Propositional Logic (2B)

Young W. Lim
11/8/15

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using LibreOffice/OpenOffice.

List of common logical connectives [edit]

Commonly used logical connectives include

- Negation (not): ¬ , N (prefix), ~
- Conjunction (and): $\wedge$, K (prefix), & , ∙
- Disjunction (or): $\vee$, A (prefix)
- Material implication (if...then): $\rightarrow$, C (prefix), $\Rightarrow$, $\supset$
- Biconditional (if and only if): $\leftrightarrow$, E (prefix), $\equiv$, $=$

Alternative names for biconditional are "iff", "xnor" and "bi-implication".

For example, the meaning of the statements *it is raining* and *I am indoors* is transformed when the two are combined with logical connectives:

- It is **not** raining ($\neg P$)
- It is raining **and** I am indoors ($P \wedge Q$)
- It is raining **or** I am indoors ($P \vee Q$)
- **If** it is raining, **then** I am indoors ($P \rightarrow Q$)
- **If** I am indoors, **then** it is raining ($Q \rightarrow P$)
- I am indoors **if and only if** it is raining ($P \leftrightarrow Q$)

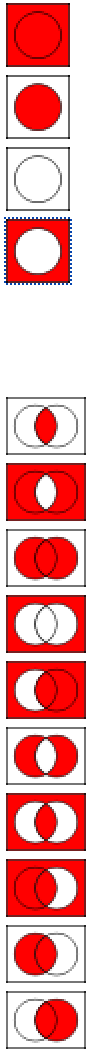
For statement P = *It is raining* and Q = *I am indoors*.

It is also common to consider the *always true* formula and the *always false* formula to be connective:

- True formula ($\top$, 1, V [prefix], or T)
- False formula ($\bot$, 0, O [prefix], or F)

Name / Symbol		Truth table		Venn diagram
		$P =$ 0	1	
Truth/Tautology	$\top$	1	1	
Proposition P		0	1	
False/Contradiction	$\bot$	0	0	
Negation	$\neg$	1	0	

Binary connectives		$P =$ 0 0 1 1	
		$Q =$ 0 1 0 1	
Conjunction	$\wedge$	0 0 0 1	
Alternative denial	$\uparrow$	1 1 1 0	
Disjunction	$\vee$	0 1 1 1	
Joint denial	$\downarrow$	1 0 0 0	
Material conditional	$\rightarrow$	1 1 0 1	
Exclusive or	$\nleftrightarrow$	0 1 1 0	
Biconditional	$\leftrightarrow$	1 0 0 1	
Converse implication	$\leftarrow$	1 0 1 1	
Proposition P		0 0 1 1	
Proposition Q		0 1 0 1	

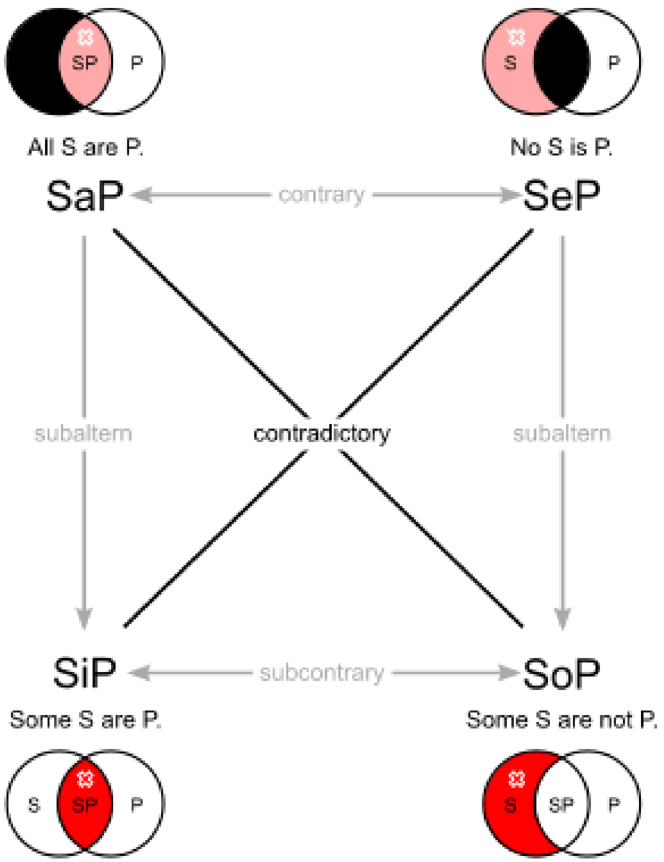


a broad use in contemporary philosophy

- the primary bearers of truth-value
- the objects of belief and other “propositional attitudes” (i.e., what is believed, doubted)
- the referents of that-clauses
- the meanings of sentences

Propositions are
- the **sharable objects** of the attitudes
- the primary bearers of **truth** and **falsity**

Does **not** certain candidates for propositions,
- **thought**- and **utterance**-tokens,
  which presumably are not sharable
- **concrete** events or facts,
  which presumably cannot be false.



A categorical proposition is a simple proposition containing two terms, subject and predicate, in which the predicate is either asserted or denied of the subject.

four logical forms.

The 'A' proposition,
the universal affirmative (universalis affirmativa),
'omne S est P',
usually translated as '**every** S is a P'.

The 'E' proposition,
the universal negative (universalis negativa),
'nullum S est P',
usually translated as '**no** S are P'.

The 'I' proposition,
the particular affirmative (particularis affirmativa),
'quoddam S est P',
usually translated as '**some** S are P'.

The 'O' proposition,
the particular negative (particularis negativa), Latin
'quoddam S non est P',
usually translated as '**some** S are **not** P'.

Material Conditional (1)

- a logical connective (or a binary operator) that is often symbolized by a forward arrow "→"

- is used to form statements of the form "p→q" (termed a conditional statement) which is read as "if p then q" and conventionally compared to the English construction "If...then...".

- But unlike as the English construction may, the conditional statement "p→q" does not specify a causal relationship between p and q
-
- is to be understood to mean "**if p is true, then q is also true**" such that the statement "p→q" is false only when p is true and q is false.

- The material conditional is also to be distinguished from **logical consequence**.

p → q

antecedent consequent

In classical logic $p \rightarrow q$ is logically equivalent to $\neg(p \wedge \neg q)$ and by De Morgan's Law to $\neg p \vee q$

Material Conditional (2)

is to be understood to mean "**if p is true, then q is also true**" such that the statement "p→q" is false only when p is true and q is false.

the negative compound: not both p and not q.

p→q is false
if and only if both p is true and q is false.

p→q is true
if and only if either p is false or q is true (or both).

The compound p→q is logically equivalent also to $\boxed{\neg p \lor q}$ (either not p, or q (or both)), and to $\boxed{\neg q \rightarrow \neg p}$ (if not q then not p).

But it is not equivalent to ¬p → ¬q,
which is equivalent to q→p.

In classical logic $p \rightarrow q$ is logically equivalent to $\neg(p \land \neg q)$ and by De Morgan's Law to $\neg p \lor q$

p	q	p→q
T	T	T
T	F	F
F	T	T
F	F	T

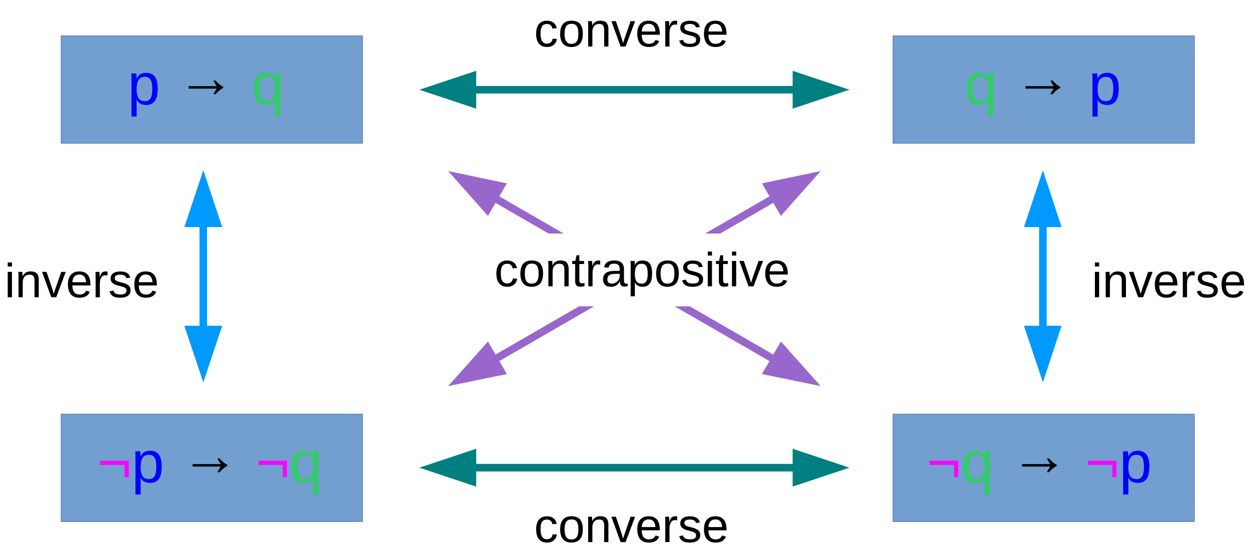
¬q → ¬p

p	q	p→q
T	T	T
T	F	**F**
F	T	T
F	F	**T**

Comparison

from en.wikipedia.org

implication	if P then Q	first statement implies truth of second
inverse	if ~P then ~Q	negation of both statements
converse	if Q then P	reversal of both statements
contrapositive	if ~Q then ~P	reversal and negation of both statements
negation	P and ~Q	contradicts the implication



In logic, contraposition is a law, which says that a conditional statement is logically equivalent to its contrapositive.

The contrapositive of the statement has
its antecedent and consequent **inverted** and **flipped**:
the contrapositive of $P \rightarrow Q$ is thus $\neg Q \rightarrow \neg P$.

For instance, the **proposition** "*All bats are mammals*"

can be restated as the **conditional** "*If something is a bat, then it is a mammal*".

Now, the law says that statement is identical to the **contrapositive** "*If something is not a mammal, then it is not a bat.*"

Note that if $P \rightarrow Q$ is true and
we are given that Q is false, $\neg Q$,
it can logically be concluded that P must be false, $\neg P$.

This is often called the **law of contrapositive**, or the **modus tollens** rule of inference.

Contraposition

Consider the Euler diagram shown. According to this diagram, if something is in A, it must be in B as well. So we can interpret "all of A is in B" as:

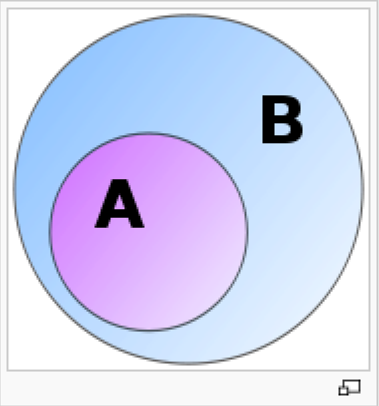
$$A \to B$$

It is also clear that anything that is **not** within B (the white region) **cannot** be within A, either. This statement,

$$\neg B \to \neg A$$

is the contrapositive. Therefore we can say that

$$(A \to B) \to (\neg B \to \neg A).$$



Practically speaking, this may make life much easier when trying to prove something. For example, if we want to prove that every girl in the United States (A) is blonde (B), we can either try to directly prove $A \to B$ by checking all girls in the United States to see if they are all blonde. Alternatively, we can try to prove $\neg B \to \neg A$ by checking all non-blonde girls to see if they are all outside the US. This means that if we find at least one non-blonde girl within the US, we will have disproved $\neg B \to \neg A$, and equivalently $A \to B$.

To conclude, for any statement where A implies B, then *not B* always implies *not A*. Proving or disproving either one of these statements automatically proves or disproves the other. They are fully equivalent.

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

A triangle and its slope

name	form	description
implication	if *P* then *Q*	first statement implies truth of second
inverse	if not *P* then not *Q*	negation of both statements
converse	if *Q* then *P*	reversal of both statements
contrapositive	if not *Q* then not *P*	reversal and negation of both statements
negation	*P* and not *Q*	contradicts the implication

Examples [edit]

Take the statement "*All red objects have color.*" This can be equivalently expressed as "*If an object is red, then it has color.*"

- The **contrapositive** is "*If an object does not have color, then it is not red.*" This follows logically from our initial statement and, like it, it is evidently true.
- The **inverse** is "*If an object is not red, then it does not have color.*" An object which is blue is not red, and still has color. Therefore in this case the inverse is false.
- The **converse** is "*If an object has color, then it is red.*" Objects can have other colors, of course, so, the converse of our statement is false.
- The **negation** is "*There exists a red object that does not have color.*" This statement is false because the initial statement which it negates is true.

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

Contraposition : Formal Definition

A proposition *Q* is implicated by a proposition *P* when the following relationship holds:

$$(P \to Q)$$

This states that, "if *P*, then *Q*", or, "if *Socrates is a man*, then *Socrates is human*." In a conditional such as this, *P* is the antecedent, and *Q* is the consequent. One statement is the **contrapositive** of the other only when its antecedent is the negated consequent of the other, and vice versa. The contrapositive of the example is

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

That is, "If not-*Q*, then not-*P*", or, more clearly, "If *Q* is not the case, then *P* is not the case." Using our example, this is rendered "If *Socrates is not human*, then *Socrates is not a man*." This statement is said to be *contraposed* to the original and is logically equivalent to it. Due to their logical equivalence, stating one effectively states the other; when one is true, the other is also true. Likewise with falsity.

Strictly speaking, a contraposition can only exist in two simple conditionals. However, a contraposition may also exist in two complex conditionals, if they are similar. Thus, $\forall x(Px \to Qx)$, or "All *P*s are *Q*s," is contraposed to $\forall x(\neg Qx \to \neg Px)$, or "All non-*Q*s are non-*P*s."

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

Contraposition : Simple Proof

In first-order logic, the conditional is defined as:

$$A \to B \iff \neg A \lor B$$

We have:

$$\begin{aligned} \neg A \lor B &\iff \neg A \lor (\neg\neg B) \\ &\iff \neg(\neg B) \lor \neg A \\ &\iff \neg B \to \neg A \end{aligned}$$

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

Conversion and Negation

The contrapositive can be compared with three other relationships between conditional statements:

Inversion (the **inverse**): **$\neg P \rightarrow \neg Q$**

"If something is not a bat, then it is not a mammal."
Unlike the contrapositive, the inverse's truth value is not at all dependent on whether or not the original proposition was true, as evidenced here. The inverse here is clearly not true.

Conversion (the **converse**): **$Q \rightarrow P$**.

"If something is a mammal, then it is a bat."
The converse is actually the contrapositive of the inverse and so always has the same truth value as the inverse, which is not necessarily the same as that of the original proposition.

Negation: **$\neg(P \rightarrow Q) = P$ and $\neg Q$**

"There exists a bat that is not a mammal. " If the negation is true, the original proposition (and by extension the contrapositive) is untrue. Here, of course, the negation is untrue.

$$\sim(\sim P \vee Q) = P \text{ and } \sim Q$$

Contradiction

$\perp$ (falsum): represents an arbitrary **contradiction**

$\top$ (tee): denotes an arbitrary **tautology**

$\vdash$ (turnstile): "yields", "proves"

Upside-down T

T

a proposition φ is a **contradiction** iff $\varphi \vdash \perp$

for a contradictory proposition φ
it is true that $\vdash \varphi \rightarrow \psi$ for all ψ ($\perp \rightarrow \psi$)

→ "from falsity, whatever you like"
one may prove any proposition
from a set of axioms which contains **contradictions**.
principle of explosion (ex falso quodlibet)

In the classical logic, especially propositional and first order logic

a proposition φ is a **contradiction** iff it is **unsatisfiable**

In complete logic

Premise

A **premise** : an assumption that something is true.

an **argument** requires

a set of (at least) **two** declarative sentences ("propositions")
known as the **premises**

along with **another** declarative sentence ("proposition")
known as the **conclusion**.

From Middle English, from Old French premisse, from Medieval Latin premissa (“set before”) (premissa propositio (“the proposition set before”)), feminine past participle of Latin praemittere (“to send or put before”), from prae- (“before”) + mittere (“to send”).

two premises and **one conclusion** :
the basic **argument** structure

Because all men are mortal and Socrates is a man,
Socrates is mortal.

2 premises
1 conclusion

3 propositions

Valid Argument Forms (Propositional)

Modus ponens (MP)

If A, then B
A
Therefore, B

Modus tollens (MT)

If A, then B
Not B
Therefore, not A

Hypothetical syllogism (HS)

If A, then B
If B, then C
Therefore, if A, then C

Disjunctive syllogism (DS)

A or B
Not A
Therefore, B

Modus ponens
(Latin) “the way that affirms by affirming"

Modus tollens
(Latin) "the way that denies by denying"

Syllogism
(Greek: συλλογισμός syllogismos) – "conclusion," "inference"

Modus Ponens

The Prolog resolution algorithm
based on the modus ponens form of inference

a general **rule** – the major premise and
a specific **fact** – the minor premise

All men are mortal	rule
Socrates is a man	fact
Socrates is mortal	

modus ponendo ponens
(Latin) “the way that affirms by affirming";
often abbreviated to **MP** or **modus ponens**

P implies Q;
P is asserted to be true,
so therefore Q must be true

one of the accepted mechanisms for the construction of deductive proofs that includes the "rule of definition" and the "rule of substitution"

Facts	a	a
Rules	$a \rightarrow b$	b :- a
Conclusion	b	b

Facts	man(’Socrates’).
Rules	mortal(X) :- man(X).
Conclusion	mortal(’Socrates’).

Syllogism (1)

A syllogism (Greek: συλλογισμός – syllogismos – "conclusion," "inference") is

a kind of logical argument that applies deductive reasoning to arrive at a conclusion based on two or more **propositions** that are asserted or assumed to be true.

In its earliest form, defined by Aristotle,
from the combination of
a general statement (the major premise) and ⟺ rule
a specific statement (the minor premise), ⟺ fact
a conclusion is deduced.

For example, knowing
that all men are mortal (major premise) and ⟺ rule
that Socrates is a man (minor premise), ⟺ fact
we may validly conclude that Socrates is mortal.

Syllogism (2)

A categorical syllogism consists of three parts:

Major premise:	All humans are mortal.	⟺	major term	(the predicate of the conclusion)
Minor premise:	All Greeks are humans.	⟺	minor term	(the subject of the conclusion)
Conclusion:	All Greeks are mortal.			

Each part - a categorical proposition - two categorical terms

In Aristotle, each of the premises is in the form

"All A are B"	universal proposition
"Some A are B"	particular proposition
"No A are B"	universal proposition
"Some A are not B"	particular proposition

Each of the premises has one term in common with the conclusion:
this common term is called
a major term in a major premise (the predicate of the conclusion)
a minor term in a minor premise (the subject of the conclusion)

Mortal is the major term,
Greeks is the minor term.
Humans is the middle term

Modus Ponens (revisited)

Facts	a	a	minor term
Rules	a → b	b :- a	major term
Conclusion	b	b	

Derivation

A **reversed modus ponens** is used in Prolog

Prolog tries to prove that
a query (b) is a consequence of
the database content (a, a ⇒ b).

Using the major premise, it goes from b to a,
and using the minor premise, from a to true.

Such a sequence of goals is called a **derivation**.

A derivation can be **finite** or **infinite**.

b :- a

a → true

Facts	a	a
Rules	a → b	b :- a
Conclusion	b	b

Facts	a
Rules	b :- a
Conclusion	b

Horn Clause

the **resolvent** of **two Horn clauses** is itself **a Horn clause**
the **resolvent** of **a goal clause** and **a definite clause** is **a goal clause**

These properties of Horn clauses can lead to greater efficiencies in proving a theorem (represented as the negation of a goal clause).

Propositional Horn clauses are also of interest in computational complexity, where the problem of finding truth value assignments to make a conjunction of **propositional Horn clauses** true is a **P-complete** problem (in fact solvable in linear time), sometimes called **HORNSAT**. (The **unrestricted Boolean satisfiability** problem is an **NP-complete** problem however.) **Satisfiability** of **first-order Horn clauses** is undecidable.

By iteratively applying the resolution rule, it is possible

- to tell whether a propositional formula is **satisfiable**
- to prove that a first-order formula is unsatisfiable;

- this method may prove the **satisfiability** of a first-order formula,
- but not always, as it is the case for all methods for first-order logic

Negation As Failure – (1)

PLANNER

if (not (goal p)), then (assert ¬p)

If the goal to prove p fails, then assert ¬p

NAF used to derive **not p** (p is assumed not to hold) from failure to derive p

not p can be different from the statement ¬p of the logical negation of p, depending on the **completeness** of the inference algorithm and thus also on the formal logic system

Prolog

NAF literals of the form of not p can occur in the body of clauses

Can be used to derive other NAF literals

$p \leftarrow q \land \text{not } r$
$q \leftarrow s$
$q \leftarrow t$
t

not p : p is assumed not to hold

¬p : the logical negation of p

completeness of the inference algorithm

semantically complete

every tautology → theorem

sound

every theorem → tautology

Negation As Failure – (2)

The semantics of NAF remained an open issue
until Keith Clark [1978] showed that
it is ***correct*** with respect to the **completion**
of the logic program, where, loosely speaking,
"**only**" and **←** are interpreted as
"**if and only if**", written as "**iff**" or "**≡**".

the completion of the four clauses above is

p ≡ q ∧ not r	
q ≡ s ∨ t	
t ≡ true	
r ≡ false	
s ≡ false	

interpreted ←

p	**← q ∧ not r**
q	**← s**
q	**← t**
t	**←**

Negation As Failure – (3)

the completion of the four clauses above is

p	≡	**q ∧ not r**
q	≡	**s ∨ t**
t	≡	**true**
r	≡	**false**
s	≡	**false**

interpreted

p	←	**q ∧ not r**
q	←	**s**
q	←	**t**
t	←	

$\cancel{\sim q} \lor r \lor p$
$\sim s \lor \cancel{q}$
$\cancel{\sim t} \lor \cancel{q}$
$\cancel{t}$

The NAF inference rule simulates
reasoning explicitly with the completion,
where *both sides* of the equivalence are *negated*
and *negation on the right-hand side* is
distributed down to atomic formulae.

to show **not p**, NAF simulates
reasoning with the equivalences

not p	≡	**not q ∨ r** (≡ false)
not q	≡	**not s ∧ not t** (≡ false)
not t	≡	**false**
not r	≡	**true**
not s	≡	**true**

The NAF derives

not r
not s
t
q
p

Negation As Failure – (4)

In the non-propositional case, (predicate logic with variables)
the completion needs to be augmented with **equality axioms**,
to formalise the assumption that
individuals with distinct names are distinct.
NAF simulates this by **failure of unification**.

For example, given only the two clauses

p(a) ←
p(b) ← t

NAF derives **not** **p(c)**.

The completion of the program is

p(X) ≡ X=a ∨ X=b **equality axioms**

augmented with **unique names axioms**
and **domain closure axioms**.

The completion semantics is closely related both to
circumscription and to the closed world assumption.

Negation As Failure – (5)

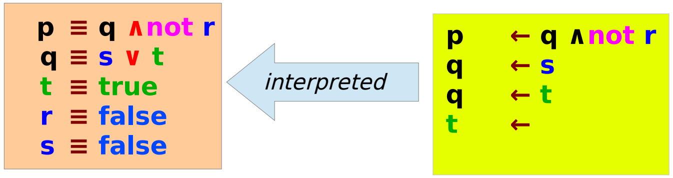
The concept of logical negation in Prolog is problematical,
in the sense that the only method that Prolog can use
to tell if a proposition is false is to try to prove it
(from the facts and rules that it has been told about),
and then if this attempt fails,
it concludes that the proposition is false.

This is referred to as negation as failure.

An obvious problem is that
Prolog may not have been told some critical fact or rule,
so that it will not be able to prove the proposition.

In such a case, the falsity of the proposition
is only relative to the "mini-world-model"
defined by the facts and rules known to the Prolog interpreter.
This is sometimes referred to as the **closed**-world assumption.

A less obvious problem is that,
depending again on the rules and facts
known to the Prolog interpreter, it may take a very long time
to determine that the proposition cannot be proven.
In certain cases, it might "take" infinite time.



Negation As Failure – (6)

Because of the problems of negation-as-failure,
negation in Prolog is represented in modern Prolog interpreters
using the symbol **\+**, which is supposed to be
a mnemonic for **not provable**
with the **** standing for **not** and the **+** for **provable**.
In practice, current Prolog interpreters
tend to support the older operator **not** as well,
as it is present in lots of older Prolog code,
which would break if **not** were not available.

Examples:

```
?- \+ (2 = 4).

true.

?- not(2 = 4).

true.
```

Negation As Failure – (7)

Arithmetic comparison operators in Prolog
each come equipped with a **negation**
which does not have a "negation as failure" problem,
because it is always possible to determine,
for example, if two numbers are equal,
though there may be approximation issues
if the comparison is between fractional (floating-point) numbers.
So it is probably best to use the arithmetic comparison operators
if numeric quantities are being compared.
Thus, a better way to do the comparisons shown above would be:

```
?- 2 =\= 4.

true.
```

Rules for Negation

1 Rules for negations

2 Rules for conditionals

3 Rules for conjunctions

4 Rules for disjunctions

5 Rules for biconditionals

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

Rules of Inference

Rules for negations [edit]

Reductio ad absurdum (or *Negation Introduction*)

$$\frac{\varphi \vdash \psi \qquad \varphi \vdash \neg\psi}{\neg\varphi}$$

Reductio ad absurdum (related to the law of excluded middle)

$$\frac{\neg\varphi \vdash \psi \qquad \neg\varphi \vdash \neg\psi}{\varphi}$$

Noncontradiction (or *Negation Elimination*)

$$\frac{\varphi \qquad \neg\varphi}{\psi}$$

Double negation elimination

$$\frac{\neg\neg\varphi}{\varphi}$$

Double negation introduction

$$\frac{\varphi}{\neg\neg\varphi}$$

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

Rules for Conditionals

Rules for conditionals [edit]

Deduction theorem (or *Conditional Introduction*)

$$\frac{\varphi \vdash \psi}{\varphi \to \psi}$$

Modus ponens (or *Conditional Elimination*)

$$\frac{\varphi \to \psi \qquad \varphi}{\psi}$$

Modus tollens

$$\frac{\varphi \to \psi \qquad \neg\psi}{\neg\varphi}$$

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

Rules for Conjunction

Rules for conjunctions [edit]

Adjunction (or ***Conjunction Introduction***)

$$\frac{\begin{array}{l}\varphi \\ \psi\end{array}}{\varphi \land \psi}$$

Simplification (or ***Conjunction Elimination***)

$$\frac{\varphi \land \psi}{\varphi}$$

$$\frac{\varphi \land \psi}{\psi}$$

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

Rules for disjunction

Rules for disjunctions [edit]

Addition (or *Disjunction Introduction*)

$$\frac{\varphi}{\varphi \lor \psi}$$

$$\frac{\psi}{\varphi \lor \psi}$$

Case analysis

$$\frac{\begin{array}{l}\varphi \lor \psi \\ \varphi \to \chi \\ \psi \to \chi\end{array}}{\chi}$$

Disjunctive syllogism

$$\frac{\begin{array}{l}\varphi \lor \psi \\ \neg\varphi\end{array}}{\psi}$$

$$\frac{\begin{array}{l}\varphi \lor \psi \\ \neg\psi\end{array}}{\varphi}$$

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

Rules for biconditionals

Rules for biconditionals [edit]

Biconditional introduction

$$\frac{\begin{array}{l}\varphi \to \psi \\ \psi \to \varphi\end{array}}{\varphi \leftrightarrow \psi}$$

Biconditional Elimination

$$\frac{\begin{array}{l}\varphi \leftrightarrow \psi \\ \varphi\end{array}}{\psi}$$

$$\frac{\begin{array}{l}\varphi \leftrightarrow \psi \\ \psi\end{array}}{\varphi}$$

$$\frac{\begin{array}{l}\varphi \leftrightarrow \psi \\ \neg\varphi\end{array}}{\neg\psi}$$

$$\frac{\begin{array}{l}\varphi \leftrightarrow \psi \\ \neg\psi\end{array}}{\neg\varphi}$$

$$\frac{\begin{array}{l}\varphi \leftrightarrow \psi \\ \psi \vee \varphi\end{array}}{\psi \wedge \varphi}$$

$$\frac{\begin{array}{l}\varphi \leftrightarrow \psi \\ \neg\psi \vee \neg\varphi\end{array}}{\neg\psi \wedge \neg\varphi}$$

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

References

[1] en.wikipedia.org
[2] en.wiktionary.org
[3] U. Endriss, “Lecture Notes : Introduction to Prolog Programming”
[4] http://www.learnprolognow.org/ Learn Prolog Now!
[5] http://www.csupomona.edu/~jrfisher/www/prolog_tutorial
[6] www.cse.unsw.edu.au/~billw/cs9414/notes/prolog/intro.html
[7] www.cse.unsw.edu.au/~billw/dictionaries/prolog/negation.html
[8] http://ilppp.cs.lth.se/, P. Nugues,` An Intro to Lang Processing with Perl and Prolog