Computability and Logic

Harvey Mudd College

CS 81

Spring Semester 2009

Prof. Bob Keller

Data

- Bob Keller
- Office: 1253 Olin
- Phone: x 1-8483
- Office Hours: M, T, W: 4:15-5:30, or by appointment, or by drop-in
- Unavailable: MTWTh: 2:45-4, TTh: 1:15-2:30, TTh 11-12, M 9-10:30, Fri AM

Computability vs. Logic

- Computability:
  - Models of computation
  - What functions, languages, etc. are computable in those models?
- Logic:
  - Languages for expressing assertions
  - What assertions can be expressed in a given language?
  - Which assertions are provable and how?

Ways in which Computability and Logic are Related

- As logic entails language, we can hope to determine the truth of (or a proof of) an assertion by computation.

- As computational models are themselves expressible in languages, we can hope to answer questions about the model by using logic.

Example: Turing Machines

- Given a Turing machine, we might be interested in whether the machine can reach state B from state A.

- There is a logical formula that is provable iff this reachability is possible.

Example: First-Order Predicate Logic

- There is a Turing machine that will "recognize" just the provable formulas in predicate logic.

- (Unfortunately, there isn't one that will recognize just the unprovable ones.)

Which is Broader?

- All of computability can be expressed in logic, but not conversely.
- Most of mathematics can be expressed in logic.
- Why logic is not universally used has to do with convenience, overhead, and other aspects of human nature.

History of Logic (WP)

(WP = Wikipedia)

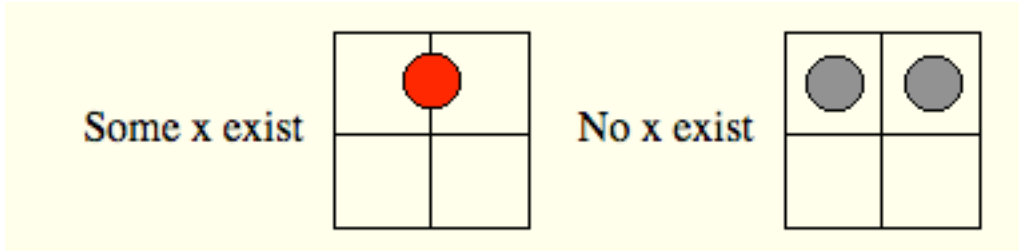
"The history of logic is the study of the development of the science of valid inference (logic). While many cultures have employed intricate systems of reasoning, and logical methods are evident in all human thought, an explicit analysis of the principles of reasoning was developed only in three traditions: those of China, India, and Greece. Although exact dates are uncertain, particularly in the case of India, it is possible that logic emerged in all three societies by the 4th century BC. The formally sophisticated treatment of modern logic descends from the Greek tradition, particularly Aristotelian logic, which was further developed by Islamic logicians and then medieval European logicians. The work of Frege in the 19th century marked a radical departure from the Aristotelian leading to the rapid development of symbolic logic, later called mathematical logic."

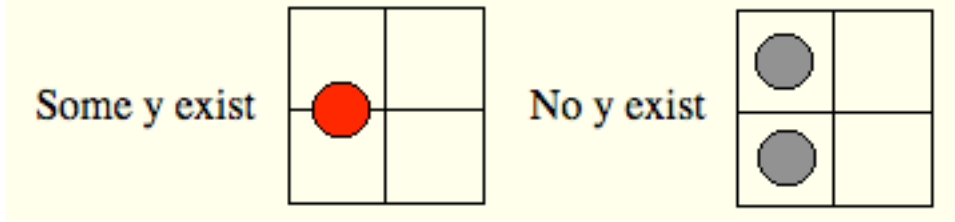
Gottlob Frege (1848-1925)

- Created modern logic by introducing the predicate calculus.
- Developed a formalized definition of "proof".
- Defined the natural numbers, anticipating Peano's axiomatization (1889).
- Did not anticipate Russell's paradox.

Lewis Caroll's "Game of Logic" (1886)

- Bi- and Tri-lateral diagrams





Russell's Paradox (1902)

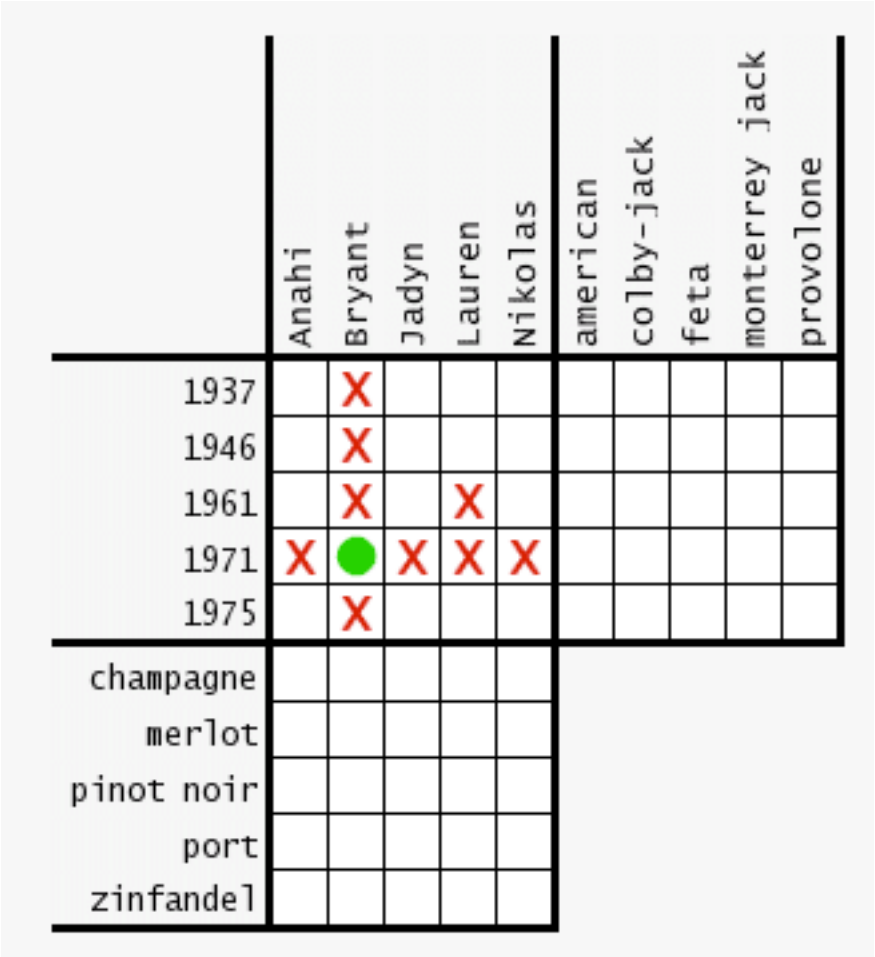
- Consider the "set" P:

  $P = \{S \mid S \text{ is a set and } S \notin S\}$

- Is $P \in P$?
  - If $P \in P$, then $P \notin P$.
  - If $P \notin P$, then $P \in P$.

- We are forced to conclude that P can't be a set after all.

“Logic Puzzles”



	Anahi	Bryant	Jadyn	Lauren	Nikolas	american	colby-jack	feta	monterrey jack	provolone
1937		X								
1946		X								
1961		X		X						
1971	X	●	X	X	X					
1975		X								
champagne										
merlot										
pinot noir										
port										
zinfandel										

WFF 'n Proof

Forty years ago, WFF 'N PROOF Publishers based its revolutionary approach to education on these fundamental truths. The learning games and puzzles offered by WFF 'N PROOF Publishers are designed by university faculty to teach a broad range of subject matter to players of all ages in a profound yet engaging manner. They are incredibly effective.

Studies show that these games positively ignite motivation and learning.

- 3 weeks of intensive play of WFF 'N PROOF: The Game of Modern Logic **increased average IQ scores by more than 20 points.**

Mastermind game (1970)

A game of Mastermind completed.

Prolog Language (1972)

```
sibling(X, Y)      :- parent_child(Z, X), parent_child(Z, Y).

parent_child(X, Y) :- father_child(X, Y).
parent_child(X, Y) :- mother_child(X, Y).

mother_child(trude, sally).

father_child(tom, sally).
father_child(tom, erica).
father_child(mike, tom).
```

Automated Theorem Provers (1960's on)

- Otter
- ACL2
- PTTP

to name a few

Proof Editors

- JAPE
  and others

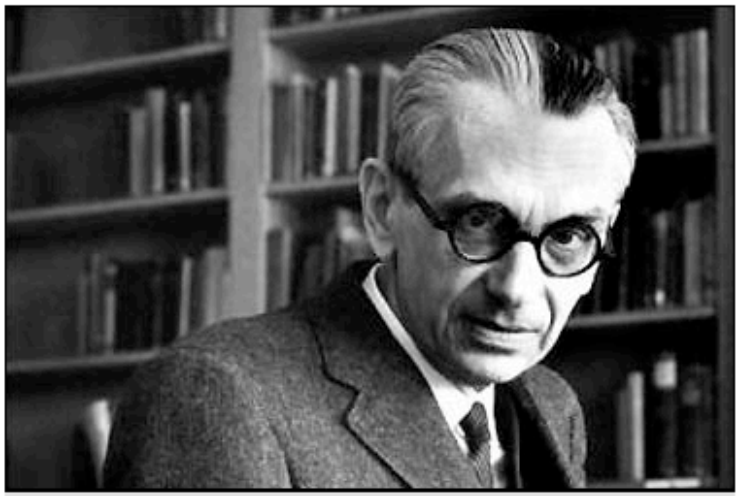
Hilbert's "Program" (WP)

- All mathematical statements should be written in a precise formal language, and manipulated according to well defined rules.
- **Completeness**: a proof that all true mathematical statements can be proved in the formalism.
- **Consistency**: a proof that no contradiction can be obtained in the formalism of mathematics. This consistency proof should preferably use only "finitistic" reasoning about finite mathematical objects.
- **Conservation**: a proof that any result about "real objects" obtained using reasoning about "ideal objects" (such as uncountable sets) can be proved without using ideal objects.
- **Decidability**: there should be an algorithm for deciding the truth or falsity of any mathematical statement.

Most Important Mathematical Result of the 20th Century?

THE TIME 100 1901 2000

LEADERS & REVOLUTIONARIES | ARTISTS & ENTERTAINERS | BUILDERS & TITANS



ALFRED EISENSTAEDT/TIME LIFE PICTURES

Kurt Godel at the Institute of Advanced Study

SCIENTISTS & THINKERS

Kurt Gödel

He turned the lens of mathematics on itself and hit upon his famous "incompleteness theorem" — driving a stake through the heart of formalism

By DOUGLAS HOFSTADTER

Meta-logic
(aka Metamathematics)

means proving things about a logic.

We will do a certain amount of this.

Meta-language vs. Object-language distinction.

We will start with logic.

- I assume you are familiar (from CS 5, 60, etc.) with **propositional logic**, and most of these symbols (from Ben-Ari's book):

x	y	$\wedge$	$\vee$	$\rightarrow$	$\leftrightarrow$	$\oplus$	$\uparrow$	$\downarrow$
T	T	T	T	T	T	F	F	F
T	F	F	T	F	F	T	T	F
F	T	F	T	T	F	T	T	F
F	F	F	F	T	T	F	T	T

Examples of Alternate Symbology

Operator	Alternates	C language
$\neg$	$\sim$	!
$\wedge$	&	&
$\vee$		|
$\rightarrow$	$\supset$, $\Rightarrow$	
$\leftrightarrow$	$\equiv$, $\Leftrightarrow$	
$\oplus$	$\not\equiv$	^
$\uparrow$	|	

Abstract Grammar for Formulas

- P is an infinite set of proposition symbols.
- fml is the start symbol for the grammar.
- This grammar is "abstract" because it does not mention precedence, grouping, or parens:

1. $fml ::= p$ for any $p \in P$
2. $fml ::= \neg fml$
3. $fml ::= fml \vee fml$
4. $fml ::= fml \wedge fml$
5. $fml ::= fml \rightarrow fml$
6. $fml ::= fml \leftrightarrow fml$
7. $fml ::= fml \oplus fml$
8. $fml ::= fml \uparrow fml$
9. $fml ::= fml \downarrow fml$.

Precedence in Ben-Ari's Notation

. In propositional formulas the order of precedence from high to low is as follows: negation, conjunction, nand, disjunction, nor, implication, equivalence. Operators are assumed to associate to the right, that is, $a \vee b \vee c$ means $(a \vee (b \vee c))$. Parentheses are used only if needed to indicate an order different from that imposed by the precedence, as in arithmetic where $a * (b + c)$ needs parentheses to denote that the addition is done before the multiplication, while $a * b + c$ does not need them to denote the multiplication before addition. With minimal use of parentheses, the propositional formulas above can be written:

$$p \rightarrow q \leftrightarrow \neg q \rightarrow \neg p,$$
$$p \rightarrow (q \leftrightarrow \neg (p \rightarrow \neg q)).$$

Formation Trees are Unambiguous

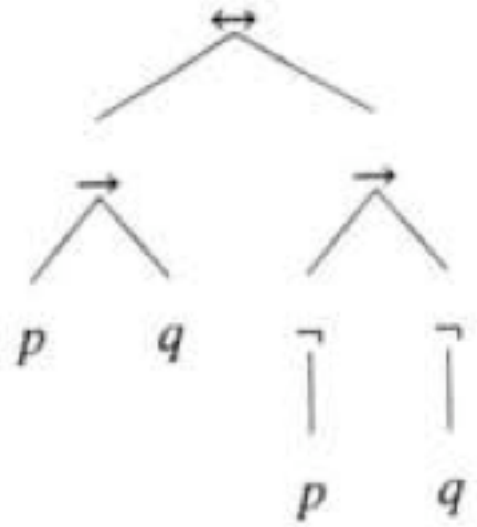


Figure 2.3 Formation tree for $p \rightarrow q \leftrightarrow \neg p \rightarrow \neg q$

Note on Arity

- All operators are 2-ary, except for ¬, which is 1-ary.

Principle of Structural Induction

Theorem 2.5 (Structural induction) *To show property(A) for all formulas $A \in \mathcal{F}$, it suffices to show:*

1. *property(p) for all atoms p.*
2. *Assuming property(A), then property($\neg A$) holds.*
3. *Assuming property(A_1) and property(A_2), then property(A_1 op A_2) hold, for each of the operators op.*

Assignment

- An **assignment** ν is a function from the set of proposition symbols into {T, F}.

- Assignments are also called "valuations".
- Note that an assignment is essentially the same as an **environment** that you probably studied in the context of language interpreters.

Interpretation

- An **interpretation** ν (Greek nu) extends an assignment ν to arbitrary formulas as follows:
  - $\nu(\varphi)$, where φ is just a proposition letter, is given by the assignment ν.
  - $\nu(\neg\varphi)$ is T if $\nu(\varphi)$ if F, and vice-versa.
  - $\nu(\varphi \circ \psi)$ is given by the truth-table for $\circ$ applied to $\nu(\varphi)$ and $\nu(\psi)$.

Clarification of Assignment vs. Interpretation

- The difference is in the domain of the two functions.
- An assignment maps the set of **proposition symbols** into {T, F}.
- An interpretation maps the set of formulas into {T, F}.
- A formula can consist of just a single proposition symbol, but technically the formula and the symbol of which it consists are distinct (just as a set of one element is not the same as the element itself).
- Given an assignment, an interpretation for any formula is determined inductively/recursively by the assignment.
- It just happens that the book "overloads" the symbol ν by using it for both assignment and interpretation.

Truth in an Interpretation Satisfaction

- A formula φ is ***true in an interpretation*** ν iff $\nu(\varphi) = T$.

- We also say that the interpretation ν **satisfies** the formula φ in the exactly this case.

Equivalence (≡)

- $\varphi \equiv \psi$ means

  for every interpretation ν

  $\nu(\varphi) = \nu(\psi)$

- Obviously we only need be concerned with the values the assignments ν assign to proposition symbols in φ and ψ in order to determine whether or not $\varphi \equiv \psi$.

Equivalence (≡) vs. ↔

- ≡ is in not in the alphabet, but rather in the meta-language.
- The iff ↔ symbol is an operator symbol in the alphabet.

Example

- Is $p \rightarrow \neg p \equiv \neg p$?
- If $\nu(p) = T$, then $\nu(\neg p) = F$, and $\nu(p \rightarrow \neg p) = (T \rightarrow F) = F$.
- If $\nu(p) = F$, then $\nu(\neg p) = T$, and $\nu(p \rightarrow \neg p) = (F \rightarrow T) = T$.
- So yes.

Example

- Is $p \rightarrow \neg q \equiv \neg(p \wedge q)$?

$v(p)$	$v(q)$	$v(p \rightarrow \neg q)$	$v(\neg(p \wedge q))$
F	F	T	T
F	T	T	T
T	F	T	T
T	T	F	F

Yes.

Equivalences to Know

- The list on page 21 of Ben Ari.

Lemma

- For any formulas, φ and ψ,

$$\varphi \equiv \psi$$

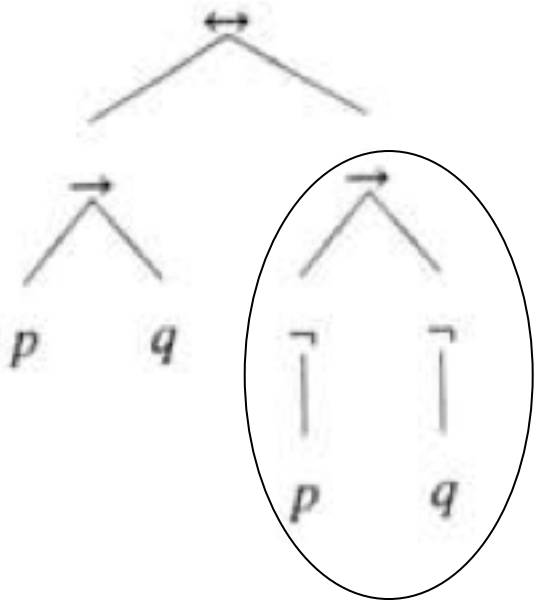
iff

for every interpretation ν
$\nu(\varphi \leftrightarrow \psi) = T$.

Proof

- Suppose $\varphi \equiv \psi$.
- Let ν be an arbitrary interpretation.
- Then $\nu(\varphi) = \nu(\psi)$ from the definition of $\equiv$.
- From the definition of $\leftrightarrow$ we see that $\nu(\varphi \leftrightarrow \psi) = T$.
- This argument is reversible.

Sub-Formulas



A sub-formula

Figure 2.3 Formation tree for $p \rightarrow q \leftrightarrow \neg p \rightarrow \neg q$

Sub-Formula Replacement

- By $A[B \leftarrow C]$ we mean the result of replacing every sub-formula instance of B in A with C.

- Theorem 2.21: Suppose $B \equiv C$. Then $A[B \leftarrow C] \equiv A$.

Proof

- Let ν be an arbitrary interpretation.
- If B is a sub-formula of A, then the value of $\nu(A)$ is a ***function*** of $\nu(B)$.
- If $B \equiv C$, then by definition $\nu(B) \equiv \nu(C)$.
- Thus $\nu(A) = \nu(A[B \leftarrow C])$.
- Hence $A \equiv A[B \leftarrow C]$.

"Definable from"

Definition 2.22 A binary operator $\circ$ *is defined from* a set of operators $\{\circ_1, \ldots, \circ_n\}$ iff there is a logical equivalence $A_1 \circ A_2 \equiv A$, where A is a formula constructed from occurrences of A_1 and A_2 using the operators $\{\circ_1, \ldots, \circ_n\}$. Similarly, the (only non-trivial) unary operator $\neg$ is defined by a formula $\neg A_1 \equiv A$, where A is constructed from occurrences of A_1 and the operators in the set. □

Theorem 2.23 *Let $\circ$ be a binary operator that can define negation and all other binary operators. Then $\circ$ is either nand or nor.*

- Read proof on page 23.

Satifiability

- A formula φ is ***true in an interpretation*** ν iff $\nu(\varphi) = T$.
- We also say that the interpretation ν **satisfies** the formula φ in the exactly this case.
- If there is *an* interpretation satisfying φ, we say φ is **satisfiable**.

Validity

- If *every* interpretation satisfies φ, we say φ is a **valid** (or is a **tautology**).

Validity vs. Satisfiability

- Lemma:

  φ is a valid

  iff

  $\neg\varphi$ is *un*satisfiable.

Proof

- Note that $\nu(\neg\varphi) = \neg\nu(\varphi)$.

  φ is valid
  iff
  for all ν, $\nu(\varphi) = T$
  iff
  for all ν, $\nu(\neg\varphi) = F$
  iff
  for no ν, $\nu(\neg\varphi) = T$
  iff
  for no ν, $\nu(\neg\varphi) = T$
  iff
  $\neg\varphi$ is not satisfiable.

Simultaneous Satisfiability

- Let U be a set of formulas (not necessarily finite).
- U is (simultaneously) **satisfiable** iff there is an interpretation that satisfies every formula in U.
- If not, U is **unsatisfiable**.

Examples

- $\{p \vee \neg q, \neg p \vee r, q, \neg r\}$
- $\{p \vee \neg q,\ p \vee \neg r, \neg p\}$
- $\{p \wedge (\neg q \vee s), \neg s, q\}$
- The set of all formulas.
- { } (empty)

Theorems 2.32-2.35

- Set +/- formula properties:
- Satisfiable – arbitrary = _____.
- Satisfiable + valid = _____.
- Unsatisfiable + arbitrary = _____.
- Unsatisfiable – valid = _____.

Logical Consequence Entailment |=

- U is a set of formulas (not necessarily finite), *A* is a formula.
- |= should be read as one symbol. It is in the meta-language.
- U |= *A* (read "U entails *A*") means:

  Every interpretation that simultaneously satisfies U also satisfies *A*.

- |= *A* means the case where U is empty, i.e. *A* is valid.

Theorem 2.38

- If U is finite, $U = \{A_1, \ldots, A_n\}$, then the following are equivalent:
  - $U \models A$
  - $\models (A_1 \wedge \ldots \wedge A_n) \rightarrow A$

Validity Algorithms

- There are various ways to check the validity of a propositional formula:
  - Truth-table
  - Boole-Shannon expansion
  - [Resolution]
  - etc.

Proof Systems

- [Here I am detouring from Ben-Ari for a bit.]
- A proof system uses symbol manipulation to derive formulas from other formulas.
- It does not rely on a specific algorithm, although connections can be established in some cases.

Natural Deduction

Gerhard Gentzen (1909-1945)

- Formulas are derived through a series of symbol manipulations involving introduction and elimination rules.
- The rules are organized by the logical connectives.
- Proofs can be shown as trees or tables.
- Natural deduction provides a useful template for meta-logic proofs as well.

|—

- |— is not the same as |=
- U |— *A* means that *A* is provable from formulas in U.
- Roughly speaking, a proof takes the form of a sequence of formulas, where each formula is either in U or is derivable from earlier formulas in the proof.

_____ notation

- A horizontal bar is used to express rules of inference:
  - From formulas above the bar (of which there may be 0 or more)
  - the formula below the bar is derivable.