INFO0051-1

LOGIC

LoGeek ?

References

— M. Ben-Ari, *Mathematical Logic for Computer Science*, Prentice-Hall, 1993.
— R. Cori et D. Lascar, *Logique mathématique* (deux volumes), Masson, 1993.
— H. Enderton, *A Mathematical Introduction to Logic*, Academic Press 1972, 2001.
— M. Fitting, *First-Order Logic and Automated Theorem Proving*, Springer-Verlag, 1990.
— J.H. Gallier, *Logic for Computer Science*, Harper & Row, 1986.
— P. Gochet et P. Gribomont, *Logique 1 – Méthodes pour l'informatique fondamentale*, Hermès, 1991 (2ème tirage), 1998 (3ème tirage).
— P. Gochet et P. Gribomont, *Logique 2 – Méthodes pour l'étude des programmes*, Hermès, 1994.
— P. Gribomont, `http://www.montefiore.ulg.ac.be/~gribomon/cours/info0510.html`
— R. Lassaigne et M. de Rougemont, *Logique et fondements de l'informatique*, Hermès, 1993.
— A. Thayse & co-auteurs, *Approche logique de l'intelligence artificielle*, Dunod, 1990.

INTRODUCTION

Logic : *Science of reasoning for itself*

Aristotle (384 BC - 322 BC)

Deduction system, logical rules

What is a sound logical rule ?

If the *premises* are true,
then the *conclusion* is also true

Example :

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

1. Beware of the natural language!

   Example :

   (a) Some cars rattle
   (b) My car is some car
   (c) Therefore, my car rattles

2. Paradoxes

   This sentence is false
   The following sentence is true. The previous sentence is false.

Since 1850 : growing interest of mathematicians. Sound logical rules for sound mathematical proofs.

Modern logic : formal logic, mathematical logic

Formal logic : deals with methods

— to determine whether a deduction is correct or not

— that can be verified by a computer

$\Rightarrow$ so a formal language is needed

The language allows to write sentences, statements, propositions

A proposition can be true or false

Example : It rains.

A piece of reasoning has a logical structure,
distinct from information about the underlying world.

- Logical structure : *formula* (or set of formulas)
  $\rightarrow$ *syntax*
- Information about the underlying world : *interpretation*
  $\rightarrow$ *semantics*

Example : If it rains, then the road is wet.

Valid, correct deductions : sound for all interpretations

Logic : which deductions are valid ?

Propositional calculus : formal language ; the sentences are propositions, which are true or false.

Examples :

— $1+1=2$
— We have logic on Sunday
— $1+1=2$ *and* we have logic on Sunday

Predicate logic : more expressive formal language ; allows to write properties and determine sets and set membership.

Examples :

— I teach(x,y)
— $x < y$

PROPOSITIONAL CALCULUS

Proposition : sentence with a truthvalue

Atomic propositions, *Boolean connectives* $->$ compound propositions

Which connectives ?

Verifunctional connectives : Only the truthvalues of the components are needed to know the truthvalue of the compound proposition.

formal connectives, natural connectives :

It rains *or* the sun shines.

If it rains, *then* the road is wet.

He is clever *and* (he is) hardworking.

He is *not* dumb.

If Heads *then* I win *else* you lose.

‘*because*’ is not a Boolean connective!

The road is wet because it rains.

The road is wet because it is Tuesday.

The truthvalue of a because-sentence does not only depend on the truthvalue of the components

Boolean connectives

There are 2^{2^n} n-ary Boolean connectives ($y = op(x_1, \ldots, x_n)$), since the truthtable of an n-ary connective has 2^n lines, each of them being true or false.

Unary (monadic) connectives (4)

x	$\circ_1$	$\circ_2$	$\circ_3$	$\circ_4$
T	T	T	F	F
F	T	F	T	F

Binary (dyadic) connectives (16)

x	y	$\circ_1$	$\circ_2$	$\circ_3$	$\circ_4$	$\circ_5$	$\circ_6$	$\circ_7$	$\circ_8$	$\circ_9$	$\circ_{10}$	$\circ_{11}$	$\circ_{12}$	$\circ_{13}$	$\circ_{14}$	$\circ_{15}$	$\circ_{16}$
T	T	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F
T	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F
F	T	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F
F	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F

Usual binary connectives

op.	name	symbol	
$\circ_2$	disjunction	$\vee$	or
$\circ_3$	converse conditional	$\Leftarrow$	if
$\circ_5$	conditional	$\Rightarrow$, $\supset$	if . . . then
$\circ_7$	biconditional	$\equiv$, $\Leftrightarrow$	if and only if *iff*
$\circ_8$	conjunction	$\wedge$	and
$\circ_9$		$\uparrow$	*nand*
$\circ_{10}$	exclusive or	$\oplus$, $\veebar$	*xor*
$\circ_{15}$		$\downarrow$	*nor*

x	y	$\wedge$	$\vee$	$\equiv$	$\oplus$	$\Rightarrow$
T	T	T	T	T	F	T
T	F	F	T	F	T	F
F	T	F	T	F	T	T
F	F	F	F	T	F	T

n-ary connectives

Theorem. Every n-ary connective ($n > 2$) can be simulated with two $(n-1)$-ary connectives, binary connectives and negations.

$$M(p_1, \ldots, p_{n-1}, p_n) \quad \longleftrightarrow$$
$$[(p_n \Rightarrow M(p_1, \ldots, p_{n-1}, \mathit{true})) \ \wedge \ (\neg p_n \Rightarrow M(p_1, \ldots, p_{n-1}, \mathit{false}))]$$

$$M(p_1, \ldots, p_{n-1}, p_n) \quad \longleftrightarrow$$
$$[(p_n \wedge M(p_1, \ldots, p_{n-1}, \mathit{true})) \ \vee \ (\neg p_n \wedge M(p_1, \ldots, p_{n-1}, \mathit{false}))]$$

Corollary. Every n-ary connective ($n > 2$) can be simulated with binary connectives and negations. (Elementary mathematical induction.)

So n-ary ($n > 2$) connectives can be omitted.

Example (ternary connective) : if p then q else r

Possible reductions :

$$(p \Rightarrow q) \wedge (\neg p \Rightarrow r) \ ; \ \ (p \wedge q) \vee (\neg p \wedge r) \ ; \ \ (p \wedge q) \ \veebar \ (\neg p \wedge r)\,.$$

SYNTAX OF PROPOSITIONAL CALCULUS

Let $\mathcal{P}$ be a set of *atomic propositions* or *atoms*. $\mathcal{P} = \{p, q, r, \ldots\}$

Definition. A *formula* of propositional calculus is a symbol string generated by the grammar

$$\begin{array}{lcl} formula & ::= & p\,, \quad \text{for all } p \in \mathcal{P} \\ formula & ::= & true \mid false \\ formula & ::= & \neg formula \\ formula & ::= & (formula\ \ op\ \ formula) \\ op & ::= & \vee \mid \wedge \mid \Rightarrow \mid \equiv \mid \Leftarrow \end{array}$$

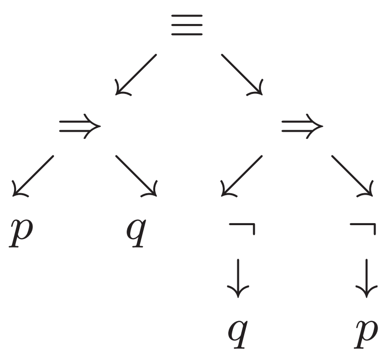
Example : Derivation of formula $(p \wedge q)$:

1. *formula*
2. (*formula* op *formula*)
3. (*formula* $\wedge$ *formula*)
4. ($p \wedge$ *formula*)
5. $(p \wedge q)$

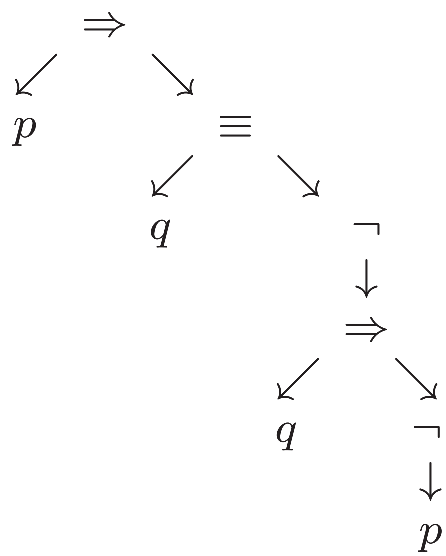
A derivation :

1. $formula$
2. $(formula \equiv formula)$
3. $((formula \Rightarrow formula) \equiv formula)$
4. $((p \Rightarrow formula) \equiv formula)$
5. $((p \Rightarrow q) \equiv formula)$
6. $((p \Rightarrow q) \equiv (formula \Rightarrow formula))$
7. $((p \Rightarrow q) \equiv (\neg formula \Rightarrow formula))$
8. $((p \Rightarrow q) \equiv (\neg q \Rightarrow formula))$
9. $((p \Rightarrow q) \equiv (\neg q \Rightarrow \neg formula))$
10. $((p \Rightarrow q) \equiv (\neg q \Rightarrow \neg p))$

Partial ordering, so syntactic tree :



$((p \Rightarrow q) \equiv (\neg q \Rightarrow \neg p))$



$(p \Rightarrow (q \equiv \neg(q \Rightarrow \neg p)))$

Simplification rules.

We use these rules :

- Outer parentheses can be omitted :
  $p \wedge q$ instead of $(p \wedge q)$, and
  $q \equiv \neg(q \Rightarrow \neg q)$ instead of $(q \equiv \neg(q \Rightarrow \neg q))$.
- Associativity of connectives $\wedge$ and $\vee$:
  $p \vee q \vee r$ instead of $(p \vee q) \vee r$ or $p \vee (q \vee r)$.

We **do not** use these rules :

- Left associativity :
  $p \Rightarrow q \Rightarrow r$ instead of $(p \Rightarrow q) \Rightarrow r$.
- Priority :
  $a + b * c \ = \ a + (b * c) \ \neq \ (a + b) * c$,
  $p \vee q \wedge r$ means $p \vee (q \wedge r)$ and not $(p \vee q) \wedge r$.
  Decreasing priority order sequence : $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftarrow$, $\equiv$

SEMANTICS OF PROPOSITIONAL CALCULUS

Semantics is defined according to the syntactic structure :

Syntax : A *formula* of propositional calculus is generated by the following context-free grammar :

$$
\begin{array}{lcl}
formula & ::= & p\,, \quad \text{for all } p \in \mathcal{P} \\
formula & ::= & true \mid false \\
formula & ::= & \neg formula \\
formula & ::= & (formula\ \ op\ \ formula) \\
op & ::= & \vee \mid \wedge \mid \Rightarrow \mid \equiv \mid \Leftarrow
\end{array}
$$

Compositional, verifunctional semantics : the semantics (truthvalue) of a formula depends only on the semantics (truthvalue) of its components.

Interpretation (or valuation)

Let A be a formula of propositional logic and $\{p_1, \ldots, p_n\}$ the set of atoms occurring in A.

An *interpretation* of A is a function $v : \{p_1, \ldots, p_n\} \rightarrow \{T, F\}$.
The domain of this function can be extended ; v assigns a truthvalue to A according to the following inductive rules :

A	$v(A_1)$	$v(A_2)$	$v(A)$
true			T
false			F
$\neg A_1$	T		F
$\neg A_1$	F		T
$A_1 \vee A_2$	F	F	F
$A_1 \vee A_2$	else		T
$A_1 \wedge A_2$	T	T	T
$A_1 \wedge A_2$	else		F
$A_1 \Rightarrow A_2$	T	F	F
$A_1 \Rightarrow A_2$	else		T
$A_1 \Leftarrow A_2$	F	T	F
$A_1 \Leftarrow A_2$	else		T
$A_1 \equiv A_2$	$v(A_1) = v(A_2)$		T
$A_1 \equiv A_2$	$v(A_1) \neq v(A_2)$		F

Examples

Formula :

$$(p \Rightarrow q) \equiv (\neg q \Rightarrow \neg p)$$

Interpretation :

$$v(p) = F,\ v(q) = T$$

The truthvalue is obtained easily :

$$\begin{aligned}
&v(p \Rightarrow q) = T \\
&v(\neg\, q) = F \\
&v(\neg\, p) = T \\
&v(\neg\, q \Rightarrow \neg\, p) = T \\
&v((p \Rightarrow q) \equiv (\neg\, q \Rightarrow \neg\, p)) = T
\end{aligned}$$

Comment. Formula *true* is a syntactic object ; truthvalue T is a semantic object.

Valuations are functional relations; each formula gets only one truthvalue.

Example : Let $v(p) = F$ and $v(q) = T$.

Therefore, $v(p \Rightarrow (q \Rightarrow p)) = T$ and $v((p \Rightarrow q) \Rightarrow p) = F$.

Example : If $v(p) = T$, $v(q) = F$, $v(r) = T$, $v(s) = T$ what about

$$\{p \Rightarrow q,\ p,\ (p \vee s) \equiv (s \wedge q)\}$$

v assigns specific truthvalues :

$$\begin{array}{l} v(p \Rightarrow q) = F \\ v(p) = T \\ v((p \vee s) \equiv (s \wedge q)) = F \end{array}$$

Propositional logic vs. natural language

Natural connectives are not always verifunctional.

- Connective *and* :

The sun shines and I have a car.
I have a car and the sun shines.

He became afraid and shoot the intruder.
He shoot the intruder and became afraid.

- Connective *or* (exclusive or not) :

An integer is even or (it is) odd.

- The conditional connective :

$$\textit{antecedent} \Rightarrow \textit{consequent}$$

When the antecedent is true, the conditional (truthvalue) reduces to the consequent (truthvalue).

If the Earth rotates around the Sun, *then* 1+1 = 3.

When the antecedent is false, the conditional is true.

If the Sun rotates around the Moon, *then* 1+1 = 3.

Only connective satisfying :

- When the antecedent is true,
  the conditional reduces to the consequent.
- The conditional is a true binary connective.
- The conditional is not commutative.

Every square integer is positive.

For all (integer) n, if n is a square, then n is positive.

$\forall n\,[Sq(n) \Rightarrow Pos(n)]\,.$

Sq–Pos	0, 1, 4, . . . , 100, . . .
Sq–n-Pos	
n-Sq–Pos	2, 3, 5, . . . , 99, 101, . . .
n-Sq–n-Pos	−1, −2, −3, . . . , −100, . . .

The Boolean conditional is exactly the mathematical conditional.

Satisfiability (consistency) and validity

— A valuation v of formula A is a *model* of A if $v(A) = T$.

— A is *satisfiable* or *consistent* if A has at least one model.

— A is *valid*, or A is a *tautology*,
if $v(A) = T$ for each interpretation v.
Notation : $\models A$

— A is *unsatisfiable* or *inconsistent* if A is not satisfiable,
that is, if $v(A) = F$, for each interpretation v.

Theorem.
A formula A is valid if and only if its negation $\neg A$ is unsatisfiable.

A valid
iff $v(A) = T$, for each interpretation v
iff $v(\neg A) = F$, for each interpretation v
iff $\neg A$ is unsatisfiable. □

Decision procedure

Definition. Let U be a formula set (i.e., a set of formulas). An algorithm is a *decision procedure* for U if, given A, the computation stops with the answer 'yes' if $A \in U$ and the answer 'no' if $A \notin U$.

Formal logic : often U will be the set of valid formulas (or consistent formulas, or inconsistent formulas)

If $\neg A$ is satisfiable, then A is not valid.
If $\neg A$ is not satisfiable, then A is valid.

$\hookrightarrow$ *Refutation procedure* :
A is proved valid if $\neg A$ is proved unsatisfiable.

Comment. "X set" stands for "set of Xs".

Formula sets

A *model* of S is an interpretation which assigns T to all formulas in S.

A (formula) set S is *consistent*, or *satisfiable*,
if if S has at least one model.

- For $\emptyset$, every valuation is a model.
- The models of $\{A\}$ are the models of A.
- The models of the finite set $\{A_1, \dots, A_n\}$ are the models of the conjunction $A_1 \wedge \cdots \wedge A_n$.

!! Beware !!

- Infinite sets are acceptable ; infinite formulas are not.
- A set of satisfiable formulas can be an unsatisfiable set.

Set consistency

- Every subset of a consistent set is consistent.
- Every superset of an inconsistent set is inconsistent.
- If S is consistent and if A is valid, then $S \cup \{A\}$ is consistent.
- If S is inconsistent and if A est valid, then $S \setminus \{A\}$ is inconsistent.

Removing formulas preserves consistency.

Adding formulas preserves inconsistency.

Adding valid formulas preserves consistency.

Removing valid formulas preserves inconsistency.

Logical consequence

A formula A is a *logical consequence* of a formula set S if every S-model is an A-model.

Notation : $S \models A$.

Comment. If S is valid, for instance if $S = \emptyset$, then A is a logical consequence of S if and only if A is valid.

$$\models A$$

can be seen as

$$\emptyset \models A$$

A formula is valid iff it is a logical consequence of the empty set.

A formula is valid iff it is a logical consequence of every formula set.

Deduction theorem

Let A be a formula and $U = \{A_1, \ldots, A_n\}$ a finite formula set.
Three equivalent statements are :

— A is a logical consequence of U,
$U \models A$;

— Set $U \cup \{\neg A\}$ is inconsistent,
$U \cup \{\neg A\} \models \mathit{false}$;

— Conditional $(A_1 \wedge \ldots \wedge A_n) \Rightarrow A$ is valid,
$\models (A_1 \wedge \ldots \wedge A_n) \Rightarrow A$.

Statements 1 and 2 still hold when U is infinite. Statement 3 does not.

Definition. The *theory* associated with formula set U is the set of logical consequences of U : $\mathcal{T}(U) = \{A : U \models A\}$; U-members are *axioms* and $\mathcal{T}(U)$-members are *theorems*. (Most often used in predicate logic.)

Consistency and inconsistency

The useful statements

Removing valid formulas from an inconsistent set leads to an inconsistent set.

Adding valid formulas to a consistent set leads to a consistent set.

can be improved into

Removing logical consequences (of what is not removed) from an inconsistent set leads to an inconsistent set.

Adding logical consequences to a consistent set leads to a consistent set.

Logical equivalence

Definition. Two propositional formulas A_1 and A_2 are *logically equivalent* (noted $A_1 \leftrightarrow A_2$) if they have the same models, that is, if $v(A_1) = v(A_2)$ for each interpretation v.

Example : $p \vee q \leftrightarrow q \vee p$

p	q	$v(p \vee q)$	$v(q \vee p)$
T	T	T	T
T	F	T	T
F	T	T	T
F	F	F	F

If A_1 and A_2 are propositional formulas then $A_1 \vee A_2 \leftrightarrow A_2 \vee A_1$.

Let v be an interpretation of $\{A_1, A_2\}$.

$v(A_1 \vee A_2) = T$ iff $v(A_1) = T$ or $v(A_2) = T$;
iff $v(A_2) = T$ or $v(A_1) = T$;
iff $v(A_2 \vee A_1) = T$.

Therefore, $A_1 \vee A_2 \leftrightarrow A_2 \vee A_1$.

Logical equivalence $\neq$ $\equiv$-connective !

There is a difference between

— *the object-language* : logic itself
($\equiv$, $p \vee q$, . . .)

— *the metalanguage* : semi-formal language used to comment about logic ($\leftrightarrow$, $A_1 \vee A_2$, . . .) !

There is a connection between logical equivalence (a metalinguistic notion) and the $\equiv$-connective (a logical object).

Theorem (logical equivalence vs. equivalence connective) :
$A_1 \leftrightarrow A_2$ iff $\models A_1 \equiv A_2$.

$v(A_1 \equiv A_2) = T$, for each interpretation v

iff $v(A_1) = v(A_2)$, for each interpretation v
(inductive definition of the semantics of '$\equiv$')

iff $A_1 \leftrightarrow A_2$ (definition of the logical equivalence '$\leftrightarrow$').

Some theorems

These four statements are equivalent :

— $A_1 \leftrightarrow A_2$;
— $\models (A_1 \equiv A_2)$;
— $\models (A_1 \Rightarrow A_2)$ and $\models (A_2 \Rightarrow A_1)$;
— $\{A_1\} \models A_2$ et $\{A_2\} \models A_1$.

Comment.
$A \models B$ can be written instead of $\{A\} \models B$, and
$E, A, B \models C$ can be written instead of $E \cup \{A, B\} \models C$.
(We use this in the sequel.)

Comment. $v \models A$ is sometimes written instead of $v(A) = T$, since interpretation v is sometimes identified with the set of v-true formulas. (We do not use this.)

Subformulas

- A is a *subformula* of B if the syntactic tree of A if a subtree of the syntactic tree of B.
- A is a *proper subformula* of B if A is a subformula of B distinct from B.

Examples :

- $p \Rightarrow q$ is a proper subformula of $(p \Rightarrow q) \equiv (\neg q \Rightarrow \neg p)$;
- $p \Rightarrow q$ is a (not proper) subformula of $p \Rightarrow q$;
- $q \equiv \neg q$ is not a subformula of $(p \Rightarrow q) \equiv (\neg q \Rightarrow \neg p)$.

Comment. A subformula can have several occurrences in a formula. For instance, (sub-)formula $\neg p$ has two occurrences in formula $\neg p \equiv \neg(p \Leftarrow \neg p)$; (sub-)formula p has three occurrences in formula $p \equiv (p \Leftarrow \neg p)$.

Replacement theorem

Let A, B and C_A be three formulas, such that A is a sub-formula of C_A, and let C_B be the formula resulting from the replacement of one or more occurrence(s) of A by B in C_A.
The *replacement theorem* states

$$(A \equiv B) \models (C_A \equiv C_B)\,.$$

Corollary. If $A \leftrightarrow B$, then $C_A \leftrightarrow C_B$.

Example.
$A =_{def} p$, $B =_{def} \neg\neg p$
$C_A =_{def} (p \Rightarrow q) \equiv (\neg q \Rightarrow \neg p)$

Three possibles choices for C_B :
$C_B =_{def} (\neg\neg p \Rightarrow q) \equiv (\neg q \Rightarrow \neg p)$
$C_B =_{def} (p \Rightarrow q) \equiv (\neg q \Rightarrow \neg\neg\neg p)$
$C_B =_{def} (\neg\neg p \Rightarrow q) \equiv (\neg q \Rightarrow \neg\neg\neg p)$.

Since A and B are logically equivalent, C_A and C_B are also logically equivalent.

Comment. We could write “the replacement of some (zero, one of more) occurrence(s)” instead of “the replacement of one of more occurrence(s)”

Replacement theorem, the proof

The case $C_B = C_A$ (zero occurrence replacement) is trivial.

Interesting case : one occurrence is replaced.

Proof. We view formulas as syntactic trees and use *mathematical induction* on the depth d of A in C_A.

Let v be an interpretation such that $v(A) = v(B)$.

— $d = 0$: $A = C_A$ and $B = C_B$, therefore $v(C_A) = v(C_B)$.

— $d > 0$: C_A can be written either as $\neg D_A$ or as (D_A *op* E_A), so C_B is either $\neg D_B$ or (D_B *op* E_B). (In the second case, one of the terms D_A, E_A contains the occurrence A to be replaced.)

If A occurs in D_A, the depth of A (in D_A) is less then d, so $v(D_B) = v(D_A)$ (inductive step) ; similarly $v(E_B) = v(E_A)$.

From $v(D_B) = v(D_A)$ and (maybe) $v(E_B) = v(E_A)$, we deduce $v(C_B) = v(C_A)$.

Algebraic laws (examples)

$$(X \wedge X) \longleftrightarrow X \longleftrightarrow (X \vee X)$$
$$(X \wedge Y) \longleftrightarrow (Y \wedge X)$$
$$(X \vee Y) \longleftrightarrow (Y \vee X)$$
$$((X \wedge Y) \wedge Z) \longleftrightarrow (X \wedge (Y \wedge Z))$$
$$((X \vee Y) \vee Z) \longleftrightarrow (X \vee (Y \vee Z))$$

$$(X \Rightarrow X) \longleftrightarrow \mathit{true}$$
$$((X \Rightarrow Y) \wedge (Y \Rightarrow X)) \longleftrightarrow (X \equiv Y)$$
$$(((X \Rightarrow Y) \wedge (Y \Rightarrow Z)) \Rightarrow (X \Rightarrow Z)) \longleftrightarrow \mathit{true}$$
$$(X \Rightarrow Y) \longleftrightarrow ((X \wedge Y) \equiv X)$$
$$(X \Rightarrow Y) \longleftrightarrow ((X \vee Y) \equiv Y)$$

$$(X \wedge (Y \vee Z)) \longleftrightarrow ((X \wedge Y) \vee (X \wedge Z))$$
$$(X \vee (Y \wedge Z)) \longleftrightarrow ((X \vee Y) \wedge (X \vee Z))$$
$$(X \Rightarrow (Y \Rightarrow Z)) \longleftrightarrow ((X \Rightarrow Y) \Rightarrow (X \Rightarrow Z))$$

$$(X \vee \neg X) \longleftrightarrow \mathit{true}$$
$$(X \wedge \neg X) \longleftrightarrow \mathit{false}$$
$$\neg\neg X \longleftrightarrow X$$

$$\neg(X \wedge Y) \longleftrightarrow (\neg X \vee \neg Y)$$
$$\neg(X \vee Y) \longleftrightarrow (\neg X \wedge \neg Y)$$

- Algebraic laws lead to simplifications.

  Example :

$$\begin{aligned} p \wedge (\neg p \vee q) &\leftrightarrow (p \wedge \neg p) \vee (p \wedge q) \\ &\leftrightarrow \textsf{false} \vee (p \wedge q) \\ &\leftrightarrow p \wedge q \end{aligned}$$

- Connective properties are stated as logical equivalences.
  - associativity, commutativity of $\wedge$, $\vee$, $\equiv$
  - idempotence of $\wedge$, $\vee$
  - ...
- All Boolean connectives can be derived from
  - $\neg$ and $\wedge$
  - *Nand* alone
  - *Nor* alone

Minimal connective systems

— $\neg$ and $\vee$:

$(a \Rightarrow b) =_{def} (\neg a \vee b)$,

$(a \wedge b) =_{def} \neg(\neg a \vee \neg b)$.

— $\neg$ and $\wedge$:

$(a \Rightarrow b) =_{def} \neg(a \wedge \neg b)$,

$(a \vee b) =_{def} \neg(\neg a \wedge \neg b)$.

— $\neg$ and $\Rightarrow$:

$(a \vee b) =_{def} (\neg a \Rightarrow b)$,

$(a \wedge b) =_{def} \neg(a \Rightarrow \neg b)$.

— "*nand*" alone (symbol : $\uparrow$)

$\neg a =_{def} (a \uparrow a)$, $(a \wedge b) =_{def} \neg(a \uparrow b)$.

— "*nor*" alone (symbol : $|$, or $\downarrow$)

$\neg a =_{def} (a \mid a)$, $(a \vee b) =_{def} \neg(a \mid b)$.

Uniform substitution

Let A_1 and A_2 be formulas and B be formula $A_1 \Rightarrow (A_1 \vee A_2)$. Even if A_1 and A_2 are complex formulas, B is obviously valid, as an "instance" of the tautology $p \Rightarrow (p \vee q)$.

If C is a formula, $C(p/A_1, q/A_2)$ is obtained by replacing *all* occurrences of propositions p and q in C by formulas A_1 and A_2, respectively.

Lemma. Let $C, A_1, \ldots, A_n$ be formulas and $p_1, \ldots, p_n$ be distinct propositions. If v is a valuation such that $v(p_i) = v(A_i)$ $(i = 1, \ldots, n)$, then $v(C(p_1/A_1, \ldots, p_n/A_n)) = v(C)$.
Comment. The domain of valuation v contains every atom occurring in C and/or in some A_i.
Comment. Formulas $C(p/A_1, q/A_2)$, $C(p/A_1)(q/A_2)$ and $C(q/A_2)(p/A_1)$ can differ !

Uniform substitution (example)

$n =_{def}$ 2
$C =_{def} p_1 \vee (q \Rightarrow p_2)$
$A_1 =_{def} p_2 \wedge (p_1 \vee r)$
$A_2 =_{def} p_1 \vee q$
$C(p_1/A_1, p_2/A_2) =_{def}$
$\qquad (p_2 \wedge (p_1 \vee r)) \vee (q \Rightarrow (p_1 \vee q))$
$v =_{def} \{(p_1, F), (p_2, T), (q, T), (r, F)\}$

$v(A_1) = v(p_1) = F$ and $v(A_2) = v(p_2) = T$, so
$v(C(p_1/A_1, p_2/A_2)) = v(C) = T$.

Comment. If no proposition p_i has any occurrence in $\{A_1, \ldots, A_n\}$, then formulas $C(p_1/A_1, \ldots, p_n/A_n)$ and $C(p_1/A_1) \ldots (p_n/A_n)$ are the same.

Uniform substitution, proving the lemma

We use mathematical induction on the *structure* of formula C.
Let C' be $C(p_1/A_1, \ldots, p_n/A_n)$.

Base case. C is a proposition.
If C is one of the p_i, then $v(C') = v(A_i) = v(p_i) = v(C)$
else $C' = C$, therefore $v(C') = v(C)$.

Induction step. If C is formula $\neg D$, then C' is formula $\neg D'$.
The induction hypothesis is $v(D') = v(D)$; the thesis $v(C') = v(C)$ is a straightforward consequence.
If C is, say, $D \vee E$, then C' is formula $D' \vee E'$, with $v(D') = v(D)$ and $v(E') = v(E)$, hence $v(C') = v(C)$.

Uniform substitution theorem

Theorem. Let $C, A_1, \ldots, A_n$ be formulas et $p_1, \ldots, p_n$ be distinct propositions. If C is a tautology, then $C(p_1/A_1, \ldots, p_n/A_n)$ is a tautology.

Proof. If none of the p_i occurs in any A_j (nor in $C' =_{def} C(p_1/A_1, \ldots, p_n/A_n)$), it is easy. Let v, an arbitrary valuation for C', and w the *extension* of v such that $w(p_i) =_{def} v(A_i)$. As a consequence of the substitution lemma, $w(C') = w(C)$. From $w(C) = T$ and $w(C') = v(C')$ we deduce $v(C') = T$.

Comment. If p_i does occur in A_k, this technique would not work.
For instance from $\models p \equiv \neg\neg p$ one cannot deduce immediately $\models (p \vee r) \equiv \neg\neg(p \vee r)$ since valuation $v : v(p) = F, v(r) = T$, such that $v(A) = v(p \vee r) = T$ cannot be extended into w such that $w(p) = T$.
However, this is easily settled : from $\models p \equiv \neg\neg p$ we deduce $\models q \equiv \neg\neg q$, and then from that $\models (p \vee r) \equiv \neg\neg(p \vee r)$.

If some p_i occur in some of the $A_1, \ldots, A_n$, we consider fresh atoms q_i. If C is a tautology, then $C'' =_{def} C(p_1/q_1, \ldots, p_n/q_n)$ is a tautology. C' is $C''(q_1/A_1, \ldots, q_n/A_n)$, where no q_i occurs in any A_k, so C' is a tautology.

SEMANTIC TABLEAUX

Decision procedure for satisfiability (consistency) in propositional logic.

Faster than the truthtable method.

Principle : (in)consistency is investigated,
through a systematic search for models.

- **Truthtables :** from smaller subformulas to bigger subformulas :
  the truthvalue of a formula is *function* of the truthvalues of its (immediate) components.
- **Semantic tableaux :** from bigger subformulas to smaller subformulas : the truthvalue of an (immediate) component is *related* to the truthvalue of a formula.

Principle of the method

A *literal* is an atom or a negated atom. If p is an atomic proposition, $\{p, \neg p\}$ is a *complementary pair of literals*. A literal set (set of literals) is consistent if and only if it includes no complementary pair (which is determined by inspection).

The principle of the tableau method is to reduce the question

Is formula A consistent ?

to the easier question

Are all members of the (finite) set $\mathcal{A}$ consistent literal sets ?

A semantic tableau is a tree ; its root is formula A and its leaves are the elements of $\mathcal{A}$.

The method is nondeterministic, but all (correctly built) semantic tableaux based on root A will lead to the same conclusion.

How to build a tableau ?

It is convenient (and not really restrictive) to avoid double negations and two binary connectives : equivalence and exclusive disjunction. This allows to partition the set of formulas into three subsets :

— literals ;
— conjunctive formulas ;
— disjunctive formulas.

Formula $\neg(X \Rightarrow Y)$ is conjunctive since it is logically equivalent to the conjunction $X \wedge \neg Y$. Formula $X \Rightarrow Y$ is disjunctive since it is logically equivalent to the disjunction $\neg X \vee Y$. Besides, $\neg\neg X$ is rewritten into X.

Two kinds of expansion rules : α-rules give a node a single child ; β-rules give a node two children.

For semantic tableaux, α-rules are used to break conjunctive formulas and β-rules are used to break disjunctive formulas.

Examples I

Let $A = p \wedge (\neg q \vee \neg p)$.
It is an α-formula, that is, a formula to which an α-rule will apply, since it is a conjunctive formula ;
its components are p and $\neg q \vee \neg p$.
We can draw the semantic tableau :

$$\begin{array}{c} p \wedge (\neg q \vee \neg p) \\ \downarrow \\ p, \neg q \vee \neg p \end{array}$$

Its meaning is : "v is a model of A if and only if v is a model of p **and** of $\neg q \vee \neg p$".

Formula $\neg q \vee \neg p$ is a disjunctive, β-formula ;
its components are $\neg q$ and $\neg p$.
We can draw the semantic tableau further :

$$\begin{array}{ccc} & p, \neg q \vee \neg p & \\ \swarrow & & \searrow \\ p, \neg q & & p, \neg p \end{array}$$

Its meaning is : "v is a model of p and of $\neg q \vee \neg p$ if and only if v is a model of p and of $\neg q$ **or** of p and of $\neg p$".

Last, set $\{p, \neg q\}$ is consistent (symbol $\bigcirc$, *open leaf*) whereas $\{p, \neg p\}$ is inconsistent (symbol $\times$, *closed leaf*). The completed semantic tableau is :

$$\begin{array}{ccc} & p \wedge (\neg q \vee \neg p) & \\ & \downarrow & \\ & p, \neg q \vee \neg p & \\ \swarrow & & \searrow \\ p, \neg q & & p, \neg p \\ \bigcirc & & \times \end{array}$$

Examples II

Conclusion. The tableau is open (its root is consistent) if and only if some (at least one) of its leaves is open (consistent). Therefore A is consistent since $\{p, \neg q\}$ is.
A model v (for both the leaf and the root) is $v(p) = T$, $v(q) = F$.

Let $B = (p \vee q) \wedge (\neg p \wedge \neg q)$.
This is a conjunctive, α-formula ;
its (semantic) components are $B_1 =_{def} p \vee q$ and $B_2 =_{def} \neg p \wedge \neg q$.

Formula B_1 is a disjunctive, β-formula ; its components are p and q.

Formula B_2 is a conjunctive, α-formula ; its components are $\neg p$ and $\neg q$.

The tableau on the left is obtained if B_1 is broken before B_2, else we get the tableau on the right. Both tableaux induce the same conclusion : all leaves are closed, the tableaux are closed and the root B is inconsistent.

$$\begin{array}{ccc} & (p \vee q) \wedge (\neg p \wedge \neg q) & \\ & \downarrow & \\ & p \vee q, \neg p \wedge \neg q & \\ \swarrow & & \searrow \\ p, \neg p \wedge \neg q & & q, \neg p \wedge \neg q \\ \downarrow & & \downarrow \\ p, \neg p, \neg q & & q, \neg p, \neg q \\ \times & & \times \end{array}
\qquad
\begin{array}{ccc} & (p \vee q) \wedge (\neg p \wedge \neg q) & \\ & \downarrow & \\ & p \vee q, \neg p \wedge \neg q & \\ & \downarrow & \\ & p \vee q, \neg p, \neg q & \\ \swarrow & & \searrow \\ p, \neg p, \neg q & & q, \neg p, \neg q \\ \times & & \times \end{array}$$

Expansion rules, semantic components

— Conjunctive, α-*formulas* give rise to a single child ; $v(\alpha) = T$ if and only if $v(\alpha_1) = v(\alpha_2) = T$.

α	α_1	α_2
$A_1 \wedge A_2$	A_1	A_2
$\neg(A_1 \vee A_2)$	$\neg A_1$	$\neg A_2$
$\neg(A_1 \Rightarrow A_2)$	A_1	$\neg A_2$
$\neg(A_1 \Leftarrow A_2)$	$\neg A_1$	A_2

— Disjunctive, β-*formulas* give rise to two children ; $v(\beta) = T$ if and only if $v(\beta_1) = T$ or $v(\beta_2) = T$.

β	β_1	β_2
$B_1 \vee B_2$	B_1	B_2
$\neg(B_1 \wedge B_2)$	$\neg B_1$	$\neg B_2$
$B_1 \Rightarrow B_2$	$\neg B_1$	B_2
$B_1 \Leftarrow B_2$	B_1	$\neg B_2$

Semantic tableau : the algorithm for formula C.

Each node is labelled with a formula set.

Init : root is labelled $\{C\}$; it is an unmarked leaf.

Induction step : select an unmarked leaf ℓ labelled $U(\ell)$.

- If $U(\ell)$ is a literal set :
  - if $U(\ell)$ contains a complementary pair, then mark ℓ as *closed* '$\times$' ;
  - else mark ℓ as *open* '$\bigcirc$'.
- If $U(\ell)$ is not a literal set, select a non-literal formula in $U(\ell)$:
  - If it is an α-formula A, generate a child node ℓ' and label it with

$$U(\ell') = (U(\ell) - \{A\}) \cup \{\alpha_1, \alpha_2\};$$

  - if it is a β-formula B, generate two child nodes ℓ' and ℓ'' ; their labels respectively are

$$U(\ell') = (U(\ell) - \{B\}) \cup \{\beta_1\}$$

$$U(\ell'') = (U(\ell) - \{B\}) \cup \{\beta_2\}.$$

Termination : when all leaves are marked '$\times$' or '$\bigcirc$'.

Termination

Theorem. The expansion algorithm always terminates.

Proof. Let T be an A-tableau, maybe not fully expanded.

Let ℓ be a leaf of T.
Let $b(\ell)$, the number of binary connectives in $U(\ell)$
and $n(\ell)$, the number of negations in $U(\ell)$.
The size $W(\ell)$ of ℓ is defined as $2b(\ell) + n(\ell)$.
The size of any child node is always less than the size of its father.
(Check this for all α-rules and β-rules.)
As sizes are natural numbers, no infinite branch is possible.
A (fully-expanded) tableau is *closed* if all its leaves are closed.
It is *open* if at least one leaf is open.

Soundness and completeness of decision procedures (in logic)

Let S be a decision procedure for logical formulas.

Soundness : every S-valid formula is valid.

Completeness : every valid formula is S-valid.

What about our case : S is the method of semantic tableaux.
A formula is S-valid if any $\neg S$-tableau is closed.

Soundness :
If $T(A)$ is closed, then A is inconsistent ;
if $T(\neg B)$ is closed, then B is valid.

Completeness :
If A is inconsistent, then $T(A)$ is closed ;
If B is valid, then $T(\neg B)$ is closed.

Soundness : the proof I

If $T(A)$ is closed, then A is inconsistent.

Proof. $T(A)$ is closed ; so are all its subtableaux. We prove by induction on the height h of node n in $T(A)$ that $U(n)$ is inconsistent. A leave has height 0 ; an α-node height is one more than the height of its child ; a β-node height is one more than the height of its highest child.

— $h = 0$: n is a closed leaf
therefore $U(n)$ contains a complementarry pair of literals and $U(n)$ is inconsistent.

— $h > 0$: An α-rule or a β-rule has been used to expand node n.

α-**rule** : n : $\{\alpha\} \cup U_0$

$$\downarrow$$

n' : $\{\alpha_1, \alpha_2\} \cup U_0$

As $h(n') < h(n)$, induction hypothesis applies and $U(n')$ is inconsistent.
For each valuation v,
there is a formula $A' \in U(n')$ such that $v(A') = F$.

Soundness : the proof II

Three possibilites for A' :

1. $A' \in U_0 \subseteq U(n)$;
2. $A' = \alpha_1$: $v(\alpha_1) = F$ hence $v(\alpha) = F$ (cf. α-rules) ;
3. $A' = \alpha_2$: $v(\alpha_2) = F$ hence $v(\alpha) = F$.

In all case a v-false formula exists in $U(n)$ which is therefore inconsistent.

β-rule :

$$
\begin{array}{ccc}
 & n : \{\beta\} \cup U_0 & \\
 \swarrow & & \searrow \\
n' : \{\beta_1\} \cup U_0 & & n'' : \{\beta_2\} \cup U_0
\end{array}
$$

$h(n') < h(n)$ and $h(n'') < h(n)$;
sets $U(n')$ and $U(n'')$ are both inconsistent.

For each valuation v, either

1. there is a formula $A' \in U_0 \subseteq U(n)$: $v(A') = F$
2. $v(\beta_1) = v(\beta_2) = F$, hence $v(\beta) = F$ (cf. β-rules).

In both cases, there is a v-false formula in $U(n)$ hence $U(n)$ is inconsistent.

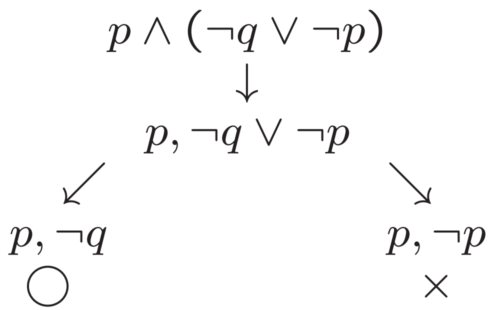
Completeness : the proof

If A is inconsistent, then $T(A)$ is closed.

Proof (contraposition). Assume $T(A)$ is open and check A is consistent.

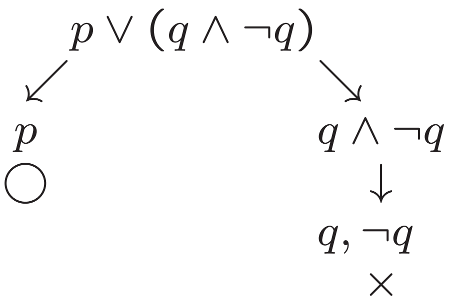
Key point : Every open leaf in $T(A)$ determines a model for A.

Example : $A \ =_{def} \ p \wedge (\neg q \vee \neg p)$



Interpretation $v(p) = T$, $v(q) = F$: model of A.

Example : $B = p \vee (q \wedge \neg q)$



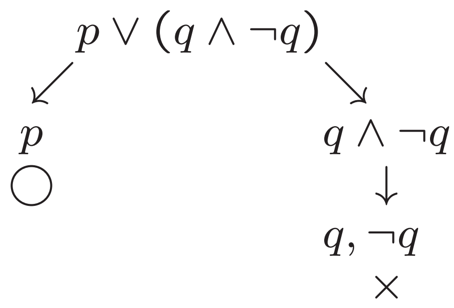
Interpretation $v(p) = T$: model of A ? Yes, with any $v(q)$.

Hintikka sets

Definition : Formula set U is a *Hintikka set* if three conditions are satisfied :

1. For each atom p, $p \notin U$ or $\neg p \notin U$
2. If $\alpha \in U$ is an α-formula, then $\alpha_1 \in U$ and $\alpha_2 \in U$.
3. If $\beta \in U$ is a β-formula, then $\beta_1 \in U$ or $\beta_2 \in U$.

Example :



$U = \{p, p \vee (q \wedge \neg q)\}$ is a Hintikka set.

To be proved :
The union set associated with an open branch is a Hintikka set.
Every Hintikka set is consistent.

Open branch lemma

Let *b* an open branch in fully expanded tableau T : $U =_{def} \bigcup_{n \in b} U(n)$ is a Hintikka set.

Proof. U satisfies the three Hintikka conditions :

1. Let ℓ the (open) leaf in *b*.
   For each literal m ($m \in \{p, \neg p\}$),
   $m \in U$ implies $m \in U(\ell)$ (no expansion rule for literals).
   Leaf ℓ is open, so no complementary pair.
   Therefore, $p \notin U$ or $\neg p \notin U$.
2. For each α-formula $\alpha \in U$, there is a node n whose child n' contains both α-components : $\alpha_1, \alpha_2 \in U(n') \subseteq U$.
3. For each β-formula $\beta \in U$, there is a node n whose children n' and n'' contain components β_1 and β_2, respectively. If, say, $n' \in b$, then $U(n') \subseteq U$ and $\beta_1 \in U$.

Hintikka's theorem

Every Hintikka set is consistent.

Proof. Let U a Hintikka set and $\mathcal{P} = \{p_1, \ldots, p_m\}$ the atom set used in U.

We define valuation v of U :

$$\begin{array}{lll} v(p) = T & \text{if} & p \in U \text{ or } \neg p \notin U \\ v(p) = F & \text{if} & \neg p \in U \end{array}$$

v assigns a single truthvalue to every atom of $\mathcal{P}$ (since U is a Hintikka set).

We have to prove that for each $A \in U$, $v(A) = T$.

by structural induction on A :

A *is a literal.*
- If $A = p$, then $v(A) = v(p) = T$.
- If $A = \neg p$, then $v(p) = F$, hence $v(A) = T$.

A *is an* α*-formula* α *:* $\alpha_1, \alpha_2 \in U$.
The induction hypothesis applies : $v(\alpha_1) = T$ and $v(\alpha_2) = T$,
hence $v(\alpha) = T$.

A *is a* β*-formula* β *:* $\beta_1 \in U$ or $\beta_2 \in U$.
The induction hypothesis applies : $v(\beta_1) = T$ or $v(\beta_2) = T$,
hence $v(\beta) = T$.

Completeness : the proof

Theorem.
If A is inconsistent, then $T(A)$ is closed.

Proof (contraposition). If $T(A)$ is open, then A is consistent.

If $T(A)$ is open, there is an open branch b in $T(A)$. The set of all formulas in b is a Hintikka set, therefore a consistent set ; any model of this set is a model of A.

Summary

Formula A is inconsistent if and only if $T(A)$ is closed.
Formula B is valid if and only if $T(\neg B)$ is closed.
Formula C is simply consistent (contingent) if and only if both $T(C)$ and $T(\neg C)$ are open.

The tableau method is a decision algorithm for validity, consistency, contingency, inconsistency.

A practical approach

If we suspect inconsistency for formula X,
$T(X)$ will be considered first ;
if we suspect validity, $T(\neg X)$ will be considered first.

If $T(Y)$ is closed, analysis is over : Y is known to be inconsistent and $\neg Y$ is valid.

If $T(Z)$ is open, Z is known to be consistent and $\neg Z$ is not valid ; $T(\neg Z)$ has to be considered to obtain more information, and to determine whether Z is valid or not.

Simplification : a branch can be closed as soon as a complementary pair $\{A, \neg A\}$ occurs, even if A is not an atom.

Heuristics : use α-rules first.

Interpolation and definability I

Two real functions f and g, domain $\mathbf{R}^2$; a subset $D \subset \mathbf{R}^3$.

If $\forall(x, y, z) \in D : f(x, y) \leq g(x, z)$, then an *interpolant* function h satisfies

$$\forall(x, y, z) \in D : [f(x, y) \leq h(x) \leq g(x, z)] \, .$$

Does h exists ? It depends on the chosen domain D.

Example. $D = D_1 \times D_2 \times D_3$, two possible, "extreme" interpolants are

$$x \mapsto \sup_{y \in D_2} f(x, y) \quad \text{and} \quad x \mapsto \inf_{z \in D_3} g(x, z)$$

Counter-example. $D = \{(0, 0, 0), (0, 1, 1)\}$, the hypothesis is

$$f(0, 0) \leq g(0, 0) \ \wedge \ f(0, 1) \leq g(0, 1)$$

and the thesis is

$$f(0, 0) \leq h(0) \leq g(0, 0) \ \wedge \ f(0, 1) \leq h(0) \leq g(0, 1) \, .$$

Obviously, the thesis is not a logical consequence of the hypothesis.
Counterexample : $f(0, 0) = 0$, $g(0, 0) = 1$, $f(0, 1) = 2$, $g(0, 1) = 3$.

Interpolation and definability II

In $\mathbf{R}^+$, equation $x^2 = x + 1$ has a single solution, the *golden ratio*;
if $y^2 = y + 1$ and $z^2 = z + 1$, then $y = z$;
the equation is an *implicit definition* of the golden ratio x.

Can this implicit definition turned into an explicit definition?
Yes : $x = (1 + \sqrt{5})/2$.

Definability corresponds to equation solving.
Interpolation corresponds to inequation solving.
In algebra and calculus,
(in)equation solving can be an intricate business.

In propositional logic, these notions are easy. Predicates "$a \leq b$" and "$a = b$" respectively become "$\models a \Rightarrow b$" and "$\models a \equiv b$".

Craig's interpolation theorem

If $\models A \Rightarrow B$, then a formula C exists, containing only atoms occurring in both A and B, such that $\models A \Rightarrow C$ and $\models C \Rightarrow B$.

Proof. Induction on the set Π of atoms occurring in both A and B.

Base case. If $\Pi = \emptyset$, $\models A \Rightarrow B$ implies either A is inconsistent (and $C =_{def} false$ is an appropriate choice) or B is valid (and $C =_{def} true$ is an appropriate choice). This can be proved by contradiction. If there were valuations u and v (defined on disjoint domains) such that $u(A) = T$ et $v(B) = F$, the valuation $w =_{def} u \cup v$ would be such that $w(A \Rightarrow B) = F$.

Induction step. If $p \in \Pi$, induction hypothesis applies to formulas $A(p/true), B(p/true)$ and also to formulas $A(p/false), B(p/false)$ (why is that ?). If C_T and C_F are corresponding interpolants, then formula $(p \wedge C_T) \vee (\neg p \wedge C_F)$ is an interpolant for A and B (again, why is that ?).

Beth's definability theorem

Let A be a formula with no occurrence of q and r,
such that formula $[A(p/q) \wedge A(p/r)] \Rightarrow (q \equiv r)$ is a tautology.

Some formula B exists, with no occurrence of p, q and r,
such that formula $A \Rightarrow (p \equiv B)$ is a tautology.

Hypothesis states that as soon as a valuation v assigns T to formula A,
it also assigns some fixed truthvalue $v(p)$ to proposition p.

Thesis states that the value $v(p)$ can be explicited into $v(B)$,
for some B without occurrence of the "unknown", proposition p.

Proof. Sequence of elementary steps.

$\models [A(p/q) \wedge A(p/r)] \Rightarrow (q \equiv r)]$,

$\models [A(p/q) \wedge A(p/r) \wedge q] \Rightarrow r$,

$\models [A(p/q) \wedge q] \Rightarrow [A(p/r) \Rightarrow r]$.

Let B be an interpolant (Craig's theorem), without p, q, r :

$\models [A(p/q) \wedge q] \Rightarrow B$, and therefore

$\models A(p/q) \Rightarrow (q \Rightarrow B)$, and by uniform substitution

$\models A(p/r) \Rightarrow (r \Rightarrow B)$.

On the other hand :

$\models B \Rightarrow [A(p/r) \Rightarrow r]$, hence

$\models [B \wedge A(p/r)] \Rightarrow r$, hence

$\models A(p/r) \Rightarrow (B \Rightarrow r)$, and by uniform substitution

$\models A(p/q) \Rightarrow (B \Rightarrow q)$.

The thesis follows :

$\models A(p/q) \Rightarrow (q \equiv B)$, or

$\models A(p/r) \Rightarrow (r \equiv B)$, or

$\models A \Rightarrow (p \equiv B)$.

Finitely consistent sets

Definition. A set is *finitely consistent* if all its finite subsets are consistent.

Comment. Every consistent set is finitely consistent.

Definition. A finitely consistent set S is *maximal* if no proper superset of S is finitely consistent.

Theorem. Let Π a set of atoms and $\mathcal{F}$ the set of formulas based on Π. A set $E \subset \mathcal{F}$ is maximal finitely consistent if and only if a valuation v on Π exists such that $E = \{\varphi \in \mathcal{F} : v(\varphi) = T\}$.

Corollary. Every maximal finitely consistent set is a consistent set with a unique model.

The unique model of a maximal finitely consistent set

Comment. The proof emphasises the one-to-one correspondence between valuations and maximal finitely consistent sets (when the atom set is fixed).

Proof. The condition is sufficient. The set $E = \{\varphi \in \mathcal{F} : v(\varphi) = T\}$ is (finitely) consistent, since v is a model of S, and is maximal : if $\psi \notin E$, then the set $E \cup \{\psi\}$ includes the finite inconsistent subset $\{\neg\psi, \psi\}$.

The condition is necessary. (We consider only the case where Π is a countable set, say Π $= \{p_1, p_2, \ldots\}$.)
Let E be a maximal finitely consistent subset of $\mathcal{F}$.
For each i, E contains either p_i or $\neg p_i$. It cannot contain both and, if $p_i \notin E$, the set $E \cup \{p_i\}$ is not finitely consistent so there is a finite subset E' of E such that $E' \cup \{p_i\}$ is inconsistent and therefore $E' \models \{\neg p_i\}$. This means $E \cup \{\neg p_i\}$ is finitely consistent [if $E'' \subset E$, every model of $E'' \cup E'$ is a model of $E'' \cup \{\neg p_i\}$] so, $\neg p_i \in E$ since E is maximal. (In the same way, if $\neg p_i \notin E$, then $p_i \in E$.) For each i, let ℓ_i the unique element of $\{p_i, \neg p_i\}$ belonging to E ; let v the unique interpretation such that all ℓ_i are true. Let $\varphi \in E$ and $\{p_{i_1}, \ldots, p_{i_n}\}$ the propositions occurring in φ. Since E is finitely consistent, the finite subset $\{\ell_{i_1}, \ldots, \ell_{i_n}, \varphi\}$ is consistent, hence $v(\varphi) = T$ and $E \subset \{\varphi \in \mathcal{F} : v(\varphi) = T\}$. Since E is maximal the inclusion is an equality.

Compactness theorem I

Every finitely consistent set is a consistent set.

Comment. It is sufficient to prove that every finitely consistent set is included in a maximal finitely consistent set.

Proof. Let D be a finitely consistent set. A chain of supersets is designed as follows : $E_0 = D$ and if $n > 0$, $E_n = E_{n-1} \cup \{p_n\}$ if this set is finitely consistent and $E_n = E_{n-1} \cup \{\neg p_n\}$ otherwise.
All these sets are finitely consistent. The base case E_0 is obvious. The induction step is also obvious when $E_n = E_{n-1} \cup \{p_n\}$; let us therefore consider the case $E_n = E_{n-1} \cup \{\neg p_n\}$. In this case a finite subset $E' \subset E_{n-1}$ exists such that $E' \cup \{p_n\}$ is inconsistent and therefore $E' \models \neg p_n$. As a result $E_n = E_{n-1} \cup \{\neg p_n\}$ is finitely consistent since for each finite subset $E'' \subset E_{n-1}$, every model of $E'' \cup E'$ (such a model exists) is also a model of $E'' \cup \{\neg p_n\}$.

Compactness theorem II

Now, let $E =_{def} \bigcup_n E_n$. The intersection $\{p_i, \neg p_i\} \cap E$ contains a single element ℓ_i.
The $\ell_i : l = 1, 2, \ldots$ define a single valuation v. For each $\varphi \in D$, the value $v(\varphi)$ must be T, otherwise $\{\ell_i :$ p_i occurs in $\varphi\} \cup \{\varphi\}$, would be a finite inconsistent subset of some E_n, which is impossible.
The set D is therefore included in the maximal finitely consistent set $\{\varphi \in \mathcal{F} : v(\varphi) = T\}$.

Mathematical comments.
The compactness theorem remains true for uncountable sets (proof by ordinal induction, AC is used).
The compactness theorem is a special case of Tychonoff's theorem : the product of any set of compact (topological) spaces is a compact space.

DEDUCTIVE METHODS

How to develop a theory?

1. Decision method (for validity);
this is the analytical approach :

If $U = \{A_1, \ldots, A_n\}$ and $\mathcal{T}(U) = \{A : U \models A\}$,
then $A \in \mathcal{T}(U)$ iff $\models (A_1 \wedge \ldots \wedge A_n) \Rightarrow A$.

Problems :

— There can be infinitely many axioms.

— Few logics allow for a validity decision method.

— The decision method provides no information about the "causal link" from axioms to theorems.

— Old theorems are of no use to obtain new ones.

2. Proof method ;
this is the deductive, synthetic approach :

A theorem results from a deductive, syntactic process, which starts from axioms.

Advantages :

— Even if infinitely many axioms are available, a proof uses only a finite subset of axioms.

— The proof emphasises the “causal link” between axioms and the proved theorem.

— As soon as it is obtained, any theorem can be used as an additional axiom, in order to deduce more theorems.

Problem : The deductive approach is a highly nondeterministic process.

HILBERT SYSTEM

The formal system $\mathcal{H}$ consists of

— *three axiom schemes* :

1. $\vdash A \Rightarrow (B \Rightarrow A)$
2. $\vdash (A \Rightarrow (B \Rightarrow C))$
$$\Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$$
3. $\vdash (\neg B \Rightarrow \neg A) \Rightarrow (A \Rightarrow B)$

— the *Modus Ponens (MP)* rule :

$$\frac{\vdash A \qquad \vdash A \Rightarrow B}{\vdash B}$$

A, B and C are any formulas,
involving only "$\neg$" and "$\Rightarrow$" connectives.

Proofs

A *proof* in $\mathcal{H}$ is a formula sequence ; each formula is

— an axiom (a scheme instance), or

— inferred from two earlier formulas (occurring earlier in the sequence) by the Modus Ponens rule.

The last member A of the sequence is a *theorem* ; the sequence is a *proof* of A.

This is written $\vdash_{\mathcal{H}} A$ *or* $\vdash A$.

Comment. Any proof prefix is also a proof (of its last element).

Example : a proof in $\mathcal{H}$

1. $\vdash p \Rightarrow ((p \Rightarrow p) \Rightarrow p)$ (Axiom 1)
2. $\vdash (p \Rightarrow ((p \Rightarrow p) \Rightarrow p)) \Rightarrow ((p \Rightarrow (p \Rightarrow p)) \Rightarrow (p \Rightarrow p))$ (Axiom 2)
3. $\vdash (p \Rightarrow (p \Rightarrow p)) \Rightarrow (p \Rightarrow p)$ (1, 2, MP)
4. $\vdash p \Rightarrow (p \Rightarrow p)$ (Axiom 1)
5. $\vdash p \Rightarrow p$ (4, 3, MP)

Comments. "Axiom 1" means "instance of axiom scheme 1" and "4, 3, MP" means "results from formulas 4 and 3 by Modus Ponens rule".

This example 5-line proof witnesses that $(p \Rightarrow p)$ is a theorem, or that assertion $\vdash (p \Rightarrow p)$ (read : "$(p \Rightarrow p)$ is a theorem") is a *metatheorem* (i.e., a usual theorem, in the mathematical sense ; the "meta" is often omitted).

$(A \Rightarrow A)$ is a *theorem scheme* ; any instance is a theorem.

A proof of $(p \Rightarrow p)$ is easily converted into a proof of, say, $(p \Rightarrow q) \Rightarrow (p \Rightarrow q)$.

Only the negation and the conditional are used in this (economical) version of Hilbert system ; this is a nonessential but convenient restriction.

Proofs as trees

$$\dfrac{\dfrac{\vdash p \Rightarrow ((p \Rightarrow p) \Rightarrow p) \qquad \begin{array}{l}\vdash (p \Rightarrow ((p \Rightarrow p) \Rightarrow p)) \Rightarrow \\ \quad ((p \Rightarrow (p \Rightarrow p)) \Rightarrow (p \Rightarrow p))\end{array}}{\vdash (p \Rightarrow (p \Rightarrow p)) \Rightarrow (p \Rightarrow p)} \qquad \vdash p \Rightarrow (p \Rightarrow p)}{\vdash p \Rightarrow p}$$

Proofs really are trees but the sequence representation is more convenient . . . at least from the typographic point ov view.

$$\dfrac{\dfrac{1 \qquad 2}{3} \qquad 4}{5}$$

Derivations

A U-based *derivation* in $\mathcal{H}$ is a formula sequence where every formula is

— a *hypothesis* (element of U), or

— an axiom, our

— inferred from two earlier formula by Modus Ponens.

If A is the last member of the sequence, the corresponding metatheorem is written $U \vdash_{\mathcal{H}} A$ or $U \vdash A$.

A proof is an $\emptyset$-based derivation.

Remarque. The last member A of a derivation is (usually) not a theorem. We will prove later that $U \vdash A$ holds if and only if $U \models A$ holds; as a result, $\vdash A$ holds if and only if A is a tautology.

Derivation : an example

1. $p \Rightarrow (q \Rightarrow r),\, q,\, p \vdash\; p \Rightarrow (q \Rightarrow r)$ (Hypothesis)
2. $p \Rightarrow (q \Rightarrow r),\, q,\, p \vdash\; p$ (Hypothesis)
3. $p \Rightarrow (q \Rightarrow r),\, q,\, p \vdash\; q \Rightarrow r$ (1, 2, MP)
4. $p \Rightarrow (q \Rightarrow r),\, q,\, p \vdash\; q$ (Hypothesis)
5. $p \Rightarrow (q \Rightarrow r),\, q,\, p \vdash\; r$ (3, 4, MP)

Comments.

Proving $A, B, C \vdash D$ is usually easier than proving $\vdash A \Rightarrow (B \Rightarrow (C \Rightarrow D))$, but we will show that any derivation for the first assertion can be mechanically converted into a derivation for the second assertion ; the derivation above therefore witnesses the assertion

$$\vdash\; (p \Rightarrow (q \Rightarrow r)) \Rightarrow (q \Rightarrow (p \Rightarrow r))\,.$$

Composition principle

Any theorem can be used as an additional axiom.

Proof. A proof using this can be converted into a "real" proof by replacing every theorem by a proof of this theorem.

Uniform substitution principle

If C is a theorem and if $p_1, \ldots, p_n$ are pairwise distinct propositions, then $C(p_1/A_1, \ldots, p_n/A_n)$ is a theorem.

Proof. The application of any uniform substitution to a proof always produces a proof.

Derived inference rules

The notation $\dfrac{U_1 \vdash A_1\,, \cdots, U_n \vdash A_n}{U \vdash B}$

is a *derived inference rule*. Its meaning is, if the assertions above the line can be proved, then so can the assertion below the line. A derived inference rule is *sound* or *correct* if that is really the case.

Comment. A derived inference rule is sound if any derivation using it can be converted into a proper derivation. If we show first that $U \vdash A$ holds iff $U \models A$ holds, proving the soundness of a derived inference rule will be easy. For instance, the derived inference rule $\dfrac{\neg X \vdash X}{\vdash X}$ is sound, since, if X is a logical consequence of $\neg X$, then X is valid.

However, the soundness of some derived rules will be established directly since they will make easier to investigate the link between $\vdash$ and $\models$.

Deduction rule

This is the rule

$$\frac{U, A \vdash B}{U \vdash A \Rightarrow B}$$

Comments. "U, A" is short for "$U \cup \{A\}$".
This rule is very often used in mathematics :

- In order to prove $A \Rightarrow B$;
- we assume A ;
- from which we deduce B.

If this rule can be proved sound, we already know it will be useful :

$p \Rightarrow (q \Rightarrow r), q, p \vdash r$ is trivial ;

$\vdash (p \Rightarrow (q \Rightarrow r)) \Rightarrow (q \Rightarrow (p \Rightarrow r))$ is not.

Soundness of the deduction rule : a direct proof

We have to convert step by step a derivation for $U, A \vdash B$ into a derivation for $U \vdash A \Rightarrow B$.

Proof. Let Π_1 a derivation for $U, A \vdash B$; we designed a derivation Π_2 for $U \vdash (A \Rightarrow B)$ by replacing each line of Π_1 by a sequence of lines for Π_2. More specifically, nth line X in Π_1 establishes

$n.\quad U, A \vdash X$

and is converted into a sequence of lines ending in establishing

$n'.\quad U \vdash A \Rightarrow X$.

Four cases are possible :

1. X is *an axiom* ;
2. X is *a hypothesis* $(X \in U)$;
3. X is *the new hypothesis* A ;
4. X is *inferred by Modus Ponens*.

In cases 1 and 2, we convert

$n.\quad U, A \vdash X$ (Ai or H)

into a three-line sequence

$n'{-}2.\quad U \vdash X$ (Ai or H)
$n'{-}1.\quad U \vdash X \Rightarrow (A \Rightarrow X)$ (A1)
$n'.\quad U \vdash A \Rightarrow X$ ($n'{-}2$, $n'{-}1$, MP)

In case 3, we convert $U, A \vdash A$ by a five-line sequence adapted fron the proof of $\vdash (p \Rightarrow p)$; the last line is therefore $U \vdash (A \Rightarrow A)$.

In case 4, we know the derivation Π_1 contains

$i.\quad U, A \vdash Y$ (. . .)
$j.\quad U, A \vdash Y \Rightarrow X$ (. . .)
$n.\quad U, A \vdash X$ (i, j, MP)

We can assume Π_2 already contains

$i'.\quad U \vdash A \Rightarrow Y$ (. . .)
$j'.\quad U \vdash A \Rightarrow (Y \Rightarrow X)$ (. . .)

and we add

$n'{-}2.\quad U \vdash (A \Rightarrow (Y \Rightarrow X)) \Rightarrow ((A \Rightarrow Y) \Rightarrow (A \Rightarrow X))$ (A2)
$n'{-}1.\quad U \vdash (A \Rightarrow Y) \Rightarrow (A \Rightarrow X)$ (j', $n'{-}2$, MP)
$n'.\quad U \vdash (A \Rightarrow X)$ (i', $n'{-}1$, MP)

Some useful theorems

Hilbert-like proofs are easy to check but sometimes their design are tricky ; this is usual with synthetic methods.

Here are some useful theorems, given without proof.

1. $\vdash (A \Rightarrow B) \Rightarrow ((B \Rightarrow C) \Rightarrow (A \Rightarrow C))$
2. $\vdash (A \Rightarrow (B \Rightarrow C)) \Rightarrow (B \Rightarrow (A \Rightarrow C))$
3. $\vdash \neg A \Rightarrow (A \Rightarrow B)$
4. $\vdash A \Rightarrow (\neg A \Rightarrow B)$
5. $\vdash \neg\neg A \Rightarrow A$
6. $\vdash A \Rightarrow \neg\neg A$
7. $\vdash (A \Rightarrow B) \Rightarrow (\neg B \Rightarrow \neg A)$
8. $\vdash \neg C \Rightarrow (B \Rightarrow \neg(B \Rightarrow C))$
9. $\vdash (B \Rightarrow A) \Rightarrow ((\neg B \Rightarrow A) \Rightarrow A)$

Proof design

The deduction rule and the composition principle are used to prove theorem 6. Even with them, the designis not easy.

1. $A, \neg\neg\neg A \vdash \neg\neg\neg A \Rightarrow \neg A$ (Composition, th. 5)
2. $A, \neg\neg\neg A \vdash \neg\neg\neg A$ (Hypothesis)
3. $A, \neg\neg\neg A \vdash \neg A$ (1, 2, MP)
4. $A \vdash \neg\neg\neg A \Rightarrow \neg A$ (Deduction, 3)
5. $A \vdash (\neg\neg\neg A \Rightarrow \neg A) \Rightarrow (A \Rightarrow \neg\neg A)$ (Axiom 3)
6. $A \vdash A \Rightarrow \neg\neg A$ (4, 5, MP)
7. $A \vdash A$ (Hypothesis)
8. $A \vdash \neg\neg A$ (6, 7, MP)
9. $\vdash A \Rightarrow \neg\neg A$ (Deduction, 8)

Comment. The *augmentation rule* is obviously sound :

$$\frac{U \vdash A}{U, B \vdash A}$$

Further useful (sound) derived inference rules

If $\vdash A \Rightarrow B$ is known then rule $\dfrac{U \vdash A}{U \vdash B}$ is sound.

This formalizes some kind of common-sense reasoning :

1. If A follows from U, i.e. $U \vdash A$,
2. use theorem $\vdash A \Rightarrow B$,
3. apply Modus Ponens to (1) and (2) and obtain $U \vdash B$.

$$\frac{U \vdash \neg B \Rightarrow \neg A}{U \vdash A \Rightarrow B}$$ *Contraposition rule*

$$\frac{U \vdash A \Rightarrow B \quad U \vdash B \Rightarrow C}{U \vdash A \Rightarrow C}$$ *Transitivity rule*

$$\frac{U \vdash \neg\neg A}{U \vdash A}$$ *Double negation rule*

$$\frac{U \vdash A \Rightarrow (B \Rightarrow C)}{U \vdash B \Rightarrow (A \Rightarrow C)}$$ *Antecedent switching rule*

Interpretation of rules I

Contraposition rule formalizes contradiction reasoning.

Transitivity formalizes chain reasoning :
to prove $\vdash A \Rightarrow B$,
lemmas are proved :
$\vdash A \Rightarrow C_1\,, \vdash C_1 \Rightarrow C_2\,, \ldots, \vdash C_n \Rightarrow B\,.$
Repeated use of transitivity rule leads to $\vdash A \Rightarrow B$.

Antecedent switching rule means that the set of hypotheses is not ordered ; they can be used in any order.

Interpretation of rules II

Double negation rule is often used in mathematics . . .

. . . but can be misleading outside mathematics.

Examples : The sentence

It is not true that I am unhappy

is not fully equivalent to

I am happy

A program which does not produce two values $x \neq y$ is not necessarily a program which produces two equal values $x = y$.

Case disjunction

This is a very useful derived rule.

$$\frac{U, B \vdash A \qquad U, \neg B \vdash A}{U \vdash A}$$

Proof outline :

$U, B \vdash A$	Hypothesis
$U \vdash B \Rightarrow A$	Deduction
$\vdash (B \Rightarrow A) \Rightarrow ((\neg B \Rightarrow A) \Rightarrow A)$	Theorem
$U \vdash (\neg B \Rightarrow A) \Rightarrow A$	MP
$U, \neg B \vdash A$	Hypothesis
$U \vdash \neg B \Rightarrow A$	Deduction
$U \vdash A$	MP

Kalmar's lemma

Let A be a formula based on propositions $p_1, \ldots, p_n$ and connectives "$\neg$" and "$\Rightarrow$" ; let v be some valuation.
If p'_k is defined as p_k if $v(p_k) = T$ and as $\neg p_k$ if $v(p_k) = F$,
and if A' is defined as A if $v(A) = T$ and as $\neg A$ if $v(A) = F$,
then

$$\{p'_1, \ldots, p'_n\} \vdash A'$$

Example. From the truthtable line

p	q	r	s	$(p \Rightarrow q) \Rightarrow \neg(\neg r \Rightarrow s)$
F	F	T	T	F

Kalmar's lemma allows us to deduce

$$\{\neg p, \neg q, r, s\} \ \vdash \ \neg[(p \Rightarrow q) \Rightarrow \neg(\neg r \Rightarrow s)]\,.$$

Kalmar's lemma : the proof

Comment. Kalmar's lemma allows us to "encode" a truthtable line into the Hilbert system ; this will be used to show that $U \models A$ implies $U \vdash A$.

Mathematical induction is used ;
the induction is based on the syntactic structure of formula A.

Base case : A is p_k.

First inductive step : A is $\neg B$.

Second inductive step : A is $B \Rightarrow C$.

Base case

If $v(p_k) = T$, then Kalmar's thesis reduces to $\{\ldots, p_k, \ldots\} \vdash p_k$.

If $v(p_k) = F$, it reduces to $\{\ldots, \neg p_k, \ldots\} \vdash \neg p_k$.

First inductive step : A is $\neg B$.

If $v(B) = F$ and $v(A) = T$ then

$\{p_1', \ldots, p_n'\} \vdash B'$,
$\{p_1', \ldots, p_n'\} \vdash \neg B$,
$\{p_1', \ldots, p_n'\} \vdash A$,
$\{p_1', \ldots, p_n'\} \vdash A'$.

If $v(B) = T$ and $v(A) = F$ then

$\{p_1', \ldots, p_n'\} \vdash B'$,
$\{p_1', \ldots, p_n'\} \vdash B$,
$B \vdash \neg\neg B$,
$\{p_1', \ldots, p_n'\} \vdash \neg\neg B$,
$\{p_1', \ldots, p_n'\} \vdash \neg A$,
$\{p_1', \ldots, p_n'\} \vdash A'$.

Second inductive step : A is $B \Rightarrow C$.

If $v(C) = T$ and $v(A) = T$ then

$\{p'_1, \ldots, p'_n\} \vdash C'\,,$
$\{p'_1, \ldots, p'_n\} \vdash C\,,$
$C \vdash (B \Rightarrow C)\,,$
$\{p'_1, \ldots, p'_n\} \vdash (B \Rightarrow C)\,,$
$\{p'_1, \ldots, p'_n\} \vdash A\,,$
$\{p'_1, \ldots, p'_n\} \vdash A'\,.$

If $v(B) = F$ and $v(A) = T$ then

$\{p'_1, \ldots, p'_n\} \vdash B'\,,$
$\{p'_1, \ldots, p'_n\} \vdash \neg B\,,$
$\neg B \vdash (B \Rightarrow C)\,,$
$\{p'_1, \ldots, p'_n\} \vdash (B \Rightarrow C)\,,$
$\{p'_1, \ldots, p'_n\} \vdash A\,,$
$\{p'_1, \ldots, p'_n\} \vdash A'\,.$

Second inductive step (bis)

If $v(B) = T$, $v(C) = F$ and $v(A) = F$ on a

$\{p'_1, \ldots, p'_n\} \vdash B'$,
$\{p'_1, \ldots, p'_n\} \vdash B$,
$\{p'_1, \ldots, p'_n\} \vdash C'$,
$\{p'_1, \ldots, p'_n\} \vdash \neg C$,
$\neg C, B \vdash \neg(B \Rightarrow C)$,
$\{p'_1, \ldots, p'_n\} \vdash \neg(B \Rightarrow C)$,
$\{p'_1, \ldots, p'_n\} \vdash \neg A$,
$\{p'_1, \ldots, p'_n\} \vdash A'$.

These lemmas have been used :

$B \vdash \neg\neg B$, $\quad C \vdash B \Rightarrow C$,
$\neg B \vdash B \Rightarrow C$, $\quad \neg C, B \vdash \neg(B \Rightarrow C)$.

Completeness of Hilbert system

Let A be a tautology based on propositions $p_1, \ldots, p_n$ and connectives "$\neg$" and "$\Rightarrow$" only. Kalmar's lemma leads to

$$\{p'_1, \ldots, p'_n\} \vdash A\,,$$

where p'_k can be either p_k or $\neg p_k$ (but $A' = A$.)
From these 2^n theorems, we obtain (by Case disjunction rule) 2^{n-1} new theorems :

$$\{p'_1, \ldots, p'_{n-1}\} \vdash A\,,$$

and, more generally, 2^k theorems, for $k = 0, 1, \ldots, n$:

$$\{p'_1, \ldots, p'_k\} \vdash A\,,$$

The special case $k = 0$ gives the intended conclusion :

$$\vdash A\,.$$

RESOLUTION

Most useful proof method in implementations.

Proof method by refutation :
as with semantic tableaux, instead of proving A is valid,
we prove $\neg A$ is inconsistent ;
instead of proving $E \models A$
we prove $E \cup \{\neg A\}$ is inconsistent.

Classical resolution requires formulas in clausal form,
or conjunctive normal form.

Normal forms

The expression $(x^2 - 4x)(x + 3) + (2x - 1)^2 + 4x - 19$
is a polynomial, but its properties are not obvious. A more convenient form for the same polynomial will emphasise its degree, its roots, ...
Normal forms (or canonical forms) are used for that purpose. The most used forms are :

$x^3 + 3x^2 - 12x - 18$ (sum of monomials, decreasing degrees) ;

$(x - 3)(x + 3 - \sqrt{3})(x + 3 + \sqrt{3})$ (product of linear factors) ;

$[(x + 3)x - 12]x - 18$ (Horner form).

Disjunctive normal form I

p	q	r	$p \Rightarrow q$	$(p \Rightarrow q) \Rightarrow r$
T	T	T	T	T
T	T	F	T	F
T	F	T	F	T
T	F	F	F	T
F	T	T	T	T
F	T	F	T	F
F	F	T	T	T
F	F	F	T	F

$$\begin{array}{ll} & (\ p \wedge \ \ q \wedge \ \ r) \\ \vee & (\ p \wedge \neg q \wedge \ \ r) \\ \vee & (\ p \wedge \neg q \wedge \neg r) \\ \vee & (\neg p \wedge \ \ q \wedge \ \ r) \\ \vee & (\neg p \wedge \neg q \wedge \ \ r) \end{array}$$

The truthtable of $(p \Rightarrow q) \Rightarrow r$ (left), demonstrates that this formula is logically equivalent to the disjunctive formula (right). Each disjunct corresponds to a "true" line of the table.

A *disjunctive normal form* is a disjunction of *cubes*, which are conjunctions of literals. Every formula has a truthtable and is therefore logically equivalent to a disjunctive normal form (DNF).

Comment. A DNF can contain any (finite) number of cubes ; a cube can contain any (finite) number of literals.

Disjunctive normal form II

The cube $(\ell_1 \wedge \ell_2 \wedge \cdots \wedge \ell_n)\,, \quad (n \in \mathbf{N})$, is sometimes written $\bigwedge\{\ell_1, \ldots, \ell_n\}$, or $\bigwedge_i \ell_i$, or simply $\{\ell_1, \ldots, \ell_n\}$.

Comment. true et *false* are not literals, but they are cubes.

A cube is inconsistent if and only if it contains a pair of opposite literals, a complementary pair.

A cube is valid if and only if it is empty.

A DNF is inconsistent if and only if all its cubes are inconsistent.
The empty DNF is therefore inconsistent.

Conjunctive normal form I

A *clause* is a disjunction of literals.

A clause can be represented as $\bigvee\{\ell_i : i = 1, \ldots, n\}$, and even as $\{\ell_i : i = 1, \ldots, n\}$, although the latter is ambiguous and should be avoided.

Comment. Sometimes, a notation like $p\overline{q}r$ is used to denote the cube $p \wedge \neg q \wedge r$ or the clause $p \vee \neg q \vee r$. This is ambiguous and should be avoided.

The only inconsistent clause is the *empty clause*, denoted $\Box$.

A clause is valid if and only if it contains a pair of opposite literals, a complementary pair.

A *unit clause* contains a single literal.

Conjunctive normal form II

A *conjunctive normal form* or CNF is a conjunction of clauses.

Examples :

— $(\neg p \vee q \vee r) \wedge (\neg q \vee r) \wedge (\neg r)$ – CNF

— $(\neg p \vee q \vee r) \wedge \neg(\neg q \vee r) \wedge (\neg r)$ – not CNF

A CNF is valid if and only if all its clauses are valid ; as a consequence, the empty CNF is valid.

Every formula is logically equivalent to some CNF.

Comment. Clauses, cubes, DNF and CNF are formulas and therefore contain finitely many terms.

Why normal forms ?

A useful normal form must be

— general enough : any formula should have a logically equivalent normal form ;

— as specific as possible, so specific algorithms can be designed to deal with normal forms, more efficient than the general algorithms.

Normal forms could be unique, but that is not true for DNF and CNF.

Example. The DNF
$(p \wedge q \wedge r) \vee (p \wedge \neg q \wedge r) \vee (p \wedge \neg q \wedge \neg r) \vee$
$(\neg p \wedge q \wedge r) \vee (\neg p \wedge \neg q \wedge \neg r)$
is logically equivalent to a shorter DNF :
$(p \wedge r) \vee (\neg q \wedge \neg r) \vee (\neg p \wedge q \wedge r)$

Normalization algorithm I

From now on, only CNF is considered.

1. Eliminate all connectives but $\neg$, $\vee$, $\wedge$.
2. Use De Morgan laws for propagating $\neg$ occurrences down the syntactic tree.

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

3. Eliminate double negations.

$$\neg\neg A \leftrightarrow A$$

4. Use distributivity laws to propagate $\vee$ downward.

$$A \vee (B \wedge C) \leftrightarrow (A \vee B) \wedge (A \vee C)$$
$$(A \wedge B) \vee C \leftrightarrow (A \vee C) \wedge (B \vee C)$$

A CNF can be viewed as a set of clauses, a *clausal form*.

Exercise. Observe the link beteen a CNF logically equivalent to A and a DNF logically equivalent to $\neg A$.

Normalization algorithm II

Variable L is a (conjunctive) set of disjunctions ; its initial value is $\{A\}$ where A is any formula (viewed as a disjunction of one term). The final value of L is a CNF, logically equivalent to A. We call *disclause* any disjunction containing at least one term which is not a literal.

$L := \{A\}$;
As long as L contains some disclause do
    $\{ \bigwedge L \leftrightarrow A$ is invariant $\}$
    select a disclause $D \in L$;
    select a non-literal $t \in D$;
    if $t = \alpha$ do
        $t_1 := \alpha_1$; $t_2 := \alpha_2$;
        $D_1 := (D - t) + t_1$; $D_2 := (D - t) + t_2$;
        $\{ D \longleftrightarrow D_1 \wedge D_2 \}$
        $L := (L \backslash \{D\}) \cup \{D_1, D_2\}$
    else ($t = \beta$) do
        $t_1 := \beta_1$; $t_2 := \beta_2$;
        $D' := ((D - t) + t_1) + t_2$;
        $\{ D \longleftrightarrow D' \}$
        $L := (L \backslash \{D\}) \cup \{D'\}$

Example

Design a CNF logically equivalent to $(\neg p \Rightarrow \neg q) \Rightarrow (p \Rightarrow q)$.

$(\neg p \Rightarrow \neg q) \Rightarrow (p \Rightarrow q)$

$\neg(\neg\neg p \vee \neg q) \vee (\neg p \vee q)$ ($\Rightarrow$ elimination)

$(\neg\neg\neg p \wedge \neg\neg q) \vee (\neg p \vee q)$ (downward propagation, $\neg$)

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

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

Formula $(\neg p \Rightarrow \neg q) \Rightarrow (p \Rightarrow q)$ is logically equivalent to CNF $(\neg p \vee \neg p \vee q) \wedge (q \vee \neg p \vee q)$.

It is also logically equivalent to $\neg p \vee q$.

Example (bis)

1.	$\{(\neg p \Rightarrow \neg q) \Rightarrow (p \Rightarrow q)\}$	$Init$
2.	$\{\neg(\neg p \Rightarrow \neg q) \vee (p \Rightarrow q)\}$	$\beta, 1$
3.	$\{\neg(\neg p \Rightarrow \neg q) \vee \neg p \vee q\}$	$\beta, 2$
4.	$\{\neg p \vee \neg p \vee q\,,\ \neg\neg q \vee \neg p \vee q\}$	$\alpha, 3$
5.	$\{\neg p \vee \neg p \vee q\,,\ q \vee \neg p \vee q\}$	$\alpha, 4$

Therefore the CNF is

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

It can be simplified into

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

and further into

$$\neg p \vee q\,.$$

Simplification of clausal forms

The normalization algorithm usually leads to CNF that can (should) be simplified.

1. Keep only one occurrence of a literal inside a clause.

   Example : $(\neg p \vee q \vee \neg p) \wedge (r \vee \neg p) \longleftrightarrow (\neg p \vee q) \wedge (r \vee \neg p)$

2. Valid clauses (containing a complementary pair) can be omitted.

   Example : $(\neg p \vee q \vee p) \wedge (r \vee \neg p) \longleftrightarrow (r \vee \neg p)$

3. If a clause c_1 is included into a clause c_2, then c_2 can be omitted.

   Example : $(r \vee q \vee \neg p) \wedge (\neg p \vee r) \longleftrightarrow (\neg p \vee r)$

These simplifications lead to a *pure* normal form, which is still not unique. For instance, $(p \vee \neg q) \wedge q$ and $p \wedge q$ are pure, logically equivalent CNFs.

Resolution rule I

A clause set (set of clauses) S is inconsistent if and only if $S \models \Box$.
($\Box$ is the empty clause, also denoted *false*.)

Idea. Demonstrate S inconsistency by "deriving" $\Box$ (*false*) from S.

Let A, B, X be formulas, let v be a valuation.

Assume $v(A \vee X) = T$ and $v(B \vee \neg X) = T$.

If $v(X) = T$, then $v(B) = T$,
therefore $v(A \vee B) = T$.

If $v(X) = F$, then $v(A) = T$,
therefore $v(A \vee B) = T$.

As a result, $\{(A \vee X), (B \vee \neg X)\} \models (A \vee B)$.

Resolution rule : special case where X is a proposition and where A, B are clauses.

Resolution rule II

Relation $\vdash_{\mathcal{R}}$ (or $\vdash$) is inductively defined
between a clause set and a clause ;
it is the smallest relation satisfying these conditions :

1. If $C \in S$, then $S \vdash C$.
2. Let $C_1 = (C_1' \vee p)$ and $C_2 = (C_2' \vee \neg p)$;
   if $S \vdash C_1$ and $S \vdash C_2$, then $S \vdash C_1' \vee C_2'$.

Clauses C_1 and C_2 can be *resolved* (with respect to p) ;
clause *Res*$(C_1, C_2) =_{def} C_1' \vee C_2'$ is their *resolvent*.

If S is a clause set, S^R is defined as the smallest superset of S
containing the resolvents of its elements.

$S^R = \{C : S \vdash C\} = \{C : S^R \vdash C\}$.

Soundness of resolution rule

Let S a clause set and C a clause. We must prove, if $S \vdash C$, then $S \models C$. It is sufficient to see that relation $\models$ (restricted to clause sets and clauses) satisfies the characteristic conditions of relation $\vdash_{\mathcal{R}}$:

1. If $C \in S$, then $S \models C$.
2. Let $C_1 = (C_1' \vee p)$ and $C_2 = (C_2' \vee \neg p)$;
   if$S \models C_1$ and $S \models C_2$, then $S \models C_1' \vee C_2'$.

Condition 1 is obviously satisfied ; condition 2 results from $\{(A \vee X), (B \vee \neg X)\} \models (A \vee B)$.

Comment. Clause sets S and S^R are always logically equivalent.

Completeness of resolution rule I

If S is a clause set, if A is a clause and if $S \models A$, can we deduce $S \vdash_{\mathcal{R}} A$?
Obviously not :

$$\{p, \neg p\} \models q \, ,$$

but

$$\{p, \neg p\} \not\vdash_{\mathcal{R}} q \, .$$

Fortunately, we do not need so much ; instead of proving $S \models A$, we prove the equivalent $S, \neg A \models \Box$.
The following result can be used :

Theorem. If $S \models \Box$, then $S \vdash_{\mathcal{R}} \Box$.

This "weak completeness" is in fact as powerful as completeness (why ?).

Semantic tree

Let S a formula or a formula set, with $\Pi_S = \{p_1, p_2, \ldots\}$.

A *semantic tree* is a complete, balanced binary tree, labelled as follows : left branches at level i are labelled p_i and right branches are labelled $\neg p_i$.

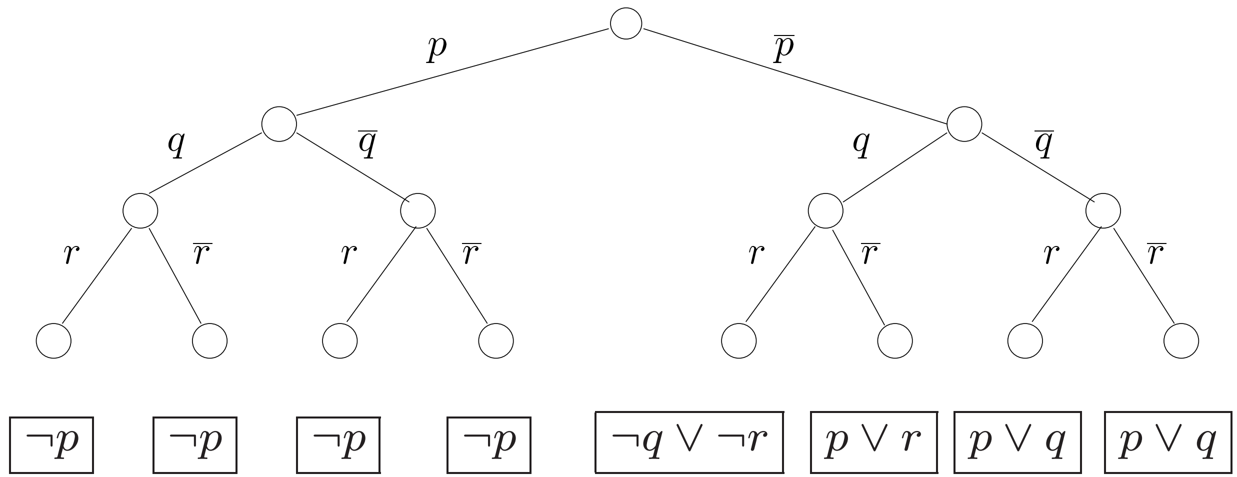
The leaves (or full branches) of the semantic tree S correspond to the valuations of Π_S and S.

Each path $\mathcal{C}$ from the root to some node n at level i defines

- a proposition set, $\Pi(n) = \{p_1, \ldots, p_i\}$;
- a valuation v_n on this set ;
  $v_n(p_k) = T$ if $p_k \in \mathcal{C}$ and $v_n(p_k) = F$ if $\neg p_k \in \mathcal{C}$.

Semantic tree : an example

Let $S = \{p \vee q, p \vee r, \neg q \vee \neg r, \neg p\}$, a clause set.
$\Pi_S = \{p, q, r\}$.
A semantic tree is :



The tree is finite since Π_S is finite.

As S is inconsistent, each leaf can be labelled with a clause made false by the valuation associated with that leaf.

Completeness of the resolution method (finite case) I

If S is a finite inconsistent clause set, then $S \vdash \Box$.

Let $\mathcal{A}$ be a semantic tree for S.

The path from the root to node n defines a proposition set $\Pi(n)$ and a valuation v_n for this set ; $v_n(\ell) = T$ for each labelling literal ℓ on the path.

S is inconsistent, so the valuation associated with any leaf f of $\mathcal{A}$ falsifies some clause $C_f \in S$. We label f with C_f. Observe that

$$\Pi_{C_f} \subseteq \Pi(f) = \Pi_S \quad \text{et} \quad v_f(C_f) = F\,.$$

(Π_{C_f} is the set of atoms occurring in C_f.)

We will attempt to propagate leaf labelling upward : each node n will be labelled with some clause $C_n \in S^R$ such that

$$\Pi_{C_n} \subseteq \Pi(n) \subseteq \Pi_S \quad \text{and} \quad v_n(C_n) = F.$$

If this propagation succeeds, root r will be labelled with $C_r \in S^R$ such that

$$\Pi_{C_r} \subseteq \Pi(r) \quad \text{et} \quad v_r(C_r) = F.$$

As $\Pi(r) = \emptyset$ and $v_r(C_r) = F$, the only possibility is $C_r = \Box$.

Completeness of the resolution method (finite case) II

How to label node n ?

Let n_1, n_2 the children of node n ; assume

$$\Pi(n_1) = \Pi(n_2) = \Pi(n) \cup \{p\}.$$

$$n(C_n)$$

$$p \swarrow \qquad\qquad \searrow \neg p$$

$$n_1(C_{n_1}) \qquad\qquad n_2(C_{n_2})$$

Assume

$C_{n_1} \in S^R$ and $\Pi_{C_{n_1}} \subseteq \Pi(n_1)$ and $v_{n_1}(C_{n_1}) = F$
$C_{n_2} \in S^R$ and $\Pi_{C_{n_2}} \subseteq \Pi(n_2)$ and $v_{n_2}(C_{n_2}) = F$

Node n is labelled as follows :

— If $p \notin \Pi_{C_{n_i}}$ for $i = 1$ or 2, then $C_n = C_{n_i}$.
— If $p \in \Pi_{C_{n_1}}$ and $p \in \Pi_{C_{n_2}}$:
$v_{n_1}(C_{n_1}) = F$ so $C_{n_1} = C'_{n_1} \vee \neg p$ and $v_{n_2}(C_{n_2}) = F$ so $C_{n_2} = C'_{n_2} \vee p$.
Let $C_n = C'_{n_1} \vee C'_{n_2}$ ($= Res_p(C_{n_1}, C_{n_2})$).

In both cases : $C_n \in S^R$ and $\Pi_{C_n} \subseteq \Pi(n)$ et $v_n(C_n) = F$.

and the completeness (finite case) is proved.

Completeness of the resolution method (infinite case)

Due to the compactness theorem, the statement

$$\square \in S^R \quad \text{iff} \quad S \text{ is inconsistent}$$

remains true if S is infinite.

If $\square \in S^R$, then S^R and therefore S are inconsistent.

If S is inconsistent, there is a finite inconsistent subset S_f, so $\square \in S_f^R$ and therefore $\square \in S^R$ since $S_f^R \subset S^R$.

Resolution procedure I

If S is a clause set,
let $\mathcal{M}_S$ be the set of all models of S.
S is inconsistent iff $\mathcal{M}_S = \emptyset$.

Resolution procedure
$S := S_0$;   (S_0 clause set)
$\{\mathcal{M}_S = \mathcal{M}_{S_0}\}$
While $\Box \notin S$, do :
  select $p \in \Pi_S$,
    $C_1 = (C_1' \vee p) \in S$,
    $C_2 = (C_2' \vee \neg p) \in S$;
  $S := S \cup \{\mathit{Res}(C_1, C_2)\}$
  $\{\mathcal{M}_S = \mathcal{M}_{S_0}\}$

Comment on selection procedure : each resolvent pair can be selected only once ; this provides termination since, if the lexicon size is n, no more than 3^n (non valid) clauses can be generated.

Resolution procedure II

Invariant : only logical consequences are inserted into S so the set $\mathcal{M}_S$ does not change.

The procedure terminates smoothly (false guard) or aborts (no possible selection).

Smooth termination : when the guard is false and the computation stops, the final value S_f is such that $\mathcal{M}_{S_f} = \mathcal{M}_{S_0}$ and $\Box \in S_f$, so S_f and S_0 are inconsistent.

Abortion : If all resolvents have been produced and none of them is $\Box$, then $\mathcal{M}_{S_f} = \mathcal{M}_{S_0}$ and $\Box \notin S_f$; both S_f and S_0 are consistent.

A derivation of $\Box$ (*false*) from S is a *refutation* of S.

Refutations : examples I

Let $S = \{(p \vee q), (p \vee r), (\neg q \vee \neg r), (\neg p)\}$.

Clause numbering :

1. $p \vee q$
2. $p \vee r$
3. $\neg q \vee \neg r$
4. $\neg p$

Two refutations :

5.	$p \vee \neg r$	$(1, 3)$
6.	q	$(1, 4)$
7.	$p \vee \neg q$	$(2, 3)$
8.	r	$(2, 4)$
9.	p	$(2, 5)$
10.	$\neg r$	$(3, 6)$
11.	$\neg q$	$(3, 8)$
12.	$\neg r$	$(4, 5)$
13.	$\neg q$	$(4, 7)$
14.	$\Box$	$(4, 9)$

5.	q	$(1, 4)$
6.	r	$(2, 4)$
7.	$\neg q$	$(3, 6)$
8.	$\Box$	$(5, 7)$

Refutations : examples II

Let $S = \{p, \neg p \vee q\}$.

Clause numbering :

1. p
2. $\neg p \vee q$

Derivation :

3. $q \quad (1, 2)$

Let $S = \{p, \neg p \vee q, \neg q\}$.

Clause numbering :

1. p
2. $\neg p \vee q$
3. $\neg q$

Refutation :

4. $q \quad (1, 2)$
5. $\Box \quad (3, 4)$