table for taps $(k-1, k)$, we get exactly the same orbit lengths.

Just to be clear: the orbits themselves are different! Here is span $k = 6$; $(5, 6)$ on top, $(1, 6)$ below.

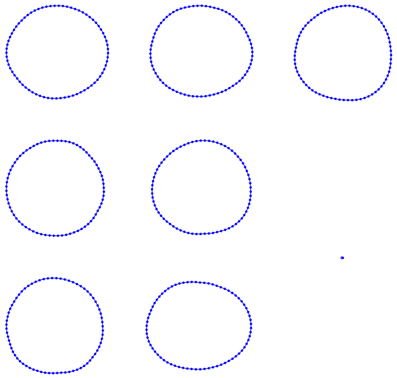




Why?

Phasespace

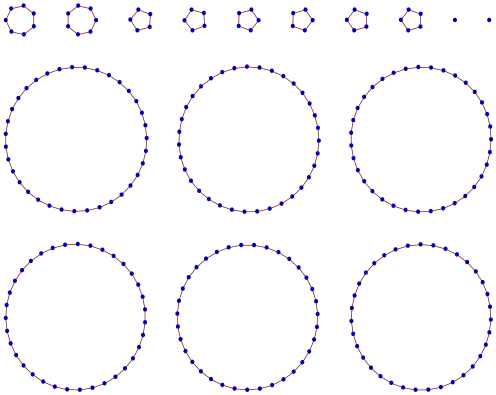
Here is why things go wrong for $k = 9$:



There is one fixed point, plus 7 cycles of length 73.

And Another

$k = 8$ with taps 01101011.



Again, too many cycles to get a long orbit.

Impulse-Response Sequences

Additivity has an important side-effect when it comes to periods: we can obtain maximal period by selecting as the starting configuration the unit vector $\boldsymbol{e}_1$.

Definition

An impulse-response sequence for F is an orbit obtained from a basis vector $\boldsymbol{e}_1$.

Lemma (Period Lemma)

The period of any configuration $\boldsymbol{a}$ divides the period of $\boldsymbol{e}_1$.

We need a little more machinery (which is also independently useful) for the proof of the lemma.

The Companion Matrix

Definition

Let $a_n = c_1 a_{n-1} + c_2 a_{n-2} + \ldots + c_k a_{n-k}$ be linear recurrence of order k over some ring R. The (Frobenius) companion matrix C of the recurrence is a $k \times k$ matrix over R defined by

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

For example, for $k = 5$ the companion matrix looks like so:

$$C = \begin{pmatrix} c_1 & c_2 & c_3 & c_4 & c_5 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Note that there are several versions of the companion matrix in the literature.

Matrix Multiplication and Iteration

By multiplying the companion matrix with a vector representing the current bit-pattern in the registers we can compute the next bit-pattern.

Proposition

$C^t \boldsymbol{a}$ is the content of the registers at time t where $\boldsymbol{a} = (a_{k-1}, \ldots, a_0)$ is the initial configuration.

Computationally this produces the following speed-up: using a standard matrix multiplication algorithm and fast exponentiation it would take us $O(k^3 \lg t)$ steps to compute the state of the system at time t.

This is an example of predictability or computational compressibility: it does not take $\Omega(t)$ steps to find the configuration at time t.

Proof of Period Lemma

Let q be the period of the impulse-response sequence generated by $\boldsymbol{e}_1$, C the companion matrix.

Then $C^q \boldsymbol{e}_1 = \boldsymbol{e}_1$ and hence for any $i \geq 0$ we have

$$C^{q+i} \boldsymbol{e}_1 = C^i \boldsymbol{e}_1$$

and therefore

$$(C^q - I) \cdot (C^i \boldsymbol{e}_1) = 0.$$

Since $c_k \neq 0$, the vectors $C^i \boldsymbol{e}_1$ must span the whole space; hence we must have $C^q = I$.

It follows that q is a period of any configuration, and the least period must divide q.

$\square$

Predicting Periods

Key Question: Where should the taps go?

Note that we have a new practical problem if we really succeed: If the periods are very long, we cannot verify this fact by running a simulation with real hardware.

Hence, we need a solid proof; in a way, a computational shortcut that establishes claims about period lengths without brute-force simulation.

Exercise

For a reasonably small value of k, say, $k = 10$, determine the complete cycle structure of $\mathbf{2}^k$ for all possible feedback shift-registers of order k.

Words of Wisdom

There is nothing more practical than a good theory.
Kurt Lewin

Generating Functions

Consider the sequence (a_n) produced from the initial configuration $\boldsymbol{a} = (a_0, a_1, \ldots, a_{k-1})$ via the recurrence

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \ldots + c_k a_{n-k}$$

Is there any chance that generating functions might be useful?

$$G(x) = \sum a_n x^n ?$$

Example: Lagged Fibonacci

Consider initial conditions $\boldsymbol{a} = (1,0,0,0)$ and taps $(1,4)$, so we are dealing with a Fibonacci-type recurrence $a_n = a_{n-1} + a_{n-4}$ for $n \geq 4$.

Then for $G(x) = \sum a_n x^n$

$$\begin{aligned} G(x) &= 1 + \sum_{n \geq 4} a_n x^n \\ &= 1 + x \sum_{n \geq 3} a_n x^n + x^4 \sum a_n x^n \\ &= 1 - x + x \sum a_n x^n + x^4 \sum a_n x^n \\ &= 1 - x + (x + x^4)\, G(x) \end{aligned}$$

so that

$$G(x) = \frac{1-x}{1-x-x^4}$$

Comments

We have used minus signs to emphasize the arithmetic, in characteristic 2 we could just as well have written plus (but then you have to make changes for the characteristic $p > 2$ case).

From the example, one can see that the denominator of the rational generating function will be

$$1 - c_1 x - c_2 x^2 - \ldots - c_k x^k.$$

The numerator is slightly more complicated to write down.

The General Case

Theorem

The generating function for our feedback shift-registers is

$$G(x) = \frac{-\sum_{i=0}^{k-1}\left(\sum_{j=0}^{i} c_j a_{i-j}\right) x^i}{1 - \sum_i c_i x^i}$$

where $c_0 = -1$.

Sketch of proof.

$$\begin{aligned} G(x) &= \sum_n \left(\sum_i c_i a_{n-i}\right) x^n \\ &= \sum_i c_i x^i \sum_n a_{n-i} x^{n-i} \\ &= \sum_i c_i x^i \left(a_{-i} x^{-i} + \ldots + a_{-1} x^1 + \sum_n a_n x^n\right) \end{aligned}$$

$\square$

Feedback Polynomial

Definition

The denominator of this rational function is called the feedback polynomial or the connection polynomial of the sequence (a_n).

Note that one can solve the following equation for the numerator:

$$-\sum_{i=0}^{k-1}\left(\sum_{j=0}^{i} c_j a_{i-j}\right) x^i = 1$$

for a_i. The solution yields the initial conditions for a sequence whose generating function simplifies to the reciprocal of the feedback polynomial:

$$G(x) = \frac{1}{1 - \sum_i c_i x^i}$$

In general, however, the numerator is some polynomial of degree less than k.

Example: $(1,5)$

For taps $(1,5)$ and the impulse-response sequence we get

$$G(x) = \frac{1-x}{1-x-x^5}$$

By Taylor expansion, we can compute a few terms of this series:

$$1+x^5+x^6+x^7+x^8+x^9+2\,x^{10}+3\,x^{11}+4\,x^{12}+5\,x^{13}+6\,x^{14}+8\,x^{15}+11\,x^{16}+\ldots$$

As written, this is just wrong: we need coefficients in $\mathbb{F}_2$, not integers.

Wrong Field

We are computing the Taylor series in characteristic 0 (actually, we don't have a choice), but we can simply drop down to characteristic 2 in the end:

$$1 + x^5 + x^6 + x^7 + x^8 + x^9 + x^{11} + x^{13} + x^{16} + x^{17} + \dots$$

and this is indeed the right answer.

Computing matching Taylor expansions is most reassuring, but we really need to get a grip on the period of these sequences.

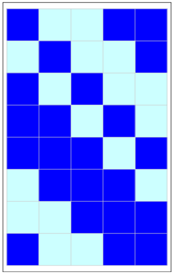
As a warmup, how do the initial conditions come into play?

Changing Initial Conditions

Suppose we change the initial conditions to $(1, 1, 0, 0, 1)$. Then

$$\begin{aligned} G(x) &= \frac{1 - x^2 + x^4}{1 - x - x^5} \\ &= 1 + x + x^4 + x^6 + x^7 + x^8 + x^{11} + x^{13} + x^{14} + \dots \end{aligned}$$

and the orbit looks like so:



Buring Question: How does the sequence relate to the beautiful picture?

Period and Polynomial Divisors

Theorem

Let (a_n) be a sequence with generating function $G(x) = 1/g(x)$.
Then the period of (a_n) is the least $p > 0$ such that $g(x)$ divides $1 - x^p$.

Proof.

If the period is p then

$$\begin{aligned} 1/g(x) &= \left(a_0 + a_1 x + \ldots + a_{p-1}x^{p-1}\right)\left(1 + x^p + x^{2p} + \ldots\right) \\ &= \left(a_0 + a_1 x + \ldots + a_{p-1}x^{p-1}\right) / \left(1 - x^p\right) \end{aligned}$$

So $1 - x^p = g(x) \cdot \left(a_0 + a_1 x + \ldots + a_{p-1}x^{p-1}\right)$ and $g(x)$ divides $1 - x^p$, as required.

And Back

On the other hand, if $1 - x^p = g(x)(b_0 + b_1 x + \ldots + b_{p-1} x^{p-1})$ then

$$\begin{aligned} 1/g(x) &= \left(b_0 + b_1 x + \ldots + b_{p-1} x^{p-1}\right) \Big/ (1 - x^p) \\ &= \left(b_0 + b_1 x + \ldots + b_{p-1} x^{p-1}\right) \left(1 + x^p + x^{2p} + \ldots\right) \end{aligned}$$

Comparing coefficients we get $a_n = b_{n \bmod p}$, so the period of the sequence must divide p.

Since p is minimal, they must agree.

□

Numerators

What happens if the generating function is a general rational function

$$G(x) = h(x)/g(x)?$$

We may assume that h and g are coprime, otherwise we can simply factor out.

Accordingly, the sequence will be eventually periodic in the general case, but strictly periodic as long as the degree of h is less than the degree of g – exactly the situation that we are in.

The denominator controls period length, the numerator expresses the initial conditions given by the first k bits in the registers.

Long Cycles and Primality

Definition

Let $g \in \mathbb{F}[x]$.
The exponent of g is the least $p > 0$ such that $g(x)$ divides $1 - x^p$.

It is not clear how to compute exponents efficiently, but one can show the following.

Theorem

If a shift-register sequence of span k has maximum length $2^k - 1$, then the corresponding polynomial $g(x)$ must be irreducible.

Unfortunately, this condition is not sufficient: we need a primitive polynomial, not just an irreducible one.

This is analogous to having x be a generator of $\mathbb{F}^\times$ in our finite field construction.

Exploiting Mersenne

We can throw more algebra at the problem. Recall that any irreducible polynomial $g(x)$ of degree k divides $x^{2^k-1} - 1$. Hence the exponent of $g(x)$ must divide $2^k - 1$.

So if $2^k - 1$ happens to be a Mersenne prime the exponent must be equal to $2^k - 1$.

It is an open problem whether infinitely many Mersenne primes exist, but for the following values of k we do get a Mersenne prime $2^k - 1$:

$$2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107, 127$$

These are all appropriate k's less than 256.

BTW, the largest known prime is a Mersenne prime, currently (October 2024):

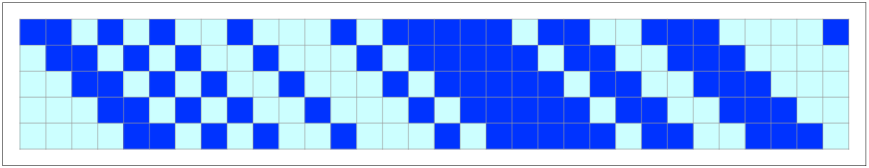
$$2^{136,279,841} - 1$$

Example Mersenne Prime

31 is a Mersenne prime, $k = 5$.

There are exactly 6 irreducible polynomials of degree 5 in $\mathbb{F}_2[x]$, the orbit for the last is plotted below, and duly has length 31.

$$1 + x^2 + x^5, 1 + x^3 + x^5, 1 + x + x^2 + x^3 + x^5,$$
$$1 + x + x^2 + x^4 + x^5, 1 + x + x^3 + x^4 + x^5, 1 + x^2 + x^3 + x^4 + x^5$$



Primitive Field Elements

Here is a closer look. We know that the multiplicative subgroup of every finite field is cyclic.

More precisely, if we have $K = \mathbb{F}[x]/(f)$ where f is irreducible and primitive, then we can choose $\alpha = x \bmod f$ as a generator.

Key Insight:

A single step in an FSR corresponds to division by α.

But for primitive f the order of α and thus of α^{-1} is $2^k - 1$. Hence if we choose the taps of our FSR according to the coefficients of f we obtain maximal period.

I don't know how to prove this without introducing more machinery (dual bases), so we'll skip. See the comment at the end of this section, though.

The Field Angle

As an indication why finite fields are critical to any real understanding of FSRs, here is one more result.

Wild Guess: Perhaps shift register sequences can be explained completely in terms of finite fields, something along the lines of

$$a_n = \text{ blah di blah } \alpha^n \text{ di blah blah}$$

The "blah di blah" should be expressed solely in terms of field operations, preferably in a nice formula.

The good news: it really works.

The bad news: it involves one more idea.

Traces

Suppose $f(x) \in \mathbb{F}_2[x]$ is irreducible of degree k. and α be a root of f over the splitting field $\mathbb{F}_{2^k}$.

Definition

The trace function of $\mathbb{F}_{2^k}$ over $\mathbb{F}_2$ is defined by

$$\mathsf{Tr} : \mathbb{F}_{2^k} \to \mathbb{F}_2 \qquad\qquad \mathsf{Tr}(z) = \sum_{i<k} z^{2^i}$$

This is less random than it may seem, we are summing over the orbit of z under the Frobenius homomorphism $z \mapsto z^2$.

Proposition

Tr *is linear and its range is indeed* $\mathbb{F}_2$*.*

Trace Representation

Theorem

Let β be any element $\mathbb{F}_{2^k}$ and initialize the registers of a FSR with

$$\mathsf{Tr}(\beta), \mathsf{Tr}(\alpha^{-1}\beta), \ldots, \mathsf{Tr}(\alpha^{2-k}\beta), \mathsf{Tr}(\alpha^{1-k}\beta).$$

Then the sequence generated by the FSR is $a_n = \mathsf{Tr}(\alpha^{-n}\beta)$.

Proof.

To see this first note that $\alpha^{-k} = \sum c_i \alpha^{i-k}$.

But then

$$\begin{aligned}&c_1\mathsf{Tr}(\alpha^{1-k}\beta) + c_2\mathsf{Tr}(\alpha^{2-k}\beta) + \ldots + c_k\mathsf{Tr}(\beta) = \\ &\mathsf{Tr}(\beta(c_1\alpha^{1-k} + c_2\alpha^{2-k} + \ldots + c_k)) = \\ &\mathsf{Tr}(\alpha^{-k}\beta)\end{aligned}$$

Done by induction. □

Initial Conditions

One might worry that the special form of the initial conditions

$$\mathsf{Tr}(\beta), \mathsf{Tr}(\alpha^{-1}\beta), \ldots, \mathsf{Tr}(\alpha^{2-k}\beta), \mathsf{Tr}(\alpha^{1-k}\beta).$$

makes the last result a bit wobbly, it might just cover a few cases.

Not so, we claim that different choices for β all produce distinct vectors.

For suppose two vectors agree; then by linearity there is a β such that $\mathsf{Tr}(\alpha^i\beta) = 0$ for all $i < k$. We can think of the condition $\mathsf{Tr}(\alpha^i\beta) = 0$ for all $i < k$, as a system of linear equations.

Vandermonde Matrices

The matrix of this system has a special form: it's (the transpose of) a Vandermonde matrix.

$$V(\gamma_1, \ldots, \gamma_m) = \begin{pmatrix} 1 & \gamma_1 & \gamma_1^2 & \ldots & \gamma_1^{n-1} \\ 1 & \gamma_2 & \gamma_2^2 & \ldots & \gamma_2^{n-1} \\ 1 & \gamma_3 & \gamma_3^2 & \ldots & \gamma_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \gamma_m & \gamma_m^2 & \ldots & \gamma_m^{n-1} \end{pmatrix}$$

The determinant of V is $\prod_{i<j}(\gamma_j - \gamma_i)$.

It follows that our matrix is invertible. But then $\beta = 0$, done. Hence all 2^k initial conditions can be generated by the right choice of the multiplier β: our description is perfectly general.

Fibonacci Feedback Shift-Registers

The feedback shift-registers we have considered so far are very natural since they generate linear recurrent sequences according to

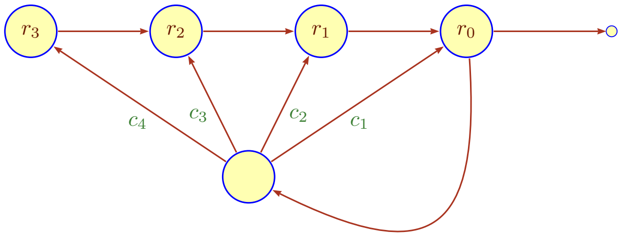
$$a_n = \sum_{i=1}^{k} c_i a_{n-i} = c_1 a_{n-1} + c_2 a_{n-2} + \ldots + c_k a_{n-k}$$

In other words, they generalize the classical Fibonacci sequence. Hence, these FSR are called Fibonacci feedback shift-registers (FFSR).

There is a dual notion of Galois feedback shift-registers (GFSR), essentially obtained by reversing all the arrows in the diagram.

Say Again?

Keeping our shift direction to go from left to right, the new device looks like so:



After shifting, the bits in the tapped registers are flipped, provided the bit in r_0 was a 1. [†]

[†]We have changed the numbering system on the tap coefficients, this makes the algebra a bit easier.

But Why?

In a Galois feedback-shift register, a single step corresponds exactly to multiplication by x in $\mathbb{F}_2[x]/(f)$.

As far as long cycles are concerned, there is no difference between Fibonacci and Galois FSRs.

Here is a brief glimpse at why this is true.

Reciprocal Polynomials

Definition

The reciprocal of a polynomial $f(x)$ of degree k is the polynomial $f^\star(x) = x^k \cdot f(1/x)$.

The map $f \mapsto f^\star$ is not very well behaved (it is not a homomorphism), but we have the following properties.

Proposition

Let f and g be two polynomials.

- *If $f(0) \neq 0$ then $(f^\star)^\star = f$.*
- $(f \cdot g)^\star = f^\star \cdot g^\star$.

Connection and Characteristic

By the last proposition, f is irreducible if, and only if, its reciprocal polynomial $f^\star$ is so irreducible.

Moreover, in characteristic 2 we have $(1+x^d)^\star = 1+x^d$, so the exponents of an irreducible polynomial and its reciprocal are the same.

Hence, as far as irreducibility and exponent are concerned, there is no difference between multiplication by and α and division by α.