Tautologies

A *truth assignment* is an assignment of T or F to every proposition.

- How hard is it to check if a formula is true under a given truth assignment?
- Easy: just plug it in and evaluate.
  - Time linear in the length of the formula

A *tautology* (or *theorem*) is a formula that evaluates to T for *every* truth assignment.

Examples:

- $(P \vee Q) \Leftrightarrow \neg(\neg P \wedge \neg Q)$
- $P \vee Q \vee (\neg P \wedge \neg Q)$
- $(P \Rightarrow Q) \vee (Q \Rightarrow P)$
  - It's necessarily true that if elephants are pink then the moon is made of green cheese or if the moon is made of green cheese, then elephants are pink.

How hard is it to check if a formula is a tautology?

- How many truth assignments do we have to try?

Arguments

Definition: An argument has the form

A_1
A_2
$\vdots$
A_n

B

$A_1, \ldots, A_n$ are called the *premises* of the argument; B is called the *conclusion*. An argument is *valid* if, whenever the premises are true, then the conclusion is true.

Logical Implication

A formula A *logically implies* B if $A \Rightarrow B$ is a tautology.

Theorem: An argument is valid iff the conjunction of its premises logically implies the conclusion.

Proof: Suppose the argument is valid. We want to show $(A_1 \wedge \ldots \wedge A_n) \Rightarrow B$ is a tautology.

- Do we have to try all 2^k truth assignments (where $k = \#$primitive propositions in $A_1, \ldots, A_n, B$).

It's not that bad.

- Because of the way we defined $\Rightarrow$, $A_1 \wedge \ldots \wedge A_n \Rightarrow B$ is guaranteed to be true if $A_1 \wedge \ldots \wedge A_n$ is false.
- But if $A_1 \wedge \ldots \wedge A_n$ is true, B is true, since the argument is valid.
- Thus, $(A_1 \wedge \ldots \wedge A_n) \Rightarrow B$ is a tautology.

For the converse, suppose $(A_1 \wedge \ldots \wedge A_n) \Rightarrow B$ is a tautology. If $A_1, \ldots, A_n$ are true, then B must be true. Hence the argument is valid.

Remember:

Borogroves are mimsy whenever it is brillig.
It is now brillig and this thing is a borogrove.
Hence this thing is mimsy.

Suppose

- P: It is now brillig
- Q: This thing is a borogrove
- R: This thing is mimsy

This becomes:

$P \Rightarrow (Q \Rightarrow R)$
$P \wedge Q$

R

This argument is valid if

$$[(P \Rightarrow (Q \Rightarrow R)) \wedge (P \wedge Q)] \Rightarrow R$$

is a tautology.

Natural Deduction

Are there better ways of telling if a formula is a tautology than trying all possible truth assignments.

- In the worst case, it appears not.
  - The problem is co-NP-complete.
  - The *satisfiability* problem—deciding if at least one truth assignment makes the formula true—is NP-complete.

Nevertheless, it often seems that the reasoning is staight-forward:
Why is this true:

$$((P \Rightarrow Q) \wedge (Q \Rightarrow R)) \Rightarrow (P \Rightarrow R)$$

We want to show that if $P \Rightarrow Q$ and $Q \Rightarrow R$ is true, then $P \Rightarrow R$ is true.

So assume that $P \Rightarrow Q$ and $Q \Rightarrow R$ are both true. To show that $P \Rightarrow R$, assume that P is true. Since $P \Rightarrow Q$ is true, Q must be true. Since $Q \Rightarrow R$ is true, R must be true. Hence, $P \Rightarrow R$ is true.

We want to codify such reasoning.

Formal Deductive Systems

A *formal deductive system* (also known as an *axiom system*) consists of

- *axioms* (special formulas)
- *rules of inference*: ways of getting new formulas from other formulas. These have the form

  $$\begin{array}{l} A_1 \\ A_2 \\ \vdots \\ A_n \\ \hline B \end{array}$$

  Read this as "from $A_1, \ldots, A_n$, infer B."
  - Sometimes written "$A_1, \ldots, A_n \vdash B$"

Think of the axioms as tautologies, while the rules of inference give you a way to derive new tautologies from old ones.

Derivations

A *derivation* (or *proof*) in an axiom system AX is a sequence of formulas

$$C_1, \ldots, C_N;$$

each formula C_k is either an axiom in AX or follows from previous formulas using an inference rule in AX:

- i.e., there is an inference rule $A_1, \ldots, A_n \vdash B$ such that $A_i = C_{j_i}$ for some $j_i < N$ and $B = C_N$.

This is said to be a *derivation* or *proof* of C_N.

A derivation is a syntactic object: it's just a sequence of formulas that satisfy certain constraints.

- Whether a formula is derivable depends on the axiom system
- Different axioms → different formulas derivable
- Derivation has nothing to do with truth!
  - How can we connect derivability and truth?

Typical Axioms

- $P \Rightarrow \neg\neg P$
- $P \Rightarrow (Q \Rightarrow P)$

What makes an axiom "acceptable"?

- it's a tautology

Typical Rules of Inference

Modus Ponens

$A \Rightarrow B$
A

B

Modus Tollens

$A \Rightarrow B$
$\neg B$

$\neg A$

What makes a rule of inference "acceptable"?

- It preserves validity:
  - if the antecedents are valid, so is the conclusion
- Both modus ponens and modus tollens are acceptable

Sound and Complete Axiomatizations

Standard question in logic:

> Can we come up with a nice *sound and complete axiomatization*: a (small, natural) collection of axioms and inference rules from which it is possible to derive all and only the tautologies?

- *Soundness* says that only tautologies are derivable
- *Completeness* says you can derive all tautologies

If all the axioms are valid and all rules of inference preserve validity, then all formulas that are derivable must be valid.

- Proof: by induction on the length of the derivation

It's not so easy to find a complete axiomatization.

A Sound and Complete Axiomatization for Propositional Logic

Consider the following axiom schemes:

A1. $A \Rightarrow (B \Rightarrow A)$

A2. $(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$

A3. $((A \Rightarrow B) \Rightarrow (A \Rightarrow \neg B)) \Rightarrow \neg A$

These are axioms schemes; each one encodes an infinite set of axioms:

- $P \Rightarrow (Q \Rightarrow P)$, $(P \Rightarrow R) \Rightarrow (Q \Rightarrow (P \Rightarrow R))$ are instances of A1.

Theorem: A1, A2, A3 + modus ponens give a sound and complete axiomatization for formulas in propositional logic involving only $\Rightarrow$ and $\neg$.

- Recall: can define $\vee$ and $\wedge$ using $\Rightarrow$ and $\neg$
  - $P \vee Q$ is equivalent to $\neg P \Rightarrow Q$
  - $P \wedge Q$ is equivalent to $\neg(P \Rightarrow \neg Q)$

A Sample Proof

Derivation of $P \Rightarrow P$:

1. $P \Rightarrow ((P \Rightarrow P) \Rightarrow P)$
   [instance of A1: take $A = P$, $B = P \Rightarrow P$]
2. $(P \Rightarrow ((P \Rightarrow P) \Rightarrow P)) \Rightarrow ((P \Rightarrow (P \Rightarrow P)) \Rightarrow (P \Rightarrow P))$
   [instance of A2: take $A = C = P$, $B = P \Rightarrow P$]
3. $(P \Rightarrow (P \Rightarrow P)) \Rightarrow (P \Rightarrow P)$
   [applying modus ponens to 1, 2]
4. $P \Rightarrow (P \Rightarrow P)$ [instance of A1: take $A = B = P$]
5. $P \Rightarrow P$ [applying modus ponens to 3, 4]

Try deriving $P \Rightarrow \neg\neg P$ from these axioms

- it's hard!

Algorithm Verification

This is (yet another) hot area of computer science.

- How do you prove that your program is correct?
  - You could test it on a bunch of instances. That runs the risk of not exercising all the features of the program.

In general, this is an intractable problem.

- For small program fragments, formal verification using logic is useful
- It also leads to insights into program design.

Algorithm Verification: Example

Consider the following algorithm for multiplication:

Input x [Integer ≥ 0]
y [Integer]
Algorithm Mult
$prod \leftarrow 0$
$u \leftarrow 0$
repeat $u = x$
$prod \leftarrow prod + y$
$u \leftarrow u + 1$
end repeat

How do we prove this is correct?

- Idea (due to Floyd and Hoare): annotate program with assertions that are true of the line of code immediately following them.
- An assertion just before a loop is true each time the loop is entered. This is a *loop invariant*.
- An assertion at the end of a program is true after running the program.

Input x [Integer ≥ 0]
y [Integer]
Algorithm Mult
$prod \leftarrow 0$
$u \leftarrow 0$
$\{prod = uy\}$ [Loop invariant]
repeat $u = x$
$prod \leftarrow prod + y$
$u \leftarrow u + 1$
end repeat
$\{prod = uy \wedge u = x\}$

Thus, we must show $prod = uy$ is true each time we enter the loop.

- Proof is by induction (big surprise)

It follows that $prod = uy \wedge u = x$ holds after exiting the program, since we exit after trying the loop (so $prod = uy$) and discovering $u = x$. It follows that $prod = xy$ at termination.

But how do we know the program terminates?

- We prove (by induction!) that after the kth iteration of the loop, $u = k$.
- Since $x \geq 0$, eventually $u = x$, and we terminate the loop (and program)

We won't be covering Boolean algebra (it's done in CS 314), although you should read Section 7.5!

Predicate Calculus

There are lots of things that can't be expressed by propositional formulas. In first-order logic, we can:

- Talk about individuals and the properties they have:
  - Bob and Alice are both American
    $American(Bob) \wedge American(Alice)$
- Talk about the relations between individuals
  - Bob loves Alice but Bob doesn't love Anne
    $Loves(Bob, Alice) \wedge \neg Loves(Bob, Anne)$.
- Quantify:
  - Everybody loves somebody
    $\forall x \exists y Loves(x, y)$

First-order logic lets us capture arguments like:

All men are mortal
Socrates is a man
Therefore Socrates is mortal

All prime numbers are integers
7 is a prime number
Therefore 7 is an integer

Syntax of First-Order Logic

We have:

- *constant symbols*: *Alice*, *Bob*
- variables: $x, y, z, \ldots$
- *predicate symbols* of each arity: P, Q, R, ...
  - A *unary* predicate symbol takes one argument: $P(Alice)$, $Q(z)$
  - A *binary* predicate symbol takes two arguments: *Loves(Bob,Alice)*, *Taller(Alice,Bob)*.

An *atomic expression* is a predicate symbol together with the appropriate number of arguments.

- Atomic expressions act like primitive propositions in propositional logic
  - we can apply $\wedge$, $\vee$, $\neg$ to them
  - we can also quantify the variables that appear in them

Typical formula:

$$\forall x \exists y (P(x, y) \Rightarrow \exists z Q(x, z))$$

Semantics of First-Order Logic

Assume we have some domain D.

- The domain could be finite:
  - $\{1, 2, 3, 4, 5\}$
  - the people in this room
- The domain could be infinite
  - N, R, ...

A statement like $\forall x P(x)$ means that $P(d)$ is true for each d in the domain.

- If the domain is N, then $\forall x P(x)$ is equivalent to

$$P(0) \wedge P(1) \wedge P(2) \wedge \ldots$$

Similarly, $\exists x P(x)$ means that $P(d)$ is true for some d in the domain.

- If the domain is N, then $\exists x P(x)$ is equivalent to

$$P(0) \vee P(1) \vee P(2) \vee \ldots$$

Is $\exists x(x^2 = 2)$ true?

Yes if the domain is R; no if the domain is N.

How about $\forall x \forall y((x < y) \Rightarrow \exists z(x < z < y))$?

First-Order Logic: Formal Semantics

How do we decide if a first-order formula is true? Need:

- a domain D (what are you quantifying over)
- an *interpretation* I that interprets the constants and predicate symbols:
  - for each constant symbol c, $I(c) \in D$
    - Which domain element is Alice?
  - for each unary predicate P, $I(P)$ is a predicate on domain D
    - formally, $I(P)(d) \in \{\text{true,false}\}$ for each $d \in D$
    - Is Alice Tall? How about Bob?
  - for each binary predicate Q, $I(Q)$ is a predicate on $D \times D$:
    - formally, $I(Q)(d_1, d_2) \in \{\text{true,false}\}$ for each $d_1, d_2 \in D$
    - Is Alice taller than Bob?
- a valuation V associating with each variable x an element $V(x) \in D$.
  - To figure out if $P(x)$ is true, you need to know what x is.

Now we can define whether a formula A is true, given a domain D, an interpretation I, and a valuation V, written

$$(I, D, V) \models A$$

- Read this from right to left, like Hebrew: A is true at ($\models$) (I, D, V)

The definition is by induction:

$(I, D, V) \models P(x)$ if $I(P)(V(x)) = \text{true}$
$(I, D, V) \models P(c)$ if $I(P)(I(c))) = \text{true}$
$(I, D, V) \models \forall x A$ if $(I, D, V') \models A$ for all valuations V' that agree with V except possibly on x

- $V'(y) = V(y)$ for all $y \neq x$
- $V'(x)$ can be arbitrary

$(I, D, V) \models \exists x A$ if $(I, D, V') \models A$ for some valuation V' that agrees with V except possibly on x.

Translating from English to First-Order Logic

All men are mortal
Socrates is a man
Therefore Socrates is mortal

There is two unary predicates: *Mortal* and *Man*
There is one constant: *Socrates*
The domain is the set of all people

$\forall x(Man(x) \Rightarrow Mortal(x))$
$Man(Socrates)$

$Mortal(Socrates)$

More on Quantifiers

$\forall x \forall y P(x, y)$ is equivalent to $\forall y \forall x P(x, y)$

- P is true for every choice of x and y

Similarly $\exists x \exists y P(x, y)$ is equivalent to $\exists y \exists x P(x, y)$

- P is true for some choice of (x, y).

What about $\forall x \exists y P(x, y)$? Is it equivalent to $\exists y \forall x P(x, y)$?

- Suppose the domain is the natural numbers. Compare:
  - $\forall x \exists y (y \geq x)$
  - $\exists y \forall x (y \geq x)$

In general, $\exists y \forall x P(x, y) \Rightarrow \forall x \exists y P(x, y)$ is *logically valid*.

- A logically valid formula in first-order logic is the analogue of a tautology in propositional logic.
- A formula is logically valid if it's true in every domain and for every *interpretation* of the predicate symbols.

More valid formulas involving quantifiers:

- $\neg \forall x P(x) \Leftrightarrow \exists x \neg P(x)$
- Replacing P by $\neg P$, we get:

$$\neg \forall x \neg P(x) \Leftrightarrow \exists x \neg \neg P(x)$$

- Therefore

$$\neg \forall x \neg P(x) \Leftrightarrow \exists x P(x)$$

- Similarly, we have

$$\neg \exists x P(x) \Leftrightarrow \forall x \neg P(x)$$

$$\neg \exists x \neg P(x) \Leftrightarrow \forall x P(x)$$

Bound and Free Variables

$\forall i(i^2 > i)$ is equivalent to $\forall j(j^2 > j)$:

- the i and j are *bound* variables, just like the i, j in

$$\sum_{i=1}^{n} i^2 \text{ or } \sum_{j=1}^{n} j^2$$

What about $\exists i(i^2 = j)$:

- the i is bound by $\exists i$; the j is *free*. Its value is unconstrained.
- if the domain is the natural numbers, the truth of this formula depends on the value of j.

Axiomatizing First-Order Logic

Just as in propositional logic, there are axioms and rules of inference that provide a sound and complete axiomatization for first-order logic, independent of the domain.

A typical axiom:

- $\forall x(P(x) \Rightarrow Q(x)) \Rightarrow (\forall x P(x) \Rightarrow \forall x Q(x))$.

A typical rule of inference is *Universal Generalization*:

$$\frac{\varphi(x)}{\forall x \varphi(x)}$$

Gödel proved completeness of this axiom system in 1930.