COMP2111 Week 2
Term 1, 2024
Discrete Mathematics Recap II

Summary of topics

- Sets
- Formal languages
- Relations
- Functions
- Propositional Logic

Functions

A **function**, $f : S \to T$, is a binary relation $f \subseteq S \times T$ such that for all $s \in S$ there is *exactly one* $t \in T$ such that $(s, t) \in f$.

We write $f(s)$ for the unique element related to s.

A **partial function** $f : S \nrightarrow T$ is a binary relation $f \subseteq S \times T$ such that for all $s \in S$ there is *at most one* $t \in T$ such that $(s, t) \in f$. That is, it is a function $f : S' \longrightarrow T$ for $S' \subseteq S$

Functions

A function $f : S \longrightarrow T$ is a pairing of the sets: it means that f assigns to every element $s \in S$ a single element $t \in T$. To emphasise where a specific element is sent, we can write $f : x \mapsto y$, which means the same as $f(x) = y$

		Symbol	
S	**domain** of f	$\mathrm{dom}(f)$	(inputs)
T	**co-domain** of f	$\mathrm{codom}(f)$	(*possible* outputs)
$f(S)$	**image** of f	$\mathrm{im}(f)$	(*actual* outputs)
$= \{ f(x) : x \in \mathrm{dom}(f) \}$			

Important!

The domain and co-domain are critical aspects of a function's definition.

$$f : \mathbb{N} \to \mathbb{Z} \quad \text{given by} \quad f : x \mapsto x^2$$

and

$$g : \mathbb{N} \to \mathbb{N} \quad \text{given by} \quad g : x \mapsto x^2$$

are different functions even though they have the same behaviour!

Composition of Functions

Composition of functions is described as

$$g \circ f : x \mapsto g(f(x)), \quad \text{requiring } \operatorname{im}(f) \subseteq \operatorname{dom}(g)$$

Composition is associative

$$h \circ (g \circ f) = (h \circ g) \circ f, \quad \text{can write } h \circ g \circ f$$

Composition of Functions

If a function maps a set into itself, i.e. when $\mathrm{dom}(f) = \mathrm{codom}(f)$ (and thus $\mathrm{im}(f) \subseteq \mathrm{dom}(f)$), the function can be composed with itself — **iterated**

$$f \circ f, f \circ f \circ f, \ldots, \quad \text{also written } f^2, f^3, \ldots$$

Identity function on S

$$\mathrm{Id}_S(x) = x, x \in S; \mathrm{dom}(i) = \mathrm{codom}(i) = \mathrm{im}(i) = S$$

For $g : S \longrightarrow T \quad g \circ \mathrm{Id}_S = g,\ \mathrm{Id}_T \circ g = g$

Extension: Composition of Binary Relations

If $R_1 \subseteq S \times T$ and $R_2 \subseteq T \times U$ then the composition of R_1 and R_2 is the relation:

$$R_1; R_2 := \{(a, c) : \quad \text{there is a } b \in T \text{ such that } (a, b) \in R_1 \text{ and } (b, c) \in R_2\}.$$

Note that if $f : S \to T$ and $g : T \to S$ are functions then $f; g = g \circ f$.

Properties of Functions

A function is called **surjective** or **onto** if every element of the codomain is mapped to by at least one x in the domain, i.e.

$$\mathrm{im}(f) = \mathrm{codom}(f)$$

Examples (of surjective functions)

- $f : \mathbb{N} \longrightarrow \mathbb{N}$ with $f(x) = x$
- Floor, ceiling

Examples (of non-surjective functions)

- $f : \mathbb{N} \longrightarrow \mathbb{N}$ with $f(x) = x^2$
- $f : \{a, \ldots, z\}^* \longrightarrow \{a, \ldots, z\}^*$ with $f(\omega) = a\omega e$

Injective Functions

A function is called **injective** or **1–1** (**one-to-one**) if different inputs give different outputs, i.e.

$$f(x) = f(y) \rightarrow x = y$$

Examples (of functions that are injective)

- $f : \mathbb{N} \longrightarrow \mathbb{N}$ with $f(x) \mapsto x$
- set complement (for a fixed universe)

Examples (of functions that are not injective)

- absolute value, floor, ceiling
- length of a word

Function is **bijective** if it is both surjective and injective.

Converse of a function

Question

$f^{\leftarrow}$ is a relation; when is it a function?

Question

$f^{\leftarrow}$ is a relation; when is it a function?

Answer

When f is a bijection.

Inverse Functions

Inverse function — $f^{-1} : T \longrightarrow S$;
for a given $f : S \longrightarrow T$ exists exactly
when f is bijective.

Image of a subdomain A under a function

$$f(A) = \{\, f(s) : s \in A \,\} = \{\, t \in T : t = f(s) \text{ for some } s \in A \,\}$$

Inverse image — $f^{\leftarrow}(B) = \{\, s \in S : f(s) \in B \,\} \subseteq S$;
it is defined for every f (recall: converse of a relation)

If f^{-1} exists then $f^{\leftarrow}(B) = f^{-1}(B)$

$f(\emptyset) = \emptyset, f^{\leftarrow}(\emptyset) = \emptyset$

Summary of topics

- Sets
- Formal languages
- Relations
- Functions
- Propositional Logic

Propositions

A sentence of a natural language (like English, Cantonese, Warlpiri) is *declarative*, or a *proposition*, if it can be meaningfully be said to be either true or false.

Examples

- Richard Nixon was president of Ecuador.
- A square root of 16 is 4.
- Euclid's program gets stuck in an infinite loop if you input 0.
- Whatever list of numbers you give as input to this program, it outputs the same list but in increasing order.
- $x^n + y^n = z^n$ has no nontrivial integer solutions for $n > 2$.
- 3 divides 24.

The following are *not* declarative sentences:

- Wingardium leviosa.
- For Pete's sake, take out the garbage!
- Are elephants made of cardboard?
- Please waive the prerequisites for this subject for me.

Declarative sentences in natural languages can be *compound* sentences, built out of other sentences.
Propositional Logic is a formal representation of some constructions for which the truth value of the compound sentence can be determined from the truth value of its components.

- My pants are on fire *and* my hat is soaked.
- Nixon won the debate *or* Nixon applied a coat of Lazy Shave.
- *It is not the case that* this program always halts.

Not all constructions of natural language are truth-functional:

- *Carroll believes that* Santa Claus is real.
- *Johannes knows* Santa Claus is real.
- This program always halts *because* it contains no loops.
- The disk crashed *after* I saved my file.

NB

Various **modal logics** *extend classical propositional logic to cover these.*

The Three Basic Connectives of Propositional Logic

symbol	**text**
$\wedge$	"and", "but", ";", ":"
$\vee$	"or", "either ...or ..."
$\neg$	"not", "it is not the case that"

Truth tables:

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

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

A	$\neg$ A
F	T
T	F

Applications I: Program Logic

Example

```
if x > 0 or (x <= 0 and y > 100):
```

Let $p \stackrel{\text{def}}{=}$ (x > 0) and $q \stackrel{\text{def}}{=}$ (y > 100)

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

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

This is equivalent to $p \vee q$. Hence the code can be simplified to

```
if x > 0 or y > 100:
```

Now consider the following constructions:

- if A then B
- A only if B
- B if A
- A implies B
- it follows from A that B
- whenever A, B
- A is a sufficient condition for B
- B is a necessary condition for A

Each has the property that if the whole statement is true, and A is true, then B is true.

We can *approximate* the English meaning of these by "not (A and not B)", written $A \to B$ which has the following truth table:

A	B	A $\to$ B
F	F	T
F	T	T
T	F	F
T	T	T

While only an approximation to the English, 100+ years of experience have shown this to be adequate for capturing *mathematical reasoning*.
(Moral: mathematical reasoning does not need all the features of English.)

“If it rains, I bring an umbrella”

Conversational implication (in informal English):

Material implication (in formal logic):

“If it rains, I bring an umbrella”

Conversational implication (in informal English):

Material implication (in formal logic):

Conversational implication also has a whiff of *causality*.

> *“If I don’t bring an umbrella, it doesn’t rain”*

In formal logic, this new sentence is equivalent to “If it rains...”.
In English, it suggests that my behaviour changes the weather, not that the weather changes my behaviour.

Examples

p = "you get an HD on your final exam"
q = "you do every exercise in the book"
r = "you get an HD in the course"
Translate into logical notation:

(a) You get an HD in the course although you do not do every exercise in the book.

(c) To get an HD in the course, you must get an HD on the exam.

(d) You get an HD on your exam, but you don't do every exercise in this book; nevertheless, you get an HD in this course.

Examples

p = "you get an HD on your final exam"
q = "you do every exercise in the book"
r = "you get an HD in the course"
Translate into logical notation:

(a) You get an HD in the course although you do not do every exercise in the book.

Examples

p = "you get an HD on your final exam"
q = "you do every exercise in the book"
r = "you get an HD in the course"
Translate into logical notation:

(a) You get an HD in the course although you do not do every exercise in the book. $r \wedge \neg q$

(c) To get an HD in the course, you must get an HD on the exam.

Examples

p = "you get an HD on your final exam"
q = "you do every exercise in the book"
r = "you get an HD in the course"
Translate into logical notation:

(a) You get an HD in the course although you do not do every exercise in the book. $r \wedge \neg q$

(c) To get an HD in the course, you must get an HD on the exam. $r \rightarrow p$

(d) You get an HD on your exam, but you don't do every exercise in this book; nevertheless, you get an HD in this course.

Examples

p = "you get an HD on your final exam"
q = "you do every exercise in the book"
r = "you get an HD in the course"
Translate into logical notation:

(a) You get an HD in the course although you do not do every exercise in the book. $r \wedge \neg q$

(c) To get an HD in the course, you must get an HD on the exam. $r \rightarrow p$

(d) You get an HD on your exam, but you don't do every exercise in this book; nevertheless, you get an HD in this course. $p \wedge \neg q \wedge r$

Unless

A unless B can be approximated as $\neg B \rightarrow A$

E.g.
I go swimming unless it rains = If it is not raining then I go swimming.
Correctness of the translation is perhaps easier to see in:
I don't go swimming unless the sun shines = If the sun does not shine then I don't go swimming.

Note that "I go swimming unless it rains, but sometimes I swim even though it is raining" makes sense, so the translation of "A unless B" should not imply $B \rightarrow \neg A$.

If and only if

A if, and only if, B is written $A \leftrightarrow B$

I will have an entree if and only if I won't have desert.
= If I have desert I will not have an entree and vice versa.

It has the following truth table:

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

Same as $(A \rightarrow B) \wedge (B \rightarrow A)$

A **Propositional formula** is made up of **propositional variables** and **logical connectives** ($\wedge, \vee, \neg, \rightarrow, \leftrightarrow$).

A **truth assignment** (aka **valuation**, aka **state**) assigns T or F to each propositional variable, and, using the logical connectives, gives a truth value to all propsitional formulas.

Logical Equivalence

Two formulas ϕ, ψ are **logically equivalent**, denoted $\phi \equiv \psi$ if they have the same truth value for all truth valuations.

Application: If ϕ and ψ are two formulae such that $\phi \equiv \psi$, then the digital circuits corresponding to ϕ and ψ compute the same function. Thus, proving equivalence of formulas can be used to *optimise* circuits.

Some Well-Known Equivalences

Excluded Middle	$p \vee \neg p \equiv \top$
Contradiction	$p \wedge \neg p \equiv \bot$
Identity	$p \vee \bot \equiv p$
	$p \wedge \top \equiv p$
	$p \vee \top \equiv \top$
	$p \wedge \bot \equiv \bot$
Idempotence	$p \vee p \equiv p$
	$p \wedge p \equiv p$
Double Negation	$\neg\neg p \equiv p$
Commutativity	$p \vee q \equiv q \vee p$
	$p \wedge q \equiv q \wedge p$

Associativity	$(p \vee q) \vee r \equiv p \vee (q \vee r)$
	$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$
Distribution	$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$
	$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$
De Morgan's laws	$\neg(p \wedge q) \equiv \neg p \vee \neg q$
	$\neg(p \vee q) \equiv \neg p \wedge \neg q$
Implication	$p \rightarrow q \equiv \neg p \vee q$
	$p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$

Satisfiability of Formulas

A formula is **satisfiable**, if it evaluates to T for *some* assignment of truth values to its basic propositions.

Example

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

Applications II: Constraint Satisfaction Problems

These are problems such as timetabling, activity planning, etc.
Many can be understood as showing that a formula is satisfiable.

Example

You are planning a party, but your friends are a bit touchy about who will be there.

1. If John comes, he will get very hostile if Sarah is there.
2. Sarah will only come if Kim will be there also.
3. Kim says she will not come unless John does.

Who can you invite without making someone unhappy?

Translation to logic: let J, S, K represent "John (Sarah, Kim) comes to the party". Then the constraints are:

1. $J \to \neg S$
2. $S \to K$
3. $K \to J$

Thus, for a successful party to be possible, we want the formula $\phi = (J \to \neg S) \wedge (S \to K) \wedge (K \to J)$ to be satisfiable.
Truth values for J, S, K making this true are called *satisfying assignments*, or *models*.

We figure out where the conjuncts are false, below. (so blank = T)

J	K	S	$J \to \neg S$	$S \to K$	$K \to J$	ϕ
F	F	F				
F	F	T		F		F
F	T	F			F	F
F	T	T			F	F
T	F	F				
T	F	T	F	F		F
T	T	F				
T	T	T	F			F

Conclusion: a party satisfying the constraints can be held. Invite nobody, or invite John only, or invite Kim and John.

Exercise

Which of the following formulae are *always* true?

(a) $(p \wedge (p \rightarrow q)) \rightarrow q$

(b) $((p \vee q) \wedge \neg p) \rightarrow \neg q$

(e) $((p \rightarrow q) \vee (q \rightarrow r)) \rightarrow (p \rightarrow r)$

(f) $(p \wedge q) \rightarrow q$

Exercise

Which of the following formulae are *always* true?

(a) $(p \wedge (p \rightarrow q)) \rightarrow q$ — always true

(b) $((p \vee q) \wedge \neg p) \rightarrow \neg q$ — not always true

(e) $((p \rightarrow q) \vee (q \rightarrow r)) \rightarrow (p \rightarrow r)$ — not always true

(f) $(p \wedge q) \rightarrow q$ — always true

Validity, Entailment, Arguments

An *argument* consists of a set of declarative sentences called *premises* and a declarative sentence called the *conclusion*.

Example

Premises:	Frank took the Ford or the Toyota.
	If Frank took the Ford he would be late.
	Frank is not late.
Conclusion:	Frank took the Toyota

An argument is *valid* if the conclusions are true *whenever* all the premises are true. Thus: if we believe the premises, we should also believe the conclusion.
(Note: we don't care what happens when one of the premises is false.)
Other ways of saying the same thing:

- The conclusion *logically follows* from the premises.
- The conclusion is a *logical consequence* of the premises.
- The premises **entail** the conclusion.

The argument above is valid. The following is invalid:

Example

Premises:	Frank took the Ford or the Toyota.
	If Frank took the Ford he would be late.
	Frank is late.
Conclusion:	Frank took the Ford.

Validity of arguments in propositional logic is captured as follows:

Let $\phi_1, \ldots, \phi_n$ and ϕ be formulae of propositional logic. Draw a truth table with columns for each of $\phi_1, \ldots, \phi_n$ and ϕ.
The argument with premises $\phi_1, \ldots, \phi_n$ and conclusion ϕ is valid, denoted

$$\phi_1, \ldots, \phi_n \models \phi$$

if for every row of the truth table where $\phi_1, \ldots, \phi_n$ are all true, ϕ is true also.

NB: $\models$ is the same thing as $\Rightarrow$ from Week 1.

We mark only true locations (blank = F)

Frd	$Tyta$	$Late$	$Frd \vee Tyta$	$Frd \rightarrow Late$	$\neg Late$	$Tyta$
F	F	F		T	T	
F	F	T		T		
F	T	F	T	T	T	T
F	T	T	T	T		T
T	F	F	T		T	
T	F	T	T	T		
T	T	F	T		T	T
T	T	T	T	T		T

This shows $Frd \vee Tyta, Frd \rightarrow Late, \neg Late \models Tyta$

The following row shows $Frd \vee Tyta, Frd \rightarrow Late, Late \not\models Frd$

Frd	$Tyta$	$Late$	$Frd \vee Tyta$	$Frd \rightarrow Late$	$Late$	Frd
F	T	T	T	T	T	F

Applications III: Reasoning About Requirements/Specifications

Suppose a set of English language requirements R for a software/hardware system can be formalised by a set of formulae $\{\phi_1, \ldots \phi_n\}$.
Suppose C is a statement formalised by a formula ψ. Then

1. The requirements cannot be implemented if $\phi_1 \wedge \ldots \wedge \phi_n$ is not satisfiable.
2. If $\phi_1, \ldots \phi_n \models \psi$ then every correct implementation of the requirements R will be such that C is always true in the resulting system.
3. If $\phi_1, \ldots \phi_{n-1} \models \phi_n$, then the condition ϕ_n of the specification is redundant and need not be stated in the specification.

Example

Requirements R: An alarm system for a house is to operate as follows. The alarm should not sound unless the system has been armed or there is a fire. If the system has been armed and a door is disturbed, the alarm should ring. Irrespective of whether the system has been armed, the alarm should go off when there is a fire.

Conclusion C: If the alarm is ringing and there is no fire, then the system must have been armed.

Questions

1. Will every system correctly implementing requirements R satisfy C?
2. Is the final sentence of the requirements redundant?

Expressing the requirements as formulas of propositional logic, with

- S = the alarm sounds = the alarm rings
- A = the system is armed
- D = a door is disturbed
- F = there is a fire

we get

Requirements:

1. $S \rightarrow (A \vee F)$
2. $(A \wedge D) \rightarrow S$
3. $F \rightarrow S$

Conclusion: $(S \wedge \neg F) \rightarrow A$

Our two questions then correspond to

1. Does $S \to (A \vee F), (A \wedge D) \to S, F \to S \models (S \wedge \neg F) \to A$?
2. Does $S \to (A \vee F), (A \wedge D) \to S \models F \to S$?

Validity of Formulas

A formula ϕ is **valid**, or a **tautology**, denoted $\models \phi$, if it evaluates to T for *all* assignments of truth values to its basic propositions.

Example

A	B	$(A \to B) \to (\neg B \to \neg A)$
F	F	T
F	T	T
T	F	T
T	T	T

Validity, Equivalence and Entailment

Theorem

The following are equivalent:

- $\phi_1, \ldots \phi_n \models \psi$
- $\models (\phi_1 \wedge \ldots \wedge \phi_n) \rightarrow \psi$
- $\models \phi_1 \rightarrow (\phi_2 \rightarrow \ldots (\phi_n \rightarrow \psi) \ldots)$

Theorem

$\phi \equiv \psi$ *if and only if* $\models \phi \leftrightarrow \psi$

Beyond propositions

Entailment captures a form of logical reasoning, but it cannot handle relatively simple logical arguments like the following:

1. Socrates is a man
2. All men are mortal
3. Therefore Socrates is mortal

We need to add **expressiveness** to propositional logic so that we can capture notions such as the *relation* between man and men in the first two statements; and the *quantified* statement "all men".

NB

Adding expressiveness comes at a cost: it is now more difficult to determine truth values.

Predicates

Predicates are functions that take inputs from a set and return either true or false – i.e. they are relations. Predicates enable us to establish relationships between different propositions, such as the man/men connection between the first and second propositions on the previous slide, allowing more expressiveness than propositional logic can give.

Quantifiers

Quantifiers allow us to make quantified statements over predicates, e.g.

> *"If there exists a satisfying assignment ..."*

or

> *"Every natural number greater than 2 ..."*

The two standard quantifiers are

- $\forall$: "for all", "for any", "every"
- $\exists$: "there exists", "there is", "for some", "at least one"

Example

Goldbach conjecture

$$\forall n \in 2\mathbb{N}\,(n > 2 \rightarrow \exists p, q \in \mathbb{N}\,(p, q \in \textsc{Primes} \wedge n = p + q))$$

Predicate logic

Predicate (or first-order) logic extends propositional logic by adding predicates and quantifiers.

More on predicate logic in Weeks 1 and 5.

Summary of topics

- Sets
- Formal languages
- Relations
- Functions
- Propositional Logic