COMSM1302
Overview of Computer Architecture

Lecture 2 – Propositional logic, Boolean algebra

In this lecture

Foundations	• Data representation, **logic.**
Building blocks	• Transistors, transistor based logic, simple devices, storage.
Modules	• Hex modules, memory, simple controller and processor.
Programming	• Assembly, assembler, language, compilation phases, boot-strapping.
Bigger systems	• ARM & Thumb, I/O, protecting shared systems, memory hierarchy, multi-processors, networks.
Wrap-up	• More examples, historical computers, contemporary systems.

The origins of logic

- Boole, **1840s**
  - Enabling the work of Shannon and others in digital logic circuits, for **communication and computation**.
- Shannon, 1948
  - See previous lecture, brief discussion of **bits** in information theory.

Propositional logic

- A statement that meets the following criteria
  - Can give a truth value (true or false), when evaluated
    - 1 bit result
  - Unambiguous
  - Can contain free variables

Propositional logic

- Which of the below statements are valid based on these rules?
  - Can give a truth value (true or false), when evaluated
  - Unambiguous
  - Can contain free variables

- The temperature is 20 degrees C
- The temperature is x degrees C
- It is too warm
- This statement is false
- There is more than x ml of milk in this jug

Propositional logic

- Which of the below statements are valid based on these rules?
  - Can give a truth value (true or false), when evaluated
  - Unambiguous
  - Can contain free variables

- The temperature is 20 degrees C
- The temperature is x degrees C
- ~~It is too warm~~
- ~~This statement is false~~
- There is more than x ml of milk in this jug

More examples?

Propositional logic

- Statements can be represented as short-hand functions/variables.

- f: The temperature is 20 degrees C
- g(x): The temperature is x degrees C
- h(x): There is more than x ml of milk in this jug

Propositional logic

- Statements can be combined with connectives, with brackets to clarify precedence, as necessary.

- f: The temperature is 20 degrees C
- ¬f: The temperature is **not** 20 degrees C
  not(The temperature is 20 degrees C)
- g(x) ∧ h(y): The temperature is x degrees C **and** there is more than y ml of milk in this jug

More formally

- With these connectives, complex logical statements can be assembled.

Natural	Formal	C
not x	$\neg x$	`!x`
x and y	$x \wedge y$	`x && y`
x or y	$x \vee y$	`x || y`
x or y but not x and y Exclusively x or y	$x \oplus y$	

Disappointing logic

- Let x be “You can have your cake”
- Let y be “You can eat your cake”
- What does the following statement mean?

$$x \oplus y$$

Implication and equivalence

- x → y — x implies y
  if x then y
  if there is greater than x ml of milk in the jug, then I have enough milk
- X ≡ y — x is equivalent to y
  x if and only if y
  x iff. Y
  it is raining, this must be Bristol…
  this is Bristol, it must be raining

Reviewing the connectives

- Natural language is flexible, but there are formal terms.
- Symbolism varies between subjects, but they are usually very similar, and the below are commonly recognisable.

Symbol	Description	Formal name
$\neg$	Not	Compliment
$\wedge$	And	Conjunction
$\vee$	Or	(Inclusive) disjunction
$\oplus$	Exclusive-or (xor)	(Exclusive) disjunction
$\rightarrow$	If x then y	Implication
$\equiv$	X if and only if y	Equivalence

Truth tables

- The result of connectives can be represented as a **truth table**, where the input(s) produce a particular output.

A	¬A
False	**True**
True	**False**

A	B	A ∧ B
False	False	**False**
False	True	**False**
True	False	**False**
True	True	**True**

A	B	???
False	False	**False**
False	True	**True**
True	False	**True**
True	True	**True**

Truth tables

- The result of connectives can be represented as a **truth table**, where the input(s) produce a particular output.

A	B	A ⊕ B
False	False	**False**
False	True	**True**
True	False	**True**
True	True	**False**

A	B	A → B
False	False	**True**
False	True	**True**
True	False	**False**
True	True	**True**

A	B	A≡B
False	False	**True**
False	True	**False**
True	False	**False**
True	True	**True**

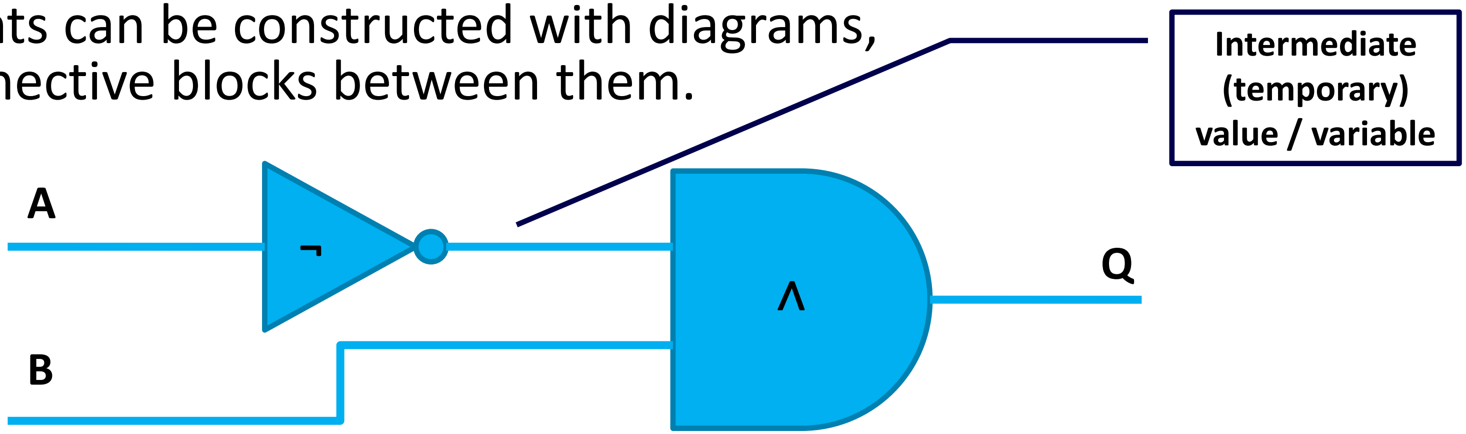
Truth tables

- The revision friendly version ☺

A	B	¬A	A ∧ B	A ∨ B	A ⊕ B	A → B	A≡B
False	False	True	False	False	False	True	True
False	True	True	False	True	True	True	False
True	False	False	False	True	True	False	False
True	True	False	True	True	False	True	True

Diagrammatically

- Statements can be constructed with diagrams, with connective blocks between them.



A	B	t0	Result (Q)
False	False	True	?
False	True	True	?
True	False	False	?
True	True	False	?

Boolean Algebra

- Represent **false as 0** and **true as 1**.
  - Binary digits
- Statements can only be **functions or variables**.
  - No more words
- Use **¬, ∧, ∨, ⊕** to connect statements, forming larger expressions.
  - No implication or equivalence
- Observe a set of **axioms** (rules).
  - These help us manipulate expressions.

Axioms in Boolean algebra

- Some rules allow us to condense or rearrange expressions, to make them simpler.

Rule	Axioms
Commutativity	$x \wedge y \equiv y \wedge x$ $x \vee y \equiv y \vee x$
Association	$(x \vee y) \vee z \equiv x \vee (y \vee z)$ $(x \wedge y) \wedge z \equiv x \wedge (y \wedge z)$
Distribution	$x \wedge (y \vee z) \equiv (x \wedge y) \vee (x \wedge z)$ $x \vee (y \wedge z) \equiv (x \vee y) \wedge (x \vee z)$

Axioms in Boolean algebra

- Some help us avoid evaluating parts, because we can know the answer regardless of variables.

Rule	Axioms
Identity	$x \wedge 1 \equiv x$ $x \vee 0 \equiv x$
Null	$x \wedge 0 \equiv 0$ $x \vee 1 \equiv 1$
Idempotency	$x \wedge x \equiv x$ $x \vee x \equiv x$
Inverse	$x \wedge \neg x \equiv 0$ $x \vee \neg x \equiv 1$

- Duality – swap 0s and 1s, conjunction and disjunction. Equivalence property is preserved.

Axioms in Boolean algebra

- Some help us avoid evaluating parts, because we can know the answer regardless of variables.

Rule	**Axioms**
Absorption	$x \wedge (x \vee y) \equiv x$ $x \vee (x \wedge y) \equiv x$
deMorgan	$\neg(x \wedge y) \equiv \neg x \vee \neg y$ $\neg(x \vee y) \equiv \neg x \wedge \neg y$
Equivalence	$(x \equiv y) \equiv (x \rightarrow y) \wedge (y \rightarrow x)$
Implication	$x \rightarrow y \equiv \neg x \vee y$
Involution	$\neg\neg x \equiv x$

Axioms in Boolean algebra

- For an expression, e, its complement, ¬e, can be formed by:
  - Complement of all variables
  - Complement of all constants
  - Interchange conjunction and disjunction
- Let's try it!

$$E = x \wedge y \wedge z$$

$$\neg e = ?$$

Standard forms: SoP and PoS

- Sum of Products, disjunctive normal form
  - Groups of conjunctions (products) connected together with disjunctions (sums)

$$\underbrace{(a \wedge \neg b \wedge c)}_{\text{Minterm}} \vee \underbrace{(\neg d \wedge e)}_{\text{Minterm}}$$

- Product of sums, conjunctive normal form
  - Groups of disjunctions connected with conjunctions

$$\underbrace{(a \vee \neg b \vee c)}_{\text{Maxterm}} \wedge \underbrace{(\neg d \vee e)}_{\text{Maxterm}}$$

Summary

- Propositional logic
  - Conjunction, implication, equivalence, etc…
  - Truth tables
  - Diagrams
- Boolean algebra
  - Constraints built upon propositional logic
  - Axioms
  - Standard forms
- Lots of maths
  - But now we can start to build digital systems!

In this unit

Foundations	Data representation, logic.
Building blocks	**Transistors, transistor based logic**, simple devices, storage.
Modules	Hex modules, memory, simple controller and processor.
Programming	Assembly, assembler, language, compilation phases, boot-strapping.
Bigger systems	ARM & Thumb, I/O, protecting shared systems, memory hierarchy, multi-processors, networks.
Wrap-up	More examples, historical computers, contemporary systems.