Contents

Logic

Connectives

- negation (¬A) - logical not
- conjuction (A ∧ B) - logical and
- disjunction (A ∨ B) - logical or
- exclusive or (A ⊕ B) - xor
- implication (A ⇒ B) - if A is true then B, the conclusion is true
- biimplication (A ⇔ B) - if and only if A is true then B is true

the above are also ordered by priority of "connection" ie $A \vee B \wedge C \vee D$ is to be read as $(A \vee (B \wedge C) \vee D)$, since ∧ has greater binding priority than ∨

implication associates to the right, so $A \Rightarrow (B \Rightarrow C)$ is the same as $A \Rightarrow B \Rightarrow C$

Propositional Formulae

- ⊤ (true) and ⊥ (false) are formulae
- Every propositional variable is a formula
- if A and B are two formulae, then A *connective* B is also a formulae

uppercase letters like *A, B, C* are used to denote formulae

lowercase letters like *p, q, x, y* are used to denote propositional variables

Truth Tables

- sigma (σ) is used for truth assignments
- $\sigma(\top)$ is True, T, 1; $\sigma(\bot)$ is False, F, 0
- there's the usual not, true, or, xor stuff
- $\sigma(A \Rightarrow B) = F$ only when $\sigma(A) = T$ and $\sigma(B) = F$, otherwise $\sigma(A \Rightarrow B) = T$
- $\sigma(A \Leftrightarrow B) = T$ only when $\sigma(A) = \sigma(B)$, otherwise $\sigma(A \Leftrightarrow B) = F$

the only way that an implication can fail is for the premise to be true, but for the conclusion to be false ($A \Rightarrow B \equiv \neg A \vee B$)

for n variables there are 2^n possible combinations of truth values

Tautologies

Formula that are always true no matter what truth assignments their variables have are called **tautologies** or **valid** formulae

Examples

- $A \vee \neg A$
- $A \Rightarrow A$
- $A \wedge B \Rightarrow A \vee C$
- $((A \wedge B \Rightarrow C) \wedge (A \Rightarrow B)) \Rightarrow (A \Rightarrow C)$

By contrast, a formula such as $A \wedge \neg A$ that is always false is called a **contradiction**

A formula A is said to be **satisfiable** or a **contingency** if there is at least one truth assignment σ such that $\sigma(A) = T$

A is satisfiable if, and only if, $\neg A$ fails to be a tautology

Equivalence

two formulae A and B are **equivalent** ($A \equiv B$) if for any combination of truth values of the variables in A and B, the truth value of A is the same as the truth value of B: $\sigma(A) = \sigma(B)$ for all truth assignments σ

any implication $A \Rightarrow B$ can be associated with three others by interchanging and/or negating the premise and conclusion:

1. Converse: $B \Rightarrow A$
2. Inverse: $\neg A \Rightarrow \neg B$
3. Contrapositive: $\neg B \Rightarrow \neg A$

any implication is equivalent to its contrapositive

Laws of Propositional Logic

important examples of equivalences before propositional formulae

- Assosiativity

  $A \vee (B \vee C) \equiv (A \vee B) \vee C$ and $A \wedge (B \wedge C) \equiv (A \wedge B) \wedge C$.
- Commutativity

  $A \vee B \equiv B \vee A$ and $A \wedge B \equiv B \wedge A$.
- Distributivity

  $A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$ and $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$.
- De Morgan

  $\neg(A \wedge B) \equiv \neg A \vee \neg B$ and $\neg(A \vee B) \equiv \neg A \wedge \neg B$.
- Identity

  $A \vee \bot \equiv A$ and $A \wedge \top \equiv A$.
- Idempotence

  $A \vee A \equiv A$ and $A \wedge A \equiv A$.
- Absorption $A \vee (A \wedge B) \equiv A$ and $A \wedge (A \vee B) \equiv A$.

Examples

- $A \Rightarrow B \equiv \neg A \vee B$
- $A \wedge B \equiv \neg(\neg A \vee \neg B)$
- $A \oplus B \equiv (A \wedge \neg B) \vee (\neg A \wedge B)$
- $A \Leftrightarrow B \equiv (A \Rightarrow B) \wedge (B \Rightarrow A)$

Simplification

If some complicated formula A has a subformula B, and we know that $B \equiv B'$, then we can replace B by B' without affecting truth values: the new formula A' will always be equivalent to A – and may well be smaller or easier to read

a connective $*$ is **associative** if the formula $p * (q * r) \equiv (p * q) * r$ $\oplus$, $\wedge$, $\vee$, and $\Leftrightarrow$ are associative connectives.

NAND connective

NAND truth table: TTTF

NAND, $p \uparrow q$, is functionally complete which means that any Boolean expression can be re-expressed by an equivalent expression utilizing only NAND operations.

$\top$	$p \uparrow (p \uparrow p)$
$\bot$	$(p \uparrow (p \uparrow p)) \uparrow (p \uparrow (p \uparrow p))$
$\neg p$	$p \uparrow p$
$p \wedge q$	$(p \uparrow q) \uparrow (p \uparrow q)$
$p \vee q$	$(p \uparrow p) \uparrow (q \uparrow q)$
$p \Rightarrow q$	$p \uparrow (q \uparrow q)$
$p \Leftrightarrow q$	$(p \uparrow q) \uparrow ((p \uparrow p) \uparrow (q \uparrow q))$

Negation, Disjunctive, Conjunctive Normal Form

Normal Form

normal form requires that a formula is written using only negations, disjunctions, and conjunctions. The first step in rewriting these formulas is to make sure that negations occur only immediately next to propositional variables ($\neg p$ or p, but not $\neg(p \wedge q)$)

Negation Normal Form (NNF)

A formula is in **negation normal form**, or **NNF**, if it only contains negations in the form of literals. To bring a formula into NNF,first eliminate all connectives other than conjunctions, disjunctions and negations, then use the rewrite rules:

$$\neg(A \wedge B) \mapsto \neg A \vee \neg B$$

$$\neg(A \vee B) \mapsto \neg A \wedge \neg B$$

$$\neg\neg A \mapsto A$$

Disjunctive Normal Form (DNF)

DNF is a ”sum of products”, (disjunction of conjuctions (of literals)) in the form:

$$(x_{11} \wedge x_{12} \wedge \ldots \wedge x_{1n_1}) \vee (x_{21} \wedge \ldots \wedge x_{2n_2}) \vee \ldots \vee (x_{m1} \wedge \ldots \wedge x_{mn_m})$$

where each x_{ij} is a literal indexed by i and j

NNF to DNF rewrite rules:

$$A \wedge (B_1 \vee B_2) \mapsto (A \wedge B_1) \vee (A \wedge B_2)$$

$$(B_1 \vee B_2) \wedge A \mapsto (B_1 \wedge A) \vee (B_2 \wedge A)$$

the DNF of $p \Leftrightarrow q$ is easily seen to be $(p \wedge q) \vee (\neg p \wedge \neg q)$.

Conjunctive Normal Form (CNF)

CNF is a “product of sums” (conjuctions of disjunctions) in the form:

$$(x_{11} \vee x_{12} \vee \ldots \vee x_{1n_1}) \wedge (x_{21} \vee \ldots \vee x_{2n_2}) \wedge \ldots \wedge (x_{m1} \vee \ldots \vee x_{mn_m})$$

NNF to CNF rewrite rules:

$$A \vee (B_1 \wedge B_2) \mapsto (A \vee B_1) \wedge (A \vee B_2)$$

$$(B_1 \wedge B_2) \vee A \mapsto (B_1 \vee A) \wedge (B_2 \vee A)$$

$p \Leftrightarrow q$ is easily seen to be $(\neg p \vee q) \wedge (p \vee \neg q)$

Large Formulae

small formulae example: the *modus ponenes*

P implies Q. P is true. Therefore Q must also be true.

$$p \wedge (p \Rightarrow q) \Rightarrow q$$

repeated sum: $\sum_{n=1}^{k} a_n$
repeated product: $\prod_{n=1}^{k} a_n$
repeated disjunction: $\bigwedge_{n=1}^{k} A_n$
repeated conjunction: $\bigvee_{n=1}^{k} A_n$

a CNF formula: $\bigwedge_i \bigvee_j \ell_i j$
a DNF formula: $\bigvee_i \bigwedge_j \ell_i j$

de Morgan’s law for any number of terms: $\neg \bigvee_i A_i \equiv \bigwedge_i \neg A_i$

Counting Functions

let EO_k be a formula with k propositional variables that is true if, and only if, exactly one of the variables is true. The general case looks like:

$$EO_k(x_1, \ldots, x_k) = \bigvee_{i=1}^{k} x_i \wedge \bigwedge_{1 \leq i < j \leq k} \neg(x_i \wedge x_j)$$

The disjunction forces at least one variable to be true and the conjunction ensures that for each pair of two variables at least one must be false

let ET_k be a formula with k propositional variables that expresses “exactly two of the variables are true”

$$ET_k(x_1, \ldots, x_k) = \bigvee_{i<j} (x_i \wedge x_j) \wedge \bigwedge_{i<j<\ell} \neg(x_i \wedge x_j \wedge x_\ell)$$

the disjunction says ”at least two” and the conjunction says ”not three or more”

Sets

Set Formation and Extensionality

$\{1,2,3\}$	a set composed of the numbers 1,2,3
$x \in S$	x is in the set S
$x \notin S$	x is not in the set S
$\{1,2,\ldots,99,100\}$	the set of numbers containing exactly the integers from 1 to 100
$\emptyset$ or $\{\}$	an empty set
$S = \{x \vert P(x)\}$	let S be the set of all x with property P
$S = \{x \in A \vert P(x)\}$	let S be the set of all x in set A with property P
$B \subseteq A$	set B is a subset of set A

in sets there is no order and no multiplicity, so $\{a,b,c\} = \{b,a,a,c,a,c,b\}$. these sets have the same **cardinality**, or the same number of elements

Principle of extensionality: two sets are equal if, and only if, they have the same elements

Principle of set Comprehension (or **Set Formation**): one can always form sets by taking items from a different set (see below)

$\mathbb{N} = \{0,1,2,...\}$	natural numbers
$\mathbb{Z} = \{\pm n \mid n \in \mathbb{N}\}$	integers
$\mathbb{Q} = \{a/b \mid a,b \in \mathbb{Z}, b \neq 0\}$	rationals
$\mathbb{R}$	reals

when you use set formation, like $B = \{x \in A \vert P(X)\}$, all of B's elements belong to A, so B is a **subset** of A

any set B is a subset of A ($B \subseteq A$) if, $\forall\, x, x \in B \Rightarrow x \in A$

$$A = B \Leftrightarrow A \subseteq B \wedge B \subseteq A$$

for any set $A, \emptyset \subseteq A$ and $A \subseteq A$

transitivity of the subset relation: $A \subseteq B$ and $B \subseteq C$ implies that $A \subseteq C$

Set operations

union	$A \cup B$	$= \{x \mid x \in A \vee x \in B\}$
intersection	$A \cap B$	$= \{x \mid x \in A \wedge x \in B\}$
difference	$A \setminus B$	$= \{x \mid x \in A \wedge x \notin B\}$
symmetric diff.	$A \Delta B$	$= \{x \mid x \in A \oplus x \in B\}$

union: combine the two sets
intersection: what do both sets have in common?
difference: $A \setminus B = A - (A \cap B)$; produces the elements that are only in A, not in B
symmetric difference: $A \Delta B = (A \cup B) - (A \cap B) = (A \setminus B) \cup (B \setminus A)$; produces the elements that are only in A and B, but not both

- Associativity

  $A \cup (B \cup C) = (A \cup B) \cup C$ and $A \cap (B \cap C) = (A \cap B) \cap C$

- Commutativity

  $A \cup B = B \cup A$ and $A \cap B = B \cap A$
- Distributivity

  $A \cup (B \cap C) = (A \cap B) \cup (A \cap C)$ and $A \cap (B \cup C) = (A \cup B) \cap (A \cup C)$
- Idempotence

  $A \cup A = A$ and $A \cap A = A$
- Absorption

  $A \cup (A \cap B) = A$ and $A \cap (A \cup B) = A$

complement: the complement of set A is often written as $\overline{A}$. In general we fix some universe $\mathbf{U}$ and consider only $A \subseteq \mathbf{U}$

$$\bar{A} = \mathbf{U} - A$$

Assume $A, B \subseteq \mathbf{U}$ for some fixed universe $\mathbf{U}$:

- Identity

  $A \cup \emptyset = A$ and $A \cap U = A$.
- Domination

  $A \cup U = U$ and $A \cap \emptyset = \emptyset$.
- Complements

  $A \cup \overline{A} = U$ and $A \cap \overline{A} = \emptyset$.
- Double Complement (involution)

  $\overline{\overline{A}} = A$.
- De Morgan's Laws

  $\overline{A \cup B} = \overline{A} \cap \overline{B}$ and $\overline{A \cap B} = \overline{A} \cup \overline{B}$.

Numbers and Lists as Sets

von Neumann numbers

You can represent the natural numbers as sets. To represent the first n natural numbers, you could use:

$$N_0 \rightsquigarrow \emptyset$$
$$N_n \rightsquigarrow S(S(\ldots S(\emptyset)\ldots))$$

N_n grows at the rate 2^n, so N_3 has 2^3, 8 nodes in it

First 4 N_n's obtained in this fashion:

$$N_0 \rightsquigarrow \{\}$$
$$N_1 \rightsquigarrow \{\{\}\}$$
$$N_2 \rightsquigarrow \{\{\}, \{\{\}\}\}$$
$$N_3 \rightsquigarrow \{\{\}, \{\{\}\}, \{\{\}, \{\{\}\}\}\}$$

These are called von Neumann numbers

Infinity

$\omega = \{N_n \mid n \geq 0\}$ can be considered as a number representing infinity. The next infinite number, $S(\omega) = \omega \cup \{\omega\}$, is written $\omega + 1$

We can get to $\omega + \omega$ and add two infinite numbers in a meaningful way (while $\infty + \infty$ is nonesense). This is helpful for analyzing the behavior of functions defined by multiple recursions (like the Ackermann function)

Pairing

$\pi(x, y)$ representa a hypothetical **pairing** operation on sets. Pairing is a method that associates any two sets with a sing set, so that individual sets can be recovered from that single set (for something like an ordered pair)

$$\pi(u, v) = \pi(x, y) \text{ implies } u = x \wedge v = y$$

$$\pi(x, y) = \{\{x\}, \{x, y\}\}$$

$\pi(x, x) = \{\{x\}\}$, a set of cardinality 1, regardless of what x is.

The **Cartesian product** of A and B is defined by

$$A \times B = \{(a, b) \mid a \in A, b \in B$$

$\{1, 2, 3\} \times \{\triangle, \Box\} = \{(1, \triangle), (2, \triangle), (3, \triangle), (1, \Box), (2, \Box), (3, \Box)\}$

the Cartesian product operation is not commutative or associative.

$$A \times B = B \times A \Rightarrow A = B$$

$$A \times (A \times A) = (A \times A) \times A \Leftrightarrow A = \emptyset$$

$A \times \emptyset = \emptyset \times A = \emptyset$ no matter what the set A is

$$(A \times B) \cap (C \times D) = (A \cap C) \times (B \cap D)$$

Limitations of Sets

Russell's Paradox

suppose we create the set of all sets which do not contain themselves

$$S = \{x \mid x \notin x\}$$

is S an element of itself? if $S \in S$, then it has the property $x \notin x$, which means that $S \notin S$. If $S \notin S$, then Set Formation would have put S in S (since it has the property $x \notin x$, so then $S \in S$... so we have to abandon the axiom of Formation or Extensionality

Functions

Domains, Codomains, Graphs

let A and B be two sets. a **function** (from A to B) is a set $f \subseteq A \times B$ s.t for any $a \in A$ $\exists$ exactly one $b \in B$ s.t $(a, b) \in f$

A is the **domain** of f and B is its **codomain**. For $(a, b) \in f$, b is the **image** of a under f, $f(a) = b$. a is a **preimage** of b. All elements $a \in A$ must have exactly one image ($b = f(a) \in B$), but not every b needs to have a preimage (they could have one, multiple, or none)

f, the set of pairs, is sometimes called the **graph** of the function.

$f : A \to B$ states that f is a function from A to B

a **partial function** is a function undefined on some elements of the domain

the **image** (or **range**) of the function is the collection of elements of B that do occur as the image of some point in A

identity function $I_A : A \to A$ defined by $I_A(x) = x$. Thus I_A returns exactly its input

constant functions $C_a : A \to A$ defined by $C_a(x) = a$. Thus C_a always returns a no matter what the input is

$$\begin{aligned} f : \mathbb{R} &\to \mathbb{R} \\ x &\mapsto x^2 + 1 \end{aligned}$$

the function above has the domain and codomain $\mathbb{R}$. $\to$ indicates the domain and codomain, $\mapsto$ indicates what a particular element maps to. this is the function $f = \{(x, x^2 + 1) \mid x \in \mathbb{R}\}$. If the domain and codomain are left out you can just assume its the reals.

Composition

given two functions $f : A \mapsto B$ and $g : B \mapsto C$ we can form the **composition** $h : A \mapsto C$ as

$$h(x) = g(f(x)) \text{ or } h(x) = g \circ f$$

h is a function with domain A and codomain C. this only works when the domain of g is the codomain of f

composition isn't usally *commutative*. the identity function and another function f are commutative, though, and simply to just f.

$$I_A \circ f = f \circ I_A = f$$

composition is *associative*:

$$h \circ (g \circ f) = (h \circ g) \circ f$$

functions f s.t $f \circ f = f$ are called **idempotent**

Classification

a function is **surjective** or **onto** if its image and codomain are exactly the same; if for every possible output there is a corresponding input that will produce this particular output a function is **injective** or **one-to-one** if no two distinct elments in the domain have the same image:

$$f(a) = f(b) \Leftrightarrow a = b$$

with injective, or *reversible* functions, one can uniquely reconstruct the preimage a s.t $f(a) = b$ given just the image b

a function is **bijective** if it is both injective and surjective

functions $\mathbb{R} \mapsto \mathbb{R}$:

$x \mapsto x^2$	not injective, not surjective
$x \mapsto x^3 - x$	not injective, surjective
$x \mapsto e^x$	injective, not surjective
$x \mapsto x^3$	injective, surjective (bijective)

Iteration

iteration is repeating a (basic) function some number of times and then returning the final output. to do this we need functions that have the same domain and codomain (sometimes called **endofunctions** or **square** functions

the sequence of elements of the domain obtained by iteration is called the **trajectory** or **orbit** of the argument under the function

let the domain and codomain of function f be $\mathbb{N}$, and $f(x) = x^2$. the trajectory of 2 under f is

$$2, 4, 16, 256, 65536, \ldots, 2^{2^n}, \ldots$$

... I'm pretty sure the collatz conjecture is an example of iteration?

Transients and Periods

if the set A is finite, then th etrajectories of any endofunction $f : A \to A$ wrap around in a **lasso**. the nonrepeated numbers are the **transient**, and the repeated sections are called the **period**

A sequence $a_0, \ldots, a_{n-1}$ in A is a **cycle** of f if $f(a_i) = a_{i+1 \bmod n}$; these points form a loop of length n. A cycle of n is also called an **n-cycle**. In the case $n = 1$, $a \in A$ is a **fixed point** of f if $f(a) = a$

$f^t(a)$ represents applying f exactly t times to a where $t \in \mathbb{N}$. $f^0 = I_A$ and $f^1 = f$. The orbit of a under f is **periodic** if for some $p > 0 \; \exists \; f^p(a) = a$ (this would create a lasso with just a period, no transient).

the orbit of a under f is **ultimately periodic** if for some $t \geq 0$ and $p > 0 \; \exists \; f^{t+p}(a) = f^t(a)$

The least t and p s.t $f^t(x) = f^{t+p}(x)$ is the **transient length** and the **period length** of the orbit of x

determining transient and periods for larger cycles

Stage One: the algorithm discovers a point on the cycle.

Stage Two: the point on the cycle just discovered is used to determine the period, i.e., the length of the cycle.

Stage Three: based on knowledge of the period, one determines the transient.

Cellular Automata

cellular automata: like conway's game of life. a **local rule** is used to determine what the new state of the center cell should be, and then is updated for all cells with the **global rule**.

elementary cellular automa: a linear sequence of **cells** being in a state of 1 or 0

example with local rule: $\rho : 2 \times 2 \times 2 \mapsto 2$

$$\ldots 0\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 0 \ldots$$

rearrange the bits into overlapping blocks of 3 bits each:

$$\ldots, 000, 001, 010, 101, 011, 110, 100, 000, \ldots$$

apply the local rule ρ to all these blocks and get back a new sequence:

$$\ldots \rho(000)\ \rho(001)\ \rho(010)\ \rho(101)\ \rho(011)\ \rho(110)\ \rho(100)\ \rho(000) \ldots$$

there are 256 possible rules for ECA. ECA 110 is "capable of universal computation" (so is turing complete?)

cyclic boundary condition: when we deal with finite bit sequences we assume the first cell is adjacent to the last cell

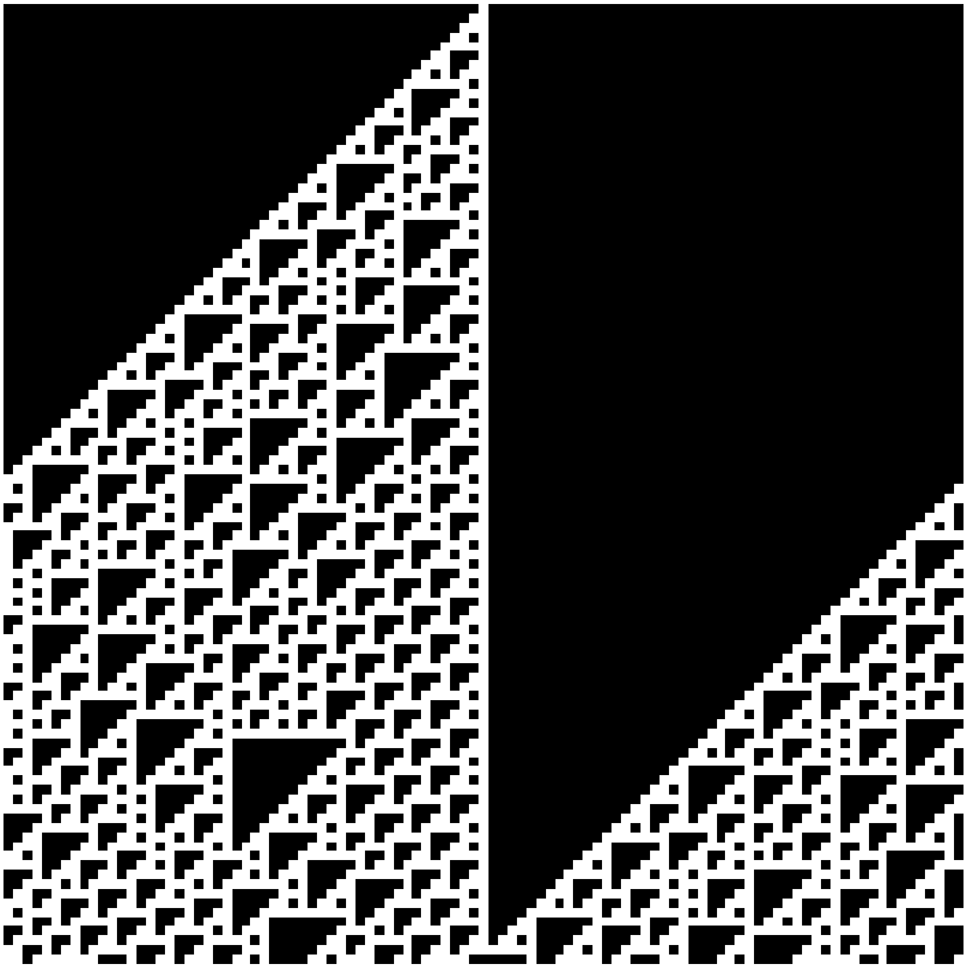
any injective global map f is automatically also surjective on $2^{\mathbb{Z}}$

one-point seed configuration: when testing out new global rules, set the sequence a to a single 1 in the middle surrounded by all 0s

code example

a python program to show the output of an ECA rule, configuration, and repeition count as an image

```python
import numpy as np
from PIL import Image
def cell(eca, conf, count):
    array=[]
    rule = bin(eca)[2:]
    if len(rule) < 8:
        rule = "0" * (8-len(rule)) + rule
    array=[[int(i) for i in conf]]
    for j in range(count):
        row=[]
        for i in range(len(conf)):
            sub=int(conf[i-1] + conf[i] + conf[(i+1) % len(conf)], 2)
            row.append(int(rule[7-sub]))
        array.append(row)
        conf = "".join([str(k) for k in row])
    Image.fromarray(np.array(array, dtype=bool)).show()
    return array
```



ECA rule 110 (image created using program above) on a one-point seed configuration