

Introduction and A Review of Elementary Logic

COMP2600 — Formal Methods for Software Engineering

Jeremy Dawson

Australian National University
Semester 2, 2011

Topics

- Logic and Natural Deduction
- Proving Properties of Functional Programs (Induction)
- Proving Properties of Imperative Programs
- Formal Specification of Systems
- Automata, Languages and Parsing
- Logic Programming
- Turing Machines, Lambda Calculus, Computability, ...

What do we mean by **FORMAL**?

OED: Concerned with the **form**, rather than the **matter**, of reasoning.

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

1. Abstracting away the content allows us to *study reasoning as reasoning*, without the content getting in the way;
2. If we can express an argument formally, we (may) have *more confidence it is correct*;
3. If we want to *automate* something, first we must formalise it.

Assessment

- **Assignments:** 4 × 9% each
- **Participation:** 4%
- **Mid-Semester Quiz:** 10% (redeemable)
- **Final Exam:** 50%
(or 60% if quiz not attempted or lower percentage mark for quiz)
- **Final mark capped at** Exams*(100/60)+10%

The mid-semester quiz, of one hour, is tentatively planned for week 8. If so, the topics in scope will be those covered in lectures in weeks 2 to 6 (tutorials weeks 3 to 7).

Things to Do:

- 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
- Any Questions? . . . Ask me later.

History of Logic, the Science of Reasoning

- Aristotle (384–322BC), the “father of logic”: syllogistic logic.
- Universal calculus of Leibniz (1646–1716): symbolic logic (and binary numerals).
- More symbolic logic by George Boole (1815–64): Boolean algebra.
- *Principia Mathematica* by Bertrand Russell: mathematical logic.
- Church, Turing, Curry, Goedel, Scott, Milner etc.: Formal frameworks for presenting models of systems and for reasoning about them.

Role of Haskell in 2600

- Until a few years ago, functional programming was introduced in 2600. COMP1100 plays that role.
- Haskell is used in 2600: for study of inductive types, definitions, and proofs.
- More on assumed knowledge in lecture 3.
- If needed see 1100 web pages.

Why study logic?

Coolness: Logic *is* cool — trust me.

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.

Propositions

- A *proposition* is a logical sentence.
- A proposition is either *true* or *false*, but not both.
- **T, F: Bool.**
- Examples:
 - I am the smartest person in the room.
 - John had toast for breakfast.
 - $2 + 2 = 5$.
- Questions, commands and expletives are not propositions.

Tautologies

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

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

In general, to prove a tautology, one can construct a truth table for the proposition and checks that its column is all **T**'s.

Definition by Truth table

Conjunction is defined to be the function specified by this table.

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

Disjunction, negation, implication are similarly defined.

Contradictions and Contingencies

Contradiction: A contradiction is a compound proposition which evaluates to **F** for all values of its elementary propositions.

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

An Important Syntactic Matter

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

- \neg (highest)
- \wedge
- \vee
- \Rightarrow
- \Leftrightarrow (lowest)

Examples:

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

Formalising Arguments

*That is, identifying the **form** of the argument. . .*

- Consider the line of reasoning.
- Identify the propositions.
- Identify the connectives.
- Translate to symbols.

(Algebraic) Laws of Propositional Calculus

Associative laws

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

Distributive laws

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

These are just a representative few.

Argument Forms

An argument may be written as:

$$\begin{array}{c} p_1 \\ \vdots \\ p_n \\ \hline q \end{array}$$

or (equivalently) as:

$$(p_1 \wedge \dots \wedge p_n) \Rightarrow q$$

Typical Application

An *argument* expressed in English:

- If the professor is naked *then* the class is amused.
- If the class is amused *and* the material is organized *then* the class is happy.
- The lecture is good *if both* the material is organized *and* the class is happy.
- Therefore, the lecture is good *if* the professor is naked.

Disjunctive Syllogism — An inference rule

One of Aristotle's patterns of valid deduction:

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

Typical Application (ctd)

Construct the propositional logic *argument form*.

Give a single propositional formula which, if it is a tautology, signifies the argument form is valid.

Is the argument form valid?

Disjunctive Syllogism — An example

The student was happy \vee The student was awake

\neg The student was happy

The student was awake

Disjunctive Syllogism — Proof in the Algebraic Style

$(p \vee q) \wedge \neg p \Rightarrow q$	
$\equiv \neg p \wedge (p \vee q) \Rightarrow q$	(commutativity)
$\equiv (\neg p \wedge p) \vee (\neg p \wedge q) \Rightarrow q$	(distribution)
$\equiv \mathbf{F} \vee (\neg p \wedge q) \Rightarrow q$	(contradiction)
$\equiv \neg p \wedge q \Rightarrow q$	(or-simplification)
$\equiv \neg(\neg p \wedge q) \vee q$	(implication)
$\equiv (p \vee \neg q) \vee q$	(De Morgan)
$\equiv p \vee (\neg q \vee q)$	(associativity)
$\equiv p \vee \mathbf{T}$	(excluded middle)
$\equiv \mathbf{T}$	(or-simplification)

Modus ponens — Inference rule

$$\frac{p \Rightarrow q \quad p}{q}$$

Modus Ponens is important!

(so important that you must know the Latin name)

NOT a Disjunctive Syllogism — Example

The student was happy \vee The student was awake
The student was happy
—————
\neg The student was awake

This “reasoning” is INVALID.

Modus Ponens — Example

The student worked hard \Rightarrow The student passed
The student worked hard
—————
The student passed

NOT Modus Ponens — Example

The student worked hard \Rightarrow The student passed
The student passed
The student worked hard

This “reasoning” is **INVALID**.

Lisa studies predicate calculus!

- What are the predicates?

$p(x) \equiv x$ is a COMP2600 student.

$q(x) \equiv x$ is happy.

- Now the argument looks like:

$\forall x. p(x) \Rightarrow q(x)$

$p(\text{Lisa})$

$\therefore q(\text{Lisa})$

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

Limitations of propositional logic

Consider this “argument”:

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

Useful? No! This is 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.)

Predicates — Introduction

- Predicates are conventionally written as expressions containing unbound variables, such as $x > 5$.
- A predicate is a mapping from some domain to a Boolean value, such as $p : \mathbb{N} \rightarrow \mathbf{Bool}$ 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 $p : \mathbb{N} \times \mathbb{N} \rightarrow \mathbf{Bool}$ where $p(x, y) \equiv (x + y = 5)$.
- Instantiating the variables in a predicate with values yields a proposition.

Universal quantification — Definitions

- General form: $\forall x \in D. P(x)$:
- $\forall x \in D. P(x)$: means that for *any* x chosen from D , $P(x)$ is a true proposition.
- In $\forall x \in D. P(x)$:
 - \forall is the *quantifier*;
 - x is the *bound variable*;
 - D is the *domain*;
 - $P(x)$ is a predicate (possibly) involving the bound variable x .

Existential quantification — Introduction

The existential quantifier is denoted by \exists , and is pronounced *for some* or *there exists*.

Examples:

- “Some students are happy”.
- “There is a happy student”.
- $\exists s \in \text{Student}. s$ is happy.
- $\exists x. (\exists y. x + y = y - x)$.

Universal quantification — Abbreviations and rules

- The domain can be omitted if it can be inferred from the context:
 $\forall x. P(x)$.
- $\forall x, y \in D. P(x, y)$ is an abbreviation for $\forall x \in D. (\forall y \in D. P(x, y))$
- Unnecessary parenthesis may be omitted: $\forall x \in D. \forall y \in D. P(x, y)$.
- $(\forall x. \forall y. P(x, y)) \equiv (\forall y. \forall x. P(x, y))$.
For example, $(\forall x. \forall y. x + y = y + x) \equiv (\forall y. \forall x. x + y = y + x)$.

Existential quantification — Definitions

General form: $\exists x \in D. P(x)$:

$\exists x \in D. P(x)$: means that for *one or more* x chosen from D , $P(x)$ is a true proposition.

In $\exists x \in D. P(x)$:

- \exists is the *quantifier*;
- x is the *bound variable*;
- D is the *domain*;
- $P(x)$ is a predicate (possibly) involving the bound variable x .

References

- *Discrete Mathematics with Applications*, 3rd Edition, Susanna Epp
(the principal textbook for MATH1005 for years)
- *The Logic Book*, Merrie Bergmann
(text for PHIL logic course)
- Many others . . .