UECM1304 Discrete Mathematics with Applications
Topic 1: Logic of Compound Statements and Quantified Statements

Dr Liew How Hui

May 2021

Three Major Topics:

- First Order Logic (Week 1–5):
  - Logic of Compound Statements and Quantified Statements → Model Theory
  - Valid and Invalid Arguments → Proof Theory
- Mathematical Proofs (Week 6–9):
  - Elementary Number Theory and Methods of Proof
- Set Theory (Week 10–14):
  - Relations

Relation To Comp Sci

Discrete Maths	Computer Science
Propositional Logic	Circuit Level
Predicate Logic	Register-Transfer Level
?	Specification Level
Proof Theory	Programming Level
Data Structure and Algorithms (Number Theory, Combinatorics)	Library Level
Model Theory?	Verification Level
Computer Simulation	?
Statistical Learning?	Knowledge Level

UECM1304 References

Main reference:

1. Epp, S.S., 2020. Discrete Mathematics with Applications. 5th ed. Boston, MA: Brooks/Cole Cengage Learning.

Additional references:

2. Rosen, K.H., 2019. Discrete Mathematics and its Applications. 8th ed. New York: McGraw-Hill.
3. Scheinerman, E.R., 2013. Mathematics: A Discrete Introduction. 3rd ed. Boston, Mass.: Brooks/Cole.

Coursework Assessment

Courseworks

- Quiz (20%, Week 5?, Covering Topic 1)
- Test (20%, Week 10?, Covering Topic 2 & Topic 3)
- Assignment (20%, End of Submission: Week 13, Covering Topic 4)

Final Assessment (40%)

- 4 Questions: Each 10%

Software & Prerecorded Videos

We are going to use the following software which are "specialised" for mathematics:

- Racket language (`https://racket-lang.org/`): Programming using functions
- Coq (`https://coq.inria.fr/`): For formal prooving
- Prolog (`https://www.swi-prolog.org/`): Programming in Logic

Additional Software: LibreOffice Calc / Excel, Python?

Prerecorded videos (complement to the lecture) will be available in MS Teams from time to time.

Outline

Purpose of "Formal" Logic

- Nothing to do with philosophy.
- Structuring human knowledge.
  - Why? Formal logic is what computer can handle.
  - Easier to make logical queries: SQL & Datalog
  - More rigorous engineering design → security and safety.
- Application in electronic Chip / IC design.

Informal propositions

We will begin "informally" with *statements* or *propositions*, which are sentences which are either **true** or **false**, but not both. Normally they will be denoted as p, q, r, etc. or the indexed letters p_1, p_2, etc. These letters are called **statement variables**, that is, variables that can be replaced by statements.

Informal propositions (cont)

Example 1.1.1: Determine if the following are informal propositions:

(a) The year 1973 was a leap year.

(b) 28234423783 is a prime number.

(c) The equation $x^2 + 3x + 2 = 0$ has two different roots in $\mathbb{R}$.

(d) $x^2 + x + 1 = 0$, x is a real number.

(e) She is a computer science major.

(f) Maths is fun.

(g) Is $2^{10} - 1$ an even integer?

(h) Read a maths book.

Outline

Syntax of Compound Statements

Most sentences or mathematical statements are the combinations of simpler statements formed through some choice of the words *not*, *and*, or, *if ... then ...*, and *if and only if*. These are called *logical connectives* (or simply *connectives*) and are denoted by the following symbols:

$\sim$, $\neg$, !	Not
$\wedge$, $\cdot$, &	And
$\vee$, +, $\|\|$	Or
$\rightarrow$, $\supset$	If ..., then ...
$\leftrightarrow$	If and only if

Formal Propositions

Definition (Well-Formed Formula): *Statements* (or *propositions*) are either atomic or compound.

1. Constants T, $\top$, $\bot$, F and single statement variables p, q, r, s, t, p_i, $i = 1, 2, 3, \cdots$ are *atomic* (or *simple* or *primitive*) *statements or formulas*.
2. If ϕ and ψ are statements (abbreviated notations), then the expressions

$$(\sim \phi),\ (\phi \wedge \psi),\ (\phi \vee \psi),\ (\phi \rightarrow \psi),\ (\phi \leftrightarrow \psi)$$

are also *statements* or *formulas*. They are normally called *compound statements*.

Formal Propositions (cont)

Note that we have introduced parentheses in all the expressions in Definition of Well-Formed Formula. Sometimes too many parentheses can be annoying. Therefore, sometimes parentheses are "simplified" based on the *precedence*:

1. Evaluate parentheses first;
2. Then evaluate negations;
3. Then evaluate $\wedge$;
4. Then evaluate $\vee$;
5. Then evaluate $\rightarrow$;
6. Then evaluate $\leftrightarrow$.

Formal Propositions (cont)

Examples:

- $p \wedge q \vee \sim r \rightarrow s$ is an abbreviation of $((p \wedge q) \vee (\sim r)) \rightarrow s$;
- $p \rightarrow q \rightarrow r \rightarrow s$ means $p \rightarrow (q \rightarrow (r \rightarrow s))$

Formal Propositions (cont)

How to "read":

(a) $\sim \phi$ or $\neg\phi$ is read as "not ϕ". This statement is called the "negation of ϕ".

(b) $\phi \wedge \psi$ is read as the *conjunction* of ϕ and ψ or just "ϕ and ψ".

(c) $\phi \vee \psi$ is read as the *disjunction* of ϕ and ψ, or just "ϕ or ψ".

Formal Propositions (cont)

How to "read":

(d) $\phi \rightarrow \psi$ is read as "ϕ implies ψ" or "if ϕ then ψ". This form of statement is called a *conditional statement* or *implication*. In $\phi \rightarrow \psi$, the statement ϕ is called the *hypothesis* and the statement ψ is called the *conclusion* or *consequent*. The statement ϕ is called the *sufficient condition* for ψ and ψ is called the *necessary condition* for ϕ.

Formal Propositions (cont)

How to "read":

(e) $p \leftrightarrow q$ is read as "ϕ if and only if ψ" and is normally written as "p iff q" (popularised by the mathematician Paul Halmos). This form of statement is called the *biconditional of ϕ and ψ* and ϕ is called the *necessary and sufficient condition* for ψ.

Formal Propositions (cont)

Example 1.1.4: Study the following statements and decide whether they are atomic or compound.

1. A dog is not an animal.
2. $5 < 3$.
3. If the earth is flat, then $3 + 4 = 7$.

Formal Propositions (cont)

Example 1.1.5: Given the informal statements:

1. p: The integer 10 is even.
2. q: $2 + 3 > 1$
3. r: $3 + 7 = 10$
4. p_1: It is snowing. q_2: I am cold.
5. p_1: 2 is a positive integer. q_2: 2 is a rational number.

Read the compound statements (informally): $\sim p$, $\sim q$, $\sim r$, $p_1 \wedge q_1$, $p_2 \vee q_2$.

Formal Propositions (cont)

Remark:

1. An English sentence "ϕ but ψ" usually means "ϕ and ψ".
2. An English sentence "neither ϕ nor ψ" usually means "$\sim \phi$ and $\sim \psi$.
3. The notation for inequalities involves "and" and "or" statements. Let a, b and c be real numbers. Then
   - $a \leq b$ means "$a < b$" or "$a = b$"
   - $a < b < c$ means "$a < b$" and "$b < c$".

Formal Propositions (cont)

A variety of terminology is used to express $p \rightarrow q$ as given below:

If p then q	q is necessary for p
p implies q	q when p
q follows from p	q if p
p is sufficient for q	p only if q

Note that to say "p only if q" means that p can take place only if q takes place also. That is, if q does not take place, then p cannot take place (symbolically "$\sim q \rightarrow \sim p$"). Another way to say this is that if p occurs, then q must also occur.

Formal Propositions (cont)

Definition: Let p and q be statement variables:

1. The *negation* of $p \to q$ is $p \wedge \sim q$.
2. The *contrapositive* of $p \to q$ is $\sim q \to \sim p$.
3. The *converse* of $p \to q$ is $q \to p$.
4. The *inverse* of $p \to q$ is $\sim p \to \sim q$.

Note: the truth tables of $p \to q$ and its contrapositive are the same, and the truth tables of the converse and the inverse are the same.

Formal Propositions (cont)

Write the negation, contrapositive, converse and inverse of the following conditional statement:

If 3 is positive then 3 is nonnegative.

- Negation: 3 is positive and 3 is not nonnegative.
- Contrapositive: If 3 is not nonnegative then 3 is not positive.
- Converse: If 3 is nonnegative then 3 is positive.
- Inverse: If 3 is not positive then 3 is not non-negative.

Formal Propositions (cont)

By using algebra, it is possible for us to construct many statements which fulfils the syntax requirement. To give "meaning" or "semantics" to a statement, we need to interpret it with a truth value.

Definition: The *truth value* of a statement/proposition is true (T or 1), if it is a true statement/proposition and false (F or 0), if it is a false statement/proposition.

Formal Propositions (cont)

Definition:

1. An *atomic truth assignment* is a function v that maps an actomic symbol to a value in $\{T, F\}$ (or $\{\top, \bot\}$).
2. A *truth assignment* (or a *model* or a *valuation*) is a function v such that for any sentences ϕ and ψ,
   1. $v(\sim \phi) = \mathrm{T}$ if $v(\phi) = \mathrm{F}$ else F
   2. $v(\phi \vee \psi) = \mathrm{T}$ if any one of $v(\phi)$, $v(\psi)$ is T else F
   3. $v(\phi \wedge \psi) = \mathrm{F}$ if any one of $v(\phi)$, $v(\psi)$ is F else T
   4. $v(\phi \rightarrow \psi) = \mathrm{T}$ if $v(\phi) = \mathrm{F}$ else $v(\psi)$
   5. $v(\phi \leftrightarrow \psi) = \mathrm{T}$ if $v(\phi) = v(\psi)$ else F

Formal Propositions (cont)

Example 1.1.23: Given that p and q are true and r, s and t are false, find $v(q \to (\sim r \to (r \to (p \vee s))))$.

Solution

Given $v(p) = v(q) = \mathrm{T}$, $v(r) = v(s) = \mathrm{F}$.

$v\Big(q \to (\sim r \to (r \to (p \vee s)))\Big)$

$= \mathrm{T}$ if $v(q) = \mathrm{F}$ else $v\Big(\sim r \to (r \to (p \vee s))\Big) = v\Big(\sim r \to (r \to (p \vee s))\Big)$

$= \mathrm{T}$ if $v(\sim r) = \mathrm{F}$ else $v\Big(r \to (p \vee s)\Big)$

$= \mathrm{T}$ if (T if $v(r) = \mathrm{F}$ else F)=F else $v\Big(r \to (p \vee s)\Big)$

$= \mathrm{T}$ if T=F else $v\Big(r \to (p \vee s))\Big) = v\Big(r \to (p \vee s))\Big) = \mathrm{T}$ if $v(r) = \mathrm{F}$ else $v(p \vee s) = \mathrm{T}$.

Note: The definition of truth assignment is defined **recursively** and it is useful in computer implementation. (For Racket, see next slide).

Formal Propositions (cont)

Using Racket language, Example 1.1.23 can be expressed as

```
(let ([p #t] [q #t] [r #f] [s #f] [t #f])
  (-> q (-> (not r) (-> r (\/ p s)))))
```

How to represent 'maths' in Racket language?

- Functions $f(x, y, z)$ are expressed as (f x y z).
- $p \vee s$ is regarded as a function $\vee(p, s)$. So it is written as $(\vee\ p\ s)$ in Racket.
- Exercise: Use a pen to write down the 'functional form' for the proposition.
- Compare to `eg_1_1_23.rkt`.

Formal Propositions (cont)

The mathematical definition is how we teach computer to determine the truth values.

For hand calculation, we usually just perform **substitution**.

For all **combinations** of truth assignments into a proposition, we obtain the truth table.

http://en.wikipedia.org/wiki/Truth_table

The *truth table* for a given statement displays the truth values that correspond to all possible combinations of truth values for its atomic statement variables.

Formal Propositions (cont)

If a statement s has n atomic statements, there will need to be 2^n rows in the truth table for s.

Truth Table for 'Not'. $n = 1$, $2^n = 2$ rows

Let p be an atomic statement, the truth table for $\sim p$ is

p	$\sim p$
T	F
F	T

This is an abbreviation for

- If $v(p) = T$, $v(\sim p) = F$;
- If $v(p) = F$, $v(\sim p) = T$.

Formal Propositions (cont)

Truth Table for Binary Operations. $n = 2$, $2^n = 4$ **rows**

Let p and q be atomic statements, the truth table for $\wedge$, $\vee$, $\rightarrow$ and $\leftrightarrow$ is

p	q	$p \wedge q$	$p \vee q$	$p \rightarrow q$	$p \leftrightarrow q$
T	T	T	T	T	T
T	F	F	T	F	F
F	T	F	T	T	F
F	F	F	F	T	T

Formal Propositions (cont)

Example 1.1.23: Determine the truth value of the following statements:

1. $3 < 5$ and $5 + 6 \neq 11$.
2. The integer 2 is even but it is a prime number.
3. 3 or -5 is negative.
4. $\sqrt{2}$ or π is an integer.
5. $5 \leq 5$
6. 2 is prime if and only if it is multiple of 2.
7. 2 is negative if and only if 4 is negative.
8. $0 < 1 \leftrightarrow 2 < 1$

Formal Propositions (cont)

Determine the hypothesis and conclusion for each of the following conditional statements. Then determine the truth value.

1. The moon is square only if the sun rises in the East.
2. 1 and 3 are prime if 1 multiply 3 is prime.
3. $(\sin\pi)(\cos\pi) = 0$ when $\sin\pi = 0$ or $\cos\pi = 0$.
4. If 1 + 1 = 3, then cats can fly.

Recall: Slide 17.

Formal Propositions (cont)

Example 1.1.26: Construct a truth table for the statement $(p \wedge q) \vee \sim r$.

Solution

There are three atoms "p, q, r". The truth table:

p	q	r	$\sim r$	$p \wedge q$	$(p \wedge q) \vee \sim r$
T	T	T	F	T	T
T	T	F	T	T	T
T	F	T	F	F	F
T	F	F	T	F	T
F	T	T	F	F	F
F	T	F	T	F	T
F	F	T	F	F	F
F	F	F	T	F	T

Formal Propositions (cont)

Example 1.1.27: Construct the truth table for ϕ: $(p_1 \wedge p_2) \vee (p_3 \wedge p_4)$.

Practice with Excel?

Formal Propositions (cont)

Example 1.1.29: Construct a truth table for each statement in (a) $q \wedge \sim (\sim p \rightarrow r)$; (b) $(\sim p \leftrightarrow \sim r) \vee (r \leftrightarrow q)$.

Practice with Excel?

Tautology

Definition 1.2.1:

1. A statement ϕ is said to be a *tautology* or a *tautologous statement* if for all truth assignments for the atomic statement variables in ϕ, $v(\phi) = \mathrm{T}$, i.e. its truth values in the truth table are all true. If ϕ is a tautology, it is denoted as $\models \phi$.
2. A statement ϕ is said to be a *contradiction* or a *contraditory statement* if its truth values in all rows in the truth table are all false.
3. A statement ϕ is said to be a *contingency* if it is neither a tautology nor a contradiction.
4. We will use the constant T or $\top$ to denote a tautology and F or $\bot$ to denote a contradiction.

Tautology (cont)

Example 1.2.2. Let p, q and r be statement variables. Show that the statement form

1. $\sim p \vee p$ is a tautology.
2. $\sim p \wedge p$ is a contradiction.
3. $(p \wedge q) \vee \sim r$ is a contingency.

Remark: By using 'truth table'.

Formal Propositions (cont)

Final Exam May 2019, Q1(a): Let p, q, r be atomic statements. **State** the truth table for the following compound statement

$$\sim (p \rightarrow ((p \vee q) \wedge r)).$$

Use the truth table to **recognise** whether the compound statement is a tautology, contingency or contradiction. (10 marks)

Formal Propositions (cont)

Lecturer's Marking Guide

The truth table is stated below. . . . . [8 marks]

p	q	r	$(p \vee q) \wedge r$	$p \rightarrow ((p \vee q) \wedge r)$	$\sim (p \rightarrow ((p \vee q) \wedge r))$
T	T	T	T	T	F
T	T	F	F	F	T
T	F	T	T	T	F
T	F	F	F	F	T
F	T	T	T	T	F
F	T	F	F	T	F
F	F	T	F	T	F
F	F	F	F	T	F

It is sometimes true, sometimes false, depending on the truth assignment, by definition, the compound statement is a *contingency*. . . . . [2 marks]

Logical Equivalence (cont)

Final Exam May 2013, Q1(a)

Use truth table to show that
$\models (p \to q) \wedge (r \to s) \to ((p \wedge r) \to (q \wedge s))$. The truth table must contain columns for both $(p \to q) \wedge (r \to s)$ and $(p \wedge r) \to (q \wedge s)$. (10 marks)

Class Discussion.

Outline

Logical Equivalence

Definition 1.2.5

Two statements ϕ and ψ are called *logically equivalent* or *tautologically equivalent* if $\phi \leftrightarrow \psi$ is a tautology. We use the notations $\phi \equiv \psi$ and $\phi \Leftrightarrow \psi$ to denote that ϕ and ψ are logically equivalent.

Example 1.2.6

Let p and q be two statement variables. Determine whether the following statements are logically equivalent or not.

1. $\sim p \vee \sim q$ and $\sim (p \vee q)$
2. $p \rightarrow q$ and $\sim q \rightarrow \sim p$

Logical Equivalence (cont)

Final Exam May 2019 Q1(b): Show that the statement $(p \to q \vee r)$ and the statement $(p \wedge q \to r)$ are not logically equivalent. (4 marks)

Lecturer's Marking Guide

One can either construct a truth table or just give a counterexample below to show that they are not equivalent:

p	q	r	$p \to q \vee r$	$p \wedge q \to r$
T	T	T	T	T
T	T	F	T	F

.....[2 marks]

When $v(p)$ =T, $v(q)$ =T and $v(r)$ =F, the two statements has different truth values and they are not logically equivalent. [2 marks]

Logical Equivalence (cont)

Final Exam May 2019 Q2(a): Let p, q, r be atomic statements. Use a truth table or a comparison table to show that

$$(p \to r) \wedge (q \to r) \equiv (p \vee q) \to r. \qquad \text{(9 marks)}$$

Logical Equivalence (cont)

Lecturer's Marking Guide

The comparison table is given below.

p	q	r	$(p \to r) \wedge (q \to r)$	$(p \vee q) \to r$
T	T	T	T	T
T	T	F	F	F
T	F	T	T	T
T	F	F	F	F
F	T	T	T	T
F	T	F	F	F
F	F	T	T	T
F	F	F	T	T

.....[8 marks]

Since the last two columns are the same for all different assignments, therefore, the two statements $(p \to r) \wedge (q \to r)$ and $(p \vee q) \to r$ are logically equivalent.

.....[1 mark]

Logical Equivalence (cont)

Mathematicians have identified the useful laws for simplifying statements based on the concept of logical equivalence (LE). They are summarised in the following theorem.

Theorem 1.3.1 (LE Rules)

Given any atomic statements p, q and r, the following logical equivalences hold.

1. Commutative laws:
   - $p \wedge q \equiv q \wedge p$;
   - $p \vee q \equiv q \vee p$.
2. Associative laws:
   - $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$;
   - $(p \vee q) \vee r \equiv p \vee (q \vee r)$.

Logical Equivalence (cont)

Theorem 1.3.1 (cont)

3. Distributive laws:
   - $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$;
   - $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$.
4. Identity laws:
   - $p \wedge T \equiv p$;
   - $p \vee F \equiv p$.
5. Negation laws:
   - $p \vee \sim p \equiv T$;
   - $p \wedge \sim p \equiv F$.
6. Double negative law:
   - $\sim (\sim p) \equiv p$.

Logical Equivalence (cont)

Theorem 1.3.1 (cont)

7. Idempotent laws:
   - $p \wedge p \equiv p$;
   - $p \vee p \equiv p$.
8. Universal bound laws:
   - $p \vee T \equiv T$;
   - $p \wedge F \equiv F$.
9. De Morgan's laws:
   - $\sim (p \wedge q) \equiv \sim p \vee \sim q$;
   - $\sim (p \vee q) \equiv \sim p \wedge \sim q$.
10. Absorption laws:
    - $p \vee (p \wedge q) \equiv p$;
    - $p \wedge (p \vee q) \equiv p$.

Logical Equivalence (cont)

Theorem 1.3.1 (cont)

11. Implication law:
    - $p \rightarrow q \equiv \sim p \vee q$
12. Biconditional law:
    - $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$.

Proof

Since any statements can only be T or F, truth table can be used to prove 1. to 12.

Logical Equivalence (cont)

Example 1.3.2 (Logical Equivalence of Negative Statements): Write the negation of the given statements and expand them as well as writing the informal negation sentence.

1. John is smart but lazy.
2. $2 \leq \sqrt{2}$

Logical Equivalence (cont)

To make the laws of logical equivalences useful, mathematicians have shown that the logical equivalence "$\equiv$" is an **equivalence relation** (last topic). We also require logically equivalent statements to be logically equivalent under substitution.

Uniform Substitution

If ϕ, ψ, ψ_i are statements/formulas, and p_i are (atomic) propositional variables, then $\phi[\psi/p_i]$ denotes the result of replacing each occurrence of p_i by an occurrence of ψ in ϕ; similarly, the *simultaneous substitution* S of p_1, $\cdots$, p_n by formulas $\psi_1, \cdots, \psi_n$ is denoted by $\psi_1/p_1, \cdots, \psi_n/p_n$.

Logical Equivalence (cont)

Substitution Theorem

Let S be a simultaneous substitution (Definition 52) and $\phi \equiv \psi$. Then $\phi[S] \equiv \psi[S]$. More generally, let ϕ_i and ψ_i be statements such that $\phi_i \equiv \psi_i$, $1 \leq i \leq n$. Then $\phi[\phi_1/p_1, \cdots, \phi_n/p_n] \equiv \psi[\psi_1/p_1, \cdots, \psi_n/p_n]$.

Example 1.3.5

Show that

1. $(p \vee q) \wedge ((p \vee q) \vee r) \wedge \sim (\sim (p \vee q)) \equiv p \vee q$
2. $(p \vee q) \wedge \sim (\sim p \wedge q) \equiv p$
3. $\sim [\sim ((p \vee q) \wedge r) \vee \sim q] \equiv q \wedge r$

Class Discussion.

Logical Equivalence (cont)

Final Exam May 2019 Q1(c): Simplify the following statement

$$((p \vee q) \rightarrow (p \wedge q)) \vee (\sim p \wedge q).$$

to a logically equivalent statement with no more than TWO(2) logical connectives from the set $\{\sim, \wedge, \vee\}$ by stating the law used in each step of the simplification.

(5 marks)

Logical Equivalence (cont)

Lecturer's Marking Guide

The steps are shown below:

$((p \vee q) \rightarrow (p \wedge q)) \vee (\sim p \wedge q)$

$\equiv (\sim (p \vee q) \vee (p \wedge q)) \vee (\sim p \wedge q)$ [Implication law, 1 mark]

$\equiv (\sim p \wedge \sim q) \vee (p \wedge q) \vee (\sim p \wedge q)$ [de Morgan law, 1 mark]

$\equiv (\sim p \wedge \sim q) \vee (p \wedge q) \vee (\sim p \wedge q) \vee (\sim p \wedge q)$ [Idempotent law, 1 mark]

$\equiv \sim p \wedge (q \vee \sim q) \vee ((p \vee \sim p) \wedge q)$ [Distributive law, 1 mark]

$\equiv \sim p \vee q$ [Negation and identity, 1 mark]

Logical Equivalence (cont)

Final Exam May 2019 Q2(b): Simplify the following statement to a logically equivalent statement with no more than TWO(2) logical connectives from the set $\{\sim, \wedge, \vee\}$ by stating the law used in each step of the simplification:

$$(\sim p \wedge q) \vee (\sim p \wedge r) \vee (p \wedge \sim q \wedge r) \vee (q \wedge r).$$ (7 marks)

Note: Lecturer can set the wrong question. It is impossible to have less than 3 connectives from $\{\wedge, \vee, \sim\}$.

Logical Equivalence (cont)

Lecturer's Marking Guide

The simplification is shown below:

$$(\sim p \wedge q) \vee \Big(\sim p \wedge r \Big) \vee (p \wedge \sim q \wedge r) \vee \Big(q \wedge r \Big)$$

$$\equiv (\sim p \wedge q) \vee \Big((\sim p \vee q) \wedge r \Big) \vee (p \wedge \sim q \wedge r)$$

[Associative, Commutative & Distributive laws, 2 marks]

$$\equiv (\sim p \wedge q) \vee ((\sim p \vee q) \wedge r) \vee \Big(\sim (\sim p \vee q) \wedge r \Big)$$

[De Morgan law & Double Negation law, 2 marks]

$$\equiv (\sim p \wedge q) \vee \Big(\Big(\sim p \vee q \Big) \vee \sim \Big(\sim p \vee q \Big) \Big) \wedge r \Big)$$

[Distributive law, 1 mark]

$$\equiv (\sim p \wedge q) \vee \Big(T \wedge r \Big)$$

[Negation law, 1 mark]

$$\equiv (\sim p \wedge q) \vee r$$

[Identity law, 1 mark]

Logical Equivalence (cont)

The laws of logical equivalence can be used to simplify some long statements to logically equivalent shorter statements as demonstrated below.

Example 1.3.7

Simplify the following statement to a statement with no more than 3 logical connectives by stating the law used in each step of the simplification:

$$(p \vee q) \wedge p \wedge (r \vee q) \wedge (p \vee \sim p \vee r) \wedge (r \wedge \sim q).$$

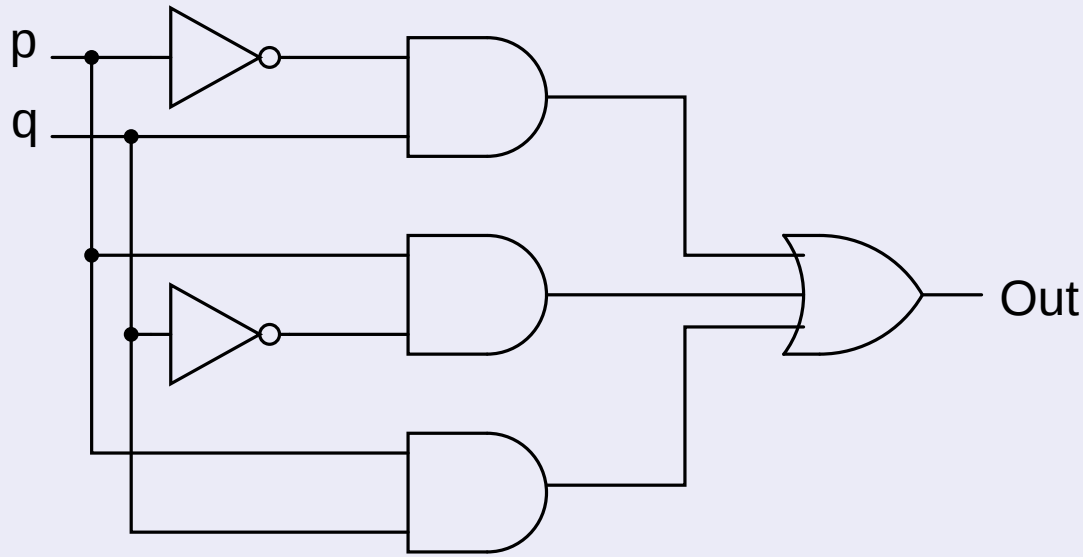
Logical Equivalence (cont)

The things we discussed so far has applications in logic circuit design under a different name called Boolean algebra. More complicated Boolean algebra could be found in digital signal processing systems. The following examples demonstrations of propositional logic (which is theoretically the same thing as the Boolean algebra) with the representation of electronic circuits and a simple logic puzzle problem.

Logical Equivalence (cont)

Example 1.3.12

Simplify the logic diagram below.



Electronics engineers use Karnaugh map (based on logical equivalence) to perform simplication.

Logical Equivalence (cont)

Four candidates A, B, C and D are to be selected to participate in a chess competition. However, only two candidates will be chosen in the final list and the following constraints must be obeyed:

- Only one from A and B will participate.
- If C participates, then D will also need to participate.
- At most one of B and D will participate.
- If D is not participating, then A is also not participating.

Logical Implication

Definition 1.4.1

We say that a statement ϕ *logically implies* a statement ψ or ψ is a *tautological consequence* of ϕ, denoted by $\phi \Rightarrow \psi$, when the statement $\phi \rightarrow \psi$ is a tautology, i.e. $\models \phi \rightarrow \psi$. When this happens, we sometimes say that the statement ϕ is *stronger* then ψ.

The link between logical equivalence and logical implication is as follows.

Theorem 1.4.2

Let ϕ and ψ be two statements. Then $\phi \equiv \psi$ iff $\phi \Rightarrow \psi$ and $\psi \Rightarrow \phi$.

Logical Implication (cont)

By definition, to show that "$\phi \Rightarrow \psi$", i.e. a statement ϕ logically implies another statement ψ, we can create the truth table for the statement $\phi \rightarrow \psi$ and examine its last column.

For statements related to logical implication, the final step of the truth table is always to evaluate the implication. The *comparison table*, which is a striped version of a truth table, lacks the last column since for implication, can be used. We know that as long as there is no $T \rightarrow F$, we know that the implication will be true.

Logical Implication (cont)

Show that $p \wedge q \Rightarrow p \vee q$.

Proof

The corresponding truth table is

p	q	$p \wedge q$	$p \vee q$	$p \wedge q \rightarrow p \vee q$
T	T	T	T	T
T	F	F	T	T
F	T	F	T	T
F	F	F	F	T

Since the last column consists entirely of T's, the implication $p \wedge q \rightarrow p \vee q$ is a *tautology*, i.e. $\models p \wedge q \rightarrow p \vee q$ so that $p \wedge q \Rightarrow p \vee q$.

Logical Implication (cont)

Definition 1.4.7

For any statements ϕ, ϕ_1, ϕ_2, $\cdots$, ϕ_n and ψ,

$$\{\phi\} \,/\, \therefore \psi, \quad \text{or} \quad \{\phi_1, \phi_2, \cdots, \phi_n\} \,/\, \therefore \psi \quad \text{or} \quad \begin{array}{l} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_n \\ \hline \therefore \ \psi \end{array}$$

are called *arguments* (or *deductive argument*). We call $\{\phi\}$, $\{\phi_1, \phi_2, \cdots, \phi_n\}$ the set of *hypotheses*, *premises* or *assumptions* and ψ the *conclusion* of the argument. When there is no hypothesis, an argument is written as $/ \therefore \psi$.

Logical Implication (cont)

Definition 1.4.8

When the premises ϕ, ϕ_1, ϕ_2, $\cdots$ and ϕ_n are true, ψ is true, then we say the argument is *valid*. ψ is said to be *deduced* or *inferred* from the premises, and that ψ *follows logically from* or is a *logical consequence* of the premises or that ψ. An argument that is not valid is said to be *invalid* or *fallacious*.

By definition, to determine the validity of an argument, we need to construct a "comparison table". For a comparison table, whenever ϕ, ϕ_1, ϕ_2, $\cdots$ and ϕ_n are true, ψ is always true, we have a valid argument; if there is **one case** where ϕ, ϕ_1, ϕ_2, $\cdots$ and ϕ_n are true, ψ is false, the argument is invalid.

Logical Implication (cont)

Example 1.4.9. Determine the validity of the argument

$$p \vee (q \vee r),\ \sim r / \therefore p \vee q.$$

Solution (using comparison table)

p	q	r	$p \vee (q \vee r)$	$\sim r$	$p \vee q$
T	T	T	T	F	T
T	T	F	T	T	T
T	F	T	T	F	T
T	F	F	T	T	T
F	T	T	T	F	T
F	T	F	T	T	T
F	F	T	T	F	F
F	F	F	F	T	F

In each assignment where the **premises are both true**, the **conclusion is also true**, so the argument is valid.

Logical Implication (cont)

Example 1.4.10

Determine the validity of the argument $p \to q \vee \sim r,\ q \to p \wedge r \ /\ \therefore p \to r$ by using the comparison table.

Practise with Excel?

Example 1.4.12

Is the following argument valid?

If interest rates are going up, stock market prices will go down.
Interest rates are not going up.
Therefore stock market prices will not go down.

Logical Implication (cont)

The validity of an argument using a 'truth table' is founded on

Deduction Theorem

Let Γ be any finite (possible empty) set of formulas, ϕ, $\phi_1, \cdots, \phi_n$ and ψ be formulas.

1. $\Gamma, \phi \models \psi$ iff $\Gamma \models \phi \rightarrow \psi$.
2. $\phi_1, \cdots, \phi_n \models \psi$ iff $\phi_1 \wedge \cdots \wedge \phi_i, \cdots, \phi_n \models \psi$ where $1 < i \leq n$.

Based on this theorem, an argument $\phi_1, \cdots, \phi_n \ / \ \therefore \psi$ is valid iff $\models \phi_1 \wedge \cdots \wedge \phi_n \rightarrow \psi$; otherwise, the argument is invalid.

Logical Implication (cont)

Example 1.4.15: Determine if $p \vee q / \therefore p \wedge q$ is valid.

Solution

The truth table for the argument is listed below.

p	q	$p \vee q$	$p \wedge q$	$(p \vee q) \rightarrow (p \wedge q)$
T	T	T	T	T
T	F	T	F	F
F	T	T	F	F
F	F	F	F	T

According to the last column the statement $(p \vee q) \rightarrow (p \wedge q)$ is not a tautology. By Deduction Theorem, the argument is invalid.

Logical Implication (cont)

Final Exam May 2019 Q3(a): Use **truth table** to explain whether the following argument is valid or invalid:

$$\begin{array}{rr} & (p \vee q) \rightarrow (p \wedge q) \\ & \sim (p \vee q) \\ \hline \therefore & \sim (p \wedge q) \end{array}$$

(9 marks)

Logical Implication (cont)

Lecturer's Marking Guide

The truth table is

p	q	$(p \vee q) \rightarrow (p \wedge q)$	$\sim (p \vee q)$	$\sim (p \wedge q)$
T	T	T	F	F
T	F	F	F	T
F	T	F	F	T
F	F	T	T	T

.....[$4 \times 2 = 8$ marks]

We observe that when the premises are true (row 4), the conclusion is true, therefore, the argument is **valid**.

.....[1 mark]

Logical Implication (cont)

Example 1.4.16: Show that the argument $\sim p \rightarrow F / \therefore p$ is valid using truth table.

Class discussion.

Logical Implication (cont)

Similar to the LE (Logical Equivalence) Rules in Slide 47, we can summarise a list "LI (Logical Implication Rules" for valid arguments which can be used in semantic logical reasoning.

Theorem 1.4.17 (Logical Implication Rules)

Let F be contradiction. Given any atomic statements p, q and r, the following arguments are valid.

1. Modus Ponens (MP in short):
$$p \to q,\ p \models q$$
2. Modus Tollens (MT in short):
$$p \to q,\ \sim q \models \sim p$$

Logical Implication (cont)

Logical Implication Rules (cont)

3. Generalisation:
$$p \models p \vee q$$
$$q \models p \vee q$$
4. Specialisation:
$$p \wedge q \models p$$
$$p \wedge q \models q$$
5. Conjunction: $p,\ q \models p \wedge q$
6. Elimination:
$$p \vee q,\ \sim q \models p$$
$$p \vee q,\ \sim p \models q$$
7. Transitivity: $p \rightarrow q,\ q \rightarrow r \models p \rightarrow r$

Logical Implication (cont)

Logical Implication Rules (cont)

7. Contradiction Rule: $\sim p \rightarrow F \models p$

The following theorems lay down the foundation for the application of the LI rules.

Substitution of Equivalence

If $\xi \equiv \phi_n$ and $\phi_1, \cdots, \phi_{n-1}, \phi_n \models \psi$, then $\phi_1, \cdots, \phi_{n-1}, \xi \models \psi$.

"Partial Ordering" Theorem

Let ϕ_i, ψ_j, ξ be formulas.

1. $\phi_1, \cdots, \phi_n \models \phi_i$ for each $i = 1, \cdots, n$.
2. If $\phi_1, \cdots, \phi_i \models \psi_j$, where $j = 1, \cdots, p$, and $\psi_1, \cdots, \psi_p \models \xi$, then $\phi_1, \cdots, \phi_n \models \xi$.

Logical Implication (cont)

Example 1.4.20: Show that the hypotheses

- ϕ_1: "If John takes the computer course, then John stays in the hostel"
- ϕ_2: "John does not stay in the hostel"
- ϕ_3: "If John does not take the computer course, then John takes the language course or stay at home"
- ϕ_4: "If John takes language course then John buys a motorcycle"
- ϕ_5: "If John buys a car, then John does not buy motorcycle"
- ϕ_6: "John buys a car"

lead to the conclusion ξ: "John stays at home".

Logical Implication (cont)

Example 1.4.20 (cont)

Solution: Let

r_1: John takes the computer course
r_2: John stays in the hostel
r_3: John takes the language course
r_4: John stays at home
r_5: John buys a motorcycle
r_6: John buys a car

The above argument can be formally written as

$$r_1 \to r_2,\ \sim r_2,\ \sim r_1 \to r_3 \vee r_4,\ r_3 \to r_5,\ r_6 \to \sim r_5,\ r_6 \ /\ \therefore r_4.$$

Logical Implication (cont)

Example 1.4.20 Solution (cont):

Instead of using a comparison table, we use the logical equivalence and logical implication rules together with the "Partial Ordering" Theorem to show that the above argument is valid.

ϕ_1 :	$r_1 \rightarrow r_2$	premise
ϕ_2 :	$\sim r_2$	premise
ϕ_3 :	$\sim r_1 \rightarrow r_3 \vee r_4$	premise
ϕ_4 :	$r_3 \rightarrow r_5$	premise
ϕ_5 :	$r_6 \rightarrow \sim r_5$	premise
ϕ_6 :	r_6	premise
ψ_1 :	$\sim r_1$	ϕ_1, ϕ_2, Modus Tollens
ψ_2 :	$r_3 \vee r_4$	ϕ_3, ψ_1, Modus Ponens
ψ_3 :	$\sim r_5$	ϕ_5, ϕ_6, Modus Ponens
ψ_4 :	$\sim r_3$	ϕ_4, ψ_3, Modus Tollens
ξ :	r_4	ψ_2, ψ_4, Elimination

Outline

Rules of Inference

Another way to "infer" a statement is to use ***natural deduction*** (or ***syntactic implication***, see `https://en.wikipedia.org/wiki/Natural_deduction`). In natural deduction, we have such a collection of *rules of inference*. They allow us to infer a conclusion (a formula) from a set of premises called premises by applying inference rules in succession (called a *proof*).

Using rules to 'proof' is developed into a mathematical subject called 'proof theory'.

Popular notation for natural deduction are

- Gentzen syntax (Coq)
- *box-and-line* syntax (used in Jape, `http://japeforall.org.uk`)
- `https://en.wikipedia.org/wiki/Fitch_notation` (to be used in this course)

Rules of Inference (cont)

Let ϕ, ψ, ξ be any statements. Mathematicians have identified the following essential rules of inference for natural deduction:

1. $\rightarrow$-introduction: $\boxed{\phi, \cdots, \psi} \vdash (\phi \rightarrow \psi)$
2. $\rightarrow$-elimination: $\phi \rightarrow \psi, \ \phi \vdash \psi$
3. $\wedge$-introduction or Conjunction: $\phi, \ \psi \vdash \phi \wedge \psi$
4. $\wedge$-elimination or Conjunctive Simplification:
   $\phi \wedge \psi \vdash \phi$ or $\phi \wedge \psi \vdash \psi$

Rules of Inference (cont)

5. $\vee$-introduction: $\phi \vdash \phi \vee \psi$ or $\psi \vdash \phi \vee \psi$
6. $\vee$-elimination: $\phi \vee \psi, \boxed{\phi, \cdots, \xi}, \boxed{\psi, \cdots, \xi} \vdash \xi$
7. $\neg$-introduction or $\sim$-introduction: $\boxed{\phi, \cdots, \perp} \vdash \sim \phi$
8. $\neg$-elimination or $\sim$-elimination: $\sim\sim \phi \vdash \phi$
9. $\perp$-introduction: $\phi, \sim \phi \vdash \perp$
10. $\perp$-elimination: $\perp \vdash \phi$

Note that 1. and 2. are called the *rules of conjunction*, 3. and 4. are called the *rules of implication*, 5. and 6. are called the *rules of disjunction*, 7. and 8. are called the *rules of negation*.

Rules of Inference (cont)

Example 1.5.1: Show that $p \to q \vdash p \to (p \wedge q)$.

Proof

1	$p \to q$	premise
2	$\quad p$	assumption
3	$\quad q$	1,2 $\to$-elimination
4	$\quad p \wedge q$	2,3 $\wedge$-introduction
5	$p \to p \wedge q$	2,4 $\to$-introduction

Lines 2–4 serve to justify line 5, but they cannot be used in any subsequent line of the proof, they are closed off from the rest of the proof, but you are free to use line 5 as you need it.

Rules of Inference (cont)

Representing “arguments” (refer to Slide 65)

$$\{\phi_1, \phi_2, \cdots, \phi_n\} \; / \; \therefore \psi$$

in Coq:

$$\phi_1 \rightarrow \phi_2 \rightarrow \cdots \rightarrow \phi_n \rightarrow \psi.$$

There is ‘no’ implication introduction in Coq, only subgoals.

```
Theorem impl_elim: forall P Q : Prop,
  (P -> Q) -> P -> Q.
Proof.
  intros P Q H1 H2.
  apply H1 in H2 as HQ.
  exact HQ.
Qed.
```

Rules of Inference (cont)

```
Theorem conj_intro : forall P Q : Prop,
  P -> Q -> (P /\ Q).
Proof.
  intros P Q HP HQ.
  split.
  exact HP.
  exact HQ.
Qed.

Theorem conj_elim_left : forall P Q : Prop,
  P /\ Q -> P.
Proof.
  intros P Q H1.
  inversion H1 as [HP HQ].  (* Alternative: destruct *)
  exact HP.
Qed.
```

Rules of Inference (cont)

```
Theorem disj_intro: forall P Q : Prop,
  P -> P \/ Q.
Proof.
  intros P Q H1.
  (* Since it is an 'or', we only need to show 1 goal *)
  left.
  apply H1.
Qed.

Theorem disj_elim: forall P Q R: Prop,
  P \/ Q -> (P -> R) -> (Q -> R) -> R.
Proof.
  intros P Q R H1 H2 H3.
  destruct H1 as [HP | HQ].
  apply H2 in HP. assumption.
  apply H3 in HQ. assumption.
Qed.
```

Rules of Inference (cont)

¬-introduction is just the definition of $\sim \phi$ in Coq and ¬-elimination is defined as NNPP in the Coq library Classical.

```
Theorem bot_intro: forall P Q: Prop,
  False -> Q.
Proof.
  intros P Q.
  contradiction.
Qed.

Theorem bot_elim: forall P: Prop,
 P -> ~P -> False.
Proof.
  intros P HP HNP.
  unfold not in HNP.
  apply HNP in HP.
  contradiction.
Qed.
```

Rules of Inference (cont)

Example 1.5.1 (variation): Show $p \rightarrow q \vdash p \rightarrow (p \wedge q)$ using Coq.

Computer Proof using Coq

```
Example eg_1_5_1 : forall P Q : Prop,
 (P -> Q) -> (P -> (P /\ Q)).
Proof.
 intros P Q H1.
 intro H2.
 apply H1 in H2 as HQ.
 split.
 exact H2.
 exact HQ.
Qed.
```

Rules of Inference (cont)

Example 1.5.2 (Modus Tollens): Show that

$p \to q, \sim q \vdash \sim p$.

Note that when we see the symbol $\vdash$, we can only use the rules of inference (8 of them only).

Proof

1	$p \to q$	premise
2	$\sim q$	premise
3	$\quad p$	assumption
4	$\quad q$	1, 3 $\to$-introduction
5	$\quad \bot$	2, 4 $\bot$-introduction
6	$\sim p$	$\neg$-introduction

Rules of Inference (cont)

Final Exam May 2019 Q3(b): Infer the argument

$$p \vee q,\ p \rightarrow r,\ \sim s \rightarrow \sim q \vdash r \vee s$$

syntatically by stating the **rules of inference** in each step. (6 marks)

Lecturer's Marking Guide

The p-assumption ..... [2 marks]
The q-assumption ..... [3 marks]
Line 12 ..... [1 mark]

Rules of Inference (cont)

Lecturer's Marking Guide (cont)

1	$p \vee q$	premise
2	$p \rightarrow r$	premise
3	$\sim s \rightarrow \sim q$	premise
4	$\quad p$	assumption
5	$\quad r$	2,4 $\rightarrow$E
6	$\quad r \vee s$	5 $\vee$I
7	$\quad q$	assumption
8	$\quad\quad \sim s$	assumption
9	$\quad\quad \sim q$	3,8 $\rightarrow$E
10	$\quad\quad \bot$	7,9 $\neg$E
11	$\quad s$	8–10 $\bot$E & $\neg$E
12	$\quad r \vee s$	11 $\vee$I
13	$r \vee s$	1,4–6,6–12 $\vee$E

Rules of Inference (cont)

```
Require Import Classical.
(* https://stackoverflow.com/questions/14644086/proving-p-q-q-p-using-coq
Lemma contrapositive:
 forall P Q : Prop,
  (~ P -> ~ Q) -> (Q -> P).
Proof.
 intros.
 apply NNPP.
 intro.
 apply H in H1.
 contradiction.
Qed.
Example FinalExamMay2019Q3b:
 forall P Q R S: Prop,
  P \/ Q -> (P -> R) -> (~ S -> ~ Q) -> R \/ S.
Proof.
 intros.
 destruct H as [HP | HQ].
 - apply H0 in HP. left. exact HP.
 - apply (contrapositive S Q) in H1.
   right. exact H1. exact HQ.
Qed.
```

Rules of Inference (cont)

Example 1.5.3 (Disjunctive Syllogism): Show that $p \vee q, \sim p \vdash q$; $p \vee q, \sim q \vdash p$.

Exercise.

Rules of Inference (cont)

Example 1.5.4 (Hypothetical Syllogism or Transitivity)
Show that $p \rightarrow q,\ q \rightarrow r \vdash p \rightarrow r$.

Exercise.

Rules of Inference (cont)

Example 1.5.5: By using the rules of inference, show that $\sim p \vee q,\ q \rightarrow r,\ q \rightarrow s,\ p \ \vdash r \wedge s$.

Exercise.

Outline

Outline

Predicates

Propositional logic has a simple syntax and simple semantics. It suffices to illustrate the process of inference but it quickly becomes impractical, even for very small worlds. For example, to describe the commutativity of two natural numbers, we need **infinite** propositions:

$$1 + 2 = 2 + 1, 1 + 3 = 3 + 1, \cdots .$$

That's too many!

Predicates (cont)

To deal with “infinite” facts of this form, the notions of variables and predicates as well as quantifiers are required.

With these notions, the infinite propositions can be summarised as a quantified statement with an equality predicate and an addition function.

$$\forall x \forall y (x + y = y + x)$$

But what about x and y?

Predicates (cont)

There are two ways to make sense of x and y:

- Model theory $\rightarrow$ they come from certain 'world' (natural numbers, integers, real numbers, complex numbers, etc.) $\rightarrow$ semantics of logic.
- Type Theory (where Coq is based).

Predicates (cont)

The mathematical formalism of predicate is stated below.

Definition

Terms and well-formed formulae are defined recursively:

- ***terms***: they represent objects, i.e.
  - Variables (normally denoted $x, y, x_1, x_2, \cdots, y_1, \cdots$) and constants (normally denoted as $a, b, a_1, a_2, \cdots$) are ***atomic terms***.
  - If f is a function (which returns a value) of *degree* (or *arity*) r and $t_1, \cdots, t_r$ are terms, then $f(t_1, \cdots, t_r)$ is a ***term***.

Predicates (cont)

Definition (cont)

- ***formulae***: they are relations or functions, i.e.
  - If P is a *relation* (refer to the Topic Sets and Relations) of *degree (or arity)* r and $t_1, \cdots, t_r$ are terms, then $P(t_1, \cdots, t_r)$ is an ***(atomic) formula***.
  - If ϕ and ψ are formulae (abbreviated notation!), and x is a variable, then the expressions

$$\sim(\phi),\ (\phi) \wedge (\psi),\ (\phi) \vee (\psi),\ (\phi) \rightarrow (\psi),\ (\phi) \leftrightarrow (\psi),$$
$$\forall x(\phi),\ \exists x(\phi)$$

    are ***formulae***.

Predicates (cont)

In English, the "predicate" is the part of the sentence that tells us something about the subject.

Example 1.6.4

live(x, y): x lives in y. . . . . predicate (of degree / arity 2)
live(Mary, Austin): "Mary lives in Austin." proposition (or predicate of degree / arity 0)

iscomplex(x): x is a complex number. . . . . predicate (of degree / arity 1)
Racket: complex?

Quantified Statements

Example 1.6.5: http://en.wikipedia.org/wiki/First-order_logic

The expression
$\forall x \forall y(P(f(x)) \rightarrow\sim (P(x) \rightarrow Q(f(y), x, z)))$ is a formula where f is a unary function symbol, P a unary predicate symbol, and Q a ternary predicate symbol.

Example 1.6.6

The expression $\forall x\ x \rightarrow$ is *not* a formula. It is just a string of symbols!

Quantified Statements (cont)

Example 1.6.9

Consider the following logic formula

$$\underbrace{(\forall x.\exists y.\underbrace{P(f(c,y))}_{\phi_2}\wedge\underbrace{Q(g(g(x)),y)}_{\psi_2})}_{\phi_1}\rightarrow\underbrace{(\exists z.\forall w.\sim\underbrace{R(z,w)}_{\phi_3})}_{\psi_1}$$

we can study the formula formally by applying the rules mentioned in Definition in Slide 102. We can also identify

- function symbols: f, c, g.
- predicate symbols: P, Q, R.

Quantified Statements (cont)

Similar to propositional calculus, conventions have been developed about the precedence of the logical operators, to avoid the need to write parentheses in some cases:

- $\sim$ is evaluated first
- $\wedge$ and $\vee$ are evaluated next
- Quantifiers $\forall$ and $\exists$ are evaluated next
- $\rightarrow$ and $\leftrightarrow$ are evaluated last.

Quantified Statements (cont)

In a formula, a *variable* may occur *free* or *bound*. Intuitively, a variable is free in a formula if it is not quantified: in $\forall y\, P(x, y)$, variable x is free while y is bound.

Definition

The free and bound variables of a formula are defined inductively as follows.

- Atomic formulas. If ϕ is an atomic formula then x is free in ϕ if and only if x occurs in ϕ. Moreover, there are no bound variables in any atomic formula.

Quantified Statements (cont)

Definition (cont)

- Negation. x is free in $\sim \phi$ if and only if x is free in ϕ. x is bound in $\sim \phi$ if and only if x is bound in ϕ.
- Binary connectives. x is free in $(\phi \to \psi)$ if and only if x is free in either ϕ or ψ. x is bound in $(\phi \to \psi)$ if and only if x is bound in either ϕ or ψ. The same rule applies to any other binary connective $\wedge$, $\vee$, $\leftrightarrow$ in place of $\to$.
- Quantifiers. x is free in $\forall y \phi$ if and only if x is free in ϕ and x is a different symbol from y. Also, x is bound in $\forall y \phi$ if and only if x is y or x is bound in ϕ. The same rule holds with $\exists$ in place of $\forall$.

Quantified Statements (cont)

Example 1.6.11

Determine if the variables x, y, z, w are free, bound or neither in the formula $\forall x \forall y(P(x) \rightarrow Q(x, f(x), z))$.

Solution

- x and y are bound variables
- z is a free variable
- w is neither because it does not occur in the formula.

Quantified Statements (cont)

Freeness and boundness are important because of variables can appear multiple times.

Example 1.6.12

For the formula $P(x) \rightarrow \forall x\, Q(x)$, the first occurrence of x is free while the second is bound, i.e. the x in $P(x)$ is free while the x in $\forall x\, Q(x)$ is bound.

Racket Programming Example

```
(let ((x 3))
  (define (f x) (+ (* 2 (sin x)) 1))
  (f x))
```

Informal & Formal Translation

We use informal languages such as English, Chinese, Malay, etc. in real life. Informal languages are **difficult for computer to process**. To ask computer to help us solve problems, we need to express human knowledge in a formal language. The overall process:

> *Informal Problem* → *Formal Logic Formula* → *Computer* → *Perform 'computation'* → *Answer in 'Abstract Representation'* → *Convert to Informal Human Language.*

Informal & Formal Translation (cont)

Racket Example — Formal Langage

```
(let ((x 3))
  (define (f x) (+ (* 2 (sin x)) 1))
  (f x))
```

Informal Language

Let x be 3 and $f(x) = 2 \sin x + 1$. Calculate $f(x)$ (same thing as calculate f(3) but computer needs to know this using variable bounding).

Informal & Formal Translation (cont)

In mathematics, we have many statements which can be formalised into various forms defined below.

Definition 1.7.1

Let $P(x)$ and $Q(x)$ be any predicates with free variable x.

1. A *universal statement* is a formula of the form

$$\forall x P(x)$$

It corresponds to these English sentences: "For all x $P(x)$", "For every x $P(x)$" and "For any x $P(x)$".

Informal & Formal Translation (cont)

Definition 1.7.1 (cont)

2. An *existential statement* is a formula of the form
$$\exists x Q(x).$$
It corresponds to these English sentences: "There is/exists an x such that $P(x)$", "There is at least one x such that $P(x)$", "Some x satisfies P", "$P(x)$ for some x", etc.
3. A *universal conditional statement* is a formula of the form
$$\forall x(P(x) \rightarrow Q(x)).$$
It corresponds to the English sentence "For every x with property P, $Q(x)$".

Informal & Formal Translation (cont)

Example:

- Universal statement:

$$\forall x(x^2 + 1 > 0)$$

  Informal: The square of a number plus one is always larger than zero.

- Existential statement:

$$\exists x(2 \sin x + 1 = 0)$$

  Informal: The equation $2 \sin x + 1 = 0$ has a solution.

- Universal conditional statement:

$$\forall n((n \geq 1) \rightarrow (1 + \cdots + n = \frac{n(n+1)}{2}))$$

  Informal: Summing a sequence from 1 to n is the same as the half of the product n and $n + 1$.

Informal & Formal Translation (cont)

Example 1.7.3: Rewrite the following formal statements in a variety of equivalent but more informal ways without using the formal symbols $\forall$ and $\exists$.

1. $\forall x(x \in \mathbb{R} \rightarrow x^2 \geq 0)$.

Solution: Any of the following English sentences are acceptable.

- All real numbers have *nonnegative* squares.
- Every real number has a nonnegative square.
- Any real number has a nonnegative square.
- The square of any real number is nonnegative.

2. $\forall x(x \in \mathbb{R} \rightarrow x^2 \neq -1)$.
3. $\exists m(m \in \mathbb{Z} \wedge m^2 = m)$.

Informal & Formal Translation (cont)

Example 1.7.4: Rewrite the following formal statement in a variety of informal ways. Do not use quantifiers or variables.

$$\forall x((x \in \mathbb{R}) \wedge (x > 2) \rightarrow (x^2 > 4)).$$

Informal & Formal Translation (cont)

It is sometimes tricky to translate from an informal sentence into a formal sentence, precisely because the informal sentence may not correspond obviously to the logical quantifiers and operators.

Example 1.7.5

Translate the sentence 'Some birds can fly' into logic. Let $B(x)$ mean 'x is a bird' and $F(x)$ mean 'x can fly'.

Solution: 'Some birds can fly' is translated as

$$\exists x(B(x) \wedge F(x)).$$

If we translate this back to 'unpolished' English, it would be 'there is something which is a bird and this thing can fly.'

Informal & Formal Translation (cont)

Warning! A common pitfall is to translate 'Some birds can fly' as

$$\exists x(B(x) \to F(x)) \quad \textbf{Wrong translation!}$$

To see why this is wrong, let p be a frog that somehow got into the universe. Now $B(p)$ is false, so $B(p) \to F(p)$ is true (remember $F \to F \equiv T$). This is just saying 'If that frog were a bird then it would be able to fly', which is true; it doesn't mean the frog actually is a bird, or that it actually can fly. However, we have now found a value of x — namely the frog p — for which $B(x) \to F(x)$ is true, and that is enough to satisfy $\exists x(B(x) \to F(x))$, which is different from $\exists x(B(x) \wedge F(x))$ which requires us to find a flying bird from birds.

Example 1.7.7

Let $A(x)$:= x is an animal.
$C(x)$:= x is a cat.
$S(x)$:= x is small.
$GP(x)$:= x is a good pet.

The English sentences in on the left side of the table below can be translated into logic formulae on the right side of the table:

Informal	Formal
Cats are animals.	$\forall x(C(x) \rightarrow A(x))$
Cats are small.	$\forall x(C(x) \rightarrow S(x))$
Cats are small animals.	$\forall x(C(x) \rightarrow S(x) \wedge A(x))$
Small animals are good pets.	$\forall x(S(x) \wedge A(x) \rightarrow GP(x))$

Informal & Formal Translation (cont)

Many mathematical quantified statements can be written in one of the form in Definition 1.7.1. However, mathematical writings normally have many implicitly quantified statements, i.e. statements that do not contain the "keywords" *all* or *every* or *any* or *each* or *exist*, which often tell us what class of quantified statements they belong. Hence, we have to look at the sentence and understand the context.

Example 1.7.10: Translate the sentence "Every integer is rational." into a quantified statement.

Solution

$\forall x(x \in \mathbb{Z} \rightarrow x \in \mathbb{Q})$.

Informal & Formal Translation (cont)

Example 1.7.11: Consider the statement "There is an integer that is both prime and even." Let $P(n)$ be "n is prime" and $E(n)$ be "n is even." Use the notation $P(n)$ and $E(n)$ to rewrite this statement formally.

Class Discussion.

Informal & Formal Translation (cont)

In calculus courses, the letter x is mostly always used to indicate a real number, hence the sentence below

"If $x > 2$ then $x^2 > 4$."

is interpreted to mean the same as the statement

"For all real numbers x, if $x > 2$ then $x^2 > 4$."

or formally $\forall x((x \in \mathbb{R} \wedge x > 2) \rightarrow (x^2 > 4))$. or

$$\forall x(x \in \mathbb{R}) \rightarrow (x > 2) \rightarrow (x^2 > 4).$$

Informal & Formal Translation (cont)

Example 1.7.13: Translate the following statement formally.

If a number is an integer, then it is a real number.

Example 1.7.14: Let $E(x)$ be the predicate "x is even". Translate the following statement

The number 24 can be written as a sum of two even integers.

into a formal logic formula.

Class Discussion.

Informal & Formal Translation (cont)

Example 1.7.17: Rewrite the following statements formally using quantifiers and variables.

1. Somebody trusts everybody.
2. Given any positive number, there is another positive number that is smaller than the given number.
3. Any even integer equal twice some other integer.
4. For every real number x and every real number y, if x is positive and y is negative, then the product of x and y is negative.

Class Discussion.

Informal & Formal Translation (cont)

Let S be the set of students in UTAR, M be the set of movies that have ever been released, and let $V(s, m)$ denote "student s has seen movie m". Translate each of the following statements into English.

1. $\forall s \exists m((s \in S) \rightarrow ((m \in M) \wedge V(s, m)))$.
2. $\exists s \exists t \exists m((s \in S) \wedge (t \in S) \wedge (m \in M) \wedge (s \neq t) \wedge V(s, m) \wedge V(t, m))$.
3. $\exists s \exists t((s \in S) \wedge (t \in S) \wedge (s \neq t) \wedge \forall m((m \in M) \rightarrow (V(s, m) \rightarrow V(t, m))))$.

Class Discussion.

Informal & Formal Translation (cont)

Often the real difficulty in translating English into logic is in figuring out what the English says, or what the speaker meant to say. For example, many people make statements like 'All people are not rich'. What this statement actually says is

$$\forall x \sim R(x),$$

where the universe is the set of people and $R(x)$ means 'x is rich'. What is usually meant, however, by such a statement is

$$\sim \forall x R(x) \text{ or equivaliently} \exists x \sim R(x),$$

that is, not all people are rich (i.e., some people are not rich)

Such problems of *ambiguity* or *incorrect grammar* in English cannot be solved mathematically and they will not be considered in this subject. However, they do illustrate one of the benefits of mathematics: simply translating a problem from English into formal logic may expose confusion or misunderstanding.

Outline

Semantics of Quantified Statements

Syntax can be used to describe if a string is a valid formula.

This is just like writing a program in some programming language. One can check whether the syntax of the program is correct. However, even if the program is correct in syntax, it doesn't mean that the program is "meaningful" to the user. E.g.

```
int main()
{
    1 + 2 == 3;
}
```

We know that one needs to "compile" a program so that it can "run" on a computer with "useful" effects.

Semantics (cont)

"Compiling" a program can be viewed as introducing "semantics" or "interpretation" to a program.

To determine if a logical formula is true or false (as an extension of the propositional logic), we need to introduce "semantics" to the formula.

In the semantics of propositional logic, we **assigned** a truth value to each atom. In predicate logic, the smallest unit to which we can assign a truth value is a predicate $P(t_1, \cdots, t_n)$ applied to terms. But we cannot arbitrarily assign a truth value, as we did for propositional atoms. There needs to be some consistency. The way to specify an **interpretation** (or **semantic**) is to specify a **model**.

Semantics (cont)

Definition 1.8.1: A ***model*** (or ***structure***) M consists of a nonempty set D that forms the *domain of discourse* (or *universe of discourse*) and an *interpretation* $()^M$, which is a mapping such that

- Each function symbol f of degree n is assigned a function f^M from D^n to D. In particular, each constant symbol is assigned an individual in the domain of discourse.
- Each predicate symbol P of degree n is assigned a relation P^M over D^n, i.e. $P^M \subset D^n$.

In lay man term: With a model, we can **assign** values to a valid predicate to determine its 'truth' value.

Semantics (cont)

Definition 1.8.2: Let M be a model and t be a term without variables. Then t^M, the interpretation of t in M, is given as follows:

- if t is a constant c, then $t^M = c^M$, $c^M \in D$;
- if t is a variable x, then $t^M = \sigma(x)$, where σ is a variable assignment;
- if t is an n-ary function $f(t_1, \cdots, t_n)$, then $t^M = f^M(t_1^M, \cdots, t_n^M)$.

For predicates and logical connectives, the interpretation goes through the process *T-schema*:

Semantics (cont)

Definition 1.8.2 (cont):

- Atomic formulae.
  1. A formula $P(t_1, \cdots, t_n)$ is associated the value T or F depending on whether $(t_1^M, \cdots, t_n^M) \in P^M \subset D^n$.
  2. A formula $t_1 = t_2$ is assigned T if $t_1^M = t_2^M$, i.e. they evaluate to the same object of the domain of discourse.
- Logical connectives. A formula in the form $\sim \phi$, $\phi \rightarrow \psi$, etc. is evaluated according to the truth table for the connective in question, as in propositional logic.

Semantics (cont)

Definition 1.8.2 (cont):

- Existential quantifiers. A formula $\exists x \phi(x)$ is true according to M and σ if there exists an evaluation σ' of the variables that only differs from σ regarding the evaluation of x and such that ϕ is true according to the interpretation M and the variable assignment σ'. This formal definition captures the idea that $\exists x \phi(x)$ is true if and only if there is a way to choose a value for x such that $\phi(x)$ is satisfied.

Semantics (cont)

Definition 1.8.2 (cont):

- Universal quantifiers. A formula $\forall x \phi(x)$ is true according to M and σ if $\phi(x)$ is true for every pair composed by the interpretation M and some variable assignment σ' that differs from σ only on the value of x. This captures the idea that $\forall x \phi(x)$ is true if every possible choice of a value for x causes $\phi(x)$ to be true.

If a formula does not contain free variables, and so is a sentence, then the initial variable assignment does not affect its truth value. In other words, a sentence is true according to M and σ if and only if is true according to M and every other variable assignment σ'.

Semantics (cont)

Example 1.8.3: Consider the sentence "None of Alma's lovers' lovers love her." Let "Alma" can be a constant a, and the concept "x loves y" can be a binary predicate $L(x, y)$.

1. Formalise this sentence.
2. Consider the model M defined by $D = \{p, q, r\}$, $a^M = p$, $L^M = \{(p, p), (q, p), (r, p)\}$. Determine the truth value of the sentence under this model M.

Semantics (cont)

Example 1.8.3 Solution:

1. We can formalise the sentence as

$$\forall x \forall y (L(x, a) \wedge L(y, x) \rightarrow \sim L(y, a)).$$

Intuitively, $L(x, a)$ says that x is Alma's lover, and $L(y, x)$ says that "y loves x", so $L(x, a) \wedge L(y, x)$ says that y is one of Alma's lovers' lovers. The formula $\sim L(y, a)$ says that y does not love Alma, and the quantifiers make sure this is true for any x, y.

Semantics (cont)

Example 1.8.3 Solution:

2. In order to interpret the formula, we need a variable assignment σ. Consider $\sigma = \{(x, p), (y, q)\}$. Then

$$
\begin{aligned}
&(\forall x \forall y (L(x, a) \wedge L(y, x) \rightarrow\sim L(y, a)))^M(\sigma) \\
&\equiv L^M(\sigma(x), a^M) \wedge L^M(\sigma(y), \sigma(x)) \rightarrow\sim L^M(\sigma(y), a^M) \\
&\equiv L^M(p, p) \wedge L^M(q, p) \rightarrow\sim L^M(q, p) \equiv T \wedge T \rightarrow\sim T \equiv T \rightarrow F \equiv F.
\end{aligned}
$$

The formula is false under this model.

To show that a universal formula is true, we need to make sure that it is true for all possible variable assignments.

Logical Equivalence

In comparison to propositional logic in which the truth assignment leads to truth table (or comparison table) as a convenient tool to verify the truth value, the situation in predicate logic is much more complicated. The truth assignment extends to models and to verify the tautology of a formula requires it to be true under all models.

The laws of logical equivalences and “substitution principle” (Slide 53) are also valid for predicate calculus. However, we need to replace the term “atomic statement” with “formula” and introduce the extended notion of logical equivalence.

Logical Equivalence (cont)

Definition 1.8.7: Let ϕ and ψ be two formulae with free variables $x_1, \cdots, x_n$, they are *logically equivalent*, i.e. $\phi \equiv \psi$ if $\models \forall x_1 \cdots \forall x_n (\phi \leftrightarrow \psi)$.

Revision

Propositional Logic:

- Encode human knowledge (especially mathematics) in logic without variables. E.g. $1 + 2 = 2 + 1$ (commutative).
- Well-formed formula involves atomic statements and connectives ($\sim$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$) in the **proper syntax**.
- Truth-table: The value of the formula for different assignments.
- Special formula: tautology
- Logical equivalence: $\phi \equiv \psi$ means $\phi \leftrightarrow \psi$ is tautology. Application: simplification of passive logic circuit.

Revision (cont)

Propositional Logic (cont): Let ϕ and ψ be any well-formed formulas.

- Logical implication: $\phi \Rightarrow \psi$ means $\phi \rightarrow \psi$ is tautology.
- Rules of inference: Using rules to derive formulas! There is no truth assignment involved.
- Link between "true" formula and rules of inference process (proof theory): All "true" formula can be derived using rules of inference.

Revision (cont)

Predicate Logic:

- To express $x + y = y + x$ in maths, we need infinitely many propositions. This is not acceptable. We need to introduce "variables", so that we have the proposition-valued "function".
- We are used to functions which returns numbers (e.g. sine, cosine), "proposition-valued" function returns "propositions".
- Problem: We have to extend "truth assignment". E.g. If we have two $n \times n$ matrces A and B, then $AB = BA$ is false in general but can sometimes be true.

Revision (cont)

Predicate Logic (cont):

- "Model" (universe or domain for the variables) are introduced to generalise the truth assignment for predicates. E.g. $x + y = y + x$ is true for the "integer" domain, "real number" domain but false for string domain. E.g. x="Albert", y="Einstein", $x + y$="AlbertEinstein" but $y + x$="EinsteinAlbert" are different.
- First order predicate logic studies all the formulas which are true under all models (generalisation of tautology)
- So, we can define logical equivalence and logical implication, but the rules are much more complex!!!

Logical Equivalence (cont)

Theorem 1.8.8 (laws of logical equivalence): Let ϕ and ψ be any formulas **with** free variable x (and y). Let ξ be any formula **without** free variable x. Then

(a) $\sim \forall x\phi \equiv \exists x \sim \phi$; ..... (Generalised de Morgan law)

(b) $\sim \exists x\phi \equiv \forall x \sim \phi$; ..... (Generalised de Morgan law)

(c) $\forall x(\phi \wedge \psi) \equiv (\forall x\ \phi) \wedge (\forall x\ \psi)$;

(d) $\exists x(\phi \vee \psi) \equiv (\exists x\ \phi) \vee (\exists x\ \psi)$;

(e) $\forall x \forall y \phi \equiv \forall y \forall x \phi$;

(f) $\exists x \exists y \phi \equiv \exists y \exists x \phi$;

(g) $\xi \equiv \forall x \xi \equiv \exists x \xi$;

(h) Suppose the variable x has no free occurrences in ξ and is substitutable for x in ξ. Then $\forall x\xi \equiv \forall y\xi[y/x]$, and $\exists x\xi \equiv \exists y\xi[y/x]$; ..... (change variable name)

(i) $\forall x(\xi \wedge \psi) \equiv \xi \wedge (\forall x\psi)$; $\exists x(\xi \wedge \psi) \equiv \xi \wedge (\exists x\psi)$; $\forall x(\xi \vee \psi) \equiv \xi \vee (\forall x\psi)$; $\exists x(\xi \vee \psi) \equiv \xi \vee (\exists x\psi)$. ..... (free variable equivalence)

Logical Equivalence (cont)

Corollary 1.8.10: Let ϕ and ψ be any predicates with free variable x. Let ξ be any formula **without** free variable x.Then

1. $\sim (\forall x(\phi \rightarrow \psi)) \equiv \exists x(\phi \wedge \sim \psi)$;
2. $\exists x(\phi \rightarrow \psi) \equiv (\forall x\ \phi) \rightarrow (\exists x \psi)$.
3. $\forall x(\xi \rightarrow \psi) \equiv \xi \rightarrow (\forall x\ \psi)$

Proof:

1. $LHS \equiv \exists x \sim (\phi \rightarrow \psi)) \equiv \exists x \sim (\sim \phi \vee \psi)) \equiv \exists x(\sim\sim \phi \wedge \sim \psi)) \equiv RHS$
2. $LHS \equiv \exists x(\sim \phi \vee \psi) \equiv (\exists x \sim \phi) \vee (\exists x \psi) \equiv\sim (\forall x \phi) \vee (\exists x \psi) \equiv RHS$
3. $LHS \equiv \forall x(\sim \xi \vee \psi) \equiv (\forall x \sim \xi) \vee (\forall x \psi) \equiv\sim (\exists x \xi) \vee (\forall x \psi) \equiv\sim \xi \vee (\forall x \psi) \equiv RHS$

Logical Equivalence (cont)

Definition 1.8.11: Let ϕ and ψ be formulae with free variable x. Consider a statement of the form

$$\forall x\ (\phi \rightarrow \psi). \tag{1}$$

1. Its *negation* is $\exists x(\phi \wedge \sim \psi)$.
2. Its *contrapositive* form is $\forall x(\sim \psi \rightarrow \sim \phi)$.
3. Its *converse* form is $\forall x(\psi \rightarrow \phi)$.
4. Its *inverse* form is $\forall x(\sim \phi \rightarrow \sim \psi)$.

Logical Equivalence (cont)

Example 1.8.13: Write formal negations for the statement "Every prime number is odd." using the predicate prime(n) for "n is prime" and odd(n) for "n is odd".

Solution

$\sim \forall n(\text{prime}(n) \rightarrow \text{odd}(n)) \equiv \exists n(\text{prime}(n) \wedge \sim \text{odd}(n))$

Logical Equivalence (cont)

Example 1.8.14: Write an informal negation (using formal reasoning) for the statement

If a computer program has more than 100,000 lines, then it contains a bug.

Let program(x) be "x is a computer program", many_lines(x) be "x has more than 100,000 lines" and bug(x) be "x has a bug."

Class Discussion.

Logical Equivalence (cont)

Example 1.8.15: Write the contrapositive, converse and inverse for the following statement formally and informally:

If a real number is greater than 2, then its square is greater than 4.

Class Discussion.

Logical Equivalence (cont)

Definition 1.8.21

A formula of the predicate calculus is in **prenex normal form** if it is written as a string of quantifiers (referred to as the *prefix*) followed by a quantifier-free part (referred to as the *matrix*).

Theorem 1.8.22

Every formula in predicate logic is logically equivalent to a formula in prenex normal form.

Logical Equivalence (cont)

Example 1.8.23 (Wikipedia)

Let $\phi(y)$, $\psi(z)$, and $\rho(x)$ be quantifier-free formulas with the free variables shown.

$$\forall x \exists y \forall z (\phi(y) \vee (\psi(z) \rightarrow \rho(x)))$$

is in prenex normal form with matrix $\phi(y) \vee (\psi(z) \rightarrow \rho(x))$, while

$$\forall x ((\exists y \phi(y)) \vee ((\exists z \psi(z)) \rightarrow \rho(x)))$$

is logically equivalent but not in prenex normal form.

Logical Equivalence (cont)

Example 1.8.24: Write the negation of $\forall x \exists y \exists z P(x, y, z)$ in prenex normal form.

Class Discussion.

Logical Equivalence (cont)

Final Exam May 2019 Q1(d): Let $F(u, x, y)$, $G(y, v)$ and $H(x)$ be predicates. **List** down the steps and the logical equivalent rules to transform the following quantified statement

$$\sim \Big[\forall x \exists y F(u, x, y) \rightarrow \exists x \Big(\sim \forall y G(y, v) \rightarrow H(x)\Big)\Big]$$

to prenex normal form. (6 marks)

Logical Equivalence (cont)

Lecturer's Marking Guide

The steps and rules are listed below:

$$\sim \Big[\forall x \exists y F(u,x,y) \rightarrow \exists x\Big(\sim \forall y G(y,v) \rightarrow H(x)\Big)\Big]$$

$$\equiv \sim \Big[\sim \forall x \exists y F(u,x,y) \vee \exists x\Big(\sim \forall y G(y,v) \rightarrow H(x)\Big)\Big]$$ [Implication law, 0.5 mark]

$$\equiv \forall x \exists y F(u,x,y) \wedge \sim \exists x\Big(\sim \forall y G(y,v) \rightarrow H(x)\Big)$$

[de Morgan law, double negative, 1 mark]

$$\equiv \forall x \exists y F(u,x,y) \wedge \Big[\forall x \sim \Big(\sim \forall y G(y,v) \rightarrow H(x)\Big)\Big]$$

[Generalised de Morgan law, 0.5 mark]

$$\equiv \forall x \exists y F(u,x,y) \wedge \Big[\forall x \sim \Big(\forall y G(y,v) \vee H(x)\Big)\Big]$$

[Implication law, double negative, 1 mark]

$$\equiv \forall x \exists y F(u,x,y) \wedge \Big[\forall x\Big(\forall y \sim G(y,v) \wedge \sim H(x)\Big)\Big]$$

[Generalised de Morgan law, 0.5 mark]

$$\equiv \forall x \exists y F(u,x,y) \wedge \Big[\forall x \forall y\Big(\sim G(y,v) \wedge \sim H(x)\Big)\Big]$$ [Free variable law, 0.5 mark]

$$\equiv \forall x\Big[\exists y F(u,x,y) \wedge \forall y\Big(\sim G(y,v) \wedge \sim H(x)\Big)\Big]$$ [Quantified conjunctive law, 0.5 mark]

$$\equiv \forall x\Big[\exists y F(u,x,y) \wedge \forall z\Big(\sim G(z,v) \wedge \sim H(x)\Big)\Big]$$ [Quantifier renaming law, 0.5 mark]

$$\equiv \forall x \exists y \forall z\Big[F(u,x,y) \wedge \sim G(z,v) \wedge \sim H(x)\Big]$$ [Free variable law, 1 mark]

Logical Equivalence (cont)

Final Exam May 2019 Q2(c): Given the following quantified statement:

$$\forall x \forall y \Big[((x > 0) \wedge (y > 0)) \rightarrow \Big(\sqrt{x + y} = \sqrt{x} + \sqrt{y} \Big) \Big]. \quad (*)$$

1. Translate the quantified statement into an informal English sentence. (2 marks)
2. Determine whether the quantified statement is true or false in the domain of real numbers. You need to defend your answer. (2 marks)
3. Write down the negation of the quantified statement $(*)$ in prenex normal form. (5 marks)

Logical Equivalence (cont)

Lecturer's Marking Guide

(i) The square root of the sum of two numbers is equal to the sum of the square roots of the two numbers

(ii) The quantified statement is *false*. . . . . [1 mark]

To defend, we write a counterexample: Let $x = y = 1$,

$\sqrt{x+y} = \sqrt{2} \neq \sqrt{1} + \sqrt{1} = 2$. . . . . [1 mark]

(iii) By applying the generalised de Morgan law, the negation of $(*)$ is logically equivalent to

$$\exists x \exists y \sim \Big[((x > 0) \wedge (y > 0)) \rightarrow \Big(\sqrt{x+y} = \sqrt{x} + \sqrt{y}\Big)\Big].$$

In prenex normal form, it can be written as

$$\exists x \exists y \Big[(x > 0) \wedge (y > 0) \wedge \Big(\sqrt{x+y} \neq \sqrt{x} + \sqrt{y}\Big)\Big]. \qquad \text{[5 marks]}$$

Logical Equivalence (cont)

Quantifiers in the prenex normal form **may not** be logically equivalent.

$\forall x \exists y P(x, y) \not\equiv \exists y \forall x P(x, y)$

$\forall y \exists x P(x, y) \not\equiv \exists x \forall y P(x, y)$

Example:

- $\forall y \exists x (x > y)$ (for every number, there is a larger number) — true
- $\exists x \forall y (x > y)$ (there is a number large than all numbers) — impossible.

Logical Equivalence (cont)

Exceptions:

- $\forall x \forall y P(x, y) \equiv \forall y \forall x P(x, y)$ (a special cased of Laws of Logical Equivalence (e))
- $\exists x \exists y P(x, y) \equiv \exists y \exists x P(x, y)$ (a special cased of Laws of Logical Equivalence (f))

Example 1.8.25

Let $P(x, y)$ denote "$x + y = 0$". What are the truth values of the quantified statements $\exists y \forall x P(x, y)$ and $\forall x \exists y P(x, y)$?

- $\exists y \forall x P(x, y)$ . . . . . □
- $\forall x \exists y P(x, y)$ . . . . . □

Class Discussion.

Logical Equivalence (cont)

Example 1.8.27: Let $Q(x, y)$ be the statement "$x + y = x - y$". If model M of this formula consist the universe of discourse $D = \mathbb{Z}$ and the operations are the normal operations associated with $\mathbb{Z}$. Determine the truth values of the following formula.

1. $\forall y Q(1, y)$
2. $\exists x \exists y Q(x, y)$
3. $\forall x \exists y Q(x, y)$
4. $\exists y \forall x Q(x, y)$
5. $\forall y \exists x Q(x, y)$

Class Discussion.

Logical Implication

The extension of logical implication from propositional logic to predicte logic is very complicated and we will only list some theoretical results.

Theorem 1.9.1: Let $\Gamma = \{\phi_1, \cdots, \phi_n\}$. If $\Gamma \subset \Gamma'$ and $\Gamma \models \psi$, then $\Gamma' \models \psi$.

(It basically says that if ϕ is true for "preconditions" Γ, it is also true in a larger system Γ' containing Γ.)

Semantic Deduction Theorem: Let $\Gamma = \{\phi_1, \cdots, \phi_n\}$. $\Gamma \cup \{\phi\} \models \psi$ iff $\Gamma \models \phi \rightarrow \psi$.

Theorem 1.9.3: If $P(x)$ is a formula with one free variable x and s and a are closed terms. Then

$$P(s) \models \exists x P(x); \qquad \forall x P(x) \models P(a).$$

Logical Implication (cont)

Proposition 1.9.4: Let ϕ be a formula with free variable x and a and s are terms free with respect to x in ϕ. Then we have the following laws.

1. Universal instantiation (or specialisation): $\forall x \phi \Rightarrow \phi[a/x]$, here $[a/x]$ means "a replaces x". In particular, when ϕ is $P(x)$, we have $\forall x P(x) \Rightarrow P(a)$.
2. Universal generalisation: If the term a in $\phi[a/x]$ is arbitrary, then $\phi[a/x] \Rightarrow \forall x \phi$.
3. Existential instantiation (or specialisation): $\exists x \phi \Rightarrow \phi[s/x]$, here $[s/x]$ means "s replaces x". In particular, when ϕ is $P(x)$, we have $\exists x P(x) \Rightarrow P(s)$.
4. Existential generalisation: For a particular term s, $\phi[s/x] \Rightarrow \exists x \phi$.

Logical Implication (cont)

Theorem 1.9.5: Let p and q be two formula without free variable x, $P(x)$ and $Q(x)$ be formula with free variable x. Then

1. $\exists x(P(x) \wedge Q(x)) \Rightarrow (\exists x\ P(x)) \wedge (\exists x\ Q(x))$
2. $(\forall x P(x)) \vee (\forall x\ Q(x)) \Rightarrow \forall x\ (P(x) \vee Q(x))$
3. $(\exists x P(x)) \rightarrow (\forall x\ Q(x)) \Rightarrow \forall x(P(x) \rightarrow Q(x))$
4. $\forall x(P(x) \rightarrow Q(x)) \Rightarrow (\forall x\ P(x)) \rightarrow (\forall x Q(x))$

Logical Implication (cont)

Non-rigorous proof:

	ϕ_1 :	$\exists x(P(x) \wedge Q(x))$	premise
1.	ψ_1 :	$P(s) \wedge Q(s)$	ϕ_1, existential specialisation
	$\therefore$	$\exists x P(x) \wedge \exists x Q(x)$	ψ_1, existential generalisation

	ϕ_1 :	$(\forall x P(x)) \vee (\forall x\ Q(x))$	premise
	ψ_1 :	$(\forall x P(x)) \vee (\forall y\ Q(y))$	ϕ_1, change variable name
2.	ψ_2 :	$\forall y(\forall x P(x) \vee Q(y))$	ψ_1, free variable equivalence
	ψ_3 :	$\forall y(P(y) \vee Q(y))$	ψ_2, universal instantiation
	ψ_4 :	$\forall x(P(x) \vee Q(x))$	ψ_3, change variable name

Logical Implication (cont)

Non-rigorous proof:

	ϕ_1 :	$(\exists x P(x)) \rightarrow (\forall x\ Q(x))$	premise
	ψ_1 :	$\sim \exists x P(x) \vee (\forall x\ Q(x))$	ϕ_1, implication law
3.	ψ_2 :	$\forall x \sim P(x) \vee (\forall x\ Q(x))$	ψ_1, generalised de Morgan
	ψ_3 :	$\forall x(\sim P(x) \vee Q(x))$	ψ_2, item 2.
	$\therefore$	$\forall x(P(x) \rightarrow Q(x))$	ψ_3, implication law

	ϕ_1 :	$\forall x(P(x) \rightarrow Q(x))$	premise
	ψ_1 :	$P(a) \rightarrow Q(a)$	ϕ_1, universal instantiation
4.	ψ_2 :	$\forall x P(x) \rightarrow P(a)$	universal instantiation???
	ψ_3 :	$\forall x P(x) \rightarrow\ Q(a)$	ψ_1, ψ_2 transitivity
	$\therefore$	$\forall x P(x) \rightarrow \forall x Q(x)$	ψ_3, universal generalisation

Logical Implication (cont)

Example 1.9.9: Rewrite the following argument using quantifiers, variables, and predicate symbols:

If a number is even, then its square is even.
k is a particular number that is even.
$\therefore$ k^2 is even.

Is the argument valid? Why?

Class Discussion

More examples will be discussed in Tutorial Class.

Logical Equivalence (cont)

Final Exam May 2019 Q3(c): Show that the following argument

$$\begin{array}{rl} & \forall x(F(x) \rightarrow \sim G(x)) \\ & \exists x(H(x) \wedge G(x)) \\ \hline \therefore & \exists x(H(x) \wedge \sim F(x) \end{array}$$

is valid using the rules of logical equivalence and implication. (5 marks)

Logical Equivalence (cont)

Lecture's Marking Guide

The semantic deduction is shown below

ϕ_1	$\forall x(F(x) \rightarrow\sim G(x))$	premise
ϕ_2	$\exists x(H(x) \wedge G(x))$	premise
ψ_1	$H(s) \wedge G(s)$	ϕ_2, existential instantiation .....[1 mark]
ψ_2	$F(s) \rightarrow\sim G(s)$	ϕ_1, universal instantiation
ψ_3	$G(s)$	ψ_1, specialisation .....[1 mark]
ψ_4	$\sim F(s)$	ψ_2, ψ_3, MT .....[1 mark]
ψ_5	$H(s)$	ψ_1, specialisation .....[1 mark]
ψ_6	$H(s)\wedge \sim F(s)$	ψ_3, ψ_4 conjunction
$\therefore$	$\exists x(H(x)\wedge \sim F(x)$	ψ_6, existential generalisation .....[1 mark]

Outline

Rules of Inference

Laws of equivalence and laws of logical impliecation are not the modern (computer) approach to "prove" quantified statements.

The foundation to modern automated proof theory (Note that syntactic inference or natural deduction is not the only method for automated proof, sequent calculus is another method).

In predicate logic, in addition to the "propositional" rules of inference (Slide 82), we have the following rules of inference related to quantifiers:

11. $\forall$-introduction: $\boxed{t \rightarrow \phi(t)} \vdash \forall x \phi(x)$
12. $\forall$-elimination: $\forall x \phi(x) \vdash \boxed{t \rightarrow \phi(t)}$

Rules of Inference (cont)

13. $\exists$-introduction: $\phi(t) \vdash \exists x \phi(x)$
14. $\exists$-elimination: $\exists x \phi(x) \vdash \boxed{s,\ \phi(s) \rightarrow \cdots}$

Here, t is a *ground term*, i.e. a term that does not contain any variables, a is a *constant* symbol (which is arbitrary) which does not occur in the conclusion $\forall \phi(x)$ or any assumption in the syntactic derivation, s is a constant symbol which does not occur in the premise $\exists x \phi(x)$, conclusion ξ or any assumptions in the syntactic derivation.

Note that $\vdash$ indicates logical entailment based on the rules of inference (there are only 12 of them) instead of the laws based on the logic semantics.

Rules of Inference (cont)

The $\forall$-introduction rule is non-constructive and probably cannot be 'proved' in Coq (at least I don't know how to do so):

```
Parameter X : Type.

Example forall_intro:
 forall (P : X->Prop) (t : X),
  P t -> (forall x : X, P x).
Admitted.
(*
Proof.
 intros P t.
 intro H.
 How to generalize dependent t ???
Qed.
*)
```

Rules of Inference (cont)

```
Example forall_elim:
 forall (P : X->Prop) (t : X),
  (forall x : X, P x) -> P t.
Proof.
 intros P t.
 intro H.
 specialize (H t).
 exact H.
Qed.
```

Rules of Inference (cont)

```
Parameter s : X.
(* s is a particular value such that P(s) is True *)

Example exists_intro:
 forall P : X->Prop,
  (P s -> (exists x : X, P x)).
Proof.
 intros P H.
 exists s.
 exact H.
Qed.

Example exists_elim:
 forall P : X->Prop,
  (exists x : X, P x) -> True. (* By defn, P(s) = True *)
Proof.
 trivial.
Qed.
```

Rules of Inference (cont)

The universal instantiation is the abstraction of the examples below.

Example 1.10.1:

$$\text{All men are mortal.} \vdash \text{Socrates is mortal.}$$

Example 1.10.2: Consider the formula "$\forall n(\text{prime}(n) \wedge (n > 2) \rightarrow \text{odd}(n))$. " By applying universal instantiation with n "instantiate" to the term "$m + 4$" we obtain

$$\text{prime}(m + 4) \wedge (m + 4 > 2) \rightarrow \text{odd}(m + 4).$$

Here $m + 4$ is a **term** and m is an **arbitrary constant**.

Rules of Inference (cont)

Let practise how to perform natural deduction.

Example 1.10.3: Show that
$\sim R(c), \forall t(P(t) \rightarrow Q(t)), \forall t(Q(t) \rightarrow R(t)) \vdash \sim P(c)$.

Proof

1	$\sim R(c)$	premise
2	$\forall t(P(t) \rightarrow Q(t))$	premise
3	$\forall t(Q(t) \rightarrow R(t))$	premise
4	$P(c) \rightarrow Q(c)$	2, $\forall$-elimination
5	$Q(c) \rightarrow R(c)$	3, $\forall$-elimination
6	$P(c) \rightarrow R(c)$	4,6, $\rightarrow$-elimination
7	$\sim P(c)$	1,6, modus tollens (Shown in Slide 90)

Rules of Inference (cont)

```
Parameter X : Type.
Parameter c : X.

Lemma modus_tollens: forall (P Q : Prop),
  ~Q -> (P -> Q) -> ~P.
Proof.
  intros P Q H1 H2.
  intro H3.  (* Related to Slide 88: ~P=(P->F) *)
  apply H2 in H3.
  contradiction.
Qed.

Example eg_1_10_3: forall (P Q R : X->Prop),
  ~ R c -> (forall t : X, P t -> Q t) -> (forall t : X, Q t -> R t)
  -> ~ P c.
Proof.
  intros P Q R H1 H2 H3.
  specialize (H2 c).
  specialize (H3 c).
  apply (modus_tollens (Q c) (R c)) in H3.
  apply (modus_tollens (P c) (Q c)) in H2.
  assumption. assumption. assumption.
Qed.
```

Rules of Inference (cont)

Final Exam May 2019 Q3(d): Let $R(x, y)$ be a predicate with two variables. Infer the argument involving quantified statements

$$\forall x \forall y (R(x, y) \rightarrow \sim R(y, x)) \vdash \forall x (\sim R(x, x))$$

syntatically by stating the **rules of inference** in each step. (5 marks)

Rules of Inference (cont)

Lecture's Marking Guide

Let t be an arbitrary term independent of variables x and y.

1	$\forall x \forall y(R(x,y) \rightarrow\sim R(y,x))$	premise
2	$\forall y(R(t,y) \rightarrow\sim R(y,t))$	1 $\forall$-E .....[1 mark]
3	$R(t,t) \rightarrow\sim R(t,t)$	2 $\forall$-E
4	$\quad R(t,t)$	assumption ..[1 mark]
5	$\quad\sim R(t,t)$	3,4 $\rightarrow$I .....[1 mark]
6	$\quad\perp$	4,5 $\perp$I
7	$\sim R(t,t)$	4–6 $\neg$I .....[1 mark]
8	$\forall x(\sim R(x,x))$	7 $\forall$-I .....[1 mark]

Rules of Inference (cont)

Example 1.10.8: Show that the premises "A student in this class has not read the book," and "Everyone in this class passed the first exam" imply the conclusion "someone who passed the first exam has not read the book."

Class Discussion

Let $C(x)$: "x is in this class"; $B(x)$: "x has read the book"; $P(x)$: "x passed the first exam".

Try to formulate into premises and conclusion symbolically and then prove the argument.

Rules of Inference (cont)

Example 1.10.9: Show that the following arguments is valid with a formal proof using the laws of inference.

> *No junior or senior is enrolled in calculus class. Ali is enrolled in calculus class. Therefore Ali is not a senior.*

Class Discussion

Let $J(x)$: "x is a junior"; $S(x)$: "x is a senior"; and $C(x)$: "x is enrolled in calculus class."

Try to formulate into premises and conclusion symbolically and then prove the argument.

Outline

Applications

The formal logic theories are all developed in the hope to develop automated theorem-proving software to ensure mathematics proofs as well as human reasoning are free from human errors.

The applications of formal logic theories:

- *software specification*
- *hardware verification*
- compiler verification
- logic programming — prolog, ECLiPSe, etc.(?) → Application to classical AI (see `https://www.sjsu.edu/faculty/watkins/5thgen.htm`)
- ...

Logic Programming with Prolog

According to Wikipedia, *logic programming* is a type of programming paradigm based on predicate logic. Any program written in a logic programming language is a set of sentences, expressing *facts* and *rules* about some problem domain. Major logic programming language families include *Prolog* (which stands for Programming in Logic), *Datalog* (a subset of Prolog for database and serves as an alternative query system to SQL, and *answer set programming* (ASP).

Prolog was once believed to be the artificial intelligence language but the limitations of the first order logic has prevented it from getting popular. However, it is still useful in knowledge-based (or ontology) systems.

Logic Programming (cont)

Definition 1.11.1

In Prolog, a ***term*** is either a *constant*, a *variable* (starts with a capital letter or underscore) or a *compound term* of the form $f(t_1, \cdots, t_n)$ where f is a function symbol of arity n and t_i ($i = 1, \cdots, n$ and $n > 0$) are terms.
An ***atomic formula or actomic term*** is an expression of the form $p(t_1, \cdots, t_n)$ where p is a predicate symbol and t_i are terms. If $n = 0$, there is no argument and the parentheses are omitted.
A *constant* can either be a number or an *atomic term* (start from small letter, a string in single quotes or symbols such as +, =\=, etc.).

Logic Programming (cont)

The rules in Prolog are written in the form of *Horn clauses*, which can be regarded as "arguments" in Definition 65.

$$\underbrace{\underbrace{H}_{\text{head}} :- \underbrace{B_1, \cdots, B_n}_{\text{body}}.}_{\text{rule}} \tag{2}$$

where is the same as $\wedge$, $:-$ is the same as $\Rightarrow$, H and B_i are usually atomic formulae. So we read (2) as a logical implication in prenex form:

$$\forall X_1 \cdots \forall X_p B_1(X_1, \cdots, X_p) \wedge \cdots \wedge B_n(X_1, \cdots, X_p) \rightarrow H(X_1, \cdots, X_p).$$

Facts are rules that have **no body**, and are written in the simplified form (which is comparable to a "tautology"):

$$H. \tag{3}$$

Logic Programming (cont)

Remarks on Prolog language:

- A string starts and ends with double quote ”
- Don’t use single quote, it is for ‘atom’ (or symbol)
- Anything that starts with small letter is used as “predicate” or atom.
- Anything that starts with capital letter is used as “variables”.

Logic Programming (cont)

Example: Analyse the subjects in UTAR DMAS with and without pre-requisites in Prolog.

```
:- module(utar_dmas, [courses_without_prerequisites/2,
 courses_with_prerequisites/2]).

%
% Facts = NoSQL Database
%
course("UECM1024", "Calculus I", core, 4, []).
course("UECM1034", "Calculus II", core, 4, ['UECM1024']).
course("UECM1204", "Probability and Statistics I",core,4,[]).
course("UECM1224", "Probability and Statistics II",core,4,['UECM1204']).
course("UECM1314", "Fundamentals of Linear Algebra",core,4,[]).
course("UECM1304", "Discrete Mathematics with Applications",core,4,[]).
course("UECM1703", "Introduction to Scientific Computing",core,3,[]).
course("UECM2353", "Linear Algebra",core,3,['UECM1314']).
course("UECS1004", "Programming and Problem Solving",core,4,[]).
course("UECS1044", "Object-Oriented Application Development",core,4,['UECS1004']).
course("UECS2083", "Problem Solving with Data Structures and Algorithms",core,3,['UECS1044']).
course("UECS2094", "Web Application Development",core,4,['UECS1044']).
course("UECM2023", "Ordinary Differential Equations",core,3,['UECM1034']).
course("UECM3034", "Numerical Methods",core,4,[]).

%
% Rules = Classical AI
%
courses_without_prerequisites(X,Y) :- course(X,Y,_,_,[]).
courses_with_prerequisites(X,Y) :- course(X,Y,_,_,Z), length(Z, N), N =\= 0.
```

Logic Programming (cont)

- To **load** the Prolog database, one can use either the commands below:

```
?- [utar_dmas].
?- consult('utar_dmas.pl').
```

- To **query** the courses with only the code without prerequisite, we can type the following query. Note that underscore is used to indicate a value we are not interested in.

```
?- courses_without_prerequisites(X,_).
```

- To **query** the courses (showing both code and name) with prerequisite, we can type the following query.

```
?- courses_with_prerequisites(X,Y).
```

- To **exit** Prolog, type `halt.`

Logic Programming (cont)

Questions for the use of predicates:

- Is the predicate ‘course’ with 5 parameters a good idea?
- How to update the ‘knowledge’ database?
- How to translate usual programming languages (C, C++, Python, Java) to Logic Programming?
  - Each programming statements are executed in sequence.
  - if (...) ...
  - for (...) ... or while(...) ...
  - Define a function: return_type function_name(arguments)

Logic Programming (cont)

Doing things in sequence in Python:

```
print("This is first line")
print("This is second line:", 1, 2, 3)
```

Doing things in sequence in Prolog:

```
% swipl -q hellolines.pl
:- format("This is first line\n").
:- format("This is second line: ~d ~d ~d\n", [1, 2, 3]).
:- halt.
```

Note that rules without head are called **directives** and are immediately executed by Prolog.

Logic Programming (cont)

if (...) ... in Python

```
def thegrade(marks):
    if marks<0 or marks>100: return "????"
    elif marks >= 90: return "A+"
    elif marks >= 80: return "A"
    elif marks >= 75: return "A-"
    elif marks >= 70: return "B+"
    elif marks >= 65: return "B"
    elif marks >= 60: return "B-"
    elif marks >= 55: return "C+"
    elif marks >= 50: return "C"
    else:             return "F"
```

Logic Programming (cont)

if (...) ... in Prolog

```
thegrade(Marks, Grade) :- Marks < 0,   Grade="????", !.
thegrade(Marks, Grade) :- Marks > 100, Grade="????", !.
thegrade(Marks, Grade) :- Marks >= 90, Grade="A+", !.
thegrade(Marks, Grade) :- Marks >= 80, Grade="A", !.
thegrade(Marks, Grade) :- Marks >= 75, Grade="A-", !.
thegrade(Marks, Grade) :- Marks >= 70, Grade="B+", !.
thegrade(Marks, Grade) :- Marks >= 65, Grade="B", !.
thegrade(Marks, Grade) :- Marks >= 60, Grade="B-", !.
thegrade(Marks, Grade) :- Marks >= 55, Grade="C+", !.
thegrade(Marks, Grade) :- Marks >= 50, Grade="C", !.
thegrade(Marks, Grade) :- Grade="F".
```

Note: '!' is called cut, it is liked ignored anything below if match.

Logic Programming (cont)

if (...) ... in Prolog Version 2

```
thegrade(Marks, Grade) :- Marks < 0,   Grade="????".
thegrade(Marks, Grade) :- Marks > 100, Grade="????".
thegrade(Marks, Grade) :- 90 <= Marks <= 100, Grade="A+".
thegrade(Marks, Grade) :- 80 <= Marks < 90, Grade="A".
thegrade(Marks, Grade) :- 75 <= Marks < 80, Grade="A-".
thegrade(Marks, Grade) :- 70 <= Marks < 75, Grade="B+".
thegrade(Marks, Grade) :- 65 <= Marks < 70, Grade="B".
thegrade(Marks, Grade) :- 60 <= Marks < 65, Grade="B-".
thegrade(Marks, Grade) :- 55 <= Marks < 60, Grade="C+".
thegrade(Marks, Grade) :- 50 <= Marks < 55, Grade="C".
thegrade(Marks, Grade) :-  0 <= Marks < 50, Grade="F".
```

Problem with this version: All instructions will be run (matched). Only one true.

Logic Programming (cont)

for (...) ... in Python

```
def sum_arithm(n):
    thesum = 0
    for k in range(1, n+1):
        thesum += k
    return thesum

def sum_sqrt(n):
    thesum = 0
    for k in range(1, n+1):
        thesum += k**0.5
    return thesum
```

What is `sum_arithm(1000)` and sum_sqrt(1000)?

The first one is $\frac{1000 \times 1001}{2} = 500500$ while the second one is 21097.455887480734

Logic Programming (cont)

for (...) ... in Prolog

Prolog has no for loop, no while loop, how to calculate?

For the first one, we know the formula below (to be proved in Topic 2):

$$1 + 2 + \cdots + n = \frac{n(n+1)}{2}.$$

So the implementation in Prolog can be simple (without using for loop):

```
sum_arithm(N, Sum) :- Sum is N*(N+1)/2.
```

What about sum_sqrt? We don't have loop and function!

Logic Programming (cont)

for (...) ... in Prolog (cont)

We can only use **predicate** and **recursion** (call itself).

$$\begin{aligned}\text{sum\_sqrt}(N) &= \sqrt{N} + \text{sum\_sqrt}(N-1) \\ &= \sqrt{N} + \sqrt{N-1} + \text{sum\_sqrt}(N-2) \\ &= \ldots = \sqrt{N} + \sqrt{N-1} + \cdots + \sqrt{2} + \sqrt{1}\end{aligned}$$

Logic Programming (cont)

for (...) ... in Prolog (cont)

The implementation is as follows:

```
sum_sqrt(1, Sum) :- Sum is 1, !.
sum_sqrt(N, Sum) :- N < 1, integer(N), Sum="Meaningless",
sum_sqrt(N, Sum) :- N1 is N-1,
                    sum_sqrt(N1, Sum2),
                    Sum is sqrt(N) + Sum2.
```

It is still difficult to understand, I believe.

Applications to Database Query

Database Systems:

- Structured: Sqlite, Postgresql, MySQL, MariaDB, Oracle DB, Microsoft SQL, etc.
- NoSQL (refers to Non-relational database):
  - Key-value-based: Redis, Memcached, Azure Cosmos DB, Apache Ignite, etc.
  - Object-based: Objectivity/DB, Perst, ZopeDB
  - Column-based: Cassandra, ScyllaDB, Amazon DynamoDB, Google Cloud Datastore, HBase, etc.
  - Graph-based: Neo4j, Twitter's FlockDB, OrientDB, Azure Cosmos DB, AllegroGraph, etc.

Applications to DB Query (cont)

Query languages:

- SQL → Structured
- Datalog → Structured & NoSQL
- Prolog (need to beward of infinite recursion!) → Structured & NoSQL
- SPARQL (stands for SPARQL Protocol and RDF Query Language) → NoSQL

Applications to Database (cont)

Real-world business software consist a lot of SQL statements. A dictionary of SQL relational algebra and Datalog is given below.

SQL Concept	Relational Algebra	Datalog
Intersection	$R(x,y) \cap T(x,y)$	$I(x,y)$:-$R(x,y), T(x,y)$.
Union	$R(x,y) \cup T(x,y)$	$U(x,y)$:-$R(x,y)$. $U(x,y)$:-$T(x,y)$.
Difference	$R(x,y) \setminus T(x,y)$	$D(x,y)$:-$R(x,y)$, not $T(x,y)$.
Projection	$\pi_x(R)$	$P(x)$:-$R(x,y)$
Selection	$\sigma_{x>10}(R)$	$S(x,y)$:-$R(x,y), x > 10$
Product	$R \times T$	$P(x,y,z,w)$:-$R(x,y), T(z,w)$
Natural Join	$R \bowtie T$	$J(x,y,z)$:-$R(x,y), T(y,z)$
Theta Join	$R \bowtie_{R.x>T.z} T$	$\theta(x,y,z,w)$:-$R(x,y), T(z,w), x > z$

Applications to Database (cont)

Why are new database query languages related to logic programming is getting popular? Because logic can help traverse the “knowledge” network:

```
#lang datalog
% Paths in a Graph: Datalog Formulation

edge(a,b).
edge(a,c).
edge(b,a).
edge(b,d).

path(X,Y) :- path(X,Z), edge(Z,Y).
path(X,Y) :- edge(X,Y).
```

E.g. With proper query such as path(X, a)?, we can find who can reach a.