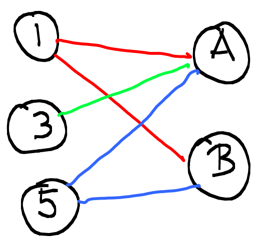
Logic

We need functions for next-state and output symbols. Since we know we can use {0,1} as a symbol basis for both states and symbols, we'll start with Boolean functions.

Set = collection of things: $\{5, 7, 9, 13\}$

Relation (binary) = pairs of things from 2 sets: $S_1 = \{1, 3, 5\}$

$S_2 = \{A, B\}$

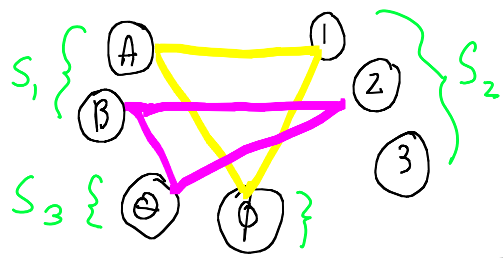


$R = \left\{ \begin{array}{l} \{1,A\}, \{3,A\}, \{5,A\}, \\ \{1,B\}, \{5,B\} \end{array} \right\}$

R is a set of "tuples"

(Ternary) $S_1 = \{A, B\}$ $S_2 = \{1, 2, 3\}$ $S_3 = \{\theta, \phi\}$

$R = \left\{ \begin{array}{l} \{A, 1, \phi\}, \{B, 2, \theta\}, \\ \{A, 2, \phi\}, \{A, 3, \theta\} \end{array} \right\}$



Function (map)

$f : S_1 \rightarrow S_2$

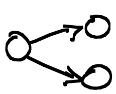
S1 = { 1, 2, 3 }
S2 = { A, B, C, D }

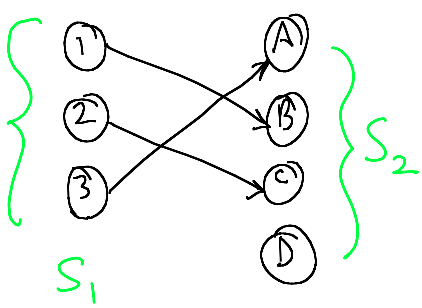
f = { (1,B), (2,C), (3,A) }

f(1) = B, f(2) = C, f(3) = A

function?
1-To-1 ?
onTo ?
operator?

exactly one arrow each

No 



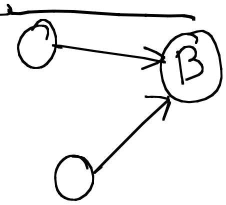
f() is a subset of crossproduct S1 X S2
(As is any relation.)

$S_1 \backslash S_2$	A	B	C	D
1	(1,A)	(1,B)	(1,C)	(1,D)
2	(2,A)	(2,B)	(2,C)	(2,D)
3	(3,A)	(3,B)	(3,C)	(3,D)

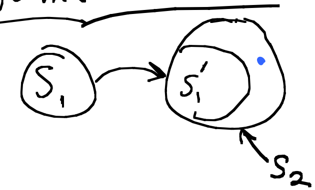
$S_1 \times S_2$

injective (1-to-1):

This doesn't happen



surjective (ON-TO):



$S_1' = f(S_1) = S_2$

image of S_1

Operator (binary)

$f : S_1 \times S_1 \rightarrow S_1$

$\boxed{+(2,3) \rightarrow 5}$

Boolean variable

$x \in \{0, 1\}$ $\qquad 0 =$ "false" $\qquad 1 =$ "True"

f, a boolean n-ary operator, is a subset of all pairs (x,y):
---- x is an binary n-tuple,
---- y is binary, i.e., 0 or 1.

f is a subset of { binary n-tuples } X { 0, 1 }

Boolean function (n-ary operator)

$$f : \{0,1\}^n \longrightarrow \{0,1\}$$

n-way X product ⇒ n tuples. E.g., 4-tuples (0,1,0,0)

$\{0,1\}^2$

$(0,0) \longrightarrow 1$
$(0,1) \longrightarrow 0$
$(1,0) \longrightarrow 0$
$(1,1) \longrightarrow 1$

variable

Value of variable

A	B	f(A,B)	
0	0	1	$f(0,0) = 1$
0	1	0	$f(0,1) = 0$
1	0	0	$f(1,0) = 0$
1	1	1	$f(1,1) = 1$

S = { (0,0), (0,1), (1,0), (1,1) } is the set of all binary 2-tuples.
f = { ((0,0), 1), ((0,1), 0), ((1,0), 0), ((1,1), 1) } is a subset of all possible pairs, S X {0, 1}.
f is a function; so, there must be exactly 4 pairs in ***f***. How many pairs in S X {0, 1}? How many different functions are possible? That is, how many different 4-element subsets of S X {0, 1} are there?

We will start from here, that is, from the simplest functions we can imagine, and see if we can build on this to construct arbitrary functions. But first, we'll see what the range is we available now.

Unary boolean functions: How many?

X	f(x)
0	0
1	0

X	G(x)
0	1
1	1

X	h(X)
0	0
1	1

X	K(X)
0	1
1	0

$f : \{0,1\} \to 0 \qquad G : \{0,1\} \to 1 \qquad h(x) = X \qquad K(x) = \text{NOT}(X) = \overline{X}$

Binary boolean functions: How many?

X Y		AND					XOR	OR
0 0	0	0	0	0	0	0	0	0
0 1	0	0	0	0	1	1	1	1
1 0	0	0	1	1	0	0	1	1
1 1	0	1	0	1	0	1	0	1

X Y	$\overline{\text{OR}}$	$\overline{\text{XOR}}$					$\overline{\text{AND}}$	
0 0	1	1	1	1	1	1	1	1
0 1	0	0	0	0	1	1	1	1
1 0	0	0	1	1	0	0	1	1
1 1	0	1	0	1	0	1	0	1

How many k-ary functions are there?

e.g. 3-ary

			f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

How many ways to form this column?

Each 8-component bit-vector represents the output of a unique 3-input boolean function.

rotate

[0,0,0,0,0,0,0,0]
[0,0,0,0,0,0,0,1]
[0,0,0,0,0,0,1,0]
...
[1,1,1,1,1,1,1,1]

all possible 3-ary function output vectors

Hmm, counting in binary from 0 to (2^8 - 1).

==> 2^(2^3) different 3-ary functions.

==> 2^(2^k) different k-ary functions.

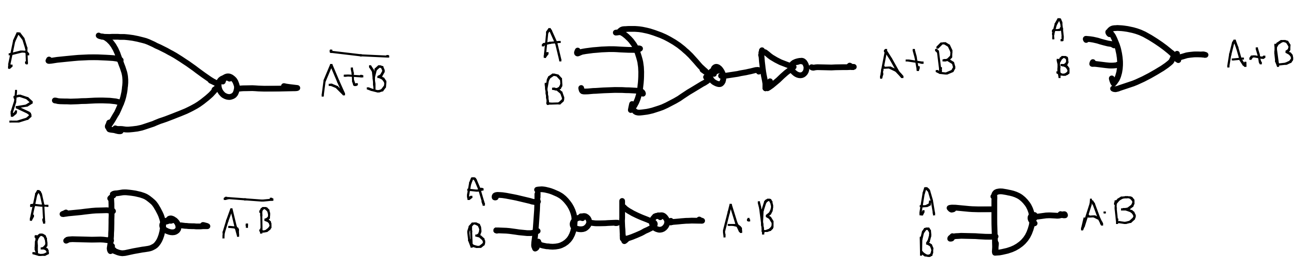
Back to our task:

We want to be able to build any arbitrary boolean function.
Why?
Because we want to build a computer, a UTM.
And why is that relevant?
UTMs are TMs, which have FSMs in them.

To be able to build any arbitrary FSM, we need to be able to build:

--- arbitrary next-state functions,
--- arbitrary output functions,
--- STATE elements.

what we can build so far:



NOT, NOR, NAND, OR, AND. Questions: Do we know how to build,

(1) EVERY k-input, n-output boolean function?   (n-output = n output columns. Seems hard.)

(2) EVERY 2-input, 1-output boolean function?   (Hmm, still unclear, try something easier.)

(3) EVERY 2-input, 1-output function whose output has exaclty one 1?

Hmm, none of these seem easy. The last one seems easiest. Maybe we should explore Boolean functions a bit more first.

Propositions and Logical Connectives

X	Y	OR
0	0	0
0	1	1
1	0	1
1	1	1

This truth table makes the following statement:

(X OR Y) is TRUE, exactly when

(NOT(X) and Y) is TRUE
OR when
(X and NOT(Y)) is TRUE
OR when
(X and Y) is TRUE

Let X = "It is raining" Y = "My hat is lost"

(X + Y) is TRUE in three cases, exactly when

(("It is raining" is FALSE) AND ("My hat is lost" is TRUE)) is TRUE

OR when
(("It is raining" is TRUE) AND ("My hat is lost" is FALSE)) is TRUE

OR when
(("It is raining" is TRUE) AND "(My hat is lost" is TRUE)) is TRUE

x	y	AND
0	0	0
0	1	0
1	0	0
1	1	1

(X AND Y) is TRUE only once, exactly when

(("It is raining" is TRUE) AND ("My hat is lost" is TRUE)) is TRUE

X	Y	f
0	0	1
0	1	0
1	0	0
1	1	1

Interpreting the truth table (1st row):

$f(0, 0)$ = TRUE Consists of two parts,

(1) The part that identifies the ROW of the truth table:

("It is NOT raining" = TRUE) AND ("My hat is NOT lost" = TRUE)

(2) The part that defines the function's VALUE for that row:

f ("It is raining", "My hat is lost") = TRUE

"It is raining" and "My hat is lost" are Boolean propositional arguments to the function, f : Each has either the value TRUE or the value FALSE.

Function Composition

$$f : \mathbb{R} \to \mathbb{R} \qquad f(x) = 2x$$

$$g : \mathbb{R} \to \mathbb{R} \qquad g(x) = \frac{1}{3}x$$

$$f(g(x)) = f(x/3) = 2\frac{x}{3}$$

$$f \circ g\,(x) \qquad f \circ g : \mathbb{R} \xrightarrow[g]{1/3} \mathbb{R} \xrightarrow[f]{2} \mathbb{R}$$

argument

argument

X	NOT(X)	NOT(NOT(X))
0	1	0
1	0	1

X	NOT(X)
0	1
1	0

X	NOT(NOT(X))
0	0
1	1

$$\mathrm{AND}(\mathrm{OR}(x,y),\ \mathrm{NOT}(x)) = (X+Y)\cdot\overline{X}$$

X	Y	X+Y	$\overline{X}$	$\{(X+Y)\cdot\overline{X}\}$
0	0	0	1	0
0	1	1	1	1
1	0	1	0	0
1	1	1	0	0

this is $\mathrm{AND}((x+y),\ \overline{X})$

All possible inputs are listed:
x = 0 , y = 0
x = 0 , y = 1
x = 1 , y = 0
x = 1 , y = 1

this appears twice?

a b

a	b	a·b
0	0	0
0	1	0
1	0	0
1	1	1

Is there a row missing?

function decomposition

We can compose simple functions to build more complex functions.

What if we have a complex function and want to build it from simpler ones? Decompose it to simpler functions.

x	y	f
0	0	0
0	1	1
1	0	0
1	1	1

x	y	g
0	0	0
0	1	1
1	0	0
1	1	0

x	y	h
0	0	0
0	1	0
1	0	0
1	1	1

Simpler functions:

Exactly one output is 1.

$$f(x,y) = OR(g(x,y), h(x,y))$$

f is TRUE exactly when
((X is FALSE) AND (Y is TRUE)) is TRUE
OR when
((X is TRUE) AND (Y is TRUE)) is TRUE

g is TRUE exactly when
((X is FALSE) AND (Y is TRUE)) is TRUE

h is TRUE exactly when
((X is TRUE) AND (Y is TRUE)) is TRUE

x	y	g	h	$\underbrace{g+h}$ = $\underline{f}$
0	0	0	0	0
0	1	1	0	1
1	0	0	0	0
1	1	0	1	1

f is TRUE exactly when

g is TRUE
OR
h is TRUE

ie., $f = (g + h)$

g is TRUE only once, but then *h* is FALSE
h is TRUE only once, but then *g* is FALSE

They completely and disjointly define *f*

Works for any 2-input function!

Decompose any *f* into simple single-1-output functions, then OR these together.

So far we can
(0) build NOT
(1) build four 2-input functions (NOR, NAND, AND, OR)
(2) decompose complex 2-input functions

Can we build any arbitrary SIMPLE function (single-1-output function)? How?

If we can, we can build ANY 2-input function.