Primitive Polynomials & Non-uniform Cellular Automata

Sumit Adak

Department of Information Technology
Indian Institute of Engineering Science and Technology, Shibpur, India

Primitive Polynomials

- A polynomial is primitive if it generates all the elements of an extension field from a base field.
- Suppose, $F(x)$ is a polynomial of degree n over $GF(2)$. Now, $F(x)$ is primitive if α is a root of $F(x)$ in $GF(2^n)$, and generates all the elements as successive power of α - $\{0, 1, \alpha, \alpha^2, \cdots, \alpha^{2^n-2}\}$.
- Deciding a polynomial as primitive in $GF(2)$ is considered as one of the difficult problems in Computer Science; see[1] for a good discussion.

[1] Sean Erik O Connor. Mathematical software and tutorials. http://www.seanerikoconnor. freeservers.com, 1999. Accessed: 2018-08-30.

Why?

- Primitive polynomials are having a range of applications in computing, from Coding Theory to Pseudo-Random Number Generation to Cryptography to many others.
- However, there is no algorithm known till date to decide primitive polynomial in polynomial time, and so we do not have primitive polynomials of arbitrary degrees.
- Till date, polynomials upto degree 5000 are synthesized using huge computing power.
- Here a natural question arises: is the problem NP-complete or NP-hard? We don't have any answer to this question either.

Problems

1. Decide whether an n-degree polynomial over $GF(2)$ is primitive.
2. Decide whether an n-cell binary (linear) CA is a maximal length CA.

Problems

1. Decide whether an n-degree polynomial over $GF(2)$ is primitive.
2. Decide whether an n-cell binary (linear) CA is a maximal length CA.

- There is a connection between primitive polynomials over $GF(2)$ and binary cellular automata (CAs).
- Further, we show that, these two problems are equivalent.

Problems

1. Decide whether an n-degree polynomial over $GF(2)$ is primitive.
2. Decide whether an n-cell binary (linear) CA is a maximal length CA.

- There is a connection between primitive polynomials over $GF(2)$ and binary cellular automata (CAs).
- Further, we show that, these two problems are equivalent.

- **Definition**

  An n-cell CA is maximal length if, for a configuration $x \in \mathcal{C} \setminus \{0^n\}$, $x = F^{2^n-1}(x)$, but $x \neq F^t(x)$ where $1 \leq t < 2^n - 1$.

Example: Maximal Length CA

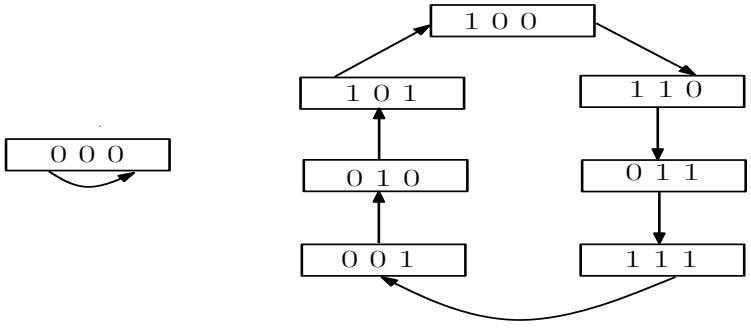

Figure: Transition diagram of CA $\langle 150, 90, 90 \rangle$

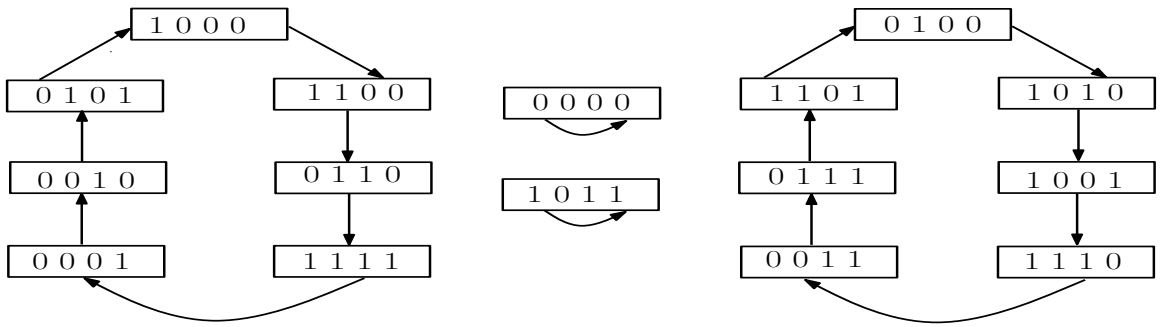

Figure: Transition diagram of CA $\langle 150, 90, 90, 90 \rangle$

Properties: Maximal Length CA

- *Non-uniform CA* : the cells may follow different rules. We, therefore, need a rule vector $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_i, \cdots, \mathcal{R}_{n-1} \rangle$ to define an n-cell non-uniform CA, where the cell i follows $\mathcal{R}_i$. Obviously, uniform CAs are special case of non-uniform CAs.
- *Reversible CA* : In a reversible CA, the initial configuration repeats after certain number of time steps.
- *Linear Rules* : If the rule of a CA cell involves only *XOR* logic. Linear Rules $= \{60, 90, 102, 150, 170, 204, 240\}$.
- *Linear CA* : Whenever all the $\mathcal{R}_i$s $(i = 0, 1, \cdots, n-1)$ of a rule vector $\mathcal{R}_n$ are linear, the CA is referred to as Linear CA.

Table: Rules 150 and 90

Present state (RMT)	111 (7)	110 (6)	101 (5)	100 (4)	011 (3)	010 (2)	001 (1)	000 (0)	Rule
(i) Next state	1	0	0	1	0	1	1	0	150
(ii) Next state	0	1	0	1	1	0	1	0	90

Getting Polynomial

- An n-cell linear CA can be expressed by a square matrix of dimension n. The matrix shows the dependency of a cell on other cells. Let T be an $n \times n$ matrix. Then,

$$T_{i,j} = \begin{cases} 1 & \text{Cell } i \text{ depends on cell } j \\ 0 & \text{Otherwise} \end{cases}$$

- Rule 90 $\quad f_i(x_{i-1}, x_i, x_{i+1}) = x_{i-1} + x_{i+1} \pmod 2$
  Rule 150 $\quad f_i(x_{i-1}, x_i, x_{i+1}) = x_{i-1} + x_i + x_{i+1} \pmod 2$

- For an example of the CA $\langle 150, 90, 90, 90 \rangle$. Now, we get the characteristic polynomial as $det(T_{CA} + Ix)$.

$$T_{CA} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}_{4 \times 4}$$

Maximal Length CAs and Primitive Polynomials

Theorem

A CA is maximal length iff the rule vector of the CA consists of rule 90 and rule 150, and its characteristic polynomial is primitive[2].

[2] Kevin Cattell and Jon C. Muzio. Synthesis of one-dimensional linear hybrid cellular automata. IEEE Trans. on CAD of Integrated Circuits and Systems, 15(3):325-335, 1996.

Maximal Length CAs and Primitive Polynomials

Theorem

A CA is maximal length iff the rule vector of the CA consists of rule 90 and rule 150, and its characteristic polynomial is primitive[2]*.*

Example

- $\langle 150, 90, 90 \rangle$: Polynomial $\rightarrow x^3 + x^2 + 1$ (Primitive)
- $\langle 150, 90, 90, 90 \rangle$: Polynomial $\rightarrow x^4 + x^3 + x^2 + 1 = (x + 1)(x^3 + x + 1)$ (Not Primitive)

[2] Kevin Cattell and Jon C. Muzio. Synthesis of one-dimensional linear hybrid cellular automata. IEEE Trans. on CAD of Integrated Circuits and Systems, 15(3):325-335, 1996.

Our Tool : Reachability Tree

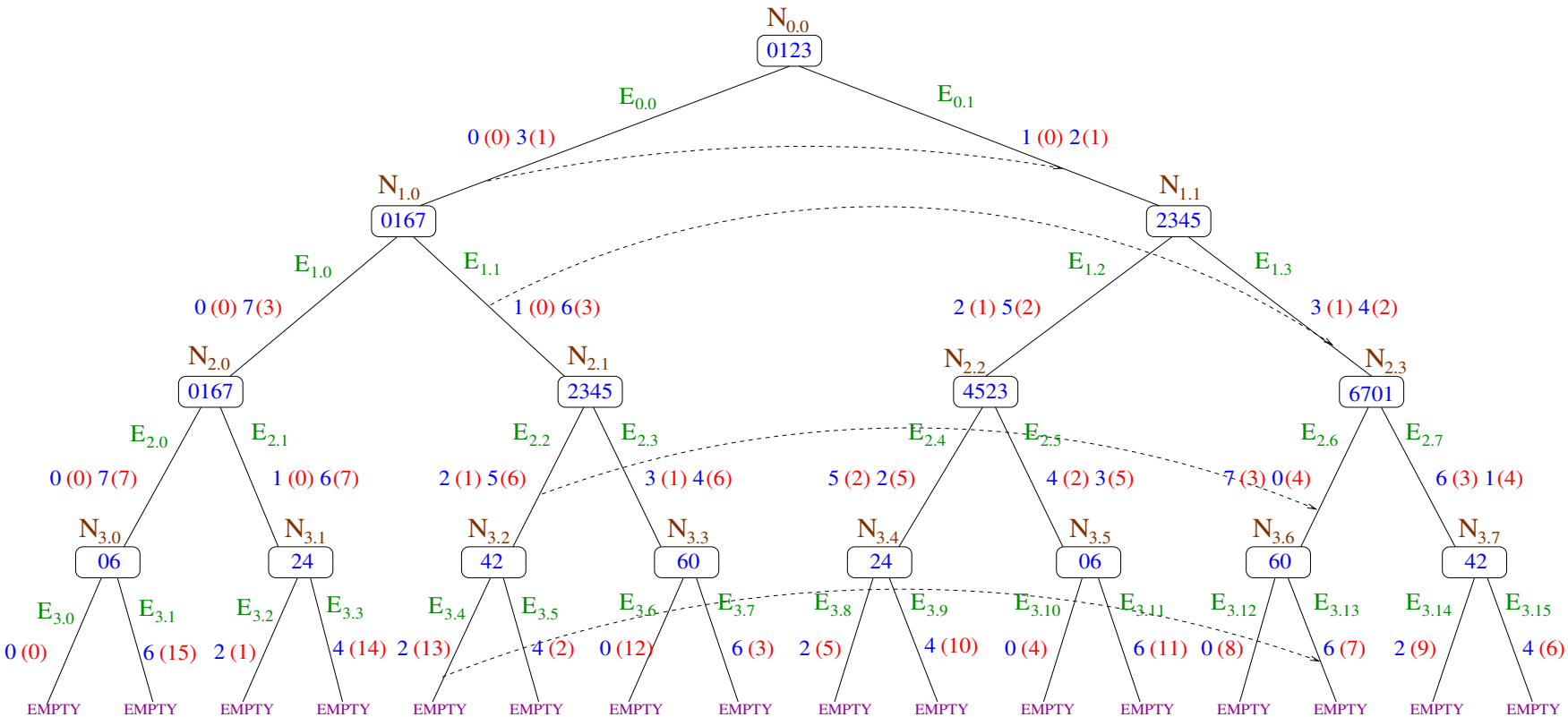

Figure: Reachability Tree of CA $\langle 150, 90, 90, 90 \rangle$

Example : Maximal Length CA

Theorem

A CA with n-cells is maximal length iff a single cross link is formed by the edges $E_{n-1.j}$ $(1 \leq j \leq 2^n - 1)$.

Example : Maximal Length CA

Theorem

A CA with n-cells is maximal length iff a single cross link is formed by the edges $E_{n-1.j}$ $(1 \leq j \leq 2^n - 1)$.

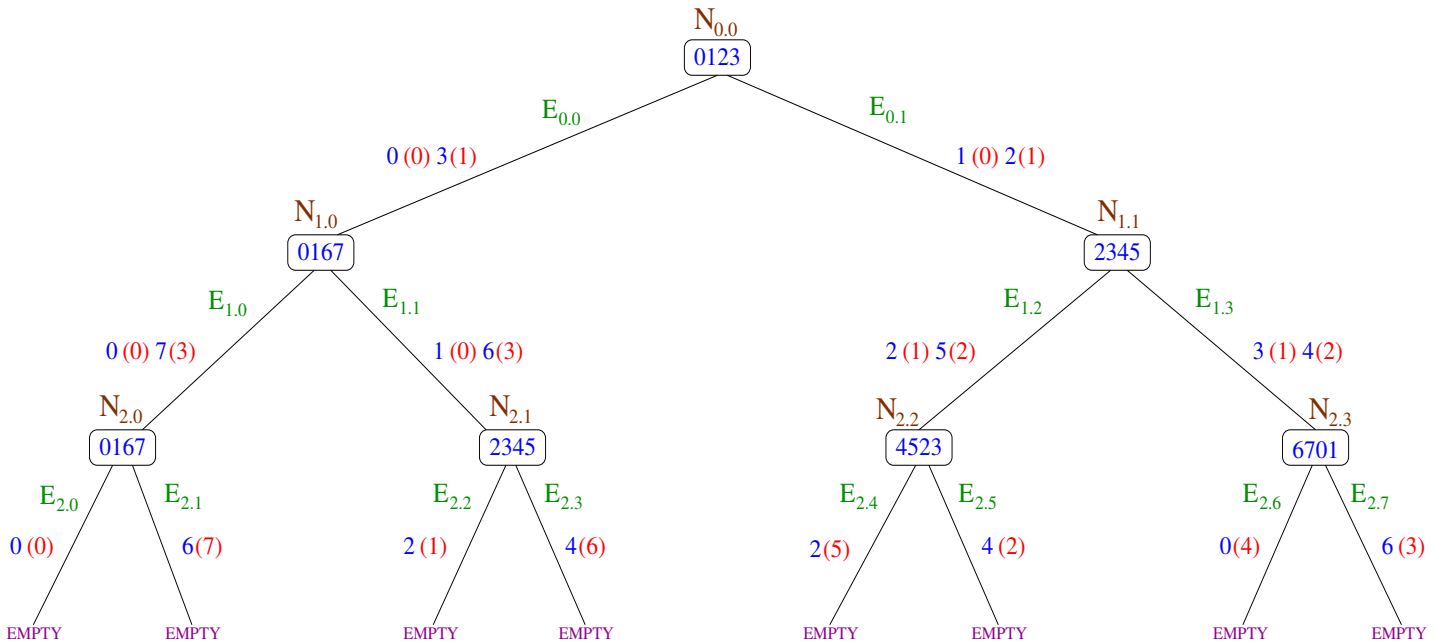

Figure: Reachability Tree of CA $\langle 150, 90, 90 \rangle$

- $E_{2.1}(6) \rightarrow E_{2.7}(6) \rightarrow E_{2.3}(4) \rightarrow E_{2.6}(0) \rightarrow E_{2.4}(2) \rightarrow E_{2.5}(4) \rightarrow E_{2.2}(2) \rightarrow E_{2.1}$
- So, $length(E_{2.1}, E_{2.1}) = 7$

$CA(90')$

Definition:

An n-cell CA with rule vector $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{n-1} \rangle$ is called $CA(90')$ with n-cells if $\mathcal{R}_0 = 150$ and $\mathcal{R}_i = 90$ where $1 \leq i \leq n-1$.

$CA(90')$

Definition:

An n-cell CA with rule vector $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{n-1} \rangle$ is called $CA(90')$ with n-cells if $\mathcal{R}_0 = 150$ and $\mathcal{R}_i = 90$ where $1 \leq i \leq n-1$.

Algorithm:

I. Take n as input

II. Set $start \leftarrow 0$, $end \leftarrow n-1$ and $count \leftarrow 0$

III. If $2 * start < n-1$, then set $start \leftarrow 2 * start + 1$. Otherwise, $start \leftarrow (n-1-start) * 2$.

IV. $count \leftarrow count + 1$

V. If $start \neq end$, then goto step III.

VI. If $count \neq n-1$, then $CA(90')$ is not Maximal Length for size n. Otherwise, $CA(90')$ is Maximal Length for size n

Results

Table: Maximal length CAs - $CA(90')$

2	3	5	6	9	11	14	18*	23	26	29
30	33	35	39	41	50*	51	53	65	69	74
81	83	86	89	90	95	98*	99*	105	113	119
131	134*	135	146	155	158	173	174*	179	183	186*
189	191	194*	209	210	221	230	231	233	239	243
245	251	254	261	270*	273	278*	281	293	299	303
306	309	323	326	329	330	338*	350*	354*	359	371
375	378*	386	393*	398	410*	411	413	414*	419	426
429	431	438*	441	443	453	470	473	483	491	495
509	515	519	530	531	543	545	554	558	561	575
585	593	606*	611	614*	615	618	629	638	639	641
645*	650*	651	653	659	683	686*	690*	713	719	723
725	726	741*	743	746	749	755	761	765	771	774
779	783*	785	791	803*	809	810	818	831	833	834
846	866	870	873	879	891	893	911	923	930*	933
935	938	939	950*	953	965	974*	975	986*	989	993
998										

Thank You