Algorithmic Foundations 2

Introduction

Simon Fowler & Gethin Norman

School of Computing Science
University of Glasgow

Administrivia

Lecturers: Simon Fowler and Gethin Norman


- Contact details on Moodle
  - In general, please feel free to message / e-mail us
  - Office hours: anytime we are available
- You can also use Teams or Padlet to ask questions
  - Preferable, so that your peers can also learn
  - **If you are not on the AF2 Team please join (the link is on Moodle);** we will be using Teams for communicating so this is important
  - The link for the Padlet is also on Moodle

Lectures: Wednesdays and Fridays either 1100 or 1200

Tutorials: Fridays (start in week 2)

- group sessions with a tutor and demonstrator
- we will set up small groups
- and suggest questions from tutorial sheets for discussion

Assessment

Assessment

- degree exam 75% (December 2024)
- assessed exercise 10% (11 October 2024 – 28 October 2024)
- assessed exercise 10% (8 November 2024 – 25 November 2024)
- in class quizzes 5% (4 questions each week – two per lecture)
  - deadline Tuesday 23:59 the following week

Exam 75%

- All level 1 and 2 exams are **in person**, but we will provide a **formula sheet** for the exam
  - This will be part of the exam paper, and we will release in advance
  - Please ask if you are unsure
- Past papers for the past four years are on Moodle
  - Model solutions are also provided
  - This year's exam will be similar to these exams (along with a similarly silly Q2)

What is this course about?

Algorithmic Foundations 2

No real content about algorithms

Don't worry, you haven't missed an Algorithmic Foundations 1!

It is not the best name…

What is this course about?

In reality, this course could be called "Discrete Mathematics for Computing Scientists"

It covers the background material you require to understand algorithms in computing science; uses include:

- algorithms & data structures, computer networks, operating systems, programming language type systems, compilers & interpreters, computer architecture, database management systems, cryptography, error correction codes, graphics & animation algorithms, game engines, …

  in fact just about everything

Also widely used throughout mathematics, science, engineering, economics, biology…

This means it is mainly focused on mathematics rather than Computing Science

…but all of the concepts have practical applications in CS. We will relate the concepts to relevant parts of CS as we go through the course.

Outline of course

Propositional logic

Predicates and quantifiers

Sets, functions and countability

Numbers: sequences, summations, integers & matrices

Methods of proof

Mathematical induction and recursive definitions

Counting

Probability

Graphs

Relations

Outline of course

A roadmap of the course is provided on Moodle

- details the material to be covered each week
- and who is teaching this material
- Gethin and I have split the course according to the parts that we each find most fun / use most frequently in our specialisms

AF2 – Road Map 2024-25

Week 1 (September 23) – Simon and Gethin
- Lecture 1: Introduction and Section 1 – Propositional logic (Part 1)
- Lecture 2: Section 1 – Propositional logic (Part 2)

Week 2 (September 30) – Simon
- Lecture 1: Section 2 – Predicates & quantifiers
- Lecture 2: Section 3 – Sets, functions & countability (Part 1)
- Lab: Latex Tutorial

Week 3 (October 7) – Simon
- Lecture 1: Section 3 – Sets, functions & countability (Part 2)
- Lecture 2: Section 4 – Numbers: sequences, summations, integers & matrices (Part 1)
- Lab: Tutorial Sheet 1 and Tutorial Sheet 2
- Assessed exercise 1 – Handout (Friday October 11)

Week 4 (October 14) – Simon
- Lecture 1: Section 4 – Numbers: sequences, summations, integers & matrices (Part 2)
- Lecture 2: Section 5 – Methods of proof (Part 1)
- Lab: Tutorial Sheet 3

Week 5 (October 21) – Simon
- Lecture 1: Section 5 – Methods of proof (Part 2)
- Lecture 2: Section 6 – Mathematical induction & recursive definitions (Parts 1)
- Lab: Tutorial Sheet 4

Week 6 (October 28) – Simon
- Lecture 1: Section 6 – Mathematical induction & recursive definitions (Parts 2)
- Lecture 2: Section 6 – Mathematical induction & recursive definitions (Parts 3)
- Lab: Tutorial Sheet 5
- Assessed exercise 1 – Deadline (Monday October 28 @1630)

Week 7 (November 4) – Gethin
- Lecture 1: Section 7 – Counting (Part 1)
- Lecture 2: Section 7 – Counting (Part 2)
- Lab: Tutorial Sheet 7
- Assessed exercise 2 – Handout (Friday November 8)

Week 8 (November 11) – Gethin
- Lecture 1: Section 8 – Probability (Part 1)
- Lecture 2: Section 8 – Probability (Part 2)
- Lab: Tutorial Sheet 8

Week 9 (November 18) – Gethin
- Lecture 1: Section 9 – Graphs
- Lecture 2: Section 10 – Relations
- Lab: Tutorial Sheet 9

Week 10 (November 25) – Gethin
- Lecture 1: Revision go through AF2 Exam December 2022
- Lecture 2: Revision go through AF2 Exam December 2023
- Lab: Revision Session
- Assessed exercise 1 – Deadline (Monday November 25 @1630)

Week 11 (December 2)
- Revision week (office hours will be set up for getting help)

Why are you being taught this course?

Underlying foundations of computer science

- mathematical techniques used throughout computer science

To teach you

- mathematical notations
- mathematical reasoning
- mathematical problem solving
- how to work symbolically

To unlock the skills you will need to understand more advanced CS courses

To have fun(!) with discrete mathematics

What do you need to do?

Attend the lectures and complete the quizzes

Attend the tutorial classes

- the aim is for the classes to be interactive
- you discuss questions/answers with fellow students
- **this only works with your participation**
- you do **not** have to complete all the tutorial exercises each week
- you can also use the exercises when revising for the exam
- the tutorials include challenging questions at the end
  - this was based on student feedback
  - these are supposed to be difficult and you do not need to complete them

Read course textbook and work through the associated exercises

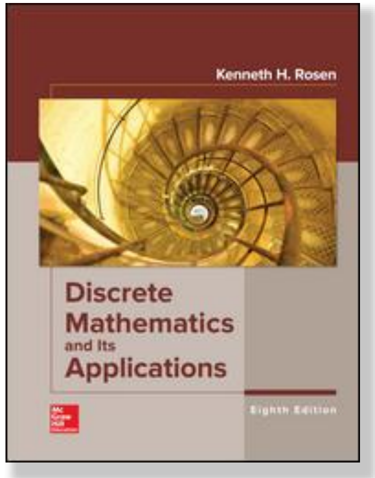
The more work you do yourself rather than passively reading or copying solutions, the more you will learn

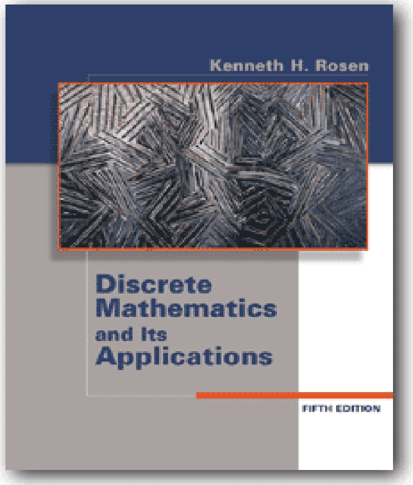
Course book

Discrete Mathematics & its Applications

Kenneth H. Rosen

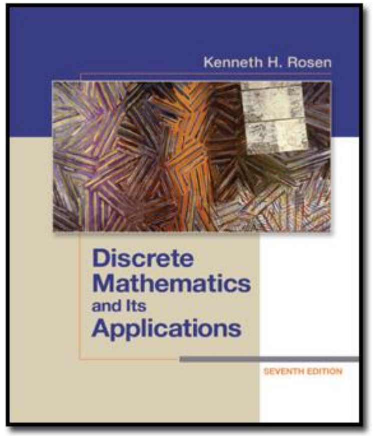
- 5th, 6th, 7th or 8th Edition
- All are absolutely fine
- The library has copies, cheap on Amazon (…and probably cheaper if you Google it)

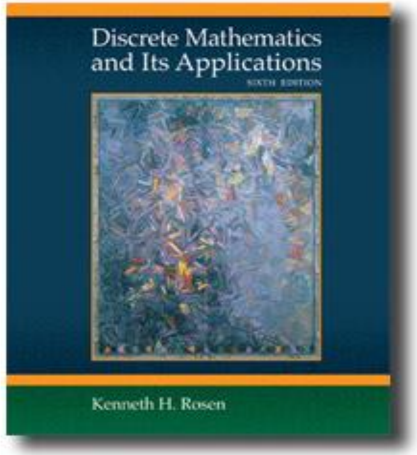




The book has its own web site

- http://www.mhhe.com/rosen





You can use the book in Levels 2, 3, and 4 and beyond

Feedback

- **You will get several forms of feedback in this course:**
  - Verbal feedback from tutors and demonstrators during your tutorials when discussing questions
  - Individual written feedback on your assessed exercises, to explain your grade
  - Automated written feedback on your quiz answers (we will also upload slides explaining quiz answers each week)
  - General written feedback on the exam to explain common patterns and misunderstandings

Typesetting mathematics

LaTeX is the standard typesetting system for the communication and publication of scientific documents

- using LaTeX should help improve the quality of your submitted assessed work (all the tutorial sheets are written in LaTeX)
- help to give your solutions more structure and improve readability
- make it easier to spot errors
- you will also use it for some of your assessments in levels 3 and 4
    - this includes your team project and individual project dissertations
    - LaTeX is part of the level 3 Unix course

Information is available on Moodle

Changes from last year based on student feedback

- **This year all content delivery is through lectures**
  - Last year, it was through a combination of videos and a live summary lecture, which was a remnant of COVID times – and the worst of both worlds
  - This year we have managed to get enough timetable slots to teach everything by lectures (this means giving the same lecture twice in a row – which will be interesting!)
- **We have made quizzes easier to find and they are open longer**
  - Everything linked from course outline document (although they will be unhidden on a per-week basis)
- **New material for learning LaTeX**
  - Done as a tutorial in Week 2, developed by 4^{th} year students in the CS Education course

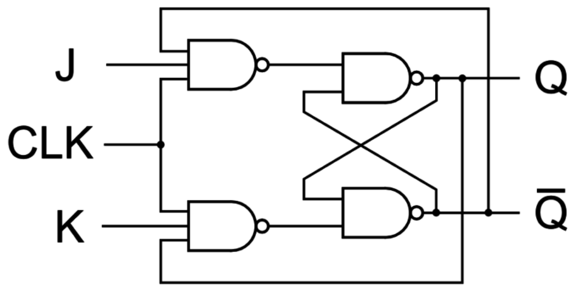
Algorithmic Foundations 2

Propositional Logic

Simon Fowler & Gethin Norman

School of Computing Science
University of Glasgow

Why Learn Logic?



Digital Circuit Design

$$\boxed{\Gamma \vdash M : A}$$

Type systems / typecheckers

Query optimisation

...and many, many more!

Logic is **pervasive**: it crops up everywhere!

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, as well as fields such as philosophy

Outline

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 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

The truth value of a proposition is either:

- `true` or alternatively written 1 or **T**
- `false` or alternatively written 0 or **F** or $\perp$

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”

Outline

Propositions

Connectives

Tautologies and contradictions

Logical equivalence

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 $p \wedge q$ (“p and q”) is the proposition “today is Friday **and** it is raining”

Truth table $p \wedge q$:

p	q	$p \wedge q$
0	0	0
0	1	0
1	0	0
1	1	1

Note: in the first two columns we 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 $p \vee q$ ("p or q") is the proposition "today is Friday **or** it is raining"

Truth table for $p \vee q$:

p	q	p∨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 $p \oplus q$ ("`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"

p	q	p∨q
0	0	0
0	1	1
1	0	1
1	1	1

Either I passed or failed the AF2 assessed exercise

But I cannot both pass and fail the exercise

- so exclusive or rather than disjunction

p	q	p⊕q
0	0	0
0	1	1
1	0	1
1	1	0

English is often imprecise: when we say 'or' in natural language we normally mean 'exclusive or'

E.g., "Would you like a cupcake or a croissant?" "Yes, please"

A slight detour…

Computers work in binary (0 and 1's) and all non-IO 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 $p \rightarrow q$ ("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→q as a contract:
it is **true** unless the **contract is broken**
(i.e., p is true but q is false)

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 (no problem)
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 – sad)
1	1	1	it was sunny and you took me to the beach (good)

Connectives – Implication

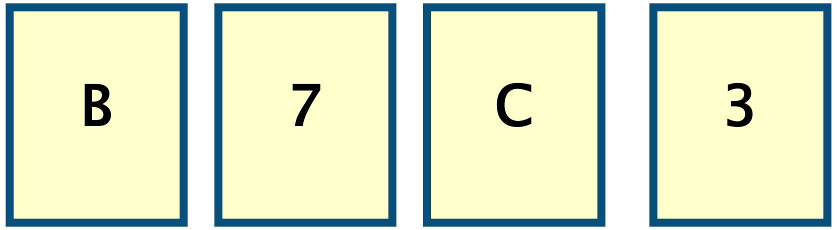
We are given 4 cards

- 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?



Connectives – Implication

We are given 4 cards

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

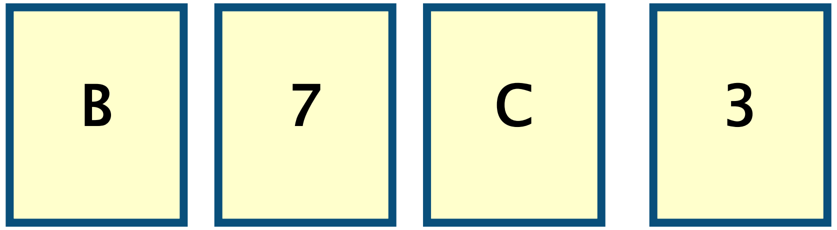
and have the following rule:

- 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



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

Do we need to turn over this (red) card?

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

Do we need to turn over this (red) card?

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

Do we need to turn over this (red) card?

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→q
0	0	1
0	1	1
1	0	**0**
1	1	**1**

B 7 C 3

Do we need to turn over this (red) card?

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

We only need to turn these (red) cards over

Connectives – Implication

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

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

Connectives – Implication

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

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

Converse:

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

notice that they are different

Connectives – Implication

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

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

Converse example:

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

- $p \rightarrow q$ if it is sunny, then you will take me to the beach
  - can still go to the beach when it is not sunny
- $q \rightarrow p$ if you take me to the beach, then it is sunny
  - can be sunny and not go to the beach

Connectives – Implication

Related implications that can be formed from $p \to q$

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

Contrapositive:

p	q	$p \to q$	$\neg q$	$\neg p$	$\neg q \to \neg p$
0	0	1	1	1	1
0	1	1	0	1	1
1	0	0	1	0	0
1	1	1	0	0	1

notice contrapositive and original implication are (logically) equivalent

(this is really important as a proof technique – we will get onto it later in the course)

Connectives – Implication

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

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

Contrapositive example:

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

- $p \rightarrow q$ if it is sunny, then you will take me to the beach
- $\neg q \rightarrow \neg p$ if you do not take me to the beach, then it is not sunny

these are equivalent (as we have seen from the truth table)

Connectives – Implication

Related 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 – Implication

Relational implications that can be formed from p→q

- converse: q→p
- contrapositive: ¬q→¬p
- inverse: ¬p→¬q

Inverse:

p	q	p→q	¬p	¬q	¬p→¬q
0	0	1	1	1	1
0	1	1	1	0	0
1	0	0	0	1	1
1	1	1	0	0	1

notice that they are different

Connectives – Implication

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

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

Inverse:

p	q	$q \to p$	$\neg p$	$\neg q$	$\neg p \to \neg q$
0	0	1	1	1	1
0	1	0	1	0	0
1	0	1	0	1	1
1	1	1	0	0	1

but the same as the converse

Connectives – Implication

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

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

Inverse example:

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

- $p \rightarrow q$ if it is sunny, then you will take me to the beach
  - can go to the beach when it is not sunny
- $\neg p \rightarrow \neg q$ if it is not sunny, you will not take me to the beach
  - can be sunny and not go to the beach

Connectives – Biconditional

Example:

- let p be the proposition "today is Friday"
- let q be the proposition "it is raining"
- then $p \leftrightarrow q$ ("`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 $(p \lor q) \land r$ is **true**, then r and either p or q must be **true**
  - (either p or q) and r
- if $p \lor (q \land r)$ 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 we will use parentheses between these operators

When in doubt use parentheses

- will avoid both errors and confusion
- using parentheses is good practice when you do exercises about logic

Outline

Propositions

Connectives

Tautologies and contradictions

Logical equivalence

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

Wrapping up

Today we've talked about…

Propositional Logic

- Propositions, truth tables, and connectives

Implication (think of it as a contract)

- Think of it as a contract
- It is central to understanding methods of proof

The contrapositive

- in particular, converting a direct proof to an indirect proof (more later)

Convince yourself that implication & its contrapositive are equivalent