

# Lecture 1 - 5<sup>th</sup> 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 program 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
  - $\Leftrightarrow$  : logical equivalence
  - $\vdash$  : proves  $\rightarrow P_1, P_2, \dots, P_n \vdash Q$  means that from  $P_1, P_2, \dots, P_n$ , we can prove  $Q$  using a proof theory.
- ↗ NOT a part of wff, but a meta-level symbol.

**Proof Theories:** methods that perform mechanical manipulations on strings of symbols.

↳ based on pattern matching!

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

**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

$P_1, P_2, \dots, P_n \vdash Q$  (we have a proof), then

$P_1, P_2, \dots, P_n \models Q$  (valid)

**Completeness:** a proof theory is complete if

whenever  $P_1, P_2, \dots, P_n \models Q$  (valid) then  $P_1, P_2, \dots, P_n \vdash Q$  (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  $\neg$  having the highest precedence and  $\Leftrightarrow$  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 conjuncts
- 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  
 $\hookrightarrow c \wedge \neg s$

2) It is cold only if it is snowing  
 $\hookrightarrow s \Rightarrow c$

3) It is snowing only if it is cold  
 $\hookrightarrow c \Rightarrow s$

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:  $\text{Tr} = \{\text{T}, \text{F}\}$ .

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,
  - $[p \wedge q] = [p] \text{ AND } [q]$
  - $[p \vee q] = [p] \text{ OR } [q]$
  - $[p \Rightarrow q] = [p] \text{ IMP } [q]$
  - $[p \Leftrightarrow q] = [p] \text{ IFF } [q]$

↳ 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 formulae  $(p \Rightarrow q) \wedge r$  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:  $P \vee \neg P$

$P$	$\neg P$	$P \vee \neg P$
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  $[Q] = T$ :

$\text{NOT}([P]) \text{ AND } \text{NOT}([Q])$

$\text{NOT}(T \text{ AND } \text{NOT}(T))$

$\text{NOT}(T \text{ AND } F)$

$\text{NOT}(F)$

T

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

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

$\text{NOT}([P]) \text{ AND } \text{NOT}([Q])$

$\text{NOT}(T \text{ AND } \text{NOT}(F))$

$\text{NOT}(T \text{ AND } T)$

$\text{NOT}(T)$

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 propositional 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  $P_1, P_2, \dots, P_n$  logically imply a formula  $Q$  if and only if, for all boolean valuations, if  $[P_1] = T$  and  $[P_2] = T$  and ...  $[P_n] = T$  then  $[Q] = T$ .

↳  $P_1, P_2, \dots, P_n \models Q \rightarrow$  also called a valid argument.

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

**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.

↳ i.e. - 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.

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

↳  $P \Leftrightarrow Q$  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:

$$1) r \Rightarrow s$$

$$2) s \Rightarrow \neg h$$

$$3) r$$

$$4) h$$

S	r	h	$r \Rightarrow s$	$s \Rightarrow \neg h$	r	h
0	0	0	1	1	0	0
0	0	1	1	0	0	1
0	1	0	0	0	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	1	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  $\neg(\neg r \Rightarrow \text{NOT}([T]))$ .

$$\neg(\neg r \Rightarrow F) \quad \therefore, [r] = F.$$

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

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

P	q	$p \Rightarrow q$	$q \Rightarrow p$	$q \wedge p$	
0	0	1	1	0	<i>this row is a counterexample!</i>
0	1	1	0	0	no, because $q \wedge p$
1	0	0	1	0	is not always 1
1	1	1	1	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 transformational 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:

$$\hookrightarrow P \wedge Q \leftrightarrow Q \wedge P \rightarrow P \wedge (Q \wedge R) \leftrightarrow (P \wedge Q) \wedge R$$

$$\hookrightarrow P \vee Q \leftrightarrow Q \vee P \rightarrow P \vee (Q \vee R) \leftrightarrow (P \vee Q) \vee R$$

$$\hookrightarrow P \leftrightarrow Q \leftrightarrow Q \leftrightarrow P$$

- lem:  $P \vee \neg P \leftrightarrow \text{true}$
- Contr:  $P \wedge \neg P \leftrightarrow \text{false}$
- 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:
  - $\hookrightarrow P \wedge \text{true} \leftrightarrow P, P \vee \text{true} \leftrightarrow \text{true}$
  - $\hookrightarrow P \wedge \text{false} \leftrightarrow \text{false}, P \vee \text{false} \leftrightarrow P.$
- dm:  $\neg(P \wedge Q) \leftrightarrow \neg P \vee \neg Q, \neg(P \vee Q) = \neg P \wedge \neg Q$
- distr:
  - $\hookrightarrow P \vee (Q \wedge R) \leftrightarrow (P \vee Q) \wedge (P \vee R)$
  - $\hookrightarrow P \wedge (Q \vee R) \leftrightarrow (P \wedge Q) \vee (P \wedge R)$
- Contrapos:  $P \Rightarrow Q \leftrightarrow \neg Q \Rightarrow \neg P$
- equiv:  $P \Leftrightarrow Q \leftrightarrow (P \Rightarrow Q) \wedge (Q \Rightarrow P)$
- Simp2:
  - $\hookrightarrow P \vee (P \wedge Q) \leftrightarrow P$
  - $\hookrightarrow P \wedge (P \vee Q) \leftrightarrow P$

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)  $p \wedge \neg(r \vee s)$  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 simp 1
- 5)  $p \vee (\text{false} \wedge q)$  by distr
- 6)  $p \vee \text{false}$  by simp 1
- 7)  $p$  by simp 1!

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 simp 1
- 6) false by simp 1

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)  $p \wedge (\neg q \vee \neg p \vee p)$  by idm
- 3)  $p \wedge (q \vee \neg p \vee p)$  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 → think of it as  $\wedge$  true?
- 5)  $p \vee q$  DNF, but also CNF as it is only one clause
- 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.

$\therefore$  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  $\rightarrow$  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):

$\vdash P, Q \vdash P \wedge Q$

$\rightarrow \text{and-i}$

$P \vdash \neg\neg P$

$\rightarrow \text{not-not-i}$

$P \vdash P \vee Q$

$\rightarrow \text{or-i}$

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

$\rightarrow \text{iff-i}$

### Elimination (e):

$\vdash P \wedge Q \vdash P$	$\rightarrow \text{and-e}$
$P \Rightarrow Q, P \vdash Q$	$\rightarrow \text{imp-e}$
$\neg\neg P \vdash P$	$\rightarrow \text{not-not-e}$
$P, \neg P \vdash Q$	$\rightarrow \text{not-e}$
$P \Leftrightarrow Q \vdash P \Rightarrow Q$	$\rightarrow \text{iff-e}$

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:  $P \Leftrightarrow Q, P \vdash Q \rightarrow \text{iff-mp}$

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

must be false implies anything!  
∴ premises must be a contradiction

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

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  $R \{$

...

$\vdash R \Rightarrow Q$

$\rightarrow$  imp-i

$Q$

}

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

1)  $b \Rightarrow c$  premise

2) assume  $\neg c \{$

3)  $\neg b$  by imp-e on 1,2

}

4)  $\neg c \Rightarrow \neg b$  by imp-i on 2-3.

Example:  $a \Rightarrow (b \Rightarrow c), b \vdash a \Rightarrow c$

1)  $a \Rightarrow (b \Rightarrow c)$  premise

2)  $b$  premise

3) assume  $a \{$

4)  $b \Rightarrow c$  by imp-e on 1,3

5)  $c$  by imp-e on 2,4

}

6)  $a \Rightarrow c$  by imp-i on 3-5.

→ no premises, so must be tautology!

Example:  $\vdash (c \Rightarrow d) \Rightarrow ((\neg c \Rightarrow \neg b) \Rightarrow (b \Rightarrow d))$

1) Assume  $c \Rightarrow d \{$

2) Assume  $\neg c \Rightarrow \neg b \{$

3) Assume  $b \{$

4)  $\neg c$  by imp-e on 2,3

5)  $c$  by not-not-e on 4

6)  $d$  by imp-e on 1,5

}

7)  $b \Rightarrow d$  by imp-i on 3-6

}

8)  $(\neg c \Rightarrow \neg b) \Rightarrow (b \Rightarrow d)$  by imp-i on 2-7

}

9)  $(c \Rightarrow d) \Rightarrow ((\neg c \Rightarrow \neg b) \Rightarrow (b \Rightarrow d))$  by imp-i on 1-8.

## Rule: Proof by Contradiction

disprove R {

...

$\vdash \neg R \rightarrow \text{raa}$

false

}

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

1)  $b \Rightarrow c$  premise

2)  $b \Rightarrow \neg c$  by premise

3) disprove  $b$  {

4)  $c$  by imp-e on 1,3

5)  $\neg c$  by imp-e on 2,3

6) false by not-e on 4,5

}

7)  $\neg b$  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  $\rightarrow$  need to get  $b$ , our premise, into the imp-i!
- 4)  $a$  by and-e on 3
- 5)  $a \Rightarrow b$  by imp-i on 2

Rule: Case analysis:

PVR

case P {

$\ddots$   
    Q

  }

$\vdash Q$

$\rightarrow$  Cases

case R {

$\ddots$   
    Q

  }

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.

$\hookrightarrow p = "I \text{ get peanut butter}"$

$j = "I \text{ get jam}"$

$s = "I \text{ make a sandwich}"$

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

- 1)  $P \vee j$  premise
  - 2)  $P \Rightarrow s$  premise
  - 3)  $j \Rightarrow s$  premise
  - 4) case  $P \notin$
  - 5)  $s$  by imp-e on 2, 4
- }
- 6) case  $j \in$
  - 7)  $s$  by imp-e on 3, 6
- }
- 8)  $s$  by cases on 1, 4-5, 6-7.

### Derived Rules:

- $P \vee Q, \neg P \vdash Q \rightarrow \text{or-e}$
- $\vdash P \vee \neg P \rightarrow \text{lem}$

Example:  $\alpha, \neg b \Leftrightarrow \alpha, c \vee b, c \Rightarrow b \vdash \text{false}$

- 1)  $\alpha$  premise
- 2)  $\neg b \Leftrightarrow \alpha$  premise
- 3)  $c \vee b$  premise
- 4)  $c \Rightarrow b$  premise
- 5)  $\neg b$  by iff-mp on 1, 2
- 6)  $\neg c$  by imp-e on 4, 5
- 7)  $b$  by or-e
- 8) false by not-e on 5, 7

Example:  $\vdash p \vee \neg p$

- 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)  $\neg(a \vee b)$

2) disprove  $a \{$

3)  $a \vee b$  by or-i on 2

4) false by not-e on 1,3

}

5)  $\neg a$  by raa on 2-4

6) disprove  $b \{$

7)  $a \vee b$  by or-i on 6

8) false by not-e on 1,7

}

9)  $\neg b$  by raa on 6-8

10)  $\neg a \wedge \neg b$  by and-i on 5, 9

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

1)  $b \Rightarrow c$  premise

2)  $b \vee \neg b$  by lem

3) case  $b \{$

4)  $c$  by imp-e on 1,3

5)  $\neg b \vee c$  by or-i on 4

}

6) case  $\neg b \{$

7)  $\neg b \vee c$  by or-i on 6

}

8)  $\neg b \vee c$  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  $\neg P$  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)  $P \Leftrightarrow Q$

by iff-br on 1  
{

2)  $P \wedge Q$

}

{

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

}

• 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)  $Q$

}

•  $\neg(P \Leftrightarrow Q)$

by not-iff-br  
{

2)  $P \wedge \neg Q$

}

{

3)  $\neg P \wedge Q$

}

• 1)  $\neg\neg P$

• 1)  $\neg(P \vee Q)$

by not-not-nb on 1

by not-or-nb on 1

2)  $P$

2)  $\rightarrow P$

3)  $\neg Q$

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

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

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

3 {

6)  $s$

by imp-br on 2 {

7)  $\neg s$

closed on 6, 7

3 {

8)  $\neg h$

closed on 4, 8

3

3

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

3 {

7)  $\rightarrow d$

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

↳  $P_1, P_2, \dots, P_n, \neg Q$  is a consistent set of formulas

- ∴, to show an argument is valid, we must show that  $P_1, P_2, \dots, P_n, \neg Q$  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)  $\neg c$

4)  $\neg b$

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"

↳  $\text{adult}(x)$  means  $x$  is an adult  
 $\text{adult}(\text{Ravi})$

b) "the Earth is round"

↳  $\text{round}(x)$  means  $x$  is round  
 $\text{round}(\text{Earth})$

## Example: binary predicates

a) "Barb plays the piano"

↳  $\text{plays}(x, y)$  means  $x$  plays  $y$   
 $\text{plays}(\text{Barb}, \text{piano})$

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"

↳  $\text{between}(a, b, c)$  means  $a$  is between  $b$  and  $c$ .

→  $\text{between}(\text{Waterloo}, \text{Toronto}, \text{London})$

Example: "John is happy if he visits Vancouver"

↳  $\text{visits}(x, y)$  means  $x$  visits  $y$  and  $\text{happy}(x)$  means  
 $x$  is happy →  $\text{visits}(\text{John}, \text{Vancouver}) \Rightarrow \text{happy}(\text{John})$

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  $\exists x \cdot \text{age}(\text{Mary}, x) \wedge x < 20$ ,  
but this does not specify that Mary has exactly  
one age.

$\therefore \rightarrow \text{age}(\text{Mary}) < 20$ , where  $\text{age}(x)$  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:  $\rightarrow$  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 $\rightarrow$ terms describe values

- Every constant is a term
- Every variable is a term

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

### 3) well-formed formulas (wff): $\rightarrow$ 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  $P(t_1, \dots, t_n)$  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.

$\hookrightarrow$  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 occurrence of a variable is bound if it falls within the scope of a quantifier for that variable.

$\hookrightarrow$  Else, the occurrence of the variable is free.

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

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

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

- 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:  
↳  $g : T_1 \cdot T_2 \cdot \dots \cdot T_n \rightarrow \text{Return Type}$
- 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  $x : T_1$  and  $y : T_2$ .

We see that  $q$  must output a boolean, but we don't know its input type. Let's assume it takes a type  $T_3$ .

Then:  $q : T_3 \rightarrow \text{bool}$

$f : T_1 \rightarrow T_3$

$g : T_2 \rightarrow T_3$

$\therefore \exists x : T_1, y : T_2 \cdot q(f(y)) \wedge q(g(x))$

• We describe the meaning of a formula in an interpretation.

↳ an interpretation consists of meanings for all the symbols over a universe of values.

• We use " $::=$ " to mean "is defined as"!

Example interpretation:

$[\text{univ}] := \{0, 1\}$

$[c : \text{univ}] := 0$

$[d : \text{univ}] := 1$        $\begin{array}{c} \text{input} \\ \downarrow \\ 1 \end{array}$        $\begin{array}{c} \text{output} \\ \uparrow \\ 1 \end{array}$   
 $[g : \text{univ} \rightarrow \text{univ}] := \{(0, 1), (1, 0)\}$   
 $[p : \text{univ} \cdot \text{univ} \rightarrow \text{bool}] := \{(0, 0, F), (0, 1, F), (1, 0, F), (1, 1, F)\}$   
 $[q : \text{univ} \cdot \text{univ} \rightarrow \text{bool}] := \{(0, 0, T), (0, 1, F), (1, 0, F), (1, 1, T)\}$   
 $\downarrow$

determine the meaning of the formula:  $p(g(c), d) \wedge q(c, d)$

$$\begin{aligned}
& [p(g(c), d) \wedge q(c, d)] \\
&= [p(g(c), d)] \text{ AND } [q(c, d)] \\
&= [p]([g](c), [d]) \text{ AND } [q]([c], [d]) \\
&= [p]([g](0), 1) \text{ AND } [q](0, 1) \\
&= [p](1, 1) \text{ AND } [q](0, 1) \\
&= F \text{ AND } F \\
&= F
\end{aligned}$$

- An interpretation in untyped predicate logic includes:
    - 1) a non-empty universe
    - 2) every constant maps to a value in the universe.
    - 3) each function of arity  $m$  maps to a set of tuples of arity  $m+1$  where the final element of each tuple is unique for every combination of  $m$  values.
    - 4) each predicate of arity  $m$  maps to a set of tuples of arity  $m+1$  where the final element of each tuple is of type bool and is unique for every combination of  $m$  values.
- 75-113

- $^d$  notation: where  $d$  is an element of the universe, in the middle of a formula  $[...]$ ,  $^d$  means "in this spot"

was a variable that has value d.

↳  $\hat{d}$  should only be used within [...].

Example: find an interpretation in which the following formula is T:  $\forall x \cdot \exists y \cdot p(x) \Leftrightarrow \neg p(y)$

$$[\text{univ}] := \{d_1, d_2\}$$

$$[p: \text{univ} \rightarrow \text{bool}] := \{(d_1, ?), (d_2, ?)\}.$$

$$\hookrightarrow [\forall x \cdot \exists y \cdot p(x) \Leftrightarrow \neg p(y)] \xrightarrow{\text{expanding } \forall x}$$

$$= [\exists y \cdot p(\hat{d}_1) \Leftrightarrow \neg p(y)] \text{ AND } [\exists y \cdot p(\hat{d}_2) \Leftrightarrow \neg p(y)]$$

$$= ([p(\hat{d}_1) \Leftrightarrow \neg p(\hat{d}_1)] \text{ OR } [p(\hat{d}_1) \Leftrightarrow \neg p(\hat{d}_2)])$$

$$\text{AND } ([p(\hat{d}_2) \Leftrightarrow \neg p(\hat{d}_1)] \text{ OR } [p(\hat{d}_2) \Leftrightarrow \neg p(\hat{d}_2)])$$

$$= (([p](d_1) \text{ IFF NOT } [p](d_1)) \text{ OR } ([p](d_1) \text{ IFF NOT } [p](d_2))) \quad (1) \quad (2)$$

$$\text{AND } (([p](d_2) \text{ IFF NOT } [p](d_1)) \text{ OR } ([p](d_2) \text{ IFF NOT } [p](d_2))) \quad (3) \quad (4)$$

= we need 1 or 2 to be true, and we need either 3 or 4 to be true.

to make 2 T:  $[p](d_1) = T$  and  $[p](d_2) = F$ . Then we see that 3 will also be T:

$$= ((T \text{ IFF NOT } T) \text{ OR } (T \text{ IFF NOT } F))$$

$$\text{AND } ((F \text{ IFF NOT } T) \text{ OR } (F \text{ IFF NOT } F))$$

$$= (F \text{ OR } T) \text{ AND } (T \text{ OR } F) = T \text{ AND } T = T.$$

$\therefore [p : \text{univ} \rightarrow \text{bool}] := \{(d_1, T), (d_2, F)\}$  !

Note: anything within [...] must be syntax and everything not within these brackets must be semantics.

↳ unless using ' $d$ :

- You only have to show enough of a conjunction or disjunction to justify a reduction to T/F!
- If the value doesn't matter, just choose one.

Review of terms:

- 1) **satisfiable** iff there is an interpretation such that the formula returns T (is satisfied)
- 2) **tautology** iff every interpretation satisfies the formula.
- 3) **contradiction** iff every interpretation does NOT satisfy the formula.
- 4) **inconsistent** iff there is no interpretation in which all the formulas are satisfied.
- 5) **Valid** iff in all interpretations where the premises are T, the conclusion is also T.
- 6) **invalid** iff there is at least one interpretation in which all the premises are T but the conclusion is F.

A formula is a **tautology (valid)** iff its **negation** is a **contradiction** (aka not satisfiable).

Typed Example: find an interpretation in which

$q(c, d) \wedge \exists z : T_1 \cdot \neg(z = c) \wedge q(z, d)$  is T.

Note:  $c : T_1$ ,  $d : T_2$ .

$$[T_1] := \{v_1, v_2\}$$

$$[T_2] := \{\omega\}$$

$$[c : T_1] := v_1$$

$$[d : T_2] := \omega$$

$$[q : T_1 * T_2 \rightarrow \text{bool}] := \{(v_1, \omega, ?), (v_2, \omega, ?)\}$$

$$\hookrightarrow [q(c, d) \wedge \exists z : T_1 \cdot \neg(z = c) \wedge q(z, d)]$$

expanding  $\exists z$

$$= [q(c, d)] \text{ AND } (\neg(v_1 = c) \wedge q(v_1, d))$$

$$\text{OR } (\neg(v_2 = c) \wedge q(v_2, d))$$

$$= [q]([c], [d]) \text{ AND } (\text{NOT}(v_1 = [c]) \text{ AND } [q](v_1, [d]))$$
$$\text{OR } (\text{NOT}(v_2 = [c]) \text{ AND } [q](v_2, [d]))$$

$$= [q](v_1, \omega) \text{ AND } (\text{NOT}(v_1 = v_1) \text{ AND } [q](v_1, \omega))$$

$\downarrow$  must be T  
for formula to be T

$$\text{OR } (\text{NOT}(v_2 = v_1) \text{ AND } [q](v_2, \omega))$$

must be T for  
formula to be T!

$$= T \text{ AND } (\text{NOT } (T) \text{ and } T \text{ OR NOT } (F) \text{ AND } T)$$

$$= T \text{ AND } (F \text{ OR } T) = T \text{ AND } T = T.$$

$$\therefore [q : T_1 * T_2 \rightarrow \text{bool}] := \{(v_1, \omega, T), (v_2, \omega, T)\}.$$

Predicate Logic: Transformational Proof Rules:

- forall-over-and:

$$\hookrightarrow (\forall x \cdot p(x) \wedge q(x)) \Leftrightarrow ((\forall x \cdot p(x)) \wedge (\forall x \cdot q(x)))$$

• exists-over-or:

$$\hookrightarrow (\exists x \cdot p(x) \vee q(x)) \Leftrightarrow ((\exists x \cdot p(x)) \vee (\forall x \cdot q(x)))$$

• dm

$$\hookrightarrow (\neg(\exists x \cdot p(x))) \Leftrightarrow (\forall x \cdot \neg p(x))$$

$$\hookrightarrow (\neg(\forall x \cdot p(x))) \Leftrightarrow (\exists x \cdot \neg p(x))$$

• Swap-vars:

$$\hookrightarrow (\exists x, y \cdot p(x, y)) \Leftrightarrow (\exists y, x \cdot p(x, y))$$

$$\hookrightarrow (\forall x, y \cdot p(x, y)) \Leftrightarrow (\forall y, x \cdot p(x, y))$$

## Predicate Logic : Natural Deduction:

$P[t/x]$  means the formula  $P$  with term  $t$  replacing all/some free occurrences of variable  $x$ .  
 ↳ it's all occurrences if used below the line forall-e  
 ↳ it's some occurrences if used above the line exists-i.

$\frac{\forall x \cdot P}{P[t/x]} \rightarrow \text{forall-e}$  if a formula is true for all values, then it is true for any particular value!

↳  $t$  can be any term (a constant/variable/ function)

Example:

$c$  is a constant.

$$p(c), \forall x \cdot p(x) \Rightarrow \neg q(x) \vdash \neg q(c)$$

- 1)  $p(c)$  premise
- 2)  $\forall x \cdot p(x) \Rightarrow \neg q(x)$  premise
- 3)  $p(c) \Rightarrow \neg q(c)$  by forall-e on 2
- 4)  $\neg q(c)$  by imp-e on 1, 3

→ look out for this!!!

Variable Capture: a location in the formula that had a free variable has a term containing a bound variable after the substitution.