COS 140: Foundations of Computer Science

Boolean Algebra

Fall 2018

Computer
Science
Foundations

Homework and announcements!

- Reading: Chapter 5
- Homework: exercises 1–6
  - Exercise 7 for extra credit
  - Due: 1 week + 1 class from today (i.e., 9/21)
- Don't forget – **recitation this week**!
- Slides: online now

Computer Science Foundations

Problem

- Computers compute digital logic functions
- Need some way to describe those functions
- For some ordinary mathematical functions, algebra works well
- But the functions we want aren't numeric, but give true/false (1/0) values
- What can we use?

Computer Science Foundations

Copyright © 2002–2018 UMaine School of Computing and Information Science

Boolean algebra

- Boolean algebra is analogous to “regular” algebra, but for true/false values
- Well-known, well-developed mathematical foundation for digital logic.
- Provides a formalism for specifying the functions that we wish to have performed.
- Provides a mechanism for proving circuits are equivalent.
- Named for George Boole, a 19^{th}-century mathematician.

Computer Science Foundations

Copyright © 2002–2018 UMaine School of Computing and Information Science

Operators

Boolean operators correspond to gates and have same truth tables as corresponding gate.

Computer Science Foundations

Copyright © 2002–2018 UMaine School of Computing and Information Science

Operators

Boolean operators correspond to gates and have same truth tables as corresponding gate.

- NOT: NOT A, $\neg A$, $\overline{A}$, A'

Computer Science Foundations

Copyright © 2002–2018 UMaine School of Computing and Information Science

Operators

Boolean operators correspond to gates and have same truth tables as corresponding gate.

- NOT: NOT $A,\ \neg A,\ \overline{A},\ A'$
- AND: A AND $B,\ A \cdot B,\ AB,\ A \wedge B$

Computer Science Foundations

Operators

Boolean operators correspond to gates and have same truth tables as corresponding gate.

- NOT: NOT A, $\neg A$, $\overline{A}$, A'
- AND: A AND B, $A \cdot B$, AB, $A \wedge B$
- OR: A OR B, $A + B$, $A \vee B$

Computer Science Foundations

Operators

Boolean operators correspond to gates and have same truth tables as corresponding gate.

- NOT: NOT A, $\neg A$, $\overline{A}$, A'
- AND: A AND B, $A \cdot B$, AB, $A \wedge B$
- OR: A OR B, $A + B$, $A \vee B$
- NAND: A NAND B, $A|B$, $A \uparrow B$, $\overline{AB}$

Computer Science Foundations

Operators

Boolean operators correspond to gates and have same truth tables as corresponding gate.

- NOT: NOT A, $\neg A$, $\overline{A}$, A'
- AND: A AND B, $A \cdot B$, AB, $A \wedge B$
- OR: A OR B, $A + B$, $A \vee B$
- NAND: A NAND B, $A|B$, $A \uparrow B$, $\overline{AB}$
- NOR: A NOR B, $A \downarrow B$, $\overline{(A + B)}$

Copyright © 2002–2018 UMaine School of Computing and Information Science

Operators

Boolean operators correspond to gates and have same truth tables as corresponding gate.

- NOT: NOT $A,\ \neg A,\ \overline{A},\ A'$
- AND: A AND $B,\ A \cdot B,\ AB,\ A \wedge B$
- OR: A OR $B,\ A + B,\ A \vee B$
- NAND: A NAND $B,\ A|B,\ A \uparrow B,\ \overline{AB}$
- NOR: A NOR $B,\ A \downarrow B,\ \overline{(A + B)}$
- XOR: A XOR $B,\ A \oplus B$

Computer Science Foundations

Boolean Algebra Expressions

- What is an *expression*?
- Value of an expression $\Leftarrow$ depends on the values of the variables
- Evaluate expression: assign values to variables, performing the operations
- Can create an expression for a function by determining when the function should be 1, then writing an expression that is 1 in only those cases.

Computer Science Foundations

Boolean Algebra Expressions

- What is an *expression*?
- Value of an expression $\Leftarrow$ depends on the values of the variables
- Evaluate expression: assign values to variables, performing the operations
- Can create an expression for a function by determining when the function should be 1, then writing an expression that is 1 in only those cases.

Example: Create a 3-variable expression that equals 1 when all of the inputs are 1, or when one, and only one, input is 0.

Copyright © 2002–2018 UMaine School of Computing and Information Science

Boolean Algebra Expressions

- What is an *expression*?
- Value of an expression $\Leftarrow$ depends on the values of the variables
- Evaluate expression: assign values to variables, performing the operations
- Can create an expression for a function by determining when the function should be 1, then writing an expression that is 1 in only those cases.

Example: Create a 3-variable expression that equals 1 when all of the inputs are 1, or when one, and only one, input is 0.

$$ABC + \overline{A}BC + A\overline{B}C + AB\overline{C}$$

Computer Science Foundations

Copyright © 2002–2018 UMaine School of Computing and Information Science

Operator Precedence for Boolean Algebra

Computer Science Foundations

Operator Precedence for Boolean Algebra

- Subexpressions inside of parentheses, beginning with innermost parentheses.
- NOT
- AND
- OR
- Evaluate A NAND B as NOT($A \wedge B$) and A NOR B as NOT($A \vee B$)
- Evaluate subexpressions of equal precedence from left to right.

Computer Science Foundations

Proofs of Equivalence

- Two ways: truth tables and algebraic substitution
- Truth table method: if truth table for expression A same as for expression B, then $A \equiv B$
- Algebraic substitution method:
  - Use laws of Boolean algebra to transform one expression into the other
  - Proofs have to be convincing to others
  - Have to provide enough detail to show how one step follows from another
  - Have to provide justification for each step

Copyright © 2002–2018 UMaine School of Computing and Information Science

Proof by Truth Table

Best to have column for each input and results of each operator

Computer Science Foundations

Proof by Truth Table

Best to have column for each input and results of each operator

Prove that $A \oplus B$ is equivalent to $(\overline{A}B) + (A\overline{B})$

Computer Science Foundations

Proof by Truth Table

Best to have column for each input and results of each operator

Prove that $A \oplus B$ is equivalent to $(\overline{A}B) + (A\overline{B})$

A	B	$A \oplus B$	$\overline{A}$	$\overline{B}$	$\overline{A}B$	$A\overline{B}$	$(\overline{A}B) + (A\overline{B})$
0	0	0	1	1	0	0	0
0	1	1	1	0	1	0	1
1	0	1	0	1	0	1	1
1	1	0	0	0	0	0	0

Computer Science Foundations

Proof by Algebraic Substitution

- If you are trying to prove that expression 1 is equivalent to expression 2:
  - Start with one of the expressions, let's say 1.
  - Change it into another expression (say $1'$) using an *identity postulate* (law).
  - Continue the process with $1'$ until you arrive at 2.
- You must justify *every* change to the current expression by listing the identity postulate used.
- Some identity postulates: double negation law, identity law, null law, idempotent law, and inverse law

Copyright © 2002–2018 UMaine School of Computing and Information Science

Double Negation Law

- $A = \overline{\overline{A}}$

Computer Science Foundations

Identity Law

- AND form: $1A = A$
- OR form: $0 + A = A$

Computer Science Foundations

Copyright © 2002–2018 UMaine School of Computing and Information Science

Null Law

- AND form: $0A = 0$
- OR form: $1 + A = 1$

Computer Science Foundations

Idempotent Law

- AND form: $AA = A$
- OR form: $A + A = A$

Computer Science Foundations

Copyright © 2002–2018 UMaine School of Computing and Information Science

Inverse Law

- AND form: $A\overline{A} = 0$
- OR form: $A + \overline{A} = 1$

Computer Science Foundations

Example: Finding Equivalent Circuits

Starting with the following expression, find the equivalent expression that uses the least number of gates (has the smallest number of boolean operators) and the least number of inputs

$$(A + \overline{A})\overline{(B\overline{\overline{B}})}$$

Computer Science Foundations

Computer Science Foundations

Example: Finding Equivalent Circuits

Starting with the following expression, find the equivalent expression that uses the least number of gates (has the smallest number of boolean operators) and the least number of inputs

$$(A + \overline{A})\overline{(B\overline{\overline{B}})}$$

- Don't know what the final result will be so use algebraic substitution instead of truth tables

Example: Finding Equivalent Circuits

Starting with the following expression, find the equivalent expression that uses the least number of gates (has the smallest number of boolean operators) and the least number of inputs

$$(A + \overline{A})\overline{(B\overline{\overline{B}})}$$

- Don't know what the final result will be so use algebraic substitution instead of truth tables
- Follow your intuition about what should be the case and what makes sense, then justify with a law.
- In other words, have a plan based on what makes sense.
- Sometimes you need to try things to make the plan.

Computer Science Foundations

Example: Finding Equivalent Circuits

$$(A + \overline{A})\overline{(B\overline{\overline{B}})}$$

Computer Science Foundations

Copyright © 2002–2018 UMaine School of Computing and Information Science

Introduction
Proofs
Laws
- Double Negation Law
- Identity Law
- Null Law
- Idempotent Law
- Inverse Law
- Example
- Commutative Law
- Associative Law
- Distributive Law
- Absorption Law
- DeMorgan's Law

Using the Laws

Example: Finding Equivalent Circuits

$$(A + \overline{A})\overline{(B\overline{\overline{B}})}$$

$$(A + \overline{A})\overline{(BB)}$$ *Double Negation Law*

Example: Finding Equivalent Circuits

$$(A + \overline{A})\overline{(B\overline{\overline{B}})}$$

$$(A + \overline{A})\overline{(BB)}$$ *Double Negation Law*

$$(A + \overline{A})\overline{B}$$ *Idempotent Law*

Computer Science Foundations

Example: Finding Equivalent Circuits

$(A + \overline{A})\overline{(B\overline{\overline{B}})}$	
$(A + \overline{A})\overline{(BB)}$	*Double Negation Law*
$(A + \overline{A})\overline{B}$	*Idempotent Law*
$1\overline{B}$	*Inverse Law*

Computer Science Foundations

Copyright © 2002–2018 UMaine School of Computing and Information Science

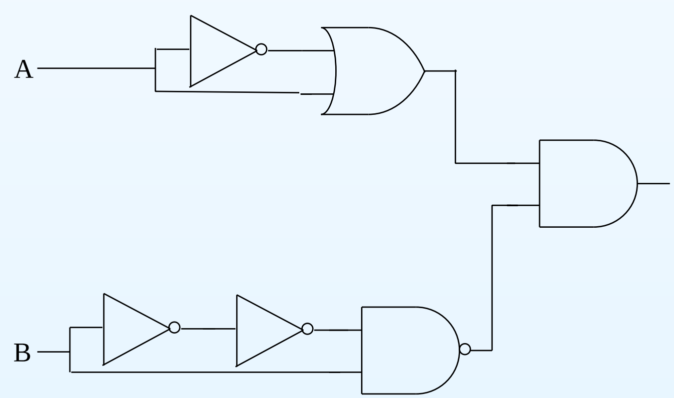
Example: Finding Equivalent Circuits

$(A + \overline{A})\overline{(B\overline{\overline{B}})}$	
$(A + \overline{A})\overline{(BB)}$	*Double Negation Law*
$(A + \overline{A})\overline{B}$	*Idempotent Law*
$1\overline{B}$	*Inverse Law*
$\overline{B}$	*Identity Law*

Copyright © 2002–2018 UMaine School of Computing and Information Science

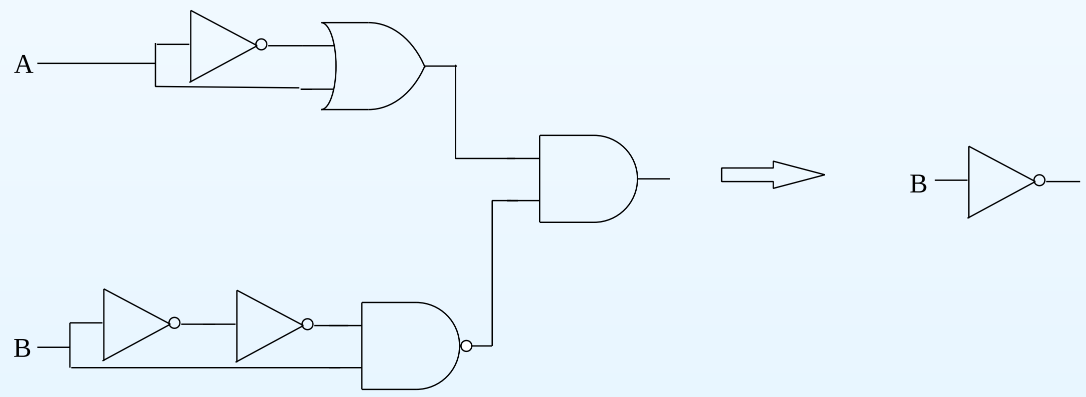
Example: Finding Equivalent Circuits

$(A + \overline{A})\overline{(B\overline{\overline{B}})}$	
$(A + \overline{A})\overline{(BB)}$	*Double Negation Law*
$(A + \overline{A})\overline{B}$	*Idempotent Law*
$1\overline{B}$	*Inverse Law*
$\overline{B}$	*Identity Law*



Example: Finding Equivalent Circuits

$(A + \overline{A})\overline{(B\overline{\overline{B}})}$	
$(A + \overline{A})\overline{(BB)}$	*Double Negation Law*
$(A + \overline{A})\overline{B}$	*Idempotent Law*
$1\overline{B}$	*Inverse Law*
$\overline{B}$	*Identity Law*



Computer Science Foundations

Commutative Law

- AND form: $AB = BA$
- OR form: $A + B = B + A$

Computer Science Foundations

Associative Law

- AND form: $(AB)C = A(BC)$
- OR form: $(A + B) + C = A + (B + C)$

Computer Science Foundations

Copyright © 2002–2018 UMaine School of Computing and Information Science

Distributive Law

- AND form: $A(B + C) = AB + AC$

Computer Science Foundations

Distributive Law

- AND form: $A(B + C) = AB + AC$
- OR form: $A + BC = (A + B)(A + C)$

Computer Science Foundations

Copyright © 2002–2018 UMaine School of Computing and Information Science

Distributive Law

- AND form: $A(B + C) = AB + AC$
- OR form: $A + BC = (A + B)(A + C)$
- ...or $A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$

Computer Science Foundations

Copyright © 2002–2018 UMaine School of Computing and Information Science

Absorption Law

- AND form: $A(A + B) = A$
- OR form: $A + AB = A$

Computer Science Foundations

Copyright © 2002–2018 UMaine School of Computing and Information Science

DeMorgan's Law

- AND form: $\overline{AB} = \overline{A} + \overline{B}$
- OR form: $\overline{A + B} = \overline{A} \cdot \overline{B}$

Computer Science Foundations

DeMorgan's Law

- AND form: $\overline{AB} = \overline{A} + \overline{B}$
- OR form: $\overline{A+B} = \overline{A} \cdot \overline{B}$

Aside: recall that $\overline{AB} = A$ NAND B, and $\overline{A+B} = A$ NOR B

Computer Science Foundations

DeMorgan's Law

- AND form: $\overline{AB} = \overline{A} + \overline{B}$
- OR form: $\overline{A + B} = \overline{A} \cdot \overline{B}$

Aside: recall that $\overline{AB} = A$ NAND B, and $\overline{A+B} = A$ NOR B

NOTE:

$$\overline{AB} \neq \overline{A}\,\overline{B}$$

$$\overline{A+B} \neq \overline{A} + \overline{B}$$

Computer Science Foundations

How would you prove DeMorgan's Law?

What approach would you use? Why?

Copyright © 2002–2018 UMaine School of Computing and Information Science

Example: Proof by Algebraic Substitution

Introduction
Proofs
Laws
Using the Laws
- Proving DeMorgan's Law
- Proof 1
- Proof 1a
- Proof 1b
- Proof 2
- Proof 3

Prove $\overline{A}B + A\overline{B} = \overline{(\overline{\overline{A}B})(\overline{A\overline{B}})}$

Copyright © 2002–2018 UMaine School of Computing and Information Science

Example: Proof by Algebraic Substitution

Prove $\overline{A}B + A\overline{B} = \overline{(\overline{\overline{A}B})(\overline{A\overline{B}})}$

- $\overline{(\overline{\overline{A}B})(\overline{A\overline{B}})}$

Copyright © 2002–2018 UMaine School of Computing and Information Science

Example: Proof by Algebraic Substitution

Prove $\overline{A}B \; + \; A\overline{B} \; = \; \overline{(\overline{\overline{A}B})(\overline{A\overline{B}})}$

- $\overline{(\overline{\overline{A}B})(\overline{A\overline{B}})}$
- $\overline{\overline{(\overline{A}B)}} + \overline{\overline{(A\overline{B})}}$ *DeMorgan's Law*

Example: Proof by Algebraic Substitution

Prove $\overline{A}B + A\overline{B} = \overline{(\overline{\overline{A}B})(\overline{A\overline{B}})}$

- $\overline{(\overline{\overline{A}B})(\overline{A\overline{B}})}$
- $\overline{\overline{(\overline{A}B)}} + \overline{\overline{(A\overline{B})}}$ *DeMorgan's Law*
- $(\overline{A}B) + \overline{\overline{(A\overline{B})}}$ *Double Negation*

Computer Science Foundations

Example: Proof by Algebraic Substitution

Prove $\overline{A}B \; + \; A\overline{B} \; = \; \overline{(\overline{\overline{A}B})(\overline{A\overline{B}})}$

- $\overline{(\overline{\overline{A}B})(\overline{A\overline{B}})}$
- $\overline{\overline{(\overline{A}B)}} + \overline{\overline{(A\overline{B})}}$ *DeMorgan's Law*
- $(\overline{A}B) + \overline{\overline{(A\overline{B})}}$ *Double Negation*
- $(\overline{A}B) + (A\overline{B})$ *Double Negation*

Computer Science Foundations

Copyright © 2002–2018 UMaine School of Computing and Information Science

Example: Proof by Algebraic Substitution

Prove $\overline{A}B + A\overline{B} = \overline{(\overline{\overline{A}B})(\overline{A\overline{B}})}$

- $\overline{(\overline{\overline{A}B})(\overline{A\overline{B}})}$
- $\overline{\overline{(\overline{A}B)}} + \overline{\overline{(A\overline{B})}}$ *DeMorgan's Law*
- $(\overline{A}B) + \overline{\overline{(A\overline{B})}}$ *Double Negation*
- $(\overline{A}B) + (A\overline{B})$ *Double Negation*
- $\overline{A}B + A\overline{B}$ *Def. of parentheses/precedence*

Copyright © 2002–2018 UMaine School of Computing and Information Science

Example: Proof by Algebraic Substitution

Prove $\overline{A}B + A\overline{B} = \overline{(\overline{\overline{A}B})(\overline{A\overline{B}})}$

Copyright © 2002–2018 UMaine School of Computing and Information Science

Example: Proof by Algebraic Substitution

Prove $\overline{A}B + A\overline{B} = \overline{(\overline{\overline{A}B})(\overline{A\overline{B}})}$

$$\overline{(\overline{\overline{A}B})(\overline{A\overline{B}})}$$

Computer Science Foundations

Example: Proof by Algebraic Substitution

Prove $\overline{A}B + A\overline{B} = \overline{(\overline{\overline{A}B})(\overline{A\overline{B}})}$

$\overline{(\overline{\overline{A}B})(\overline{A\overline{B}})}$

$\overline{XY}$ *Let* $X = (\overline{\overline{A}B}),\ Y = (\overline{A\overline{B}})$

Computer Science Foundations

Copyright © 2002–2018 UMaine School of Computing and Information Science

Example: Proof by Algebraic Substitution

Prove $\overline{A}B + A\overline{B} = \overline{(\overline{\overline{A}B})(\overline{A\overline{B}})}$

$\overline{(\overline{\overline{A}B})(\overline{A\overline{B}})}$	
$\overline{XY}$	*Let* $X = (\overline{\overline{A}B})$, $Y = (\overline{A\overline{B}})$
$\overline{X} + \overline{Y}$	*DeMorgan's Law*

Computer Science Foundations

Example: Proof by Algebraic Substitution

Prove $\overline{A}B + A\overline{B} = \overline{(\overline{\overline{A}B})(\overline{A\overline{B}})}$

$\overline{(\overline{\overline{A}B})(\overline{A\overline{B}})}$	
$\overline{XY}$	*Let* $X = (\overline{\overline{A}B}),\ Y = (\overline{A\overline{B}})$
$\overline{X} + \overline{Y}$	*DeMorgan's Law*
$\overline{(\overline{\overline{A}B})} + \overline{(\overline{A\overline{B}})}$	*Substitution for* $X,\ Y$

Computer Science Foundations

Example: Proof by Algebraic Substitution

Prove $\overline{A}B + A\overline{B} = \overline{(\overline{\overline{A}B})(\overline{A\overline{B}})}$

$\overline{(\overline{\overline{A}B})(\overline{A\overline{B}})}$	
$\overline{XY}$	*Let* $X = (\overline{\overline{A}B}),\ Y = (\overline{A\overline{B}})$
$\overline{X} + \overline{Y}$	*DeMorgan's Law*
$\overline{\overline{(\overline{A}B)}} + \overline{\overline{(A\overline{B})}}$	*Substitution for* $X,\ Y$
$(\overline{A}B) + (A\overline{B})$	*Double Negation*

Computer Science Foundations

Example: Proof by Algebraic Substitution

Prove $\overline{A}B + A\overline{B} = \overline{\overline{(\overline{A}B)}\,\overline{(A\overline{B})}}$

$\overline{\overline{(\overline{A}B)}\,\overline{(A\overline{B})}}$	
$\overline{XY}$	*Let* $X = \overline{(\overline{A}B)},\ Y = \overline{(A\overline{B})}$
$\overline{X} + \overline{Y}$	*DeMorgan's Law*
$\overline{\overline{(\overline{A}B)}} + \overline{\overline{(A\overline{B})}}$	*Substitution for* $X,\ Y$
$(\overline{A}B) + \overline{\overline{(A\overline{B})}}$	*Double Negation*
$(\overline{A}B) + (A\overline{B})$	*Double Negation*

Computer Science Foundations

Example: Proof by Algebraic Substitution

Prove $\overline{A}B + A\overline{B} = \overline{(\overline{\overline{A}B})(\overline{A\overline{B}})}$

$\overline{(\overline{\overline{A}B})(\overline{A\overline{B}})}$	
$\overline{XY}$	*Let* $X = (\overline{\overline{A}B}),\ Y = (\overline{A\overline{B}})$
$\overline{X} + \overline{Y}$	*DeMorgan's Law*
$\overline{\overline{(\overline{A}B)}} + \overline{\overline{(A\overline{B})}}$	*Substitution for* $X,\ Y$
$(\overline{A}B) + \overline{\overline{(A\overline{B})}}$	*Double Negation*
$(\overline{A}B) + (A\overline{B})$	*Double Negation*
$\overline{A}B + A\overline{B}$	*Def. of parentheses/precedence*

Copyright © 2002–2018 UMaine School of Computing and Information Science

Example: Proof by Algebraic Substitution

Prove $\overline{A}B + A\overline{B} = \overline{(\overline{\overline{A}B})(\overline{A\overline{B}})}$

Copyright © 2002–2018 UMaine School of Computing and Information Science

Example: Proof by Algebraic Substitution

Prove $\overline{A}B + A\overline{B} = \overline{(\overline{\overline{A}B})(\overline{A\overline{B}})}$

- $\overline{A}B + A\overline{B}$

Introduction
Proofs
Laws
Using the Laws
- Proving DeMorgan's Law
- Proof 1
- Proof 1a
- Proof 1b
- Proof 2
- Proof 3

Example: Proof by Algebraic Substitution

Prove $\overline{A}B \;+\; A\overline{B} \;=\; \overline{(\overline{\overline{A}B})(\overline{A\overline{B}})}$

- $\overline{A}B \;+\; A\overline{B}$
- $\overline{\overline{\overline{A}B \;+\; A\overline{B}}}$ *Double Negation*

Copyright © 2002–2018 UMaine School of Computing and Information Science

Example: Proof by Algebraic Substitution

Prove $\overline{A}B \; + \; A\overline{B} \; = \; \overline{(\overline{\overline{A}B})(\overline{A\overline{B}})}$

- $\overline{A}B \; + \; A\overline{B}$
- $\overline{\overline{\overline{A}B \; + \; A\overline{B}}}$ *Double Negation*
- $\overline{(\overline{\overline{A}B})(\overline{A\overline{B}})}$ *DeMorgan's Law*

Computer Science Foundations

Example: Another Proof by Algebraic Substitution

Prove $ACB + C(B + C) = C$

Computer Science Foundations

Copyright © 2002–2018 UMaine School of Computing and Information Science

Example: Another Proof by Algebraic Substitution

Prove $ACB + C(B + C) = C$

- $ACB + C(B + C)$

Computer Science Foundations

Example: Another Proof by Algebraic Substitution

Prove $ACB + C(B + C) = C$

- $ACB + C(B + C)$
- $ACB + CB + CC$ *Distributive Law*

Computer Science Foundations

Copyright © 2002–2018 UMaine School of Computing and Information Science

Example: Another Proof by Algebraic Substitution

Prove $ACB + C(B + C) = C$

- $ACB + C(B + C)$
- $ACB + CB + CC$ *Distributive Law*
- $ACB + CB + C$ *Idempotent Law*

Computer Science Foundations

Example: Another Proof by Algebraic Substitution

Prove $ACB + C(B + C) = C$

- $ACB + C(B + C)$
- $ACB + CB + CC$ *Distributive Law*
- $ACB + CB + C$ *Idempotent Law*
- $CB + ACB + C$ *Commutative Law*

Computer Science Foundations

Computer Science Foundations

Example: Another Proof by Algebraic Substitution

Prove $ACB + C(B + C) = C$

- $ACB + C(B + C)$
- $ACB + CB + CC$ *Distributive Law*
- $ACB + CB + C$ *Idempotent Law*
- $CB + ACB + C$ *Commutative Law*
- $CB + CBA + C$ *Commutative Law*

Example: Another Proof by Algebraic Substitution

Prove $ACB + C(B + C) = C$

- $ACB + C(B + C)$
- $ACB + CB + CC$ *Distributive Law*
- $ACB + CB + C$ *Idempotent Law*
- $CB + ACB + C$ *Commutative Law*
- $CB + CBA + C$ *Commutative Law*
- $CB + C + CBA$ *Commutative Law*

Computer Science Foundations

Example: Another Proof by Algebraic Substitution

Prove $ACB + C(B + C) = C$

- $ACB + C(B + C)$
- $ACB + CB + CC$ *Distributive Law*
- $ACB + CB + C$ *Idempotent Law*
- $CB + ACB + C$ *Commutative Law*
- $CB + CBA + C$ *Commutative Law*
- $CB + C + CBA$ *Commutative Law*
- $CB + C$ *Absorption Law*

Computer Science Foundations

Example: Another Proof by Algebraic Substitution

Prove $ACB + C(B + C) = C$

- $ACB + C(B + C)$
- $ACB + CB + CC$ *Distributive Law*
- $ACB + CB + C$ *Idempotent Law*
- $CB + ACB + C$ *Commutative Law*
- $CB + CBA + C$ *Commutative Law*
- $CB + C + CBA$ *Commutative Law*
- $CB + C$ *Absorption Law*
- $C + CB$ *Commutative Law*

Computer Science Foundations

Example: Another Proof by Algebraic Substitution

Prove $ACB + C(B + C) = C$

- $ACB + C(B + C)$
- $ACB + CB + CC$ *Distributive Law*
- $ACB + CB + C$ *Idempotent Law*
- $CB + ACB + C$ *Commutative Law*
- $CB + CBA + C$ *Commutative Law*
- $CB + C + CBA$ *Commutative Law*
- $CB + C$ *Absorption Law*
- $C + CB$ *Commutative Law*
- C *Absorption Law*

Computer Science Foundations

Copyright © 2002–2018 UMaine School of Computing and Information Science

Example: Another Proof by Algebraic Substitution

Alternate proof of $ACB + C(B + C) = C$

Computer Science Foundations

Example: Another Proof by Algebraic Substitution

Alternate proof of $ACB + C(B + C) = C$

- $ACB + C(B + C)$

Computer Science Foundations

Example: Another Proof by Algebraic Substitution

Alternate proof of $ACB + C(B + C) = C$

- $ACB + C(B + C)$
- $ACB + CB + CC$ *Distributive Law*

Computer Science Foundations

Example: Another Proof by Algebraic Substitution

Alternate proof of $ACB + C(B + C) = C$

- $ACB + C(B + C)$
- $ACB + CB + CC$ *Distributive Law*
- $ACB + CB + C$ *Idempotent Law*

Computer Science Foundations

Copyright © 2002–2018 UMaine School of Computing and Information Science

Example: Another Proof by Algebraic Substitution

Alternate proof of $ACB + C(B + C) = C$

- $ACB + C(B + C)$
- $ACB + CB + CC$ *Distributive Law*
- $ACB + CB + C$ *Idempotent Law*
- $ABC + CB + 1C$ *Identity Law*

Computer Science Foundations

Copyright © 2002–2018 UMaine School of Computing and Information Science

Example: Another Proof by Algebraic Substitution

Alternate proof of $ACB + C(B + C) = C$

- $ACB + C(B + C)$
- $ACB + CB + CC$ *Distributive Law*
- $ACB + CB + C$ *Idempotent Law*
- $ABC + CB + 1C$ *Identity Law*
- $CAB + CB + C1$ *Commutative Law, twice*

Computer Science Foundations

Example: Another Proof by Algebraic Substitution

Alternate proof of $ACB + C(B + C) = C$

- $ACB + C(B + C)$
- $ACB + CB + CC$ *Distributive Law*
- $ACB + CB + C$ *Idempotent Law*
- $ABC + CB + 1C$ *Identity Law*
- $CAB + CB + C1$ *Commutative Law, twice*
- $C(AB + B + 1)$ *Distributive Law*

Computer Science Foundations

Example: Another Proof by Algebraic Substitution

Alternate proof of $ACB + C(B + C) = C$

- $ACB + C(B + C)$
- $ACB + CB + CC$ *Distributive Law*
- $ACB + CB + C$ *Idempotent Law*
- $ABC + CB + 1C$ *Identity Law*
- $CAB + CB + C1$ *Commutative Law, twice*
- $C(AB + B + 1)$ *Distributive Law*
- $C(1 + AB + B)$ *Commutative Law, twice*

Computer Science Foundations

Example: Another Proof by Algebraic Substitution

Alternate proof of $ACB + C(B + C) = C$

- $ACB + C(B + C)$
- $ACB + CB + CC$ *Distributive Law*
- $ACB + CB + C$ *Idempotent Law*
- $ABC + CB + 1C$ *Identity Law*
- $CAB + CB + C1$ *Commutative Law, twice*
- $C(AB + B + 1)$ *Distributive Law*
- $C(1 + AB + B)$ *Commutative Law, twice*
- $C1$ *Null Law*

Computer Science Foundations

Example: Another Proof by Algebraic Substitution

Alternate proof of $ACB + C(B + C) = C$

- $ACB + C(B + C)$
- $ACB + CB + CC$ *Distributive Law*
- $ACB + CB + C$ *Idempotent Law*
- $ABC + CB + 1C$ *Identity Law*
- $CAB + CB + C1$ *Commutative Law, twice*
- $C(AB + B + 1)$ *Distributive Law*
- $C(1 + AB + B)$ *Commutative Law, twice*
- $C1$ *Null Law*
- $1C$ *Commutative Law*

Computer Science Foundations

Copyright © 2002–2018 UMaine School of Computing and Information Science

Example: Another Proof by Algebraic Substitution

Alternate proof of $ACB + C(B + C) = C$

- $ACB + C(B + C)$
- $ACB + CB + CC$ *Distributive Law*
- $ACB + CB + C$ *Idempotent Law*
- $ABC + CB + 1C$ *Identity Law*
- $CAB + CB + C1$ *Commutative Law, twice*
- $C(AB + B + 1)$ *Distributive Law*
- $C(1 + AB + B)$ *Commutative Law, twice*
- $C1$ *Null Law*
- $1C$ *Commutative Law*
- C *Identity Law*

Computer Science Foundations

Example: Another Proof by Algebraic Substitution

Alternate proof of $ACB + C(B + C) = C$

- $ACB + C(B + C)$
- $ACB + CB + CC$ *Distributive Law*
- $ACB + CB + C$ *Idempotent Law*
- $ABC + CB + 1C$ *Identity Law*
- $CAB + CB + C1$ *Commutative Law, twice*
- $C(AB + B + 1)$ *Distributive Law*
- $C(1 + AB + B)$ *Commutative Law, twice*
- $C1$ *Null Law*
- $1C$ *Commutative Law*
- C *Identity Law*

$\Rightarrow$ Often more than one way to do proof!

Computer Science Foundations