Algorithmic Foundations 2

Section 1 – Propositional logic

Dr. Gethin Norman

School of Computing Science
University of Glasgow

Logic

Logic is the basis of all mathematical reasoning

The rules of logic give precise meaning to mathematical statements

Provides a methodology for objectively reasoning about the truth or falsity of such statements

It is the foundation for expressing formal proofs in all branches of mathematics

Propositional logic

Propositional Logic is the logic of compound statements built from simpler statements using Boolean connectives

Some applications in computing science

- design of digital electronic circuits
- expressing conditions in programs
- queries to databases and search engines

Propositions

Propositions are the basic building blocks of logic

- declarative sentences that are either **true** or **false** (but not both)

Examples:

a) Glasgow is a city in Scotland

b) 1+1 = 2

c) 2+2 = 3

d) every day is Friday

- a) and b) are **true** while c) and d) are **false**

The area of logic that deals with propositions is called propositional logic or propositional calculus

Propositions

The following are **not** propositions:

a) what's is the weather like?
b) Drink tea
c) `x+1 = 2`
d) `x+y = z`

a) & b) are not declarative sentences and are neither **`true`** nor **`false`**

c) & d) have unassigned variables, and are neither **`true`** or **`false`**

- can be **`true`** or **`false`** depending on the values the variables are assigned

Notation

Truth value of a proposition is either:

- **true** or alternatively written 1 or **T**
- **false** or alternatively written 0 or **F**

Lower case letters usually used to represent propositions

- typically use the letters p, q, r, ….

Examples:

- let p be the proposition “today is Friday”
- let q be the proposition “it is raining”

Connectives

Used to generate new mathematical statements by combining one or more propositions

Generated statements are called compound propositions or formulae

We use capital letters to denote such statements

- typically use the letters P, Q, R, …

We will use truth tables which display the relationship between the truth value of a formula and the truth values of the propositions (and subformulae) within it

Connectives – Negation

Examples:

- let p be the proposition “today is Friday”
- then ¬p is the proposition “it is not the case that today is Friday” or better “it is not Friday”
- let q be the proposition “it is raining”
- then ¬q is the proposition “it is not the case that it is raining” or better “it is not raining”

Truth table for ¬p:

p	¬p
false	true
true	false

or equivalently

p	¬p
0	1
1	0

Connectives – Conjunction

Example:

- let p be the proposition "today is Friday"
- let q be the proposition "it is raining"
- then p∧q (equivalently p and q) is the proposition
  "today is Friday **and** it is raining"

Truth table **p∧q**:

p	q	p∧q
0	0	0
0	1	0
1	0	0
1	1	1

Note: in the first two columns count up in binary as we move down the rows

Connectives – Disjunction

Example:

- let p be the proposition "today is Friday"
- let q be the proposition "it is raining"
- then $p \vee q$ (equivalently p or q) is the proposition "today is Friday **or** it is raining"

Truth table for $p \vee q$:

p	q	$p \vee q$
0	0	0
0	1	1
1	0	1
1	1	1

Connectives – Exclusive Or

Example:

- let p be the proposition "today is Friday"
- let q be the proposition "it is raining"
- then $p \oplus q$ (equivalently `p xor q`) is the proposition "either today is Friday **or** it is raining but **not both**"

Truth table for $p \oplus q$:

p	q	$p \oplus q$
0	0	0
0	1	1
1	0	1
1	1	0

Connectives – Exclusive Or

Another example:

- p: “I passed the AF2 assessed exercise”
- q: “I failed the AF2 assessed exercise”

Either I passed or failed the AF2 assessed exercise

But I cannot both pass and fail the exercise

- so exclusive or rather than disjunction

English is often imprecise…

p	q	p∨q
0	0	0
0	1	1
1	0	1
1	**1**	**1**

p	q	p⊕q
0	0	0
0	1	1
1	0	1
1	**1**	**0**

A slight detour…

Computers work in binary (0 and 1's) and all computation a computer performs (on a chip) reduces to the operations

- negation
- conjunction
- disjunction
- exclusive or

So any program you write in Java, C++, … when compiled is reduced to these operations on bits…

Connectives – Implication

Example:

– let p be the proposition "today is Friday"
– let q be the proposition "it is raining"
– then $p \rightarrow q$ (equivalently p implies q) is the proposition
"**if** today is Friday, **then** it is raining"

Truth table for $p \rightarrow q$:

p	q	$p \rightarrow q$
0	0	1
0	1	1
1	0	0
1	1	1

Connectives – Implication

Implication is often misunderstood

- p implies q
- if p, then q
- if p, q
- q whenever p
- … see Rosen for a much longer list

p	q	p→q
0	0	1
0	1	1
1	0	0
1	1	1

think of $p \rightarrow q$ as a contract

Connectives – Implication

Think of $p \rightarrow q$ as a contract

- the contract holds, or it does not

If it is sunny, then you will take me to the beach

- p: it is sunny
- q: you will take me to the beach

p	q	$p \rightarrow q$	what does this mean?
0	0	1	it was not sunny and you did not take me to the beach (alright)
0	1	1	it was not sunny and you did take me to the beach (a bonus)
1	0	0	it was sunny and you did not take me to the beach (contract broken)
1	1	1	it was sunny and you took me to the beach (good)

Connectives – Implication

Think of $p \rightarrow q$ as a contract

- the contract holds, or it does not

The car does not start whenever the battery is dead

- written another way: if the battery is dead, then the car does not start
- p: the battery is dead
- q: the car does not start

p	q	$p \rightarrow q$	what does this mean?
0	0	1	battery fine and car starts (good news)
0	1	1	battery fine, but car does not start (maybe no petrol?)
1	0	0	battery dead and car starts (contract broken)
1	1	1	battery dead and car does not start (as advertised)

Connectives – Implication

We are given 4 cards

- cards have a letter on one side and number on the other side

and have the following rule:

- if a card has number 3 on one side, then it has letter B on the other

What cards must be turned over to confirm that the rule holds?

- p: card has the number 3 on one side
- q: card has a B on the other side
- $p \rightarrow q$

p	q	$p \rightarrow q$
0	0	1
0	1	1
1	0	0
1	1	1

B	7	C	3

Connectives – Implication

We are given 4 cards

- cards have a letter on one side and number on the other side

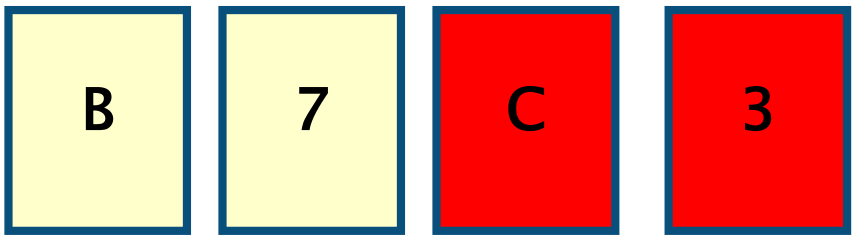
and have the following rule:

- if a card has number 3 on one side, then it has letter B on the other

What cards must be turned over to confirm that the rule holds?

- p: card has the number 3 on one side
- q: card has a B on the other side
- $p \rightarrow q$

p	q	$p \rightarrow q$
0	0	1
0	1	1
1	0	0
1	1	1



We only need to turn these (red) cards over

Connectives – Implication

Relational implications that can be formed from $p \rightarrow q$

- converse: $q \rightarrow p$
- contrapositive: $\neg q \rightarrow \neg p$
- inverse: $\neg p \rightarrow \neg q$

The contrapositive is equivalent to the original statement

The inverse is actually the contrapositive of the converse

- so the converse and inverse are equivalent

Connectives – Biconditional

Example:

- let p be the proposition "today is Friday"
- let q be the proposition "it is raining"
- then $p \leftrightarrow q$ (equivalently `p if and only if q`) is the proposition "today is Friday **if and only if** it is raining"

Truth table for **$p \leftrightarrow q$**:

p	q	$p \leftrightarrow q$
0	0	1
0	1	0
1	0	0
1	1	1

for $p \leftrightarrow q$ to hold either both p and q are **`true`** or both are **`false`**

Connectives – Precedence

We can construct compound propositions from propositions using the connectives we have introduced

- we use parentheses to specify the order the connectives are applied
- important as this order will change the truth values of statements

Example:

- if $(p \vee q) \wedge r$ is **true**, then r and either p or q must be **true**
  - (either p or q) and r
- if $p \vee (q \wedge r)$ is **true**, then either p or both q and r must be **true**
  - either p or (q and r)

Connectives – Precedence

- if $(p \vee q) \wedge r$ is **true**, then r and either p or q must be **true**
- if $p \vee (q \wedge r)$ is **true**, then either p or both q and r must be **true**

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

Connectives – Precedence

To reduce the number of parenthesis we assume negation is applied before all other operators

- negation has the highest precedence
- $\neg p \wedge r$ is the same as $(\neg p) \wedge r$ and does not mean $\neg(p \wedge r)$

As a general rule $\wedge$ has the next highest precedence, followed by $\vee$, then $\rightarrow$ and finally $\leftrightarrow$

- however, to avoid ambiguity and confusion I will use parenthesis between these operators

When in doubt use parenthesis

- will avoid errors and confusion

Tautologies and Contradictions

A **tautology** is a formula that is always `true`

- classic examples: $p \rightarrow p$ and $p \vee \neg p$

A **contradiction** is a formula that is always `false`

- classic example: $p \wedge \neg p$

What else is there…

- a contingency is something which is neither a tautology or a contradiction
- examples: $p \rightarrow q$, $p \vee q$ and $p \wedge q$

A formula is called **satisfiable** if there is an assignment of truth values to the propositions that makes the formula `true`

- i.e. the formula is not a contradiction

Logical equivalence

Two syntactically (i.e. textually) different compound propositions may be semantically identical (i.e. have the same meaning)

- it such cases they are called logically equivalent

The statement P ≡ Q expresses that P is logically equivalent to Q

- given any assignment to the propositions appearing in P and Q the truth values of P and Q are the same

Logical equivalence can be proved by:

- by laws of logical equivalence (this lecture)
- by a truth table (i.e. show truth table columns are the same)
  - number of rows of a truth table is 2^n where n is number of propositions
- by some other line of reasoning

Logical equivalence

Two syntactically (i.e. textually) different compound propositions may be semantically identical (i.e. have the same meaning)

- it such cases they are called logically equivalent

To show two compound propositions are not logically equivalent

- need to give an assignment to the propositions that makes one of the formulae **true** and the other formula **false**
  - notice do not need to give the full truth table just one row

Example: $(p \lor q) \land r$ and $p \lor (q \land r)$ are not logically equivalent

- if p is **true** while both q and r are **false**, then
- $(p \lor q) \land r$ evaluates to **false**
- $p \lor (q \land r)$ evaluates to **true**

Laws of logical equivalence

These are similar to the arithmetic identities such as:

- $x \cdot 0 = 0$
- $x \cdot 1 = x$
- $x + y = y + x$
- $x \cdot (y + z) = x \cdot y + x \cdot z$
- $x + (y + z) = (x + y) + z$

Equality replaced by logical equivalence and applied to compound propositions (formulae)

- provides a pattern or template that can be used to match all or part of a much more complicated proposition and to find an equivalence for it
  - examples to come
- can replace pattern on either side of the law with that on the other

Laws – Identity & domination

Identity laws:

- $P \land \textbf{true} \equiv P$
- $P \lor \textbf{false} \equiv P$

Domination laws:

- $P \lor \textbf{true} \equiv \textbf{true}$
- $P \land \textbf{false} \equiv \textbf{false}$

Notice **P** is a compound proposition so example applications are

- $((p \lor q) \land r) \land \textbf{true} \equiv ((p \lor q) \land r)$ identity law
- $((p \lor q) \land r) \equiv ((p \lor q) \land r) \land \textbf{true}$ identity law

Works both ways around (do not have to go from lhs of law to rhs)

Laws – Identity & domination

Identity laws:

- $P \land \mathbf{true} \equiv P$
- $P \lor \mathbf{false} \equiv P$

Domination laws:

- $P \lor \mathbf{true} \equiv \mathbf{true}$
- $P \land \mathbf{false} \equiv \mathbf{false}$

Can also apply to subexpressions, e.g.

- $(P \land \mathbf{(Q \land true)}) \equiv P \land \mathbf{Q}$ identity law
- $\mathbf{(P \land false)} \rightarrow R \equiv \mathbf{false} \rightarrow R$ domination law
  - notice removed extra parenthesis, e.g. (Q) becomes Q

Slight detour – Identities and zeros

Identities and zeros

- identity "id" of an operator "**op**" is such that `p op id = p` for all `p`
- zero "zero" of an operator "**op**" is such that `p op zero = zero` for all `p`

Examples:

- identity of addition is 0: $x+0 = x$ for all x
- identity of multiplication is 1 while zero is 0 ($x \cdot 1 = x$ and $x \cdot 0 = 0$)

For conjunction: identity is **true** and zero is **false**

- $P \wedge \textbf{true} \equiv P$ and $P \wedge \textbf{false} \equiv \textbf{false}$

For disjunction: identity is **false** and zero is **true**

- $P \vee \textbf{false} \equiv P$ and $P \vee \textbf{true} \equiv \textbf{true}$

Laws – Idempotent & double negation

Idempotent laws:

- $P \wedge P \equiv P$
- $P \vee P \equiv P$

Double negation law:

- $\neg(\neg P) \equiv P$

Laws – Commutative and associative

Commutative laws:

- $P \land Q \equiv Q \land P$
- $P \lor Q \equiv Q \lor P$

Associative laws:

- $(P \land Q) \land R \equiv P \land (Q \land R)$
- $(P \lor Q) \lor R \equiv P \lor (Q \lor R)$

From the associative laws

- can write $(P \land Q) \land R$ or $P \land (Q \land R)$ as $P \land Q \land R$ without ambiguity
- or in general express the conjunction of multiple propositions as $P_1 \land P_2 \land \ldots \land P_n$
- we express multiple disjunctions in the same way

Laws – Distributive

Distributive laws:

- $P \lor (Q \land R) \equiv (P \lor Q) \land (P \lor R)$
- $P \land (Q \lor R) \equiv (P \land Q) \lor (P \land R)$

Consider when $(P \lor Q) \land (P \lor R)$ is **`true`**

- i.e. when (P or Q) and (P or R) holds
- if P holds, then we are good (since both sides of the "and" hold)
- if P does not hold, then Q and R must hold for the formula to be **`true`**
- so… P or (Q and R)

Consider when $(P \land Q) \lor (P \land R)$ is **`true`**

- i.e. when (P and Q) or (P and R) holds
- P must hold for either side to hold
- and we need one side to hold, i.e. Q or R
- so… P and (Q or R)

Laws – De Morgan

De Morgan laws:

- $\neg(P \land Q) \equiv \neg P \lor \neg Q$
- $\neg(P \lor Q) \equiv \neg P \land \neg Q$

Consider when $\neg(P \land Q)$ is `true`

- i.e. when not (P and Q) holds
- one of P or Q must be false
- so… (not P) or (not Q)

Consider when $\neg(P \lor Q)$ is `true`

- i.e not (P or Q) holds
- this means neither P or Q can hold
- so… (not P) and (not Q)

Laws – Contradiction, tautology & implication

Contradiction and tautology laws:

- $P \land \neg P \equiv$ **false**
- $P \lor \neg P \equiv$ **true**

Implication law:

- $P \rightarrow Q \equiv \neg P \lor Q$

Similarly we can reduce exclusive or and biconditional:

- $P \oplus Q \equiv (P \lor Q) \land \neg(P \land Q)$
- $P \leftrightarrow Q \equiv (P \rightarrow Q) \land (Q \rightarrow P)$

Slight detour…

Recall the double negation and de Morgan laws

- $\neg(\neg P) \equiv P$
- $\neg(P \wedge Q) \equiv \neg P \vee \neg Q$
- $\neg(P \vee Q) \equiv \neg P \wedge \neg Q$

therefore we have

- $P \wedge Q \equiv \neg(\neg(P \wedge Q))$ double negation law
  $\equiv \neg(\neg P \vee \neg Q)$ De Morgan law

so can define conjunction using disjunction and negation

- $P \vee Q \equiv \neg(\neg(P \vee Q))$ double negation law
  $\equiv \neg(\neg P \wedge \neg Q)$ De Morgan law

so can define disjunction using conjunction and negation

Slight detour…

Recall also the implication law

- $P \to Q \equiv \neg P \lor Q$

it follows that

- $P \lor Q \equiv \neg(\neg P) \lor Q$ double negation
  $\equiv \neg P \to Q$ implication law

- $P \land Q \equiv \neg(\neg(P \land Q))$ double negation
  $\equiv \neg(\neg P \lor \neg Q)$ De Morgan law
  $\equiv \neg(P \to \neg Q)$ implication law

so can define disjunction and conjunction using implication and negation

Slight detour…

Recall also the implication law

- $P \to Q \equiv \neg P \lor Q$

it follows that

- $P \lor Q \equiv \neg(\neg P) \lor Q \equiv \neg P \to Q$
- $P \land Q \equiv \neg(\neg(P \land Q)) \equiv \neg(\neg P \lor \neg Q) \equiv \neg(P \to \neg Q)$

so can define disjunction and conjunction using implication and negation

We can also define negation with implication (and **`false`**)

- $\neg P$ is logically equivalent to $P \to$ **`false`**
  - for $P \to$ **`false`** to be **`true`**, then P must be **`false`**

It follows all operators can be defined using implication (and **`false`**)

Example 1 – $\neg(P \lor (\neg P \land Q)) \equiv \neg P \land \neg Q$

$\neg(P \lor (\neg P \land Q)) \equiv \neg P \land \neg Q$

- we start with either the left or right hand side and apply laws of logical equivalence to derive the other side

$\neg(P \lor (\neg P \land Q))$	$\equiv$	$\neg((P \lor \neg P) \land (P \lor Q))$	distributive law
	$\equiv$	$\neg(\text{true} \land (P \lor Q))$	tautology law
	$\equiv$	$\neg((P \lor Q) \land \text{true})$	commutative law
	$\equiv$	$\neg(P \lor Q)$	identity law
	$\equiv$	$\neg P \land \neg Q$	De Morgan law

De Morgan law: $\neg(P \lor Q) \equiv \neg P \land \neg Q$

Example 2 – $(P \wedge Q) \rightarrow (P \vee Q) \equiv$ **true**

We start with either the left or right hand side and apply laws of logical equivalence to derive the other side

- when showing something is equivalent to **true** or **false** easier to start with that side
- otherwise hard to know where to begin
  - i.e. which law to apply first

Example 2 – $(P \wedge Q) \rightarrow (P \vee Q) \equiv \mathbf{true}$

$(P \wedge Q) \rightarrow (P \vee Q) \equiv$	$\neg(P \wedge Q) \vee (P \vee Q)$	implication law
$\equiv$	$(\neg P \vee \neg Q) \vee (P \vee Q)$	De Morgan law
$\equiv$	$\neg P \vee (\neg Q \vee (P \vee Q))$	associative law
$\equiv$	$\neg P \vee (\neg Q \vee (Q \vee P))$	commutative law
$\equiv$	$\neg P \vee ((\neg Q \vee Q) \vee P)$	associative law
$\equiv$	$\neg P \vee (\mathbf{true} \vee P)$	tautology & commutative
$\equiv$	$\neg P \vee \mathbf{true}$	domination & commutative
$\equiv$	$\mathbf{true}$	domination law

Domination law: $P \vee \mathbf{true} \equiv \mathbf{true}$

Example 3

If we have `true=1` and `false=0`, then we can define negation and conjuction by:

- `not(P) = 1-P`
- `and(P,Q) = P·Q`

But what about disjunction, i.e. what is `or(P,Q)`?

- be careful, what is $P+Q$?

Can be derived using logical equivalences

- by expressing disjunction in terms of a logically equivalent formula including only conjunction and negation
- then applying above definitions for `not(P)` and `and(P,Q)`

Example 3

We showed earlier that $P \vee Q \equiv \neg(\neg P \wedge \neg Q)$

Using this equivalence we have

```
or(P,Q)  = not(and(not(P),not(Q))
         = not(and(1-P,1-Q))
         = not((1-P)·(1-Q))
         = 1 - (1-P)·(1-Q))
         = 1 – (1 - P – Q + P·Q)
         = P + Q - P·Q
```