Artificial Intelligence and Security Lab
Digital Security Group
Radboud University

Counting coprime polynomials... with complications

Luca Mariot

luca.mariot@ru.nl

DISCo – Milan, 1 September 2022

Coprime Polynomials

Object: pairs of binary polynomials of degree $n \in \mathbb{N}$:

$$f(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1} + x^n \ ,$$
$$g(x) = b_0 + b_1 x + \cdots + b_{n-1} x^{n-1} + x^n \ ,$$

where $a_i, b_i \in GF(2) = \mathbb{F}_2 = \{0, 1\}$

$$f, g \in \mathbb{F}_2[x] \text{ are } \textbf{coprime} \Leftrightarrow \gcd(f, g) = 1$$

Applications cryptography and coding theory:

- *Discrete logarithms* in finite fields [C84]
- Decoding *alternant codes* [F95]

Euclid's Algorithm

Check if $\gcd(f,g) = 1 \Rightarrow$ **Euclid's algorithm**

Example: $n = 3, \quad f(x) = x^3 + x^2 + x + 1, \quad g(x) = x^3 + 1$

Euclid's Algorithm

Check if $\gcd(f, g) = 1 \Rightarrow$ **Euclid's algorithm**

Example: $n = 3, \quad f(x) = x^3 + x^2 + x + 1, \quad g(x) = x^3 + 1$

$$f(x) = q(x) \cdot g(x) + r(x)$$

Euclid's Algorithm

Check if $\gcd(f, g) = 1 \Rightarrow$ **Euclid's algorithm**

Example: $n = 3, \quad f(x) = x^3 + x^2 + x + 1, \quad g(x) = x^3 + 1$

$$f(x) = q(x) \cdot g(x) + r(x)$$

$x^3 + x^2 + x + 1 = 1 \cdot (x^3 + 1) + (x^2 + x)$

Euclid's Algorithm

Check if $\gcd(f,g) = 1 \Rightarrow$ **Euclid's algorithm**

Example: $n = 3$, $f(x) = x^3 + x^2 + x + 1$, $g(x) = x^3 + 1$

$$f(x) = q(x) \cdot g(x) + r(x)$$

$x^3 + x^2 + x + 1 = 1 \cdot (x^3 + 1) + (x^2 + x)$
$x^3 + 1 = (x + 1) \cdot (x^2 + x) + (x + 1)$

Euclid's Algorithm

Check if $\gcd(f,g) = 1 \Rightarrow$ **Euclid's algorithm**

Example: $n = 3, \quad f(x) = x^3 + x^2 + x + 1, \quad g(x) = x^3 + 1$

$$f(x) = q(x) \cdot g(x) + r(x)$$

$x^3 + x^2 + x + 1 = 1 \cdot (x^3 + 1) + (x^2 + x)$
$x^3 + 1 = (x+1) \cdot (x^2 + x) + (x + 1)$
$x^2 + x = x \cdot (x + 1) + 0$

Euclid's Algorithm

Check if $\gcd(f,g) = 1 \Rightarrow$ **Euclid's algorithm**

Example: $n = 3, \quad f(x) = x^3 + x^2 + x + 1, \quad g(x) = x^3 + 1$

$$f(x) = q(x) \cdot g(x) + r(x)$$

$$x^3 + x^2 + x + 1 = 1 \cdot (x^3 + 1) + (x^2 + x)$$
$$x^3 + 1 = (x + 1) \cdot (x^2 + x) + (x + 1)$$
$$x^2 + x = x \cdot (x + 1) + 0$$

Compact notation:

$$(x^3 + x^2 + x + 1, x^3 + 1) \xrightarrow{1} (x^3 + 1, x^2 + x) \xrightarrow{x+1} (x^2 + x, x + 1) \xrightarrow{x} (x + 1, 0)$$

Euclid's Algorithm

Check if $\gcd(f,g) = 1 \Rightarrow$ **Euclid's algorithm**

Example: $n = 3, \quad f(x) = x^3 + x^2 + x + 1, \quad g(x) = x^3 + 1$

$$f(x) = q(x) \cdot g(x) + r(x)$$

$$x^3 + x^2 + x + 1 = 1 \cdot (x^3 + 1) + (x^2 + x)$$
$$x^3 + 1 = (x+1) \cdot (x^2 + x) + (x+1)$$
$$x^2 + x = x \cdot (x+1) + 0$$

Compact notation:

$$(x^3 + x^2 + x + 1, x^3 + 1) \xrightarrow{1} (x^3 + 1, x^2 + x) \xrightarrow{x+1} (x^2 + x, x + 1) \xrightarrow{x} (x+1, 0)$$

$gcd(f,g) = x + 1 \Rightarrow (f,g)$ **not** coprime

DilcuE's Algorithm

- **Remark:** (f, g) can be recovered from $(x+1, 0)$ with the same quotients in reverse order
- Called **DilcuE's algorithm** by Benjamin and Bennett [BB07]

$(x+1, 0) \xrightarrow{x}$

DilcuE's Algorithm

- **Remark:** (f, g) can be recovered from $(x+1, 0)$ with the same quotients in reverse order
- Called **DilcuE's algorithm** by Benjamin and Bennett [BB07]

$$(x+1, 0) \xrightarrow{x} (x^2+x, x+1) \xrightarrow{x+1}$$

DilcuE's Algorithm

- **Remark:** (f, g) can be recovered from $(x + 1, 0)$ with the same quotients in reverse order
- Called **DilcuE's algorithm** by Benjamin and Bennett [BB07]

$$(x+1,0) \xrightarrow{x} (x^2+x, x+1) \xrightarrow{x+1} (x^3+1, x^2+x) \xrightarrow{1} (x^3+x^2+x+1, x^3+1) = (f, g)$$

DilcuE's Algorithm

- **Remark:** (f, g) can be recovered from $(x+1, 0)$ with the same quotients in reverse order
- Called **DilcuE's algorithm** by Benjamin and Bennett [BB07]

$$(x+1,0) \xrightarrow{x} (x^2+x, x+1) \xrightarrow{x+1} (x^3+1, x^2+x) \xrightarrow{1} (x^3+x^2+x+1, x^3+1) = (f,g)$$

- Suppose we change the **last** remainder from 0 to 1:

$$(x+1,\mathbf{1}) \xrightarrow{x}$$

DilcuE's Algorithm

- **Remark:** (f, g) can be recovered from $(x+1, 0)$ with the same quotients in reverse order
- Called **DilcuE's algorithm** by Benjamin and Bennett [BB07]

$$(x+1, 0) \xrightarrow{x} (x^2+x, x+1) \xrightarrow{x+1} (x^3+1, x^2+x) \xrightarrow{1} (x^3+x^2+x+1, x^3+1) = (f, g)$$

- Suppose we change the **last** remainder from 0 to 1:

$$(x+1, \mathbf{1}) \xrightarrow{x} (x^2+x+1, x+1) \xrightarrow{x+1}$$

DilcuE's Algorithm

- **Remark:** (f,g) can be recovered from $(x+1,0)$ with the same quotients in reverse order
- Called **DilcuE's algorithm** by Benjamin and Bennett [BB07]

$$(x+1,0) \xrightarrow{x} (x^2+x, x+1) \xrightarrow{x+1} (x^3+1, x^2+x) \xrightarrow{1} (x^3+x^2+x+1, x^3+1) = (f,g)$$

- Suppose we change the **last** remainder from 0 to 1:

$$(x+1,\mathbf{1}) \xrightarrow{x} (x^2+x+1, x+1) \xrightarrow{x+1} (x^3+x^2, x^2+x+1) \xrightarrow{1} (x^3+x+1, x^3+x^2) = (f',g')$$

- By construction, (f',g') **are coprime**

Counting by Bijection

In essence: bijection for coprime/non-coprime pairs over $\mathbb{F}_2$:

1. Apply Euclid to (f, g)
2. If the last remainder is 0, change it to 1. Otherwise, set it to the second-last remainder
3. Apply DilcuE's algorithm to the reversed quotients

Theorem ([BB07, R00])

Let $f, g \in \mathbb{F}_2[x]$ of degree n be randomly chosen. Then, the probability that $\gcd(f, g) = 1$ is $\frac{1}{2}$.

In other words: the number of coprime pairs is 2^{2n-1}

Enter the complication

We require now that both f and g have a **nonzero constant term**:

$$f(x) = \mathbf{1} + a_1 x + \cdots + a_{n-1} x^{n-1} + x^n \ ,$$
$$g(x) = \mathbf{1} + b_1 x + \cdots + b_{n-1} x^{n-1} + x^n \ .$$

Problems:

1. *Count* all such pairs
2. *Enumeration algorithm*

Remark: the trick above does not work!

$$\textbf{non-coprime} \leftrightarrow \textbf{coprime}$$

$$(x^3 + x^2 + x + \mathbf{1}, x^3 + \mathbf{1}) \leftrightarrow (x^3 + x + \mathbf{1}, x^3 + x^2 \ (+\mathbf{0}))$$

... Why do we want to do that?

Orthogonal Latin Squares by Linear Cellular Automata

- Bipermutive Linear rule: $f(x) = x_1 \oplus a_1 x_2 \oplus \cdots \oplus a_{n-1} x_{n-1} \oplus x_n$
- Associated Polynomial: $P_f(X) = 1 + a_1 X + \cdots + a_{n-1} X^{n-1} + X^n$

Theorem ([MGFL20])

Two bipermutive linear CA generates orthogonal Latin squares if and only if their associated polynomials are coprime

1	4	3	2
2	3	4	1
4	1	2	3
3	2	1	4

(a) Rule 150

1	2	3	4
2	1	4	3
3	4	1	2
4	3	2	1

(b) Rule 90

1, 1	4, 2	3, 3	2, 4
2, 2	3, 1	4, 4	1, 3
4, 3	1, 4	2, 1	3, 2
3, 4	2, 3	1, 2	4, 1

(c) Superposition

Figure: $P_{150}(X) = 1 + X + X^2$, $P_{90}(X) = 1 + X^2$ (coprime)

Asking for clues (precisely 6 years ago...)

Asking for clues (precisely 6 years ago...)

Dear Arthur, what do you think of this complication?

Luca

Asking for clues (precisely 6 years ago...)

Dear Arthur, what do you think of this complication?

Luca

... One hour later...

Arthur Benjamin

Dear Luca, off the top of my head, there are $q^2 - 1$ equivalence classes, all of which are co-prime except one? But I may be wrong.

Asking for clues (precisely 6 years ago...)

Dear Arthur, what do you think of this complication?

Luca

... One hour later...

Arthur Benjamin

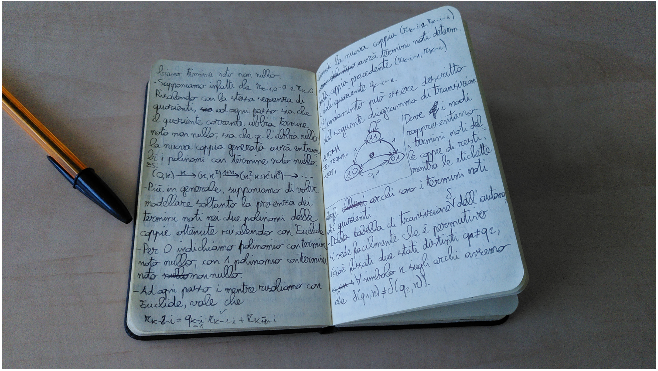
Dear Luca, off the top of my head, there are $q^2 - 1$ equivalence classes, all of which are co-prime except one? But I may be wrong.

?

One MONTH later...

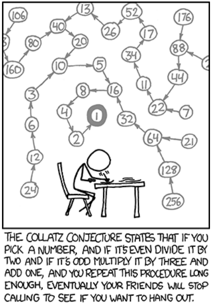
One MONTH later...

... He was indeed right! But took me several weeks to prove it



Sadly, the clue was not enough to solve the counting problem

Counting by Recurrence



S `https://xkcd.com/710/`

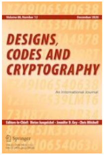
- Number of coprime polynomial pairs of degree n and nonzero constant term:

$$a(n) = 4^{n-1} + a(n-1) = \frac{4^{n-1}-1}{3}$$
$$= 0, 1, 5, 21, 85, \ldots$$

- Corresponds to OEIS A002450

- Generalized for any finite field $\mathbb{F}_q$ in [MGFL20] (but enumeration not addressed)

L. Mariot, M. Gadouleau, E. Formenti, and A. Leporati. Mutually orthogonal latin squares based on cellular automata. Des. Codes Cryptogr. 88(2):391–411 (2020)



Problem Structure

Strategy: characterize the *sequences* of quotients that gives only $(1,1)$ coprime pairs when starting from the remainders $(1,0)$

Three parts of the problem:

Problem Structure

Strategy: characterize the *sequences* of quotients that gives only $(1,1)$ coprime pairs when starting from the remainders $(1,0)$

Three parts of the problem:

$$
\begin{array}{lllll}
q_1 \rightarrow & \overbrace{x^{d_1}}^{\textit{degrees}} & + \overbrace{q_{1,d_1-1}x^{d_1-1} + \cdots + q_{1,1}x}^{\text{middle terms}} & + & \overbrace{s_1}^{\text{constant terms}} \\
q_2 \rightarrow & x^{d_2} & + q_{2,d_2-1}x^{d_2-1} + \cdots + q_{2,1}x & + & s_2 \\
\vdots \rightarrow & \vdots & + \quad \vdots \qquad\quad + \cdots + \ \vdots & + & \vdots \\
q_k \rightarrow & x^{d_k} & + q_{k,d_k-1}x^{d_k-1} + \cdots + q_{k,1}x & + & s_k
\end{array}
$$

Problem Structure

Strategy: characterize the *sequences* of quotients that gives only $(1,1)$ coprime pairs when starting from the remainders $(1,0)$

Three parts of the problem:

$$
\begin{array}{lcccc}
q_1 \rightarrow & \overbrace{x^{d_1}}^{\text{degrees}} & + \overbrace{q_{1,d_1-1}x^{d_1-1} + \cdots + q_{1,1}x}^{\text{middle terms}} + & \overbrace{s_1}^{\text{constant terms}} \\
q_2 \rightarrow & x^{d_2} & + q_{2,d_2-1}x^{d_2-1} + \cdots + q_{2,1}x + & s_2 \\
\vdots \rightarrow & \vdots & + \quad \vdots \quad + \cdots + \vdots \quad + & \vdots \\
q_k \rightarrow & x^{d_k} & + q_{k,d_k-1}x^{d_k-1} + \cdots + q_{k,1}x + & s_k
\end{array}
$$

Notation: $r_i, r_{i+1} \rightarrow$ consecutive remainders produced by Euclid's algorithm at step i. Step $i+1$:

$$r_i(x) = q_{i+1}(x)r_{i+1}(x) + r_{i+2}(x)$$

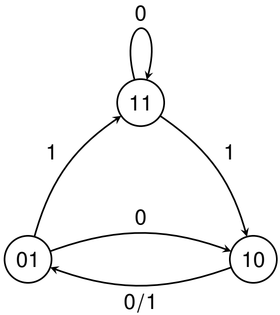
Finite State Automaton of Remainders

$$r_i(x) = q_{i+1}(x) r_{i+1}(x) + r_{i+2}(x)$$

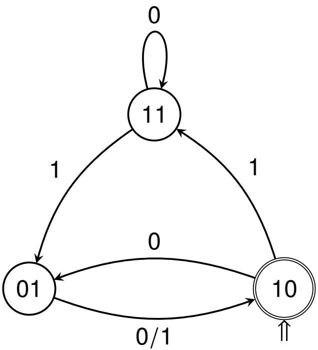
- $(c_i, c_{i+1}) \rightarrow$ constant terms of r_i and r_{i+1}
- $s_{i+1} \rightarrow$ constant term of q_{i+1}
- $\delta((c_i, c_{i+1}), s_{i+1}) \rightarrow$ *next* pair (c_{i+1}, c_{i+2})

(c_i, c_{i+1})	s_{i+1}	$\delta((c_i, c_{i+1}), s_{i+1})$
$(1,1)$	0	$(1,1)$
$(1,1)$	1	$(1,0)$
$(1,0)$	0	$(0,1)$
$(1,0)$	1	$(0,1)$
$(0,1)$	0	$(1,0)$
$(0,1)$	1	$(1,1)$

Remark: the pair $(0,0)$ *never* occurs



The Regular Language of Constant Terms Sequences



Inverse FSA

- The FSA is *permutative*: for DilcuE's, simply reverse the arrows
- **Initial state**: 10
- **Final state**: 11 (but we can use 10)

Regular Expression of the Language:

$$L = (0(0+1) + (10^*1(0+1)))^*$$

Enumeration/counting of Constant Terms Sequences

- **Enumeration**: generate all words of length k [M97]
- **Counting**: exploit *algebraic language theory*

Transform $L = (0(0+1)+(10^*1(0+1)))^*$ in a FPS as follows:

- $0,1 \Rightarrow X$
- $+,\cdot \Rightarrow +,\cdot$
- $^* \Rightarrow \frac{1}{1-X}$

Generating Function:

$$\sum_{k=0}^{\infty} a_k \cdot X^k = \frac{1-X}{1-X-2X^2} \ ,$$

Closed Form:

$$a_k = \frac{2^k + 2\cdot(-1)^k}{3}$$

Sequences of quotients' degrees

Second part: Characterize the *degrees* of the quotients

Example: $n = 4$, $\{1, x, x^2, x, 1\}$

$$(1,0) \xrightarrow{1} (1,1) \xrightarrow{x} (x+1,1) \xrightarrow{x^2} (x^3+x^2+1, x+1) \xrightarrow{x} (x^4+x^3+1, x^3+x^2+1) \xrightarrow{1} (x^4+x^2+1, x^4+x^3+1)$$

Sum of degrees: $1+2+1 = 4$, $k = 3$

Question: what are the combinations of *ordered sums* of n?

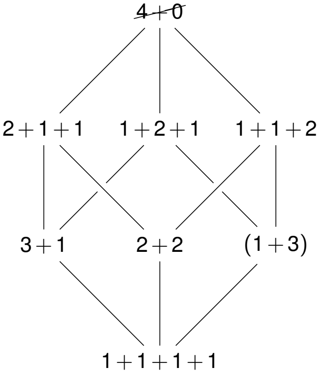
$\Rightarrow$ **compositions** of $n \in \mathbb{N}$

Quotients' degrees as compositions of n

- **Representation:** $n-1$ *boxes* that can be either "+" or ","

$$1\overbrace{\square 1 \square \ldots \square 1 \square}^{n-1} 1$$

- **Example:** $1,1+1,1 \rightarrow 1+2+1 \quad (n=4,\ k=3)$



- We remove the top of the poset
- **Enumeration**: generate all binary strings of length $1 < k < n$
- **Counting**: $\binom{n-1}{k-1}$

Enumeration Algorithm

- **Third part**: middle terms are *free*
- once k is fixed, all three parts are *independent*

Enumeration Algorithm

- **Third part**: middle terms are *free*
- once k is fixed, all three parts are *independent*

So for **enumeration**, given $n \in \mathbb{N}$:

For each composition *comp* of n of length k (except $k = 0$) do:

- Generate all quotients' sequences of *comp* (2^{n-k})
- For each quotients' sequence *seq* do:
  - For each constant term sequence of length k do:
    - Add the constant terms to the quotients
    - Apply DilcuE's from $(1, 0)$ by applying *seq*

Enumeration Algorithm

- **Third part**: middle terms are *free*
- once k is fixed, all three parts are *independent*

So for **enumeration**, given $n \in \mathbb{N}$:

For each composition *comp* of n of length k (except $k = 0$) do:

- Generate all quotients' sequences of *comp* (2^{n-k})
- For each quotients' sequence *seq* do:
  - For each constant term sequence of length k do:
    - Add the constant terms to the quotients
    - Apply DilcuE's from $(1, 0)$ by applying *seq*

And for **counting**, we reobtain the formula $\frac{4^{n-1}-1}{3}$ from:

Enumeration Algorithm

- **Third part**: middle terms are *free*
- once k is fixed, all three parts are *independent*

So for **enumeration**, given $n \in \mathbb{N}$:

For each composition *comp* of n of length k (except $k = 0$) do:

- Generate all quotients' sequences of *comp* (2^{n-k})
- For each quotients' sequence *seq* do:
  - For each constant term sequence of length k do:
    - Add the constant terms to the quotients
    - Apply DilcuE's from $(1, 0)$ by applying *seq*

And for **counting**, we reobtain the formula $\frac{4^{n-1}-1}{3}$ from:

$$\sum_{k=2}^{n}$$

Enumeration Algorithm

- **Third part**: middle terms are *free*
- once k is fixed, all three parts are *independent*

So for **enumeration**, given $n \in \mathbb{N}$:

For each composition *comp* of n of length k (except $k = 0$) do:

- Generate all quotients' sequences of *comp* (2^{n-k})
- For each quotients' sequence *seq* do:
  - For each constant term sequence of length k do:
    - Add the constant terms to the quotients
    - Apply DilcuE's from $(1, 0)$ by applying *seq*

And for **counting**, we reobtain the formula $\frac{4^{n-1}-1}{3}$ from:

$$\sum_{k=2}^{n} \underbrace{2^{n-k}}_{middle}$$

Enumeration Algorithm

- **Third part**: middle terms are *free*
- once k is fixed, all three parts are *independent*

So for **enumeration**, given $n \in \mathbb{N}$:

For each composition *comp* of n of length k (except $k = 0$) do:

- Generate all quotients' sequences of *comp* (2^{n-k})
- For each quotients' sequence *seq* do:
  - For each constant term sequence of length k do:
    - Add the constant terms to the quotients
    - Apply DilcuE's from $(1, 0)$ by applying *seq*

And for **counting**, we reobtain the formula $\frac{4^{n-1}-1}{3}$ from:

$$\sum_{k=2}^{n} \underbrace{2^{n-k}}_{\text{middle}} \cdot \underbrace{\binom{n-1}{k-1}}_{\textit{degrees}}$$

Enumeration Algorithm

- **Third part**: middle terms are *free*
- once k is fixed, all three parts are *independent*

So for **enumeration**, given $n \in \mathbb{N}$:

For each composition *comp* of n of length k (except $k = 0$) do:

- Generate all quotients' sequences of *comp* (2^{n-k})
- For each quotients' sequence *seq* do:
  - For each constant term sequence of length k do:
    - Add the constant terms to the quotients
    - Apply DilcuE's from $(1, 0)$ by applying *seq*

And for **counting**, we reobtain the formula $\frac{4^{n-1}-1}{3}$ from:

$$\sum_{k=2}^{n} \underbrace{2^{n-k}}_{\text{middle}} \cdot \underbrace{\binom{n-1}{k-1}}_{degrees} \cdot \underbrace{\frac{2^k + 2 \cdot (-1)^k}{3}}_{constant}$$

Conclusions and Future Work

Summing up:

- Enumeration more complicated with nonzero constant terms
- We divided the problem in three tasks:
  1. sequences of constant terms (⇒ regular language)
  2. sequences of degrees (⇒ compositions)
  3. sequences of middle terms (⇒ free)
- Results informally published in [FM22]

Future directions:

- Generalize to any finite field $\mathbb{F}_q$ and to m-tuples of polynomials
- Applications to cryptography [GM20, M21, GMP22]

Thank you!

Appendix: Orthogonal Latin Squares (OLS)

Definition

A *Latin square* is a $n \times n$ matrix where all rows and columns are permutations of $[n] = \{1, \cdots, n\}$. Two Latin squares are *orthogonal* if their superposition yields all the pairs $(x, y) \in [n] \times [n]$.

1	3	4	2
4	2	1	3
2	4	3	1
3	1	2	4

1	4	2	3
3	2	4	1
4	1	3	2
2	3	1	4

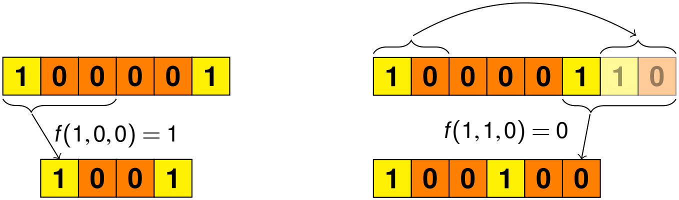
1 1	3 4	4 2	2 3
4 3	2 2	1 4	3 1
2 4	4 1	3 3	1 2
3 2	1 3	2 1	4 4

- k pairwise OLS are denoted as k-MOLS (**Mutually Orthogonal Latin Squares**)
- k-MOLS are **equivalent** $OA(n^2, k, n, 2)$

Appendix: Cellular Automata

- One-dimensional Cellular Automaton (CA): a discrete parallel computation model composed of a finite array of n cells

Example: $n = 6$, $d = 3$, $\omega = 0$, $f(s_i, s_{i+1}, s_{i+2}) = s_i \oplus s_{i+1} \oplus s_{i+2}$ (rule 150)



No Boundary CA – NBCA

Periodic Boundary CA – PBCA

- Each cell updates its state $s \in \{0,1\}$ by applying a local rule $f : \{0,1\}^d \to \{0,1\}$ to itself, the ω cells on its left and the $d-1-\omega$ cells on its right

Latin Squares through Bipermutive CA (1/2)

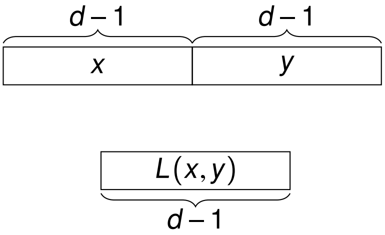
- Bipermutive CA: denoting $\mathbb{F}_2 = \{0,1\}$, local rule f is defined as

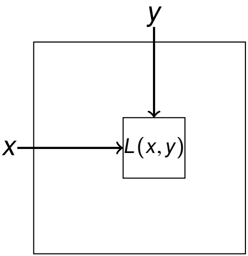
$$f(x_1,\cdots,x_d) = x_1 \oplus \varphi(x_2,\cdots,x_{d-1}) \oplus x_d$$

- $\varphi : \mathbb{F}_2^{d-2} \to \mathbb{F}_2$: generating function of f

Lemma ([MGFL20])

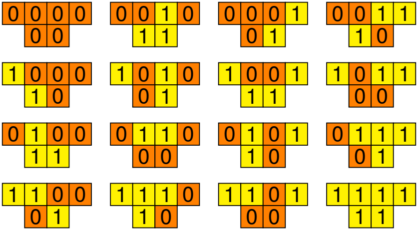
A CA $F : \mathbb{F}_2^{2(d-1)} \to \mathbb{F}_2^d$ with bipermutive rule $f : \mathbb{F}_2^d \to \mathbb{F}_2$ generates a Latin square of order $N = 2^{d-1}$





Latin Squares through Bipermutive CA (2/2)

- Example: CA $F : \mathbb{F}_2^4 \to \mathbb{F}_2^2$, $f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3$ (Rule 150)
- Encoding: $00 \mapsto 1, 10 \mapsto 2, 01 \mapsto 3, 11 \mapsto 4$



(a) Rule 150 on 4 bits

1	4	3	2
2	3	4	1
4	1	2	3
3	2	1	4

(b) Latin square L_{150}

Mutually Orthogonal Cellular Automata (MOCA): set of k bipermutive CA generating k-MOLS

References

[BB07] Benjamin, A.T., Bennett, C.D.: The probability of relatively prime polynomials. Mathematics Magazine 80(3): 196-202 (2007)

[C84] Coppersmith, D.: Fast evaluation of logarithms in fields of characteristic two. IEEE Trans. Inf. Theory 30(4): 587-593 (1984)

[F95] Fitzpatrick, P.: On the key equation. IEEE Trans. Inf. Theory 41(5): 1290-1302 (1995)

[FM22] Formenti, E., Mariot, L.: An Enumeration Algorithm for Binary Coprime Polynomials with Nonzero Constant Term. CoRR abs/2207.00406 (2022)

[GMP22] Gadouleau, M., Mariot, L., Picek, S.: Bent functions in the partial spread class generated by linear recurring sequences. Des. Codes Cryptogr. In press. DOI: https://doi.org/10.1007/s10623-022-01097-1 (2022)

[GM20] Gadouleau, M., Mariot, L.: Latin Hypercubes and Cellular Automata. Proceedings of Automata 2020, pp. 139-151 (2020)

[GR11] Ghorpade, S. R., Ram, S.: Block companion Singer cycles, primitive recursive vector sequences, and coprime polynomial pairs over finite fields. Finite Fields Their Appl. 17(5): 461-472 (2011)

[M97] E. Mäkinen: On Lexicographic Enumeration of Regular and Context-Free Languages. Acta Cybern. 13(1): 55-61 (1997)

[M21] Mariot, L.: Hip to Be (Latin) Square: Maximal Period Sequences from Orthogonal Cellular Automata. In: Proceedings of CANDAR 2021, pp. 29-37 (2021)

[MGFL20] Mariot, L., Gadouleau, M. Formenti, E., Leporati A.: Mutually orthogonal latin squares based on cellular automata. Des. Codes Cryptogr. 88(2):391–411 (2020)

[R00] Reifegerste, A.: On an involution concerning pairs of polynomials in $\mathbb{F}_2$. J. Combin. Theory Ser. A 90, 216-220 (2000)