Circuits

Predicate Logic and Specifying Circuits

EECS4312: Software Engineering Requirements

JSO. EECS. Lassonde

September 5, 2016

Table of contents

Circuits

EECS4312: Software Engineering

Nand Gate Functional Proof Tree Relational

Complete & Disjoint Specification

Majority Vote Specifaction Implementation Validation Tabular

1 Nand Gate

- Functional
- Proof Tree
- Relational
- **2** Complete & Disjoint Specifications
- 3 Majority Vote
 - Specifaction
 - Implementation
 - Validation
 - Tabular Specification

Nand Gate Specification

Circuits

EECS4312: Software Engineering

Nand Gate

Functional Proof Tree Relational

Complete & Disjoint Specifications

Majority Vote

Specifaction Implementation Validation Tabular NAND gate Specification (Complete and Disjoint Function Table)



NAND GATE

Input		Output
Α	В	NAND
0	0	1
0	1	1
1	0	1
1	1	0

Nand Gate Specification

Circuits

EECS4312: Software Engineering

Nand Gate

Functional Proof Tree Relational

Complete & Disjoint

Majority Vote
Specifaction
Implementation

Implementation
Validation
Tabular
Specification

NAND gate Specification (Complete and Disjoint Function Table)

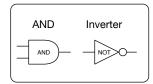


NAND GATE

Input		Output
Α	В	NAND
0	0	1
0	1	1
1	0	1
1	1	0

Implementation of NAND gate

using basic gates



```
Circuits
```

```
Functional
```

```
subrange(i,j): TYPE = \{k : \mathbb{Z} | i \leq k \leq j\}
nand: THEORY
BEGIN
  BIT: TYPE = subrange (0,1)
```

```
Circuits
```

EECS4312: Software Engineering Requirements

```
Nand Gate
Functional
Proof Tree
Relational
```

Complete & Disjoint Specifications

Majority Vote

Implementation
Validation

Validation Tabular

```
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 \rightarrow 1,
                  -> 0
       ELSE
     ENDCOND
```

Question

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

```
Circuits
```

```
Functional
```

```
subrange(i,j): TYPE = \{k : \mathbb{Z} | i \le k \le j\}
nand: THEORY
BEGIN
  BIT: TYPE = subrange (0,1)
  andf(x:BIT, y:BIT): BIT =
    COND
       x=1 AND y=1 \rightarrow 1,
                 -> 0
       ELSE
    ENDCOND
  invf(x:BIT): BIT = 1 - x
```

```
Circuits
```

Functional

```
subrange(i,j): TYPE = \{k : \mathbb{Z} | i \le k \le j\}
nand: THEORY
BEGIN
  BIT: TYPE = subrange (0,1)
  andf(x:BIT, y:BIT): BIT =
    COND
       x=1 AND y=1 \rightarrow 1,
       ELSE
                 -> 0
    ENDCOND
  invf(x:BIT): BIT = 1 - x
```

Question

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

```
Circuits

EECS4312:
Software
Engineering
```

Nand Gate Functional Proof Tree Relational

Complete & Disjoint Specifications

Majority Vote Specifaction Implementation Validation

```
subrange(i,i): TYPE = \{k : \mathbb{Z} | i < k < j\}
nand:
        THEORY
BEGIN
  BIT: TYPE = subrange (0,1)
```

```
Circuits
```

EECS4312: Software Engineering Requirements

Nand Gate Functional Proof Tree Relational

Complete & Disjoint Specifications

Majority Vote Specifaction Implementation Validation Tabular

```
subrange(i,i): TYPE = \{k : \mathbb{Z} | i < k < j\}
nand: THEORY
BEGIN
  BIT: TYPE = subrange (0,1)
. . .
  impf(x:BIT, y:BIT): BIT =
     invf(andf(x,y))
```

```
Circuits
```

EECS4312: Software Engineering Requirements

Nand Gate Functional Proof Tree Relational

Complete & Disjoint Specifications

Majority Vote Specifaction Implementation Validation Tabular

```
subrange(i,i): TYPE = \{k : \mathbb{Z} | i < k < 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,
                       -> 0
       ELSE
    ENDCOND
```

```
Circuits
```

EECS4312: Software Engineering Requirements

Nand Gate Functional Proof Tree Relational

Complete & Disjoint Specifications

Majority Vote Specifaction Implementation Validation Tabular

```
subrange(i,i): TYPE = \{k : \mathbb{Z} | i < k < 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,
                      -> 0
      ELSE
    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

Circuits

EECS4312: Software Engineering

Nand Gate Functional Proof Tree Relational

Complete & Disjoint Specification

Majority Vote Specifaction Implementation

(expand "impf" (expand "invf") (expand "specf") (expand "andf") (prop)

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

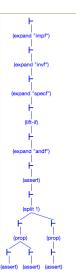
Circuits

EECS4312: Software Engineering Requirements

Nand Gate Functional Proof Tree Relational

Complete & Disjoint Specification

Majority Vote
Specifaction
Implementation
Validation
Tabular



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

Circuits

EECS4312: Software Engineering

Nand Gate Functional Proof Tree Relational

Complete & Disjoint Specification

Majority Vote Specifaction Implementation Validation Tabular Specification



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?

Circuits

EECS4312: Software Engineering

Functional Proof Tree

Complete & Disjoint

Disjoint Specification

Majority Vote Specifaction

Implementation Validation

Tabular Specification

```
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)
```

imp nandr(a.b.c) => spec nandr(a.b.c)

```
Circuits
```

Relational

```
a,b,c,x,y,z: VAR BIT
andr(x,v,z):bool =
  z = (x = 1 \text{ AND } y = 1)
invr(x,z): bool =
  z = (1 - x)
```

```
Circuits
```

EECS4312: Software Engineering

Nand Gate Functional Proof Tree Relational

Disjoint
Specification

Majority Vote Specifaction Implementation

Validation
Tabular

```
a,b,c,x,y,z: VAR BIT
andr(x, y, z): bool =
  z = (x = 1 \text{ 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))
```

Circuits

EECS4312: Software Engineering

Nand Gate Functional Proof Tree Relational

Complete & Disjoint Specifications

Majority Vote
Specifaction
Implementation
Validation

```
a,b,c,x,y,z: VAR BIT
andr(x, y, z): bool =
  z = (x = 1 \text{ 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) \iff (a=1 \text{ AND } b = 1)
```

```
Circuits
```

EECS4312: Software Engineering

Nand Gate Functional Proof Tree Relational

Complete & Disjoint Specification

Majority Vote
Specifaction
Implementation
Validation

```
a,b,c,x,y,z: VAR BIT
andr(x, y, z): bool =
  z = (x = 1 \text{ 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) \iff (a=1 \text{ AND } b = 1)
verify_relational_implementation: CONJECTURE
  imp_nandr(a,b,c) => spec_nandr(a,b,c)
```

```
Circuits
```

EECS4312: Software Engineering

Nand Gate Functional Proof Tree Relational

Complete & Disjoint Specification

Majority Vote

Implementation
Validation
Tabular

```
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 =
```

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.

Circuits

EECS4312: Software Engineering Requirement

Nand Gate Functional Proof Tree Relational

Complete & Disjoint Specification

Majority Voto

Implementation
Validation
Tabular

```
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.

Circuits

EECS4312: Software Engineering

Nand Gate Functional Proof Tree Relational

Complete & Disjoint Specification

Majority Vote
Specifaction
Implementation
Validation
Tabular
Specification

```
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 \boldsymbol{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

Circuits

EECS4312: Software Engineering

Nand Gate Functional Proof Tree Relational

Complete & Disjoint Specifications

Majority Vote Specifaction Implementation Validation Tabular

NAND gate Specification (Complete and Disjoint Function Table)



NAND GATE

Input		Output
Α	В	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

Circuits

EECS4312: Software Engineering

Nand Gate Functional Proof Tree Relational

Complete & Disjoint Specifications

Majority Vote
Specifaction
Implementation
Validation
Tabular

NAND gate Specification (Complete and Disjoint Function Table)



NAND GATE

Input		Output
Α	В	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

Circuits

EECS4312: Software Engineering

Nand Gate Functional Proof Tree Relational

Complete & Disjoint Specifications

Majority Vote
Specifaction
Implementation
Validation
Tabular
Specification

NAND gate Specification (Complete and Disjoint Function Table)



NAND GATE

Input		Output
Α	В	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

Circuits

EECS4312: Software Engineering

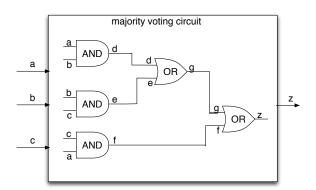
Nand Gate Functional Proof Tree

Complete & Disjoint Specification

Majority Vote

Specifaction Implementation Validation

Tabular Specification



Questior

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

Majority Voting Circuit Specification

Circuits

EECS4312: Software Engineering

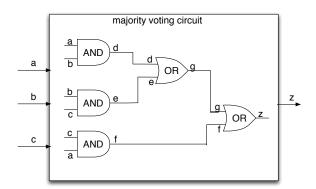
Nand Gate Functional Proof Tree Relational

Disjoint
Specification

Majority Vote

Specifaction Implementation Validation

Validation Tabular Specification



Question

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

Majority Voting Circuit Specification

```
Circuits
```

EECS4312: Software Engineering

Nand Gate Functional Proof Tree Relational

Complete & Disjoint Specifications

Majority Vote
Specifaction

Implementation Validation Tabular 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 =

 $z = 1 \ll (a + b + c \gg 2)$

Majority Voting Circuit Implementation

Circuits

EECS4312: Software Engineering

Nand Gate Functional Proof Tree Relational

Complete & Disjoint Specification:

Specifaction
Implementation
Validation

```
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 TFF (v=1 OR w=1)
implementation: bool =
                               % implementation
      (exists (d,e,f,g: BIT):
          andGate(a,b,d)
      AND and Gate (b, c, e)
      AND andGate(c,a,f)
      AND orGate(d,e,g)
      AND or Gate(q, f, z)
```

Validation

```
Circuits
```

EECS4312: Software Engineering Requirement

Nand Gate Functional Proof Tree Relational

Complete & Disjoint Specification

Majority Vote Specifaction

Implementation Validation Tabular

```
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

Circuits

EECS4312: Software Engineering

Nand Gate Functional

Relational
Complete &

Disjoint Specifications

Majority Vot

Implementation

Tabular Specification

Complete and Disjoint Table

What is completeness? What is Disjointness?

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

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

Tabular Specification

Circuits

EECS4312: Software Engineering

Nand Gate Functional Proof Tree Relational

Complete & Disjoint Specification

Specifaction Implementation Validation

Validation Tabular Specification

Complete and Disjoint Table

What is completeness? What is Disjointness?

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

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

Tabular Specification

Circuits

EECS4312: Software Engineering

Nand Gate Functional Proof Tree Relational

Complete & Disjoint Specification

Specifaction Implementation Validation

Validation
Tabular
Specification

Complete and Disjoint Table

What is completeness? What is Disjointness?

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

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