



Foundations of Computing I

Pre-Lecture Problem

Create a Boolean Algebra expression for the following truth table (for the function F):

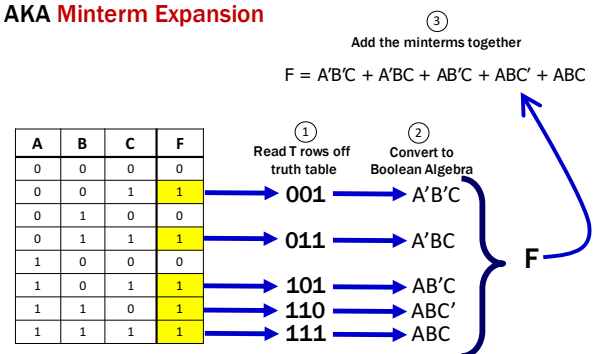
A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Canonical Forms

- Truth table is the **unique signature** of a Boolean Function
- The same truth table can have many gate realizations
 - We've seen this already
 - Depends on how good we are at Boolean simplification
- Canonical forms
 - Standard forms for a Boolean expression
 - We all come up with the same expression

Sum-of-Products Canonical Form

- AKA **Disjunctive Normal Form (DNF)**
- AKA **Minterm Expansion**



Sum-of-Products Canonical Form

Product term (or minterm)

- ANDed product of literals – input combination for which output is true
- each variable appears exactly once, true or inverted (but not both)

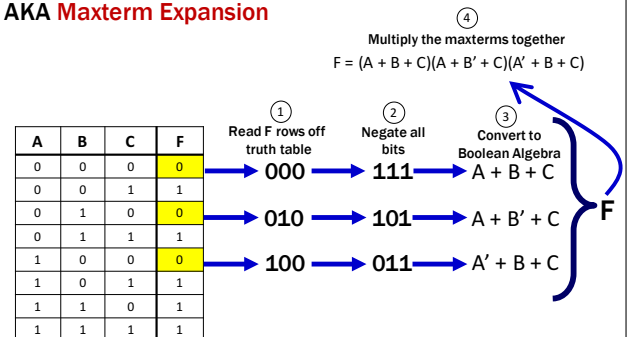
A	B	C	minterms
0	0	0	A'B'C'
0	0	1	A'B'C
0	1	0	A'BC'
0	1	1	A'BC
1	0	0	AB'C'
1	0	1	AB'C
1	1	0	ABC'
1	1	1	ABC

F in canonical form:
 $F(A, B, C) = A'B'C + A'BC + AB'C + ABC' + ABC$

canonical form ≠ minimal form
 $F(A, B, C) = A'B'C + A'BC + AB'C + ABC + ABC'$
 $= (A'B' + A'B + AB' + AB)C + ABC'$
 $= (A' + A)(B' + B)C + ABC'$
 $= C + ABC'$
 $= ABC' + C$
 $= AB + C$

Product-of-Sums Canonical Form

- AKA **Conjunctive Normal Form (CNF)**
- AKA **Maxterm Expansion**



Product-of-Sums: Why does this procedure work?

Useful Facts:

- We know $(F')' = F$
- We know how to get a **minterm** expansion for F'

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$F' = A'B'C' + A'BC' + AB'C'$$

Taking the complement of both sides...

$$(F')' = (A'B'C' + A'BC' + AB'C')'$$

And using DeMorgan/Comp....

$$F = (A'B'C')' (A'BC')' (AB'C')'$$

$$F = (A + B + C)(A + B' + C)(A' + B + C)$$

Gates Again!

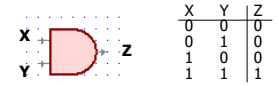
NOT

$$X' \quad \bar{X} \quad \neg X$$



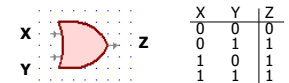
AND

$$X \cdot Y \quad XY \quad X \wedge Y$$



OR

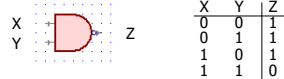
$$X + Y \quad X \vee Y$$



More Gates!

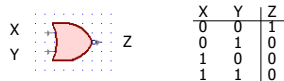
NAND

$$\neg(X \wedge Y) \quad (XY)'$$



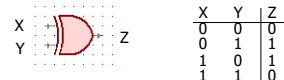
NOR

$$\neg(X \vee Y) \quad (X + Y)'$$



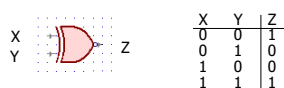
XOR

$$X \oplus Y$$



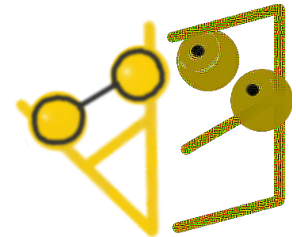
XNOR

$$X \leftrightarrow Y$$



CSE 311: Foundations of Computing

Lecture 6: Predicate Logic



Predicate Logic

• Propositional Logic

- If the tortoise walks at a rate of one node per step, and the hare walks at a rate of two nodes per step, ...

• Predicate Logic

- If the tortoise is on node x , and the hare is on node $2x$, then ...

Predicate Logic

• Propositional Logic

- Break down a statement into pieces

• Predicate Logic

- Relates pieces of a statement to each other

What is a "Predicate"?

A **predicate** is a **method (function)** with arguments that returns a **boolean**.

Examples:

- isPrime(x)
- isLessThan(x, y)
- hasSumOf(x, y, z)

We will not give "implementations" of predicates. Instead, we'll assumed they're already defined "the way we want".

Defining a Predicate

Cat(x) ::= "x is a cat"

Prime(x) ::= "x is prime"

HasTaken(x, y) ::= "student x has taken course y"

LessThan(x, y) ::= "x > y"

Sum(x, y, z) ::= "x + y = z"

GreaterThan5(x) ::= "x > 5"

HasNChars(s, n) ::= "string s has length n"

Notice that predicates can have varying numbers of arguments and input types.

Domain of Discourse

For ease of use, we define one "type"/"domain" that we work over. This set of objects is called the "**domain of discourse**".

For each of the following, what might the domain be?

(1) "x is a cat", "x barks", "x ruined my couch"

"mammals" or "sentient beings" or "cats and dogs" or ...

(2) "x is prime", "x = 0", "x < 0", "x is a power of two"

"numbers" or "integers" or "integers greater than 5" or ...

(3) "student x has taken course y" "x is a pre-req for z"

"students and courses" or "university entities" or ...

A Quick Note on "Variable Definition"

What's wrong here?

isEven(x) ::= "y is even"

The definition doesn't make sense, because y isn't defined. It's like writing the following code:

```
isEven(x) { return y % 2 == 0; }
```

Lessons:

- Be very careful with using "undefined variables"
- We need some way of introducing new variables...

Quantifiers

We use **quantifiers** to talk about collections of objects.

Universal Quantifier ("for all"): $\forall x P(x)$

P(x) is true for **every** x in the domain

read as "**for all x, P of x**"

Examples: Are these true? It depends on the domain. For example:

• $\forall x \text{ Odd}(x)$

{1, 3, -1, -27}	Integers	Odd Integers
True	False	True
True	False	False

• $\forall x \text{ LessThan5}(x)$

Universal Quantifier ("forall") (Programmatically)

$\forall x P(x)$

```
forallP(x) {  
  for (x : DOMAIN) {  
    if (!P(x)) {  
      return false;  
    }  
  }  
  return true;  
}
```

Quantifiers

We use **quantifiers** to talk about collections of objects.

Existential Quantifier ("exists"): $\exists x P(x)$

There is an x in the domain for which $P(x)$ is true
read as "**there exists x , P of x** "

Examples: Are these true? It depends on the domain. For example:

• $\exists x \text{ Odd}(x)$

• $\exists x \text{ LessThan5}(x)$

{1, 3, -1, -27}	Integers	Non-Zero Multiples of 10
True	True	False
True	True	False

Existential Quantifier ("exists") (Programmatically)

$\exists x P(x)$

```
existsP(x) {
  for (x : DOMAIN) {
    if (P(x)) {
      return true;
    }
  }
  return false;
}
```

Statements with Quantifiers

Just like with propositional logic, we need to define variables (this time **predicates**) before we do anything else. We must also now define a **domain of discourse** before doing anything else.

Domain of Discourse
Positive Integers

Predicate Definitions
Even(x) ::= "x is even" Greater(x, y) ::= "x > y"
Odd(x) ::= "x is odd" Equal(x, y) ::= "x = y"
Prime(x) ::= "x is prime" Sum(x, y, z) ::= "x + y = z"

Statements with Quantifiers (Literal Translations)

Domain of Discourse
Positive Integers

Predicate Definitions
Even(x) ::= "x is even" Greater(x, y) ::= "x > y"
Odd(x) ::= "x is odd" Equal(x, y) ::= "x = y"
Prime(x) ::= "x is prime" Sum(x, y, z) ::= "x + y = z"

Translate the following statements to English

$\forall x \exists y \text{ Greater}(y, x)$

"For every pos. int. x, there is a pos. int. y, such that $y > x$."

$\forall x \exists y \text{ Greater}(x, y)$

"For every pos. int. x, there is a pos. int. y, such that $x > y$."

$\forall x \exists y (\text{Greater}(y, x) \wedge \text{Prime}(y))$

"For every positive integer x, there is a pos. int. y such that $y > x$ and y is prime."

$\forall x (\text{Prime}(x) \rightarrow (\text{Equal}(x, 2) \vee \text{Odd}(x)))$

"For each pos. int. x, if x is prime, then $x = 2$ or x is odd."

$\exists x \exists y (\text{Sum}(x, 2, y) \wedge \text{Prime}(x) \wedge \text{Prime}(y))$

"There exist positive integers x and y such that $x + 2 = y$ and x and y are prime."

Statements with Quantifiers (Better Translations)

Domain of Discourse
Positive Integers

Predicate Definitions
Even(x) ::= "x is even" Greater(x, y) ::= "x > y"
Odd(x) ::= "x is odd" Equal(x, y) ::= "x = y"
Prime(x) ::= "x is prime" Sum(x, y, z) ::= "x + y = z"

Translate the following statements to English

$\forall x \exists y \text{ Greater}(y, x)$

"There is no greatest integer."

$\forall x \exists y \text{ Greater}(x, y)$

"There is no least integer."

$\forall x \exists y (\text{Greater}(y, x) \wedge \text{Prime}(y))$

"There is always a prime number greater than any positive integer."

$\forall x (\text{Prime}(x) \rightarrow (\text{Equal}(x, 2) \vee \text{Odd}(x)))$

"Every prime positive integer is either 2 or odd."

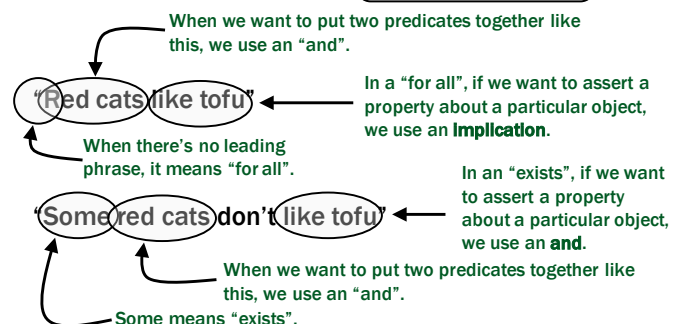
$\exists x \exists y (\text{Sum}(x, 2, y) \wedge \text{Prime}(x) \wedge \text{Prime}(y))$

"There exist prime positive integers that differ by two."

English to Predicate Logic

Domain of Discourse
Mammals

Predicate Definitions
Cat(x) ::= "x is a cat"
Red(x) ::= "x is red"
LikesTofu(x) ::= "x likes tofu"



English to Predicate Logic

Domain of Discourse
Mammals

Predicate Definitions
 Cat(x) ::= "x is a cat"
 Red(x) ::= "x is red"
 LikesTofu(x) ::= "x likes tofu"

"Red cats like tofu"

$$\forall x ((\text{Red}(x) \wedge \text{Cat}(x)) \rightarrow \text{LikesTofu}(x))$$

"Some red cats don't like tofu"

$$\exists y ((\text{Red}(y) \wedge \text{Cat}(y)) \wedge \neg \text{LikesTofu}(y))$$

Negations of Quantifiers

Predicate Definitions
 PurpleFruit(x) ::= "x is a purple fruit"

(*) $\forall x \text{ PurpleFruit}(x)$ ("All fruits are purple")

Some possible negations of (*):

- (a) "there exists a purple fruit"
- (b) "there exists a non-purple fruit"
- (c) "all fruits are not purple"

Try your intuition! Which one "feels" right?

Key Idea: In **every** domain, **exactly one** of a statement and its negation should be true.

Negations of Quantifiers

Predicate Definitions
 PurpleFruit(x) ::= "x is a purple fruit"

(*) $\forall x \text{ PurpleFruit}(x)$ ("All fruits are purple")

Some possible negations of (*):

- (a) "there exists a purple fruit"
- (b) "there exists a non-purple fruit"
- (c) "all fruits are not purple"

Key Idea: In **every** domain, **exactly one** of a statement and its negation should be true.

Domain of Discourse
{plum}

(*), (a)

Domain of Discourse
{apple}

(b), (c)

Domain of Discourse
{plum, apple}

(a), (b)

The only choice that ensures exactly one of the statement and its negation is (b).

De Morgan's Laws for Quantifiers

$$\begin{aligned} \neg \forall x P(x) &\equiv \exists x \neg P(x) \\ \neg \exists x P(x) &\equiv \forall x \neg P(x) \end{aligned}$$

"There is no largest integer"

$$\begin{aligned} \forall x (\neg(\exists y(x \geq y))) &\equiv \forall x (\exists y(\neg(x \geq y))) \\ &\equiv \forall x (\exists y(x < y)) \end{aligned}$$

"For every integer there is a larger integer"

Negations of Quantifiers

- not every positive integer is prime
- some positive integer is not prime
- prime numbers do not exist
- every positive integer is not prime

Scope of Quantifiers

Example: $\text{NotLargest}(x) \equiv \exists y \text{ Greater}(y, x)$
 $\equiv \exists z \text{ Greater}(z, x)$

truth value:

doesn't depend on y or z "bound variables"
 does depend on x "free variable"

quantifiers only act on free variables of the formula they quantify

$$\forall x (\exists y (P(x, y) \rightarrow \forall x Q(y, x)))$$

Scope of Quantifiers

$\exists x (P(x) \wedge Q(x))$ vs. $\exists x P(x) \wedge \exists x Q(x)$

This one asserts P
and Q of the *same* x.

This one asserts P and Q
of potentially different x's.