MATHEMATICS
FOR
INFORMATION PROCESSING

Derek G. Bridge
(Office: V/X034; Email: dgb)

Autumn term 1996-97

Course arrangements

- Taught to:
  - B.A./B.Sc. in ITBML (Information Technology, Business Management and a modern foreign Language, first-year students)
  - M.Sc.I.P. (Information Processing and the Man-Machine Interface)
- Contact time:
  - *Lectures* Two $\times$ 50-minute lectures per week.
  - *Problem class* One $\times$ 2-hour problem class per week (compulsory attendance!).
  - *Surgery hour*
- Examination:
  - *Three $\times$ 30 min. mock exams* These will take place during problem classes; compulsory attendance; but the marks 'don't count'.
  - *One $\times$ 90-minute exam* In week 0 of the Spring term, all questions compulsory; 'Examination Notes' included with the exam paper.

Course arrangements, cont'd.

- MIP provides prerequisite material for many courses (IPP, ADP, GNO, DIS, LAI and, ITBML only, MFB).
  - ITBML students must pass MIP+MFB.
  - M.Sc.I.P. students must pass closed papers as a whole.
- Course materials:
  - *'Expanded slides'* On sale from Departmental Office (late mornings). Note that these have deliberate 'gaps' in them, which will be filled in during lectures.
  - *Problem sheets and model answers* Issued during problem classes.
  - *'Examination Notes'* (N.B. You have these now; you know what's on them; you take your own risks about what else to learn.)
  - *Books*
- How to study maths:
  - Read ahead
  - Summarise the 'Expanded slides', learning the essentials by heart
  - Do lots of exercises; then do them again

Models

This course is about **modelling**. Building models is a fundamental human activity used in most, if not all, problem-solving.

> *A model is an artifact that represents a part of reality at some level of abstraction that can be used to explain or predict features of the modelled reality.*

Example:

> A map of some terrain is a model of that terrain.

Think about how the definition of a model applies to maps.

Models, cont'd.

Three sets of rules

To understand a model of a particular type, you have to understand three sets of rules that govern models of that type: formulation, evaluation and manipulation rules.

- **Formulation:** the constructs you can use and how to fit them together. Formulation rules will tell you what basic features (symbols, words, expressions, etc.) you can use and how to combine the basic features to produce a model.

- **Evaluation:** the way you interpret your model in terms of the modelled reality. Evaluation rules will tell you the correspondence between the features of the model and the modelled reality.

- **Manipulation:** the reasoning that you can carry out on the model to give you new information about the modelled reality. Manipulation rules refer to the features of the model, not to the things they represent. (If they referred to the represented reality, then you'd be reasoning with the real system itself, so you wouldn't be using the model!) Because of the correspondence between the model and the modelled reality, manipulation of (i.e. reasoning with) the model yields information about the represented reality.

The formulation and evaluation rules together define a **language** in which models of that type are expressed.

Models: familiar examples

- Maps, plans, blueprints, even drawings, paintings and photos.
- Wind tunnels, reduce-sized replicas of engines, prototypes of devices.
- Business games.
- Computer simulations of economies, of brains, of meteorological conditions, of Post Office queues and of warehouse stock re-ordering.

Of particular interest to us are those in which ‘symbols’ represent objects in the modelled reality.

Models: computerised information system development

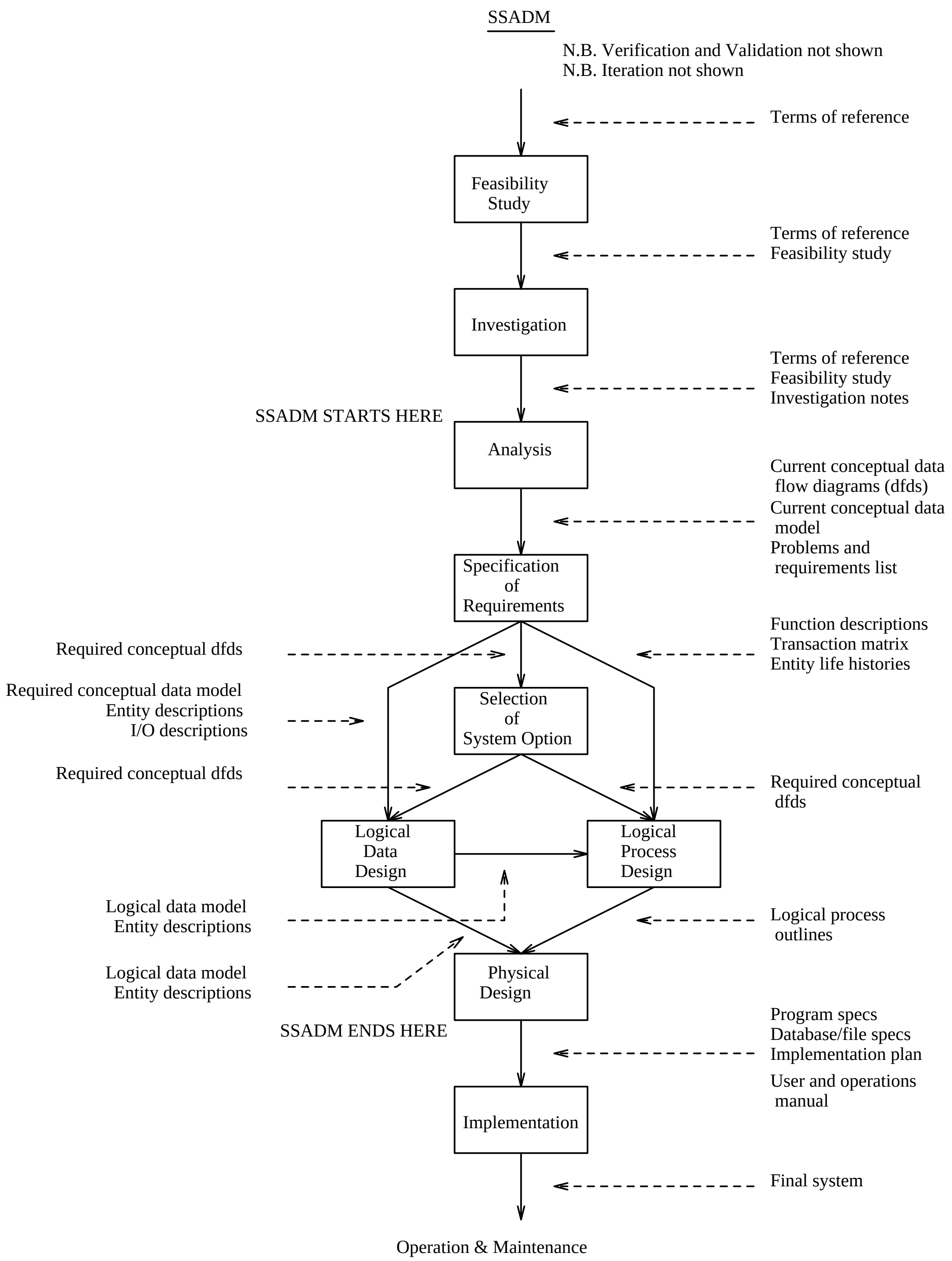
Models have a role to play *during* the development of an information system.

A typical information systems development methodology involves a series of stages, each of which constructs a model.

These models are used

- to confirm understanding (verification and validation of the model)
- to analyse problems and propose solutions
- to aid the development of the next model through a process of elaboration or refinement of the current model

Typical stages of the process are shown on the next slide. These are the steps recommended by the methodology called SSADM (Structured Systems Analysis & Design Method). Details are not important.


SSADM
N.B. Verification and Validation not shown
N.B. Iteration not shown
Terms of reference
Feasibility Study
Terms of reference
Feasibility study
Investigation
Terms of reference
Feasibility study
Investigation notes
SSADM STARTS HERE
Analysis
Current conceptual data flow diagrams (dfds)
Current conceptual data model
Problems and requirements list
Specification of Requirements
Function descriptions
Transaction matrix
Entity life histories
Required conceptual dfds
Required conceptual data model
Entity descriptions
I/O descriptions
Selection of System Option
Required conceptual dfds
Required conceptual dfds
Logical Data Design
Logical Process Design
Logical data model
Entity descriptions
Logical process outlines
Logical data model
Entity descriptions
Physical Design
SSADM ENDS HERE
Program specs
Database/file specs
Implementation plan
User and operations manual
Implementation
Final system
Operation & Maintenance

Models: computerised information systems development, cont'd.

In the development of (computerised) information systems, a failure to meet the customer's requirements (e.g. through a misunderstanding) can result in user dissatisfaction with the final system and even a final system that endangers life.

But, making changes becomes more costly as the development process proceeds.

One key to a solution then is a better characterisation of customer requirements from the outset.

We need models that are complete, consistent, non-redundant, unambiguous and precise

What language (formulation and evaluation rules) should we use to achieve this?

Models: a choice of language

- Using natural languages such as English:
  - expressive, require no special training
  - but ambiguous and vague
- Using diagrammatic notations:
  - easy to understand
  - but lack expressiveness, (often) lack a precise unambiguous meaning
- Using mathematics:
  - unambiguous and precise
  - but less expressive than natural languages and requires courses such as this one to educate its potential users.

Increasingly, the models people will use in information systems development will have mathematical elements to them (even if they are, e.g., diagrammatic but supplemented by bits of English and bits of maths). To be competent to participate in this development process, the professional consultants, managers, systems analysts, systems designers, programmers, etc. of the future must have a knowledge of some maths.

Formal methods, formal languages and formal systems

Formal methods: sensible use of mathematics to assist in the process of information systems development.

To indicate our increased precision, we no longer use my informal terminology (formulation, evaluation, manipulation):

Formal systems:

- **Formal languages:**
  - *Syntax:* precise formulation rules
  - *Semantics:* precise evaluation rules
- **Calculi:** one or more sets of precise manipulation rules (*deductive apparatus*)

Language and metalanguage

Consider learning a new language (even a language such as French, say). How are you taught what words there are and how to put these together to make grammatical sentences (the syntax/formulation)? How are you taught what words and phrases mean (the semantics/evaluation)?

Someone tells you these rules in a language whose rules you already know.

- **Object language:** the language under discussion e.g. the one you are being taught

- **Metalanguage:** the language in which the object language is discussed

For most of us, the usual metalanguage is English. The object language might be French, German, Japanese or the programming language Pascal.

We will use English as the metalanguage for describing our object languages (which are various formal languages from parts of maths).

However, sometimes we'll use symbols as 'abbreviations' for English phrases (e.g. $\mathcal{I}, \mathcal{U}, T, F, \models, \equiv, \vdash$ and $\stackrel{\triangle}{=}$). These are metalanguage symbols.

Beware: always be clear whether a symbol is an abbreviation for English (i.e. part of the metalanguage) or a symbol of the object language.

Some books make matters worse by having symbols that are common to both languages. We will not do this.

Conclusions

> *A firm foundation in mathematics is the hallmark of a professional and an excellent defence against technological obsolescence.*

EXPRESSIONS
AND
VALUES

This section of the course introduces concepts that recur throughout the course: you will see the concepts again when we consider sets, relations, functions and logic. Here, we will use familiar examples drawn from simple arithmetic. But we will look at these examples from a perspective that might feel quite new.

Expressions

Expressions are linguistic objects.

To study them, we must consider:

- *Formulation* of expressions (*syntax*). The symbols we are allowed to use and the rules that govern how those symbols may be put together to form syntactically well-formed expressions.

- *Evaluation* of expressions (*semantics*). There are objects 'out there'. Evaluation is used to:

  - determine what object an expression **refers** to;

  - determine what object an expression **denotes**;

  - determine what object an expression has as its **value**.

- *Manipulation* of expressions (*calculus*). Transforming expressions into different expressions. Various laws and definitions will tell us what manipulations we can legitimately carry out to ensure that the new expressions are well-formed and have some significant semantic relationship with the original expressions.

We consider these for **arithmetic expressions**. Our treatment of all these is fairly informal at this stage.

Formulation

1. The simplest expressions (for now) comprise just a **numeral**:

$$0, 1, 2, 3, \ldots, 10.2, 10.34$$

2. More complex expressions comprise **operators** and their **operands** (or **arguments**), where the operands are themselves expressions. The operators we use (for now) are

$$+, -, *, /, \uparrow, \mathsf{sin}, \mathsf{cos}, \mathsf{tan}, !,$$

$$=, <, \leq, >, \geq, \neq, \not<, \not\leq, \not>, \not\geq$$

(Some of these will be unfamiliar: their meanings are given later.)

To build expressions using these, we need to know the **arity** and **fixity** of each operator.

It is important to note that operands can be *any* expressions, i.e. simple numerals

$$3 + 2$$

or expressions that themselves consist of an operator applied to operands

$$2 * \mathsf{sin}(90)$$

$$2 * \mathsf{sin}(3 * 90)$$

3. Any expression may be enclosed in parentheses and will still be an expression.

$$(15)$$

$$(2 * 4) + 6$$

$$(2 * \mathsf{sin}(90))$$

$$(((((2 * (4 + 6))))))$$

4. No other string of symbols is an arithmetic expression.

We will later allow ourselves the possibility of introducing new symbols into these expressions, and this causes us to revise these rules on a later slide.

Formulation: operator arity

The **arity** of an operator is the number of operands it requires.

- Unary (or 1-ary) operators
  - e.g. $\mathsf{sin}, \mathsf{cos}, \mathsf{tan}, !$
  - require one operand, e.g. $\mathsf{sin}(90), 3!$
- Binary (or 2-ary) operators
  - e.g. $+, -, *, /, \uparrow, =, <, \leq, >, \geq, \neq, \not<, \not\leq, \not>, \not\geq$
  - require two operands, e.g. $2 + 3, 2 \uparrow 3$
- Ternary (3-ary) operators, quaternary (4-ary) operators, etc. are also possible
  - you will encounter some on the exercise sheets.

Formulation: operator fixity

The **fixity** of an operator determines the placement of its operands.

- **Prefix** operators, e.g. $\mathsf{sin}, \mathsf{cos}, \mathsf{tan}$
  - precede their operands, e.g. $\mathsf{sin}(90)$
- **Infix** operators
  - e.g. $+, -, *, /, \uparrow, =, <, \leq, >, \geq, \neq, \not<, \not\leq, \not>, \not\geq$
  - come between their operands, e.g. $2 + 3$, $2 * 3$, $2 \uparrow 3$
- **Postfix** operators, e.g. !
  - come after their operand(s), e.g. $3!$

In everyday maths, infix appears to be most common for binary operators and prefix for unary ones. When we get on to n-ary operators for $n > 2$ (i.e. 3-ary, 4-ary, etc.), we find that prefix is most common. Postfix looks quite unfamiliar, but perhaps it isn't: on some calculators, sin and other similar operators are postfix. You can't press the $\boxed{\text{sin}}$ key and *then* $\boxed{9}\boxed{0}$ to compute $\mathsf{sin}(90)$. Instead, you press $\boxed{9}\boxed{0}$ *then* $\boxed{\text{sin}}$.

Of course, it's all a matter of convention. We could decide henceforth to have $+$ as a prefix operator

$$+ \ 2 \ 3$$

or a postfix operator (as was the case in some early calculators):

$$2 \ 3 \ +$$

Evaluation: simple expressions

In English, the simplest expressions are names; they evaluate to people, dogs, towns, etc.

Sweep

The expression “Sweep” (a very simple linguistic expression, a name) evaluates to a particular object (a furry one).

We also use names in maths.

- Cast your minds back to computing the area of circles at school — the area is $\pi\times$ the square of the radius;
  π is the name of a number:

  π the number **3.141592...**

- One very slippery idea is that *numerals* are linguistic expressions that denote *numbers*:

  5 the number **5**

Here, I've used different fonts to try to emphasise the difference between numerals (linguistic expressions) and numbers (objects that are ‘out there’). However, I will not do this henceforth.

Evaluation: more complex expressions

Operators are symbols that denote **operations**.

- The operations construct new objects out of existing objects, transform an object from being in one state to being in another state, or relate several objects to each other.
- To evaluate an expression that contains an operator, we work out the values of the operator's operands, we work out what operation the operator denotes, and we apply this operation to the values of the operands; the resulting object is the value of the whole expression.

There are lots of examples in English, e.g.

the_best_friend_of Sooty

Here there is an operator 'the_best_friend_of'; the operation that this operator denotes is the one that, given a creature, works out that creature's best friend. In this example, the operation is applied to the denotation of the name 'Sooty', i.e. it is applied to the bear Sooty, whose best friend is Sweep; so the value of the whole expression is Sweep.

A maths example:

$2 + 5$ the number 7

+ is the operator which denotes the operation of addition of numbers. Here, the numerals 2 and 5 evaluate to the numbers 2 and 5 respectively. Adding these two numbers gives the number 7, so the number 7 is the value of this expression.

Evaluation: denotations of operators

We can now say what operations the 'standard' operators stand for.

$+$	Addition of numbers.
$-$	Subtraction of the second number from the first.
$*$	Multiplication of numbers.
$/$	Division of the first number by the second.
$\uparrow$	Exponentiation. See the next slide.
$\mathsf{sin}, \mathsf{cos}, \mathsf{tan}$	Sine, cosine and tangent from Geometry. Not important in this course — I just needed some examples of prefix operators!
!	Factorial. See the next slide.

Note that we are using the word 'operation' informally, quite broadly and in a more neutral way than is sometimes the case elsewhere in mathematics. Many books reserve the word 'operation' for the more restricted case where the operands and result are all of the same *type*. ('Type' is defined below.)

Evaluation: denotations of operators, cont'd.

Exponentiation

$m \uparrow n$ is more commonly written m^n in maths.

The expressions $m \uparrow n$ or m^n are read 'm to the power n' or 'm raised to the power n'.

$$m^n \quad = m \uparrow n \quad = m * m * m * \ldots * m$$
(with m occurring n times)

2^3	$= 2 \uparrow 3$	$= 2 * 2 * 2$	which evaluates to the number 8
3^2	$= 3 \uparrow 2$	$= 3 * 3$	which evaluates to the number 9
3^1	$= 3 \uparrow 1$	$= 3$	which evaluates to the number 3
3^0	$= 3 \uparrow 0$	$= 1$	which evaluates to the number 1

It is taken that *any* number, m, raised to the power zero, $m \uparrow 0$ or m^0, evaluates to the number 1.

Factorial

The expression $n!$ is read 'n factorial'.

$$n! = n * (n - 1) * (n - 2) * \ldots * 2 * 1$$

$4!$	$= 4 * 3 * 2 * 1$	which evaluates to the number 24
$3!$	$= 3 * 2 * 1$	which evaluates to the number 6
$2!$	$= 2 * 1$	which evaluates to the number 2
$1!$	$= 1$	which evaluates to the number 1
$0!$	$= 1$	which evaluates to the number 1

It is taken that $0!$ evaluates to the number 1.

Evaluation: denotations of operations, cont'd.

$=$	Equality.
$\neq$	Not equal.
$<, \ngeq$	The first number is less than the second.
$\leq, \ngtr$	The first number is less than or equal to the second.
$>, \nleq$	The first number is greater than the second.
$\geq, \nless$	The first number is greater than or equal to the second.

Question: what is the difference between the operators $+, -, *, /, \uparrow, \mathsf{sin}, \mathsf{cos}, \mathsf{tan}$ and ! and the operators $=, <, \leq, >, \geq, \neq, \nless, \nleq, \ngtr\ \ngeq$?

Evaluation: types

Values fall into classes of similar kinds of objects, these classes being called **types**.

An expression containing an operation can only be evaluated if the operands denote objects of the types required by the operation.

The expression as a whole will return a value of a certain type.

For example, + denotes an operation both of whose operands must denote numbers and the resulting value is a number.

$/$ (division), however, has a more specific type: the first argument can denote any number, but the second argument can denote any number other than zero.

The operands for $>$ can denote any numbers, but the result is either true or false. (We say the result is of type **Boolean** or is a **truth-value**.)

So, in this course,

$$3 * (4 > 2)$$

is syntactically well-formed but cannot be evaluated because of a *type violation*. $(4 > 2)$ evaluates to a Boolean (in this case, true), but this cannot be the argument to $*$.

Evaluation: denoting the same object

In English, several names or more complex expressions can denote the same object:

Derek
DGB
the_lecturer_of MIP

Similarly in maths, several expressions (whether simple or complex) can denote the same object:

5	five	
cinq	funf	
V	101	
卌	$2+3$	the number 5
$5+0$	$6-1$	
$(2 \uparrow 2)+1$	$((2*4)+2)/(3-1)$	

Evaluation: ambiguity

When an operator's operands are themselves complex expressions, how should evaluation proceed?

Consider

$$2 + 3 * 4$$

This evaluates either to the number 20 or to the number 14. It depends on which operator you evaluate first. If you apply $+$ to 2 and 3 to give 5 and then apply $*$ to this and to 4 the whole expression has 20 as its value; however, if first you apply $*$ to 3 and 4 to give 12 and then apply $+$ to 2 and 12 the whole expression has 14 as its value.

Consider also

$$5 - 2 + 3$$

This evaluates either to the number 6 or to the number 0. $5 - 2$ gives 3, then $3 + 3$ gives 6; alternatively, $2 + 3$ gives 5 and $5 - 5$ gives 0.

Even though this was supposed to be a formal language we have AMBIGUITY!

Evaluation: using parentheses

Parentheses provide one way to disambiguate. They make the evaluation order explicit:

$(2+3)*4$	evaluates to the number 20
$2+(3*4)$	evaluates to the number 14
$(5-2)+3$	evaluates to the number 6
$5-(2+3)$	evaluates to the number 0
$(2+3)*(4\uparrow 2)$	evaluates to the number 80
$(2+(3*4))\uparrow 2$	evaluates to the number 196
$2+((3*4)\uparrow 2)$	evaluates to the number 146

We will use parentheses liberally in the formal languages we introduce. But there are ways to avoid *some* parentheses. We illustrate one way on the next slide.

Evaluation: precedence and associativity

One way to do away with parentheses in *some* expressions is to have a convention about which of two operators is applied first: these conventions are known as **operator precedence rules** and **operator associativity rules**.

For arithmetic, the conventional **operator precedences** are:

$$\begin{array}{cl} \uparrow & \text{highest precedence} \\ *, / & \\ +, - & \\ =, <, \leq, >, \geq, \neq, \not<, \not\leq, \not>, \not\geq & \text{lowest precedence} \end{array}$$

So

$$2 + 3 * 4$$

is evaluated as

$$2 + (3 * 4)$$

because $*$ has higher precedence than $+$.

And

$$2 + 3 * 4 \uparrow 2$$

is evaluated as

$$2 + (3 * (4 \uparrow 2))$$

because $\uparrow$ has highest precedence, and then $*$ has higher precedence than $+$.

Evaluation: precedence and associativity, cont'd.

For arithmetic, the conventional **operator associativities** are:

Associate to the left	Associate to the right
$*, /, +, -,$ $=, <, \leq, >, \geq, \neq, \not<, \not\leq, \not>, \not\geq$	$\uparrow$

So

$$5 - 2 + 3$$

is evaluated as

$$(5 - 2) + 3$$

because $+$ and $-$ associate to the left.

And

$$2 \uparrow 3 \uparrow 2$$

is evaluated as

$$2 \uparrow (3 \uparrow 2)$$

because $\uparrow$ associates to the right.

How do you signal that you want an expression to be evaluated in an order that is different from that given by the above rules? You override the precedence and associativity rules with parentheses.

E.g. you would use

$$(2 + 3) * 4$$

$$5 - (2 + 3)$$

$$(2 \uparrow 3) \uparrow 2$$

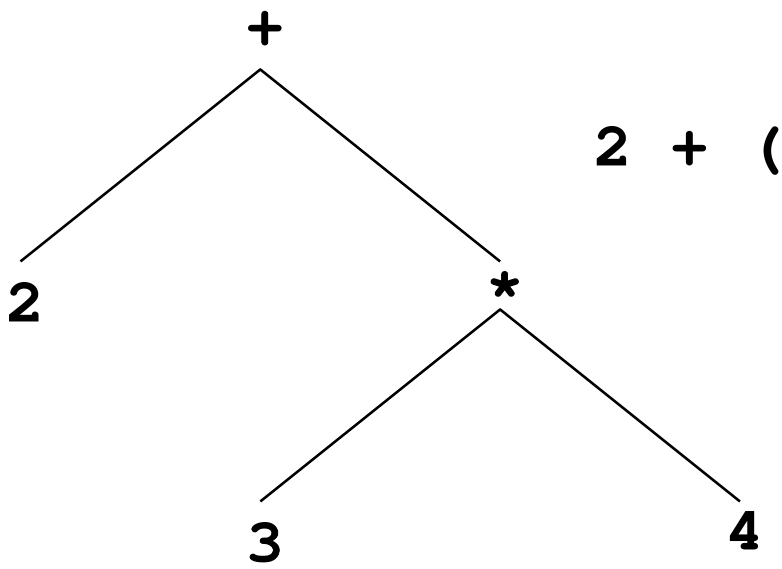
to obtain different evaluation orders.

Evaluation: trees

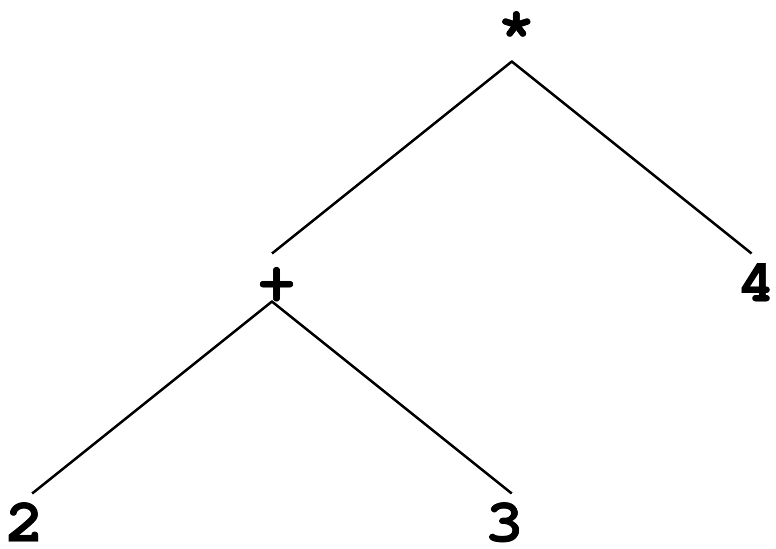
Trees give us another unambiguous representation.

In Computer Science, trees grow from the top of the page towards the bottom!

$2 + 3 * 4$ can be depicted as:



`2 + (3 * 4)`



`(2 + 3) * 4`

Definitions: new names

Definitions allow us to associate a **name** with a value

- the new name might be more concise or more mnemonic than using a whole expression

- the definition **binds** the name to the value.

We state the new name and an expression whose value will be associated with the name.

$$\begin{aligned} \mathit{No\_of\_Lectures} &\triangleq 18 \\ \mathit{Lecture\_Length} &\triangleq 60 - 10 \\ \mathit{Contact\_Time} &\triangleq \mathit{No\_of\_Lectures} * \mathit{Lecture\_Length} \end{aligned}$$

The symbol $\triangleq$ is part of the metalanguage. Using it enables us to extend the object language. For the above examples, the new names that can henceforth be used in the object language are *No_of_Lectures*, *Lecture_Length* and *Contact_Time* and these names are bound to (i.e. evaluate to) the same objects as the expressions 18, $60 - 10$ and $\mathit{No\_of\_Lectures} * \mathit{Lecture\_Length}$ evaluate to, i.e. the numbers 18, 50 and 900.

Note how a definition may use any names defined in earlier definitions.

Maths textbooks sometimes use other notation to bind names to values, e.g.

$$\begin{aligned} \text{Let } \mathit{No\_of\_Lectures} &\text{ be } 18 \\ \text{Let } \mathit{No\_of\_Lectures} &= 18 \\ \mathit{No\_of\_Lectures} &=_d 18 \end{aligned}$$

Now we can write expressions that use these new names.

$\mathit{Lecture\_Length} + 10$	evaluates to the number 60
$\mathit{Contact\_Time} * 12$	evaluates to the number 10800

Definitions: new names, cont'd.

Names can also be associated with operations. We use $\triangleq$ as before, but we must also show the arity and fixity of the new operator and the way the operands participate in the operation.

In these definitions, we use **parameters**: (temporary) names that can be taken to stand for an arbitrary value to which the operation is applied.

Examples (x and y are the parameters)

$$
\begin{array}{rcl}
\textit{double}(x) & \triangleq & x + x \\
\textit{treble}(y) & \triangleq & 3 * y \\
x \heartsuit y & \triangleq & y - x \\
\textit{funnysubtract}(x, y) & \triangleq & x \heartsuit y \\
x \spadesuit y & \triangleq & (x \uparrow 2) + (y \uparrow 2) \\
\textit{sumdoubles}(x, y) & \triangleq & \textit{double}(x) + \textit{double}(y) \\
\textit{sixtimes}(x) & \triangleq & \textit{double}(\textit{treble}(x))
\end{array}
$$

Each parameter is introduced 'temporarily' for that definition. Hence, the x and y used in one definition are unrelated to the x and y used in another definition. See the definition of **scope** for a more precise statement of this.

Definitions : new names, cont'd.

Further examples (operations that return true or false, rather than returning a number)

$$\begin{array}{rcl} \textit{positive}(x) & \triangleq & x > 0 \\ \textit{negative}(x) & \triangleq & x < 0 \end{array}$$

The new operators can be used in the formulation of expressions; the operations to which they are bound are used in the evaluation of those expressions.

$2 \heartsuit 3$	evaluates to 1
positive(double(2) + treble(2))	evaluates to true

Formulation (revised)

1. The simplest expressions comprise just a **numeral** or a **name** that we define using $\triangleq$.

2. More complex expressions comprise **operators** and their **operands** (or **arguments**), where the operands are themselves expressions.

   The operators we use are

$$+, -, *, /, \uparrow, \mathsf{sin}, \mathsf{cos}, \mathsf{tan}, !,$$

$$=, <, \leq, >, \geq, \neq, \not<, \not\leq, \not>, \not\geq$$

   and any we define using $\triangleq$.

3. Any expression may be enclosed in parentheses and will still be an expression.

4. No other string of symbols is an arithmetic expression.

Definitions: occurrences of names

We have introduced new names in two ways:

- the 'main' name in a definition using $\triangleq$
- the parameters in the definition of new operators

Both of these occurrences of names are said to be **binding occurrences** of the name. E.g. in the definitions

$$\begin{aligned} \textit{No\_of\_Lectures} &\triangleq 18 \\ \textit{double}(x) &\triangleq x + x \end{aligned}$$

the occurrences of *No_of_Lectures*, *double* and the x that is on the left hand side of $\triangleq$ are binding occurrences.

Other occurrences of names are called **applied occurrences**. E.g. in the definition

$$\textit{double}(x) \triangleq x + x$$

the two occurrences of x on the right hand side of $\triangleq$ are applied occurrences. Also, in the expression

$$\textit{double}(\textit{No\_of\_Lectures})$$

the occurrences of *double* and *No_of_Lectures* are both applied occurrences.

Definitions: scope

The binding of a name to a value is not normally intended to be in force for all time. Rather, the association is intended to hold for a certain context. The textual context of a definition is known as the **scope** of that definition.

- In the case of parameters, the scope is limited to the right hand side of the definition.

- In the case of the main name introduced by a definition, the scope is usually some obvious section of subsequent text. It might be a whole exam paper/exercise sheet, an exam question/exercise question, a program specification, one module of a program specification, one overhead in a lecture, all the overheads of a lecture, a section of a book, a chapter of a book, a whole book, etc.

An applied occurrence of a name is **bound** if the applied occurrence falls within the scope of a binding occurrence for that name.

An applied occurrence of a name is **free** if the applied occurrence does not fall within the scope of a binding occurrence for that name.

We usually assume that for arithmetic expressions the standard operators ($+, -, *, /, \uparrow, \mathsf{sin}, \mathsf{cos}, \mathsf{tan}, !, =, <, \leq, >, \geq, \neq, \not<, \not\leq, \not>, \not\geq$) and the standard names (numerals) $0, 1, 2, 3, \ldots, 10.2, 10.34$, etc. are bound: their binding occurrences are lost in the mists of time, and their scope is pretty much permanent.

We shall usually only write expressions in which every name is bound.

Definitions: scope examples

You are advised to annotate this sheet of your notes during the lecture. We will indicate binding occurrences and scopes and, for each applied occurrence, we will say whether that occurrence is bound or free.

On this overhead,

$$\begin{array}{rcl}
\textit{DayLength} & \triangleq & 24 \\
\textit{halve}(x) & \triangleq & x/2 \\
\textit{quarter}(x) & \triangleq & x * 0.25 \\
\textit{eighth}(x) & \triangleq & \textit{halve}(\textit{quarter}(y))
\end{array}$$

Here are expressions for evaluation:

$$\textit{halve}(\textit{DayLength}) + 10$$

$$\textit{quarter}(z)$$

$$\textit{even}(32)$$

Manipulation

Manipulation is transforming one expression into another expression that has a significant semantic relationship with the original expression. For now, the 'significant semantic relationship' will be that the new expression will evaluate to the same value as the original expression.

Crucially, in manipulation the transformations are done without reference to the meanings (values) of the expressions involved i.e. we do no evaluation. All that happens is textual substitution of one expression for another expression.

We will do manipulation with reference to **laws**.

Manipulation: laws

An example of a **law**:

For all arithmetic expressions AE_1 and AE_2,

$$AE_1 + AE_2 = AE_2 + AE_1$$

Think of the names AE_1 and AE_2 as being a bit like parameters: (temporary) names that can be taken to stand for an arbitrary value. But note that we are saying that the expression that results from consistently substituting *any* number-denoting expressions for these names will be a true equality. This law acts as a template for an (infinite) number of true statements of equality. It says all of the following (and more) all in one go:

$$\begin{array}{rr} 0 + 1 = 1 + 0 & 5 + 4 = 4 + 5 \\ 2 + 3 = 3 + 2 & (2 * 3) + 4 = 4 + (2 * 3) \end{array}$$

The law seems quite reasonable, but actually it's an amazingly bold statement: it says that it holds for the sum of *any* pair of arithmetic expressions! Strictly, laws like these therefore need to be proved (although we'll take them on board without proof), otherwise how do we know that the manipulations that they license will guarantee that the new expression is indeed equal to the original one?

This law allows us to swap the order of the operands to an addition operator in any expression or subexpression.

And, in fact, there are lots of other laws, giving even more fodder to the manipulation process.

Manipulation: laws of the algebra of arithmetic

For brevity and simplicity, we will only present laws that pertain to addition and multiplication.

For all arithmetic expressions AE_1, AE_2 and AE_3,...

Commutativity of $+$ and $*$

$$AE_1 + AE_2 = AE_2 + AE_1$$

$$AE_1 * AE_2 = AE_2 * AE_1$$

This is the law from the previous slide, and the corresponding law for the multiplication operation.

Note that the law does not hold for subtraction or division: you cannot in general swap their operands and get the same result. For example, $3 - 2 \neq 2 - 3$. This **counterexample** disproves the statement that for all AE_1 and AE_2, $AE_1 - AE_2 = AE_2 - AE_1$.

Associativity of $+$ and $*$

$$AE_1 + (AE_2 + AE_3) = (AE_1 + AE_2) + AE_3$$

$$AE_1 * (AE_2 * AE_3) = (AE_1 * AE_2) * AE_3$$

The first of these laws say that if you're adding more than two things, it doesn't matter how you group them together to carry out the operations: you can add the second and third and add the first to this, or you can add the first and second and add the third to that. The second law says the same thing but for multiplication.

Of course, since these laws say that order doesn't matter in these cases, this does mean that the parentheses in expressions of this kind aren't really necessary.

Manipulation: laws of the algebra of arithmetic, cont'd.

For all arithmetic expressions AE, AE_1, AE_2 and AE_3...

Distributivity of $*$ over $+$

$$AE_1 * (AE_2 + AE_3) = (AE_1 * AE_2) + (AE_1 * AE_3)$$

For example, $3 * (2 + 4) = (3 * 2) + (3 * 4)$. Remember, however, that this example is only a demonstration; it is not a **proof**. Demonstrating truth for one example is not the same as proving that it is true for all arithmetic expressions.

Note that $+$, on the other hand, is not in general distributive over $*$. The following is a counterexample, $3 + (2 * 4) \neq (3 + 2) * (3 + 4)$.

Identities

$$0 + AE = AE$$

$$1 * AE = AE$$

These two laws show that there are values that, when used as one of the operands in these two operations, make the result of the whole expression the same as the value of the other operand.

There are other laws, but this is enough.

Manipulation: using the laws in proofs

We can use laws to prove that two expressions are equal. Asked to do this, we take one of the expressions and make appropriate textual substitutions until it is the same as the other expression. (Sometimes it helps to do a few manipulations on one expression and a few on the other and show they end up being the same.)

Example: to show that for all arithmetic expressions E_1 and E_2,

$$E_1 * (E_2 + 1) = (E_1 * E_2) + E_1$$

Take the left hand side (LHS) and manipulate it until it is the same as the right hand side (RHS):

$$\begin{aligned}
\text{LHS} \quad &= E_1 * (E_2 + 1) \\
&= (E_1 * E_2) + (E_1 * 1) \\
&= (E_1 * E_2) + (1 * E_1) \\
&= (E_1 * E_2) + E_1 \\
&= \text{RHS}
\end{aligned}$$

Each line needs some justification: add this during the lecture.

Manipulation: another example

Show that for all arithmetic expressions E_1, E_2, E_3 and E_4,

$$\begin{aligned}&E_1 + (E_2 + (E_3 + E_4)) = \\ &\qquad E_4 + (E_3 + (E_2 + E_1))\end{aligned}$$

Take the left hand side (LHS) and manipulate it until it is the same as the right hand side (RHS):

$$\begin{aligned}\text{LHS} \quad &= E_1 + (E_2 + (E_3 + E_4)) \\ &= (E_1 + E_2) + (E_3 + E_4) \\ &= (E_2 + E_1) + (E_3 + E_4) \\ &= ((E_2 + E_1) + E_3) + E_4 \\ &= (E_3 + (E_2 + E_1)) + E_4 \\ &= E_4 + (E_3 + (E_2 + E_1)) \\ &= \text{RHS}\end{aligned}$$

Each line needs some justification: add this during the lecture.

Manipulation: another example

Show that for all arithmetic expressions E_1, E_2, E_3 and E_4,

$$
\begin{aligned}
&E_1 * (E_2 + (E_3 + E_4)) = \\
&\qquad ((E_1 * E_2) + (E_1 * E_3)) + (E_1 * E_4)
\end{aligned}
$$

Take the left hand side (LHS) and manipulate it until it is the same as the right hand side (RHS):

$$
\begin{aligned}
\text{LHS} \quad &= E_1 * (E_2 + (E_3 + E_4)) \\
&= (E_1 * E_2) + (E_1 * (E_3 + E_4)) \\
&= (E_1 * E_2) + ((E_1 * E_3) + (E_1 * E_4)) \\
&= ((E_1 * E_2) + (E_1 * E_3)) + (E_1 * E_4) \\
&= \text{RHS}
\end{aligned}
$$

Each line needs some justification: add this during the lecture.

SET THEORY

In this section of the course, the notions of formulation, evaluation and manipulation will resurface (though we'll probably make less of a meal of them). What's new is that, instead of dealing with numbers, we will be dealing with a new kind of object: sets. These are fundamental in maths (arguably more fundamental than numbers!), and essential in information processing.

Sets: what are they?

A **set** is a collection of objects.

- The objects can be any identifiable things of interest, whether physical or abstract.

- The objects in a set are called its **members** or **elements**.

- Examples:
  - The set of people in this room.
  - The set of books in the Library.
  - The set of course units offered by this University.
  - The set of Computer Science Department course units.
  - The set of positive integers: 1, 2, 3,…
  - The set made up of the four numbers 5, 7, 10 and 103.
  - The set containing just the number 3.
  - The set made up of the number 3, the letter *‘a’*, this room, last Tuesday and an aardvark.

- An object can belong to more than one set.

- The members of a set do not have to be the same kind of thing, though in information processing they usually are.

Simple expressions that denote sets

There are two ways to define a set.

Extensional definition: *enumerate* the members (separate by commas, enclose in braces).

- The set made up of the four numbers 5, 7, 10 and 103:

$$\{5, 7, 10, 103\}$$

- The vowels:

$$\{a, e, i, o, u\}$$

- Autumn term courses shared by 1st year ITBML and MScIP:

$$\{\mathit{IPP}, \mathit{MIP}\}$$

This method is feasible only for small sets.

Simple expressions that denote sets

Intensional definition: specify a property possessed by all the members and no other objects, the **defining property**.

Use a 'temporary' name which stands for any member being specified. The binding occurrence of the 'temporary' name precedes a vertical bar (read 'such that'). The property, expressed in terms of the 'temporary' name, using an appropriate mix of formal and informal expressions, follows the vertical bar and is the scope of the 'temporary' name.

- $\{x \mid x$ is a number and $x \geq 0$ and $x < 4\}$

  i.e. the set of objects x such that x is a number greater than or equal to 0 and less than 4. (Extensionally, $\{0, 1, 2, 3\}$.)

- $\{l \mid l$ is a lecturer in the Comp. Sci. Dept. at the Uni. of York$\}$

  (Extensionally, $\{$*DGB, KCM,...*$\}$.)

- The defining property can mention other sets:

  $\{c_1 \mid c_1$ is a city of a country c_2, where c_2 is a country in the set $\{$*France, Germany*$\}\}$

You can use any 'temporary' name, e.g. n or *num* or *number* would do just as well as x. This name allows (repeated) unambiguous reference to an object.

You can mix formal and informal expressions in the defining property, but there is a preference for it to be as formal as possible for precision and absence of ambiguity. For example, there is a problem in interpreting the informal definition $\{w \mid w$ is a word of English whose length is between 1 and 4$\}$. What is the problem?

Two fundamental properties of sets

1. Each element appears only once. An object can only be a member of a set in one way; there is no notion of 'multiple membership'.

   A source of confusion is that in *defining* a set, particularly by enumeration, an object may occur several times. But remember to distinguish the expression from its value. For example, the two expressions $\{0, 1, 2\}$ and $\{0, 1, 1, 2\}$ both have the same set as their value, the one that contains the objects 0, 1, and 2. Clearly, it is better to avoid expressions such as $\{0, 1, 1, 2\}$ that exhibit redundancy.

2. The elements have no order. There is no notion of one element 'coming before' or 'coming after' another in a set.

   Again, one can be confused when *defining* a set by enumeration because one has to write the elements down in some order. But again remember to distinguish the expression from its value. For example, the two expressions $\{0, 1, 2\}$ and $\{2, 1, 0\}$ both have the same set as their value, the one that contains the objects 0, 1, and 2.

The above two properties emphasise the importance of distinguishing an expression that defines a set from its value (as with expressions of arithmetic).

For example, the following different expressions all denote the same set:

$$\begin{array}{cc} \{2, 3, 4\} & \{3, 2, 4\} \\ \{2, 2, 3, 4, 4, 4\} & \{n \mid 2 \leq n \text{ and } 2 \leq 4\} \\ \{m \mid 2 \leq m \text{ and } 2 \leq 4\} & \{n \mid 1 < n \text{ and } n < 5\} \end{array}$$

(assuming n and m only range over 'whole' numbers in the last three).

Set theory: formulation

1. The simplest expressions are extensional and intensional set definitions.

2. More complex expressions comprise **operators** and their **operands**, where the operands are themselves expressions.

   The operators (each of which is fully defined on subsequent slides) are
   (i) the following binary infix operators

   $$=, \neq, \in, \notin, \subseteq, \subset, \not\subseteq, \not\subset, \supseteq, \supset, \not\supseteq, \not\supset, \cup, \cap, \setminus$$

   (ii) the following unary prefix operators

   $$\wp, \mathit{card}$$

   (iii) the following unary postfix operator

   $$'$$

   (These operators do not all have the same type.)

3. Any expression may be enclosed in parentheses and will still be an expression.

4. No other string of symbols is an expression of set theory.

Definitions of new names

We can also name sets and new operations using $\stackrel{\triangle}{=}$, and use the new names in expressions.

E.g.

$$\mathit{DGB\_Courses} \stackrel{\triangle}{=} \{\mathit{MIP}, \mathit{DEC}, \mathit{NLP}\}$$

Some names are already in common use:

$\mathbb{Z}$	the **integers** (i.e. all 'whole' numbers)
$\mathbb{N}$	the **natural numbers** (i.e. all integers ≥ 0)
$\mathbb{P}$	the **positive integers** (i.e. all integers > 0)
$\mathbb{Q}$	the **rationals**
$\mathbb{R}$	the **reals**

There are also a couple of 'special' sets that are introduced on the next slide and have names that are already in common use.

Special sets

The **universal set**

- The set of all elements of interest to your present purpose.
- The universal set is usually written U.
- Sometimes it is defined explicitly; sometimes it is supposed to be 'obvious'.
- E.g. the universal set for a treatment of numbers might be the set of natural numbers.
- E.g. the universal set for one description of the University might be all students, all lecturers and all courses.

The **empty set**

- The empty set has no elements.
- The empty set is usually written $\{\}$ or $\emptyset$.

Set equality and set inequality

$=$ denotes the binary infix operation that returns true if its operands denote the same set and false otherwise.

$\neq$ denotes the binary infix operation that returns true if its operands denote different sets and false otherwise.

E.g.

$$\{2, 3, 4\} = \{n \mid 1 < n \text{ and } n < 5\}$$

$$\{t \mid t \text{ is a U.K. university town}\} \neq$$

$$\{c \mid c \text{ is a U.K. cathedral city}\}$$

These operations return true or false, as do $\in, \notin, \subseteq, \subset, \not\subseteq, \not\subset, \supseteq, \supset, \not\supseteq$ and $\not\supset$, which we look at next.

Operations that take in sets and return sets, and operations that take in sets and return numbers, will come later.

Set membership

For any set A and any object a, either a is a member of A or it is not.

- To indicate that a is a member of A, we write

$$a \in A$$

- To indicate that a is not a member of A, we write

$$a \notin A$$

Examples

- $1 \in \{0, 1, 2\}$
- $4 \in \{x \mid 1 \leq x \text{ and } x \leq 4\}$
- $4 \in \mathbb{N}$
- *Rome* $\in \{c \mid c \text{ is a capital city}\}$
- $3 \notin \{0, 1, 2\}$
- $10.3 \notin \mathbb{N}$
- *York* $\notin \{c \mid c \text{ is a capital city}\}$

Subset

For any two sets A_1 and A_2, A_1 is **included in** A_2, or A_1 is a **subset** of A_2, if and only if for all a, if $a \in A_1$ then $a \in A_2$.

- To indicate that A_1 is a subset of A_2, we write

$$A_1 \subseteq A_2$$

- To indicate that A_1 is not a subset of A_2, we write

$$A_1 \nsubseteq A_2$$

Examples

- $\{0, 2\} \subseteq \{0, 1, 2\}$ because every member of the former set is also a member of the latter set.

- $\{$*Rome, Paris*$\} \subseteq \{c \mid c$ is a capital city$\}$

- $\{0, 1, 3\} \nsubseteq \{0, 1, 2\}$

- $\{0, 1, 2\} \subseteq \{0, 1, 2\}$ because every member of the former set is also a member of the latter set, which is all that the definition requires.

To make explicit the case where the second set does have some *extra* members (i.e. unlike the last example above), we use the notion of **proper subset**.

Proper subset

For any two sets A_1 and A_2, A_1 is a **proper subset** of A_2 if and only if $A_1 \subseteq A_2$ and there exists a such that $a \in A_2$ but $a \notin A_1$.

We write $A_1 \subset A_2$ and $A_1 \not\subset A_2$.

Examples

- $\{0, 2\} \subset \{0, 1, 2\}$
- $\{$*Rome, Paris*$\} \subset \{c \mid c$ is a capital city$\}$
- $\{0, 1, 3\} \not\subset \{0, 1, 2\}$
- $\{0, 1, 2\} \not\subset \{0, 1, 2\}$
- $\mathbb{P} \subset \mathbb{N} \subset \mathbb{Z} \subset \mathbb{Q} \subset \mathbb{R}$

Set supersets and proper supersets

Superset

$$A_1 \supseteq A_2 \stackrel{\triangle}{=} A_2 \subseteq A_1$$

Proper superset

$$A_1 \supset A_2 \stackrel{\triangle}{=} A_2 \subset A_1$$

$\not\supseteq$ and $\not\supset$ are defined as you might expect.

Examples

- $\{0, 1, 2\} \supseteq \{0, 1\}$
- $\{0, 1, 2\} \supset \{0, 1\}$
- $\{0, 1, 2\} \supseteq \{0, 1, 2\}$
- $\{c \mid c$ is a capital city$\} \supseteq \{$*Rome, Paris*$\}$
- $\{0, 1\} \not\supseteq \{0, 1, 2\}$
- $\{0, 1, 2\} \not\supset \{0, 1, 2\}$

Subsets: special cases

For any set A,

$$A \subseteq A$$

(because the definition of $\subseteq$ simply requires every member of the first set to be a member of the second and this is clearly the case if the sets are equal). Of course $A \not\subset A$.

For any set A,

$$\emptyset \subseteq A$$

We can explain this one in two ways. (i) Given any set A, as $\emptyset$ has *no* members then it must be true that every member a of $\emptyset$ is a member of A. This property is *vacuously true* because there are no members. (ii) For any two sets A_1 and A_2, on what condition can we say that A_1 is not a subset of A_2? There must exist an object a such that $a \in A_1$ but $a \notin A_2$. But if A_1 is the empty set (no members), then there can't be any such object a.

Of course, this means that $\emptyset \subseteq \emptyset$ (although $\emptyset \not\subset \emptyset$). And, there is no other set that is a subset of $\emptyset$.

Set equality revisited

Now that we have the notions of set membership and subset, we can look at $=$ in a new light.

Using $\in$

For any two sets A_1 and A_2,

$A_1 = A_2$ if and only if for all a,

$$a \in A_1 \text{ if and only if } a \in A_2$$

i.e. they are equal if and only if they have exactly the same members.

Using $\subseteq$

For any two sets A_1 and A_2,

$A_1 = A_2$ if and only if

$$A_1 \subseteq A_2 \text{ and } A_2 \subseteq A_1$$

Intensional set definitions revisited

It is very common for an intensional set definition to take the following form:

$$\{x \mid x \in A \text{ and } P(x)\}$$

(This set will be that subset of A whose members satisfy property P.)

E.g.

$$\{n \mid n \in \mathbb{N} \text{ and } 1 < n \text{ and } n < 10\}$$

i.e. that subset of $\mathbb{N}$ that falls between 1 and 10.

For this common form of definition, a shorthand is popular:

$$\{x \in A \mid P(x)\}$$

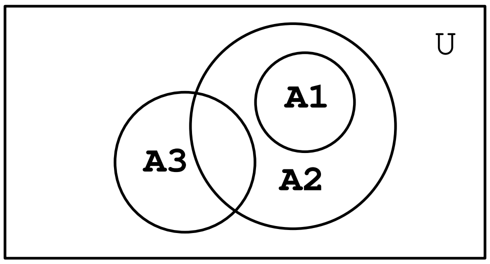
$$\{n \in \mathbb{N} \mid 1 < n \text{ and } n < 10\}$$

I do not use this shorthand in this course but have mentioned it here in case you encounter it in your textbooks.

Venn diagrams

We now turn to some operations that take sets and return other sets. Before that, we introduce a diagrammatic notation, **Venn diagrams**, that can aid understanding of the operations that we will look at.

In a Venn diagram, a set, A, is represented by a circle, the members of A being the points that lie within the circle. The universal set, U, is portrayed by a rectangle that encloses the circles.



$$\begin{array}{lll} A_1 \subseteq A_2 & A_1 \not\subseteq A_3 & A_1 \subseteq U \\ A_2 \not\subseteq A_1 & A_2 \not\subseteq A_3 & A_2 \subseteq U \\ A_3 \not\subseteq A_1 & A_3 \not\subseteq A_2 & A_3 \subseteq U \end{array}$$

Venn diagrams do not in general allow you to show the distinction between $\subseteq$ and $\subset$.

Set union

The **union** of two sets A_1 and A_2 is the set denoted by $A_1 \cup A_2$ and defined by the condition:

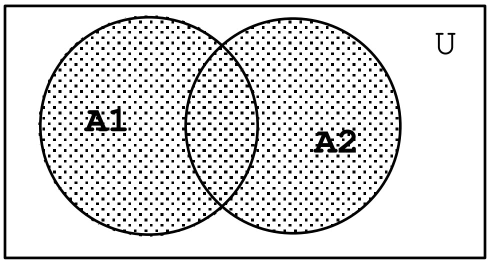
for all a, $a \in A_1 \cup A_2$ if and only if $a \in A_1$ or $a \in A_2$

More formally,

$$A_1 \cup A_2 \stackrel{\triangle}{=} \{a \mid a \in A_1 \text{ or } a \in A_2\}$$

For example

- $\{0, 1, 2\} \cup \{3, 4\} = \{0, 1, 2, 3, 4\}$
- $\{0, 1, 2\} \cup \{2, 3, 4\} = \{0, 1, 2, 3, 4\}$ (We could write $\{0, 1, 2, 2, 3, 4\}$ but, remembering that sets themselves contain no duplicates, we prefer non-redundant descriptions.)



The shaded area is $A_1 \cup A_2$.

Set intersection

The **intersection** of two sets A_1 and A_2 is the set denoted by $A_1 \cap A_2$ and defined by the condition:

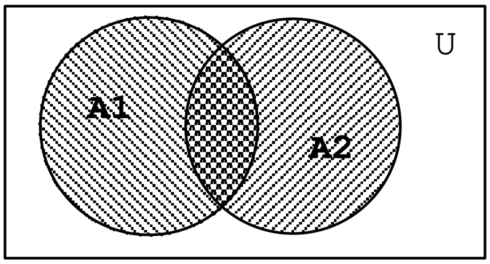
for all a, $a \in A_1 \cap A_2$ if and only if $a \in A_1$ and $a \in A_2$

More formally,

$$A_1 \cap A_2 \stackrel{\triangle}{=} \{a \mid a \in A_1 \text{ and } a \in A_2\}$$

For example

- $\{0, 1, 2\} \cap \{1, 2, 3\} = \{1, 2\}$
- $\{0, 1\} \cap \{2, 3, 4\} = \emptyset$

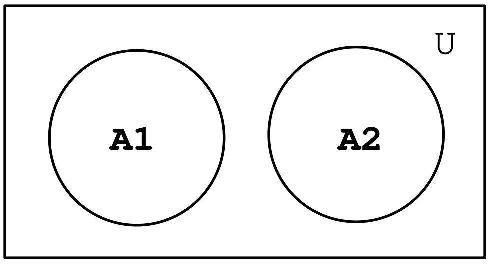


The dark shaded area in the middle is $A_1 \cap A_2$.

For any sets A_1 and A_2, A_1 and A_2 are said to be **disjoint** if they have no element in common, i.e.

$$A_1 \cap A_2 = \emptyset$$

When we know sets to be disjoint, we can draw the Venn diagram as follows:



Set difference

The **difference** of two sets A_1 and A_2 is the set denoted by $A_1 \backslash A_2$ and defined by the condition:

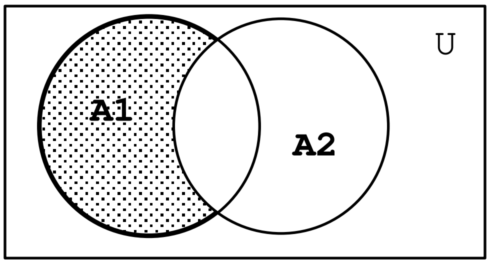
for all a, $a \in A_1 \setminus A_2$ if and only if $a \in A_1$ and $a \notin A_2$

More formally,

$$A_1 \setminus A_2 \stackrel{\triangle}{=} \{a \mid a \in A_1 \text{ and } a \notin A_2\}$$

For example

- $\{0, 1, 2, 3\} \setminus \{1, 2\} = \{0, 3\}$
- $\{0, 1, 2, 3\} \setminus \{2, 3, 4\} = \{0, 1\}$
- $\{0, 1, 2\} \setminus \{3, 4\} = \{0, 1, 2\}$



The shaded area is $A_1 \setminus A_2$.

The set difference of A_1 and A_2 is also called the **relative complement** of A_1 and A_2. Some people use a minus sign for this operator.

All the operators introduced so far have been binary prefix ones. Now we look at some unary ones. The first is postfix, and the second and third are prefix.

Set complement

The **complement** of a set A is the set denoted by A' and defined by the condition:

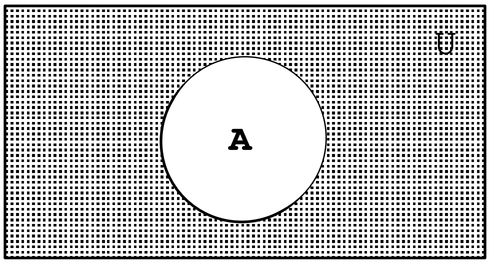
for all a in U, $a \in A'$ if and only if $A \in U$ but $a \notin A$

More formally,

$$A' \stackrel{\triangle}{=} \{a \mid a \in U \text{ and } a \notin A\}$$

For example, if $U \stackrel{\triangle}{=} \{0, 1, 2, 3, 4\}$,

- $\{0, 1, 3\}' = \{2, 4\}$



The shaded area is A'.

Obviously, this operation requires the definition of a universal set. It is clear that for all sets A, $A' = U \setminus A$.

The complement of A is also called the **absolute complement** of A. Other notation commonly used includes A^c and $\bar{A}$.

Class exercises

1. $U \triangleq \{n \mid n \in \mathbb{N} \text{ and } n > 4\}$

   Evaluate
   $(\{5, 6, 7\} \setminus \{7, 8\}) \subseteq \{x \mid x \in \mathbb{N} \text{ and } x > 10\}'$

2. $A \triangleq \{s \mid s \text{ was a student at York last year}\}$
   $B \triangleq \{s \mid s \text{ is a student who left York at the end of last year}\}$
   $C \triangleq \{s \mid s \text{ is a newly arrived York student}\}$

Write an expression that, if evaluated, would compute York's current students.

3. $A \triangleq \{s \mid s \text{ is a student who will take ADP}\}$
   $B \triangleq \{s \mid s \text{ is a student who will take BMG}\}$

Why might the timetabling officer evaluate

$$(A \cap B) = \emptyset$$

Sets as members of sets

Before we introduce the next operation, let's stress a point we've hidden up to now.

A set is a collection of objects. Sets are themselves objects. A set may have other sets as its members.

$$\{1, \{2, 3\}, 3\}$$
$$\{\{1\}, \emptyset, \{1, 3\}\}$$

You now need to think hard about what kinds of objects you are dealing with.

$$1 \neq \{1\} \neq \{\{1\}\} \neq \{\{\{1\}\}\}$$

$$\emptyset \neq \{\emptyset\} \neq \{\{\emptyset\}\} \neq \{\{\{\emptyset\}\}\}$$

In general, for any object a,

$$a \neq \{a\}$$

Be extra careful with the distinction between $\in$ and $\subseteq$ now!

$$\{1\} \in \{\{1\}, \emptyset, \{1, 3\}\}$$

$$1 \notin \{\{1\}, \emptyset, \{1, 3\}\}$$
$$\{\{1\}\} \subseteq \{\{1\}, \emptyset, \{1, 3\}\}$$
$$\{1\} \not\subseteq \{\{1\}, \emptyset, \{1, 3\}\}$$
$$\emptyset \in \{\{1\}, \emptyset, \{1, 3\}\}$$
$$\emptyset \notin \{\{1\}, \{1, 3\}\}$$
$$\{\emptyset\} \subseteq \{\{1\}, \emptyset, \{1, 3\}\}$$
$$\emptyset \subseteq \{\{1\}, \emptyset, \{1, 3\}\}$$

Partitions

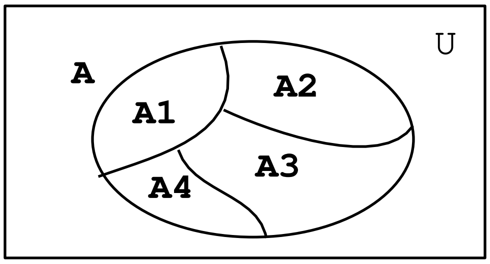
Let A be any non-empty set. A **partition** of A is a subdivision of all the members of A into disjoint, non-empty sets.

More formally, a partition of A is a set of non-empty sets $\{A_1, A_2, \ldots, A_n\}$ such that

(i) for any $a \in A$, $a \in A_i$ for some i

(ii) for all i and j, $i \neq j$, $A_i \cap A_j = \emptyset$

Diagrammatically,



Partitions: examples

Example: let $S \triangleq \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

One partition is:

$$\{\{1, 3, 5\}, \{2, 4, 6, 8\}, \{7, 9\}\}$$

Others are:

$$\{\{1, 5, 7\}, \{2, 4, 8\}, \{3, 6, 9\}\}$$

$$\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}, \{8\}, \{9\}\}$$
$$\{\{1, 2, 3, 4, 5, 6, 7, 8, 9\}\}$$

The following are not (the first is missing an element of S, the second has non-disjoint members and the third contains the empty set).

$$\{\{1, 5, 7\}, \{4, 8\}, \{3, 6, 9\}\}$$

$$\{\{1, 5, 7\}, \{2, 4, 6, 8\}, \{1, 3, 6, 9\}\}$$
$$\{\{1, 5, 7\}, \emptyset, \{2, 4, 6, 8\}, \{3, 6, 9\}\}$$

Powerset

The **powerset** of a set A is the set denoted by $\wp(A)$ and defined by the condition:

for all a, $a \in \wp(A)$ if and only if $a \subseteq A$

More formally,

$$\wp(A) \stackrel{\triangle}{=} \{a \mid a \subseteq A\}$$

— the set of all subsets of A.

(Some books use 2^A instead of $\wp(A)$ to denote the powerset of A. They do this because there is then a tie-in with the cardinality of the powerset —see later.)

For example,

- $\wp(\{0, 1\}) = \{\emptyset, \{0\}, \{1\}, \{0, 1\}\}$.
- $\wp(\{a, b, c\}) = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$

Note how $\emptyset$ is always a member of the powerset of any set. Why?

Set cardinality

We have looked at set operations that return true or false, and those that return sets. Now we look at an operation that takes in a set and returns a number.

The number of members of a set A is called the **cardinality** of A.

The cardinality of set A will be denoted *card(A)*.

Different books use different operators, e.g.

$$\#A, \quad |A|, \quad \text{num}(A)$$

card(A) has the virtue of being mnemonic.

Examples

$$\begin{aligned} \textit{card}(\{0,1,2\}) &= 3 \\ \textit{card}(\{3\}) &= 1 \\ \textit{card}(\emptyset) &= 0 \end{aligned}$$

Sets whose cardinality is 1 are often called **singleton sets**.

A set A can be **finite**, where *card(A)* $\in \mathbb{N}$, or **infinite**, where *card(A)* $\notin \mathbb{N}$. For example, the set $\{0, 1, 2\}$ is finite; the set $\mathbb{N}$ is infinite. We won't trouble ourselves with the cardinalities of infinite sets.

Computing cardinalities

- For any finite, disjoint sets A_1 and A_2,

$$\textit{card}(A_1 \cup A_2) = \textit{card}(A_1) + \textit{card}(A_2)$$

- For any finite sets A_1 and A_2 (not necessarily disjoint):

$$\begin{aligned}&\textit{card}(A_1 \cup A_2) = \\ &\qquad (\textit{card}(A_1) + \textit{card}(A_2)) - \textit{card}(A_1 \cap A_2)\end{aligned}$$

- For any finite set A,

$$\textit{card}(\wp(A)) = 2 \uparrow \textit{card}(A)$$

E.g. $\textit{card}(\wp(\{0, 1, 2\})) = 2 \uparrow \textit{card}(\{0, 1, 2\}) = 2 \uparrow 3 = 8$

Since $\wp(\{0, 1, 2\}) = \{\emptyset, \{0\}, \{1\}, \{2\}, \{0, 1\}, \{0, 2\}, \{1, 2\}, \{0, 1, 2\}\}$, we see the cardinality is indeed 8.

Computing cardinalities, cont'd

There are 60 students studying maths in a particular Computer Science Department. 50 of them also study one or more of the languages French, German and Italian. If U is the set of these 60 students, and $F, G,$ and I are the sets studying French, German and Italian respectively, then we are told:

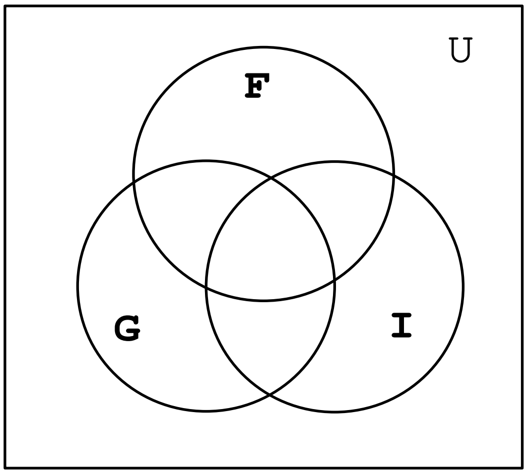
$$\begin{aligned} \textit{card}(F) &= 32 \\ \textit{card}(G) &= 22 \\ \textit{card}(I) &= 21 \\ \textit{card}(F \cap G) &= 10 \\ \textit{card}(F \cap I) &= 12 \\ \textit{card}(G \cap I) &= 7 \end{aligned}$$

Use the following law

For any finite sets A_1, A_2 and A_3,

$$\begin{aligned} &\textit{card}((A_1 \cup A_2) \cup A_3) = \\ &\quad \textit{card}(A_1) + \textit{card}(A_2) + \textit{card}(A_3) \\ &\quad -\textit{card}(A_1 \cap A_2) - \textit{card}(A_1 \cap A_3) - \textit{card}(A_2 \cap A_3) \\ &\quad +\textit{card}((A_1 \cap A_2) \cap A_3) \end{aligned}$$

to fill in the correct number of students in each of the eight regions of the Venn diagram.



Laws of the algebra of sets

For any sets SE_1, SE_2 and SE_3,...

Commutativity of $\cup$ and $\cap$

$$SE_1 \cup SE_2 = SE_2 \cup SE_1$$

$$SE_1 \cap SE_2 = SE_2 \cap SE_1$$

By way of demonstration, consider $S1 \triangleq \{1, 2, 3, 4\}$ and $S2 \triangleq \{3, 4, 5, 6\}$, then $S1 \cup S2 = S2 \cup S1 = \{1, 2, 3, 4, 5, 6\}$; similarly, $S1 \cap S2 = S2 \cap S1 = \{3, 4\}$. The order of the operands makes no difference.

By contrast, set difference is *not* in general commutative. For example, for the sets $S1$ and $S2$, $S1 \setminus S2 = \{1, 2\}$ but $S2 \setminus S1 = \{5, 6\}$ (a counterexample).

Associativity of $\cup$ and $\cap$

$$(SE_1 \cup SE_2) \cup SE_3 = SE_1 \cup (SE_2 \cup SE_3)$$

$$(SE_1 \cap SE_2) \cap SE_3 = SE_1 \cap (SE_2 \cap SE_3)$$

By way of demonstration, consider $S1 \triangleq \{1, 3\}$, $S2 \triangleq \{2, 3, 4\}$ and $S3 \triangleq \{5\}$, then

$$(S1 \cup S2) \cup S3 = \{1, 2, 3, 4\} \cup \{5\} = \{1, 2, 3, 4, 5\}$$
$$S1 \cup (S2 \cup S3) = \{1, 3\} \cup \{2, 3, 4, 5\} = \{1, 2, 3, 4, 5\}$$

The same result is reached by two different routes.

Similarly for intersection ($\cap$).

Given that these laws show that the order in which we union together/intersect more than two sets doesn't matter, it also follows that we can drop the parentheses if we wish: if we are asked to compute $A_1 \cup A_2 \cup A_3$ there is no risk of getting different results for different evaluation orders.

Laws of the algebra of sets, cont'd.

For any sets SE_1, SE_2 and SE_3,...

Distributivity of $\cup$ over $\cap$, and of $\cap$ over $\cup$

$$SE_1 \cup (SE_2 \cap SE_3) = (SE_1 \cup SE_2) \cap (SE_1 \cup SE_3)$$

$$SE_1 \cap (SE_2 \cup SE_3) = (SE_1 \cap SE_2) \cup (SE_1 \cap SE_3)$$

De Morgan's laws

$$(SE_1 \cup SE_2)' = SE_1' \cap SE_2'$$

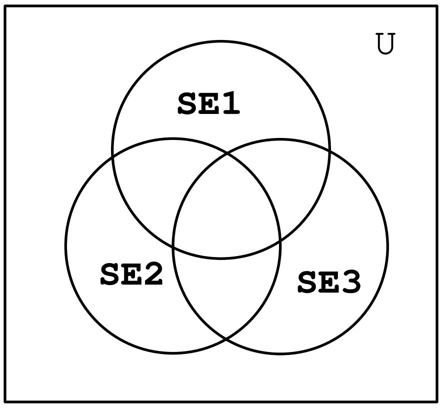
$$(SE_1 \cap SE_2)' = SE_1' \cup SE_2'$$

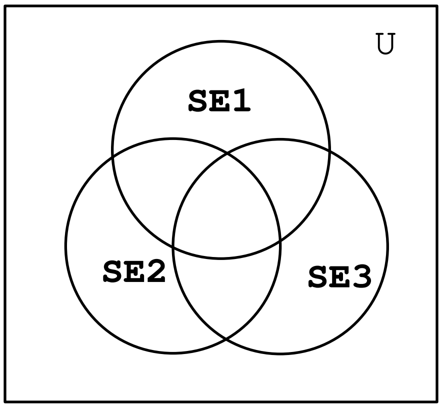
Demonstration of two of the laws

We will colour these diagrams in during the lecture.

Distributivity of $\cup$ over $\cap$

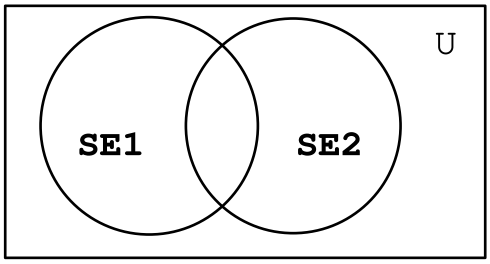
$$SE_1 \cup (SE_2 \cap SE_3) = (SE_1 \cup SE_2) \cap (SE_1 \cup SE_3)$$

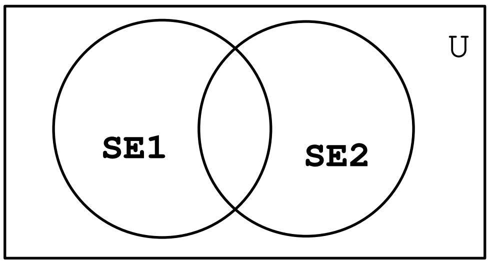




One of De Morgan's laws

$$(SE_1 \cup SE_2)' = SE_1' \cap SE_2'$$





The other two

$$SE_1 \cap (SE_2 \cup SE_3) = (SE_1 \cap SE_2) \cup (SE_1 \cap SE_3)$$

$$(SE_1 \cap SE_2)' = SE_1' \cup SE_2'$$

are on the Problem Sheet.

Laws of the algebra of sets, cont'd

For any set SE,…

Idempotence of $\cup$ and $\cap$

$$SE \cup SE = SE$$

$$SE \cap SE = SE$$

Identities

$$\emptyset \cup SE = SE$$

$$U \cup SE = U$$
$$\emptyset \cap SE = \emptyset$$
$$U \cap SE = SE$$

Laws of the algebra of sets, cont'd.

For any set SE,...

Involution

$$(SE')' = SE$$

Complement laws

$$SE \cup SE' = U$$

$$SE \cap SE' = \emptyset$$
$$U' = \emptyset$$
$$\emptyset' = U$$

Note that we have given no laws for set difference. See the Problem Sheet!

Using the laws in proofs

We use the laws much as we did for arithmetic.

Example: prove that for any sets A_1 and A_2,

$$(A_1 \cap A_2) \cup (A_1 \cap U) = A_1$$

$$\begin{aligned}
\text{LHS} \quad &= (A_1 \cap A_2) \cup (A_1 \cap U) \\
&= A_1 \cap (A_2 \cup U) \\
&= A_1 \cap (U \cup A_2) \\
&= A_1 \cap U \\
&= U \cap A_1 \\
&= A_1 \\
&= \text{RHS}
\end{aligned}$$

Again, we will annotate each line with its justification during the lecture.

RELATIONS AND FUNCTIONS

In this section of the course, we introduce a new kind of object: the tuple. Sets of tuples then form the basis for relations and a special case of relations, functions. In modelling, relations and functions are used to capture relationships between objects, properties of objects, changes in the state of objects, and so on.

Couples

A **couple** or **ordered pair** (sometimes abbreviated to just the word *pair*) or **2-tuple** consists of *two* objects *taken in a certain order*.

The notation we use is to enclose the expressions that denote the two objects within angular brackets:

$$\langle 1, 2 \rangle \qquad \langle 1, 1 \rangle$$

$$\langle \textit{DGB, MIP} \rangle \qquad \langle \textit{DGB, } \{\textit{MIP, DEC, NLP}\} \rangle$$

$$\langle \textit{DGB}, \langle \textit{MIP}, \langle \textit{DEC, NLP} \rangle \rangle \rangle \qquad \langle \textit{DGB}, \langle \langle \textit{MIP, DEC} \rangle, \textit{NLP} \rangle \rangle$$

The **first co-ordinate** and the **second co-ordinate**:

- can be any type of object (they can be different types or the same type)
- can be different objects or the same object
- and *order matters* i.e. for any object a_1 and any object a_2 where $a_1 \neq a_2$ then $\langle a_1, a_2 \rangle \neq \langle a_2, a_1 \rangle$.

Cartesian product

Here is another operation on sets: in the simple case that we present here, it is a binary operation, taking two sets and returning a set of couples. We generalise it shortly so that it is not limited to being a binary operation.

For any two sets A_1 and A_2, the **Cartesian product** of A_1 and A_2, denoted by $A_1 \times A_2$, is the set of all the couples $\langle a_1, a_2 \rangle$ such that $a_1 \in A_1$ and $a_2 \in A_2$.

More formally,

$$A_1 \times A_2 \stackrel{\triangle}{=} \{\langle a_1, a_2 \rangle \mid a_1 \in A_1 \text{ and } a_2 \in A_2\}$$

Examples: Let $S1 \stackrel{\triangle}{=} \{1, 2, 3\}$ and $S2 \stackrel{\triangle}{=} \{a, b\}$.

$$\begin{array}{ll} S1 \times S2 & = \{\langle 1, a\rangle, \langle 2, a\rangle, \langle 3, a\rangle, \langle 1, b\rangle, \langle 2, b\rangle, \langle 3, b\rangle\} \\ S2 \times S1 & = \{\langle a, 1\rangle, \langle a, 2\rangle, \langle a, 3\rangle, \langle b, 1\rangle, \langle b, 2\rangle, \langle b, 3\rangle\} \\ S2 \times S2 & = \{\langle a, a\rangle, \langle a, b\rangle, \langle b, a\rangle, \langle b, b\rangle\} \end{array}$$

Note that, in general, for sets A_1 and A_2,

$$A_1 \times A_2 \neq A_2 \times A_1$$

Note that for sets A_1 and A_2,

$$\mathit{card}(A_1 \times A_2) = \mathit{card}(A_1) * \mathit{card}(A_2)$$

We use Cartesian products when defining relations.

n-tuples

We now give an obvious extension of the idea of a couple.

An **n-tuple** consists of n objects, where n is any positive integer, taken in a certain order.

Examples

1-tuple	$\langle a_1 \rangle$	(a degenerate case)
2-tuple	$\langle a_1, a_2 \rangle$	(couples, ordered pairs)
3-tuple	$\langle a_1, a_2, a_3 \rangle$	(triples)
4-tuple	$\langle a_1, a_2, a_3, a_4 \rangle$	
$\vdots$	$\vdots$	
n-tuple	$\langle a_1, a_2, \ldots, a_n \rangle$	(tuple)

Note that, e.g., $\langle a_1, \langle a_2, a_3 \rangle \rangle \neq \langle \langle a_1, a_2 \rangle, a_3 \rangle \neq \langle a_1, a_2, a_3 \rangle$: the first two are couples (albeit couples in which one of the co-ordinates is itself a couple); the third is a triple.

Give three ways that n-tuples and sets differ.

Cartesian product: generalised

We extend our definition of Cartesian product so that it works for any finite number of sets, n (where n is any integer).

For any sets $A_1, A_2, \ldots, A_n$, the **Cartesian product**, denoted

$$A_1 \times A_2 \times \ldots \times A_n$$

is the set of all n-tuples $\langle a_1, a_2, \ldots, a_n \rangle$ where $a_1 \in A_1, a_2 \in A_2, \ldots, a_n \in A_n$.

Note that for sets $A_1, A_2, \ldots, A_n$

$$\begin{aligned} &\textit{card}(A_1 \times A_2 \times \ldots \times A_n) = \\ &\quad \textit{card}(A_1) * \textit{card}(A_2) * \ldots * \textit{card}(A_n) \end{aligned}$$

Example

$$\begin{aligned} S1 &\triangleq \{1, 2, 3\} \\ S2 &\triangleq \{a, b\} \\ S3 &\triangleq \{\alpha, \beta\} \end{aligned}$$

$$\begin{aligned} S1 \times S2 \times S3 \quad = \quad &\{\langle 1, a, \alpha \rangle, \langle 1, a, \beta \rangle, \\ &\langle 1, b, \alpha \rangle, \langle 1, b, \beta \rangle \\ &\langle 2, a, \alpha \rangle, \langle 2, a, \beta \rangle, \\ &\langle 2, b, \alpha \rangle, \langle 2, b, \beta \rangle \\ &\langle 3, a, \alpha \rangle, \langle 3, a, \beta \rangle, \\ &\langle 3, b, \alpha \rangle, \langle 3, b, \beta \rangle\} \end{aligned}$$

Compare this with $S1 \times (S2 \times S3)$ and with $(S1 \times S2) \times S3$.

Cartesian product, cont'd.

There is special notation for when you form the Cartesian product of a set with itself.

For any set A,

$$A \times A \times \ldots \times A \text{ where there are } n \text{ occurrences of } A$$

$$\text{is also written as } A^n$$

For example, $S \stackrel{\triangle}{=} \{1, 2, \}$

$$\begin{array}{lll} = S^2 & = S \times S & = \quad \{\langle 1,1\rangle, \langle 1,2\rangle, \langle 2,1\rangle, \langle 2,2\rangle\} \\ = S^3 & = S \times S \times S & = \quad \{\langle 1,1,1\rangle, \langle 1,1,2\rangle \\ & & \quad\ \ \langle 1,2,1\rangle, \langle 1,2,2\rangle \\ & & \quad\ \ \langle 2,2,1\rangle, \langle 2,1,2\rangle \\ & & \quad\ \ \langle 2,2,1\rangle, \langle 2,2,2\rangle\} \end{array}$$

What about S^1? $S^1 = \{\langle 1\rangle, \langle 2\rangle\}$. So $S^1 \neq S$.

Sometimes textbooks explicitly say that they'll take A^1 and A for any set A to be equal (since there are straightforward ways of converting one to the other).

What about S^0? $S^0 = \emptyset$

Binary relations

Relations give a mathematical way to capture 'relationships' that hold between objects. Note that the word 'relation' is mathematical terminology; the word 'relationship' is being used in a wholly informal way. Don't mix the two up.

For now, we restrict our attention to relationships between pairs of objects (binary relationships, hence **binary relations**). A *very* informal explanation of binary relations follows.

A *couple* seems like a good way to capture one individual relationship between two objects, e.g. *Ann BelongsTo Vanbrugh*.

$$\langle \textit{Ann, Vanbrugh} \rangle$$

Capturing the *'BelongsTo'* relationship as a whole will require a set of couples: e.g.

$$\{\langle \textit{Ann, Vanbrugh} \rangle, \langle \textit{Bill, Vanbrugh} \rangle, \langle \textit{Clare, Goodricke} \rangle, \ldots\}$$

Any couples not in the set are not related by the *'BelongsTo'* relationship, e.g. $\langle$*Ann, Goodricke*$\rangle$. So, if we take the set of students and the set of colleges and we take the Cartesian product of these two sets, then some subset of the Cartesian product will capture the *'BelongsTo'* relationship. Some other subset of this Cartesian product might capture some other relationship, e.g. the *'LivesIn'* relationship:

$$\{\langle \textit{Ann, Vanbrugh} \rangle, \langle \textit{Clare, Wentworth} \rangle, \ldots\}$$

or the *'EnjoysTheCollegeFoodOf'* relationship:

$$\emptyset$$

More formally, a **binary relation** R is an object with three features:

- a set called the **domain** of R and written *Dom*(R)
- a set called the **codomain** of R and written *Cod*(R)
- a subset of the Cartesian product *Dom*(R) $\times$ *Cod*(R) called the **graph** or **extension** of R and denoted simply by R

$$R \subseteq \textit{Dom}(R) \times \textit{Cod}(R)$$

The domain and codomain of a relation R can be any sets you choose. (They can be the same set if you wish.)

Binary relations, cont'd.

To name a new relation R, we must specify all three components (domain, codomain and graph/extension).

To specify domain D and codomain C, we write:

$$R : D \leftrightarrow C$$

To specify the graph/extension of R, we write

$$R \stackrel{\triangle}{=} \ldots \text{some specification of a set of couples} \ldots$$

On the right hand side of the above, where you define the set of couples, you may enumerate them, give a defining property or give an expression that uses set operations such as $\cup$, $\cap$, etc.

Binary relations, cont'd.

To show that objects a_1 and a_2 are elements of the relation R (i.e. are related according to relationship R) we write

$$\langle a_1, a_2 \rangle \in R$$

or, more compactly,

$$a_1 \; R \, a_2$$

or

$$R(a_1, a_2)$$

or, when it is clear what relation we are talking about,

$$a_1 \mapsto a_2$$

(these sometimes being called **maplets**).

We sometimes say that R **associates** a_1 with a_2 or R **maps** a_1 to a_2. To show that objects a_1 and a_2 are *not* elements of relation R (i.e. are not related according to relationship R) we write one of

$$\langle a_1, a_2 \rangle \notin R$$

$$a_1 \; \not R \, a_2$$

Examples of binary relations

$$S \triangleq \{\textit{Ann, Bill, Clare}\}$$
$$C \triangleq \{\textit{Goodricke, Vanbrugh, Wentworth}\}$$

We introduce the relation *BelongsTo* as follows:

$$\textit{BelongsTo} : S \leftrightarrow C$$

$$\textit{BelongsTo} \triangleq \{\langle \textit{Ann, Vanbrugh} \rangle, \langle \textit{Bill, Vanbrugh} \rangle, \langle \textit{Clare, Goodricke} \rangle\}$$

So, $\textit{Dom(BelongsTo)} = S$ and $\textit{Cod(BelongsTo)} = C$, and we have the following maplets:

$$\textit{Ann} \mapsto \textit{Vanbrugh}$$

$$\textit{Bill} \mapsto \textit{Vanbrugh}$$

$$\textit{Clare} \mapsto \textit{Goodricke}$$

Here are two further relations.

$$\textit{LivesIn} : S \leftrightarrow C$$

$$\textit{LivesIn} \triangleq \{\langle \textit{Ann, Vanbrugh} \rangle, \ldots \langle \textit{Clare, Goodricke} \rangle\}$$

$$\textit{EnjoysTheCollegeFoodOf} : S \leftrightarrow C$$

$$\textit{EnjoysTheCollegeFoodOf} \triangleq \emptyset$$

Further examples of binary relations

Here are examples that we described previously as being ‘operations’. We see now that they are relations. We review them only informally.

$\in$ and $\notin$ denote relations whose domains are all possible objects and whose codomains are all possible sets.

$\subseteq, \subset, \not\subseteq, \not\subset$ denote relations whose domains are all possible sets and whose codomains are all possible sets.

$<, \leq, >, \geq$ denote relations between numbers and other numbers.

Let’s introduce $\leq$ more formally (but, we will define it only for the natural numbers).

$$\leq : \mathbb{N} \leftrightarrow \mathbb{N}$$

$$\leq \ \triangleq \ \{\langle x, y\rangle \mid \text{ there exists some } z \text{ such that } z \in \mathbb{N} \text{ and } y - z = x\}$$

(The graph/extension here is infinite.)

Binary relations: pictorial representations

To strengthen intuitions, we will show some pictorial representations. (Obviously, these are only useful for relations whose domain and codomain are finite and, preferably, small.)

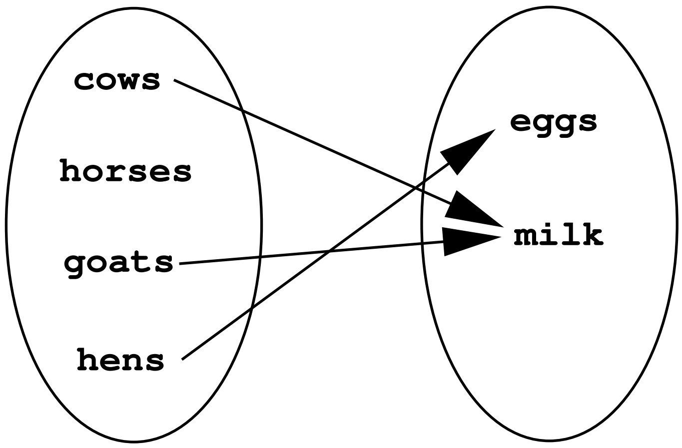
$$
\begin{aligned}
\textit{Livestock} &\stackrel{\triangle}{=} \{\textit{cows, horses, goats, hens}\} \\
\textit{Foodstuffs} &\stackrel{\triangle}{=} \{\textit{eggs, milk}\}
\end{aligned}
$$

$$\textit{Produces} : \textit{Livestock} \leftrightarrow \textit{Foodstuffs}$$

$$
\begin{aligned}
\textit{Produces} \stackrel{\triangle}{=} \quad & \{\langle \textit{hens, eggs} \rangle, \langle \textit{cows, milk} \rangle, \\
& \langle \textit{goats, milk} \rangle\}
\end{aligned}
$$

Arrow diagrams

For relation R from set A_1 to set A_2, write down the elements of A_1 and the elements of A_2 in two disjoint discs, and then for each couple $\langle a_1, a_2 \rangle \in R$, draw an arrow from a_1 to a_2.



Binary relations: pictorial representations, cont'd.

Matrices

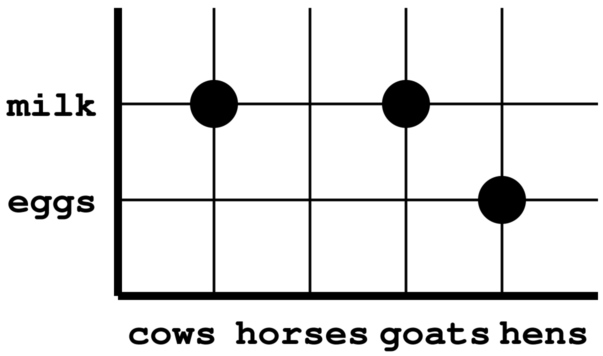
Form a rectangular array whose rows are labelled by elements of A_1 and whose columns are labelled by elements of A_2. Put a 1 or 0 in each cell, according to whether its row and column labels are related by the relation or not respectively.

(This representation is particularly interesting. It is suggestive of one way to store relations in computer data structures (arrays). And it gives us a connection to another area of maths (matrices). Some operations on matrices correspond to useful operations on relations.)

	eggs	*milk*
cows	0	1
horses	0	0
goats	0	1
hens	1	0

Coördinate diagrams

Draw an x-axis labelled by elements of A_1 and a y-axis labelled by elements of A_2. For each couple in the relation's graph/extension, plot a point on the diagram.

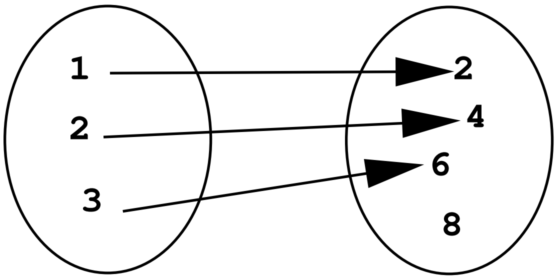


Binary relations: pictorial representations, cont'd.

$$S1 \triangleq \{1,2,3\}$$
$$S2 \triangleq \{2,4,6,8\}$$

$$\textit{Double} \triangleq \{\langle x,y\rangle \mid y = 2*x\}$$

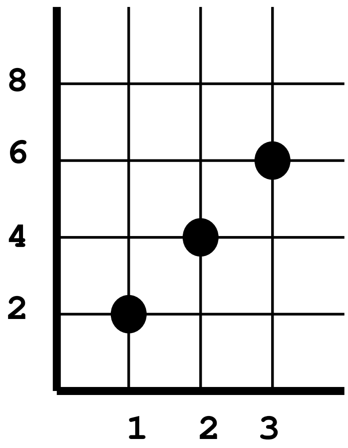
Arrow diagram



Matrix

	2	4	6	8
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0

Coördinate diagram



Binary relations: definition domains and ranges

The **definition domain** of a relation R is the set denoted by *Def*(R) and given by

$$\textit{Def}(R) \stackrel{\triangle}{=} \{a_1 \mid \text{there is some } a_2 \text{ such that } \langle a_1, a_2 \rangle \in R\}$$

i.e. the set that contains those elements of the domain that map to an object in the codomain.

The **range** of a relation R is the set denoted by *Ran*(R) and given by

$$\textit{Ran}(R) \stackrel{\triangle}{=} \{a_2 \mid \text{there is some } a_1 \text{ such that } \langle a_1, a_2 \rangle \in R\}$$

i.e. the set comprising those elements of the codomain that are mapped to from an object in the domain.

Clearly,

$$\textit{Def}(R) \subseteq \textit{Dom}(R)$$

$$\textit{Ran}(R) \subseteq \textit{Cod}(R)$$

Binary relations: definition domains and ranges example

$$\begin{array}{lcl} S1 & \triangleq & \{1,2,3,4,5\} \\ S2 & \triangleq & \{a,b,c\} \end{array}$$

$$\mathit{REL} : S1 \leftrightarrow S2$$

$$\mathit{REL} \triangleq \{\langle 1,b\rangle, \langle 1,c\rangle, \langle 3,b\rangle, \langle 5,c\rangle\}$$

Then

$$\begin{array}{rl} \mathit{Def(REL)} & = \{1,3,5\} \\ \mathit{Ran(REL)} & = \{b,c\} \end{array}$$

What are the definition domains and ranges of the *Produces* and *Double* relations?

Beware! The phrase ‘definition domain’ is not widespread in the literature, but is too useful to ignore. The word ‘range’ is widely used, but not always with the same definition as here: the words ‘codomain’, ‘range’, and ‘image’ have somewhat fluid definitions!

Binary relations: total and partial

A relation R is **total** if and only if

$$\textit{Def}(R) = \textit{Dom}(R)$$

or, if you prefer,

> for all $a_1 \in \textit{Dom}(R)$ there is at least one a_2 with
> $\langle a_1, a_2 \rangle \in R$

i.e. every element of the domain maps to something.

Otherwise, the relation is **partial**. (In this case, *Def(*R*)* $\subset$ *Dom(*R*)*.)

Example using *REL*

$$\textit{Def(REL)} = \{1, 3, 5\}, \quad \textit{Dom(REL)} = \{1, 2, 3, 4, 5\}$$

Since *Def(REL)* $\subset$ *Dom(REL)*, *REL* is partial.

Are *Produces* and *Double* total or partial?

Binary relations: surjective or not

A relation R is **surjective** (or **onto**) if and only if

$$\textit{Ran}(R) = \textit{Cod}(R)$$

or, if you prefer,

> for all $a_2 \in \textit{Cod}(R)$ there is at least one a_1 with $\langle a_1, a_2 \rangle \in R$

i.e. every element of the codomain is mapped to.

Otherwise, the relation is **not surjective** (or **into**). (In this case, *Ran(*R*)* $\subset$ *Cod(*R*)*.)

Example using *REL*

$$\textit{Ran(REL)} = \{b, c\}, \;\; \textit{Cod(REL)} = \{a, b, c\}$$

Since *Ran(REL)* $\subset$ *Cod(REL)*, *REL* is not surjective.

Are *Produces* and *Double* surjective or not?

Binary relations: functional or not

A relation R is **functional** if and only if

> for all $a_1 \in$ *Def*(R) there is exactly one a_2 with
> $\langle a_1, a_2 \rangle \in R$

i.e. every element that maps to something maps to no more than one thing. (This doesn't stop two elements mapping to the same thing; it stops one element mapping to two things: no diverging arrows in the arrow diagram.)

Otherwise, the relation is **not functional**. (There will be at least one element that maps to more than one thing.)

Example using *REL*.

$$1 \ \textit{REL} \ b$$

$$1 \ \textit{REL} \ c$$

There *is* an element that maps to more than one thing, so *REL* is not functional.

Are *Produces* and *Double* functional or not?

Binary relations: injective or not

A relation R is **injective** if and only if

> for all $a_2 \in$ *Ran*(R) there is exactly one a_1 with
> $\langle a_1, a_2 \rangle \in R$

i.e. every element that is mapped to is mapped to by no more than one thing. (This doesn't stop two elements being mapped to by the same thing; it stops one element being mapped to by two things: no converging arrows in the arrow diagram.)

Otherwise, the relation is **not injective**. (There will be at least one element that is mapped to by more than one thing.)

Example using *REL*

$$1 \; \textit{REL} \; b$$

$$2 \; \textit{REL} \; b$$

There *is* an element that is mapped to by more than one thing, so *REL* is not injective.

Are *Produces* and *Double* injective or not?

Binary relations: further terminology

It is common to bring the notions of 'functional' and 'injective' together and define terminology that deals with both of them in one go.

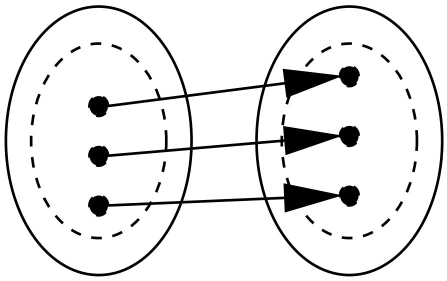
	Functional	Not functional
Injective	One-to-one	One-to-many
Not injective	Many-to-one	Many-to-many

When a relation is one-to-one (i.e. functional and injective) and also total and surjective, it is called a **bijection**.

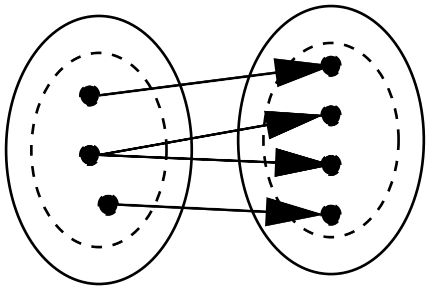
Question: if we know there is a bijection from a set A_1 to a set A_2 what else do we know about A_1 and A_2?

Binary relations: terminology summary

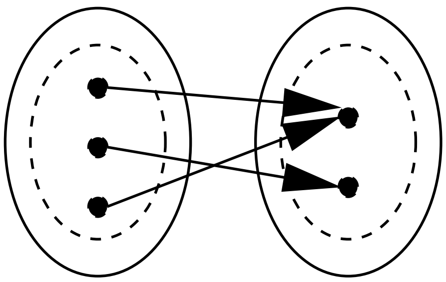
Here are diagrams demonstrating these ideas.



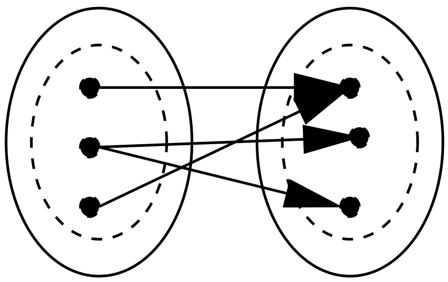
Functional & injective (One-to-one)



Not functional but injective (One-to-many)



Functional but not injective (Many-to-one)



Not functional & not injective (Many-to-many)

Whether there are elements outside of the dotted lines determines whether the relation is total or partial and surjective (onto) or not (into).

Which of these four describes *REL*, *Produces* and *Double*?

Operations on binary relations

We have seen four operations on binary relations already. They are operations that take in relations and return sets. Here they are again.

For any relation R,

Dom(R)	is the operation that returns R's domain
Cod(R)	is the operation that returns R's codomain
Def(R)	is the operation that returns R's definition domain
Ran(R)	is the operation that returns R's range

It might be useful to define operations that take in relations and return true or false, e.g., we could define operations that say whether a relation is total, not total (partial), surjective, not surjective, functional, not functional, injective, not injective, one-to-one, many-to-one, etc. And we could define $=$ and $\neq$ for relations. These crop up on the exercise sheet! We look here at operations that take in relations and return sets or return other relations.

Operations on binary relations: relational image

This operation takes in a relation and a set, and returns another set.

For a relation R from a set A_1 to a set A_2 and for a set A, **the image of A under R** is written *Im(R, A)*, and is defined as follows:

$$\mathit{Im}(R, A) \triangleq \{a_2 \mid \text{ there is some } a_1, \\ a_1 \in A \text{ and } a_1\ R\ a_2\}$$

i.e. it is the set of codomain values that are mapped to by those domain values that are in A.

E.g.

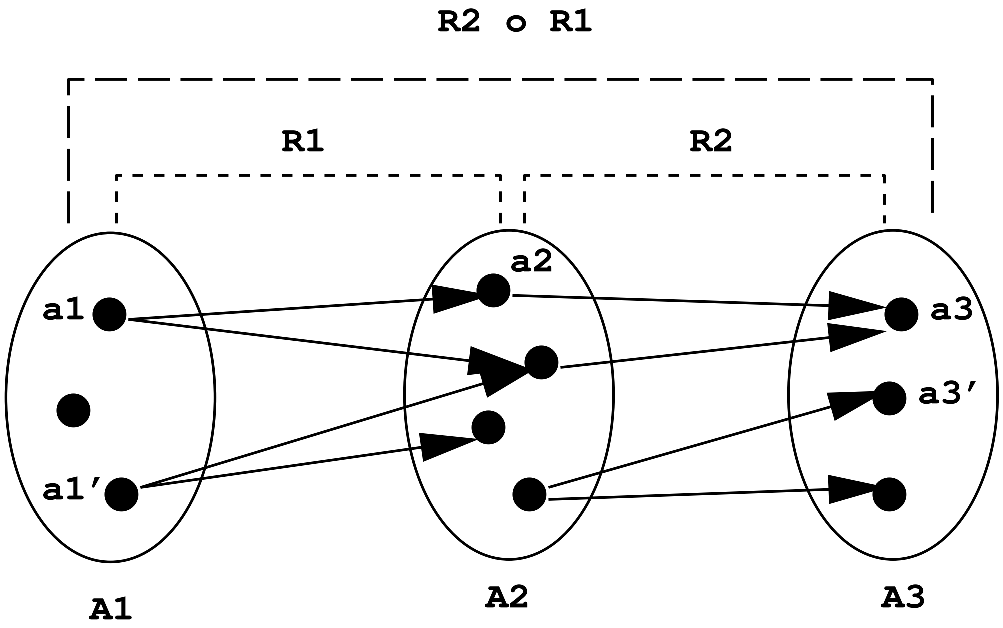
$$\begin{aligned} \mathit{Im(REL, \{1\})} &= \{b, c\} \\ \mathit{Im(REL, \{2, 4\})} &= \emptyset \\ \mathit{Im(Double, \{1, 2\})} &= \{2, 4\} \\ \mathit{Im(Produces, \{cows\})} &= \{\mathit{milk}\} \end{aligned}$$

Operations on binary relations: relation composition

Perhaps of more interest are operations that take in relations and return a new relation. Defining these operations requires that we specify the new relation's domain, codomain and graph/extension.

Consider two binary relations R_1 and R_2 such that

$$\textit{Cod}(R_1) = \textit{Dom}(R_2)$$



Operations on binary relation: relation composition, cont'd.

We can derive a third relation, the **composition** of R_1 and R_2, written $R_2 \circ R_1$ such that

$$
\begin{array}{rcll}
\textit{Dom}(R_2 \circ R_1) & \stackrel{\triangle}{=} & \textit{Dom}(R_1) & \\
\textit{Cod}(R_2 \circ R_1) & \stackrel{\triangle}{=} & \textit{Cod}(R_2) & \\
R_2 \circ R_1 & \stackrel{\triangle}{=} & \{\langle a_1, a_3 \rangle \mid & a_1 \in \textit{Dom}(R_1) \text{ and} \\
& & & a_3 \in \textit{Cod}(R_2) \text{ and} \\
& & & \text{there exists some } a_2, \\
& & & \text{ such that } a_1 \; R_1 \; a_2 \\
& & & \text{ and } a_2 \; R_2 \; a_3\}
\end{array}
$$

E.g. for the picture, $\langle a_1, a_3 \rangle \in R_2 \circ R_1$ because set A_2 contains an element a_2 such that $a_1 \; R_1 \; a_2$ and $a_2 \; R_2 \; a_3$. However, $\langle a'_1, a'_3 \rangle \notin R_2 \circ R_1$ because there is no intermediate point a'_2 such that $a'_1 \; R_1 \; a'_2$ and $a'_2 \; R_2 \; a'_3$.

Note that we write $R_2 \circ R_1$ for the composition of R_1 and R_2, which seems like putting the operands 'the wrong way around'. Perhaps the only way to remember this is to remember that it is counterintuitive!

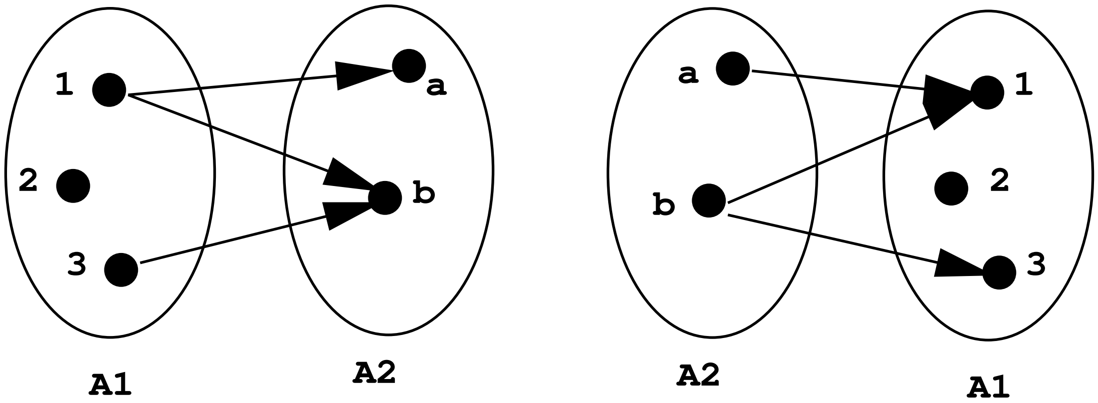
Operations on binary relation: relation composition example

$$
\begin{aligned}
\mathit{REL1} &: \{2, 4, 6\} \leftrightarrow \{1, 2, 3\} \\
\mathit{REL1} &\triangleq \{\langle x, y\rangle \mid x = 2 * y\} \cup \{\langle 4, 3\rangle\} \\
\mathit{REL2} &: \{1, 2, 3\} \leftrightarrow \{1, 4, 9\} \\
\mathit{REL2} &\triangleq \{\langle x, y\rangle \mid y = x \uparrow 2\} \setminus \{\langle 1, 1\rangle\} \\
\mathit{REL2} \circ \mathit{REL1} &: \{2, 4, 6\} \leftrightarrow \{1, 4, 9\} \\
\mathit{REL2} \circ \mathit{REL1} &\triangleq \{\langle 4, 4\rangle, \langle 4, 9\rangle, \langle 6, 9\rangle\}
\end{aligned}
$$

REL1 $\circ$ *REL2*, on the other hand, is undefined. Why?

Operations on binary relations: inverse

This operation takes in one relation and produces another relation. From relation $R : A_1 \leftrightarrow A_2$, we construct the **inverse** of R, denoted by R^{-1}, which is the relation from A_2 to A_1 whose graph/extension consists of those ordered pairs which, when reversed, belong to R.



More formally, for any relation R, the inverse of R is the relation R^{-1} such that

$$\begin{aligned} \mathit{Dom}(R^{-1}) &\triangleq \mathit{Cod}(R) \\ \mathit{Cod}(R^{-1}) &\triangleq \mathit{Dom}(R) \\ R^{-1} &\triangleq \{\langle a_2, a_1\rangle \mid \langle a_1, a_2\rangle \in R\} \end{aligned}$$

E.g.

$$\begin{aligned} \mathit{Double}^{-1} &: S2 \leftrightarrow S1 \\ \mathit{Double}^{-1} &= \{\langle 2, 1\rangle, \langle 4, 2\rangle, \langle 6, 3\rangle\} \end{aligned}$$

Class exercise: an airline

$$
\begin{aligned}
\textit{Passengers} &\triangleq \{\ldots, \textit{Max, Nell}, \ldots\} \\
\textit{Seats} &\triangleq \{\textit{1A, 1B, 1C, 2A, 2B}, \ldots\} \\
\textit{WindowSeats} &\triangleq \{\textit{1A, 1C, 2A, 2C}, \ldots\} \\
\textit{TravCompanies} &\triangleq \{\textit{SunFun, HotSpot}, \ldots\}
\end{aligned}
$$

$$
\begin{aligned}
\textit{Alloc} &: \textit{Passengers} \leftrightarrow \textit{Seats} \\
\textit{Alloc} &\triangleq \{\ldots, \langle \textit{Max, 46A} \rangle, \langle \textit{Nell, 52C} \rangle, \ldots\}
\end{aligned}
$$

$$
\begin{aligned}
\textit{BookedFlightFor} &: \textit{TravCompanies} \leftrightarrow \textit{Passengers} \\
\textit{BookedFlightFor} &\triangleq \{\ldots, \langle \textit{SunFun, Max} \rangle, \\
&\quad\ \langle \textit{HotSpot, Quentin} \rangle, \ldots\}
\end{aligned}
$$

Class exercise

1. What happens if *Alloc* is *not* injective?
2. What can we conclude if *Alloc* is *not* functional?
3. What does the following test for?

$$Ran(Alloc) \subset Cod(Alloc)$$

4. Write an expression that would test whether every passenger has a seat.

Class exercise

5. Max, Nell, Oswald and Penny are a family. What would the following compute for them?

$$\mathit{Im(Alloc,}\ \{\mathit{Max,\ Nell,\ Oswald,\ Penny}\}\mathit{)}$$

6. Write an expression that would tell Quentin where he should sit.

7. Quentin wants to know whether he has a window seat. What's wrong with the following?

$$\mathit{Im(Alloc,}\ \{\mathit{Quentin}\}\mathit{)} \in \mathit{WindowSeats}$$

Write an expression that would answer Quentin's question.

8. What would the following compute?

$$\mathit{Cod(BookedFlightFor)} \setminus \mathit{Ran(BookedFlightFor)}$$

Class exercise

9. What would the following compute?

$$\mathit{Alloc} \circ \mathit{BookedFlightFor}$$

10. I'm the travel rep. for HotSpot. Write an expression that would tell me where my passengers are sitting.

11. What would the following compute?

$$\mathit{Alloc}^{-1}$$

and

$$\mathit{BookedFlightFor}^{-1}$$

12. The passengers in seats 46A, 46B and 45B misbehave during dinner. The steward(ess) wants to complain to their travel company/companies. Write an expression that would determine which companies are involved.

We have been discussing relations in general (well, binary ones anyway). We are now going to look at some special kinds of relations: firstly, relations on a set, and secondly, functions. But, we stress right now, these special kinds of relations are still relations, so everything we have said so far still applies (all the terminology, all the operations).

Relations on a set

We continue to restrict ourselves to binary relations. A binary relation from a set A to the same set is called a binary **relation on** A, i.e. relation R is a relation on A if and only if

$$\textit{Dom}(R) = \textit{Cod}(R)$$

$$R \subseteq A^2$$

Example

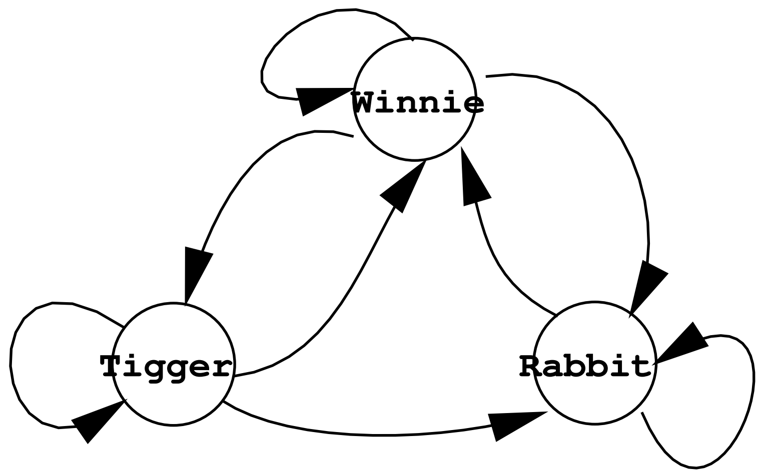
$$\begin{array}{rcl} \textit{Creatures} & \triangleq & \{\textit{Winnie, Tigger, Rabbit}\} \\ \textit{Likes} & : & \textit{Creatures} \leftrightarrow \textit{Creatures} \\ \textit{Likes} & \triangleq & (\textit{Creatures} \times \textit{Creatures}) \setminus \\ & & \quad \{\langle \textit{Rabbit, Tigger} \rangle\} \end{array}$$

Other examples: *ParentOf* is a relation on *People* (say), $\leq$ is a relation on $\mathbb{N}$ (or some other set of numbers), and $\subseteq$ is a relation on the set of all sets.

Relations on a set: pictorial representation

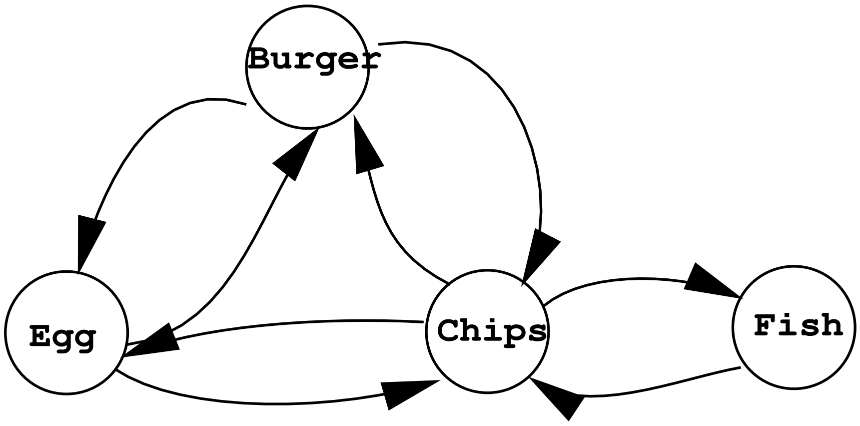
A relation on a set, when the set is finite and small, can be depicted in any of the three ways we have seen before (arrow diagrams, matrices, coördinate diagrams). But, there is now a fourth way that can only be used for these kinds of relations: **directed graphs**, also called **digraphs**. Each element of the set is written down (once) and arrows link related elements.

Directed graph



Relations on a set: another example

$$\begin{array}{rcl} \textit{Grub} & : & \{\textit{Burger, Chips, Eggs, Fish}\} \\ \textit{GoTogether} & : & \textit{Grub} \leftrightarrow \textit{Grub} \\ \textit{GoTogether} & \triangleq & \{\langle \textit{Burger, Chips}\rangle, \ldots\} \end{array}$$



Now, just as we spent time saying whether binary relations were total, surjective, functional, injective, etc, we will introduce terminology that *further* classifies relations on sets.

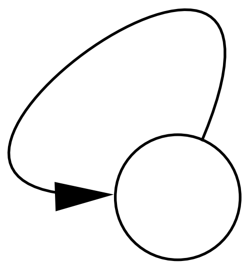
Relations on sets: reflexive

A binary relation R on a set A is **reflexive** if and only if for every element $a \in A$,

$$a \; R \; a$$

i.e. everything is related to itself.

You can tell whether a relation is reflexive from the relation's directed graph: *every* node should have a link to itself.



A binary relation R on a set A is **non-reflexive** if there's at least one element that is not related to itself.

A binary relation R on a set A is **antireflexive** if no element is related to itself.

Books sometimes use the term 'irreflexive' and they vary over whether this term means the same as what I have called non-reflexive or what I have called antireflexive.

Were *Likes* and *GoTogether* reflexive, non-reflexive or antireflexive?

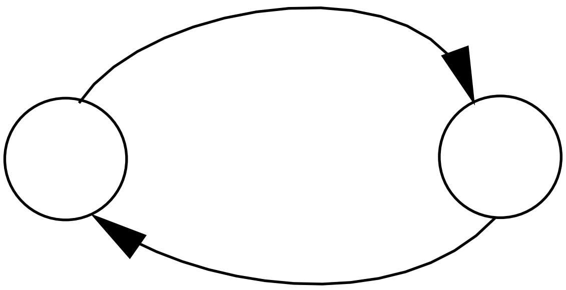
Relations on sets: symmetric

A binary relation R on a set A is **symmetric** if and only if for every pair of elements a_1 and a_2,

$$\text{if } a_1 \; R \; a_2 \text{ then } a_2 \; R \; a_1$$

i.e. if a_1 is mapped to a_2 then a_2 is also mapped to a_1.

You can tell whether a relation is symmetric by checking the relation's directed graph: if there's a link from one node to another, then there should be a link in the other direction too.



A binary relation R on a set A is **non-symmetric** if there's at least one maplet $a_1 \mapsto a_2$ for which there is no maplet $a_2 \mapsto a_1$.

A binary relation R on a set A is **antisymmetric** if none of the maplets $a_1 \mapsto a_2$ where $a_1 \neq a_2$ have corresponding maplets in the other direction ($a_2 \mapsto a_1$).

Books sometimes use the term 'asymmetric' and they vary over whether this term means the same as what I have called non-symmetric or what I have called antisymmetric.

Were *Likes* and *GoTogether* symmetric, non-symmetric or antisymmetric?

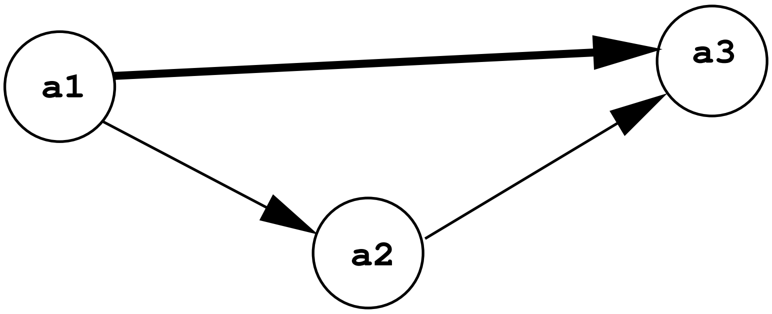
Relations on sets: transitive

A binary relation R on a set A is **transitive** if and only if for every triple of elements a_1, a_2 and a_3,

$$\text{if } a_1 \; R \; a_2 \text{ and } a_2 \; R \; a_3 \text{ then } a_1 \; R \; a_3$$

i.e. if a_1 is mapped to a_2 and a_2 is also mapped to a_3 then a_1 is mapped to a_3.

You can tell whether a relation is transitive by checking its directed graph: whenever there is an arrow from some a_1 to some a_2 and one from a_2 to some a_3 there should be a third arrow from a_1 to a_3 that 'completes the triangle'.



People don't usually bother to define 'non-transitive' or 'antitransitive', so we won't either.

Were *Likes* and *GoTogether* transitive or not?

Matters become more interesting when a relation satisfies several of these properties.

Relations on sets: equivalence relations

A relation R on a set A is called an **equivalence relation** if it is *reflexive, symmetric* and *transitive*.

The idea behind an equivalence relation is that it relates together objects that are in some way ‘alike’.

Examples

Consider the relation *IsTheSameSpeciesAs* on the set of animals. Clearly, every animal a is the same species as itself (reflexive); if animal a_1 is the same species as animal a_2 then animal a_2 is the same species as animal a_3 (symmetric); and if animal a_1 is the same species as animal a_2 and animal a_2 is the same species as animal a_3, then animal a_1 is the same species as animal a_3 (transitive). Hence, *IsTheSameSpeciesAs* is an equivalence relation.

Other examples: the relation *IsTheSameHeightAs* on the set of people (say); the relation *IsTheSameAgeAs* on the set of students (say); the relation $=$ on $\mathbb{N}$ (say).

Relations on sets: equivalence relations, cont'd.

An equivalence relation R on set A forms a *partition* of set A. Each set within the partition is called an **equivalence class**. Elements related to each other by R will belong to the same equivalence class.

The directed graph of an equivalence relation has a number of 'chunks', one per partition.

Example

$$\mathit{People} \triangleq \{\mathit{Jack, Joan, Jill, Jim, Joe, Judith}\}$$

Consider the relation on *People* called *SAA* (same age as) given that: *Jack* and *Joan* are 19, *Jim, Joe* and *Judith* are 20, and *Jill* is 21. The graph/extension is as follows.

$$\begin{aligned}
&\mathit{SAA} \triangleq \\
&\quad \{\langle \mathit{Jack, Jack}\rangle, \langle \mathit{Joan, Joan}\rangle, \\
&\quad\ \ \langle \mathit{Jill, Jill}\rangle, \langle \mathit{Jim, Jim}\rangle, \\
&\quad\ \ \langle \mathit{Joe, Joe}\rangle, \langle \mathit{Judith, Judith}\rangle, \\
&\quad\ \ \langle \mathit{Jack, Joan}\rangle, \langle \mathit{Joan, Jack}\rangle, \\
&\quad\ \ \langle \mathit{Jim, Joe}\rangle, \langle \mathit{Joe, Jim}\rangle, \\
&\quad\ \ \langle \mathit{Joe, Judith}\rangle, \langle \mathit{Judith, Joe}\rangle, \\
&\quad\ \ \langle \mathit{Jim, Judith}\rangle, \langle \mathit{Judith, Jim}\rangle\}
\end{aligned}$$

Confirm that this relation is reflexive, symmetric and transitive.

The partition is

$$\{\{\mathit{Jack, Joan}\}, \{\mathit{Jim, Joe, Judith}\}, \{\mathrm{Jill}\}\}$$

There are three equivalence classes here.

Relations on sets: equivalence relations, cont'd.

We can work matters the other way. Give a partition of a set A, we can determine an equivalence relation R that would partition A in the way given. Specifically, we will have $a_1 \; R \; a_2$ if and only if a_1 and a_2 are members of the same set within the partition.

Example: from this partition of the set *People*,

$$\{\{\textit{Jack}\}, \{\textit{Jim, Jill}\}, \{\textit{Joan, Joe}\}, \{\textit{Judith}\}\}$$

we can derive the equivalence relation on *People* (which we will call *SSL* (we could call it anything we want)) whose graph/extension is

$$\begin{aligned} &\textit{SSL} \triangleq \\ &\quad \{\langle \textit{Jack, Jack}\rangle, \langle \textit{Joan, Joan}\rangle, \langle \textit{Jill, Jill}\rangle, \\ &\quad\; \langle \textit{Jim, Jim}\rangle, \langle \textit{Joe, Joe}\rangle, \langle \textit{Judith, Judith}\rangle, \\ &\quad\; \langle \textit{Jim, Jill}\rangle, \langle \textit{Jill, Jim}\rangle, \langle \textit{Joan, Joe}\rangle, \langle \textit{Joe, Joan}\rangle\} \end{aligned}$$

Confirm that this is reflexive, symmetric and transitive.

Textbooks add extra notation to this topic, which we won't need. Specifically, they often use symbols such as $\sim$ rather than R or *SSL* as the names of equivalence relations; then they introduce the notation $[a]$ to denote the equivalence class that element a belongs to; and they refer to the partition of set A using equivalence relation R as the *quotient* of A by R, and use the notation A/R to denote this.

Relations on sets: orderings

Some relations on a set are **orderings** of the elements of the set.

Examples are: $<$ on $\mathbb{N}$ (say), $\leq$ on $\mathbb{N}$ (say), *IsShorterThan* on the set of people (say), *IsYoungerThan* on the set of buildings (say), *ScoredHigherThan* on a set of maths students (say).

Crucial to ordering is transitivity (if a_1 is less than a_2 and a_2 is less than a_3 then a_1 is less than a_3).

It is possible to define perfectly reasonable orderings called 'preorderings' that have transitivity but not antisymmetry. However, we'll ignore those. We will require antisymmetry (in no case is a_1 less than a_2 and a_2 also less than a_1 where $a_1 \neq a_2$.)

So, we require **transitivity** and **antisymmetry**.

Relations on sets: orderings example

$$People \triangleq \{Jack, Joan, Jill, Jim, Joe, Judith\}$$

Consider the relation on *People* called *YT* (younger than) given that: *Jack* and *Joan* are 19, *Jim, Joe* and *Judith* are 20, and *Jill* is 21. The graph/extension is as follows.

$$\begin{aligned} YT \triangleq \\ & \{\langle Jack, Jim\rangle, \langle Jack, Joe\rangle, \\ & \ \langle Jack, Judith\rangle, \langle Jack, Jill\rangle, \\ & \ \langle Joan, Jim\rangle, \langle Joan, Joe\rangle, \\ & \ \langle Joan, Judith\rangle, \langle Joan, Jill\rangle, \\ & \ \langle Jim, Jill\rangle, \langle Joe, Jill\rangle, \\ & \ \langle Judith, Jill\rangle\} \end{aligned}$$

Confirm that this relation is transitive and antisymmetric.

Relations on sets: orderings example

$$\mathit{SomeNumbers} \triangleq \{1, 2, 3, 4\}$$

Consider the relations *LT* (less than) and *LTOET* (less than or equal to) on *SomeNumbers*

The graphs/extensions are as follows.

$$\begin{aligned}\mathit{LT} \triangleq \\ &\{\langle 1, 2\rangle, \langle 1, 3\rangle, \langle 1, 4\rangle, \\ &\ \langle 2, 3\rangle, \langle 2, 4\rangle, \langle 3, 4\rangle\}\end{aligned}$$

$$\begin{aligned}\mathit{LTOET} \triangleq \\ &\{\langle 1, 2\rangle, \langle 1, 3\rangle, \langle 1, 4\rangle, \\ &\ \langle 2, 3\rangle, \langle 2, 4\rangle, \langle 3, 4\rangle, \\ &\ \langle 1, 1\rangle, \langle 2, 2\rangle, \langle 3, 3\rangle, \\ &\ \langle 4, 4\rangle\}\end{aligned}$$

Confirm that these relations are transitive and antisymmetric.

Relations on sets: strict/non-strict orderings

- **Strict** or **non-strict**
  - If an ordering is *antireflexive* (no object is related to itself), as with $<$ on $\mathbb{N}$, then the ordering is *strict*.
  - If an ordering is *reflexive* (every element is related to itself), as with $\leq$ on $\mathbb{N}$, then the ordering is *non-strict*.

An advanced point is that maths books define non-strict orders as reflexive, antisymmetric, transitive relations. But they define strict partial orders as antireflexive, transitive relations. You do not need to explicitly say that a strict partial order is antisymmetric. Indeed, to do so, as we have, is redundant. The antisymmetry follows from the antireflexivity and the transitivity. Don't worry about this point if you find it confusing.

Which of *YT*, *LT* and *LTOET* were strict and which were non-strict?

Relations on sets: total and partial orderings

- **Total** versus **partial** N.B. This use of these words is different from their use elsewhere.
  - If for all elements a_1 and a_2 where $a_1 \neq a_2$, a_1 is related to a_2 or a_2 is related to a_1, then the ordering is *total*.
  - If there is at least one pair of elements a_1 and a_2 where $a_1 \neq a_2$ such that a_1 is not related to a_2 and a_2 is not related to a_1 (i.e. they are incomparable), then the ordering is *partial*.

This gives us four kinds of ordering:

$$\left\{ \begin{array}{c} \text{strict} \\ \text{non-strict} \end{array} \right\} \left\{ \begin{array}{c} \text{total} \\ \text{partial} \end{array} \right\} \text{ordering.}$$

Which of *YT*, *LT* and *LTOET* were total and which were partial?

Operations on relations on sets: closures

We can obviously define operations that take in relations on sets and return true or false according to whether the relation is reflexive, non-reflexive, anti-reflexive, symmetric, non-symmetric, antisymmetric, transitive or not, an equivalence relation or not, a strict partial order, a strict total order, a non-strict partial order or a non-strict total order or not an order.

We can also define operations that take a partition and return an equivalence relation, or take an equivalence relation and return a partition.

All very useful, but we restrict ourselves to operations that take in a relation on a set an return another relation on a set.

The **reflexive closure** operation takes in a relation R on a set A and returns another relation denoted *RC(R)* whose graph/extension is the same as the graph/extension of R except that it also contains the minimum additional tuples needed to make the new relation reflexive.

The **transitive closure** operation takes in a relation R on a set A and returns another relation denoted *TC(R)* whose graph/extension is the same as the graph/extension of R except that it also contains the minimum additional tuples needed to make the new relation transitive.

The **reflexive, transitive closure** operation takes in a relation R on a set A and returns another relation denoted *RTC(R)* whose graph/extension is the same as the graph/extension of R except that it also contains the minimum additional tuples needed to make the new relation reflexive and transitive.

There are algorithms that tell you how to systematically form these closures for a given relation. We will not look at them in this course. Instead, we will just use our common-sense when forming closures.

Operations on relations on sets: closure examples

$$S \triangleq \{0, 1, 2\}$$

$$\mathit{REL} : S \leftrightarrow S$$

$$\mathit{REL} \triangleq \{\langle 0, 0\rangle, \langle 0, 1\rangle, \langle 1, 2\rangle, \langle 2, 0\rangle\}$$

Then,

$$\begin{aligned}
\mathit{RC(REL)} &\triangleq \{\langle 0, 0\rangle, \langle 0, 1\rangle, \langle 1, 2\rangle, \langle 2, 0\rangle, \\
&\quad \langle 1, 1\rangle, \langle 2, 2\rangle\} \\
\mathit{TC(REL)} &\triangleq \{\langle 0, 0\rangle, \langle 0, 1\rangle, \langle 1, 2\rangle, \langle 2, 0\rangle, \\
&\quad \langle 0, 2\rangle, \langle 1, 0\rangle, \langle 2, 1\rangle\} \\
\mathit{RTC(REL)} &\triangleq \{\langle 0, 0\rangle, \langle 0, 1\rangle, \langle 1, 2\rangle, \langle 2, 0\rangle, \\
&\quad \langle 1, 1\rangle, \langle 2, 2\rangle, \langle 0, 2\rangle, \langle 1, 0\rangle, \langle 2, 1\rangle\}
\end{aligned}$$

Operations on relations on sets: closure example

$$A \quad \triangleq \quad \{\textit{Adam, Dan, Christine, Phil, David,} \\ \textit{Elizabeth, Shula, Kenton}\}$$

Consider the *Parent* relation on *A*.

The graph/extension is as follows.

$$\begin{aligned} &\textit{Parent} \triangleq \\ &\quad \{\langle \textit{Adam, Dan} \rangle, \langle \textit{Dan, Christine} \rangle, \\ &\quad \langle \textit{Dan, Phil} \rangle, \langle \textit{Phil, David} \rangle, \\ &\quad \langle \textit{Phil, Elizabeth} \rangle, \langle \textit{Phil, Shula} \rangle, \\ &\quad \langle \textit{Phil, Kenton} \rangle\} \end{aligned}$$

What does *TC(Parent)* compute?

That's enough on relations on sets. Now we turn to another special kind of relation.

Functional relations

We defined what it means for a relation to be functional earlier. To recap: a relation is functional if no domain element maps to more than one codomain element, i.e. if the relation is many-to-one or one-to-one.

Functional relations are simply called **functions** for short.

We restrict ourselves to binary functional relations for now. (As we shall see later, a binary functional relation is a unary function!)

To specify a function F you give a domain, codomain and graph/extension as before, but the graph/extension must satisfy the following condition:

for any a_1, there is at most one a_2 such that $\langle a_1, a_2 \rangle \in F$

To introduce a function F with domain D and codomain C, instead of writing $F : D \leftrightarrow C$, it is common to introduce it using:

$$F : D \to C$$

which signals to the reader that this relation F is functional.

The pictorial representations may all be used (arrow diagrams, matrices, coördinate diagrams and, if the domain and codomain are the same set, directed graphs).

All the usual terminology still applies: definition domain, range, total/partial, surjective (onto)/not surjective (into), injective/not injective, etc.

Be warned that maths books sometimes use the word ‘function’ to mean a *total* function, and computing books sometimes use the word ‘function’ to mean a *partial* function, whereas we continue to use it neutrally between the two.

Functions

To specify elements of the graph/extension we can write the following as before:

$$\langle a_1, a_2 \rangle \in F$$
$$a_1 F a_2$$
$$F(a_1, a_2)$$
$$a_1 \mapsto a_2$$

but we can also write

$$F(a_1) = a_2$$

For any object a in the domain of function F, the object assigned to a, if there is one, is denoted $F(a)$. We say that F is **applied to** a, a is the **argument** of F, $F(a)$ is **the result of applying** F to a, and $F(a)$ is the **value of** F **at** a.

To specify the whole graph/extension of a function, we can use the methods we have used for relations in general (enumeration, a defining property or an expression that uses set operations) or we can use the method we used to define 'operations' in the section of the course that dealt with arithmetic expressions, i.e. using a parameter.

Example: *treble* is a function on $\mathbb{N}$. It is functional because each element of the domain maps to no more than one element of the codomain.

$$\textit{treble} : \mathbb{N} \to \mathbb{N}$$

To specify the graph/extension, we could write

$$\textit{treble} \triangleq \{\langle x, y \rangle \mid y = 3 * x\}$$

or,

$$\textit{treble}(x) \triangleq 3 * x$$

(And these say the same thing).

Function: non-numerical examples

The relation *Mother* from people to their mothers is functional; so is *Father*.

Parents and *Children* (from people to their parents/children) are not functions.

Height, *Age*, *Gender* and *Birthday* (from people to their heights/ ages/ genders/ birthdays) are functions (if the domain does not include the Queen and Paddington Bear).

For books, *Title*, *NoOfPages* and *Price* (from books to their titles, lengths and prices) are (probably) functional; *Authors* is not.

This section of the course has, therefore, shown us that many of the things that we called ‘operations’ were really relations and that many of these were functions. (The word ‘operation’ was merely an informal word.) The functions included *treble*, *sixtimes*, $\cup$, $\cap$, $\wp$, *card* and $\circ$.

Notice how the statement $F : D \to C$ gives the **type** of the function: D is the set from which the argument must be drawn, and C is the set from which the result will be drawn.

Representing (non-functional) relations as functions

$$\mathit{Royals} \triangleq \{\mathit{Liz, Phil, Chas, Andy, Willie}\}$$

Consider the following (non-functional) relation:

$$\begin{array}{lcl} \mathit{Child} & : & \mathit{Royals} \leftrightarrow \mathit{Royals} \\ \\ \mathit{Child} & \triangleq & \{\langle \mathit{Chas, Liz}\rangle, \langle \mathit{Chas, Phil}\rangle, \\ & & \langle \mathit{Andy, Liz}\rangle, \langle \mathit{Andy, Phil}\rangle, \\ & & \langle \mathit{Willie, Chas}\rangle\} \end{array}$$

Now consider the following function:

Alone

$$\mathit{ChildFunc} : \mathit{Royals} \rightarrow \wp(\mathit{Royals})$$

$$\begin{array}{rcl} \mathit{ChildFunc(Chas)} & = & \{\mathit{Liz, Phil}\} \\ \mathit{ChildFunc(Andy)} & = & \{\mathit{Liz, Phil}\} \\ \mathit{ChildFunc(Willie)} & = & \{\mathit{Chas}\} \end{array}$$

(We could make this function total by mapping *Liz* and *Phil* to $\emptyset$.)

This demonstrates a general technique for converting a non-functional relation to a function. We map objects to sets of objects (the function's codomain is the powerset of the relation's codomain). It is functional because no object maps to more than object in the codomain.

Representing sets as functions

Consider these sets:

$$U \triangleq \{\textit{Liz, Phil, Chas, Andy, Willie}\}$$

$$\textit{Alone} \triangleq \{\textit{Chas, Andy}\}$$

Now consider the following function, where the set *Bool* $\triangleq \{T, F\}$ (T being true and F being false)

$$\textit{AloneFunc} : U \to \textit{Bool}$$

$$\begin{aligned}
\textit{AloneFunc(Liz)} &= F \\
\textit{AloneFunc(Phil)} &= F \\
\textit{AloneFunc(Chas)} &= T \\
\textit{AloneFunc(Andy)} &= T \\
\textit{AloneFunc(Willie)} &= F
\end{aligned}$$

For any set A drawn from some universal set U, the function from U to *Bool* that maps an element $a \in U$ to T (true) if $a \in A$ and to F (false) if $a \notin A$ is called the **characteristic function of set** A.

This gives us a general way to get a function from a set: form the characteristic function of the set. It also gives another general way to obtain a function from a relation: form the characteristic function of the graph/extension of the relation.

Representing tuples as functions

Consider this tuple:

$$\langle \textit{Phil, Chas, Willie} \rangle$$

Now consider this function:

$$\textit{TupleFunc} : \mathbb{P} \rightarrow \{\textit{Phil, Chas, Willie}\}$$

$$\begin{array}{lcl} \textit{TupleFunc(1)} & = & \textit{Phil} \\ \textit{TupleFunc(2)} & = & \textit{Chas} \\ \textit{TupleFunc(3)} & = & \textit{Willie} \end{array}$$

Any n-tuple has a corresponding function from the set of integers from 1 to n to the elements of the tuple.

Up to now, all the relations we have looked at have been binary ones. To close this section of the course, we briefly look at n-ary relations.

n-ary relations

We can capture relationships that hold between n objects (for n being any positive integer) using n-tuples. For example, customer *Alf* ordering financial service *Pension3* from salesperson *Betty*:

$$\langle \textit{Alf, Pension3, Betty} \rangle$$

Again, to capture the relationship as a whole requires a set of such tuples.

An n**-ary relation** is an object R with the following features:

- n sets, called the **domains** of R, each denoted $\textit{Dom}_i(R)$ for $i = 1$ to n. Note that we don't use the word codomain any more.

- a subset of the Cartesian product of the domains called the **graph/extension** of R and denoted simply by R.

$$R \subseteq \textit{Dom}_1(R) \times \textit{Dom}_2(R) \times \ldots \times \textit{Dom}_n(R)$$

To introduce a new n-ary relation, we specify its domains

$$R : D_1 \leftrightarrow D_2 \leftrightarrow \ldots \leftrightarrow D_n$$

and its graph/extension

$$R \triangleq \ldots\text{some specification of a set of } n\text{-tuples}\ldots$$

n-ary relations, cont'd.

To show that objects $a_1, a_2, \ldots, a_n$ are elements of an n-ary relation R, we write

$$\langle a_1, a_2, \ldots, a_n \rangle \in R$$

or

$$R(a_1, a_2, \ldots, a_n)$$

Example of a 3-ary relation

The relation *Employees* has the following domains:

$$\begin{aligned} \mathit{Names} &\triangleq \{\mathit{Ann, Bill, Clare, \ldots, Zoë}\} \\ \mathit{Genders} &\triangleq \{\mathit{Male, Female}\} \\ \mathit{MarStatus} &\triangleq \{\mathit{Single, Married}\} \end{aligned}$$

$$\mathit{Employees} : \mathit{Names} \leftrightarrow \mathit{Genders} \leftrightarrow \mathit{MarStatus}$$

$$\begin{aligned} &\mathit{Employees} \triangleq \\ &\quad \{\langle \mathit{Ann, Female, Single,} \rangle, \\ &\quad\ \langle \mathit{Bill, Male, Single,} \rangle, \\ &\quad\ \langle \mathit{Clare, Female, Married} \rangle, \ldots\} \end{aligned}$$

n-ary relations, cont'd.

n-ary relations crop up a great deal in your DIS course, particularly in connection with databases.

There you will see that a very common depiction of n-ary relations is to use a tabular format.

Tabular representation

EMPLOYEES		
EmpNames	EmpGenders	EmpMarStatus
Ann	Female	Single
Bill	Male	Single
Clare	Female	Married
⋮	⋮	⋮

There are many differences though between mathematical relations and database relations. Firstly, database relations come with an amount of extra terminology (e.g. 'intension', 'extension', 'attribute', etc.). Secondly, the set of tuples in the graph/extension may change over time as updates are made to the database relation; this is not allowed in maths.

Most of what we said about binary relations can be applied to n-ary relations if desired, but would require a bit of re-working.

n-ary functions

An n-ary function would be introduced as follows:

$$F : A_1 \times A_2 \times \ldots \times A_n \rightarrow A_{n+1}$$

i.e.

$$\textit{Dom}(F) = A_1 \times A_2 \times \ldots \times A_n$$

$$\textit{Cod}(F) = A_{n+1}$$

And, in the graph/extension of F, each tuple $\langle a_1, a_2, \ldots, a_n \rangle$ from *Dom(F)* will map to no more than one object in *Cod(F)*.

Note that an n-ary function is a $(n+1)$-ary relation. So, e.g., a unary function is a binary relation; a binary function is a ternary relation; a ternary function is a 4-ary relation; and so on.

Examples of n-ary functions

Addition, subtraction, multiplication and division on $\mathbb{N}$ (say) binary functions (3-ary functional relations):

$$+ : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$$

$$* : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$$

A timetable is a 4-ary relation, and, if there are no double bookings, then it is a 3-ary function:

$$\textit{Timetable} : \textit{Day} \times \textit{Time} \times \textit{Room} \to \textit{Course}$$

E.g. here is a maplet

$$\langle \textit{Tue}, 12.15 - 13.15, \textit{B/B006} \rangle \mapsto \textit{MIP}$$

Using our newer way of showing maplets, we would write:

$$\textit{Timetable(Tue, 12.15-13.15, B/B006)} = \textit{MIP}$$

Note that you do not have to write

$$\textit{Timetable}(\langle \textit{Tue, 12.15-13.15, B/B006} \rangle) = \textit{MIP}$$

although, if we did, it might make it clearer what's actually happening here: the function maps from 3-tuples.

PROPOSITIONAL LOGIC

In this section of the course, we introduce logic in general, and then look at one particular logic, propositional logic. We use this to introduce many of the main ideas that are to be found in every logic. We will look at how to translate simple sentences of English into syntactically well-formed expressions of this logic, how to evaluate such expressions, and how to show the validity of arguments in a number of ways.

You are warned that logic terminology and notation is quite variable between different books. I have been consistent within these notes, and have also tried to point out cases where books use different terminology and notation.

Logic

Logic is the study of reasoning. It is used to study arguments, comprising premisses and conclusions, such as the following:

1. Every elephant is grey.

2. Clyde is an elephant.

3. Therefore, Clyde is grey.

This a valid argument. Why? Because: If all the premisses are true, then the conclusion must be true. This definition of a valid argument says nothing about whether the premisses are in fact true: their truth is not *necessary* for the validity of the argument, as witnessed by the validity of the following argument (despite the falsity of its first premiss):

1. Every elephant is pink.

2. Clyde is an elephant.

3. Therefore, Clyde is pink.

Nor is truth of the premisses *sufficient* for validity of an argument, as witnessed by the invalidity of the following (despite the truth of both premisses):

1. Every elephant is grey.

2. Martie is grey.

3. Therefore, Martie is an elephant.

(Martie could be a grey mouse, for example.)

The validity of an argument rests on its *form*, not the truth of the sentences that comprise it.

The uses of logic in Computer Science

- Circuit design
- Boolean data types
- Program specification
- Program verification
- Logic programming
- Automated theorem-proving
- Artificial Intelligence

Symbolic logic

For most of the uses of logic, for reasons we discussed at the start of the course, a well-defined, precise, unambiguous notation is needed, leading us to what is referred to as **symbolic logic**.

In fact, there is a large variety of symbolic logics. We look at

- Propositional logic

- First-order predicate logic

We choose to study these because (a) they are essential precursors to the study of more exotic logics, and (b) they have been immensely productive in all the uses of logic we mentioned on the previous slide.

Since symbolic logic tries to formalise everyday reasoning to some extent, there will be connections between the logics we study and English, which we will note as they arise. However, symbolic logics are formal languages and so the connection will be less than perfect. Therefore, you must learn the formal semantics of the language. Students who try to 'get by' using informal paraphrases will regularly be led astray.

Statements

The two logics we look at (like most varieties of logic) give analyses of sentences that make **statements**. (This is not a serious restriction: these are the sentences most often used in arguments and in the sort of descriptions we would use to specify a software system, for example.) Statements state a fact, or describe a state of affairs. Such sentences in English are also called 'declarative' or 'indicative' sentences. E.g.:

- It is sunny
- $4 > 5$
- Roses are red and violets are blue

So, we exclude questions, commands, requests, instructions, and so on. E.g.:

- Is it sunny?
- `x := x + 1`
- Get off my case!

A statement will be true or false. It must be one or the other. (It can't be both, nor can it be undefined or some halfway house). You can see we are already diverging from English, in which statements of less than determinate truth value are commonplace. True (T) and false (F) are referred to as **truth values**.

The structure of a statement

We form **compound statements** from **atomic statements**, by joining them using **connectives**; we form atomic statements from expressions that refer to objects, their properties and their relationships. E.g.

> 'Clyde is cold and Clyde is wet.'

This is a compound statement: the word 'and' is an example of a connective. The atomic statements are 'Clyde is cold' and 'Clyde is wet'. These atomic statements are made up of expressions such as the name 'Clyde', which denotes some object in the world, and the phrases 'is cold' and 'is wet', which denote properties of objects.

The truth value of an *atomic* statement depends on the objects, properties and relationships denoted by its subexpressions. The truth value of a *compound* statement depends on the truth values of its atomic statements.

This distinction of levels is reflected in the way in which logic is taught. First, we concentrate on the higher level. To analyse statements, we decompose them into their atomic statements, and we simply stipulate whether the atomic statements are true or false: we do not determine the truth values of the atomic statements by decomposing them into subexpressions that denote objects, properties and relationships. This latter decomposition is provisionally suspended. A statement which is subjected to this level of analysis is called a **proposition**. This kind of logic is called **propositional logic**.

When we go on to break the atomic statements down, then we are looking at the second 'stage' of logic, called **(first order) predicate logic**.

Truth-functional connectives

We make a further simplification, one which makes us diverge further from English.

The meaning (truth value) of a compound statement depends *only* on the meanings (truth values) of the statements of which it is composed.

We must therefore restrict our attention to connectives that are **truth-functional**.

> ‘Clyde is cold and Clyde is wet.’

Knowing the truth value of ‘Clyde is cold’ and of ‘Clyde is wet’ allows us to determine the truth value of ‘Clyde is cold and Clyde is wet’.

> ‘Clyde is cold because Clyde is wet.’

Knowing the truth value of ‘Clyde is cold’ and of ‘Clyde is wet’ would still not be enough for us to determine the truth value of ‘Clyde is cold because Clyde is wet’: we would still know nothing about the reasons for his coldness.

So, ‘and’ is a truth-functional connective but ‘because’ is not. (At least, as far as these examples show.)

This restriction gives simplicity and yet logics that use only truth-functional connectives have proved very productive.

Propositional logic

Given the limited amount of decomposition of statements that we are allowing for the present (i.e. our atomic statements are not decomposed into subexpressions that denote objects, properties and relationships, but instead are merely stipulated as being true or false), an atomic proposition may just as well be represented by a name such as p, p_1 or q.

These names are called **propositional symbols**.

If desired, they can be related to English statements, as follows:

$$p \triangleq \text{Clyde is cold}$$

$$q \triangleq \text{Clyde is wet}$$

We now look at the syntax and semantics of propositional logic.

The syntax of propositional logic

The expressions of this language are called **well-formed formulas** or **wffs**.

1. Every **propositional symbol** is a wff. (We use $p, p_1, p_2, p_3, \ldots, q, q_1, q_2, q_3 \ldots$ as propositional symbols.)

2. For any wff W, the following is also a wff:

$$\neg W$$

3. For any two wffs W_1 and W_2, the following are also wffs:

$$W_1 \wedge W_2$$

$$W_1 \vee W_2$$

$$W_1 \Rightarrow W_2$$

$$W_1 \Leftrightarrow W_2$$

The syntax of propositional logic, cont'd.

4. Any wff may be enclosed in parentheses and will still be a wff.

5. No other string of symbols is a wff.

Rules (2) and (3) form compound wffs from other wffs (either atomic wffs from rule (1) or other compound wffs from rules (2), (3) and (4)).

The symbols $\neg$, $\wedge$, $\vee$, $\Rightarrow$ and $\Leftrightarrow$ are the **connectives**.

Here are examples of wffs:

$$\begin{array}{ccc}
p & q & p_1 \\
(\neg p) & (\neg q) & (\neg(\neg q)) \\
(p \wedge p) & (p \vee q) & (p_1 \Rightarrow (\neg q_1)) \\
(p \Leftrightarrow (q_1 \vee q_2)) & ((p_1 \wedge p_2) \vee (q_1 \wedge q_2)) & (\neg((p_1 \wedge p_2) \vee (q_1 \wedge q_2)))
\end{array}$$

Precedence and associativity

When it comes to evaluation (semantics) of a wff, we will need to know which subwffs to evaluate first.

- Parentheses determine evaluation order.

- Precedence rules can be given:

  $$\text{Highest } \neg \lor \land \Rightarrow \Leftrightarrow \text{ Lowest}$$

  e.g. $p \lor q \land r \Rightarrow p$ evaluates as $((p \lor q) \land r) \Rightarrow p$

- Associativity rules can be given:

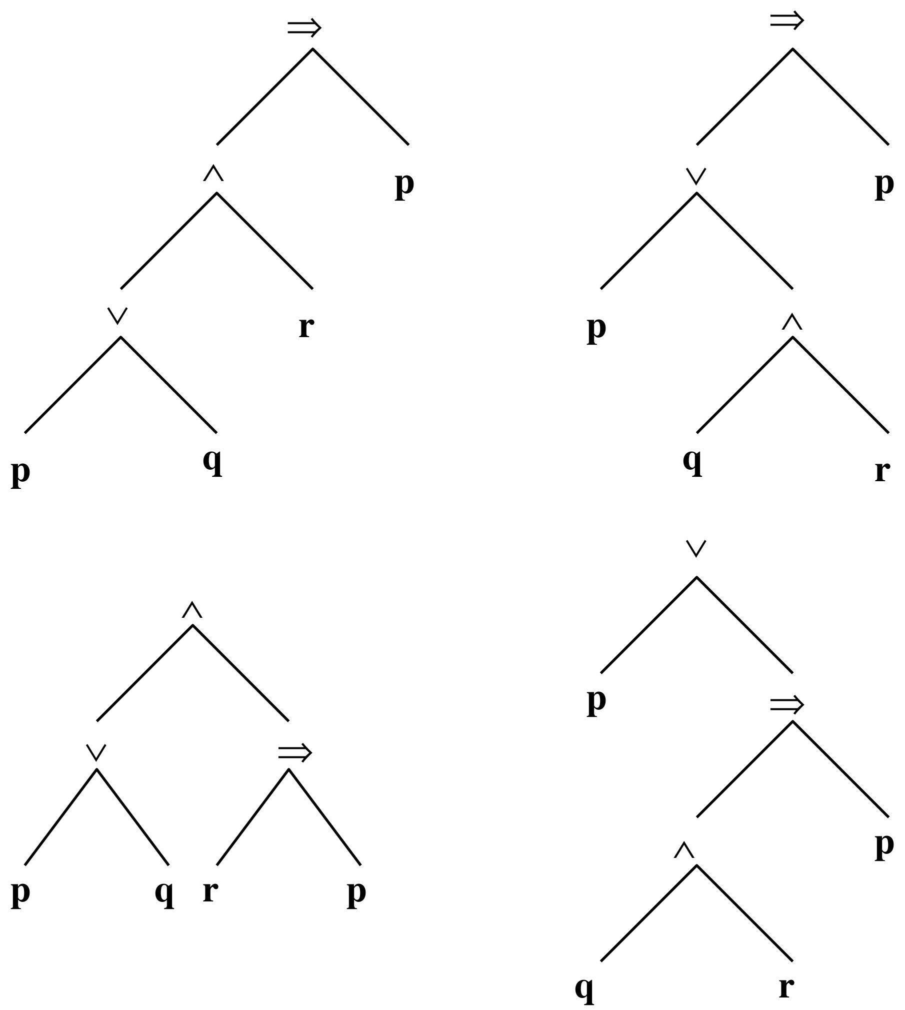
Associate to the left:	$\land \lor \Leftrightarrow$
Associate to the right:	$\Rightarrow$

  e.g. $p \land q \land r \Rightarrow p \Rightarrow r$ evaluates as $((p \land q) \land r) \Rightarrow (p \Rightarrow r)$

On the whole, we use a lot of parentheses in this course. But, you must remember that $\neg$ has highest precedence. This is very useful in reducing the number of parentheses. So, e.g., $\neg p \land q$ evaluates as $(\neg p) \land q$, and, for the other evaluation order, you would write $\neg(p \land q)$.

Trees

We can, of course, unambiguously depict a wff in a tree diagram. This will turn out to be particularly useful when we move on to first-order predicate logic.



Annotate these trees during the lecture with the wffs they depict.

The semantics of propositional logic

Wffs represent propositions, and these are either true or false.

Atomic wffs

The truth values of atomic wffs are *stipulated*. In other words, we do not spend time debating whether a proposition symbol such as p,

$$p \stackrel{\triangle}{=} \text{The moon is made of cheese}$$

denotes a true proposition or a false one. Instead, I stipulate what its truth value is. (In fact, a significant part of what we do will involve considering both what happens when p denotes a true proposition and when p denotes a false proposition, so these stipulations aren't required too often anyway.)

A stipulation of the truth values of atomic wffs is called an **interpretation** of the atomic wffs. Some books use the phrases **truth assignment** or **value assignment** or **truth valuation** instead of 'interpretation'.

Example (in two different notations):

$$p \text{ has truth value } T;\ q \text{ has truth value } T;\text{ and, } r \text{ has truth value } F$$

$$\mathcal{I}(p) \text{ is } T; \qquad \mathcal{I}(q) \text{ is } T; \qquad \mathcal{I}(r) \text{ is } F$$

Both of these ways of writing this interpretation are sentences of the metalanguage. Symbols used in these sentences, other than the proposition symbols themselves, are metalanguage symbols. In particular, $\mathcal{I}$ is a metalanguage symbol, acting as shorthand for the more verbose version. I've tried to emphasise that it is a metalanguage symbol by using a different font. I've also resisted using $=$ instead of 'is'. We've already used $=$ as an object language symbol (e.g. in set theory), so we don't now want to use it here as a metalanguage symbol.

Compound wffs

The truth value of a compound wff is determined by the truth values of its component wffs. So, we must now look at each connective in turn and see how a wff that is formed using that connective receives its truth value from the truth values of the subwffs of the compound.

Negation

Given any wff W, another wff, called the **negation** of W, can be formed, and is denoted

$$\neg W$$

This is read as 'not W'. Some books use

$$\sim W \qquad W' \qquad \overline{W} \qquad \textbf{not}\ W$$

For any wff W,

$$\mathcal{I}(\neg W) \stackrel{\triangle}{=} \text{ if } \mathcal{I}(W) \text{ is } T \\ \text{then } F \\ \text{else } T$$

i.e. to determine the truth value of $\neg W$, you must determine the truth value of W, i.e. $\mathcal{I}(W)$. Then, $\mathcal{I}(\neg W)$ is T if $\mathcal{I}(W)$ is F and $\mathcal{I}(\neg W)$ is F if $\mathcal{I}(W)$ is T.

The above is a prefectly good specification of the semantics of $\neg W$, but a tabular representation is more often used, as it is clearer:

W	$\neg W$
T	F
F	T

You can see that this says the same thing as we said above.

Negation, cont'd.

$$\begin{aligned} p_1 &\triangleq \text{ Paris is in France} \\ p_2 &\triangleq 2+2=5 \\ p_3 &\triangleq \text{ Clyde is intelligent} \end{aligned}$$

Consider the interpretation

$$\mathcal{I}(p_1) \text{ is } T; \quad \mathcal{I}(p_2) \text{ is } F; \quad \mathcal{I}(p_3) \text{ is } T$$

then

$$\mathcal{I}(\neg p_1) \text{ is } F; \quad \mathcal{I}(\neg p_2) \text{ is } T; \quad \mathcal{I}(\neg p_3) \text{ is } F$$

Negation, cont'd.

In English, sentences are negated using phrases such as 'It is not the case that', or, perhaps, 'It is false that' and 'It is incorrect that'.

So, using the same propositional symbols from the previous overhead, we have:

$\neg p_1$	'It is not the case that Paris in in France'
$\neg p_2$	'It's not the case that $2 + 2 = 5$'

There are other forms of negation in English, where individual words and phrases are negated using 'not' or prefixes such as 'un-' or 'dis-'. Sometimes, these can be straightforwardly represented using propositional logic negation, e.g.

$\neg p_1$	'Paris is not in France'
$\neg p_2$	$2 + 2 \neq 5$

Others are less clear. Should 'It's not the case that Clyde is intelligent', 'Clyde isn't intelligent', 'Clyde is unintelligent' and 'Clyde is stupid' all be rendered as $\neg p_3$?

Or consider $p_4 \triangleq$ All elephants are intelligent.

Should 'It's not the case that all elephants are intelligent', 'Not all elephants are intelligent', 'All elephants are not intelligent', "All elephants are unintelligent' and 'All elephants are stupid' all be rendered by $\neg p_4$?

Negation: (easy) class exercise

$$q_1 \triangleq \text{Clyde likes Flopsy}$$
$$q_2 \triangleq \text{Clyde is happy}$$

Paraphrase into English

$$\neg q_1$$

Translate into propositional logic

It's not the case that Clyde is happy.
Clyde isn't happy.
Clyde is unhappy.
Clyde is sad.

Conjunction

Any two wffs W_1 and W_2 can be combined to form a compound wff, called the **conjunction** of W_1 and W_2, and denoted

$$W_1 \wedge W_2$$

W_1 and W_2 are called the **conjuncts**. Some books use

$$W_1.W_2 \qquad W_1\&W_2 \qquad W_1 \textbf{ and } W_2 \qquad W_1W_2$$

For any wffs W_1 and W_2,

$$\mathcal{I}(W_1 \wedge W_2) \stackrel{\triangle}{=} \begin{array}{l}\text{if } \mathcal{I}(W_1) \text{ is } T \text{ and } \mathcal{I}(W_2) \text{ is } T \\ \text{then } T \\ \text{else } F\end{array}$$

Again, the tabular presentation is clearer, so this is what you should learn:

W_1	W_2	$W_1 \wedge W_2$
T	T	T
T	F	F
F	T	F
F	F	F

Note that we need four rows to capture all the possible combinations of truth values for W_1 and W_2.

Conjunction, cont'd.

$$\begin{aligned} p_1 &\triangleq \text{Paris is in France} \\ p_2 &\triangleq \text{York is in England} \\ p_3 &\triangleq 2+2=5 \end{aligned}$$

Consider

$$\mathcal{I}(p_1) \text{ is } T; \quad \mathcal{I}(p_2) \text{ is } T; \quad \mathcal{I}(p_3) \text{ is } F$$

then, e.g.,

$$\mathcal{I}(p_1 \wedge p_2) \text{ is } T; \quad \mathcal{I}(p_1 \wedge p_3) \text{ is } F; \quad \mathcal{I}(p_3 \wedge p_3) \text{ is } F$$

The logical behaviour of $\wedge$ is in some accord with that of the English word 'and' (when used to join two sentences):

$p_1 \wedge p_3$	'Paris is in France and $2+2=5$'
	'Both, Paris is in France and $2+2=5$'

Conjunction: (easy) class exercise

$$q_1 \triangleq \text{Clyde likes Flopsy}$$
$$q_2 \triangleq \text{Clyde is happy}$$

Paraphrase into English

$$q_1 \wedge \neg q_2$$

Translate into propositional logic

It's not the case that both Clyde likes Flopsy and Clyde is happy.

Conjunction, cont'd.

Certain uses of words such as 'but', 'while', although', 'though' and 'whereas' can also be rendered as logical conjunction.

$p_1 \wedge p_2$	'Paris is in France but York is in England'
	'Paris in in France whereas York is in England'
	'Although Paris is in France, York is in England'

Some of these words, e.g. 'while' also have completely different meanings that concern the timing of events and these are clearly not rendered using $\wedge$.

Also, $\wedge$ clearly fails to capture the *additional* meaning conveyed by the words 'but', 'while', 'although', 'though' and 'whereas', i.e. that the two sentences express contrasting propositions.

Similarly, not all uses of 'and' have their full meaning captured by $\wedge$. For example:

> 'Batman pulled off his socks and jumped into bed.'
>
> 'Batman jumped into bed and pulled off his socks.'

In these, an ordering over the two events being described is also conveyed, but will not be conveyed if we translate these sentences using $\wedge$.

The word 'and' can also be used to join phrases and words other than full sentences. Sometimes these can still be translated into propositional logic:

> 'Batman and Robin are superheroes.'
>
> 'Batman is a superhero and Robin is a superhero.'

But other times, such a paraphrase is not necessarily possible, e.g.

> 'Batman and Robin are fun at parties.'

It might be that they're only fun if they are together, so paraphrasing as 'Batman is fun at parties and Robin is fun at parties' is not necessarily correct.

Disjunction

Any two wffs W_1 and W_2 can be combined to form a compound wff, called the **disjunction** of W_1 and W_2, and denoted

$$W_1 \vee W_2$$

This is read as 'W_1 or W_2' or 'W_1 or W_2 or both' or 'W_1 inclusive-or W_2'. W_1 and W_2 are called the **disjuncts**. Some books use

$$W_1 + W_2 \qquad W_1 \textbf{ or } W_2$$

For any wffs W_1 and W_2,

$$\mathcal{I}(W_1 \vee W_2) \stackrel{\triangle}{=} \text{if } \mathcal{I}(W_1) \text{ is } T \text{ or } \mathcal{I}(W_2) \text{ is } T \text{ (or both)} \\ \text{then } T \\ \text{else } F$$

More simply…

W_1	W_2	$W_1 \vee W_2$
T	T	T
T	F	T
F	T	T
F	F	F

Disjunction, cont'd.

$$
\begin{aligned}
p_1 &\triangleq \text{Paris is in France}\\
p_2 &\triangleq \text{York is in England}\\
p_3 &\triangleq 2+2=5
\end{aligned}
$$

Consider

$$\mathcal{I}(p_1) \text{ is } T; \qquad \mathcal{I}(p_2) \text{ is } F; \qquad \mathcal{I}(p_3) \text{ is } F$$

"Hold on!" you cry. "p_2 is T in our world!"

"I know," I reply. "I just wanted to emphasise the point that *I stipulate* the truth values and you have to go along with them."

then, e.g.

$$\mathcal{I}(p_1 \vee p_2) \text{ is } T; \qquad \mathcal{I}(p_2 \vee p_3) \text{ is } F; \qquad \mathcal{I}(p_1 \vee p_1) \text{ is } T$$

The logical behaviour of $\vee$ is in some accord with that of the English word 'or' (when used to join two sentences):

$p_1 \vee p_3$ 'Paris is in France or $2+2=5$ (or both)'

Disjunction, cont'd.

The English word 'or' has at least two distinct uses:

'W_1 or W_2 or both'	**inclusive-or**
'W_1 or W_2 but not both'	**exclusive-or**

Inclusive-or is the one we have introduced already.

Exclusive-or is also truth-functional, so we could have a connective for it. $\oplus$ is the symbol that is often used. We would give this connective a semantics just like that of $\vee$ except that in the first row of the table, where both $\mathcal{I}(W_1)$ and $\mathcal{I}(W_2)$ are T, $\mathcal{I}(W_1 \oplus W_2)$ would be F, not T. Unless otherwise stated, we are using inclusive-or in this course. Exclusive-or can be captured using $\vee$, $\wedge$ and $\neg$ together (see below).

We see again the advantage of using a formal language in making clear an ambiguity in English. Of course, this means that we need to be extra careful when translating sentences of English into logic: should we use $\vee$ or $\oplus$? Generally, $\vee$ will do. However, sentences that begin with the word 'either' often signal use of $\oplus$. Also, many people would say that the word 'unless' (assuming that it is truth-functional at all) translates as $\oplus$. For example, 'We will go to the cinema unless we go for a pizza' is probably a sentence being used to convey an exclusive-or: if the speaker of the sentence eventually went to the cinema *and* had a pizza (which is perfectly possible, of course) then I would say that s/he had uttered a false statement. Do you agree? (If not, then you should translate this sentence using $\vee$.)

$p_1 \vee p_2$	'Paris is in France or York is in England (or both)'
$(p_1 \vee p_2) \wedge \neg(p_1 \wedge p_2)$	'Paris is in France or York is in England (but not both)' '*Either* Paris is in France or York is in England' 'Paris is in France unless York is in England'

'or', like 'and', can be used to join phrases other than sentences. Only some of these uses of 'or' can be translated using the disjunction connective.

Conditional

Any two wffs W_1 and W_2 can be combined to form a compound wff denoted

$$W_1 \Rightarrow W_2$$

called a **conditional**. This can be read as 'if W_1 then W_2'. W_1 is called the **antecedent** and W_2 is called the **consequent**. Some books use

$$W_1 \rightarrow W_2 \qquad W_1 \supset W_2$$

instead of $W_1 \Rightarrow W_2$. Since we've already used these symbols elsewhere (Question: where?), I decided to avoid them here.

Note also that many books call this connective **implication** or **material implication**, instead of the 'conditional'. I try to avoid this because it encourages students to ignore the formal semantics (below) and try to 'get by' using informal paraphrases into English, resulting in them being led astray by this somewhat inappropriate paraphrase.

For any wffs W_1 and W_2,

$$\mathcal{I}(W_1 \Rightarrow W_2) \stackrel{\triangle}{=} \begin{array}{l}\text{if } \mathcal{I}(W_1) \text{ is } F \text{ or } (W_2) \text{ is } T \\ \text{then } T \\ \text{else } F\end{array}$$

More simply…

W_1	W_2	$W_1 \Rightarrow W_2$
T	T	T
T	F	F
F	T	T
F	F	T

Conditional, cont'd.

I shall attempt an explanation of why this connective is thought to do something like the job of English 'if…then'. Consider

$$p \triangleq \text{I win the election}$$
$$q \triangleq \text{Taxes will fall}$$

A politician utters 'If I win the election, then taxes will fall', i.e. $p \Rightarrow q$. Under what circumstances would this be a lie?

- Suppose s/he wins and taxes fall:

  $\mathcal{I}(p)$ is T; $\mathcal{I}(q)$ is T; so $\mathcal{I}(p \Rightarrow q)$ is T

  The semantics says s/he was truthful. Quite right.

- Suppose s/he wins and taxes rise or stay the same:

  $\mathcal{I}(p)$ is T; $\mathcal{I}(q)$ is F; so $\mathcal{I}(p \Rightarrow q)$ is F

  The semantics says s/he lied in this case. Quite right too.

- Suppose s/he loses and taxes fall:

  $\mathcal{I}(p)$ is F; $\mathcal{I}(q)$ is T; so $\mathcal{I}(p \Rightarrow q)$ is T

  The semantics says s/he was truthful. See below.

- Suppose s/he loses and taxes rise or stay the same:

  $\mathcal{I}(p)$ is F; $\mathcal{I}(q)$ is F; so $\mathcal{I}(p \Rightarrow q)$ is T

  The semantics says s/he was truthful. See below.

In these last two cases, it would seem very harsh to say that s/he lied, i.e. that $\mathcal{I}(p \Rightarrow q)$ is F. Of course, it also feels a bit bizarre to say that $\mathcal{I}(p \Rightarrow q)$ is T. But every wff is either T or F. So we need a convention and the convention is that $\mathcal{I}(p \Rightarrow q)$ is T in both these cases.

Conditional, cont'd.

$$\begin{aligned} p_1 &\triangleq \text{Paris is in France} \\ p_2 &\triangleq \text{York is in England} \\ p_3 &\triangleq 2+2=5 \end{aligned}$$

The connection between English and this connective is tenuous at the best of times.

$p_1 \Rightarrow p_2$ 'If Paris is in France then York is in England'
'Paris is in France implies York is in England' (avoid!!!)
'Paris is in France only if York is in England'
'York is in England if Paris is in France'
'Paris being in France is a sufficient condition for York to be in England'
'York being in England is a necessary condition for Paris to be in France'

Certain uses of 'when' and 'whether' might also translate using the conditional.

In the above, I've deliberately avoided more traditional examples where the antecedent and the consequent are related in some *causal* fashion. For example, if $p_4 \triangleq$ 'It is raining' and $p_5 \triangleq$ 'I get wet' then $p_4 \Rightarrow p_5$ can be paraphrased as 'If it is raining, then I get wet'. This sounds very sensible. But now, you are almost certainly thinking that $\Rightarrow$ says that the antecedent *causes* the consequent. But, we have already seen that 'because' is not truth-functional, so treating $\Rightarrow$, which is supposed to be a truth-functional connective, as having anything to do with causality must be wrong.

Biconditional

Any two wffs W_1 and W_2 can be combined to form a compound wff, denoted

$$W_1 \Leftrightarrow W_2$$

which is called a **biconditional**. This is read as 'W_1 if and only if W_2'. Some books use

$$W_1 \leftrightarrow W_2 \qquad W_1 \equiv W_2$$

instead of $W_1 \Leftrightarrow W_2$. We'll need the symbol $\equiv$ as part of our metalanguage later, so we don't want to use it for something else here.

Some books call this connective the 'equivalence' connective but that risks confusion with the metalanguage too (as we shall see). And others call it **bi-implication**. But I'm avoiding the word 'implies' altogether.

For any wffs W_1 and W_2,

$$\mathcal{I}(W_1 \Leftrightarrow W_2) \stackrel{\triangle}{=} \text{if } \mathcal{I}(W_1) \text{ is the same as } \mathcal{I}(W_2) \text{ then } T \text{ else } F$$

More simply…

W_1	W_2	$W_1 \Leftrightarrow W_2$
T	T	T
T	F	F
F	T	F
F	F	T

Biconditional, cont'd.

$$\begin{aligned} p_1 &\triangleq \text{Paris is in France} \\ p_2 &\triangleq \text{York is in England} \\ p_3 &\triangleq 2+2=5 \end{aligned}$$

Consider

$$\mathcal{I}(p_1) \text{ is } T; \quad \mathcal{I}(p_2) \text{ is } T; \quad \mathcal{I}(p_3) \text{ is } F$$

then, e.g.,

$$\mathcal{I}(p_1 \Leftrightarrow p_2) \text{ is } T; \quad \mathcal{I}(p_1 \Leftrightarrow p_3) \text{ is } F; \quad \mathcal{I}(p_3 \Leftrightarrow p_3) \text{ is } T$$

Biconditional, cont'd.

The following show possible English words and phrases that might translate as the biconditional.

$p_1 \Leftrightarrow p_2$	'Paris is in France if and only if York is in England' 'Paris being in France is a necessary and sufficient condition for York to be in England' 'Paris is in France provided York is in England' 'Paris is in France exactly if York is in England' Paris is in France just if York is in England'

Certain uses of 'when' (e.g. 'just when', 'exactly when') may also translate as the biconditional.

In maths textbooks, the English phrase 'if and only if', which sees fairly regular use in the metalanguage, is often abbreviated to 'iff'.

Translations of more complex sentences

$$p_1 \stackrel{\triangle}{=} \text{Paris is in France}$$
$$p_2 \stackrel{\triangle}{=} \text{York is in England}$$
$$p_3 \stackrel{\triangle}{=} 2 + 2 = 5$$

$(p_1 \land p_2) \Rightarrow \neg p_3$	'If Paris is in France and York is in England, then $2 + 2 \neq 5$

But, beware of ambiguity in English compound statements. English has no precedence and associativity rules to sort this out!

$\neg((p_1 \lor p_2) \Rightarrow p_3)$	
$(p_1 \lor p_2) \Rightarrow \neg p_3$	'It is not the case that
$\neg(p_1 \Rightarrow p_2) \lor p_2$	$2 + 2 = 5$ if Paris is in
$(p_1 \Rightarrow \neg p_3) \lor p_2$	France or York is in
$\neg((p_1 \Rightarrow p_3) \lor p_2)$	England'

The semantics of (more complex) compound wffs

The previous overheads have included some examples in which we have computed the semantics of simple compound wffs (e.g. $p_1 \wedge p_2$ and $p_3 \Leftrightarrow p_3$). Here, we look at compound wffs that have more internal structure to them.

We will determine $\mathcal{I}(p_1 \wedge (p_2 \Rightarrow p_3))$ given

$$\mathcal{I}(p_1) \text{ is } T; \quad \mathcal{I}(p_2) \text{ is } F; \quad \mathcal{I}(p_3) \text{ is } T$$

Parentheses determine that the first subwff to be evaluated is $p_2 \Rightarrow p_3$. Because $\mathcal{I}(p_2)$ is F and $\mathcal{I}(p_3)$ is T,

$$\mathcal{I}(p_2 \Rightarrow p_3) \text{ is } T$$

from row three of the table defining the semantics of $\Rightarrow$.

And now we know that $\mathcal{I}(p_1)$ is T and $\mathcal{I}(p_2 \Rightarrow p_3)$ is T, we can determine that

$$\mathcal{I}(p_1 \wedge (p_2 \Rightarrow p_3)) \text{ is } T$$

from the first row of the table defining the semantics of $\wedge$.

So the whole wff is true (*for this interpretation of the atomic wffs*): this wff is true *under* this interpretation.

Given a wff, W, and an interpretation of the wff's atomic wffs, $\mathcal{I}$, we say that $\mathcal{I}$ **satisfies** W if and only if $\mathcal{I}(W)$ is T i.e. if the wff is true under that interpretation then that interpretation satisfies that wff.

Hence, $\mathcal{I}$ from above satisfies $p_1 \wedge (p_2 \Rightarrow p_3)$.

The semantics of (more complex) compound wffs, cont'd.

Let's do it all again for a different interpretation of the atomic wffs:

$$\mathcal{I}(p_1) \text{ is } T; \quad \mathcal{I}(p_2) \text{ is } T; \quad \mathcal{I}(p_3) \text{ is } F$$

This time, we'll set things out in a tabular fashion, which results in a tidier presentation.

We draw a table with a column for each subwff of the wff:

p_1	p_2	p_3	$(p_2 \Rightarrow p_3)$	$p_1 \wedge (p_2 \Rightarrow p_3)$

Then we can fill in the truth values for the atomic wffs (using our new interpretation $\mathcal{I}$ from above):

p_1	p_2	p_3	$(p_2 \Rightarrow p_3)$	$p_1 \wedge (p_2 \Rightarrow p_3)$
T	T	F		

Now we fill in the next column with reference to the semantics of $\Rightarrow$. This time, the second row of the table that defines the semantics of $\Rightarrow$ applies:

p_1	p_2	p_3	$(p_2 \Rightarrow p_3)$	$p_1 \wedge (p_2 \Rightarrow p_3)$
T	T	F	F	

And now we can fill in the final column using the semantics of $\wedge$. Since $\mathcal{I}(p_1)$ is T and $\mathcal{I}(p_2 \Rightarrow p_3)$ is F (from the third column above), $\mathcal{I}(p_1 \wedge (p_2 \Rightarrow p_3))$ is F (second row of the table defining the semantics of $\wedge$):

p_1	p_2	p_3	$(p_2 \Rightarrow p_3)$	p_1 $\wedge$ $(p_2 \Rightarrow p_3)$
T	T	F	F	F
				$\triangle$

Obviously, the fourth column here is now not really important. It was just part of our 'working'.

So the whole wff is false under this interpretation, i.e. this interpretation does not satisfy $p_1 \wedge (p_2 \Rightarrow p_3)$.

The semantics of (more complex) compound propositions, cont'd.

We'll do it yet again, with yet another interpretation:

$$\mathcal{I}(p_1) \text{ is } F; \quad \mathcal{I}(p_2) \text{ is } F; \quad \mathcal{I}(p_3) \text{ is } F$$

And we'll illustrate another way of doing the 'working'.

We draw a table as follows:

p_1	p_2	p_3	p_1	$\wedge$	$(p_2$	$\Rightarrow$	$p_3)$

We use the interpretation above to fill in the truth values for the atomic wffs:

p_1	p_2	p_3	p_1	$\wedge$	$(p_2$	$\Rightarrow$	$p_3)$
F	F	F					

Then we copy these values to columns headed by occurrences of those wffs in the compound wff:

p_1	p_2	p_3	p_1	$\wedge$	$(p_2$	$\Rightarrow$	$p_3)$
F	F	F	F		F		F

Now, since parentheses dictate that the $\Rightarrow$ is evaluated first, we work out $\mathcal{I}(p_2 \Rightarrow p_3)$ (which, given that $\mathcal{I}(p_2)$ is F and $\mathcal{I}(p_3)$ is F, is T), and we place this into the column headed by the $\Rightarrow$:

p_1	p_2	p_3	p_1	$\wedge$	$(p_2$	$\Rightarrow$	$p_3)$
F	F	F	F		F	T	F

And now we evaluate the conjunction $\mathcal{I}(p_1 \wedge (p_2 \Rightarrow p_3)$ (which, given that we now know that $\mathcal{I}(p_1)$ is F and $\mathcal{I}(p_2 \Rightarrow p_3)$ is T, is F), and we write this into the column headed by $\wedge$:

p_1	p_2	p_3	p_1	$\wedge$	$(p_2$	$\Rightarrow$	$p_3)$
F	F	F	F	F	F	T	F
				$\triangle$			

Obviously, the fourth, sixth, seventh and eighth columns here are not really important: they're just 'working'. The 'result' is in the fifth column.

So, the whole wff is false under this interpretation, i.e. this interpretation does not satisfy $p_1 \wedge (p_2 \Rightarrow p_3)$.

Truth tables

Most often we need to compute the truth value of a wff under *every possible* interpretation of its atomic wffs. The tabular presentation on the previous slides can be extended to do a neat job of this: we use one row for each interpretation.

If there are n different proposition symbols in a compound wff, then there are $2 \uparrow n$ different interpretations of those atomic wffs.

The wff $p_1 \wedge (p_2 \Rightarrow p_3)$ has three different proposition symbols, so there are $2 \uparrow 3 = 8$ possible interpretations, hence our truth table will have 8 rows.

p_1	p_2	p_3	p_1	$\wedge$	$(p_2 \Rightarrow p_3)$
T	T	T		T	
T	T	F		F	
T	F	T		T	
T	F	F		T	
F	T	T		F	
F	T	F		F	
F	F	T		F	
F	F	F		F	

Notes:

1. The third, second and eighth rows are the three interpretations we looked at on the previous slides.

2. In the lectures I will show you a systematic way to make sure that none of the $2 \uparrow n$ rows is forgotten.

3. In the truth table above, I haven't shown any of my 'working'. I'll show this in the lecture.

Truth tables: examples

We will complete these truth tables during the lectures.

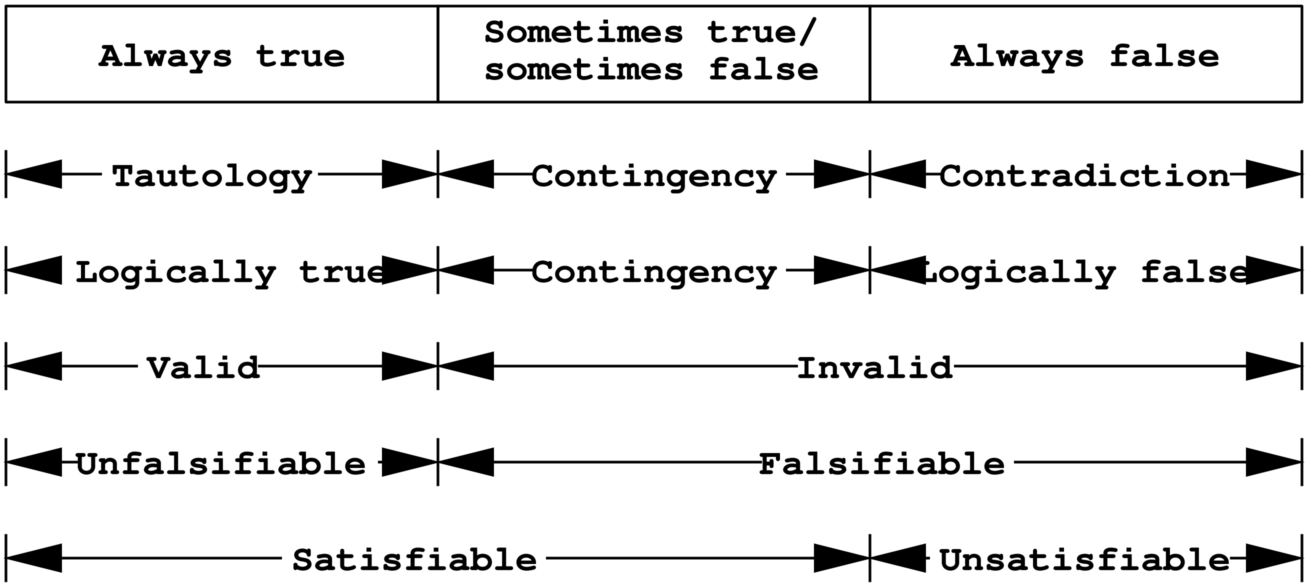
p_1	p_2	$(p_1$	$\wedge$	$p_2)$	$\Leftrightarrow$	$(p_1$	$\vee$	$p_2)$
T	T							
T	F							
F	T							
F	F							

p	p	$\vee$	$\neg$	p
T				
F				

p	p	$\wedge$	$\neg$	p
T				
F				

Valid, invalid, satisfiable and unsatisfiable

The examples from the previous slide show that some wffs evaluate to true no matter how their atomic wffs are interpreted (every interpretation satisfies them), some evaluate to false for every interpretation of their atomic wffs (no interpretation satisfies them), and others are true for some interpretations and false for others (some interpretations satisfy them, others don't.) There is some terminology for all of this. (There would be, wouldn't there?)



Some books also use 'consistent' for 'satisfiable' and 'inconsistent' for 'unsatisfiable' but this is to be avoided.

We will use just the following: valid/invalid and satisfiable/unsatisfiable. I ignore all the others to avoid terminology overload:

- A wff is **valid** if all interpretations satisfy it. Otherwise (i.e. if at least one does not satisfy it), it is **invalid**.
- A wff is **satisfiable** if there is at least one interpretation that satisfies it. Otherwise (i.e. if no interpretation satisfies it), it is **unsatisfiable**.

Note how valid and satisfiable overlap: every valid wff is satisfiable, but not every satisfiable wff is valid. Similarly, every unsatisfiable wff is invalid, but not every invalid wff is unsatisfiable.

Valid, invalid, satisfiable and unsatisfiable

To determine which of this terminology applies to a wff, W, you would need to consider $\mathcal{I}(W)$ for a number of different interpretations $\mathcal{I}$:

To determine validity, invalidity, satisfiability, unsatisfiability, draw up a truth table.

E.g.

Is $(p_1 \wedge p_2) \Leftrightarrow (p_1 \vee p_2)$ valid or invalid? Is it satisfiable or unsatisfiable?

Is $p \vee \neg p$ valid or invalid? Is it satisfiable or unsatisfiable?

Is $p \wedge \neg p$ valid or invalid? Is it satisfiable or unsatisfiable?

Question: do you always need to draw up all $2 \uparrow n$ rows of the truth table to answer questions such as those given above?

Logical consequence

We're now ready to formalise the idea of argument that we introduced at the start of this material. We use the notion of logical consequence. (Other books refer to **logical implication** and **entailment**.)

A wff W (the conclusion) is a **logical consequence** of a set of wffs A (the premisses) if and only if for *every* interpretation in which *all* the premisses are true, the conclusion is also true.

In other words, whenever all the wffs in A are true, then W must also be true.

This definition doesn't say anything about interpretations under which one or more of the wffs in A are false. In this case, we don't care whether W is true or false.

Logical consequence, cont'd.

Example: q is a logical consequence of $\{p \Rightarrow q, p\}$.

Since, we need to consider several interpretations, we need a truth table.

p	q	$p \Rightarrow q$	p	q
T	T	T	T	T
T	F	F	T	F
F	T	T	F	T
F	F	T	F	F

Look for all the interpretations where each wff in the set of premisses is true. There's only one in this example: the first row. Now, since the conclusion is also true under that interpretation, we can conclude that q *is* a logical consequence of $\{p \Rightarrow q, p\}$.

Since writing 'W is a logical consequence of A' is too much effort, we use some notation:

Notation:

$$A \models W$$

E.g.

$$\{p \Rightarrow q, p\} \models q$$

$\models$ is called the 'semantic turnstile'.

It is important to note that $\models$ is another metalanguage symbol, like $\mathcal{I}$. But you knew this already because (a) it was introduced here as shorthand for English, and English is our metalanguage, and (b) had it been part of our object language I would have introduced it when I introduced the other connectives.

Logical consequence: examples

We'll do these in the lecture.

Show that $\{\neg p, \neg q\} \models \neg(p \vee q)$

p	q	$\neg p$	$\neg q$	$\neg(p \vee q)$
T	T			
T	F			
F	T			
F	F			

Show that $\{p \vee q, q\} \models (\neg p \wedge q) \vee p$

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

Show that $\{p \Rightarrow q, q\} \not\models p$

p	q	$p \Rightarrow q$	q	p
T	T			
T	F			
F	T			
F	F			

Logical equivalence

Two wffs W_1 and W_2 are **logically equivalent** (or simply **equivalent**) if and only if $\{W_1\} \models W_2$ and $\{W_2\} \models W_1$.

In this case, we write

$$W_1 \equiv W_2$$

$\equiv$ is another metalanguage symbol. (Some books use $=_T$.)

To show $W_1 \equiv W_2$:

- draw up truth tables for W_1 and for W_2 and confirm that $\mathcal{I}(W_1)$ is the same as $\mathcal{I}(W_2)$ for each $\mathcal{I}$ i.e. they should have the same truth value in each row

Logical equivalence: examples

We'll do these in the lectures.

Show that $\neg(p \wedge q) \equiv \neg p \vee \neg q$

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

Show that $(p \Rightarrow q) \equiv \neg(p \wedge \neg q)$

p	q	$(p \Rightarrow q)$	$\neg(p \wedge \neg q)$
T	T		
T	F		
F	T		
F	F		

We can generalise equivalences to make them into 'laws'.

Laws of the algebra of propositions

Note the similarities with the algebra of sets!

For any propositions W_1, W_2 and W_3,...

Commutativity of $\wedge$ and $\vee$

$$W_1 \wedge W_2 \equiv W_2 \wedge W_1$$

$$W_1 \vee W_2 \equiv W_2 \vee W_1$$

Associativity of $\wedge$ and $\vee$

$$(W_1 \wedge W_2) \wedge W_3 \equiv W_1 \wedge (W_2 \wedge W_3)$$

$$(W_1 \vee W_2) \vee W_3 \equiv W_1 \vee (W_2 \vee W_3)$$

Laws of the algebra of propositions, cont'd.

For any propositions W_1 and W_2,...

Distributivity of $\wedge$ over $\vee$, and of $\vee$ over $\wedge$

$$W_1 \wedge (W_2 \vee W_3) \equiv (W_1 \wedge W_2) \vee (W_1 \wedge W_3)$$

$$W_1 \vee (W_2 \wedge W_3) \equiv (W_1 \vee W_2) \wedge (W_1 \vee W_3)$$

De Morgan's laws

$$\neg(W_1 \wedge W_2) \equiv \neg W_1 \vee \neg W_2$$

$$\neg(W_1 \vee W_2) \equiv \neg W_1 \wedge \neg W_2$$

Laws of the algebra of propositions, cont'd.

For any proposition W,...

Idempotence of $\wedge$ and $\vee$

$$W \wedge W \equiv W$$

$$W \vee W \equiv W$$

Identities

To show identities, we need proposition symbols that denote the wff that is always true and the wff that is always false. We use t and f respectively.

$$t \wedge W \equiv W$$

$$f \wedge W \equiv f$$

$$t \vee W \equiv t$$

$$f \vee W \equiv W$$

Question: what is the difference between T and t, and between F and f?

Laws of the algebra of propositions, cont'd.

For any proposition W,…

Involution

$$\neg\neg W \equiv W$$

Complement laws

$$W \wedge \neg W \equiv f$$

$$W \vee \neg W \equiv t$$
$$\neg t \equiv f$$
$$\neg f \equiv t$$

Laws for propositions: biconditionals and conditionals

The laws above considered only negation, conjunction and disjunction. Here are some concerning biconditionals and conditionals.

Definition of biconditional

For any propositions W_1 and W_2,

$$W_1 \Leftrightarrow W_2 \equiv (W_1 \Rightarrow W_2) \wedge (W_2 \Rightarrow W_1)$$

Definition of conditional

For any propositions W_1 and W_2,

$$W_1 \Rightarrow W_2 \equiv \neg W_1 \vee W_2$$

Confirm both of the above in your own time by drawing truth tables.

Laws for propositions: conditionals

Since intuitions about the conditional are so tricky, let's see what we learn from the following:

W_1	W_2	Conditional $W_1 \Rightarrow W_2$	Converse $W_2 \Rightarrow W_1$
T	T	T	T
T	F	F	T
F	T	T	F
F	F	T	T

W_1	W_2	Inverse $\neg W_1 \Rightarrow \neg W_2$	Contrapositive $\neg W_2 \Rightarrow \neg W_1$
T	T	T	T
T	F	T	F
F	T	F	T
F	F	T	T

So,

Contrapositive law

For any propositions W_1 and W_1,

$$W_1 \Rightarrow W_2 \equiv \neg W_2 \Rightarrow \neg W_1$$

The truth-table above also shows that a conditional is not equivalent to its converse: you would agree that 'If it is raining, then I get wet' does not have the same truth conditions as 'If I get wet, then it is raining'.

The laws of the algebra of propositions: their use

The laws give another ‘method’ for showing $W_1 \equiv W_2$

- use algebraic manipulation.

E.g. to show $p \vee q \equiv \neg(\neg p \wedge \neg q)$ algebraically.

$$\begin{aligned} \text{RHS} \quad &= \neg(\neg p \wedge \neg q) \\ &= \neg\neg(p \vee q) && \text{(by De Morgan's Law)} \\ &= p \vee q && \text{(by Involution)} \\ &= \text{LHS} \end{aligned}$$

Deduction systems

To show $\{W_1, W_2, \ldots, W_n\} \models W$, we must check all interpretations of the wffs, i.e. we must draw a truth table for $\{W_1, W_2, \ldots, W_n\} \models W$

Checking all interpretations is manageable for wffs that have 2 or 3 proposition symbols, but becomes increasingly difficult when there are more.

This motivates the development of apparatus that enables reasoning to take place at the purely syntactic level, before interpretations are considered.

The apparatus is called a **(formal) deduction system**, a **proof theory**, an **inference system** or a **logical calculus**.

Such systems can be broadly classified into deduction systems and refutation systems. We look at the former here. You look at an example of the latter in your LAI course.

Deduction systems, cont'd

A **deduction system** enables us to derive conclusions from a set of wffs by *syntactic operations* alone. We manipulate the set of wffs without reference to their semantics. If the manipulation rules are 'right' (in a sense to be explored later), the new wffs we generate will in fact be logical consequences of the original set of wffs. The process is 'mechanical' and thus amenable to automation.

A deduction system comprises a finite set of **inference rules** and a set of wffs called **logical axioms**. The latter set is often infinite, but this is not a problem as one can use templates for these wffs, these templates being called **logical axiom schemata**. In fact, the deduction system we present below and the one you meet in the LAI course have no logical axioms, so we'll not discuss these any further.

Inference rules

An **inference rule** comprises a set of patterns called the **conditions**, and another pattern called the **conclusion**.

E.g.

$\Rightarrow$-ELIMINATION

$$\frac{W_1, W_1 \Rightarrow W_2}{W_2}$$

Above the line are the conditions, below is the conclusion. If you have wffs that match the conditions, then you can, in a single step, infer a wff that matches the conclusion.

Suppose we have wffs p and $p \Rightarrow q$, then using the inference rule we can infer q. We set this out as follows:

1.	p	premiss
2.	$p \Rightarrow q$	premiss
3.	q	from 1,2 by $\Rightarrow$-ELIMINATION

Now suppose we have the wff $q \Rightarrow r$ as well as p and $p \Rightarrow q$. Can we infer r?

(We 'ought' to be able to if this deduction system is any good, because r is certainly a *logical consequence* of the premisses, i.e. $\{p, p \Rightarrow q, q \Rightarrow r\} \models r$, as you can confirm for yourself with a truth table.)

But, using the inference rule, we cannot infer r in a single step. To remedy this, we could introduce another inference rule, e.g.

$$\frac{W_1, W_1 \Rightarrow W_2, W_2 \Rightarrow W_3}{W_3}$$

But, while some additional inference rules would be useful, we cannot keep introducing them in this way: we would need an infinite number of them.

Derivations

We allow *sequences* of inferences that might make use of the same inference rule repeatedly on the newly generated wffs:

1.	p	premiss
2.	$p \Rightarrow q$	premiss
3.	$q \Rightarrow r$	premiss
4.	q	from 1,2 by $\Rightarrow$-ELIMINATION
5.	r	from 3,4 by $\Rightarrow$-ELIMINATION

In applying an inference rule, we allow ourselves to match its conditions with premisses *or* with wffs inferred earlier. (In fact, in a system that allows logical axioms you can use these too, and, as we will see, 'assumptions' can also be made and matched with the conditions of inference rules.)

Such a sequence is called a **deduction** or a **derivation**.

If A is the set of premisses and W is the *final* conclusion, then we write

$$A \vdash W$$

We say that W is **deducible** or **derivable** from A.

$\vdash$ is called the 'syntactic turnstile' symbol, and is a symbol of the metalanguage.

There are many choices of deduction system. They offer different sets of logical axioms (including none) and different sets of inference rules.

We look at one example of what is called a **natural deduction** system.

The inference rules

The idea is to provide a rule of inference to 'introduce' and 'eliminate' each of the five connectives.

The $\wedge$-INTRODUCTION rule says that if wffs W_1 and W_2 are among the premisses or have been derived earlier, then you can infer wff $W_1 \wedge W_2$:

$\wedge$-INTRODUCTION

$$\frac{W_1, W_2}{W_1 \wedge W_2}$$

The $\wedge$-ELIMINATION rules say that if you have the wff $W_1 \wedge W_2$ among the premisses or you have derived them earlier, you may infer W_1 on its own and/or W_2 on its own:

$\wedge$-ELIMINATION-LEFT

$$\frac{W_1 \wedge W_2}{W_1}$$

$\wedge$-ELIMINATION-RIGHT

$$\frac{W_1 \wedge W_2}{W_2}$$

There are lots more rules yet, but to make matters clearer, we immediately show an example that uses the rules just given.

Derivation: example

E.g. show that $\{p \wedge q\} \vdash q \wedge p$

1.	$p \wedge q$	premiss
2.	p	1, $\wedge$-E-L
3.	q	1, $\wedge$-E-R
4.	$q \wedge p$	2,3, $\wedge$-I

I've abbreviated the explanations in the rightmost column.

E.g. show that $\{p \wedge (q \wedge r)\} \vdash (p \wedge q) \wedge r$

We'll do this example in the lecture.

1.
2.
3.
4.
5.
6.
7.

Finding a derivation is a process of trial and error: often, it will be possible to apply several rules, and you will have to make a choice. If that choice doesn't 'work out', you'll have to go back and try something else. Do not be discouraged by this process. It happens to *all* of us!

Inference rules, cont'd.

The next rule, $\Rightarrow$-ELIMINATION, is the rule we saw earlier. Some books call it **modus ponens**.

$\Rightarrow$-ELIMINATION

$$\frac{W_1, W_1 \Rightarrow W_2}{W_2}$$

Now we look at $\vee$-INTRODUCTION-RIGHT and $\vee$-INTRODUCTION-LEFT.

If W_1 is among the premisses or has been derived earlier, then you can infer $W_1 \vee W_2$. This may seem unnatural. As far as these course notes are concerned, all I will say by way of justification of this rule is that inference rules are syntactic operations, and we can have them do whatever we want. Of course, if derivation is to have any 'tie in' with logical consequence, then we will in fact need the 'right' rules. This is a topic we return to later. I will attempt to explain informally why the following rules are 'right' in the lecture.

$\vee$-INTRODUCTION-RIGHT

$$\frac{W_1}{W_1 \vee W_2}$$

$\vee$-INTRODUCTION-LEFT

$$\frac{W_2}{W_1 \vee W_2}$$

Inference rules, cont'd.

$\Leftrightarrow$-ELIMINATION-LEFT

$$\frac{W_1 \Leftrightarrow W_2, W_1}{W_2}$$

$\Leftrightarrow$-ELIMINATION-RIGHT

$$\frac{W_1 \Leftrightarrow W_2, W_2}{W_1}$$

$\neg$-ELIMINATION

$$\frac{\neg\neg W}{W}$$

Derivation: examples

Show that $\{p \wedge q\} \vdash p \vee q$

1.	$p \wedge q$	premiss
2.	p	1, $\wedge$-E-L
3.	$p \vee q$	2, $\vee$-I-R

Show that $\{p, q, (p \wedge q) \Rightarrow r\} \vdash r$

1.	p	premiss
2.	q	premiss
3.	$(p \wedge q) \Rightarrow r$	premiss
4.	$p \wedge q$	1,2, $\wedge$-I
5.	r	3,4, $\Rightarrow$-E

Subderivations

More complicated inference rules make use of **subderivations**. Roughly, the idea is that, within some derivation, we might have a subderivation of some wff, this subderivation making use of an **assumption** of some wffs, i.e. within the subderivation, new wffs are derived from the premisses, wffs that have been derived earlier and the assumption(s). When we 'come out of' the subderivation, we *discharge* the assumption: it cannot be used in any further steps. But, the more complicated inference rules that we are about to see state that when a subderivation of some wff has been found, when we 'come out of' the subderivation, we can infer some other wff (often one that mentions the assumption).

It is important not to confuse assumptions with premisses: once discharged, assumptions and other wffs derived *within* the subderivation cannot be used in later steps of the derivation.

The condition of the $\Rightarrow$-INTRODUCTION rule requires a subderivation of W_2 from assumption W_1. If such a subderivation can be found, then the rule allows us to discharge the assumption and infer the wff $W_1 \Rightarrow W_2$.

$\Rightarrow$-INTRODUCTION

$$\frac{\frac{W_1}{W_2}}{W_1 \Rightarrow W_2}$$

Example: we show that $\{p \Rightarrow q, q \Rightarrow r\} \vdash p \Rightarrow r$

1.	$p \Rightarrow q$		premiss
2.	$q \Rightarrow r$		premiss
3.	Subderivation:		
	3.1	p	assumption
	3.2	q	1, 3.1, $\Rightarrow$-E
	3.3	r	2, 3.2, $\Rightarrow$-E
4.	$p \Rightarrow r$		3, $\Rightarrow$-I

Subderivations: example

We'll do this one in the lecture.

Show that $\{p \Rightarrow q, p \Rightarrow r\} \vdash p \Rightarrow (q \wedge r)$

1.
2.
3.
    3.1
    3.2
    3.3
    3.4
4.

Inference rules, cont'd.

Here are the remaining rules.

$\vee$-ELIMINATION

$$\frac{W_1 \vee W_2, \frac{W_1}{W_3}, \frac{W_2}{W_3}}{W_3}$$

$\Leftrightarrow$-INTRODUCTION

$$\frac{\frac{W_1}{W_2}, \frac{W_2}{W_1}}{W_1 \Leftrightarrow W_2}$$

$\neg$-INTRODUCTION

$$\frac{\frac{W_1}{W_2 \wedge \neg W_2}}{\neg W_1}$$

Deploying the above three is all sufficiently hard that it is not something I expect you to be able to do. Where asked to give derivations, I guarantee you that only the simple rules plus $\Rightarrow$-INTRODUCTION and REPETITION (below) will ever be needed.

One final rule, which is not really essential, but can make derivations easier to read is:

REPETITION

$$\frac{W}{W}$$

This simply allows us to write a wff again later in a derivation.

Proofs

A proof is a derivation from no premisses. Obviously, they're almost bound to begin with some assumptions!

E.g. prove that $\vdash p \Rightarrow p$

1.	Subderivation:		
	1.1	p	assumption
	1.2	p	1.1, repetition
2.	$p \Rightarrow p$		1, $\Rightarrow$-I

Soundness and completeness: soundness

The deduction system derives wffs using syntactic operations alone, without reference to the semantics. We could invent a set of inference rules that license even quite bizarre inferences. However, if a deduction system is to be useful, the wffs we can derive from a set of wffs must tie in with logical consequences of that set.

At least two things may go wrong.

An inference rule might license an inference that is not a logical consequence. For example,

$\Rightarrow$-DUFF

$$\frac{W_1 \Rightarrow W_2, W_2}{W_1}$$

We can use this rule to show, e.g., $\{p \Rightarrow q, q\} \vdash p$. But, we know that $\{p \Rightarrow q, q\} \not\models p$. (We showed this earlier. If you can't find it, draw a truth table and confirm it again!)

An inference rule of this kind is said to be **unsound**. A deduction system that contains such a rule is **unsound**.

An inference rule is **sound** if the conclusions one can infer from any set of wffs using the rule are logical consequences of the set of wffs.

A deduction system is sound if it contains only sound inference rules.

Soundness and completeness: completeness

Another thing that may go wrong is that there may be logical consequences of a set of wffs that the deduction system fails to derive.

For example, we know that $\{p \vee q, \neg p\} \models q$. (Confirm it with a truth table if you don't believe it!). But if our deduction system has insufficient inference rules or insufficiently 'powerful' inference rules, it may be that we cannot show that $\{p \vee q, \neg p\} \vdash q$. In this case, the deduction system is said to be **incomplete**.

A deduction system is **complete** if every logical consequence of any set of wffs can also be derived from the set of wffs.

Soundness and completeness: summary

A deduction system is **sound** so long as for any set of wffs A and any wff W,

$$\text{if } A \vdash W \text{ then } A \models W$$

A deduction system is **complete** so long as for any set of wffs A and any wff W,

$$\text{if } A \models W \text{ then } A \vdash W$$

Soundness is essential, but completeness might be sacrificed in Computer Science uses of deduction systems, where the efficiency of the automated deduction system is important.

The soundness and completeness of a deduction system is obviously something that the logician who proposes the system should prove.

Soundness and completeness: summary

If a deduction system is sound and complete, then for any wff W,

$$W \text{ is valid if and only if } \vdash W$$

i.e. any valid wff will be provable and vice versa.

And, for any set of wffs A and wff W,

there will be no interpretation that satisfies all the wffs in A
if and only if $A \vdash W \wedge \neg W$

(When a wff of the form $W \wedge \neg W$ can be derived from a set of wffs A, we say that A is **inconsistent**.)

FIRST-ORDER PREDICATE LOGIC

In this section of the course, we look at another logic, one which can be viewed as providing an extension in expressiveness to propositional logic. We concentrate on its syntax and semantics, and formulating English sentences as formulae of this logic. Discussions of validity, satisfiability, logical consequence, logical equivalence and deduction systems are much as they were for propositional logic, and so are given only a summary treatment.

Limitations of propositional logic

Our very first example of a valid argument from the previous section of the course is not, in fact, valid in propositional logic.

1. Every elephant is grey.
2. Clyde is an elephant.
3. Therefore, Clyde is grey.

In propositional logic, e.g.,

$$\{p, q\} \not\models r$$

To move to a logic in which the argument is valid requires that we no longer treat English statements such as those above as atomic. Instead, we will decompose them further, thereby adding expressiveness to propositional logic. The result will be **predicate logic**.

In predicate logic, in addition to the connectives we saw in propositional logic, there will also be constant symbols to represent the entities or objects about which we intend to talk (the universe of discourse), function symbols to represent functions that map tuples of entities to other entities, and predicate symbols to represent relations that hold of tuples of objects. There will also be variables to represent unspecified objects from the universe of discourse, and quantifiers, which will determine the number of objects these variables range over.

The fact that variables ‘stand for’ unspecified objects, but not unspecified functions, relations, etc., is what makes this logic ‘first order’. Hence, we are studying **first-order predicate logic**.

An informal introduction: constant and predicate symbols

'Clyde is an elephant.'

'Clyde likes Martie.'

'Clyde likes Martie and Martie likes Clyde.'

These sentences describe properties of entities or relationships between entities.

We use **constant symbols** to denote entities. In this course, lowercase letters or names that begin with lowercase letters will be used, e.g.:

$$c, c_1, c_2, \ldots, \textit{clyde}, \ldots$$

We use **predicate symbols** to denote relations.

We use lowercase letters or names that begin with lowercase letters for predicate symbols, e.g.:

$$p, p_1, p_2, \ldots, \textit{elephant, likes}, \ldots$$

The set of predicate symbols and the set of constant symbols will be disjoint. But, predicate symbols can also be distinguished from constant symbols by their position in a wff: predicate symbols precede their arguments. The arity of the predicate symbol determines the number of arguments it must have.

So, e.g., if c is a constant symbol that denotes Clyde and m is a constant symbol that denotes Martie, *elephant* is a unary predicate symbol that denotes the unary relation whose graph/extension is the set of 1-tuples $\langle a \rangle$ where a is an elephant, and *likes* is the binary predicate symbol that denotes the binary relation whose graph/extension is the set of couples $\langle a_1, a_2 \rangle$ where a_1 likes a_2, then we can translate the sentences above as:

elephant(c)

likes(c, m)

likes(c, m) $\wedge$ *likes(m, c)*

Note how the connectives of propositional logic remain available to us.

An informal introduction: function symbols

'Clyde's father is an elephant.'

'Clyde and Martie's mutual friend likes Clyde and Martie.'

The expressions 'Clyde's father' and 'Clyde and Martie's mutual friend', like the simpler expressions 'Clyde' and 'Martie', refer to objects in the world. Instead of 'naming' these objects directly, we use an expression which denotes the application of a function to (named) objects.

We use **function symbols** to denote functions from tuples of entities to other entities. Functions cannot map a tuple of entities to more than one object. This is what makes the use of a function symbol suitable in translation of the phrase 'Clyde's father' (Clyde will have only one father). In this regard, the 'mutual friend' example is, perhaps, a poor one, as some people can have several mutual friends (though use of a function symbol here means that Clyde and Martie do not).

We use lowercase letters and words that begin with lowercase letters for function symbols, e.g.:

$$f, f_1, f_2, \ldots, \textit{fatherOf}, \textit{mf}, \ldots$$

The set of predicate symbols, the set of function symbols and the set of constant symbols are all disjoint from each other. But, function symbols can also be distinguished from the other types of symbols by their position in wffs: they appear in arguments to predicates, but they require arguments themselves. They too have an arity that specifies how many arguments they require.

An informal introduction: function symbols, cont'd.

Using the same symbols as were used on the previous slide, plus the symbol *fatherOf* to denote the function from one entity to that entity's father, and *mf* to denote the function that maps a pair of entities to their mutual friend, then we can translate the sentences from the previous slide as:

$$\textit{elephant(fatherOf(c))}$$

$$\textit{likes(mf(c, m), c)} \wedge \textit{likes(mf(c, m), m)}$$

Question: The following two wffs both say that c's father is an elephant. Explain the different ways in which they say this.

$$\textit{father(c, g)} \wedge \textit{elephant(g)}$$

$$\textit{elephant(fatherOf(c))}$$

An informal introduction: variables and quantification

'Everybody is happy!' (Hoorah!)

$$\textit{happy(c)} \wedge \textit{happy(m)} \wedge \textit{happy(f)} \wedge \textit{happy(g)} \wedge$$

$$\text{happy(fatherOf(g))} \wedge \text{happy(fatherOf(fatherOf(g)))} \wedge \ldots$$

Such a wff is unwieldy for a large universe of discourse, and impossible for an infinite one.

To overcome this, we need a way of saying that something is true for some number of objects without having to write a separate expression to denote each object. We achieve this using **variables**: the object referred to by a variable may vary over the given set of objects: think of it as cycling through the set of objects, denoting each one in turn.

We use lowercase letters from the end of the alphabet for variables, e.g.:

$$x, x_1, x_2, \ldots, y, y_1, y_2, \ldots, z, z_1, z_2, \ldots$$

Then, we use **quantifiers** to say how many of the objects the statement holds for.

$\forall$ is the symbol used for **universal quantification**: the statement holds for *all* objects in the universe of discourse.

$$\forall x \textit{happy}(x)$$

i.e. for all x, x is happy. Everybody is happy! Hoorah!

An informal introduction: variables and quantification, cont'd.

'Someone is wealthy.'

$$\textit{wealthy(c)} \vee \textit{wealthy(m)} \vee \textit{wealthy(f)} \vee \textit{wealthy(g)} \vee$$

$$\text{wealthy(fatherOf(g))} \vee \text{wealthy(fatherOf(fatherOf(g)))} \vee \ldots$$

$\exists$ is the symbol used for **existential quantification**: the statement holds for *at least one* object in the universe of discourse.

$$\exists x \textit{wealthy}(x)$$

i.e. there is at least one x such that x is wealthy.

$\forall$ and $\exists$ are the only two quantifiers used in this logic. With some ingenuity, we can use them to convey other forms of English quantification, such as 'at least two…', 'no…'. But there are plenty of forms of English quantification that they cannot capture, e.g. 'most…' 'about half…', etc. Other logics include expressions to capture these, but we'll stick to universal and existential quantification, which have proved to be of immense usefulness in their own right.

Now let's formalise all this.

The syntax of first-order predicate logic: terms

We explain how to construct well-formed formulae (wffs) of first-order predicate logic in two steps: first, we show how to construct **terms**; then we show how to construct **wffs**.

1. Every constant symbol is a term.

2. Every variable is a term.

3. For every function symbol F of arity n and any n terms $T_1, T_2, \ldots, T_n$ then

   $$F(T_1, T_2, \ldots, T_n)$$

   is a term.

4. Nothing else is a term.

(Terms will be the arguments to predicate symbols. They are, as you see above, also the arguments to function symbols.)

The syntax of first-order predicate logic: wffs

Now we give the rules for wffs.

1. For any predicate symbol P of arity n and any n terms $T_1, T_2, \ldots T_n$ then

   $$P(T_1, T_2, \ldots T_n)$$

   is a wff. (Such wffs are called **atomic wffs**, or simply **atoms**.)

2. For any variable X and any wff W

   $$\forall X W$$

   is a wff, and

   $$\exists X W$$

   is a wff. (Such wffs are called **quantified wffs**.)

The syntax of first-order predicate logic: wffs, cont'd.

3. For any wffs W, W_1 and W_2, the following are also wffs:

$$\neg W$$

$$\begin{array}{cc} W_1 \wedge W_2 & W_1 \vee W_2 \\ W_1 \Rightarrow W_2 & W_1 \Leftrightarrow W_2 \end{array}$$

4. Any wff may be enclosed in parentheses and will still be a wff.

5. No other string of symbols is a wff.

Notice how all function and predicate symbols are prefix in this language. Some books also allow *infix* binary function and predicate symbols to be used where it looks nicer doing so. We will not do this.

The syntax of first-order predicate logic: example wffs

In the lecture, we will verify that these are wffs.

$$\textit{likes(c, m)}$$

$$\textit{likes(c, fatherOf(m))} \Rightarrow \textit{elephant(c)}$$
$$\textit{likes(mf(fatherOf(m), fatherOf(c)), c)}$$
$$\forall x \; \textit{likes}(x, x)$$
$$\forall x(\textit{elephant}(x) \Rightarrow \neg\textit{likes}(x, m))$$
$$\forall x(\textit{elephant}(x) \Rightarrow \exists y(\textit{mouse}(y) \wedge \neg\textit{likes}(x, y)))$$
$$\textit{mouse}(x)$$
$$\forall x \; \textit{likes}(y, y)$$
$$\textit{elephant}(x) \wedge \exists x \textit{mouse}(x)$$
$$(\forall x \textit{elephant}(x)) \Rightarrow \exists y(\textit{mouse}(y) \wedge \neg\textit{likes}(x, y))$$
$$\forall y(\textit{elephant}(y) \Rightarrow \exists y(\textit{mouse}(y) \wedge \neg\textit{likes}(y, y)))$$

Precedence and associativity

- Parentheses determine evaluation order.
- Precedence rules are:

  Highest $\quad \forall \exists \neg \vee \wedge \Rightarrow \Leftrightarrow \quad$ Lowest

  (Be warned that some books use different precedences: there's little consistency on this issue.)
- Associativity rules are as before.

E.g. by the precedence rules,

$$\forall x \; \textit{elephant}(x) \Rightarrow \exists y \; \textit{mouse}(y)$$

evaluates as

$$(\forall x \; \textit{elephant}(x)) \Rightarrow (\exists y \; \textit{mouse}(y))$$

The other evaluation order is given by

$$\forall x (\textit{elephant}(x) \Rightarrow \exists y \; \textit{mouse}(y))$$

But we will use parentheses liberally.

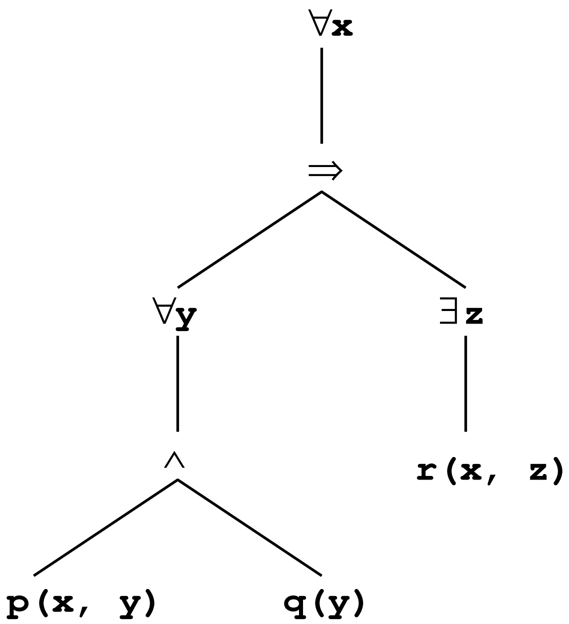
Scope of a quantifier

If a wff contains a subwff of the form $\forall XW$ or $\exists XW$, then W is called the **scope** of this particular quantifier. Obviously, precedence rules and/or parentheses are important in determining the scope.

Example

$$\underline{\forall x(\underline{\forall y(p(x,y) \wedge q(y))} \Rightarrow \underline{\exists z r(x,z)})}$$

Tree diagrams show scope clearly: the subtree that sits below a quantifier is its scope.



Free and bound variables

The occurrence of a variable adjacent to a quantifier is its **binding occurrence**.

All other occurrences are **applied occurrences**.

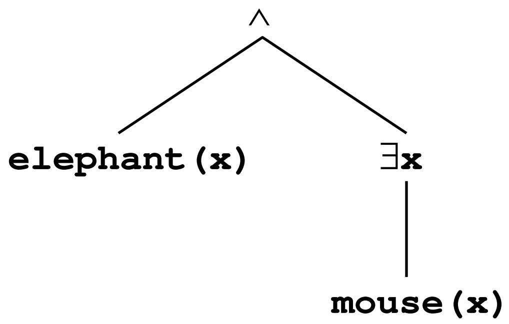
If a wff contains a subwff of the form $\forall X W$ or $\exists X W$, then applied occurrences of X in W are said to be **bound**. I.e. an applied occurrence of a variable is bound if it is in the scope of a quantifier that binds that variable.

Other applied occurrences of variables are said to be **free**.

Scope, and free and bound variables: example

We'll show scope, and free and bound variables during the lecture.

$$elephant(x) \wedge \exists x\, mouse(x)$$



A wff with no free variables is called a **sentence**. (The above example is *not* a sentence.)

Scope, and free and bound variables: examples

We'll show scopes, and free and bound variables during the lecture. We'll also say which of these wffs are sentences.

$$\textit{likes(c, m)}$$

$$\textit{likes(c, fatherOf(m))} \Rightarrow \textit{elephant(c)}$$

$$\textit{likes(mf(fatherOf(m), fatherOf(c)), c)}$$

$$\forall x \; \textit{likes}(x, x)$$

$$\forall x(\textit{elephant}(x) \Rightarrow \neg \textit{likes}(x, m))$$

$$\forall x(\textit{elephant}(x) \Rightarrow \exists y(\textit{mouse}(y) \wedge \neg \textit{likes}(x, y)))$$

$$\textit{mouse}(x)$$

$$\forall x \; \textit{likes}(y, y)$$

$$(\forall x \textit{elephant}(x)) \Rightarrow \exists y(\textit{mouse}(y) \wedge \neg \textit{likes}(x, y))$$

$$\forall y(\textit{elephant}(y) \Rightarrow \exists y(\textit{mouse}(y) \wedge \neg \textit{likes}(y, y)))$$

The semantics of first-order predicate logic

Since this logic is still concerned with the analysis of English *statements*, it is still the case that a wff will be either true or false.

In propositional logic, the meaning of the atomic wffs was *stipulated* by an interpretation. Then, the meaning of a *compound* wff was determined by the stipulated meanings of its atomic wffs.

In predicate logic, the meaning of the lowest level expressions (constant symbols, function symbols and predicate symbols) will be *stipulated* by an *interpretation*. Then, the meaning of a wff will be determined by the meanings of its subexpressions (in this case, its terms and subwffs). (In fact, strictly speaking, we cheat a bit in this course when defining the semantics of quantified wffs: our specification of the meaning of a quantified wff is not straightforwardly determined by the meaning of its subexpressions. But, our version of the semantics is hugely simplified by this cheat.)

We assume that there is some set of entities (possibly an infinite set) which we refer to as the **universe of discourse**, and which we denote as $\mathcal{U}$.

Entities can be concrete (e.g. the lake, my head) or abstract (e.g. the number 2, the set of all integers, the concept of justice) — any object of thought.

It should be clear that these entities are going to be the denotations of the constant symbols.

It is obviously possible to define a large number of relations *on* $\mathcal{U}$. We can define all 1-ary, 2-ary, 3-ary, etc. relations. Some of these will be functional. Relations can be thought of as specifying that a property or relationship holds for a tuple of objects; functions can be thought of as mapping tuples of objects to some other object.

Universe of discourse: example

The Books World

- Entities: Here we have five entities (excluding the table): Down&Out, Zen, Cuckoo, Jude and Spire.

- Functions: Here we have such functions as the unary function that maps from a book to the book directly on top of it.

- Relations: Here we have such relations as the binary relation that tells us whether a pair of books is such that the second is *somewhere* above the first, and the unary relation that tells us which books have a clear top.

Interpretations

In predicate logic, an interpretation $\mathcal{I}$ makes the following mappings:

1. For every constant symbol C, $\mathcal{I}(C)$ is an entity from $\mathcal{U}$. (Different constant symbols need not denote different entities, and there may be unnamed entities.)

2. For every n-ary function symbol F, $\mathcal{I}(F)$ is an n-ary function on $\mathcal{U}$, i.e. one that maps n-tuples to an entity. (Different function symbols need not denote different functions, and there may be unnamed functions.)

3. For every n-ary predicate symbol P, $\mathcal{I}(P)$ is an n-ary relation on $\mathcal{U}$, i.e. one that holds of n-tuples. (Different predicate symbols need not denote different relations, and there may be unnamed relations.)

Note that, since $\mathcal{I}$ is used to fix the denotations of symbols, their apparent English meaning (if any) is irrelevant. A constant symbol *kanga* could be made to denote the entity Tigger under one interpretation or the entity Piglet under another, for example. A predicate symbol *loves* could be made to denote the hates relation under one interpretation or the numerical less than ($<$) relation under another, for example. It's all a question of what $\mathcal{I}$ stipulates. (Of course, for our own convenience, the symbols will probably be in some way mnemonic for the intended interpretation.)

Interpretation: example

For constant symbols a, b, c, d, e, a unary function symbol f and the binary predicate symbols p and q, we give an interpretation $\mathcal{I}_1$:

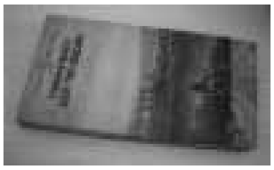
$\mathcal{I}_1(a)$ is 

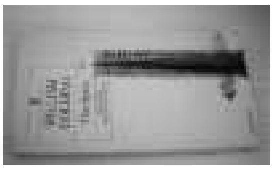
(Down&Out)

$\mathcal{I}_1(b)$ is 

(Zen)

$\mathcal{I}_1(c)$ is  (Cuckoo)

$\mathcal{I}_1(d)$ is  (Jude)

$\mathcal{I}_1(e)$ is  (Spire)

Interpretation: example, cont'd.

$\mathcal{I}_1(f)$ is	the unary function that maps a book to the book *directly on top* of it i.e. the function whose maplets are $\{$ZEN $\mapsto$ Down&Out, Cuckoo $\mapsto$ Zen, Spire $\mapsto$ Jude$\}$
$\mathcal{I}_1(p)$ is	the binary relation whose graph/extension contains pairs of books where the second book of the pair is *directly on top* of the first book of the pair i.e. its tuples are $\{\langle$Zen, Down&Out$\rangle, \langle$Cuckoo, Zen$\rangle, \langle$Spire, Jude$\rangle\}$
$\mathcal{I}_1(q)$ is	the binary relation whose graph/extension contains pairs of books where the second book of the pair is *somewhere above* the first book of the pair i.e. its tuples are $\{\langle$Zen, Down&Out$\rangle, \langle$Cuckoo, Zen$\rangle, \langle$Cuckoo, Down&Out$\rangle, \langle$Spire, Jude$\rangle\}$

Interpretation: example

Here is another example interpretation, $\mathcal{I}_2$, for the same universe of discourse and the same symbols. $\mathcal{I}_2$ is identical to $\mathcal{I}_1$ except:

$\mathcal{I}_2(a)$ is 

(Zen)

$\mathcal{I}_2(b)$ is 

(Down&Out)

$\mathcal{I}_2(p)$ is the binary relation whose graph/extension contains pairs of books where the second book of the pair is *directly below* the first book of the pair i.e. its tuples are $\{\langle$Down&Out, Zen$\rangle$, $\langle$Zen, Cuckoo$\rangle$, $\langle$Jude, Spire$\rangle\}$

$\mathcal{I}_2(q)$ is the binary relation whose graph/extension contains pairs of books where the second book of the pair is *directly on top* of the first book of the pair i.e. its tuples are $\{\langle$Zen, Down&Out$\rangle$, $\langle$Cuckoo, Zen$\rangle$, $\langle$Spire, Jude$\rangle\}$

Obviously, many other interpretations are possible for these symbols and this universe. And even more are possible if we consider different universes of discourse.

The semantics of first-order predicate logic, cont'd.

The interpretation stipulates the denotations of the constant, function and predicate symbols. These stipulations are then used in computing the semantics of terms and wffs.

The semantics of terms

- For any constant C, $\mathcal{I}(C)$ is stipulated.

- For any variable X, the semantics will be dealt with below, when we consider quantified wffs.

- For any n-ary function symbol F with arguments $T_1, T_2, \ldots, T_n$, then

  $\mathcal{I}(F(T_1, T_2, \ldots, T_n))$ is
  $$\mathcal{I}(F)(\mathcal{I}(T_1), \mathcal{I}(T_2), \ldots, \mathcal{I}(T_n))$$

  i.e. you get the function that corresponds to F (as stipulated by $\mathcal{I}$), you work out the objects that correspond to $T_1, T_2, \ldots, T_n$, and then you apply the function to these objects. This gets you an object, and this object is the meaning of the term $F(T_1, T_2, \ldots, T_n)$.

The semantics of first-order predicate logic, cont'd.

The semantics of atomic wffs

- For any atomic wff comprising an n-ary predicate symbol P with arguments $T_1, T_2, \ldots, T_n$ then

  $\mathcal{I}(P(T_1, T_2, \ldots, T_n))$ is true if and only if
  $\langle \mathcal{I}(T_1), \mathcal{I}(T_2), \ldots, \mathcal{I}(T_n) \rangle \in \mathcal{I}(P)$.
  Otherwise, it is false.

  I.e. you get the relation that corresponds to P (as stipulated by $\mathcal{I}$), you work out the objects that correspond to $T_1, T_2, \ldots, T_n$, and you see whether this n-tuple of objects is an element of the graph/extension of the relation.

The semantics of first-order predicate logic, cont'd

The semantics of quantified wffs

- For any quantified wff $\forall X W$, $\mathcal{I}(\forall X W)$ is true if and only if $\mathcal{I}(W)$ is true for all assignments of entities in $\mathcal{U}$ to X. Otherwise, it is false.

- For any quantified wff $\exists X W$, $\mathcal{I}(\exists X W)$ is true if and only if $\mathcal{I}(W)$ is true for at least one assignment of an entity in $\mathcal{U}$ to X. Otherwise, it is false.

The semantics of compound wffs

- For any compound wff $\neg W, W_1 \wedge W_2, W_1 \vee W_2$, $W_1 \Rightarrow W_2$ and $W_1 \Leftrightarrow W_2$, determine $\mathcal{I}(W), \mathcal{I}(W_1)$ and $\mathcal{I}(W_2)$ and use these truth values as before (as in propositional logic) to determine the truth value of the compound wff.

The sharpest amongst you might notice that this semantics is defined for *sentences* of first-order predicate logic. We have not defined a semantics for wffs that have free variables. Since wffs with free variables are of limited use, this seems acceptable (though remember I did say that the way I give a semantics to quantified wffs is a bit of a cheat).

Semantics: examples on the 'Books World'

We will determine the truth value of the following wffs under interpretation $\mathcal{I}_1$.

- $p(a, b)$
  - $\mathcal{I}_1(a)$ is Down&Out; $\mathcal{I}_1(b)$ is Zen.
  - Is $\langle$ Down&Out, Zen $\rangle \in \mathcal{I}_1(p)$?
  - No, so the wff is false.

- $p(a, b) \vee q(c, f(b))$
  - We know $\mathcal{I}_1(p(a, b))$ is false (from above).
  - $\mathcal{I}_1(c)$ is Cuckoo; $\mathcal{I}_1(b)$ is Zen; the function denoted by f maps Zen to Down&Out.
  - Is $\langle$ Cuckoo, Down&Out $\rangle \in \mathcal{I}_1(q)$?
  - Yes, and since this means that one of the disjuncts is true, the wff as a whole is true.

Semantics: more examples on the 'Books World'

- $\exists x \, p(c, x)$
  - We know that $\mathcal{I}(c)$ is Cuckoo.
  - Take $\mathcal{I}_1(x)$ to be Down&Out.
    - Is $\langle$ Cuckoo, Down&Out $\rangle \in \mathcal{I}_1(p)$?
    - No.
  - Take $\mathcal{I}_1(x)$ to be Zen.
    - Is $\langle$ Zen, Cuckoo $\rangle \in \mathcal{I}_1(p)$?
    - Yes.
  - We have found an assignment of an entity to x that makes $p(c, x)$ true. Existential quantification only requires at least one such assignment, so the wff as a whole is true.

Semantics: more examples on the 'Books World'

- $\forall x\, p(c, x)$
  - Take $\mathcal{I}_1(x)$ to be Down&Out.
    - Is $\langle$ Cuckoo, Down&Out $\rangle \in \mathcal{I}_1(p)$?
    - No.
  - Universal quantification requires that every assignment make $p(c, x)$ true. Since we have found an assignment that makes this false, there is no point in checking other assignments. The wff as a whole is false.

Semantics: another example on the 'Books World'

We'll sketch some of this one out in the lectures. We will use interpretation $\mathcal{I}_2$ and evaluate

$\mathcal{I}_2(\forall x \forall y(q(x, y) \Rightarrow p(y, x))$

From English statements to predicate logic

The English word ‘every’ generally signals use of universal quantification, and the English word ‘some’ generally signals use of existential quantification.

‘Every elephant is happy’

$$\forall x(\textit{elephant}(x) \Rightarrow \textit{happy}(x))$$

i.e. for every x, if x is a elephant, then x is happy.

‘Some elephant is happy’

$$\exists x(\textit{elephant}(x) \wedge \textit{happy}(x))$$

i.e. there exists an x such that x is a elephant and x is happy.

The question we want to explore is: why does the universal example use $\Rightarrow$ and the existential example use $\wedge$?

One way to explain this is to consider what happens when you use these connectives the other way round.

From English statements to predicate logic, cont'd.

Suppose you translated 'Every elephant is happy' as $\forall x($*elephant*$(x) \wedge$ *happy*$(x))$.

This translation is too strong. It says that everything in the universe of discourse is both an elephant and happy.

Consider the universe $\{$Clyde, Flopsy$\}$. Let $\mathcal{I}($*elephant*$)$ be the unary relation true of Clyde but not of Flopsy and let $\mathcal{I}($*happy*$)$ be the unary relation true of Clyde (and possibly of Flopsy: it doesn't matter).

Now, intuitively, you'd agree that in the world just described every elephant is happy: the English statement is true.

But $\forall x($*elephant*$(x) \wedge$ *happy*$(x))$ will be false in this world.

To show this:

- Take $\mathcal{I}(x)$ to be Clyde.
  - Since Clyde is an elephant and is happy, the conjunction is true.
- Take $\mathcal{I}(x)$ to be Flopsy.
  - Since Flopsy is not an elephant, the conjunction is false.

Since the conjunction is not true for every assignment to x, the wff as a whole is false (contrary to our intuitions).

On the other hand, the truth value for $\forall x($*elephant*$(x) \Rightarrow$ *happy*$(x))$ concurs with our intuitions:

- Take $\mathcal{I}(x)$ to be Clyde.
  - The antecedent and consequent are true (since Clyde is an elephant and is happy), so the conditional is true.
- Take $\mathcal{I}(x)$ to be Flopsy.
  - The antecedent is false, so the conditional is true.

Since the conditional is true for all assignments to x, the wff as a whole is true (in accord with our intuitions).

So, in general, universal quantification governs a conditional wff. (Of course, this is just a rule of thumb: there might be times when you *do* want to use a conjunction in a universally quantified wff.)

From English statements to predicate logic, cont'd

Suppose you translated 'Some elephant is happy' as $\exists x($*elephant*$(x) \Rightarrow$ *happy*$(x))$

This translation is too weak. If the antecedent is false, i.e. if there is so much as one object in the universe that is not an elephant, then the wff will be true.

Consider the universe $\{$Clyde, Flopsy$\}$. Let $\mathcal{I}$(*elephant*) be the unary relation true of Clyde but not of Flopsy, and let $\mathcal{I}$(*happy*) be the unary relation which might be true of Flopsy (it doesn't matter) but is *not* true of Clyde.

Now, intuitively, you'd agree that in the world just described there isn't a happy elephant: the English statement is false.

But $\exists x($*elephant*$(x) \Rightarrow$ *happy*$(x))$ will be true in this world.

To show this:

- Take $\mathcal{I}(x)$ to be Clyde.
  - Although the antecedent is true (Clyde is an elephant), the consequent is false (he isn't happy), so the conditional is false.
- Take $\mathcal{I}(x)$ to be Flopsy.
  - Since the antecedent is false (Flopsy is not an elephant), the conditional is true.

Since the conditional is true for at least one assignment to x, the wff as a whole is true (contrary to our intuitions).

On the other hand, the truth value for $\exists x($*elephant*$(x) \wedge$ *happy*$(x))$ concurs with our intuitions. (Check this for yourself.)

So, in general, existential quantification governs a conjunction. (Of course, this is just a rule of thumb: there might be times when you *do* want to use a conditional in an existentially quantified wff.)

Vacuous quantification

Universal quantification usually governs a conditional in translations of English sentences. But something counter-intuitive follows from these translations.

Suppose we have the following universe of discourse $\{$Clyde, Dumbo$\}$. Let the unary predicate *elephant* be the relation true of both Clyde and Dumbo, the unary predicate *mouse* be true of neither, the unary predicate *happy* be true of neither and the unary predicate *canFly* be true of Dumbo but not of Clyde.

Consider 'Every happy elephant can fly'. What are your intuitions? Is this statement true or false in this world?

Consider its translation into predicate logic:

$$\forall x((\textit{happy}(x) \wedge \textit{elephant}(x)) \Rightarrow \textit{canFly}(x))$$

Is this true or false?

- Take $\mathcal{I}(x)$ to be Clyde.
  - Since Clyde is an elephant but not happy, the antecedent is false, so the conditional is true.
- Take $\mathcal{I}(x)$ to be Dumbo.
  - Similarly, the conditional is true.

Since the conditional is true for all assignments of objects to x, the wff as a whole is true. Does this tally with your intuitions?

Or consider 'Every mouse can fly':

$$\forall x(\textit{mouse}(x) \Rightarrow \textit{canFly}(x))$$

By similar reasoning, this is true: all assignments to x, make the antecedent false (since none of the objects is a mouse) and hence the conditional true, so the wff as a whole is true.

Vacuous quantification, cont'd.

Our intuitions are that

- if there are no happy elephants, then the statement that every happy elephant can fly is false;

- if there are no mice, then the statement that every mouse can fly is false.

But our translations are wffs that are true in these cases.

More abstractly, it seems that on hearing a statement of the form 'Every P Qs' (e.g. 'Every mouse can fly'), people assume the non-emptiness of the set denoted by P, and are inclined to say that the statement is false if the set denoted by P turns out to be empty. However, translations of the form $\forall x(P(x) \Rightarrow Q(x))$ are true in these circumstances.

If you badly want to capture your intuitions, you need translations such as

$$\forall x((\textit{happy}(x) \wedge \textit{elephant}(x)) \Rightarrow \textit{canFly}(x)) \wedge \exists x(\textit{happy}(x) \wedge \textit{elephant}(x))$$

$$\forall x(\textit{mouse}(x) \Rightarrow \textit{canFly}(x)) \wedge \exists x \textit{mouse}(x)$$

Generally, we do not add these extra conjuncts, and we simply make do with the fact that the truth values of the wffs on the previous slide and our intuitive truth values may conflict.

In fact, sometimes a universally quantified statement would not lead you to assume the non-emptiness of the set denoted by P, in which case the translation and your intuitions would agree. E.g.:

> 'Every student who fails is offered a resit.'

Your intuitions on this one are more likely to be that this regulation is true even if no one fails.

From English statements to predicate logic, cont'd

We have said that the English word 'every' is generally translated by $\forall$. Similarly, the words 'all' and 'each' are also translated by $\forall$:

> 'All creatures are mortal.'
>
> 'Bill has attended every practical.'
>
> 'Each house has a roof.

But be careful, some uses of 'all' and, more rarely, some uses of 'every' do not translate as $\forall$:

> 'All the children play together'
>
> 'Every house has a different colour.'

These can't be straightforwardly translated at all! The problem is that the phrases 'play together' and 'has a different colour' say things about *the set* of children and houses as a whole. But, $\forall$ involves taking each object in turn.

From English statements to predicate logic, cont'd

We have said that 'some' is generally translated by $\exists$. Similarly, the words and phrases 'at least one', 'a', 'an', 'there is', 'there are' and, perhaps more controversially, 'a few', 'one or more' and 'a number of' are also translated by $\exists$:

> 'Some Computer Scientist is a poet.'
>
> 'Some Computer Scientists are poets.'
>
> 'There is a Computer Scientist who is a poet."
>
> 'There are Computer Scientists who are poets."
>
> 'There exists a Computer Scientist who is a poet.'
>
> 'There exist Computer Scientists who are poets.'

But, be careful because some uses of 'a' (or 'an') signal universal quantification:

> 'A country with no monarch is a republic.'
>
> 'A student who fails is offered a resit.'

From English statements to predicate logic, cont'd.

'Any' in unnegated sentences usually signals universal quantification:

> 'Anyone who cheats will be thrown out.'

'Any' in negated sentences usually gives a choice:

> 'Clyde didn't eat any buns.'

This can be paraphrased into 'pseudo-English' as 'for all x if x is a bun, then it is not the case that Clyde ate x' or as 'it is not the case that there exists an x that is a bun that Clyde ate'. The relationship between the two paraphrases and their corresponding wffs of predicate logic is further covered when we look at logical equivalences.

Consider plural nouns. Universal quantification is generally signaled.

> 'Students who fail will be excluded from the University.'

But, this is a bit controversial. Consider:

> 'Clyde attends practicals (so he'll be safe in the exam).'

Does Clyde attend every practical or just some?

Scope ambiguities

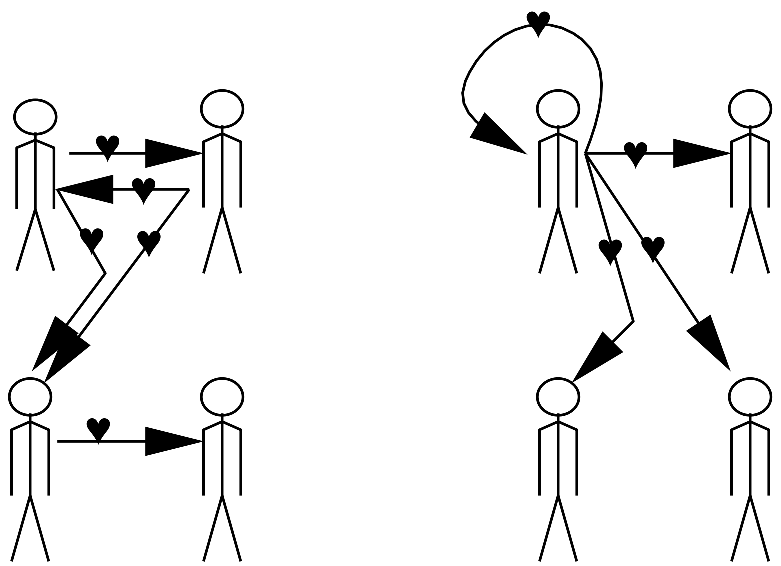
In general, English sentences that contain more than one quantifying expression of different kinds, one within the scope of the other, are liable to be ambiguous.

‘Everyone is loved by someone.’

$$\forall x \exists y \textit{ loves(y, x)}$$

$$\exists y \forall x \textit{ loves(y, x)}$$

The first translation is true in the left-hand situation; both are true in the right-hand situation:



This slide and the previous slides should convince you that translating quantified statements is hard! *You must always think about the meaning of the English very carefully.* Rules of thumb will lead you astray if applied without thought.

Validity and satisfiability

The definitions are unchanged:

- An interpretation $\mathcal{I}$ satisfies a wff W if the wff is true in that interpretation, i.e. $\mathcal{I}(W)$ is T.

- A wff is satisfiable if there is at least one interpretation that satisfies it. Otherwise (i.e. if no interpretation satisfies it), it is **unsatisfiable**.

- A wff is **valid** if all interpretations satisfy it. Otherwise (i.e. if at least one does not satisfy it), it is **invalid**.

Logical consequence and logical equivalence

The definitions are unchanged:

- A wff W is a **logical consequence** of a set of wffs A if and only if every interpretation that satisfies all the wffs in A also satisfies W.

- Two wffs W_1 and W_2 are **logically equivalent** (or simply **equivalent**) if and only if $\{W_1\} \models W_2$ and $\{W_2\} \models W_1$.

Example of a logical consequence:

$$\{\forall x(\textit{elephant}(x) \Rightarrow \textit{grey}(x)), \textit{elephant(clyde)}\} \models \textit{grey(clyde)}$$

Laws for quantifiers

We can define $\exists$ in terms of $\forall$ and vice versa.

Definition of existential

$$\exists XW \equiv \neg\forall X \neg W$$

Definition of universal

$$\forall XW \equiv \neg\exists X \neg W$$

Since one is definable in terms of the other, we strictly only need to have one of the two quantifiers in the language.

Substitution of variables

Given a wff W_1 containing a variable X_1, the wff W_2 which is just like W_1 except that X_2 has been substituted for some or all occurrences of X_1, then $W_1 \equiv W_2$ provided the substitution changes no bindings i.e. as long as what was free is still free and what was bound by a particular binding occurrence is still bound by the same binding occurrence.

Logical equivalences: example

'No duck likes Clyde.'

We suspect that this has two equivalent translations:

$$\neg\exists x(\textit{duck}(x) \wedge \textit{likes}(x,c)) \equiv \\ \quad \forall x(\textit{duck}(x) \Rightarrow \neg\textit{likes}(x,c))$$

We'll use the equivalences to show this algebraically.

$$\begin{aligned} \text{LHS} \quad & \equiv \neg\exists x(\textit{duck}(x) \wedge \textit{likes}(x,c)) \\ & \equiv \neg\neg\forall x\neg(\textit{duck}(x) \wedge \textit{likes}(x,c)) \\ & \equiv \forall x\neg(\textit{duck}(x) \wedge \textit{likes}(x,c)) \\ & \equiv \forall x(\neg\textit{duck}(x) \vee \neg\textit{likes}(x,c)) \\ & \equiv \forall x(\textit{duck}(x) \Rightarrow \neg\textit{likes}(x,c)) \\ & \equiv \text{RHS} \end{aligned}$$

Annotate each line with its justification.

Similar translations would be used for the example we mentioned earlier: 'Clyde didn't eat any buns'.

A problem

The problem with the definitions of satisfiability and particularly of unsatisfiability, validity, logical consequence and logical equivalence is that they require us to test *every* interpretation.

This was easily possible in propositional logic. For n different proposition symbols, there were $2 \uparrow n$ interpretations. For small n, constructing a truth table with $2 \uparrow n$ rows (one per interpretation) was easily manageable (though less so for large n).

For predicate logic, there is an infinite number of interpretations, some with infinite domains. There is nothing we can do in predicate logic that parallels the use of truth tables in propositional logic.

So, in predicate logic, the need for a deduction system, which works at a syntactic level, is now very strong.

Deduction system

The definition of inference rule, derivation, proof, soundness and completeness, etc. are all as before.

To extend the deduction system we gave for propositional logic, in addition to all the rules we used in propositional logic, we need introduction and elimination rules for the two quantifiers.

∃-INTRODUCTION

$$\frac{W(t)}{\exists x W(x)}$$

∃-ELIMINATION

$$\frac{\exists x W_1(x), \frac{W_1(a)}{W_2}}{W_2}$$

(so long as a does not occur in $W_1(x)$ or in W_2 or in a premiss, or in any assumption that is current other than $W_1(a)$)

Deduction system, cont'd.

$\forall$-INTRODUCTION

$$\frac{W(a)}{\forall x W(x)}$$

(so long as a does not occur in $W(x)$, in a premiss or in any assumption that is current)

$\forall$-ELIMINATION

$$\frac{\forall x W(x)}{W(t)}$$

We will not explain these rules: they are harder to apply than those we saw in propositional logic (although, remember, they still do nothing more than say how to perform structural manipulations). You will not be asked to use them.