Logic, Computability and Incompleteness

First-order Logic Revisited

What is Logic?

A standard characterization:

Logic is the ‘science’ of valid arguments.

Modern ‘symbolic’ or formal logic is the *mathematical theory* of valid arguments

What is an argument?

Intuitively, an argument can be thought of as an inference, or piece of reasoning,

where certain statements are meant to support or establish a conclusion.

What is an Argument?

More precisely,

an argument is a finite set of (declarative) sentences, where one sentence is singled out as the **conclusion** and the other sentences are the *premises*.

Standard Argument Form:

Premise 1
Premise 2
…
Premise *n*
Therefore, **Conclusion**

What is Validity?

An argument is valid iff it is <u>not</u> <u>possible</u> for *all* the premises to be **true** and the conclusion **false**.

Alternatively, *if* all the premises *were* **true**,

then the conclusion *would have to be* **true** as well.

Some arguments are valid and some are not:

If it is snowing, then it is cold outside.

<u>It is snowing.</u>

Therefore, it is cold outside.

If the earth is round then the sky is blue.

<u>The sky is blue.</u>

Therefore the earth is round.

Some Examples

All politicians are human.

Some humans are wise.

Therefore, some politicians are wise.

If the sky is blue then the earth is flat.

The earth is not flat.

Therefore, the sky is not blue.

Some philosophers are politicians.

All politicians are corrupt.

Therefore, some philosophers are corrupt.

Either today is Wednesday or pigs can fly.

Pigs can’t fly.

Therefore, today is Wednesday.

Formal Logic

As rational beings, we have *intuitions* about which arguments are valid and which are not.

Modern logic provides a **mathematical theory** of validity

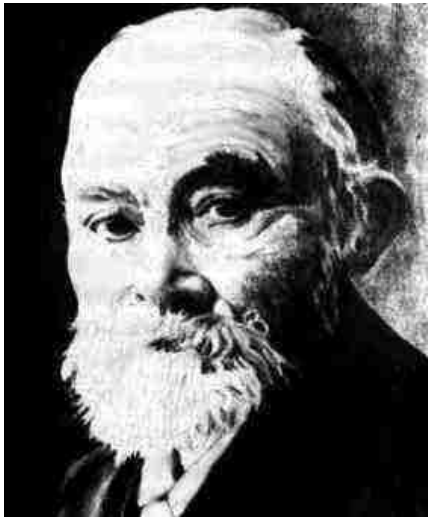
whereby it is possible to **prove** that an argument is valid.

Although logic began as a branch of Philosophy and has been studied since ancient times, it underwent dramatic mathematical development in the 19th and 20th centuries.

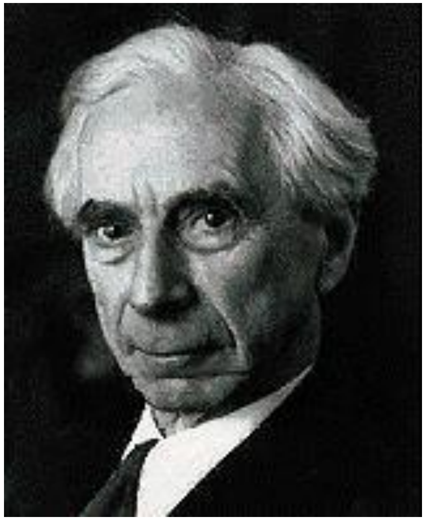
The biggest advance in logic for 2000 years was due to Gottlob Frege, who developed a system of quantifiers and variables to capture the logic of **generality** first explored by Aristotle.

Frege’s system is now known as **First-order Logic**.

Some Famous Logicians



Gottlob Frege



Bertrand Russell

Kurt Gödel

Alfred Tarski

Natural *vs* Artificial Languages

The *medium* of logic is language.

When studying the grammar of a natural language such as English, we try to devise rules that accurately characterize a *pre-existing* phenomenon.

In contrast, artificial (or formal) languages, of the kind used in logic (and computer science), are **defined** by the grammatical rules we give.

Logic requires a precisely defined artificial **object language** in which formal arguments can be expressed and analyzed.

This step of ‘idealization’ is necessary to obtain mathematically rigorous results.

First-order Language

- **Generic First-order Syntax:**
- Vocabulary:
- Logical symbols: constants $\neg \vee \wedge \rightarrow \leftrightarrow \exists \forall =$

  variables x, y, z,…

  [and (,) for punctuation]
- Non-logical symbols:

  individual constants a,b,c …

  function symbols f_1^n, f_2^n, f_3^n ...

  sentence letters Q, R, S …

  predicate letters P_1^n, P_2^n, P_3^n ...

First-order Language

- A language ***L*** possesses a denumerable supply of non-logical symbols (and where each of the 4 categories above possess denumerably many elements).
- These combine with the logical symbols, according to the Formation Rules, to yield terms and formulas of the language:

Generic First-order Syntax

- **Formation Rules**:
- Definition of Terms and Formulas

Terms of ***L***

(1) Atomic terms

(i) all individual constants of ***L*** are terms of ***L***

(ii) all variables of ***L*** are terms of ***L***

(2) Compound terms

if f^n is an n-place function symbol and $t_1,\ldots, t_n$ are terms, then $f^n(t_1,\ldots, t_n)$ is a term of ***L*** (where $n > 0$)

- **Nothing else** is a term of ***L***!

Generic First-order Syntax

Formulas of ***L***

(1) Atomic formulas:

(i) all sentence letters of ***L*** are formulas of ***L***

(ii) if P^n is an n-place predicate letter and $t_1,\ldots, t_n$ are terms, then $P^n\, t_1,\ldots, t_n$ is a formula of ***L*** (where $n > 0$)

(2) Compound formulas

(i) if Φ is a formula of ***L***, then $\neg\, \Phi$ is a formula of ***L***

(ii) if Φ, Ψ are formulas of ***L***, then $(\Phi \wedge \Psi)$, $(\Phi \vee \Psi)$, $(\Phi \rightarrow \Psi)$, and $(\Phi \leftrightarrow \Psi)$ are formulas of ***L***.

(iii) if Φ is a formula of ***L*** and **v** is a variable, then $\forall \mathbf{v}\Phi$ and $\exists \mathbf{v}\Phi$ are formulas of ***L***.

Generic First-order Syntax

- **Nothing else** is a formula of ***L***!
- Note that there are an infinite number of formulas,

  but no formula is infinitely long.
- Since formulas are finite sequences of symbols from a denumerable set,

  a *language* ***L*** can have only denumerably many formulas.

Generic First-order Semantics

- **Generic First-order Semantics**
- An interpretation $\mathcal{I}$ (or structure or model) of the language $\boldsymbol{L}$ is a way of "giving meaning" to the symbols of $\boldsymbol{L}$.
- In particular, an interpretation $\mathcal{I}$ of $\boldsymbol{L}$ specifies the following:

(i) a non-empty set D (the domain or universe of discourse).

(ii) for each individual constant **c**,

$$\mathcal{I}(\mathbf{c}) \text{ is an object } e \in D.$$

(iii) for each n-ary function symbol $f^{\,n}$,

$$\mathcal{I}(f^{\,n}) \text{ is an } n\text{-ary function } F^n: D^n \longmapsto D$$

(where D^n is the n^{th} Cartesian product of D)

Generic First-order Semantics

(iv) for each sentence letter S

$\mathcal{I}(S)$ is a truth value, either **0** or **1**

(v) we treat '=' as a privileged 2-place predicate symbol,

where $\mathcal{I}(=)$ is the set of all pairs $< e,e >$

such that $e \in D$.

(vi) for each n-ary predicate letter P^n [other than '=']

$\mathcal{I}(P^n)$ is a set of ordered n-tuples

such that $\mathcal{I}(P^n) \subseteq D^n$.

Generic First-order Semantics

The notation used in clause (iii), where $\mathcal{J}(f^{n})$ is F^{n}: $D^{n} \bullet\!\!\rightarrow D$, connotes the fact that the interpretation of a function symbol f^{n} is an n-ary function

mapping n-tuples of elements of D to elements of D

while (vi) indicates that $\mathcal{J}(P^{n})$ is an n-ary relation.

In general, n-place functions are equivalent to a (proper) subset of the set of $(n+1)$-place relations:

- So $\mathcal{J}(f^{n}) \subset D^{n+1}$, with the constraint that

for every n-tuple $< d_1,\ldots, d_n > \in D^{n}$,

there exists exactly one $d_{n+1} \in D$ such that

$< d_1,\ldots, d_n, d_{n+1} > \in \mathcal{J}(f^{n})$.

Example

- Consider the following very simple (fragment of an) interpretation $\mathcal{J}$ as an illustration:
- Let the domain D of $\mathcal{J}$ be the 2-member set {Jack, Jill}.

  So $\mathcal{J}(=)$ is the set { <Jack, Jack>, < Jill, Jill> }
- Consider the first two individual constants c_1 and c_2

  Let $\mathcal{J}(c_1)$ = Jack and $\mathcal{J}(c_2)$ = Jill
- Consider the 1-place predicate symbol P^1

  The definition requires that $\mathcal{J}(P^1) \subseteq D^1$

  D^1 = {Jack, Jill} (which is equivalent to { <Jack>, < Jill> })

  So let $\mathcal{J}(P^1)$ = {Jack}

Example

- Consider the 2-place predicate symbol L^2

  The definition requires that $\mathcal{J}(L^2) \subseteq D^2$

  D^2 = {<Jack, Jack>, <Jack, Jill>, <Jill, Jack>, <Jill, Jill>}

  So let $\mathcal{J}(L^2)$ = {<Jack, Jack>, <Jack, Jill>}
- Consider the 1-place function symbol f^1

  The definition requires that $\mathcal{J}(f^1)$ is a 1-place <u>function</u>

  $F^1: D^1 \bullet\!\!\longrightarrow D.$

  As before, D^1 = {Jack, Jill}

  So let $\mathcal{J}(f^1)$ = {<Jack, Jack>, <Jill, Jack>}
- Note that the interpreted 2-place predicate L^2 <u>cannot</u> be used to define a corresponding 1-place function (why?).

Generic First-order Semantics

- In accord with clauses (ii)-(vi) above, the non-logical symbols of $\boldsymbol{L}$ (plus '**=**') are given some fixed interpretation by $\boldsymbol{\mathcal{J}}$.
- This then fixes the truth value of every **sentence**

  (i.e. **closed** formula – contains no *free variables*)

  Φ of $\boldsymbol{L}$, **relative** to the interpretation $\boldsymbol{\mathcal{J}}$.
- The formation rules entail that all sentences of $\boldsymbol{L}$ are either atomic or have one of the following 7 forms (as determined by the main 'connective'):

  $\neg\, \Phi$, $(\Phi \wedge \Psi)$, $(\Phi \vee \Psi)$, $(\Phi \rightarrow \Psi)$, $(\Phi \leftrightarrow \Psi)$ $\forall \mathbf{v}\Phi$, $\exists \mathbf{v}\Phi$

  So the following Rules of Truth give the <u>exhaustive</u> procedures for computing the truth value (either **1** for True or **0** for False) for <u>every</u> sentence of $\boldsymbol{L}$ **relative** to a given interpretation $\boldsymbol{\mathcal{J}}$:

Truth in an Interpretation

(I) If Φ is **atomic**, then it's either a sentence letter S or has the form $P^n\, t_1,\ldots, t_n$

(i) If Φ is a sentence letter S, then the truth value of Φ relative to $\mathcal{J}$, written $\mathcal{J}(\Phi)$ is simply $\mathcal{J}(S)$.

(ii) If Φ has the form $P^n\, t_1,\ldots, t_n$, then, because Φ is **closed**, the terms $t_1,\ldots, t_n$ must also be **closed**, and

- $\mathcal{J}(\Phi) = \mathbf{1}$ iff $< \mathcal{J}(t_1),\ldots, \mathcal{J}(t_n) > \in \mathcal{J}(P^n)$ (and $\mathbf{0}$ otherwise), where, for each term t_i in the series $t_1,\ldots, t_n$, if t_i is a constant $\mathbf{c}$ then $\mathcal{J}(t_i) = \mathcal{J}(\mathbf{c})$;

otherwise t_i is a k-place function term $f^k\,(t_1,\ldots, t_k)$ applied to closed terms $t_1,\ldots, t_k$ and $\mathcal{J}(t_i) = \mathcal{J}(f^k)\,\big(\mathcal{J}(t_1), \ldots, \mathcal{J}(t_k)\big)$

Truth in an Interpretation

(II) For **compound** formulas:

(1) $\mathscr{I}(\neg \Phi) = \mathbf{1}$ iff $\mathscr{I}(\Phi) = \mathbf{0}$ (and **0** otherwise).

(2) $\mathscr{I}(\Phi \wedge \Psi) = \mathbf{1}$ iff $\mathscr{I}(\Phi) = \mathbf{1}$ **and** $\mathscr{I}(\Psi) = \mathbf{1}$ (and **0** otherwise).

(3) $\mathscr{I}(\Phi \vee \Psi) = \mathbf{1}$ iff $\mathscr{I}(\Phi) = \mathbf{1}$ **or** $\mathscr{I}(\Psi) = \mathbf{1}$ (and **0** otherwise).

(4) $\mathscr{I}(\Phi \rightarrow \Psi) = \mathbf{1}$ iff $\mathscr{I}(\Phi) = \mathbf{0}$ **or** $\mathscr{I}(\Psi) = \mathbf{1}$ (and **0** otherwise).

(5) $\mathscr{I}(\Phi \leftrightarrow \Psi) = \mathbf{1}$ iff $\mathscr{I}(\Phi) = \mathscr{I}(\Psi)$ (and **0** otherwise).

Truth Table Format

Φ	$\neg \Phi$
1	**0**
0	**1**

Φ	Ψ	$(\Phi \wedge \Psi)$
1	**1**	**1**
1	**0**	**0**
0	**1**	**0**
0	**0**	**0**

Φ	Ψ	$(\Phi \vee \Psi)$
1	**1**	**1**
1	**0**	**1**
0	**1**	**1**
0	**0**	**0**

Φ	Ψ	$(\Phi \rightarrow \Psi)$
1	**1**	**1**
1	**0**	**0**
0	**1**	**1**
0	**0**	**1**

Φ	Ψ	$(\Phi \leftrightarrow \Psi)$
1	**1**	**1**
1	**0**	**0**
0	**1**	**0**
0	**0**	**1**

Truth in an Interpretation

(6) $\mathscr{I}(\forall \mathbf{v}\Phi)$ **= 1** iff $\mathscr{I}_e^a(\Phi\mathbf{v}/a) = 1$ **for every** $e \in D$, where a is a new individual constant, $\mathscr{I}_e^a$ is the interpretation exactly like $\mathscr{I}$ except that $\mathscr{I}_e^a(a) = e$, and $\Phi\mathbf{v}/a$ is the result of substituting a for every free occurrence of **v** in Φ (and **0** otherwise).

(7) $\mathscr{I}(\exists \mathbf{v}\Phi)$ **= 1** iff $\mathscr{I}_e^a(\Phi\mathbf{v}/a) = 1$ **for some** $e \in D$, again where a is a new individual constant, $\mathscr{I}_e^a$ is the interpretation exactly like $\mathscr{I}$ except that $\mathscr{I}_e^a(a) = e$, and $\Phi\mathbf{v}/a$ is the result of substituting a for every free occurrence of **v** in Φ (and **0** otherwise).

Example

- Consider the previous interpretation $\mathscr{I}$, where the domain D is the 2-member set {Jack, Jill},

  $\mathscr{I}(c_1)$ = Jack and $\mathscr{I}(c_2)$ = Jill, $\mathscr{I}(P^1)$ = {Jack}

  $\mathscr{I}(L^2)$ = {<Jack, Jack>, <Jack, Jill>}

  $\mathscr{I}(f^1)$ = {<Jack, Jack>, <Jill, Jack>}

- Now consider the **sentence** P^1c_1.

  $\mathscr{I}(c_1)$ = Jack, and Jack $\in$ {Jack}

  Hence $\mathscr{I}(c_1) \in \mathscr{I}(P^1)$, so $\mathscr{I}(P^1c_1) = \mathbf{1}$

- Consider the **sentence** P^1c_2.

  $\mathscr{I}(c_2)$ = Jill, and Jill $\notin$ {Jack}

  Hence $\mathscr{I}(c_2) \notin \mathscr{I}(P^1)$, so $\mathscr{I}(P^1c_2) = \mathbf{0}$

Example

- Consider the **sentence** $P^1 f^1(c_2)$

  $\mathscr{J}(f^1(c_2)) = \mathscr{J}(f^1)(\mathscr{J}(c_2))$,

  where $(\mathscr{J}(c_2))$ = Jill and $\mathscr{J}(f^1)$ (Jill) = Jack

  As before, Jack $\in$ {Jack}, hence $\mathscr{J}(f^1(c_2)) \in \mathscr{J}(P^1)$,

  so $\mathscr{J}(P^1 f^1(c_2)) = \mathbf{1}$

- Consider the **sentence** $L^2 c_2 c_1$

  $\mathscr{J}(c_2)$ = Jill, $\mathscr{J}(c_1)$ = Jack

  and $\langle \text{Jill, Jack} \rangle \notin \{\langle \text{Jack, Jack} \rangle, \langle \text{Jack, Jill} \rangle\}$

  Hence $\langle \mathscr{J}(c_2), \mathscr{J}(c_1) \rangle \notin \mathscr{J}(L^2)$

  so $\mathscr{J}(L^2 c_2 c_1) = \mathbf{0}$

Example

- Consider the **sentence** $\forall x\, P^1 x$

$\mathcal{J}(\forall x\, P^1 x) = \textbf{?}$

i) $\mathcal{J}^a_{Jack}(P^1 a) = \textbf{?}$

$\mathcal{J}^a_{Jack}(a) = \text{Jack}$

and Jack $\in$ {Jack}, so $\mathcal{J}^a_{Jack}(a) \in \mathcal{J}(P^1)$,

so $\mathcal{J}^a_{Jack}(P^1 a) = \mathbf{1}$

ii) $\mathcal{J}^a_{Jill}(P^1 a) = \textbf{?}$

$\mathcal{J}^a_{Jill}(a) = \text{Jill}$

and Jill $\notin$ {Jack}, so $\mathcal{J}^a_{Jill}(a) \notin \mathcal{J}(P^1)$,

so $\mathcal{J}^a_{Jill}(P^1 a) = \mathbf{0}$

Hence $\mathcal{J}(\forall x\, P^1 x) = \mathbf{0}$

Example

- Consider the **sentence** $\exists x\, P^1 x$

  $\mathcal{I}(\exists x\, P^1 x) = \textbf{?}$

  i) $\mathcal{I}^a_{Jack}(P^1 a) = \textbf{?}$

  $\mathcal{I}^a_{Jack}\,(a) = \text{Jack}$

  and Jack $\in$ {Jack}, so $\mathcal{I}^a_{Jack}\,(a) \in \mathcal{I}(P^1)$,

  so $\mathcal{I}(P^1 a) = \mathbf{1}$

  Hence $\mathcal{I}(\exists x\, P^1 x) = \mathbf{1}$

- Consider the sentence $\forall x\, \exists y\, L^2 xy$
- Consider the sentence $\exists x\, \forall y\, L^2 xy$

Example

- Consider the **sentence** $= (f^1(c_2), c_1)$

$\mathcal{J}(= (f^1(c_2), c_1)) =$ **?**

As before, $\mathcal{J}(f^1(c_2)) = \mathcal{J}(f^1)(\mathcal{J}(c_2))$,

where $(\mathcal{J}(c_2)) =$ Jill and $\mathcal{J}(f^1)$ (Jill) $=$ Jack

And $\mathcal{J}(c_1) =$ Jack

so $<\mathcal{J}(f^1(c_2)), \mathcal{J}(c_1)>$ is <Jack, Jack>.

$\mathcal{J}(=)$ is the set { <Jack, Jack>, < Jill, Jill> }

and <Jack, Jack> $\in$ { <Jack, Jack>, < Jill, Jill> }

So $< \mathcal{J}(f^1(c_2)), \mathcal{J}(c_1) > \in \mathcal{J}(=)$

and hence $\mathcal{J}(= (f^1(c_2), c_1)) =$ **1**

Truth in an Interpretation

- The 'Mates Quantification' scheme in clauses (6) and (7) uses substitution to attain the same semantical results as *variable interpretation sequences*.

  On the Mates approach we don't need to assign values to variables and we only ever need to consider the truth values of **closed formulas**.

- For the interpretation of n-ary predicate letters P^n, B&J explicitly assign a characteristic function, say $\mathcal{C}^n$, of n-tuples of elements of the domain.

  Thus $\mathcal{I}(P^n) = \mathcal{C}^n$ and $\mathcal{I}(P^n\, t_1,\ldots, t_n) = \mathcal{C}^n(\mathcal{I}(t_1),\ldots,\mathcal{I}(t_n))$.

  This is equivalent to assigning a set of n-tuples, as above.

Some Standard Model-Theoretic Notions

- **Some standard model-theoretic notions**:

  (i) $\mathcal{J}$ **satisfies** (or is a model of) Φ iff $\mathcal{J}(\Phi) = 1$

  (ii) Φ is **satisfiable** (or consistent) iff $\mathcal{J}(\Phi) = 1$ **for some** interpretation $\mathcal{J}$

  (iii) Φ is **valid** (or a logical truth) iff $\mathcal{J}(\Phi) = 1$ **for every** interpretation $\mathcal{J}$. In this case we write $\models \Phi$

  (iv) **logical implication**: $\Phi \models \Psi$ iff **for every** interpretation $\mathcal{J}$ such that $\mathcal{J}(\Phi) = 1$, it's the case that $\mathcal{J}(\Psi) = 1$

  in other words, **for all** interpretations $\mathcal{J}$, $\mathcal{J}(\Phi) \leq \mathcal{J}(\Psi)$

  Clearly $\Phi \models \Psi$ iff $\models (\Phi \rightarrow \Psi)$, so logical implication can be expressed in terms of the validity of the material conditional.

Some Standard Model-Theoretic Notions

(v) **logical equivalence**: $\Phi \equiv \Psi$ iff **for all** interpretations $\mathcal{J}$, $\mathcal{J}(\Phi) = \mathcal{J}(\Psi)$.

- Clearly $\Phi \equiv \Psi$ iff $\Phi \vDash \Psi$ **and** $\Psi \vDash \Phi$, iff $\vDash (\Phi \leftrightarrow \Psi)$
- Familiar generalization of logical implication to multiple premises: $\Phi_1,\ldots, \Phi_n \vDash \Psi$ iff for all interpretations $\mathcal{J}$, if $\mathcal{J}$ satisfies each of $\Phi_1,\ldots, \Phi_n$ then $\mathcal{J}$ satisfies Ψ,

  iff $\vDash ((\Phi_1 \wedge \ldots \wedge \Phi_n) \rightarrow \Psi)$
- For a set of sentences $\mathbf{\Gamma}$, $\mathcal{J}$ satisfies $\mathbf{\Gamma}$ iff $\mathcal{J}(\Theta) = 1$

  for every $\Theta \in \mathbf{\Gamma}$. $\mathcal{J}$ is then a model of $\mathbf{\Gamma}$.

The logical consequence relation extended to sets of sentences:

Some Standard Model-Theoretic Notions

- $\mathbf{\Gamma} \vDash \Psi$ iff for every interpretation $\mathcal{J}$, if $\mathcal{J}$ satisfies Γ then $\mathcal{J}$ satisfies Ψ
- It is in the above format that we will think of valid arguments, recasting the 'standard argument form'

	Premise 1
	Premise 2
	…
	Premise *n*
Therefore,	**Conclusion**

so that $\mathbf{\Gamma}$ is the set of premises and Ψ is the conclusion.

A **sentence** Φ is valid iff $\varnothing \vDash \Phi$

Some Standard Model-Theoretic Notions

- Essential connection between implication and satisfiability:

  $\mathbf{\Gamma} \vDash \Psi$ iff the set $\mathbf{\Gamma} \cup \{\neg \Psi\}$ is **unsatisfiable**.

  If the set $\mathbf{\Gamma} \cup \{\neg \Psi\}$ did have a model then it would be a *counterexample* to the claim $\mathbf{\Gamma} \vDash \Psi$.

- **Formal Theories**:

  A Formal Theory **T** is a set of sentences (in some formal language ***L***) which is **closed** under the relation of logical consequence.

  So for all sentences Φ of ***L***, if **T** $\vDash \Phi$ then $\Phi \in$ **T**

More on Truth Functions

Definition: A **truth function** is a function whose only inputs and outputs are the truth values **T** and **F**.

The logical connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$ and $\leftrightarrow$ represent truth functions. These truth functions are given by their truth tables.

For example, the negation truth function (which corresponds to our interpretation of '$\neg$') is a one-place function which maps **T** to **F** and maps **F** to **T**.

The conjunction truth function (our interpretation of '$\wedge$') is a two-place function which maps the pair of arguments **(T, T)** to **T** and maps all other pairs to **F**.

How Many Truth Functions are There?

An interesting theoretical question is:

"How many distinct truth functions are there for ***n*** arguments?".

It is easy to see that for 1 argument, there are 2 assignments, and then 4 different truth functions, which we call $\mathbf{J}_1$, $\mathbf{J}_2$, $\mathbf{J}_3$, $\mathbf{J}_4$.

Thus, here is a table of the **unary truth functions** (truth functions of one argument)

A	$\mathbf{J_1}$	$\mathbf{J_2}$	$\mathbf{J_3}$	$\mathbf{J_4}$
T	**T**	**T**	**F**	**F**
F	**T**	**F**	**T**	**F**

How Many Truth Functions are There?

Similarly, for **binary truth functions** (2 arguments), there are $2^2 = 4$ assignments and $2^4 = 16$ truth functions.

Let's list all the truth functions of 2 arguments.

A	**B**	$\mathbf{K}_5$	$\mathbf{K}_6$	$\mathbf{K}_7$	$\mathbf{K}_8$	$\mathbf{K}_9$	$\mathbf{K}_{10}$	$\mathbf{K}_{11}$	$\mathbf{K}_{12}$
T	**T**	**T**	**F**	**T**	**F**	**T**	**F**	**T**	**F**
T	**F**	**T**	**T**	**F**	**F**	**T**	**T**	**F**	**F**
F	**T**	**T**	**T**	**T**	**T**	**F**	**F**	**F**	**F**
F	**F**	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**

How Many Truth Functions are There?

A	B	$\mathbf{K}_{13}$	$\mathbf{K}_{14}$	$\mathbf{K}_{15}$	$\mathbf{K}_{16}$	$\mathbf{K}_{17}$	$\mathbf{K}_{18}$	$\mathbf{K}_{19}$	$\mathbf{K}_{20}$
T	T	T	F	T	F	T	F	T	F
T	F	T	T	F	F	T	T	F	F
F	T	T	T	T	T	F	F	F	F
F	F	F	F	F	F	F	F	F	F

The truth function $\mathbf{K}_{13}$ is the same as $\vee$,

The truth function $\mathbf{K}_{19}$ is the same as $\wedge$;

$\mathbf{K}_7$ is the same as $\rightarrow$ and $\mathbf{K}_{11}$ is the same as $\leftrightarrow$.

Later, we will see that there is something quite special about the truth functions $\mathbf{K}_6$ and $\mathbf{K}_{12}$.

How Many Truth Functions are There?

For **3** arguments, there are $\mathbf{2^3 = 8}$ assignments, and $\mathbf{2^8 = 256}$ truth functions.

In general, for ***n*** arguments, there are $\mathbf{2}^{\boldsymbol{n}} \mathbf{=} \boldsymbol{k}$ assignments and $\mathbf{2}^{\boldsymbol{k}}$ truth functions.

Since there is *no upper bound* on the number of inputs ***n***, it follows that there are **infinitely many** distinct truth functions.

Truth-functional Adequacy

A logical language **L** is **truth-functionally adequate**

iff

for <u>every</u> one of the (infinitely many) truth functions that exist,

there is a **formula** of **L** that computes the function (i.e. has the same **truth table**).

So a **truth-functionally adequate** language is powerful enough to express <u>all possible</u> truth functions!

Question: is our language **L** truth-functionally adequate?

Answer: It turns out that a proper subset of the language is enough…

<u>Every</u> truth function can be defined using just $\neg$, $\wedge$ and $\vee$.

The Adequacy of $\{\neg, \wedge, \vee\}$

The first thing to notice is that we did not need to introduce the connectives $\rightarrow$ and $\leftrightarrow$ as primitive,

because the truth functions represented by the connectives $\rightarrow$ and $\leftrightarrow$ are **definable** using $\{\neg, \wedge, \vee\}$.

What do we mean by **definable**?

Definition: A **2-place** connective **C** is **definable** using the set of connectives $\{C_1, C_2, \ldots\}$

just in case the formula **C(A, B)** is **logically equivalent** to some formula **[…A…B…]**,

where the expression **[…A…B…]** contains only connectives from the set $\{C_1, C_2, \ldots\}$.

Defining Truth Functions Using $\neg$, $\wedge$ and $\vee$

So the definitions of the truth functions represented by the connectives $\rightarrow$ and $\leftrightarrow$ in terms of **{$\neg$, $\wedge$, $\vee$} can be** given using **truth-functional equivalences**.

By using truth tables you can prove that,

(i) $\mathbf{A \rightarrow B \quad \equiv \quad \neg A \vee B}$

(ii) $\mathbf{A \leftrightarrow B \quad \equiv \quad (A \wedge B) \vee (\neg A \wedge \neg B)}$

Thus, both $\rightarrow$ and $\leftrightarrow$ are **definable** using **{$\neg$, $\wedge$, $\vee$}.**

The Adequacy of $\{\neg, \wedge, \vee\}$

The fact that **every truth function** can be defined

using just **$\neg$,** $\wedge$ and $\vee$

can be established by providing a general procedure such that given *any* arbitrary ***n*-place** truth function $\mathcal{F}$,

it is possible to construct a formula **A**,

using only *n* distinct <u>statement</u> <u>letters</u>

and the connectives **$\neg$,** $\wedge$ and $\vee$**,**

such that **A** has the *same* **truth table** as $\mathcal{F}$**.**

Example

Example: For the sake of illustration, let $\boldsymbol{n} = 3$,

Now, consider the **arbitrary** 3-place truth function $\mathcal{F}$ specified by the truth table:

3 inputs			$\mathcal{F}$
T	**T**	**T**	**T**
T	**T**	**F**	**T**
T	**F**	**T**	**F**
T	**F**	**F**	**F**
F	**T**	**T**	**T**
F	**T**	**F**	**F**
F	**F**	**T**	**F**
F	**F**	**F**	**F**

3 inputs yields

$2^3 = 8$ assignments

yields

$2^8 = 256$ truth functions:

$\mathcal{F}$ is just <u>one</u> of them.

Example

P	Q	R	$\mathcal{F}$	
T	T	T	T	◀
T	T	F	T	
T	F	T	F	
T	F	F	F	
F	T	T	T	
F	T	F	F	
F	F	T	F	
F	F	F	F	

Method of construction:

Consider the *first* assignment where $\mathcal{F}$ yields the output **T**.

Make a conjunction of the 3 corresponding **literals**

(*i.e.* a statement letter or its negation)

in this case **P** $\wedge$ **Q** $\wedge$ **R**

P	Q	R	$\mathcal{F}$
T	T	T	T
T	T	F	T ◀
T	F	T	F
T	F	F	F
F	T	T	T
F	T	F	F
F	F	T	F
F	F	F	F

Then:

Consider the *next* assignment where $\mathcal{F}$ yields the output T.

Make a conjunction of the 3 corresponding **literals**, in this case $\mathbf{P \land Q \land \neg R}$

P	**Q**	**R**	$\mathcal{F}$
T	**T**	**T**	**T**
T	**T**	**F**	**T**
T	**F**	**T**	**F**
T	**F**	**F**	**F**
F	**T**	**T**	**T** ◀
F	**T**	**F**	**F**
F	**F**	**T**	**F**
F	**F**	**F**	**F**

Then:

Consider the *last* assignment where $\mathcal{F}$ yields the output **T**.

Make a conjunction of the 3 corresponding **literals**, in this case $\neg\mathbf{P} \land \mathbf{Q} \land \mathbf{R}$

Finally, let the formula **A** be the **<u>disjunction</u>** of all the **<u>conjunctions</u>** of 3 literals obtained from the input configurations where $\mathcal{F}$ yields the output **T**.

In this case **A** is the formula

$$(\mathbf{P} \wedge \mathbf{Q} \wedge \mathbf{R}) \vee (\mathbf{P} \wedge \mathbf{Q} \wedge \neg\mathbf{R}) \vee (\neg\mathbf{P} \wedge \mathbf{Q} \wedge \mathbf{R})$$

And **A** computes *exactly the same* truth function as $\mathcal{F}$!

This can be confirmed by examining its truth table…

P	Q	R	$\mathcal{F}$	$(P \wedge Q \wedge R) \vee (P \wedge Q \wedge \neg R) \vee (\neg P \wedge Q \wedge R)$			
T	T	T	T	T	F	F	T
T	T	F	T	F	T	F	T
T	F	T	F	F	F	F	F
T	F	F	F	F	F	F	F
F	T	T	T	F	F	T	T
F	T	F	F	F	F	F	F
F	F	T	F	F	F	F	F
F	F	F	F	F	F	F	F

Disjunctive Normal Form Theorem

The formula **A** is in **Disjunctive Normal Form**:

an overall <u>disjunction</u> of <u>conjunctions</u> of sentence literals.

The foregoing method of construction is entirely general,

and underwrites the

Disjunctive Normal Form Theorem:

for any formula **B** in our language **L** of propositional logic,
there is a formula **A** in <u>Disjunctive Normal Form</u> such that

A ≡ **B**.

Defining Truth Functions using $\neg$ and one of $\wedge, \vee$

Just as in the case of $\rightarrow$ and $\leftrightarrow$**,** it turns out that we don't need to take the entire set of connectives **{$\neg$, $\wedge$, $\vee$}** as primitive:

all we need is negation and **one** of our remaining binary connectives.

E.g. $\vee$ is **definable** from the set **{$\neg$, $\wedge$}** as follows:

$$\mathbf{A \vee B \equiv \neg(\neg A \wedge \neg B)}$$

And similarly $\wedge$ is **definable** from the set **{$\neg$, $\vee$}** as follows:

$$\mathbf{A \wedge B \equiv \neg(\neg A \vee \neg B)}$$

And this shows that the sets **{$\neg$, $\wedge$}** and **{$\neg$, $\vee$}** are both truth-functionally adequate.

Adequate Sets of Connectives (Cont.)

While **{¬, ∧, ∨}**, **{¬, ∧}** and **{¬, ∨}** are all **adequate set of connectives,**

some sets of connectives are **not adequate**.

For example, the set **{∧, ∨}** is **not** adequate.

You cannot define negation **¬** using **{∧, ∨}**.

But **{¬, ∧}** and **{¬, ∨}** are **not** the *smallest* adequate sets.

We will now show that one can find a **single** 2-place connective which is adequate

(in fact there are *two* such connectives).

That is, a single connective **K** such that *any* logical connective (representing any truth function) can be defined

using just **K** alone.

Two New Connectives: NAND | and NOR ↓

Let us define two new truth-functional binary connectives:

NAND, meaning **'not (… and …)'**

NOR, meaning **'not (… or …)'**

NAND(A, B) **written $A \mid B$** $\equiv \neg(A \wedge B)$

NOR(A, B) **written $A \downarrow B$** $\equiv \neg(A \vee B)$

NAND $|$ and NOR $\downarrow$

Given these definitions we can quickly figure out their truth tables

$$\neg(A \wedge B) \qquad \neg(A \vee B)$$

A	**B**	**A|B**	**A↓B**
T	**T**	**F**	**F**
T	**F**	**T**	**F**
F	**T**	**T**	**F**
F	**F**	**T**	**T**

These correspond to the truth functions we called $\mathbf{K_6}$ and $\mathbf{K_{12}}$ above.

NAND { | } is an Adequate Set!

Every truth functional connective can be defined using { | } alone

Since we already know that **{¬, ∧}** is adequate, we just need to show how to define negation ¬ and conjunction ∧ from |.

It will then immediately follow that { | **}** alone is adequate.

The crucial trick is to define negation ¬ using |.

Consider the truth table for any formula of the form **A | A**.

A	**A | A**
T	**F**
F	**T**

Thus, we see that

(1) **¬A** ≡ **(A | A)**

[A similar truth table will reveal that: **¬A ≡ (A ↓ A).**]

NAND { | } is an Adequate Set

Next we want to define $\wedge$ from |.

Since $\mathbf{(A \mid B) \equiv \neg (A \wedge B)}$ and $\mathbf{A \wedge B \equiv \neg\neg(A \wedge B)}$ it follows that $\mathbf{A \wedge B \equiv \neg(A \mid B).}$

Thus,

$$(2) \qquad \mathbf{A \wedge B} \qquad \equiv \qquad \mathbf{\neg(A \mid B)} \qquad \equiv \qquad \mathbf{(A \mid B) \mid (A \mid B)}$$

I.e., using the definition of $\mathbf{\neg,}$ we convert $\mathbf{\neg(\ldots)}$ to $\mathbf{(\ldots) \mid (\ldots)}$.

(1) and (2) mean that both $\neg$ and $\wedge$ can be defined using | alone.

From this it follows that { | **}** alone is an adequate set.

NAND { | } is an Adequate Set

How do we find a formula equivalent to $\mathbf{A \lor B}$?

$$\mathbf{A \lor B} \quad \equiv \quad \mathbf{\neg(\neg A \land \neg B)}$$

$$\equiv \quad \mathbf{\neg((A \mid A) \land (B \mid B))}$$

$$\equiv \quad \mathbf{(A \mid A) \mid (B \mid B)}$$

So, we have

(3) $\mathbf{A \lor B \quad \equiv \quad \neg((A|A) \land (B|B)) \;\equiv\; (A \mid A) \mid (B \mid B)}$

Exactly analogous reasoning involving $\{\mathbf{\neg, \lor}\}$ shows that $\{\mathbf{\downarrow}\}$ is also adequate.

Historical Note

- The American logician Charles Saunders Peirce discovered the truth-functional adequacy of both NAND and NOR in 1880, but never published his findings.
- Henry Sheffer independently published results on the adequacy of NOR in 1913, whence the ‘Sheffer stroke’.
- The NAND logic gate is crucial to modern digital electronics, and plays a vital role in computer processor design.

Prenex Normal Form

- **Prenex Form**:

  $Q_1\mathbf{v}_1,\ldots, Q_n\mathbf{v}_n\ \Phi$ where $Q_i = \forall$ or $\exists$ and Φ is quantifier free

  Theorem: for every formula Ψ there is some formula Θ such that Θ is in prenex form and $\Psi \equiv \Theta$.

  Proof: give rules for successively moving quantifiers to the left which preserve logical equivalence:

  (i) quantifier duality: $\neg\, Q\mathbf{v}\ \Phi \equiv Q'\mathbf{v}\ \neg\, \Phi$

  where $Q = \forall$ or $\exists$, and $\forall' = \exists$, $\exists' = \forall$

  (ii) can directly pull quantifiers out from conjunctions, disjunctions, and the consequents of conditionals (provided $\mathbf{v}$ does not occur free in Ψ):

Prenex Normal Form

(1) $(Q\mathbf{v}\ \Phi \wedge \Psi) \equiv Q\mathbf{v}\ (\Phi \wedge \Psi)$

$(\Psi \wedge Q\mathbf{v}\ \Phi) \equiv Q\mathbf{v}\ (\Psi \wedge \Phi)$

(2) $(Q\mathbf{v}\ \Phi \vee \Psi) \equiv Q\mathbf{v}\ (\Phi \vee \Psi)$

$(\Psi \vee Q\mathbf{v}\ \Phi) \equiv Q\mathbf{v}\ (\Psi \vee \Phi)$

(3) $(\Psi \longrightarrow Q\mathbf{v}\ \Phi) \equiv Q\mathbf{v}\ (\Psi \longrightarrow \Phi)$

(4) But must reverse quantifier in the antecedent

$(Q\mathbf{v}\ \Phi \longrightarrow \Psi) \equiv Q'\mathbf{v}\ (\Phi \longrightarrow \Psi)$

Because $(Q\mathbf{v}\ \Phi \longrightarrow \Psi) \equiv (\neg\ Q\mathbf{v}\ \Phi \vee \Psi) \equiv (Q'\mathbf{v}\ \neg\ \Phi \vee \Psi) \equiv$

$Q'\mathbf{v}\ (\neg\ \Phi \vee \Psi) \equiv Q'\mathbf{v}\ (\Phi \longrightarrow \Psi)$ ■

Examples

- $\forall y Qy \vee \neg\exists x Px$

  $\equiv \quad \forall y Qy \vee \forall x \neg Px$

  $\equiv \quad \forall y\,(Qy \vee \forall x\,\neg Px)$

  $\equiv \quad \forall y \forall x\,(Qy \vee \neg Px)$

- $\exists x Fx \longrightarrow \neg\,\forall y Py$

  $\equiv \quad \exists x Fx \longrightarrow \exists y\,\neg Py$

  $\equiv \quad \forall x\,(Fx \longrightarrow \exists y\,\neg Py)$

  $\equiv \quad \forall x\,\exists y\,(Fx \longrightarrow \neg Py)$

- We will use prenex normal form in our forthcoming deductive system.