Predicate Logic and Specifying Circuits

EECS4312: Software Engineering Requirements

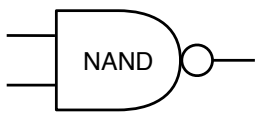
JSO, EECS, Lassonde

September 5, 2016

Table of contents

Nand Gate Specification

NAND gate Specification
(Complete and Disjoint Function Table)

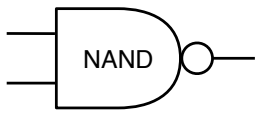


NAND GATE

Input		Output
A	B	NAND
0	0	1
0	1	1
1	0	1
1	1	0

Nand Gate Specification

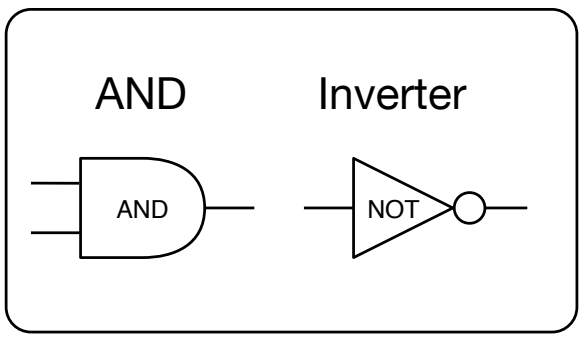
NAND gate Specification
(Complete and Disjoint Function Table)



NAND GATE

Input		Output
A	B	NAND
0	0	1
0	1	1
1	0	1
1	1	0

Implementation of NAND gate
using basic gates



Nand Gate Specification: Functional

subrange(i,j): TYPE $= \{k : \mathbb{Z} | i \leq k \leq j\}$

```
nand : THEORY
BEGIN
  BIT: TYPE = subrange(0,1)

  andf(x:BIT,y:BIT): BIT =
    COND
      x=1 AND y=1 -> 1,
      ELSE        -> 0
    ENDCOND

  invf(x:BIT): BIT = 1 - x
```

Question

Would BIT: TYPE $= \{0, 1\}$ work?

Nand Gate Specification: Functional

subrange(i,j): TYPE $= \{k : \mathbb{Z} | i \leq k \leq j\}$

```
nand : THEORY
BEGIN
  BIT: TYPE = subrange(0,1)

  andf(x:BIT,y:BIT): BIT =
    COND
      x=1 AND y=1 -> 1,
      ELSE        -> 0
    ENDCOND

  invf(x:BIT): BIT = 1 - x
```

Question

Would BIT: TYPE $= \{0, 1\}$ work?

Nand Gate Specification: Functional

subrange(i,j): TYPE $= \{k : \mathbb{Z} | i \leq k \leq j\}$

```
nand : THEORY
BEGIN
  BIT: TYPE = subrange(0,1)

  andf(x:BIT,y:BIT): BIT =
    COND
      x=1 AND y=1 -> 1,
      ELSE        -> 0
    ENDCOND

  invf(x:BIT): BIT = 1 - x
```

Question

Would BIT: TYPE $= \{0, 1\}$ work?

Nand Gate Specification: Functional

subrange(i,j): TYPE $= \{k : \mathbb{Z} | i \leq k \leq j\}$

```
nand : THEORY
BEGIN
  BIT: TYPE = subrange(0,1)

  andf(x:BIT,y:BIT): BIT =
    COND
      x=1 AND y=1 -> 1,
      ELSE        -> 0
    ENDCOND

  invf(x:BIT): BIT = 1 - x
```

Question

Would BIT: TYPE $= \{0, 1\}$ work?

Nand Gate Specification: Functional

subrange(i,j): TYPE $= \{k : \mathbb{Z} | i \leq k \leq j\}$

```
nand : THEORY
BEGIN
  BIT: TYPE = subrange(0,1)
...

  impf(x:BIT,y:BIT): BIT =
    invf(andf(x,y))

  specf(x:BIT, y:BIT): BIT =
    COND
      x = 0 OR y = 0 -> 1,
      ELSE           -> 0
    ENDCOND

  verify_functional_implementation: CONJECTURE
    impf(x,y) = specf(x,y)
```

Nand Gate Specification: Functional

subrange(i,j): TYPE $= \{k : \mathbb{Z} | i \leq k \leq j\}$

```
nand : THEORY
BEGIN
  BIT: TYPE = subrange(0,1)
...

  impf(x:BIT,y:BIT): BIT =
    invf(andf(x,y))

  specf(x:BIT, y:BIT): BIT =
    COND
      x = 0 OR y = 0 -> 1,
      ELSE           -> 0
    ENDCOND

  verify_functional_implementation: CONJECTURE
    impf(x,y) = specf(x,y)
```

Nand Gate Specification: Functional

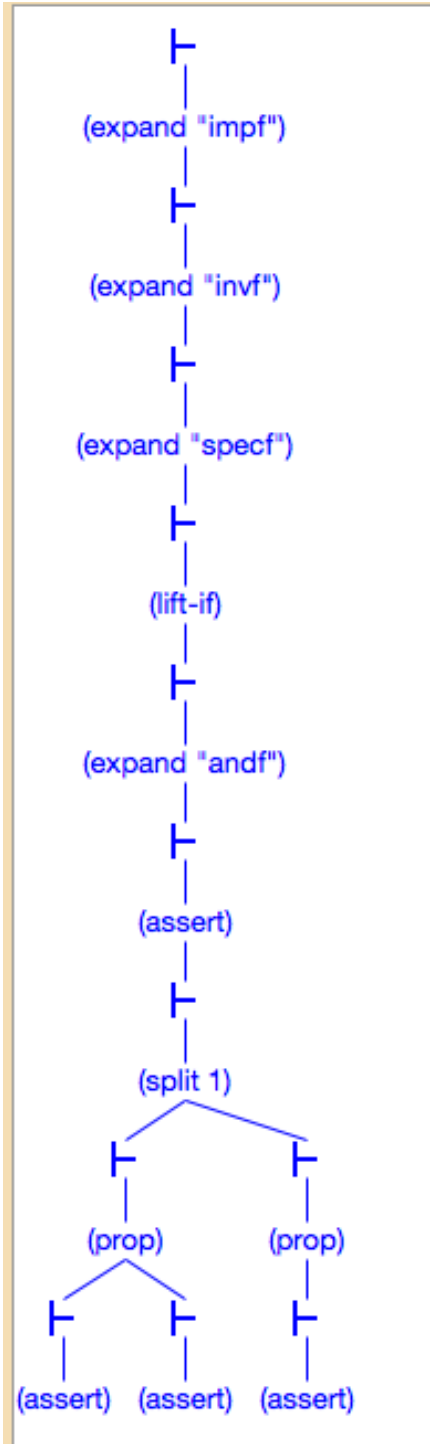
subrange(i,j): TYPE $= \{k : \mathbb{Z} | i \leq k \leq j\}$

```
nand : THEORY
BEGIN
  BIT: TYPE = subrange(0,1)
...

  impf(x:BIT,y:BIT): BIT =
    invf(andf(x,y))

  specf(x:BIT, y:BIT): BIT =
    COND
      x = 0 OR y = 0 -> 1,
      ELSE           -> 0
    ENDCOND

  verify_functional_implementation: CONJECTURE
    impf(x,y) = specf(x,y)
```

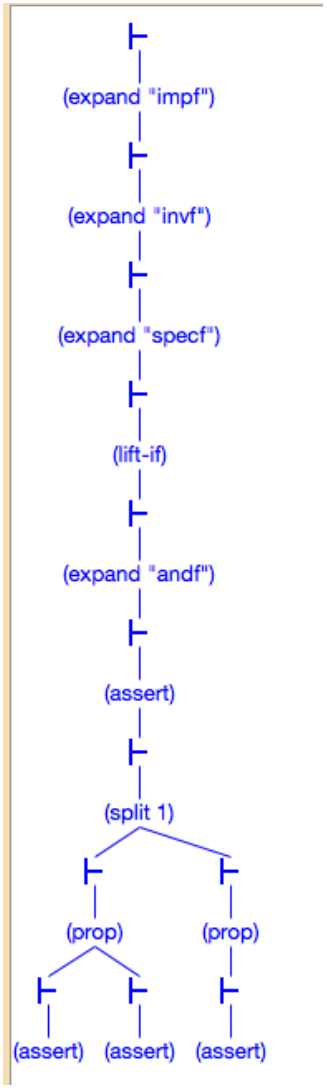
Nand Gate Specification: Functional

subrange(i,j): TYPE $= \{k : \mathbb{Z} | i \leq k \leq j\}$

```
nand : THEORY
BEGIN
  BIT: TYPE = subrange(0,1)
...

  impf(x:BIT,y:BIT): BIT =
    invf(andf(x,y))

  specf(x:BIT, y:BIT): BIT =
    COND
      x = 0 OR y = 0 -> 1,
      ELSE           -> 0
    ENDCOND

  verify_functional_implementation: CONJECTURE
    impf(x,y) = specf(x,y)
```


(expand "impf")
(expand "invf")
(expand "specf")
(lift-if)
(expand "andf")
(assert)
(split 1)
(prop)
(prop)
(assert)
(assert)
(assert)

Proof Tree



Note

M-x x-show-proof brings up a figure like the one on the left.

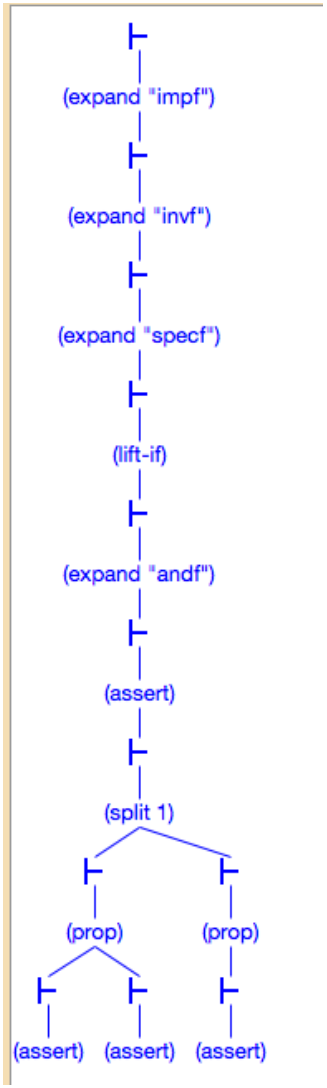
Question

How do you read the proof tree, starting at the (a) Top or (b) the Bottom? Where are the Axioms? What does the turnstyle represent?

Question

Proof rules transform one sequent into another. How do you interpret the proof rules in the figure?

Proof Tree



Note

M-x x-show-proof brings up a figure like the one on the left.

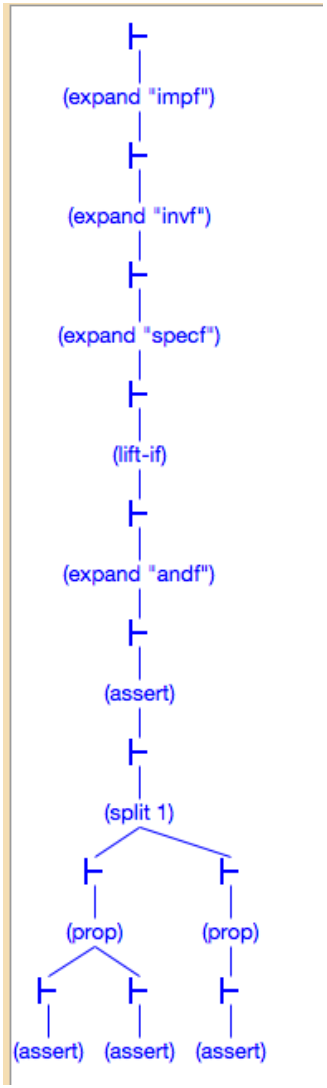
Question

How do you read the proof tree, starting at the (a) Top or (b) the Bottom? Where are the Axioms? What does the turnstyle represent?

Question

Proof rules transform one sequent into another. How do you interpret the proof rules in the figure?

Proof Tree



Note

M-x x-show-proof brings up a figure like the one on the left.

Question

How do you read the proof tree, starting at the (a) Top or (b) the Bottom? Where are the Axioms? What does the turnstyle represent?

Question

Proof rules transform one sequent into another. How do you interpret the proof rules in the figure?

Nand Gate Specification: Relational 1


```
a,b,c,x,y,z: VAR BIT

andr(x,y,z): bool =
  z = (x = 1 AND y = 1)
invr(x,z): bool =
  z = (1 - x)

imp_nandr(a,b,c): bool =
  (EXISTS (v:BIT) : andr(a,b,v) AND invr(v,c))

spec_nandr(a,b,c): bool =
  (c = 0) <=> (a=1 AND b = 1)

verify_relational_implementation: CONJECTURE
  imp_nandr(a,b,c) => spec_nandr(a,b,c)
```

Nand Gate Specification: Relational 1


```
a,b,c,x,y,z: VAR BIT

andr(x,y,z): bool =
  z = (x = 1 AND y = 1)
invr(x,z): bool =
  z = (1 - x)

imp_nandr(a,b,c): bool =
  (EXISTS (v:BIT) : andr(a,b,v) AND invr(v,c))

spec_nandr(a,b,c): bool =
  (c = 0) <=> (a=1 AND b = 1)

verify_relational_implementation: CONJECTURE
  imp_nandr(a,b,c) => spec_nandr(a,b,c)
```

Nand Gate Specification: Relational 1


```
a,b,c,x,y,z: VAR BIT

andr(x,y,z): bool =
  z = (x = 1 AND y = 1)
invr(x,z): bool =
  z = (1 - x)

imp_nandr(a,b,c): bool =
  (EXISTS (v:BIT) : andr(a,b,v) AND invr(v,c))

spec_nandr(a,b,c): bool =
  (c = 0) <=> (a=1 AND b = 1)

verify_relational_implementation: CONJECTURE
  imp_nandr(a,b,c) => spec_nandr(a,b,c)
```

Nand Gate Specification: Relational 1


```
a,b,c,x,y,z: VAR BIT

andr(x,y,z): bool =
  z = (x = 1 AND y = 1)
invr(x,z): bool =
  z = (1 - x)

imp_nandr(a,b,c): bool =
  (EXISTS (v:BIT) : andr(a,b,v) AND invr(v,c))

spec_nandr(a,b,c): bool =
  (c = 0) <=> (a=1 AND b = 1)

verify_relational_implementation: CONJECTURE
  imp_nandr(a,b,c) => spec_nandr(a,b,c)
```

Nand Gate Specification: Relational 1


```
a,b,c,x,y,z: VAR BIT

andr(x,y,z): bool =
  z = (x = 1 AND y = 1)
invr(x,z): bool =
  z = (1 - x)

imp_nandr(a,b,c): bool =
  (EXISTS (v:BIT) : andr(a,b,v) AND invr(v,c))

spec_nandr(a,b,c): bool =
  (c = 0) <=> (a=1 AND b = 1)

verify_relational_implementation: CONJECTURE
  imp_nandr(a,b,c) => spec_nandr(a,b,c)
```

Nand Gate Specification: Relational 2


```
a,b,c,x,y,z: VAR BIT

andr(x,y,z): bool =
  z = (x = 1 AND y = 1)
invr(x,z): bool =
  z = (1 - x)

imp_nandr(a,b,c): bool =
  (EXISTS (v:BIT) : andr(a,b,v) AND invr(v,c))
```

Note

The dummy variable v allows us to wire the output of the AND gate to the input of the INVERTER.
The Existential Quantifier allows us to hide the dummy variable. Thus the only free variables of `imp_nandr` are the inputs a and b and the output c.

Nand Gate Specification: Relational 2


```
a,b,c,x,y,z: VAR BIT

andr(x,y,z): bool =
  z = (x = 1 AND y = 1)
invr(x,z): bool =
  z = (1 - x)

imp_nandr(a,b,c): bool =
  (EXISTS (v:BIT) : andr(a,b,v) AND invr(v,c))
```

Note

The dummy variable v allows us to wire the output of the AND gate to the input of the INVERTER.
The Existential Quantifier allows us to hide the dummy variable. Thus the only free variables of `imp_nandr` are the inputs a and b and the output c.

Nand Gate Specification: Relational 2

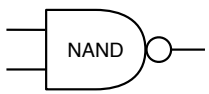

```
a,b,c,x,y,z: VAR BIT

andr(x,y,z): bool =
  z = (x = 1 AND y = 1)
invr(x,z): bool =
  z = (1 - x)

imp_nandr(a,b,c): bool =
  (EXISTS (v:BIT) : andr(a,b,v) AND invr(v,c))
```

Note

The dummy variable v allows us to wire the output of the AND gate to the input of the INVERTER.
The Existential Quantifier allows us to hide the dummy variable. Thus the only free variables of `imp_nandr` are the inputs a and b and the output c.

Complete and Disjoint Specifications

NAND gate Specification
(Complete and Disjoint Function Table)



NAND GATE

Input		Output
A	B	NAND
0	0	1
0	1	1
1	0	1
1	1	0

Question 1

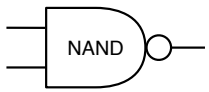
What does it mean for a function table to be (a) Complete and (b) Disjoint?

Question 2

What is the significance of (a) Completeness and (b) Disjointness?

Complete and Disjoint Specifications

NAND gate Specification
(Complete and Disjoint Function Table)



NAND GATE

Input		Output
A	B	NAND
0	0	1
0	1	1
1	0	1
1	1	0

Question 1

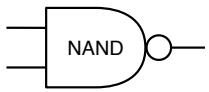
What does it mean for a function table to be (a) Complete and (b) Disjoint?

Question 2

What is the significance of (a) Completeness and (b) Disjointness?

Complete and Disjoint Specifications

NAND gate Specification
(Complete and Disjoint Function Table)



NAND GATE

Input		Output
A	B	NAND
0	0	1
0	1	1
1	0	1
1	1	0

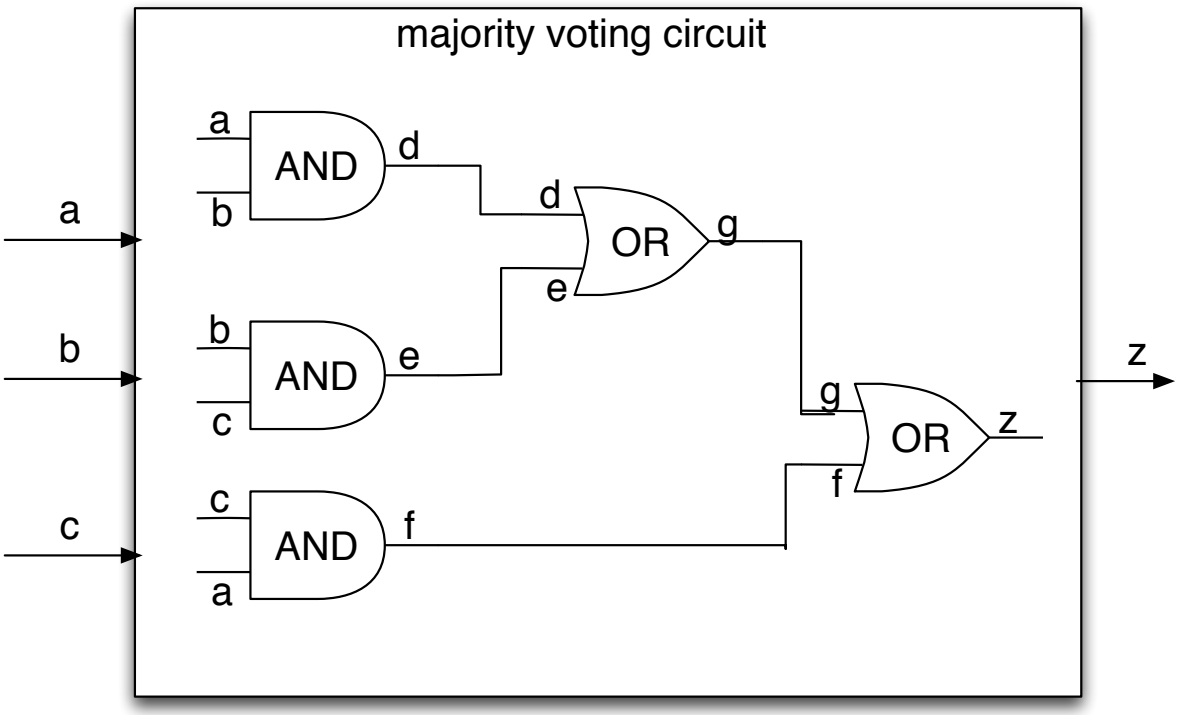
Question 1

What does it mean for a function table to be (a) Complete and (b) Disjoint?

Question 2

What is the significance of (a) Completeness and (b) Disjointness?

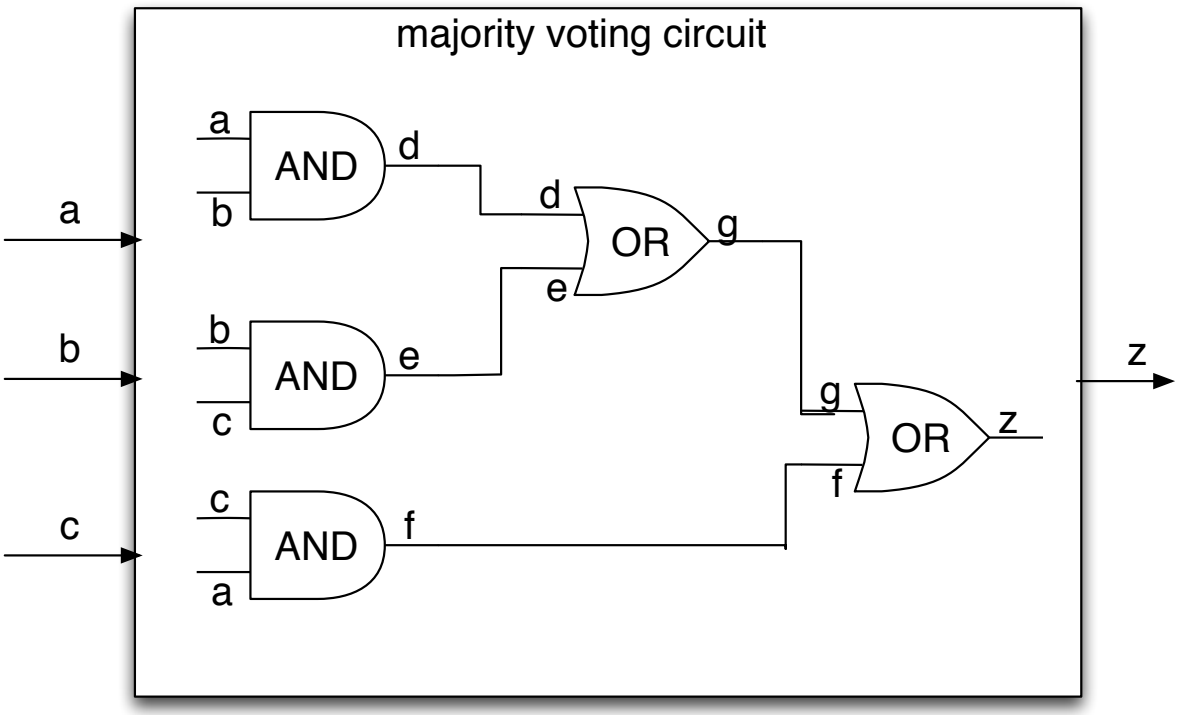
Majority Voting Circuit Specification



Question

What is a complete and disjoint function table for the majority Voting Circuit?

Majority Voting Circuit Specification



Question

What is a complete and disjoint function table for the majority Voting Circuit?

Majority Voting Circuit Specification


```
majority_vote: THEORY
BEGIN
  BIT: TYPE = subrange(0,1) CONTAINING 0
    % what would happen without "containing 0"?

  a,b,c,z : BIT % input and output

  % Majority Vote Specification
  spec: bool =
    z = 1 <=> (a + b + c >= 2)
```

Majority Voting Circuit Implementation


```
andGate(v,w,x: BIT): bool =    % define and_gate
      x=1 IFF (v=1 AND w=1)

orGate(v,w,x:BIT): bool =       % define or_gate
      x=1 IFF (v=1 OR w=1)

implementation: bool =          % implementation
      (exists (d,e,f,g: BIT):
          andGate(a,b,d)
      AND andGate(b,c,e)
      AND andGate(c,a,f)
      AND orGate(d,e,g)
      AND orGate(g,f,z)
      )
```

Validation


```
  implementation_correctness: CONJECTURE
    implementation
    IMPLIES
    spec

  % Check that antecedent of above is not false. Why?
  implementable: CONJECTURE
    a=1 AND b=1 AND c=1 AND z=1
    IMPLIES
    implementation /= False

END majority_vote
```

Tabular Specification

Complete and Disjoint Table

What is completeness? What is Disjointness?

Input	z
$a + b + c \geq 2$	1
$a + b + c < 2$	0

Assume $a, b, c \in \{0, 1\}$

Tabular Specification

Complete and Disjoint Table

What is completeness? What is Disjointness?

Input	z
$a + b + c \geq 2$	1
$a + b + c < 2$	0

Assume $a, b, c \in \{0, 1\}$

Tabular Specification

Complete and Disjoint Table

What is completeness? What is Disjointness?

Input	z
$a + b + c \geq 2$	1
$a + b + c < 2$	0

Assume $a, b, c \in \{0, 1\}$