Lecture 1 - 5th Sept 2024

Formal Verification (FV): viewing a program as a logical argument and determining whether it behaves correctly using symbolic reasoning.

What is logical reasoning?

- **Premises:** facts
- **Conclusions:** what we deduce from the premises. The conclusion logically follows from the facts.
  - ↳ **Example:** if the train arrives late and there are no taxis at the station, then John is late for his meeting. John is not late for his meeting. The train did arrive late. Were there taxis at the station?
    - ↳ Yes!

What is formal logic?

- **Syntax:** acceptable sentence in the logic
- **Semantics:** what the symbols and sentences in the logic mean.
- **Proof theory:** how the valid proofs in the logic are constructed.
- **Logic provides:**
  - ↳ A language for expressing knowledge precisely (modelling / specification)
  - ↳ A way to reason about the consequences of that knowledge rigorously (proof / verification)
- A logic is formal if there is only one possible interpretation of a formula in the language.

- formal logic is concerned with the structure of the argument.
- we use symbols in the syntax to represent phrases in the sentences. It's also called symbolic logic.
  ↳ using symbols allows us to conquer complexity and eliminate non-logical aspects of argument.

- Propositional logic: a declarative sentence that is either true or false, but not both.

- Predicate logic: a means of describing relationships between objects, and a quantification over objects.
  ↳ eg: "every course has an instructor."

- specification: ways of describing what a system is required to do.

- program correctness: a progam is correct iff the output is correct for every input.

- the syntax of a logic defines how to arrange symbols into a well-formed formula (wff)

Semantics:

- $\models$: logical implication ← NOT a part of wff, but a meta-level symbol.
- $\Leftrightarrow$: logical equivalence
- $\vdash$: proves → $P_1, P_2, \ldots, P_n \vdash Q$ means that from $P_1, P_2, \ldots, P_n$, we can prove Q using a proof theory.

Proof Theories: methods that perform **mechanical manipulations** on strings of symbols.
↳ based on pattern matching!

On $P_1, P_2, \ldots, P_n \vdash Q$ and $P_1, P_2, \ldots, P_n \models Q$,

goals: if we **aren't yet sure they are valid** / proven
theorems: if they've been **determined to be valid** / proven

Soundness: a proof theory is sound **if whenever** $P_1, P_2, \ldots, P_n \vdash Q$ (we have a proof), **then** $P_1, P_2, \ldots, P_n \models Q$ (valid)

Completeness: a proof theory is complete **if whenever** $P_1, P_2, \ldots, P_n \models Q$ (valid) **then** $P_1, P_2, \ldots, P_n \vdash Q$ (there is a proof).

Propositional Connectives:

symbol	informal meaning	george
$\neg$	negation (not)	!
$\wedge$	conjunction (and / both)	&
$\vee$	disjunction (or / at least one)	|
$\Rightarrow$	implication (if / implies)	=>
$\Leftrightarrow$	equivalent (biconditional, iff)	<=>

Brackets and Precedence:

- Brackets around the outermost formula are usually omitted.

- The order of precedence for the operators is shown in the above table, with ¬ having the highest precedence and <=> having the lowest.

- All binary logical connectives are right-associative.
  - ↳ Eg: $a \vee b \vee c \rightarrow a \vee (b \vee c)$
    $a \Rightarrow b \Rightarrow c \rightarrow a \Rightarrow (b \Rightarrow c)$

Terminology:

- For $P \wedge Q$, P and Q are conjucts
- For $P \vee Q$, P and Q are disjuncts
- For $P \Rightarrow Q$, P is the premise/antecedent/hypothesis, and Q is the consequent/conclusion

- The contrapositive of $p \Rightarrow q$ is $\neg q \Rightarrow \neg p$.

Prime Proposition: a proposition that cannot be broken down further (indecomposable).
↳ Eg: "the snow is red"

Compound Proposition: a proposition that contains multiple prime propositions joined by connectives.
↳ Eg: "the snow is red and the grass is green".

Formalizing Natural Language:

$\neg P$: not P, P does not hold, it is not the case

that P, P is false

$P \wedge Q$: P and Q, P but Q, P despite Q, not only P but Q, P while Q, P yet Q, P although Q.

$P \vee Q$: P or Q, P or Q or both, P "and/or" Q, P unless Q

$P \Rightarrow Q$: if P then Q, Q if P, P only if Q, Q when P, P is sufficient for Q, Q is necessary for P, P implies Q.

$P \Leftrightarrow Q$: P if and only if Q, P is necessary and sufficient for Q, P exactly if Q, P is exactly Q.

Example: formalize the following. Let C = "it is cold" and S = "it is snowing".

1) It is cold but it is not snowing
   ↳ $C \wedge \neg S$

2) It is cold only if it is snowing
   ↳ $S \Rightarrow C$

3) It is snowing only if it is cold
   ↳ $C \Rightarrow S$

Example: "it is not the case that I pass the course if I fail the exam". Let F be "I fail the exam" and C be "I pass the course".

↳ $F \Rightarrow \neg C$; "failing the exam implies not passing", or $\neg (F \Rightarrow C)$?

both seem viable, so let's use a truth table!

↓

F	C	$F \Rightarrow \neg C$	$\neg (F \Rightarrow C)$	
0	0	1	0	
0	1	1	1	→ same!
1	0	1	0	
1	1	0	0	→ same!

↓ more correct.

∴, $F \Rightarrow \neg C$.

- Sometimes, logical connectives don't exactly match their meanings in English.
  - ↳ Eg: "the driver hit the cyclist and drove on" is very different to "the driver drove on and hit the cyclist", even though $P \wedge Q == Q \wedge P$. ↳ bc of the temporal (time) element!

Exclusive OR: $(P \vee Q) \wedge \neg (P \wedge Q)$

↳ Eg: "You may take Thursday off or you may take Friday off".

- We have to be careful of the meaning of implication.

↳ especially the "false implies anything" problem – if the antecedent is false, the implication is true.

- **Classical logic is two-valued:** the two possible truth values are T and F.
  - ↳ T denotes the property of a formula being true.
  - ↳ F denotes the property of a formula being false.

↳ the range of the semantic function for propositional logic is the set of truth values: $Tr = \{T, F\}$.

Boolean Valuation (BV): a function from the set of wffs in propositional logic to the set Tr.

↳ given a formula $p \wedge q$, we write $[p \wedge q]$ to mean "the meaning of the formula" in a certain boolean valuation. The [] is a function mapping syntax to its value.

In all boolean valuations:

- $[false] = F$, $[true] = T$
- $[\neg P] = NOT([P])$
- For the connectives,
  - $[P \wedge Q] = [P]$ AND $[Q]$
  - $[P \vee Q] = [P]$ OR $[Q]$
  - $[P \Rightarrow Q] = [P]$ IMP $[Q]$
  - $[P \Leftrightarrow Q] = [P]$ IFF $[Q]$

↳ when describing a boolean valuation, we only need to describe the association of truth values with the proposition symbols.

Example: show the truth value associated with the formula $(p \Rightarrow q) \wedge r$ in the boolean valuation where $p = [T]$, $q = [F]$, and $r = [F]$.

$$
\begin{aligned}
[(P \Rightarrow q) \wedge r] &= [p \Rightarrow q] \text{ AND } [r] \\
&= ([P] \text{ IMP } [q]) \text{ AND } [r] \\
&= (T \text{ IMP } F) \text{ AND } F \\
&= F \text{ AND } F \\
&= F.
\end{aligned}
$$

↳ "can it be true?"

Satisfiability: a formula P is satisfiable if there is a boolean valuation such that $[P] = T$.

↳ "is it always true?"

Tautology: a formula P is a tautology (or valid) if $[P] = T$ for all Boolean valuations.

↳ Eg: $P \vee \neg P$

P	¬P	P v ¬P
0	1	1
1	0	1

↳ since the expression is true for all boolean valuations, it is a tautology (and is also obviously satisfiable).

↳ Eg: $\neg(p \wedge \neg q)$

↳ when $[P] = T$ and $[Q] = T$:

NOT([P] AND NOT([Q]))

NOT(T AND NOT(T))

NOT(T AND F)

NOT(F)

T

since there is a boolean valuation for which the statement is true, it is satisfiable.

when $[P] = T$ and $[Q] = F$:

NOT([P] AND NOT([Q]))

NOT(T AND NOT([F]))

NOT(T AND T)

NOT(T)

F

since there is a boolean valuation for which the statement is false, it is not a tautology.

∴, the statement is satisfiable, but not a tautology!

- When a formula Q is a tautology, we write:

$\models Q$, which is pronounced "entails Q".

↳ $\models$ is a meta-symbol, it's not a part of the syntax of propositional logic - it tells us something about a proposational logic formula.

Logical Implication: a formula P logically implies a formula Q if and only if, for all boolean valuations, if $[P] = T$, then $[Q] = T$.

↳ $P \models Q$, pronounced "P entails Q" or "P logically implies Q"

this can be generalized: a set of formulas $P_1, P_2, \ldots, P_n$ logically imply a formula Q if and only if, for all boolean valuations, if $[P_1] = T$ and $[P_2] = T$ and $\ldots [P_n] = T$ then $[Q] = T$.

↳ $P_1, P_2, \ldots, P_n \models Q$ → also called a valid argument.

↳ equivalent to saying $P_1 \wedge P_2 \wedge \ldots P_n \models Q$, or also $\models P_1 \wedge P_2 \wedge \ldots \wedge P_n \Rightarrow Q$!

Contradiction: a propositional formula A is a contradiction (or a falsehood) if $[A] = F$ for all boolean valuations.

↳ Eg: $P \wedge \neg P$.

Contingent: A contingent formula is one that is neither a tautology nor a contradiction.

↳ ie - has a mixture of Ts and Fs.

Logical Equivalence: two formulas, P and Q, are logically

equivalent if and only if $[P] = [Q]$ in all boolean valuations.

↳ $P \iff Q$, pronounced "P is logically equivalent to Q"

↳ $P \iff Q$ iff $\models P \Leftrightarrow Q$.

Consistency: a collection of formulas is consistent if there is a boolean valuation in which all the formulas are T.

↳ Example: are the following sentences consistent?

1) Sales of houses fall off if interest rates rise
2) Auctioneers aren't happy if sales of houses fall off.
3) Interest rates are rising.
4) Auctioneers are happy.

First, we must formalize these sentences:

- let s be "sales of houses fall off"
- let r be "interest rates rise"
- let h be "auctioneers are happy".

∴ the formulas are:

1) $r \Rightarrow s$
2) $s \Rightarrow \neg h$
3) r
4) h

→

			1	2	3	4
s	r	h	r⇒s	s⇒¬h	r	h
0	0	0	1	1	0	0
0	0	1	1	1	0	1
0	1	0	0	1	1	0
0	1	1	0	1	1	1
1	0	0	1	1	0	0
1	0	1	1	0	0	1
1	1	0	1	1	1	0
1	1	1	1	0	1	1

↳ since there's no boolean valuation for which each sentence is true, the set of formulas is inconsistent!

Example: is this formula satisfiable?:

$$\neg(p \Rightarrow q) \wedge \neg(\neg r \Rightarrow \neg p)$$

since the "terms" are conjuncts, for the formula to be satisfiable, there needs to be some boolean valuation of p, q, and r, for which each term = T.

↳ for $\neg(p \Rightarrow q)$ to be T, $p \Rightarrow q$ must be F. this only occurs if $[P] = 1$ and $[Q] = 0$.

since $[P]$ must be 1, $\neg(\neg r \Rightarrow \neg p)$ can be written as $\neg(\neg r \Rightarrow NOT([T]))$.

↳ $\neg(\neg r \Rightarrow F)$ ∴, $[r] = F$.

we see that when $[p] = T$, $[q] = F$, and $[r] = F$, the formula evaluates to true. ∴, the formula is satisfiable!

Example: is $p \Rightarrow q, q \Rightarrow p \models q \wedge p$ a valid argument?

↳

p	q	p⇒q	q⇒p	q ∧ p
0	0	1	1	0
0	1	1	0	0
1	0	0	1	0
1	1	1	1	1

this row is a counterexample!

no, because $q \wedge p$ is not always 1 when $p \Rightarrow q$ and $q \Rightarrow p$ are both 1.

- We can use truth tables to check these properties, but since a truth table needs 2^n rows, they can be very tedious. Therefore, we can use proof theories to determine these properties.
  - ↳ As long as the proof theory is sound, we can use it to determine tautologies and valid arguments.

Proof Theory: Transformational Proof

Transformational Proof: an algebraic manipulation of formulas in propositional logic following rules.

Formal definition: a transitional proof is a means of determining if two well-formed formulas, P and Q, are logically equivalent by the (repeated) exchange of subformulas of P for logically equivalent subformulas that result in P being transformed into Q.

- Each step must follow a logical law
- The logical laws are expressed using the symbol $\leftrightarrow$.
- Equivalences that we can derive using transformational proof are expressed using the symbol $\leftrightarrow$.

Transformational Proof Rules:

- **Comm - assoc:**
  - ↳ $P \wedge Q \leftrightarrow Q \wedge P \quad \rightarrow P \wedge (Q \wedge R) \leftrightarrow (P \wedge Q) \wedge R$
  - ↳ $P \vee Q \leftrightarrow Q \vee P \quad \rightarrow P \vee (Q \vee R) \leftrightarrow (P \vee Q) \vee R$
  - ↳ $P \Leftrightarrow Q \leftrightarrow Q \Leftrightarrow P$

- **lem:** $P \vee \neg P \leftrightarrow true$
- **contr:** $P \wedge \neg P \leftrightarrow false$
- **impl:** $P \Rightarrow Q \leftrightarrow \neg P \vee Q$
- **idemp:** $P \vee P \leftrightarrow P$, $P \wedge P \leftrightarrow P$
- **neg:** $\neg(\neg P) \leftrightarrow P$
- **simp1:**
  - $P \wedge true \leftrightarrow P$, $P \vee true \leftrightarrow true$
  - $P \wedge false \leftrightarrow false$, $P \vee false \leftrightarrow P$.
- **dm:** $\neg(P \wedge Q) \leftrightarrow \neg P \vee \neg Q$, $\neg(P \vee Q) = \neg P \wedge \neg Q$
- **distr:**
  - $P \vee (Q \wedge R) \leftrightarrow (P \vee Q) \wedge (P \vee R)$
  - $P \wedge (Q \vee R) \leftrightarrow (P \wedge Q) \vee (P \wedge R)$
- **contrapos:** $P \Rightarrow Q \leftrightarrow \neg Q \Rightarrow \neg P$
- **equiv:** $P \Leftrightarrow Q \leftrightarrow (P \Rightarrow Q) \wedge (Q \Rightarrow P)$
- **simp2:**
  - $P \vee (P \wedge Q) \leftrightarrow P$
  - $P \wedge (P \vee Q) \leftrightarrow P$

Example: prove $\neg(\neg p \vee \neg(r \vee s)) \leftrightarrow p \wedge r \wedge s$

1) $\neg(\neg p \vee \neg(r \vee s))$
2) $\neg\neg p \wedge \neg\neg(r \vee s)$ by dm
3) $p \wedge \neg\neg(r \vee s)$ by neg
4) $p \wedge (r \vee s)$ by neg
5) $p \wedge r \vee s$ by distr.

↱ aka prove simp2!

Example: prove that $p \vee p \wedge q \leftrightarrow p$

1) $p \vee p \wedge q$
2) $(p \vee p) \wedge (p \vee q)$ by distr
3) $p \wedge (p \vee q)$ by idemp
4) $(p \vee \text{false}) \wedge (p \vee q)$ by simp1
5) $p \vee (\text{false} \wedge q)$ by distr
6) $p \vee \text{false}$ by simp1
7) p by simp1!

Example: prove $\neg((p \wedge q) \Rightarrow p) \leftrightarrow \text{false}$ ↑ aka, prove that it's a contradiction!

1) $\neg((p \wedge q) \Rightarrow p)$
2) $\neg(\neg(p \wedge q) \vee p)$ by impl
3) $\neg\neg(p \wedge q) \wedge \neg p$ by dm
4) $p \wedge q \wedge \neg p$ by neg
5) $q \wedge \text{false}$ by simp1
6) false by simp1

Example: prove $\neg\text{true} \leftrightarrow \text{false}$

1) $\neg\text{true}$
2) $\neg(p \vee \neg p)$ by lem
3) $\neg p \wedge \neg\neg p$ by dm
4) $\neg p \wedge p$ by neg
5) false by contr!

Example: $p \wedge (\neg(\neg q \wedge \neg p) \vee p) \leftrightarrow p$

1) $p \wedge (\neg(\neg q \wedge \neg p) \vee p)$

2) $p \wedge (\neg\neg q \vee \neg\neg p \vee p)$ by dm
3) $p \wedge (q \vee \neg\neg p \vee p)$ by neg
4) $p \wedge (q \vee p \vee p)$ by neg
5) $p \wedge (q \vee p)$ by idemp
6) p by simp2.

Normal Forms:

- A literal that is a proposition symbol or the negation of a proposition symbol.

- A formula is in conjunctive normal form (CNF) if it is a conjunction of one or more clauses, where a clause is a disjunction of literals or a single literal.

- A formula is in disjunctive normal form (DNF) if it is a disjunction of one or more clauses, where a clause is a conjunction of literals or a single literal.

Example: are the following in CNF or DNF or both?

1) $(p \wedge q \wedge r) \vee (\neg q \wedge \neg r)$: DNF
2) $\neg(p \Rightarrow q) \wedge (r \Rightarrow q)$ neither (no implications allowed)
3) $p \wedge q \vee \neg q \vee \neg r$ DNF
4) $(p \vee q) \wedge \neg r$ CNF
5) $p \vee q$ DNF, but also CNF as it is only one clause
   ↳ think of it as $\wedge$ true?
6) $\neg(p \wedge q)$ neither, negation must be on the literal
7) $\neg p$ both
8) $p \wedge r \vee p \wedge (q \vee \neg r)$ neither

Converting Formulas to CNF:

- Remove all $\Rightarrow$ and $\Leftrightarrow$ using impl/equiv laws
- If the formula contains any negated compound subformulas, either remove the negation by using the negation law or use DM to push the negation in.
- Once a formula with no negated compound subformulas is found, use the following distributivity laws:
  - ↳ $A \vee (B \wedge C) \leftrightarrow (A \vee B) \wedge (A \vee C)$
  - ↳ $(A \wedge B) \vee C \leftrightarrow (A \vee C) \wedge (B \vee C)$
- Simplify so there are no repeated literals in a clause and no two clauses with the same literals.
  - ↳ note: true/false may only appear if they're the entire formula.

↳ Example: find the conjunctive normal form of $\neg((p \vee \neg q) \wedge \neg r)$

$\neg((p \vee \neg q) \wedge \neg r)$

1) $\neg(p \vee \neg q) \vee \neg\neg r$ by dm
2) $\neg(p \vee \neg q) \vee r$ by neg
3) $\neg p \wedge \neg\neg q \vee r$ by dm
4) $\neg p \wedge q \vee r$ by neg
5) $(\neg p \vee r) \wedge (q \vee r)$ by distr.

∴, the CNF is $(\neg p \vee r) \wedge (q \vee r)$

Converting Formulas to DNF:

- Remove all $\Rightarrow$ and $\Leftrightarrow$ using impl/equiv laws

- If the formula contains any negated compound subformulas, either remove the negation by using the negation law or use DM to push the negation in.
- Once a formula with no negated compound subformulas is found, use the following distributivity laws:
  - ↳ $A \wedge (B \vee C) \leftrightarrow (A \wedge B) \vee (A \wedge C)$
  - ↳ $(A \vee B) \wedge C \leftrightarrow (A \wedge C) \vee (B \wedge C)$
- Simplify so there are no repeated literals in a clause and no two clauses with the same literals.
  - ↳ note: true/false may only appear if they're the entire formula.

- If a formula is in DNF, it's very easy to find a satisfying assignment, as we just need to satisfy one of the clauses.

• Natural Deduction:

- Review: Argument → a collection of formulas, one of which, referred to as the conclusion, is justified by the others, referred to as the premise(s).
  - ↳ an argument is valid if in all Boolean valuations where the premises have the value T, the conclusion also has the truth value T.
    - ↳ note: no premises implies the conclusion is a tautology ($\models Q$).
    - ↳ generally, to prove an argument is invalid, we have to show the boolean valuation for which the

argument does not hold.

Natural Deduction is a proof theory for proving the validity of an argument in propositional logic.
↳ it's a collection of rules, called inference rules, each of which allows us to infer new formulas from given formulas.
↳ a form of forward proof

- **Inference Rules:** a primitive valid argument form. Each inference rule enables the elimination or the introduction of a logical connective.

↳ most inference rules have names that consist of:
  - a logical connective
  - a letter:
    ↳ "i" for introducing a formula that contains the connective
    ↳ "e" for eliminating (aka using) a formula that contains the connective

The formulas used in Natural Deduction are like "mini-arguments". Eg: and_i : $P, Q \vdash P \wedge Q$
and_e: $P \wedge Q \vdash P$ / $P \wedge Q \vdash Q$

Summary of general Natural Deduction Rules:

Introduction (i):

↳ $P, Q \vdash P \wedge Q$ → and_i

$P \vdash \neg\neg P$ → not_not_i

$P \vdash P \vee Q$ → or_i

$P \Rightarrow Q, Q \Rightarrow P \vdash P \Leftrightarrow Q$ → iff_i

Elimination (e):

↳ $P \wedge Q \vdash P$ → and_e

$P \Rightarrow Q, P \vdash Q$ → imp_e

$\neg\neg P \vdash P$ → not-not_e

$P, \neg P \vdash Q$ → not_e

$P \Leftrightarrow Q \vdash P \Rightarrow Q$ → iff_e

Example: $a, \neg\neg(b \wedge c) \vdash \neg\neg a \wedge c$

1) a premise
2) $\neg\neg(b \wedge c)$ premise
3) $\neg\neg a$ by not_not_i on 1
4) $b \wedge c$ by not_not_e on 2
5) c by and_e on 4
6) $\neg\neg a \wedge c$ by and_i on 3, 5

Example: $a, a \Rightarrow b, b \Leftrightarrow c \vdash b \wedge c$

1) a premise
2) $a \Rightarrow b$ premise
3) $b \Leftrightarrow c$ premise
4) b by imp_e on 1, 2
5) $b \Rightarrow c$ by iff_e on 3

6) c by imp_e on 4,5

7) $b \wedge c$ by and_i on 4,6

↗ lets us skip one step using iff_e

Additional Rule: $P \Leftrightarrow Q, P \vdash Q$ → iff_mp

Example: $\neg\neg a \Leftrightarrow b \wedge c, a \vdash b \vee s$

1) $\neg\neg a \Leftrightarrow b \wedge c$ premise

2) a premise

3) $\neg\neg a$ by not_not_i on 2

4) $b \wedge c$ by iff_mp on 1,3

5) b by and_e on 4

6) $b \vee s$ by or_i on 5

Example: $\neg a \Leftrightarrow \neg b, \neg b \wedge a \vdash c$ ↗ must be false implies anything! ∴ premises must be a contradiction

1) $\neg a \Leftrightarrow \neg b$ premise

2) $\neg b \wedge a$ premise

3) $\neg b$ by and_e on 2

4) $\neg a$ by iff_mp on 1,3

5) a by and_e on 2

6) c by not_e on 4,5

Subordinate Proofs (subproofs): start by choosing a formula that we assume is true within the subproof. Then we see what we can prove using our assumption.

↳ used by imp_i, raa, cases (defined later)

assume R {
    ... ⊢ R ⇒ Q → imp_i
    Q
}

Example: b ⇒ c ⊢ ¬c ⇒ ¬b

1) b ⇒ c premise
2) assume ¬c {
    3) ¬b by imp_e on 1,2
}
4) ¬c ⇒ ¬b by imp_i on 2-3.

Example: a ⇒ (b ⇒ c), b ⊢ a ⇒ c

1) a ⇒ (b ⇒ c) premise
2) b premise
3) assume a {
    4) b ⇒ c by imp_e on 1,3
    5) c by imp_e on 2,4
}
6) a ⇒ c by imp_i on 3-5.

↳ no premises, so must be tautology!

Example: ⊢ (c ⇒ d) ⇒ ((¬c ⇒ ¬b) ⇒ (b ⇒ d))

1) Assume c ⇒ d {
    2) Assume ¬c ⇒ ¬b {
        3) Assume b {

4) ¬¬c by imp_e on 2,3
5) c by not_not_e on 4
6) d by imp_e on 1, 5
}
7) b ⇒ d by imp_i on 3-6
}
8) (¬c ⇒ ¬b) ⇒ (b ⇒ d) by imp_i on 2-7
}
9) (c ⇒ d) ⇒ ((¬c ⇒ ¬b) ⇒ (b ⇒ d)) by imp_i on 1-8.

Rule: Proof by Contradiction

disprove R {
...
false
}

$\vdash \neg R \rightarrow$ raa

Example: $b \Rightarrow c,\ b \Rightarrow \neg c \vdash \neg b$

1) b ⇒ c premise
2) b ⇒ ¬c by premise
3) disprove b {
4) c by imp_e on 1, 3
5) ¬c by imp_e on 2, 3
6) false by not_e on 4, 5
}
7) ¬b by raa on 3-6.

Example: $b \vdash a \Rightarrow b$

1) b premise
2) assume a {
   3) $b \wedge a$ by and_i on 1,2
   4) a by and_e on 3
}
5) $a \Rightarrow b$ by imp_i on 2

→ need to get b, our premise, into the imp_i !

Rule: case analysis:

P V R
case P {
  ...
  Q
}
case R {
  ...
  Q
}

$\vdash Q$ → cases

Example: I get peanut butter or I get jam. If I get peanut butter then I make a sandwich. If I get jam then I make a sandwich. Therefore, I make a sandwich.

↳ p = "I get peanut butter"
j = "I get jam"
s = "I make a sandwich"

↳ $\therefore, p \vee j, j \Rightarrow s, p \Rightarrow s \vdash s.$

1) p ∨ j premise
2) p ⇒ s premise
3) j ⇒ s premise
4) case p {
    5) s by imp.e on 2, 4
}
6) case j {
    7) s by imp.e on 3, 6
}
8) s by cases on 1, 4-5, 6-7.

Derived Rules:

- P ∨ Q, ¬P ⊢ Q → or.e
- ⊢ p ∨ ¬p → lem

Example: a, ¬b ⇔ a, c ∨ b, c ⇒ b ⊢ false

1) a premise
2) ¬b ⇔ a premise
3) c ∨ b premise
4) c ⇒ b premise
5) ¬b by iff_mp on 1, 2
6) ¬c by imp.e on 4, 5
7) b by or_e
8) false by not_e on 5, 7

Example: ⊢ p ∨ ¬p

1) disprove $\neg(p \vee \neg p)$ {
    2) disprove p {
        3) $p \vee \neg p$ by or-i on 2
        4) false by not-e on 3
    }
    5) $\neg p$ by raa on 2-4
    6) $\neg p \vee p$ by or-i on 5
    7) false by not-e on 6
}
8) $p \vee \neg p$ by raa on 1-7

- Natural deduction for propositional logic is sound and complete!

Example: $\neg a \vee \neg b \vdash \neg(a \wedge b)$

1) $\neg a \vee \neg b$ premise
2) disprove $(a \wedge b)$ {
    3) a by and-e on 2
    4) b by and-e on 2
    5) $\neg\neg b$ by not-not-i on 4
    6) $\neg a$ by or-e on 1,5
    7) false by not-e on 3,6
}
$\neg(a \wedge b)$ by raa.

Example: $\neg(a \vee b) \vdash \neg a \wedge \neg b$

1) ¬(a ∨ b)
2) disprove a {
    3) a ∨ b by or_i on 2
    4) false by not_e on 1,3
}
5) ¬a by raa on 2-4
6) disprove b {
    7) a ∨ b by or_i on 6
    8) false by not_e on 1,7
}
9) ¬b by raa on 6-8
10) ¬a ∧ ¬b by and_i on 5, 9

Example: b ⇒ c ⊢ ¬b ∨ c

1) b ⇒ c premise
2) b ∨ ¬b by lem
3) case b {
    4) c by imp_e on 1,3
    5) ¬b ∨ c by or_i on 4
}
6) case ¬b {
    7) ¬b ∨ c by or_i on 6
}
8) ¬b ∨ c by cases on 2,3-5, 6-7

- Semantic Tableaux

↳ a tree representing all ways the conjunction of

the formulas at the root of the tree can be true.

- Every formula on a path from the root to a leaf must be true in a boolean valuation that makes the conjunction of the formulas at the root true.

In each step in a Semantic Tableaux proof, we either:

- use a semantic tableaux rule to decompose one compound formula and add new formula(s) to that branch. The rules are based on the outermost propositional connectives used in the compound formula.
- OR, close a branch because it contains contradictory formulas. A branch is a path from the root to the leaf. A closed branch means that there is no boolean valuation that can make all the formulas on this path T.

- With each rule, we keep breaking down the ways that the formulas at the root can be T.

- a branch is closed if P and ¬P both appear on the path from the root of the tree to the leaf of the branch (ie - there's a contradiction on the branch). → Note: P can be a compound formula.

- the conjunction of the formulas on a closed branch is a contradiction.

Summary of Semantic Tableaux Rules:

- 1) $P \wedge Q$

  by and_nb on 1

  2) P

  3) Q

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

  by not_imp_nb on 1

  2) P

  3) $\neg Q$

- 1) $P \vee Q$

  by or_br on 1

  {

  2) P

  }

  {

  3) Q

  }

- 1) $\neg (P \wedge Q)$

  by not_and_br on 1

  {

  2) $\neg P$

  }

  {

  3) $\neg Q$

  }

- 1) $P \Rightarrow Q$

  by imp_br on 1

  {

  2) $\neg P$

  }

  {

  3) Q

  }

- 1) $P \Leftrightarrow Q$

  by iff_br on 1

  {

  2) $P \wedge Q$

  }

  {

  3) $\neg P \wedge \neg Q$

  }

- $\neg (P \Leftrightarrow Q)$

  by not_iff_br

  {

  2) $P \wedge \neg Q$

  }

  {

  3) $\neg P \wedge Q$

  }

- 1) $\neg\neg P$

  by not_not_nb on 1

  2) P

- 1) $\neg(P \vee Q)$

  by not_or_nb on 1

  2) $\neg P$

  3) $\neg Q$

Example: prove $\neg(a \Rightarrow b), \neg a \vee b$ is an inconsistent set of formulas.

$\neg(a \Rightarrow b), \neg a \vee b \vdash$ false

1) $\neg(a \Rightarrow b)$

2) $\neg a \vee b$

by not_imp_nb on 1

3) a

4) $\neg b$

by or_br on 2 {

  5) $\neg a$

  closed on 3, 5

} {

  6) b

  closed on 4, 6

}

all branches closed shows that this is inconsistent!

Example: prove $r \Rightarrow s$, $s \Rightarrow \neg h$, r, h is an inconsistent set of formulas.

1) $r \Rightarrow s$

2) $s \Rightarrow \neg h$

3) r

4) h

by imp-br on 1 {

5) $\neg r$

closed on 3, 5

} {

6) s

by imp-br on 2 {

7) $\neg s$

closed on 6, 7

} {

8) $\neg h$

closed on 4, 8

}

}

all branches closed!

Example: $b \wedge c,\ d,\ \neg(c \wedge d) \vdash \text{false}$

1) $b \wedge c$

2) d

3) $\neg(c \wedge d)$

by add-nb on 1

4) b

5) c

by not-and-br on 3 {

6) $\neg c$

closed on 5, 6

}{

7) ¬d

closed on 2, 7

}

- Generally, it's faster to apply the non-branching rules first.

- For an argument to be invalid,
  - ↳ there has to be a Boolean valuation in which the premises are T and the conclusion is F.
  - ↳ there has to be a Boolean valuation in which the premises are T and the negation of the conclusion is T
  - ↳ $P_1, P_2, \ldots, P_n, \neg Q$ is a consistent set of formulas

∴, to show an argument is valid, we must show that $P_1, P_2, \ldots, P_n, \neg Q$ is an inconsistent set of formulas.
- ↳ put the premises and the negation of the conclusion at the root. If we close all the branches, then this set of formulas is inconsistent, and therefore the argument is valid.

Example: $b \vee c \vdash c \vee b$

1) $b \vee c$

2) $\neg(c \vee b)$ → negation of the conclusion:

by not_or_nb on 2

3) ¬c

4) ¬b

by or-br on 1 {

5) b

closed on 4, 5

} {

6) c

closed on 3, 6

}

Predicate Logic

constant: a symbol denoting a particular value

predicate: a symbol denoting an attribute/property of a value or denoting a relationship between two or more values.

↳ something that is T/F when applied to arguments.

unary predicate: predicate that takes only one single argument.

binary predicate: predicate that takes 2 arguments.

n-ary predicate: predicate that takes n arguments.

Generally, constants are the nouns and predicates are the adjectives / adverbs / verbs.

Example: unary predicates

a) "Ravi is an adult"

↳ adult(x) means x is an adult

adult(Ravi)

b) "the Earth is round"

↳ round(x) means x is round

round(Earth)

Example: binary predicates

a) "Barb plays the piano"

↳ plays(x, y) means x plays y

plays(Barb, piano)

b) "10 is greater than 5"

↳ "10" and "5" are constants, > is an infix predicate

10 > 5

Example: 3-ary predicate: "Waterloo is between Toronto and London"

↳ between(a, b, c) means a is between b and c.

→ between(Waterloo, Toronto, London)

Example: "John is happy if he visits Vancouver"

↳ visits(x, y) means x visits y and happy(x) means x is happy → visits(John, Vancouver) ⇒ happy(John)

Universal Quantification: $\forall$ ("for all")

Existential Quantification: $\exists$ ("there exists")

Example:

a) "Everything likes Fridays"

↳ likes(x, y) means that x likes y

$\forall x \cdot \text{likes}(x, \text{Fridays})$

b) "Something likes Fridays"

↳ likes(x, y) means that x likes y

$\exists x \cdot \text{likes}(x, \text{Fridays})$

c) "Everything is a tall adult"

↳ tall(x) means x is tall and adult(x) means x is an adult

$\forall x \cdot \text{tall}(x) \wedge \text{adult}(x)$

d) "Something is happy or hungry"

↳ happy(x) means x is happy and hungry(x) means x is hungry

$\exists x \cdot \text{happy}(x) \vee \text{hungry}(x)$

e) "Every child likes Mickey Mouse"

↳ child(x) means x is a child and likes(x, y) means x likes y.

$\forall x \cdot \text{child}(x) \Rightarrow \text{likes}(x, \text{Mickey Mouse})$

Functions:

Example: "Mary's age is less than 20"

↳ one way is $\exists x \cdot age(Mary, x) \wedge x < 20$, but this does not specify that Mary has exactly one age.

∴ → $age(Mary) < 20$, where $age(x)$ is a function whose return value is the age of x.

- we can also have n-ary functions.
- a function must be defined for all inputs.

Predicate Logic Syntax: ↗ for UNTYPED predicate logic!

1) **Alphabet:**
   - constants (eg: John, true)
   - variables (eg: x, y)
   - function symbols (unary, binary, n-ary)
   - predicate symbols (unary, binary, n-ary)
   - logical connectives ($\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$, $\neg$)
   - quantifiers ($\forall$, $\exists$)
   - punctuation ($\cdot$)
   - brackets ()

2) **Terms** → terms describe values
   - Every constant is a term
   - Every variable is a term

- If $t_1, \ldots, t_n$ are terms and g is an n-ary function symbol, then $g(t_1, \ldots, t_n)$ is a term.
- Nothing else is a term!!

3) well-formed formulas (wff): → have truth values
- proposition symbols and the constants true and false are wff. These are atomic formulas.
- if the t_i's are terms and P is an n-ary predicate symbol, then $P(t_1, \ldots, t_n)$ is a wff. These are also atomic formulas.
- if P and Q are wffs, then so are $(\neg P)$, $(P \wedge Q)$, $(P \vee Q)$, $(P \Rightarrow Q)$, and $(P \Leftrightarrow Q)$.
- if P is a wff and x is a variable, then $\forall x \cdot P$ and $\exists x \cdot P$ are also wffs.
- Nothing else is a wff!!

- Scope of a Quantifier: the subformula over which the quantifier applies in the given formula.
↳ extends to the right end of the formula, unless an unmatched end bracket stops it.

- A variable is bound to the closest quantifier to the left of its name whose scope it is within.

- An occurance of a variable is bound if it falls within the scope of a quantifier for that variable.

↳ Else, the occurance of the variable is free.

- A wff is closed if it contains no free variables.

Example: "all students who like SE also like logic"
↳ $\forall x \cdot Student(x) \wedge likes(x, SE) \Rightarrow likes(x, logic)$.

Example: "all rich actors collect some valuables"
↳ $\forall x \cdot rich(x) \wedge actor(x) \Rightarrow \exists y \cdot collect(x,y) \wedge valuable(y)$

- In untyped predicate logic, we classify symbols as constants, predicates, or functions.

- In typed predicate logic, we will instead associate a type expression with every symbol.

- A function has a compound type expression:
  ↳ $g : T1 \cdot T2 \cdot \ldots \cdot Tn \rightarrow$ Return Type

- Type Environment: a set of basic types and types for all the constants and the functions in the formula.

- In typed predicate logic, every quantified variable must have a type.

- To be a wff in typed predicate logic, the formula must also be well-typed → basically, all the type declarations, inputs, and outputs must match up.

- **Type inference:** process of figuring out a type environment.

- **Most General Typing:** using the maximum number of basic types such that the formula is well-typed.

Example: $\exists x, y \cdot q(f(y)) \wedge q(g(x))$

↳ let $x : T1$ and $y : T2$.

We see that q must output a boolean, but we don't know it's input type. Let's assume it takes a type $T3$.

Then: $q : T3 \to bool$

$f : T1 \to T3$

$g : T2 \to T3$

$\therefore \exists x : T1, y : T2 \cdot q(f(y)) \wedge q(g(x))$