PROGRAMMING LOGIC AND TECHNIQUES

Boolean algebra

Introduction

- **Boolean algebra**, invented by George Boole, is a variety of algebra in which variables can have one of only two possible values (**true** or **false**). These values are called **truth values.**

- Variables that can only have one of two values are called **Boolean variables.**

- Boolean variables are very useful in computing, because they enable computers to make decisions when executing a program. However, prior to using Boolean algebra, we must understand the basic functioning of this algebra.

Boolean propositions

- A Boolean proposition can be thought of as a statement that allows for **exactly two possibilities**: it is either true or false.

- These are the two possible truth values of the proposition.
  - A proposition cannot have a truth value other than these two.
  - A proposition cannot have more than one truth value (at a time).
  - A proposition cannot have no truth value.
  - Therefore, a proposition must have exactly one of these two values.

Boolean propositions

- For a proposition to be Boolean, it must be possible (at least in principle) to determine its truth value, i.e. whether it is true or false.

- *Example:* It is currently raining outside. (**true** or **false**)

Boolean propositions

Example: Is the following a Boolean proposition?

What is the weather like outside?

No

Exercises

- For each of the following, is it a Boolean proposition?
  - a) It is 3:25pm right now.
  - b) I like mathematics.
  - c) What I ate for breakfast.
  - d) There are 5 students in this class.
  - e) What is the homework for tonight?
  - f) $4 + 6 = 12$
  - g) $4 + 6$

Exercises

- For each of the following, is it a Boolean proposition?

Yes
- a) It is 3:25pm right now.

Yes
- b) I like mathematics.

No
- c) What I ate for breakfast.

Yes
- d) There are 5 students in this class.

No
- e) What is the homework for tonight?

Yes
- f) $4 + 6 = 12$

No
- g) $4 + 6$

Boolean propositions

- We’ve just seen that a Boolean proposition is either true or false. These truth values are often represented by the digits “0” and “1”, where zero (0) signifies that a proposition is false and one (1) signifies that it is true.

- We use these two digits because there exists an important connection between Boolean algebra and binary code, which uses “0” and “1” for writing numerals.

Boolean propositions

- Thus, when a proposition is true, we write 1, and when a proposition is false, we write 0.

Example:

The truth value for the proposition:

"The board is white"

is 1 (because it is true).

What is the truth value (0 or 1) for $2 + 5 = 8$?

Exercises

- What is the truth value (0 or 1) for the following propositions?
  - a) The sky is entirely blue.
  - b) $3 > 5$ (3 is greater than 5.)
  - c) $8 < 17$ (8 is less than 17.)
  - d) $5 - 3 = 4$
  - e) It snowed yesterday in this city.
  - f) There are 6 tables in this classroom.

Boolean propositions

- So as we have seen, for any proposition, there are exactly two possibilities:
  either it is true (1), or else it is false (0).

Boolean propositions

- But what if we have two propositions? For example:

  "The sun is shining AND I just ate a meal."

- In this case, there are four possibilities :
- Both can be false:

  (the sun **IS NOT** shining **AND** I **DID NOT** just eat a meal)
- The first can be false while the second is true:

  (the sun **IS NOT** shining **AND** I **DID** just eat a meal)
- The first can be true while the second is false:

  (the sun **IS** shining **AND** I **DID NOT** just eat a meal)
- Both can be true:

  (the sun **IS** shining **AND** I **DID** just eat a meal)

Boolean propositions

- To help us better visualize all of these possibilities, we can structure the information in a table called a **truth table.**

Truth tables

- A truth table is a table that allows us to visualize, in a concise and structured way, all of the possibilities with respect to one or multiple propositions.

Truth tables

- Proposition:

  "My boss just gave me a raise."

To simplify our notation, we will represent the proposition above with the letter "p". The proposition **p** has two possible truth values: false and true, or in other words, 0 and 1. We will represent these two possibilities in the truth table of **p**, seen opposite.

p
0
1

Truth tables

- This truth table allows us to represent the proposition **p** ("My boss just gave me a raise") and its two truth values (false or true: 0 or 1) in a condensed form.

p
0
1

Truth tables

- Now let's consider two propositions:
  - Proposition **p** (I am sick)
  - Proposition **q** (I am taking vitamins).

- Here, we have two propositions and consequently, 4 possibilities:
  - both are false, or
  - the first is false and the second is true, or
  - the first is true and the second is false, or
  - both are true.

Truth tables

- Let's write all of these possibilities in a table in order to construct the truth table for the propositions **p** and **q**.

Observe the sequence of binary numerals that appears in the table.

$$00_{(2)} = 0_{(10)}$$
$$01_{(2)} = 1_{(10)}$$
$$10_{(2)} = 2_{(10)}$$
$$11_{(2)} = 3_{(10)}$$

	p	**q**
Both propositions are false	0	0
The first is false and the second is true	0	1
The first is true and the second is false	1	0
Both propositions are true	1	1

Truth tables

- When inputting simple propositions into a truth table, they should **always** be presented in this order.

	p	**q**
Both propositions are false	0	0
The first is false and the second is true	0	1
The first is true and the second is false	1	0
Both propositions are true	1	1

- *Example 1:*

  Let's make the truth table for the simple propositions **p** and **q**.
  We must always write this table in the same way, as seen above.

- In this case, we have 2 propositions, so 4 possibilities ($2^2 = 4$).

Truth tables

- *Example 2:* Let's make the truth table for the three simple propositions **p**, **q** and **r**.
- In this case, we have three propositions, so 8 possibilities ($2^3 = 8$).

p	**q**	**r**
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

These are all of the possibilities when there are three propositions.

The first line indicates that all three propositions are false, the last line indicates that all three are true, and the lines in between indicate all of the other possibilities.

Truth tables

- Notice once again the order in which the binary numerals – which represent all of the different possible combinations – are written.

$$000_{(2)} = 0_{(10)}$$
$$001_{(2)} = 1_{(10)}$$
$$010_{(2)} = 2_{(10)}$$
$$011_{(2)} = 3_{(10)}$$
$$100_{(2)} = 4_{(10)}$$
$$101_{(2)} = 5_{(10)}$$
$$110_{(2)} = 6_{(10)}$$
$$111_{(2)} = 7_{(10)}$$

These are the 8 possibilities.

Truth tables

For reasons of standardization, it is important to always write the possibilities as an ordered binary sequence.

We can also see that the number of possibilities is always determined by the number of propositions.

1 proposition	=	2	(2^1) possibilities
2 propositions	=	4	(2^2) possibilities
3 propositions	=	8	(2^3) possibilities
4 propositions	=	16	(2^4) possibilities

etc.

Boolean operators

- Boolean propositions can be simple or complex. The propositions that we have seen so far were simple, due to the fact that we had not yet transformed them or applied Boolean operations to them.

Boolean operators

- **Boolean operators** can be compared to the basic operators of arithmetic. The Boolean operators we will cover will enable us to perform operations upon the simple propositions seen previously.

- We will cover the following operators:
  - Negation (¬ / NOT)
  - Conjunction (∧ / AND)
  - Disjunction (∨ / OR)

Boolean operators

- Each operator connects to either 1 or 2 propositions (the **operands** of the operation), thereby producing a new proposition which evaluates to a truth value that is completely determined by the truth value(s) of the operand(s).

- We can make an analogy here between logic and mathematics:

Boolean operators

- The **factorial** of a positive integer **n**:
  - **n**! = **n** × (**n** – 1) × … × 1
  - 5! = 5 × 4 × 3 × 2 × 1 = 120
  - The factorial operator applies to 1 operand, **n**.
  - As soon as we establish the value of **n** (5), the value of **n**! (120) becomes completely determined and fixed.
- The **product** of two real numbers **a** and **b**:
  - $\mathbf{a} \times \mathbf{b} = \underbrace{\mathbf{a} + \cdots + \mathbf{a}}_{\mathbf{b}}$
  - 3 × 5 = 3 + 3 + 3 + 3 + 3 = 15
  - The multiplication operator applies to 2 operands, **a** and **b**.
  - As soon we establish the values of **a** and **b** (3 and 5), the value of **a** × **b** (15) becomes completely determined and fixed.

Negation (NOT / ¬)

- **Negation (NOT / ¬)** is a Boolean operator that applies only to a single proposition (1 operand). In essence, it is a logical inverter, meaning that:
  - It transforms a true proposition into a false proposition
  - It transforms a false proposition into a true proposition

Negation (NOT / ¬)

- Consider the proposition **p**: “It is snowing”.
- This proposition can be true or false. The negation of this proposition, notated in logic as **¬p** (pronounced “not **p**”), is “It is not snowing”.
- So, when **p** is true, **¬p** is false and when **p** is false, **¬p** is true. In other words, if it is true that it is snowing, it is false that it is not snowing. And if it is false that it is snowing, then it is true that it is not snowing. The proposition **¬p** is the inverse of proposition **p**.

Negation (NOT / ¬)

- To better visualize this Boolean operator, let's construct its truth table.

p	**¬p**
0	**1**
1	**0**

We must always start by writing the truth table with the simple propositions (in this case just **p**) forming an ordered binary sequence.

In this case, we have 2 possibilities because we have only one simple proposition ($2^1=2$).

It is now possible to complete the **¬p** column by inverting the **p** column.

- The negation of proposition **p**, written "**¬p**" (pronounced "not p"), is false when **p** is true, and vice versa.

Conjunction (AND / ∧)

- **Conjunction (and / ∧)** is a Boolean operator that applies to 2 propositions (2 operands). It produces a new proposition that is true, if and only if, both of the two propositions it connects are true.

Conjunction (AND / ∧)

- Consider proposition **p**: “We received an increase in salary” and proposition **q** : “We will have more vacation days”.
- The complex proposition **p ∧ q** (read “**p** and **q**”) is interpreted as

  “We received an increase in salary **and** we will have more vacation days”.
- This complex proposition is true if and only if the two simple propositions **p** and **q** are true. If one of these is false, then the conjunction is necessarily false, because in order for it to be true, **both** of the operands need to be true.

Conjunction (AND / $\wedge$)

To get a better understanding, let's construct the truth table for **p ∧ q**.

p	**q**	**p ∧ q**
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

We must always start by writing the truth table with the simple propositions (in this case **p** and **q**) forming an ordered binary sequence.

In this case, we have 2 simple propositions, so 4 possibilities ($2^2 = 4$).

We can now complete the **p ∧ q** column, ending with a 1 on the last line only, as the conjunction can only be true when both operands are true.

Conjunction (AND / ∧)

- The conjunction of two propositions **p** and **q**, notated as **p** ∧ **q** (pronounced "**p** and **q**"), is true if and only if both propositions **p** and **q** are true.

Disjunction (OR / V)

- **Disjunction (OR / V)** is a Boolean operator that applies to 2 propositions (2 operands). It produces a new proposition that is true, if and only if, at least one of the two propositions it connects is true.

Disjunction (OR / V)

- Consider proposition **p**: “We will be receiving an increase in salary” and proposition **q**: “We will have more vacation days”.
- The complex proposition **p V q** (read “**p** or **q**”) is interpreted as:
  “We will be receiving an increase in salary **or** we will have more vacation days”.
- This complex proposition is true if and only if **at least one** of the two simple propositions, **p** or **q**, is true. If both propositions are false, the disjunction is necessarily false, because in order for a disjunction to be false, both of the operands must be false.

Disjunction (OR / V)

- To get a better understanding, let's put together the truth table for **p V q**.

p	q	p V q
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

- We must always start by writing the truth table with the simple propositions (in this case **p** and **q**) forming an ordered binary sequence.
- In this case, we have 2 simple propositions, so 4 possibilities ($2^2 = 4$).
- We can now complete the **p V q** column by writing 0 on the first line only, since the disjunction can only be false if both of the propositions it connects are false. We write 1 on all of the other lines, as each of those possibilities includes at least one operand that is true.

Disjunction (OR / V)

- The disjunction of the two propositions **p** and **q**, notated as "**p V q**" (pronounced "**p** or **q**"), is true if and only if at least one of the two propositions, **p** and **q**, is true.

Mixed propositions

- **Mixed propositions** are propositions that include multiple Boolean operators, and that can also include parentheses with Boolean operators nested inside of them.

Mixed propositions

Consider the following mixed proposition:

$$\neg p \land (q \lor p)$$

It contains:

- Negation: $\neg$
- Conjunction: $\land$
- Disjunction: $\lor$
- Parentheses: ( )

Mixed propositions

Consider the following mixed proposition:

$$\neg p \wedge (q \vee p)$$

Let's construct the truth table for this proposition. In order to do so, we must proceed step by step.

Mixed propositions

- First, it is important to know the **order of operations**:
  - 1) Parentheses: ( )
  - 2) Negations: ¬
  - 3) Conjunctions: ∧ (from left to right)
  - 4) Disjunctions: ∨ (from left to right)

Mixed propositions

- Let's return to the previous proposition: **$\neg p \land (q \lor p)$**

- Let's construct the truth table for this proposition:

p	q	q V p	¬p	¬p Λ (q V p)
0	0	0	1	**0**
0	1	1	1	**1**
1	0	1	0	**0**
1	1	1	0	**0**

Mixed propositions

Proposition: **¬p Λ (q V p)**

We must always start by writing the truth table with the simple propositions (in this case **p** and **q**) forming an ordered binary sequence.

Subsequently, we process the operation inside the parentheses **(q V p)**, followed by the negation **¬p**. Finally, we can establish the truth table values for the mixed proposition **¬p Λ (q V p)**.

p	**q**	**q V p**	**¬p**	**¬p Λ (q V p)**
0	0	0	1	**0**
0	1	1	1	**1**
1	0	1	0	**0**
1	1	1	0	**0**

Mixed propositions

Thus, we must proceed step by step, respecting the order of operations.

Example : Construct the truth table for **p ∧ (¬p ∨ q)**

The first row in the table below displays the order of operations, from left to right.

In this case, we have started by writing the **p** and **q** columns as we should always do ($2^2=4$ possibilities).

Next is **¬p**, because this negation is contained inside the parentheses.

Then, the entire expression in parentheses **(¬p ∨ q)**, and finally the proposition as a whole, **p ∧ (¬p ∨ q)**.

Mixed propositions

p	**q**	**¬p**	**(¬p ∨ q)**	**p ∧ (¬p ∨ q)**
0	0	1	1	**0**
0	1	1	1	**0**
1	0	0	0	**0**
1	1	0	1	**1**

Propositions with three or more variables

- Consider the following mixed proposition:

$$\mathbf{p} \wedge (\mathbf{q} \vee \mathbf{r})$$

Here we have 3 simple propositions, **p**, **q**, and **r**. These must be written first in the truth table.

Since we have 3 simple propositions, there will be 8 possibilities ($2^3=8$).

Propositions with three or more variables

p	q	r	(q ∨ r)	p ∧ (q ∨ r)
0	0	0	0	**0**
0	0	1	1	**0**
0	1	0	1	**0**
0	1	1	1	**0**
1	0	0	0	**0**
1	0	1	1	**1**
1	1	0	1	**1**
1	1	1	1	**1**

Tautologies

A proposition is **tautological** if it is always true – that is, if it is necessarily true, or true in every possible case.

Consider this proposition: **p V ¬p**

p	**¬p**	**p V ¬p**
0	1	**1**
1	0	**1**

We can see that the proposition **p V ¬p** is always true, since there are only 1s in the last column. Therefore this proposition is tautological (it is a tautology).

Contradictions

A proposition is a **contradiction** if it is always false – if it is necessarily false, or false in every possible case.

Consider this proposition: $\mathbf{p \land \neg p}$

$\mathbf{p}$	$\mathbf{\neg p}$	$\mathbf{p \land \neg p}$
0	1	**0**
1	0	**0**

We can see that the proposition $\mathbf{p \land \neg p}$ is always false, since there are only 0s in the last column. Therefore, this proposition is a contradiction.

Equivalence

Two propositions are **equivalent** if their respective truth table columns are identical. This means that they necessarily share the same truth value in every possible case, regardless of the truth of their simple propositions.

Consider the following two propositions:

$$\neg(\mathbf{p} \lor \mathbf{q}) \quad \text{and} \quad \neg\mathbf{p} \land \neg\mathbf{q}$$

Let's verify whether they are equivalent.

Equivalence

To test their equivalence, we must construct a truth table for each proposition.

Below is the truth table for proposition **¬(p V q)**:

p	**q**	**p V q**	**¬(p V q)**
0	0	0	**1**
0	1	1	**0**
1	0	1	**0**
1	1	1	**0**

Equivalence

Now here is the truth table for the proposition **¬p ∧ ¬q**:

p	**q**	**¬p**	**¬q**	**¬p ∧ ¬q**
0	0	1	1	**1**
0	1	1	0	**0**
1	0	0	1	**0**
1	1	0	0	**0**

Equivalence

Observe the last column of each table :

p	q	¬p	¬q	¬p ∧ ¬q
0	0	1	1	**1**
0	1	1	0	**0**
1	0	0	1	**0**
1	1	0	0	**0**

p	q	p ∨ q	¬(p ∨ q)
0	0	0	**1**
0	1	1	**0**
1	0	1	**0**
1	1	1	**0**

You will notice that, for identical input, they yield identical output. Therefore, the propositions are equivalent. In other words, they necessarily share the same truth value.