Basics of Propositional Logic

Dr. Isaac Griffith

Idaho State University

Propositional Logic

- Logic is used extensively through CS
- It provides an extremely powerful tool for reasoning
- Examples of Use:
  - SE: Software Specification
  - Program Correctness
  - IR: Web Search
  - AI: Simulated Intelligent Thought
  - ECE: Digital Circuit Design
  - CS: Typechecking in Compilers
  - PL: Lambda Calculus
  - Theory: Abstract Machine Models

Logic Forms

- **Propositional Logic:** The simplest form of logic, for which we will consider the following forms of reasoning
  - **Truth Tables:** Define the meanings of logical operators, to calculate the values of expressions, and to prove two propositions are equal
  - **Natural Deduction:** formalization of the basic principles of reasoning based on a set of *inference rules* and uses a syntactic approach
  - **Boolean Algebra:** another form of syntactic formalism, using a set of equations to specify that certain propositions are equal
- **Predicate Logic**
- **Advanced Logics**

Lecture Outline

Today's lecture will cover:

- Propositional Logic
- Logical Operators
- Language of Logic
- Sematic Reasoning using Truth Tables
- Inference Reasoning using Natural Deduction

Propositional Logic

CS 1187

ROAR

The Need for Formalism

- Natural languages, by their very nature are rich, expressive, and unfortunately ambiguous
- This presents significant difficulty in attempting to reason in natural language directly
- Instead, we separate the logical structure of an argument from the English connotations

- **Propositions** – symbolic variables whose vale must be either `True` or `False` and which stand for some english statement.
- These *propositional variables** are used instead of the natural language statements

Propositions and Variables

The process of reasoning using logic:

1. We identify the propositions from text:

$$\begin{aligned} A &= \text{The sun is shining} \\ B &= \text{I feel happy.} \\ C &= \text{Cats are furry.} \\ D &= \text{Elephants are heavy.} \end{aligned}$$

2. We translate complete English sentences into mathematical statements composed of only Propositional variables and logical operators:

$$\begin{aligned} \text{The sun is shining and I feel happy.} &\longrightarrow A \text{ and } B \\ \text{Cats are furry and elephants are heavy.} &\longrightarrow C \text{ and } D \end{aligned}$$

- Unfortunately, not all statements can be expressed using Propositional logic

Logical Operators

CS 1187

Logical And ($\wedge$)

- Corresponds to *'and'* in English
- Claims that two statements are both **true**
- **Symbols:** $\wedge$, $\cdot$, `&&`, `AND`
- $A \wedge B$ as a proposition means:
  - A is True
  - B is True
  - No connection between A and B
- We can build large propositions by simply adding them together
  - $A \wedge B$
  - $(A \wedge B) \wedge C$

Truth Tables

A	B	$A \wedge B$
F	F	F
F	T	F
T	F	F
T	T	T

A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

Inclusive Logical Or ($\vee$)

- Corresponds to English word *'or'*
- Returns true if either argument (or both) are true, false otherwise
- **Symbols:** $\vee$, $+$, OR, ||
- $A \vee B$ as a proposition means;
  - Perhaps A is true
  - Perhaps B is true
  - No connection between A or B

Truth Tables

A	B	$A \vee B$
F	F	F
F	T	T
T	F	T
T	T	T

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

Exclusive Logical Or (⊕)

- Corresponds to the English use of the word 'or': *one or the other but not both*
  - Thus, we must be careful when translating sentences using "or"
- **Symbols:** $\oplus$, `XOR`

Truth Tables

A	B	A ⊕ B
F	F	F
F	T	T
T	F	T
T	T	F

A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

Logical Not ($\neg$)

- Corresponds to english word *'not'*
- Negates a given proposition
- **Symbols:** $\neg$, $\overline{X}$, `!`, `NOT`

Truth Tables

A	$\neg$A
F	T
T	F

A	$\neg$A
0	1
1	0

ROAR

Logical Implication ($\rightarrow$)

- Corresponds to conditional statements in English
  - i.e., The proposition $A \rightarrow B$ corresponds to "If A is true, then B is true"
- Also closely related to the **if … then …** construct in programming
- **Symbols:** $\rightarrow$, $\Rightarrow$, $\supset$

Truth Tables

A	B	$A \rightarrow B$
F	F	T
F	T	T
T	F	F
T	T	T

A	B	$A \rightarrow B$
0	0	1
0	1	1
1	0	0
1	1	1

Logical Implication ($\rightarrow$)

- Does not actually imply a *cause-and-effect* relationship.
- It merely states that *if A is true, then B is true* **NOT** *because A is true, B is true*

Examples

A	It is sunny today.
B	There will be a picnic.
$A \rightarrow B$	If it is sunny today, then there will be a picnic.

A	The moon orbits the earth.
B	The sun is hot.
$A \rightarrow B$	If the moon orbits the earth, then the sun is hot.

Logical Equivalence ($\leftrightarrow$)

- Claims that two propositions have the same value
- May be translated to English as:
  "Saying A is just the same as saying B"
- $A \leftrightarrow B$ can also be expressed as $(A \rightarrow B) \land (B \rightarrow A)$
- **Note:** Logical equivalence is similar to *but not the same* as ordinary equality

Truth Tables

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

A	B	$A \leftrightarrow B$
0	0	1
0	1	0
1	0	0
1	1	1

Language of Logic

CS 1187

ROAR

Language of Logic

- Logic has the following advantages of English
  - Formal
  - Precise
  - Unambiguous

- Furthermore Logic has a formal grammar
- **Well-Formed Formula (WFF):** a proposition which “makes sense”
  - Has a well-defined meaning
  - its truth value can be discerned

- Saying a formula is WFF is like saying a Boolean expression in a programming language is syntactically valid

WFF Syntax

- **Formula** – a term in propositional logic
- **Well-Formed Formula** – a term that is correctly constructed, following all syntax rules

- **WFF Examples:**
  - $(P \wedge (\neg P))$
  - $(A \rightarrow (B \wedge (\neg A)))$

- **Non-WFF Examples:**
  - $\vee AB \neg C$
  - $P \vee \rightarrow Q$

WFF Syntax Rules

The rules for constructing WFFs are as follows:

- The constants `False` and `True` are WFFs
- Any propositional variable is a WFF
- If a and b are WFFs, then so is
  - $(\neg a)$
  - $(a \wedge b)$
  - $(a \vee b)$
  - $(a \rightarrow b)$
  - $(a \leftrightarrow b)$

WFF Syntax Rules

- These rules can be applied *recursively* to build nested formulas, or to verify a formula is a WFF
    - $(P \rightarrow (Q \wedge R))$
        1. P, Q, and R are propositional variables, so they are all WFFs
        2. Since, Q and R are WFFs, so is $(Q \wedge R)$
        3. Since, P and $(Q \wedge R)$ are WFFs, so is $(P \rightarrow (Q \wedge R))$

- **Exercise:** Use this reasoning to determine whether or not the following are WFFs:
    - $(P \vee \wedge Q)$
    - $P \rightarrow \neg$
    - $(P \wedge Q) \vee \neg(A \wedge \neg B)$

WFF Syntax Rules

- These rules can be applied *recursively* to build nested formulas, or to verify a formula is a WFF
  - $(P \rightarrow (Q \wedge R))$

    1. P, Q, and R are propositional variables, so they are all WFFs
    2. Since, Q and R are WFFs, so is $(Q \wedge R)$
    3. Since, P and $(Q \wedge R)$ are WFFs, so is $(P \rightarrow (Q \wedge R))$

- **Exercise:** Use this reasoning to determine whether or not the following are WFFs:
  - $(P \vee \wedge Q) \longrightarrow$ **Not a WFF**
  - $P \rightarrow \neg \longrightarrow$ **Not a WFF**
  - $(P \wedge Q) \vee \neg(A \wedge \neg B) \longrightarrow$ **A WFF**

Logical Operator Precedence

WFFs are fully parenthesized, so there is no ambiguity in interpretation. However, for the sake of readability parentheses are often omitted.

- Thus, we have the following order of precedence for operators. The lower the number, the more closely the operator binds with its arguments. Which means that it is evaluated before those below it.

1. $\neg$
2. $\wedge$
3. $\vee$
4. $\rightarrow$
5. $\leftrightarrow$

Meta-Logic

- Propositional logic provides an adequate language for WFFs
  - We call this the **object language**, and WFFs are the **objects**

- However, to reason *about* WFFs we need a different and richer language
  - We call this the **meta-language** as it allows us to talk *about* propositions
  - We add three additional operators: $\vdash$, $\vDash$, and $=$ as part of our meta-logic

Logic and Bit Operation

- **Bit** - (binary digit) is a symbol with two possible values: 0 (zero) and 1 (one)
  - $1 \leftrightarrow$ *True*
  - $0 \leftrightarrow$ *False*

Logic and Bit Operation

- **Bit** - (binary digit) is a symbol with two possible values: 0 (zero) and 1 (one)
  - $1 \leftrightarrow$ *True*
  - $0 \leftrightarrow$ *False*
- **Boolean variable** - a variable whose value can be either true or false
  - can be represented by a bit

Logic and Bit Operation

- **Bit** - (binary digit) is a symbol with two possible values: 0 (zero) and 1 (one)
  - $1 \leftrightarrow$ *True*
  - $0 \leftrightarrow$ *False*
- **Boolean variable** - a variable whose value can be either true or false
  - can be represented by a bit
- **Bit Operations**
  - *AND* $\leftrightarrow \wedge$
  - *OR* $\leftrightarrow \vee$
  - *XOR* $\leftrightarrow \oplus$

Logic and Bit Operation

- **Bit** - (binary digit) is a symbol with two possible values: 0 (zero) and 1 (one)
  - $1 \leftrightarrow True$
  - $0 \leftrightarrow False$
- **Boolean variable** - a variable whose value can be either true or false
  - can be represented by a bit
- **Bit Operations**
  - $AND \leftrightarrow \wedge$
  - $OR \leftrightarrow \vee$
  - $XOR \leftrightarrow \oplus$
- **Bit String** - a sequence of zero or more bits
  - **length** of this string is the number of bits in the string

Logic and Bit Operation

- **Bit** - (binary digit) is a symbol with two possible values: 0 (zero) and 1 (one)
  - $1 \leftrightarrow$ *True*
  - $0 \leftrightarrow$ *False*
- **Boolean variable** - a variable whose value can be either true or false
  - can be represented by a bit
- **Bit Operations**
  - *AND* $\leftrightarrow \wedge$
  - *OR* $\leftrightarrow \vee$
  - *XOR* $\leftrightarrow \oplus$
- **Bit String** - a sequence of zero or more bits
  - **length** of this string is the number of bits in the string

- **Example**

```
    1 0 1 1 0 1 1 0
    0 0 0 1 1 1 0 1
    ---------------
OR  1 0 1 1 1 1 1 1
AND 0 0 0 1 0 1 0 0
XOR 1 0 1 0 1 0 1 1
```

Bit Operator Truth Table

p	q	p AND q	p OR q	p XOR q
1	1	1	1	0
1	0	0	1	1
0	1	0	1	1
0	0	0	0	0

Computing w/ Boolean Expressions

- Using the provided software, we can write logic statements in Haskell:
  - $\neg x$ is written as: `not x`
  - $a \wedge b$ is written as either: `a && b` or as `a /\ b`
  - $a \vee b$ is written as either: `a || b` or as `a \/ b`
  - $a \to b$ is written as: `a ==> b`
  - $a \leftrightarrow b$ is written as: `a <=> b`

Exercises

- Check your understanding of logic by evaluating the following statements, first in you own head, and then using the `Stdm` module and Haskell:
  - `False /\ True`
  - `True \/ (not True)`
  - `True <=> (False ==> False)`
  - `False ==> False`
  - `False <=> (True /\ (False ==> True))`

Exercises

- Check your understanding of logic by evaluating the following statements, first in you own head, and then using the `Stdm` module and Haskell:
  - `False /\ True` $\longrightarrow$ **False**
  - `True \/ (not True)` $\longrightarrow$ **True**
  - `True <=> (False ==> False)` $\longrightarrow$ **True**
  - `False ==> False` $\longrightarrow$ **True**
  - `False <=> (True /\ (False ==> True))` $\longrightarrow$ **False**

Semantic Reasoning

CS 1187

Truth Tables

- Truth tables make for an easy method to reason about propositions
- Each proposition is treated as an expression to be evaluated
- It can be quite tedious to write out `True`, and `False` for each value
  - Can be shortened to `T` and `F` or `1` and `0`

Calculating Truth Tables

- **Tautology:** a proposition that is always `True`, regardless of the values of its variables
- **Contradiction:** a proposition that is always `False`, regardless of the values of its variables
- **Contingency:** a proposition whose truth value is dependent upon the truth values of its variables
- **Satisfiable:** a proposition wherein at least one combination of values for its variables leads to a `True` value for the entire proposition.

- You can use a truth table to determine if formula is a *tautology* or *contradiction*, by looking at the column of the formula
  - If the column contains only `True` values, then it is a *tautology*
  - If the column contains only `False` values, then it is a *contradiction*
- $P_1, P_2, \ldots, P_n \models Q$ means that if all the propositions $P_1, P_2, \ldots, P_n$ are true then, the proposition Q is also true.
  - $\models$ is a meta-logic operator (double-turnstile) concerning the *semantics*, or meaning, of the proposition(s).

Examples

A	B	$A \rightarrow B$	$\neg B$	$(A \rightarrow B) \wedge \neg B$	$\neg A$	$((A \rightarrow B) \wedge \neg B) \rightarrow \neg A$
F	F	T	T	T	T	T
F	T	T	F	F	T	T
T	F	F	T	F	F	T
T	T	T	F	F	F	T

P	$\neg P$	$P \vee \neg P$	$P \wedge \neg P$
F	T	T	F
T	F	T	F

Limitations of Truth Tables

- Although simple to use, truth tables have some limitations
  - Provides no insight into *why* a proposition is True or False
  - The Table requires 2^k lines, where *k* is the number of propositional variables
  - Tables for a large number of variables are unwieldy

Exercises

- Use truth tables to determine which of the following are tautologies:

$$(\text{True} \wedge P) \vee Q \qquad\qquad (P \vee Q) \rightarrow (Q \vee P)$$

Exercises

- Use truth tables to determine which of the following are tautologies:

$$(\text{True} \wedge P) \vee Q \qquad\qquad (P \vee Q) \rightarrow (Q \vee P)$$

P	Q	$\text{True} \wedge P$	$(\text{True} \wedge P) \vee Q$
F	F	F	F
F	T	F	T
T	F	T	T
T	T	T	T

Not a Tautology

P	Q	$P \vee Q$	$Q \vee P$	$(P \vee Q) \rightarrow (Q \vee P)$
F	F	F	F	T
F	T	T	T	T
T	F	T	T	T
T	T	T	T	T

Tautology

Exercises

- Verify each of the following is a WFF, then build a truth table for the formula, finally, determine if the formula is either a *tautology*, a *contradiction*, or not a tautology but *satisfiable*

$$(P \to Q) \land (P \to \neg Q) \qquad\qquad (P \to Q) \leftrightarrow (\neg Q \to \neg P)$$

Exercises

- Verify each of the following is a WFF, then build a truth table for the formula, finally, determine if the formula is either a *tautology*, a *contradiction*, or not a tautology but *satisfiable*

$$(P \to Q) \land (P \to \neg Q)$$

$$(P \to Q) \leftrightarrow (\neg Q \to \neg P)$$

P	Q	$P \to Q$	$P \to \neg Q$	$(P \to Q) \land (P \to \neg Q)$
F	F	T	T	T
F	T	T	T	T
T	F	F	T	F
T	T	T	F	F

Satisfiable

P	Q	$P \to Q$	$\neg Q \to \neg P$	$(P \to Q) \leftrightarrow (\neg Q \to \neg P)$
F	F	T	T	T
F	T	T	T	T
T	F	F	F	T
T	T	T	T	T

Tautology

Inference Reasoning

CS 1187

ROAR

Natural Deduction

- **Natural Deduction** – A formal logic system that allows for reasoning directly with propositions using *inference*
  - Does not require substituting truth values for variables or evaluation of expressions
  - Provides a solid foundation for formal logic
  - Focuses on *why* propositions are `True` or `False`

- **Infer** – to reason about statements in order to reach a conclusion

- **Logical Inference** – reasoning *formally* about a set of statements in order to decide what is true. This requires:
  - The set of statements to reason about $\rightarrow$ *The Object Language*
  - Methods for inferring facts from given information $\rightarrow$ *Rules of Inference*
  - A precise argument form from which we can determine argument validity $\rightarrow$ *The Meta-Language*

Natural Deduction

- **Sequent** – The notation $P_1, P_2, \ldots, P_n \vdash Q$
  - **Meaning:** if all propositions $P_1, P_2, \ldots, P_n$ are known, then the proposition Q can be inferred using the rules of inference
  - The notation $P \vdash Q \to P$ means *there is a proof* of $Q \to P$ which assumes P
- We can't simply make intuitive arguments in Logic
  - Every step of reasoning must be backed up by an inference rule

Natural Deduction

Idaho State University | Computer Science

- Inference rules are written using **meta-variables**
  - these are lower-case letters which can stand for any WFF
  - On the other hand propositional, or object, variables use upper-case letters

- **Example:**

$$\frac{a \qquad b}{a \wedge b}\{\wedge I\}$$

  - Here *a* and *b* are the meta-variables, which can be replaced by WFFs such as P, $Q \rightarrow R$, etc.
  - the components above the line are called the **assumptions**
  - we use the assumptions to *infer* the **conclusion** (component below the line)
  - The rule of inference labels the line itself

True, ¬, ↔

- Natural deduction uses a minimal set of operators, including:
  - The constant `False`
  - The three basic operators: $\vee$, $\wedge$, and $\rightarrow$
- The constant `True` and the operators $\neg$ and $\leftrightarrow$ are abbreviations defined as follows:

$$\begin{aligned} \text{True} &= \text{False} \rightarrow \text{False} \\ \neg a &= a \rightarrow \text{False} \\ a \leftrightarrow b &= (a \rightarrow b) \wedge (b \rightarrow a) \end{aligned}$$

ROAR

And Introduction $\{\wedge I\}$

Rule:

$$\frac{a \qquad b}{a \wedge b}\{\wedge I\}$$

Meaning: If you know that some prop a is $\texttt{True}$ and you also know that b is $\texttt{True}$, then you may infer that the prop $a \wedge b$ is $\texttt{True}$

Examples:

- **Theorem:** $P, Q \vdash P \wedge Q$

$$\frac{\mathrm{P} \qquad \mathrm{Q}}{P \wedge Q}\{\wedge I\}$$

- **Theorem:** $P, Q, R \vdash (P \wedge Q) \wedge R$

$$\frac{\dfrac{P \qquad Q}{P \wedge Q}\{\wedge I\} \qquad R}{(P \wedge Q) \wedge R}\{\wedge I\}$$

ROAR

Exercises

- Consider the following two propositions:

$$
\begin{aligned}
x &= A \wedge (B \wedge (C \wedge D)) \\
y &= (A \wedge B) \wedge (C \wedge D)
\end{aligned}
$$

- Describe the shapes of the proofs for x and y. Assuming A, B, C, and D

ROAR

And Elimination $\{\wedge E_L\}$, $\{\wedge E_R\}$

Idaho State University | Computer Science

Rules:

$$\frac{a \wedge b}{a}\ \{\wedge E_L\}$$

$$\frac{a \wedge b}{b}\ \{\wedge E_R\}$$

Examples:

- **Theorem:** $P, Q \wedge R \vdash P \wedge Q$

$$\dfrac{P \qquad \dfrac{Q \wedge R}{Q}\ \{\wedge E_L\}}{P \wedge Q}\ \{\wedge I\}$$

- **Theorem:** $P \wedge Q \vdash Q \wedge P$

$$\dfrac{\dfrac{P \wedge Q}{Q}\ \{\wedge E_R\} \qquad \dfrac{P \wedge Q}{P}\ \{\wedge E_L\}}{Q \wedge P}\ \{\wedge I\}$$

Additional Example:

- **Theorem:** For any well-formed propositions *a*, *b*, and *c*

$$a \wedge (b \wedge c) \vdash (a \wedge b) \wedge c$$

- *Proof:*

$$\dfrac{\dfrac{\dfrac{a \wedge (b \wedge c)}{a}\{\wedge E_L\} \quad \dfrac{\dfrac{a \wedge (b \wedge c)}{b \wedge c}\{\wedge E_R\}}{b}\{\wedge E_L\}}{a \wedge b}\{\wedge I\} \quad \dfrac{\dfrac{a \wedge (b \wedge c)}{b \wedge c}\{\wedge E_R\}}{c}\{\wedge E_R\}}{(a \wedge b) \wedge c}\{\wedge I\}$$

Exercise

- Prove $(P \land Q) \land R \vdash P \land (Q \land R)$

Exercise

- Prove $(P \wedge Q) \wedge R \vdash P \wedge (Q \wedge R)$

Solution:

$$
\dfrac{
  \dfrac{\dfrac{(P \wedge Q) \wedge R}{P \wedge Q}\{\wedge E_L\}}{P}\{\wedge E_L\}
  \qquad
  \dfrac{
    \dfrac{\dfrac{(P \wedge Q) \wedge R}{P \wedge Q}\{\wedge E_L\}}{Q}\{\wedge E_R\}
    \qquad
    \dfrac{(P \wedge Q) \wedge R}{R}\{\wedge E_R\}
  }{Q \wedge R}\{\wedge I\}
}{P \wedge (Q \wedge R)}\{\wedge I\}
$$

Imply Elimination $\{\rightarrow E\}$

Rule:

$$\frac{a \qquad a \rightarrow b}{b}\{\rightarrow E\}$$

Meaning: If you know a is true, and also that $a \rightarrow b$, then you can infer b.

Latin Name: *Modus Ponens*

Example:

- **Theorem:** $Q \wedge P, P \wedge Q \rightarrow R \vdash R$

$$\dfrac{\dfrac{\dfrac{Q \wedge P}{P}\{\wedge E_R\} \qquad \dfrac{Q \wedge P}{Q}\{\wedge E_L\}}{P \wedge Q}\{\wedge I\} \qquad P \wedge Q \rightarrow R}{R}\{\rightarrow E\}$$

- **Theorem:** For all propositions a, b and c, a, $a \rightarrow b, b \rightarrow c \vdash c$.

$$\dfrac{\dfrac{a \qquad a \rightarrow b}{b}\{\rightarrow E\} \qquad b \rightarrow c}{c}\{\rightarrow E\}$$

Exercise

- Prove $P, P \rightarrow Q, (P \wedge Q) \rightarrow (R \wedge S) \vdash S$.

Exercise

- Prove $P, P \to Q, (P \wedge Q) \to (R \wedge S) \vdash S$.

Solution:

$$\dfrac{\dfrac{\dfrac{P \qquad \dfrac{\cancel{P} \qquad P \to C}{Q}\{\to E\}}{P \wedge Q}\{\wedge I\} \qquad (P \wedge Q) \to (R \wedge S)}{R \wedge S}\{\to E\}}{S}\{\wedge E_R\}$$

ROAR

Imply Introduction $\{\rightarrow I\}$

Idaho State University | Computer Science

Rule:

$$\frac{a \vdash b}{a \rightarrow b}\{\rightarrow I\}$$

Meaning: in order to infer the logical implication $a \rightarrow b$, you must have a proof of b using a as an assumption.

Example:

- **Theorem:** $\vdash (P \wedge Q) \rightarrow Q$

$$\cfrac{\cfrac{P \wedge Q}{Q}\{\wedge E_R\}}{P \wedge Q \rightarrow Q}\{\wedge \rightarrow I\}$$

Imply Introduction $\{\rightarrow I\}$

- **Discharged Assumptions** – assumptions used temporarily to establish a sequent, after which they are thrown away.
  - In proofs we will either place discharged assumptions in square brackets, "[ ]", surround them by a box, or draw a line through the discharged assumptions

- Using the prior example:

$$\dfrac{\dfrac{\cancel{P \land Q}}{Q}\{\land E_R\}}{P \land Q \rightarrow Q}\{\rightarrow I\}$$

- **Theorem:** (Implication chain rule). For all propositions a, b and c. $a \rightarrow b$, $b \rightarrow c \vdash a \rightarrow c$

$$\dfrac{\dfrac{\dfrac{\cancel{a} \qquad a \rightarrow b}{b}\{\rightarrow E\} \qquad b \rightarrow c}{c}\{\rightarrow E\}}{a \rightarrow c}\{\rightarrow I\}$$

Or Introduction $\{\vee I_L\}$, $\{\vee I_R\}$

Rules:

$$\frac{a}{a \vee b}\ \{\vee I_L\}$$

$$\frac{b}{a \vee b}\ \{\vee I_R\}$$

Meaning: If the proposition a is true, then both $a \vee b$ and $b \vee a$ must also be true.

Example:

- **Theorem:** $P \wedge Q \vdash P \vee Q$
  - Alternative Proof 1:

$$\dfrac{\dfrac{P \wedge Q}{P}\ \{\wedge E_L\}}{P \vee Q}\ \{\vee I_L\}$$

  - Alternative Proof 2:

$$\dfrac{\dfrac{P \wedge Q}{Q}\ \{\wedge E_R\}}{P \vee Q}\ \{\vee I_R\}$$

Exercise

- Prove $P \rightarrow \text{False} \vee P$

Exercise

- Prove $P \rightarrow \text{False} \vee P$

Solution:

$$\frac{P}{\text{False} \vee P}\ \{\vee I_R\}$$

Or Elimination $\{\vee E\}$

Rule:

$$\frac{a \vee b \qquad a \vdash c \qquad b \vdash c}{c}\ \{\vee E\}$$

Meaning: Unfortunately if we know $a \vee b$ is true, we cannot directly conclude anything about a or b. However, if we also know there is a conclusion c which can be inferred from both a and b, then we know c must be true.

Example:

- **Theorem:** $(P \wedge Q) \vee (P \wedge R) \vdash P$

$$\frac{(P \wedge Q) \vee (P \wedge R)) \qquad \dfrac{\cancel{P \wedge Q}}{P}\ \{\wedge E_L\} \qquad \dfrac{\cancel{P \wedge R}}{P}\ \{\wedge E_L\}}{P}\ \{\vee E\}$$

ROAR

Identity $\{ID\}$

Rule:

$$\frac{a}{a}\{ID\}$$

Meaning: If we know *a* is true, then we know *a* is true

Examples:

- **Theorem:** $P \vdash P$

$$\frac{P}{P}\{ID\}$$

- **Theorem:** $\vdash$ True

$$\dfrac{\dfrac{\cancel{\text{False}}}{\text{False}}\{ID\}}{\text{False} \to \text{False}}\{\to I\}$$

Contradiction $\{CTR\}$

Rule:

$$\frac{\text{False}}{a}\ \{CTR\}$$

Meaning: We can infer *anything at all* given the assumption that `False` is true. This then describes the truthfulness of `False` indirectly.

Example:

- **Theorem:** $P, \neg P \vdash Q$

$$\dfrac{\dfrac{P \qquad P \rightarrow \text{False}}{\text{False}}\ \{\rightarrow E\}}{Q}\ \{CTR\}$$

Contradiction $\{CTR\}$

- **Theorem:** For all propositions a and b, $a \vee b, \neg a \vdash b$

$$
\dfrac{a \vee b \qquad \dfrac{\dfrac{\cancel{a} \qquad a \rightarrow \text{False}}{\text{False}}\{\rightarrow E\}}{b} \qquad \dfrac{\cancel{b}}{b}\{ID\}}{b}\{\vee E\}
$$

Reductio ad Absurdum $\{RAA\}$

Rule:

$$\frac{\neg a \vdash \text{False}}{a}\ \{RAA\}$$

Meaning: If we can infer `False` from an assumption of $\neg a$, then a must be true.

- This is the underpinning of the *proof by contradiction strategy*
  - First, we assume the contradiction $\neg a$ and infer False
  - Then, using $\{RAA\}$ we infer a

Examples:

- **Theorem:** (Double negation) $\neg\neg a \vdash a$

  We will use the general form:

$$\frac{\neg a \qquad \neg\neg a}{\text{False}}$$

  Recall: $\neg a = a \rightarrow \text{False}$ thus,
  $\neg\neg a = (a \rightarrow \text{False}) \rightarrow \text{False}$

$$\dfrac{\dfrac{\cancel{a \rightarrow \text{False}} \qquad (a \rightarrow \text{False}) \rightarrow \text{False}}{\text{False}}\ \{\rightarrow E\}}{a}\ \{RAA\}$$

For Next Time

- Review DMUC Chapter 6.1 – 6.5
- Review this Lecture
- Read Chapter 6.6 – 6.10
- Come To Lecture

Are there any questions?