JavaScript #3 Monday, August 26, 2019

[illegible]

2
4
7
6
———
19

☐ Logic

Given two Boolean Variables P and Q, there are 4 possible combinations of values that can produce 16 possible outcomes.

☐ Inputs P and Q to the operators:

```
  P | Q
---+---
  T | T
  T | F
  F | T
  F | F
```

☐ Enumerating the 16 possible outcomes of Boolean Operations
Table is labeled in hexadecimal

0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F
T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F
T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F
T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F

☐ NAMING AND PROVING THE SIXTEEN OPERATORS

```
operator 0 is = not(f(P,Q)) = NXAOPQ  = not(xor(and( or(P ,    Q))))
operator 1 is = not(e(P,Q)) =    OPQ  =                or(P ,   Q )
operator 2 is = not(d(P,Q)) =   OPNQ  =                or(P , not(Q))
operator 3 is = not(c(P,Q)) =     P   =                  P
operator 4 is = not(b(P,Q)) =   ONPQ  =           or(not(P),    Q )
operator 5 is = not(a(P,Q)) =      Q  =                         Q
operator 6 is = not(9(P,Q)) =   NXPQ  =   not(xor(     P ,     Q ))
operator 7 is = not(8(P,Q)) =    APQ  =        and(     P ,     Q  )
operator 8 is = not(7(P,Q)) =   NAPQ  =   not(and(      P ,    Q ))
operator 9 is = not(6(P,Q)) =    XPQ  =        xor(      P ,    Q  )
operator a is = not(5(P,Q)) =     NQ  =      not(                Q  )
operator b is = not(4(P,Q)) =  NONPQ  =  not( or(not(    P),    Q ) )
operator c is = not(3(P,Q)) =     NP  =  not(              P         )
operator d is = not(2(P,Q)) =  NOPNQ  = not( or(          P , not(Q)) )
operator e is = not(1(P,Q)) =   NOPQ  = not( or(          P ,    Q ) )
operator f is = not(0(P,Q)) =  XAOPQ  =    xor(and( or(P ,     Q)) )
```

☒ Words are chords of letters

Three-state logic is equivalent to yes/no/maybe, white/black/gray
if-then-else is a non-commutative ternary operator
d = ifThenElse(a,b,c) // A whole theory of computation emerges...