Introduction

COMP2600 / COMP6260

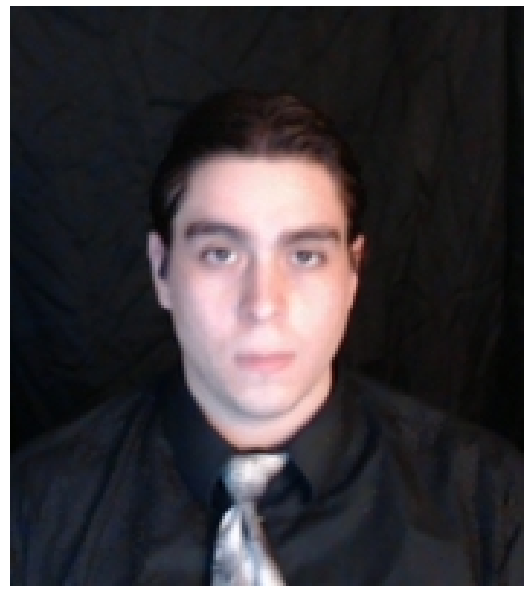
Dirk Pattinson
Australian National University

Semester 2, 2015

People

Rajeev Gorè
(convenor)

Dirk Pattinson



Daniel Alarcon

What do we mean by FORMAL?

Oxford Dictionary

- in accordance with convention or etiquette
- denoting a style of writing or public speaking characterized by more elaborate grammatical structures and more conservative and technical vocabulary
- officially sanctioned or recognized
- concerned with outward form or appearance as distinct from content

The validity of logical arguments depends on form, rather than content.

1. Abstracting content allows us to study the mechanics of reasoning
2. We have more confidence in formalised arguments
3. Automation requires formalisation.

Example: Vote Counting in the ACT

```
/* STEP 19 */
/* We haven't incremented count yet, so this applies to NEXT count */
mark_elected(cand, (void *)(count+1));
/* STEP 20 */
vote_value_of_surplus = cand->c[count].total - quota;
/* STEP 21 */
increment_count();
/* STEP 22 */
pile = cand->c[cand->count_when_quota_reached].pile;
non_exhausted_ballots
= (number_of_ballots(pile)
   - for_each_ballot(pile, &is_exhausted, candidates));
/* STEP 23 */
if (non_exhausted_ballots == 0)
new_vote_value = fraction_one;
else
new_vote_value = ((struct fraction) { vote_value_of_surplus,
                                                  non_exhausted_ballots });
/* STEP 24 */
update_vote_values(pile, new_vote_value);
/* STEP 24b */
/* Report actual value (may be capped) */
report_transfer(count,
pile->ballot->vote_value,
vote_value_of_surplus);
distribute_ballots(pile, candidates,vacating);
```

(Part of the EVACS code used for vote counting in the ACT in 2012)

Do you trust the code?

Topics in COMP 2600

- Logic and Natural Deduction –mechanics of constructing arguments
- Proving Properties of Functional Programs – applied to Haskell
- Automata, Languages and Parsing – what is a program?
- Turing Machines and Computability – what can be computed?
- Proving Properties of Imperative Programs – C-like languages

Assessment

Assignments: 36%

- One for each major topic, due in weeks 6, 9 and 12

Tutorials: 4%

- Demonstrate a reasonable attempt at the tutorial questions

Mid-Semester Quiz: 10%

- Redeemable: replaced by exam mark if better, covers weeks 2 – 6

Final Exam: 50%

- Or 60% if quiz not attempted or quiz score $<$ exam score

Final Mark

- Capped at 10 percentage points above exam percentage

Policies

Academic honesty

All assignment submissions and exam answers must be your own work. Detailed policy in Sections 6.4 and 6.5 of the RSCS student handbook (`http://cs.anu.edu.au/files/2014StudentHandbookFinal210314.pdf`).

Timely assignment submissions

For assignments submitted after the due date and time, 5% will be deducted from the assignment mark for each business day (or part thereof) that the assignment is late. No exceptions unless approved by the course convenor (justified by e.g. medical certificate).

Check your marks

It is your responsibility to check that the marks you receive for tutorial participation, and assignment submissions, are up to date and accurate in streams.

Admin stuff

- The course convenor is Prof. Rajeev Goré
  - Contact Rajeev.Gore@anu.edu.au for any queries related to the course overall, special consideration, etc
- Register for a tutorial in Streams. Tutorials start in week 3.
- Check out the COMP2600 website
  `http://cs.anu.edu.au/Student/comp2600/`
  - Textbooks
  - Tentative Schedule
  - Forum

Why study logic?

Hardware: Binary logic is logical.

Software: Programming languages have logical constructs.

Specification and semantics: The language of logic is unambiguous and can be used to give meaning to programs.

Proof: Arguments should be logical.

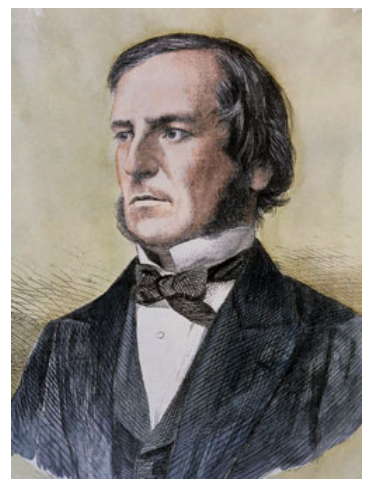
Every day: Clearer thinking in every day situations. More effective communication.

History of Logic, the Science of Reasoning

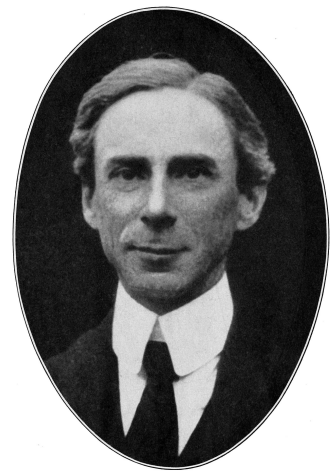
- Aristotle: syllogistic logic
- Leibniz: symbolic logic
- Boole: algebraic logic (boolean algebras)
- Russel: logic as foundation of mathematics
- Later: Church, Turing, Curry, Goedel, Scott, Milner etc.: Formal models of computational systems

Aristotle
(384–322BC)

Leibniz
(1646– 1716)



Boole
(1815 – 1884)



Russel
(1872 – 1970)

Logic = Syntax, Semantics and Calculus

Syntax: the language of logic

How do we write down logical statements?

(in calculus: $\int$, lim, d/dx, etc.)

Semantics: the meaning of the symbols

What does it mean for a statement to be valid?

(in calculus: $\int$ is the area ...)

Calculus: how do we construct an argument?

How can we prove that a statement is valid?

(in calculus: e.g. the chain rule)

The syntax of propositional logic: propositions

Definition

A *statement* is a sentence for which it makes sense to ask whether it is true or false. A *proposition* is a statement that does not depend on any variables.

Example

- John had toast for breakfast.
- Mary is enrolled in COMP2600.
- $2 + 2 = 5$.
- Every even integer > 2 is the sum of two primes (Goldbach's conjecture)

We often use variables p; q; r to denote propositions ('atomic propositions' or 'propositional variables').

The syntax of propositional logic: formulae

Definition

Formulae are built from a set of atomic propositions using the logical connectives $\neg$, $\vee$, $\wedge$, $\rightarrow$:

$$A\ B\ ::=\ p \mid \neg A \mid A \vee B \mid A \wedge B \mid A \rightarrow B \mid A \leftrightarrow B$$

Example

Given the atomic propositions p, q, r, 'John had toast for breakfast', 'John is hungry', 'John had tea for breakfast', the following are formulae:

- 'John had tea for breakfast' $\rightarrow$ 'John is hungry'.
- 'John had toast for breakfast' $\rightarrow \neg$ 'John is hungry'.
- $p \vee (q \wedge r)$.

The syntax of propositional logic: operator precedence

To avoid ambiguities and minimise parentheses, the logical operators can be assigned a **precedence**:

- $\neg$ (highest)
- $\wedge$
- $\vee$
- $\rightarrow$
- $\leftrightarrow$ (lowest)

Example

- $\neg p \vee q \rightarrow r \wedge s \quad \equiv \quad ((\neg p) \vee q) \rightarrow (r \wedge s)$
- $p \vee q \wedge r \quad \equiv \quad p \vee (q \wedge r)$

The semantics of propositional logic: in words

The semantics of propositional logic: in words

- $p \wedge q$: "*p and q*" : true iff both p and q are true

The semantics of propositional logic: in words

- $p \wedge q$: "*p and q*" : true iff both p and q are true
- $p \vee q$: "*p or q*" : true iff either p or q (or both) is true

The semantics of propositional logic: in words

- $p \wedge q$: "*p and q*" : true iff both p and q are true
- $p \vee q$: "*p or q*" : true iff either p or q (or both) is true
- $p \rightarrow q$: "*p implies q*", or "*if p then q*" :
  true iff either p is false or q is true (or both of these)

The semantics of propositional logic: in words

- $p \wedge q$: "*p and q*" : true iff both p and q are true
- $p \vee q$: "*p or q*" : true iff either p or q (or both) is true
- $p \rightarrow q$: "*p implies q*", or "*if p then q*" :
  true iff either p is false or q is true (or both of these)
- $\neg p$: "*not p*" : true iff p is false

The semantics of propositional logic: in words

- $p \wedge q$: "*p and q*" : true iff both p and q are true
- $p \vee q$: "*p or q*" : true iff either p or q (or both) is true
- $p \rightarrow q$: "*p implies q*", or "*if p then q*" :
  true iff either p is false or q is true (or both of these)
- $\neg p$: "*not p*" : true iff p is false
- $p \leftrightarrow q$: "*p is equivalent to q*", true when p and q are either both false or both true (actually it is just $p = q$, as boolean values)

The semantics of propositional logic: truth tables

We can express the logical operators using truth tables

p	q	$p \vee q$	$p \wedge q$	$p \rightarrow q$	$\neg p$	$p \leftrightarrow q$

The semantics of propositional logic: truth tables

We can express the logical operators using truth tables

p	q	$p \vee q$	$p \wedge q$	$p \rightarrow q$	$\neg p$	$p \leftrightarrow q$
T	T	T	T	T	F	T

The semantics of propositional logic: truth tables

We can express the logical operators using truth tables

p	q	$p \vee q$	$p \wedge q$	$p \rightarrow q$	$\neg p$	$p \leftrightarrow q$
T	T	T	T	T	F	T
T	F	T	F	F	F	F

The semantics of propositional logic: truth tables

We can express the logical operators using truth tables

p	q	$p \vee q$	$p \wedge q$	$p \rightarrow q$	$\neg p$	$p \leftrightarrow q$
T	T	T	T	T	F	T
T	F	T	F	F	F	F
F	T	T	F	T	T	F

The semantics of propositional logic: truth tables

We can express the logical operators using truth tables

p	q	$p \vee q$	$p \wedge q$	$p \rightarrow q$	$\neg p$	$p \leftrightarrow q$
T	T	T	T	T	F	T
T	F	T	F	F	F	F
F	T	T	F	T	T	F
F	F	F	F	T	T	T

The semantics of propositional logic: truth tables

We can express the logical operators using truth tables

p	q	$p \vee q$	$p \wedge q$	$p \rightarrow q$	$\neg p$	$p \leftrightarrow q$
T	T	T	T	T	F	T
T	F	T	F	F	F	F
F	T	T	F	T	T	F
F	F	F	F	T	T	T

We can use these tables to calculate the truth values of logical expressions

The semantics of propositional logic: validity

Definition

A propositional formula is said to be *valid* if it is true for all possible assignments of truth values to its variables.

Example

$p \vee \neg p$ is valid:

p	$\neg p$	$p \vee \neg p$
T	F	T
F	T	T

In general, to show that a proposition is valid, one can construct a truth table for the proposition and check that its column is all T's.
Note: a valid formula is also known as a *tautology*.

Contradictions and Contingencies

Definition

A contradiction is a compound proposition which evaluates to **F** for all values of its elementary propositions.
A contingency is a compound proposition which may evaluate to **T** or **F** for different values of its elementary propositions.

Example

- 'John had toast for breakfast' is a contingency.
- 'John had toast for breakfast' $\wedge\neg$ 'John had toast for breakfast' is a contradiction.
- $p \rightarrow (\neg q \vee p) \rightarrow (p \wedge q) \vee r$ – can be complicated

Laws of Propositional Logic

Associative laws

- $p \vee (q \vee r) = (p \vee q) \vee r$
- $p \wedge (q \wedge r) = (p \wedge q) \wedge r$

Distributive laws

- $p \wedge (q \vee r) = (p \wedge q) \vee (p \wedge r)$
- $p \vee (q \wedge r) = (p \vee q) \wedge (p \vee r)$

Reading

- the truth value on the LHS is the same as the truth value on the RHS
- cf algebra: $x + y = y + x$: same result for all values of x and y

(These are just a few. Replacing $=$ by $\leftrightarrow$ they are all validities.)

Inference rules

Logic = Syntax, Semantics and (proof) calculus
We have covered the Syntax and Semantics of propositional logic.
How do we construct an argument? One way is to use inference rules.

Definition

An *inference rule* is a blueprint for a valid argument. Its propositions are variables, and applying an inference rule amounts to providing formulae for the variables.

Modus Ponens

Definition

The following is the *Modus Ponens* inference rule:

$$\frac{p \rightarrow q \qquad p}{q}$$

Example

$$\frac{\text{'The student worked hard'} \rightarrow \text{'The student passed'} \qquad \text{'The student worked hard'}}{\text{'The student passed'}}$$

Disjunctive syllogism

Definition

The following is Aristotle's *Disjunctive Syllogism* inference rule:

$$\frac{p \vee q \qquad \neg p}{q}$$

Example

$$\frac{\text{'The student was happy'} \vee \text{'The student was awake'} \qquad \neg\text{'The student was happy'}}{\text{'The student was awake'}}$$

More on inference rules later in the course.

Limitations of propositional logic

Is this argument valid? useful?

Natural language	Propositional logic
All COMP2600 students are happy.	p
Lisa is a COMP2600 student.	q
Therefore, Lisa is happy.	$\therefore r$

Maybe - but not in propositional logic!

Not a valid argument form in terms of propositional logic, since $p \wedge q \rightarrow r$ is not a tautology. (There is no relationship between the propositions.)

Problem

The identity of Lisa is not maintained through the propositions. In other words, the propositions don't take arguments.

First-Order logic to the rescue

First-Order logic uses predicates as the basic building blocks of formulae.

What are the predicates in our example?

$$\begin{aligned} P(x) &\equiv x \text{ is a COMP2600 student.} \\ Q(x) &\equiv x \text{ is happy.} \end{aligned}$$

Logical form of the argument

$$\frac{\forall x.\ P(x) \rightarrow Q(x) \qquad P(\text{Lisa})}{Q(\text{Lisa})}$$

. . . which we might be able to deal with. . .

Syntax of First-Order logic: Predicates

Definition

A statement that depends on (zero, one or more) variables is called a *predicate*. They are usually written as expressions with variables, e.g. $x > 5$.

How to think about predicates

- As a mapping from some domain to a Boolean value, such as $P : \mathbb{N} \to \text{Bool}$ where $P(x) \equiv x > 5$
- The domain of a predicate is some set of appropriate values for the variable(x) in the predicate expression. In the above example the domain of P is $\mathbb{N}$.
- If a predicate contains more than one variable, then the elements of the domain are tuples. That is $x + y = 5$ is the function $P : \mathbb{N} \times \mathbb{N} \to \text{Bool}$ where $P(x, y) \equiv (x + y = 5)$.
- Instantiating the variables in a predicate with values yields a proposition, e.g. $P(12, 17) \equiv 12 + 17 = 5$.

Syntax of First-Order logic: Formulae

Definition (Syntax)

Formulae of First-Order logic are built from atomic predicates, propositional connectives and the quantifiers $\forall$ and $\exists$:

$$A\ B\ ::=\ P \mid \neg A \mid A \vee B \mid A \wedge B \mid A \rightarrow B \mid A \leftrightarrow B \mid \forall x.P(x) \mid \exists x.P(x)$$

Example

- $P \rightarrow \forall x.Q(x)$
  - P is a predicate with zero variables, i.e. a proposition
- $\forall x.$'x is enrolled in COMP2600'
- $\exists y.\forall x.Q(y) \rightarrow P(x)$

Semantics of quantifiers

Definition (Semantics of quantifiers)

If $P(x)$ is a predicate that depends on a variable x, then

$$\forall x.P(x)$$

means that $P(x)$ is true for *all* values of x (chosen from a domain of discourse), and

$$\exists x.P(x)$$

means that $P(x)$ is true for *some* choice of x (from a suitable domain of discourse).

- $\forall$ and $\exists$ are the quantifiers
- x is the quantified (or bound) variable

Abbreviations and Simple Laws of First-Order Logic

Abbreviations

$$\forall x, y.P(x, y) \equiv \forall x.\forall y.P(x, y) \equiv \forall x.(\forall y.P(x, y))$$

Quantifiers *of the same type* commute

$$\exists x.\exists y.P(x, y) \equiv \exists y.\exists x.P(x, y)$$

$$\forall x.\forall y.P(x, y) \equiv \forall y.\forall x.P(x, y)$$

Quantifiers *of different types* don't commute

- $C(x, y)$ – car x has colour y
- $\forall x.\exists y.C(x, y)$ – every car has a colour
- $\exists y.\forall x.C(x, y)$ – there is a colour such that every car has that colour, i.e., every car has the same colour

Translating from propositional logic to English

Example

Use the predicates I – I'm going surfing, Y – you're going surfing, and W – there'll be a big wave that kills us all, to translate the following statements to English:

1. $I \wedge Y \rightarrow W$
2. $(I \rightarrow W) \vee (Y \rightarrow W)$

Translating from English to propositional logic

Example

Use the predicates R – it's raining, I – I'm going surfing and Y – you're going surfing, to translate the following sentences to propositional logic.

1. I'm going surfing but it rains.
2. I'm going surfing unless it rains. (How about 'I'm going surfing if it doesn't rain'?)
3. I'm going surfing only if you're going surfing.

Translating from FOL to English

Example

Use the implied meaning of the predicates to translate the following predicate logic formulae to English.

1. $\forall x.(\mathit{likes}(\mathit{Alice}, x) \rightarrow \mathit{likes}(\mathit{Bob}, x))$
2. $\exists x.(\mathit{likes}(\mathit{Alice}, x) \wedge \mathit{likes}(\mathit{Bob}, x))$

Translating from English to FOL

Example

Use the predicates $D(x)$ – x is a dragon, $C(x, y)$ – x is a child of y, $H(x)$ – x is happy and $F(x)$ – x can fly, to translate the following English sentences to predicate logic.

1. All dragons can fly unless they are unhappy.
2. At least one dragon can fly despite being unhappy.
3. A dragon is happy if all of its children can fly.

Common patterns in translations

- 'unless' means 'or' and 'but' means 'and' (most of the time)
- A if B can be understood as 'if B then A and translates to $B \to A$.
- A only if B can be understood as 'if B doesn't happen, then A doesn't happen', i.e. 'if not B, then not A' which translates to $\neg B \to \neg A$ which is equivalent to $A \to B$.
- $\forall$ can often be expressed using 'everything' and $\exists$ by 'something'
- the effect of $\forall x.(P(x) \to \dots)$ is to restrict the quantifier to all those x that are P
- the effect of $\exists x.(P(x) \wedge \dots)$ is to say that there is a P