Counting coprime polynomials...
with complications

Luca Mariot

luca.mariot@ru.nl

DiS Lunch Talks – June 10, 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 $a_i, b_i \in GF(2) = \mathbb{F}_2 = \{0, 1\}$

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

Applications of coprime pairs in cryptography and coding:

- *Discrete logarithms* in finite fields [C84]
- Decoding *alternant codes* [F95]
Euclid’s Algorithm

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

Example: $n = 3, \ f(x) = x^3 + x^2 + x + 1, \ g(x) = x^3 + 1$
Euclid’s Algorithm

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

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

$$f(x) = q(x) \cdot g(x) + r(x)$$
Euclid’s Algorithm

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

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

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

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

Luca Mariot

Counting coprime polynomials... with complications
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 $\gcd(f, g) = 1 \Rightarrow \text{Euclid’s algorithm}$

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

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

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

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

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

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:**

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

$$\gcd(f, g) = x + 1 \Rightarrow (f, g) \text{ not coprime}$$
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]

\[
(x + 1, 0) \xrightarrow{x} \]

Luca Mariot
Counting coprime polynomials... with complications
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]

\[
(x + 1, 0) \xrightarrow{x} (x^2 + x, x + 1) \xrightarrow{x+1}
\]
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:

  \[
  (x + 1, 1) \xrightarrow{x}
  \]
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:

\[
(x + 1, 1) \xrightarrow{x} (x^2 + x + 1, x + 1) \xrightarrow{x+1}
\]
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: 

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

- By construction, $(f', g')$ are coprime.
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, CSWZ98, 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 $2^{2n-1}$
We require now that both $f$ and $g$ have a **nonzero constant term**:

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

**Problems:**

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

**Remark:** the trick above does not work! Changing the last remainder gives no control over the final constant terms

\[
\text{non-coprime} \leftrightarrow \text{coprime}
\]

\[
(x^3 + x^2 + x + 1, x^3 + 1) \leftrightarrow (x^3 + x + 1, x^3 + x^2 (+0))
\]

... 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

---

<table>
<thead>
<tr>
<th>Rule 150</th>
<th>Rule 90</th>
<th>Superposition</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Rule 150" /></td>
<td><img src="image2.png" alt="Rule 90" /></td>
<td><img src="image3.png" alt="Superposition" /></td>
</tr>
</tbody>
</table>

(a) Rule 150  
(b) Rule 90  
(c) Superposition

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

---

Luca Mariot  
Counting coprime polynomials... with complications
Asking for clues (precisely 6 years ago...)

Luca

Dear Arthur, what do you think of this complication?

... 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.

Luca Mariot

Counting coprime polynomials... with complications
Dear Arthur, what do you think of this complication?

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

**Luca**

Dear Arthur, what do you think of this complication?

... 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...

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

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

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

$$a(n) = 4^{n-1} + a(n-1) = \frac{4^{n-1} - 1}{3}$$

$$= 0, 1, 5, 21, 85, ...$$

Corresponds to OEIS A002450

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

**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:
**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:

- **degrees**
  - \(q_1 \rightarrow x^{d_1}\)
  - \(q_2 \rightarrow x^{d_2}\)
  - \(\vdots \rightarrow \vdots\)
  - \(q_k \rightarrow x^{d_k}\)

- **middle terms**
  - \(q_{1,d_1-1}x^{d_1-1} + \cdots + q_{1,1}x + S_1\)
  - \(q_{2,d_2-1}x^{d_2-1} + \cdots + q_{2,1}x + S_2\)
  - \(\vdots + \vdots + \cdots + \vdots\)
  - \(q_{k,d_k-1}x^{d_k-1} + \cdots + q_{k,1}x + S_k\)

- **constant terms**
**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:

\[
\begin{align*}
q_1 \rightarrow x^{d_1} + q_{1,d_1-1}x^{d_1-1} + \cdots + q_{1,1}x + S_1 \\
q_2 \rightarrow x^{d_2} + q_{2,d_2-1}x^{d_2-1} + \cdots + q_{2,1}x + S_2 \\
\vdots \rightarrow \vdots + \vdots + \cdots + \vdots + \vdots \\
q_k \rightarrow x^{d_k} + q_{k,d_k-1}x^{d_k-1} + \cdots + q_{k,1}x + S_k
\end{align*}
\]

**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 \((c_{i+1}, c_{i+2})\)

<table>
<thead>
<tr>
<th>((c_i, c_{i+1}))</th>
<th>(s_{i+1})</th>
<th>(\delta((c_i, c_{i+1}), s_{i+1}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1, 1)</td>
<td>0</td>
<td>(1, 1)</td>
</tr>
<tr>
<td>(1, 1)</td>
<td>1</td>
<td>(1, 0)</td>
</tr>
<tr>
<td>(1, 0)</td>
<td>0</td>
<td>(0, 1)</td>
</tr>
<tr>
<td>(1, 0)</td>
<td>1</td>
<td>(0, 1)</td>
</tr>
<tr>
<td>(0, 1)</td>
<td>0</td>
<td>(1, 0)</td>
</tr>
<tr>
<td>(0, 1)</td>
<td>1</td>
<td>(1, 1)</td>
</tr>
</tbody>
</table>

**Remark:** the pair \((0, 0)\) *never* occurs
The Regular Language of Constant Terms Sequences

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 $L = (0(0 + 1) + (10 \cdot 1(0 + 1)))^*$ 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:**
\[
a_k = \frac{2^k + 2 \cdot (-1)^k}{3}
\]

Luca Mariot

Counting coprime polynomials... with complications
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 ","

- **Example:** $1, 1 \rightarrow 1 + 1$ (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 \( \text{comp} \) of \( n \) of length \( k \) (except \( k = 0 \)) do:
  - Generate all quotients’ sequences of \( \text{comp} \) \((2^{n-k})\)
  - For each quotients’ sequence \( \text{seq} \) 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 \( \text{seq} \)
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:

$$\sum_{k=2}^{n} \left( \frac{2^{n-k}}{\text{middle}} \right)$$
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:

$$\sum_{k=2}^{n} \text{middle} \cdot \frac{2^{n-k}}{k-1} \cdot \binom{n-1}{k-1}$$
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:

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

Luca Mariot

Counting coprime polynomials... with complications
Conclusions and Future Work

Summing up:

- Enumeration of binary coprime polynomials is more complicated when both constant terms are nonzero
- We divided the problem in three enumeration tasks:
  - sequences of constant terms ($\Rightarrow$ regular language)
  - sequences of degrees ($\Rightarrow$ compositions)
  - sequences of middle terms ($\Rightarrow$ free)

Future directions:

- Generalize to polynomials over any finite field $\mathbb{F}_q$
- Generalize to $m$-tuples of pairwise coprime polynomials
- Applications to cryptography and coding theory [GMP20, GM20, M21]
Thank you!
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 $(x, y) \in [n] \times [n]$.

$k$ 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)

\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}
\]
\( f(1,0,0) = 1 \)

\[
\begin{bmatrix}
1 & 0 & 0 \\
\end{bmatrix}
\]

No Boundary CA – NBCA

\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
\end{bmatrix}
\]
\( f(1,1,0) = 0 \)

\[
\begin{bmatrix}
1 & 0 & 0 & 1 & 0 & 0 \\
\end{bmatrix}
\]

Periodic Boundary CA – PBCA

- Each cell updates its state \( s \in \{0, 1\} \) by applying a local rule \( f : \{0, 1\}^d \rightarrow \{0, 1\} \) to itself, the \( \omega \) cells on its left and the \( d - 1 - \omega \) cells on its right.
Bipermutive CA: denoting $\mathbb{F}_2 = \{0, 1\}$, local rule $f$ is defined as
\[ f(x_1, \cdots, x_d) = x_1 \oplus \varphi(x_2, \cdots, x_{d-1}) \oplus x_d \]

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

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 $N = 2^{d-1}$

\[ L(x, y) \]

\[ d-1 \]

\[ d-1 \]

\[ x \]

\[ y \]

\[ x \]

\[ y \]

\[ L(x, y) \]
Example: CA $F : \mathbb{F}_2^4 \rightarrow \mathbb{F}_2^2$, $f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3$ (Rule 150)

Encoding: $00 \mapsto 1, 10 \mapsto 2, 01 \mapsto 3, 11 \mapsto 4$

(a) Rule 150 on 4 bits

(b) Latin square $L_{150}$

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


