CS2002 Logic Lecture 2

Introduction to Logic

Ian Gent

This Time …

- Why cover logic in a course called “Computer Systems”?
- Some key logical connectives
- Truth Tables
- Modus Ponens and encoding English into Logic
  - “A puff of logic”

Part 1: Why Cover Logics In this Course?

Why cover logics here?

- There is a relationship between ***symbolic logic*** (here Boolean Algebra or Boolean Logic) and ***digital circuits***, and hence the design of modern computing systems
- As a computer scientist, you may never have to design digital circuits or other physical components (and you will not learn it here either)
- Understanding how Boolean logic affects the design of various computer system components will allow you to use, from a programming perspective, any computer system more effectively

Why cover logics here?

- It is essential for establishing **hardware** and **software *correctness***
- Logics are used to express properties over the design of an integrated circuit (hardware) or a software system
- Also it’s incredibly beautiful, deep, and fascinating!

Part 2: Some Key Logical Connectives

NOT

- Represented by various symbols, e.g. ¬ , !, -

A	¬ A
F	T
T	F

AND

- Represented by various symbols, e.g. ∧, &, ∩, &&

A	B	A ∧ B
F	F	F
F	T	F
T	F	F
T	T	T

OR

- Represented by various symbols, e.g. ∨, |, ∪, ||

A	B	A∨B
F	F	F
F	T	T
T	F	T
T	T	T

Exclusive OR (XOR)

- Represented by various symbols, e.g. $\oplus$, xor, $\veebar$

A	B	A $\oplus$ B
F	F	F
F	T	T
T	F	T
T	T	F

Implication

- Represented by various symbols, e.g. →, ⇒, ⊃

A	B	A → B
F	F	T
F	T	T
T	F	F
T	T	T

Note that F → T = T

Sometimes called “material implication”.

Material Implication: Odd One Out

- You don’t have to like these logical definitions
  - Just learn them!
  - Especially for material implication →
  - If you don’t like that one, you can do a PhD ….
  - … in intuitionistic logic, or relevance logic, or …
- The logic functions we’ve seen also are common logic gates
  - Again except for material implication
  - But implication is critical for general use of logic

Study it for your PhD …

- Like I did …
- And so did smartest person I know
- And she got it *the same day* that she married me.

Picture of Ian Gent with his (later) wife, about 1992

ASPECTS OF THE COMPUTATIONAL CONTENT OF PROOFS

Judith Lynne Underwood, Ph.D.

Cornell University 1994

In this thesis, we explore three aspects of the computational content of proofs. These are: a computational interpretation of the metatheory of intuitionistic propositional logic, an extension of this approach to intuitionistic predicate logic, and a computational interpretation of classical sequent proofs.

We begin with a study of the computational aspects of validity, provability, and completeness for intuitionistic propositional logic. We give a constructive proof of completeness of Kripke models for intuitionistic propositional calculus, such that the computational content of the proof is a form of the tableau algorithm. Since the evidence for provability we construct is actually a term in typed λ-calculus, we can interpret this result as a formal relationship between Kripke semantics and realizability semantics. We also show how a formal proof of the completeness theorem in Nuprl could be used to add the tableau decision procedure for provability to Nupri's collection of proof techniques via reflection.

Cornell University, 1994

Equivalence

- Represented by various symbols, e.g. $\leftrightarrow$, = , A ≡ B, iff
- "iff" short for "if and only if"

A	B	A $\leftrightarrow$ B
F	F	T
F	T	F
T	F	F
T	T	T

Note that A $\leftrightarrow$ B means the same thing as:

$(A \rightarrow B) \wedge (B \rightarrow A)$

Part 3: Truth tables

- Have seen a few truth tables for connectives
- Haven’t told you how to build more complex ones
- So let’s see this now

Truth tables

- Collect all the *n* propositional variables in formula, make a column for each
- Write down line for each of the 2^n combinations of T/F for these
- Build up *subformulas* you need, one column for each subformula
- E.g. Let's build a truth table for

$$\neg(A \vee B) \rightarrow (C \vee \neg A)$$

- Subformulas are

$$A \vee B$$
$$\neg(A \vee B)$$
$$\neg A$$
$$C \vee \neg A$$

- Evaluate each subformula for each combination of T/F
  - starting from the innermost subformula

¬(A v B) → (C v ¬ A)

A	B	C	A v B	¬(A v B)	¬A	C v ¬ A	¬(A v B) → (C v ¬ A)
T	T	T					
T	T	F					
T	F	T					
T	F	F					
F	T	T					
F	T	F					
F	F	T					
F	F	F					

¬(A v B) → (C v ¬A)

A	B	C	A v B	¬(A v B)	¬A	C v ¬ A	¬(A v B) → (C v ¬ A)
T	T	T	**T**				
T	T	F	**T**				
T	F	T	**T**				
T	F	F	**T**				
F	T	T	**T**				
F	T	F	**T**				
F	F	T	**F**				
F	F	F	**F**				

¬(A v B) → (C v ¬A)

A	B	C	A v B	¬(A v B)	¬A	C v ¬ A	¬(A v B) → (C v ¬ A)
T	T	T	T	**F**			
T	T	F	T	**F**			
T	F	T	T	**F**			
T	F	F	T	**F**			
F	T	T	T	**F**			
F	T	F	T	**F**			
F	F	T	F	**T**			
F	F	F	F	**T**			

¬(A v B) → (C v ¬A)

A	B	C	A v B	¬(A v B)	¬A	C v ¬ A	¬(A v B) → (C v ¬ A)
T	T	T	T	F	**F**		
T	T	F	T	F	**F**		
T	F	T	T	F	**F**		
T	F	F	T	F	**F**		
F	T	T	T	F	**T**		
F	T	F	T	F	**T**		
F	F	T	F	T	**T**		
F	F	F	F	T	**T**		

¬(A v B) → (C v ¬ A)

A	B	C	A v B	¬(A v B)	¬A	C v ¬ A	¬(A v B) → (C v ¬ A)
T	T	T	T	F	F	**T**	
T	T	F	T	F	F	**F**	
T	F	T	T	F	F	**T**	
T	F	F	T	F	F	**F**	
F	T	T	T	F	T	**T**	
F	T	F	T	F	T	**T**	
F	F	T	F	T	T	**T**	
F	F	F	F	T	T	**T**	

¬(A v B) → (C v ¬A)

A	B	C	A v B	¬(A v B)	¬A	C v ¬ A	¬(A v B) → (C v ¬ A)
T	T	T	T	F	F	T	**T**
T	T	F	T	F	F	F	**T**
T	F	T	T	F	F	T	**T**
T	F	F	T	F	F	F	**T**
F	T	T	T	F	T	T	**T**
F	T	F	T	F	T	T	**T**
F	F	T	F	T	T	T	**T**
F	F	F	F	T	T	T	**T**

Logical Operators in C

- C operators | and & are NOT logical, but bitwise
- E.g. 2 & 1 == 0
- (because 2 == 10_2 and 1 == 01_2 and 10_2 & 01_2 == 00_2)
- Remember to use || and && in C for logical operations
- Similarly for logical negation
  - bitwise operator is ~
  - logical operator is !
- Not all operators have logical equivalents
  - bitwise XOR is ^ but no logical equivalent

Part 4: Modus Ponens and ”A Puff of Logic”

Modus Ponens

- If I know P
- And I know P → Q
- Then I know Q

- One of the oldest (the oldest?) logical law

A puff of logic?

"I refuse to prove that I exist," says God, "for proof denies faith, and without faith I am nothing."

"But," says Man, "the Babel fish is a dead giveaway, isn't it? It could not have evolved by chance. It proves you exist, and so therefore, by your own arguments, you don't. QED."

"Oh dear," says God, "I hadn't though of that" and promptly vanishes in a *puff of logic.*

Douglas Adams

The Hitch-Hiker's Guide to the Galaxy

Let’s formalise this:

- E: *God exists*
- P: *there is a proof*
- F: *there is faith*
- B: *the Babel Fish exists*
- C: *living beings evolved by chance*

Formalise the English into Logic

Proof denies faith	**$P \rightarrow \neg F$**
[If there is a proof then God exists]	**$P \rightarrow E$**
Without Faith I am nothing	**$\neg F \rightarrow \neg E$**
The Babel fish is a dead giveaway	**B**
It could not have evolved by chance	**$B \rightarrow \neg C$**
It proves you exist	**$\neg C \rightarrow P$**

Use Modus Ponens on the statements we have.

1.P → ¬F
2.P → E
3.¬F → ¬E
4.B
5.B → ¬C
6.¬C → P

7.¬C *{ Modus Ponens, 4, 5}*
8.P *{ Modus Ponens, 7, 6}*
9.¬F *{ Modus Ponens, 8, 1}*
10. ¬E *{ Modus Ponens, 9, 3}*
11. E *{ Modus Ponens, 8, 2}*

Vanishes in a Puff of Logic?

- We have proved **¬E** *and* **E**
- What have we learnt?

Vanishes in a Puff of Logic?

- We have proved **¬E** *and* **E**
- What have we learnt?

- Exactly and only that the set of premises are logically inconsistent
- Not all the statements can be true
- Not too surprising since HHGttG is a work of fiction

Next Time

- Equivalence and Tautology
- Another example of formalization
- A larger example of formalization
- (If we have time) Proving things in Logic