Propositional Logic, Predicate Logic, and Logic Programming

Propositional Logic

DEF: A ***proposition*** is a statement that is either ***true*** or ***false*** (but not both).

In propositional logic, we assume a collection of atomic propositions are given: *p, q, r, s, t,* ....

Then we form compound propositions by using ***logical connectives*** (***logical operators***).

Logical Connectives

Operator	Symbol	Usage	C++/Java
Negation	$\neg$	not	`!`
Conjunction	$\wedge$	and	`&&`
Disjunction	$\vee$	or	`||`
Exclusive or	$\oplus$	xor	`(p||q)&&(!p||!q)` `p!=q`
Conditional	$\rightarrow$	if-then	`p?q:true`
Biconditional	$\leftrightarrow$	iff	`(p&&q)||(!p&&!q)` `p==q`

Compound Propositions: Examples

p = "CS 603 covers logic programming."

q = "CS 603 only covers fun topics."

r = "Logic programming is a fun topic."

$\neg p$ = "CS 603 does not cover logic programming."

$p \wedge q$ = "CS 603 covers logic programming and CS 603 only covers fun topics."

$p \wedge q \rightarrow r$ = "If CS 603 covers logic programming and CS 603 only covers fun topics, then logic programming is a fun topic."

Negation

Negation ("not") turns a true proposition into false, or a false proposition into true.

Logical operators are defined by **truth tables** – tables which give the output of the operator in the right-most column.

Here is the truth table for negation:

p	$\neg p$
T	F
F	T

Conjunction

Conjunction ("and") is only true when both of its components are true:

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

Disjunction

Disjunction (“or”) is true when at least one of its components is true:

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

Exclusive-Or

p	q	$p \oplus q$
T	T	F
T	F	T
F	T	T
F	F	F

Beware: In English language, the word “or” sometimes means “exclusive-or”. Example: The entrée is served with soup or salad.

Conditional (Implication)

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

Beware: This operator might be less intuitive.
Why should $p \rightarrow q$ be true when p is false?
Because $p \rightarrow q$ can't be false if condition is false.

Bi-Conditional

For $p \leftrightarrow q$ to be true, p and q must have the same truth value. Else, $p \leftrightarrow q$ *is false:*

p	q	$p \leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

Q : Which operator is the negation of $\leftrightarrow$?

Tautologies and contradictions

DEF: A ***tautology*** is a compound proposition that always evaluates to ***true***.

EX: $p \vee \neg p$

DEF: A ***contradiction*** is a compound proposition that always evaluates to ***false***.

EX: $p \wedge \neg p$

p	$\neg p$	$p \vee \neg p$
T	F	T
F	T	T

p	$\neg p$	$p \wedge \neg p$
T	F	F
F	T	F

Tautology example

Show that $[\neg p \wedge (p \vee q)] \rightarrow q$ is a tautology:

p	q	$\neg p$	$p \vee q$	$\neg p \wedge (p \vee q)$	$[\neg p \wedge (p \vee q)] \rightarrow q$
T	T	F	T	F	T
T	F	F	T	F	T
F	T	T	T	T	T
F	F	T	F	F	T

Logical Equivalences

DEF: Two compound propositions p, q are ***logically equivalent*** if their biconditional joining $p \leftrightarrow q$ is a tautology. Logical equivalence is denoted by $p \Leftrightarrow q$.

The easiest way to check for logical equivalence is to verify that the truth tables of both expressions yield identical columns.

Example of logical equivalence

Show that $p \rightarrow q \Leftrightarrow \neg p \vee q$:

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

Contrapositives

DEF: The ***contrapositive*** of a logical implication is formed by reversing the implication while negating both components. So the contrapositive of $p \rightarrow q$ is $\neg q \rightarrow \neg p$.

Show that $p \rightarrow q \Leftrightarrow \neg q \rightarrow \neg p$:

p	q	$p \rightarrow q$	$\neg q$	$\neg p$	$\neg q \rightarrow \neg p$
T	T	**T**	F	F	**T**
T	F	**F**	T	F	**F**
F	T	**T**	F	T	**T**
F	F	**T**	T	T	**T**

Converses and inverses

The ***converse*** of a logical implication is the reversal of the direction of the implication. I.e. the converse of $p \rightarrow q$ is $q \rightarrow p$.

EX: The converse of "If Donald is a duck then Donald is a bird." is "If Donald is a bird then Donald is a duck."

The ***inverse*** of a logical implication is obtained by negating both its components. I.e. the inverse of $p \rightarrow q$ is $\neg p \rightarrow \neg q$.

Note: the converse $q \rightarrow p$ is logically equivalent to the inverse $\neg p \rightarrow \neg q$:

Q: Why?

Logical Non-Equivalence of Conditional and Its Converse

The conditional $p \rightarrow q$ is not logically equivalent to its converse $q \rightarrow p$.

p	q	$p \rightarrow q$	$q \rightarrow p$	$(p \rightarrow q) \leftrightarrow (q \rightarrow p)$
T	T	T	T	T
T	F	F	T	F
F	T	T	F	F
F	F	T	T	T

Note: the conditional $p \rightarrow q$ is also not logically equivalent to its inverse $\neg p \rightarrow \neg q$. Q: Why?

Derivational Proofs

When compound propositions involve more and more atomic components, the size of the truth table for the compound propositions increases

Q: How many rows are required to construct the truth-table of a proposition involving n atomic components?

A: In general, 2^n rows are required, so truth tables are too inefficient for large n.

Derivational Proofs

EX: Consider the compound proposition

$$(p \rightarrow p) \vee (\neg(s \wedge r) \vee \neg t) \vee (\neg q \rightarrow r)$$

Q: Why is this a tautology?

Derivational Proofs

A: The first component is a tautology $(p \rightarrow p)$, and the disjunction of True with any other compound proposition yields True:

$(p \rightarrow p) \vee (\neg(s \wedge r) \vee \neg t)) \vee (\neg q \rightarrow r)$

$\Leftrightarrow \quad \mathrm{T} \vee (\neg(s \wedge r) \vee \neg t)) \vee (\neg q \rightarrow r)$

$\Leftrightarrow \quad \mathrm{T}$

Derivational proofs formalize the intuition of this example. It is often not necessary to consider every case in the truth table individually.

Table of Logical Equivalences

- Identity laws
  - Like adding 0
- Domination laws
  - Like multiplying by 0
- Idempotent laws
  - Delete redundancies
- Double negation
  - "I don't not enjoy CS 603"
- Commutativity
  - Like "$x+y = y+x$"
- Associativity
  - Like "$(x+y)+z = y+(x+z)$"
- Distributivity
  - Like "$(x+y)z = xz+yz$"
- De Morgan

Equivalence	*Name*
$p \wedge \mathbf{T} \iff p$ $p \vee \mathbf{F} \iff p$	Identity laws
$p \vee \mathbf{T} \iff \mathbf{T}$ $p \wedge \mathbf{F} \iff \mathbf{F}$	Domination laws
$p \vee p \iff p$ $p \wedge p \iff p$	Idempotent laws
$\neg(\neg p) \iff p$	Double negation law
$p \vee q \iff q \vee p$ $p \wedge q \iff q \wedge p$	Commutative laws
$(p \vee q) \vee r \iff p \vee (q \vee r)$ $(p \wedge q) \wedge r \iff p \wedge (q \wedge r)$	Associative laws
$p \vee (q \wedge r) \iff (p \vee q) \wedge (p \vee r)$ $p \wedge (q \vee r) \iff (p \wedge q) \vee (p \wedge r)$	Distributive laws
$\neg(p \wedge q) \iff \neg p \vee \neg q$ $\neg(p \vee q) \iff \neg p \wedge \neg q$	De Morgan's laws

DeMorgan's Laws

DeMorgan's laws allow for simplification of negations of complex expressions

- Conjunctional negation:

$$\neg(p_1 \wedge p_2 \wedge \ldots \wedge p_n) \Leftrightarrow (\neg p_1 \vee \neg p_2 \vee \ldots \vee \neg p_n)$$

"It's not the case that all are true iff one is false."

- Disjunctional negation:

$$\neg(p_1 \vee p_2 \vee \ldots \vee p_n) \Leftrightarrow (\neg p_1 \wedge \neg p_2 \wedge \ldots \wedge \neg p_n)$$

"It's not the case that one is true iff all are false."

More Logical Equivalences

Also recall these equivalences from earlier:

- $p \vee \neg p \Leftrightarrow \mathbf{T}$ (Maximum law)
- $p \wedge \neg p \Leftrightarrow \mathbf{F}$ (Minimum law)
- $p \rightarrow q \Leftrightarrow \neg p \vee q$ (Implication law)

Tautology example (revisited)

Show that $[\neg p \wedge (p \vee q)] \rightarrow q$ is a tautology without using a truth table:

Tautology proof

$[\neg p \wedge (p \vee q)] \rightarrow q$	
$\Leftrightarrow [(\neg p \wedge p) \vee (\neg p \wedge q)] \rightarrow q$	Distributive
$\Leftrightarrow [\ F \vee (\neg p \wedge q)] \rightarrow q$	Minimum
$\Leftrightarrow [\neg p \wedge q\ ] \rightarrow q$	Identity
$\Leftrightarrow\ \neg\ [\neg p \wedge q\ ] \vee q$	Implication
$\Leftrightarrow [\neg(\neg p) \vee\ \neg q\ ] \vee q$	DeMorgan
$\Leftrightarrow [p \vee\ \neg q\ ] \vee q$	Double Negation
$\Leftrightarrow p \vee [\neg q \vee q\ ]$	Associative
$\Leftrightarrow p \vee [q \vee \neg q\ ]$	Commutative
$\Leftrightarrow p \vee T$	Maximum
$\Leftrightarrow T$	Domination

Predicate Logic

Consider this compound proposition:

If Ollie is an octopus then Ollie has 8 limbs.

Let p = "Ollie is an octopus" and q = "Ollie has 8 limbs".

The compound proposition is represented by: $p \rightarrow q$.

Next consider:

For all x, if x is an octopus then x has 8 limbs.

Let $P(x)$ = "x is an octopus" and $Q(x)$ = "x has 8 limbs".

This can be represented by: (for all x) ($P(x) \rightarrow Q(x)$).

Semantics

For logical propositions to have meaning, they need to describe something. Previously, propositions such as “Ollie is an octopus” and “Ollie has 8 limbs” had no intrinsic meaning. Either they are true or false, but no more.

In order to give meanings to propositions, we need to have a ***universe of discourse***, i.e. a collection of *subjects* (or *nouns*) about which the propositions relate.

Q: What is the universe of discourse for the two propositions above?

A: There are many possible answers:

- Ollie (this is the smallest correct answer)
- Sea creatures
- All animals

Predicates

A ***predicate*** is a property or description of subjects in the universe of discourse. In the following, predicates are *italicized* :

- Ollie *is an octopus*.
- Johnny *is tall*.
- 17 *is a prime number.*

Propositional Functions

By taking a variable subject denoted by symbols such as *x*, *y*, *z*, and applying a predicate one obtains a ***propositional function*** (or ***formula***). When an object from the universe is plugged in for *x*, *y*, *z*, etc. a truth value results:

- x is an octopus . ...e.g. plug in *x* = Ollie
- *y* is tall. ...e.g. plug in *y* = Johnny
- *n* is a prime number. ...e.g. plug in *n* = 1000

Multivariable Predicates

Multivariable predicates generalize predicates to allow descriptions of relationships between subjects. These subjects may or may not even be in the same universe of discourse. For example:

- Johnny *is at least* 5 *inches taller than* Debbie.

Q: What universes of discourse are involved?

A: The most obvious answer is: The first and third variable have the set of people as their universe of discourse, while the second variable has the set of numbers.

Generalization: *x* is at least *n* inches taller than *y*

Quantifiers

There are two quantifiers:

- Existential Quantifier

  "$\exists$" reads "there exists"

- Universal Quantifier

  "$\forall$" reads "for all"

Each is placed in front of a propositional function and ***binds*** it to obtain a proposition with semantic value.

$$(\exists\ x)\ (P(x) \rightarrow Q(x))$$

$$(\forall\ x)\ (P(x) \rightarrow Q(x))$$

Existential Quantifier

- "$\exists x \; P(x)$" is true when *any* instance can be found which when plugged in for x makes $P(x)$ true
- Like disjunctioning over entire universe

$$\exists x \, P(x) \iff P(x_1) \vee P(x_2) \vee P(x_3) \vee \ldots$$

Existential Quantifier

Consider a universe consisting of:

- Leo: a lion
- Jan: an octopus with all 8 tentacles
- Bill: an octopus with only 7 tentacles

And recall the propositional functions:

- $P(x)$ = "x is an octopus"
- $Q(x)$ = "x has 8 limbs"

$$\exists x \ (P(x) \rightarrow Q(x))$$

Q: Is the proposition true or false?

Existential Quantifier

A: True. Proposition is equivalent to

$(P(\text{Leo}) \rightarrow Q(\text{Leo})) \vee (P(\text{Jan}) \rightarrow Q(\text{Jan})) \vee (P(\text{Bill}) \rightarrow Q(\text{Bill}))$

P(Leo) is false because Leo is a Lion, not an octopus, therefore the conditional P(Leo) $\rightarrow Q$(Leo) is true, and the disjunction is true.

Leo is called a *positive example*.

Universal Quantifier

- "$\forall x \; P(x)$" is true when *every* instance of x makes $P(x)$ true when plugged in
- Like conjunctioning over entire universe

$$\forall x \, P(x) \Leftrightarrow P(x_1) \wedge P(x_2) \wedge P(x_3) \wedge \ldots$$

Universal Quantifier

Consider the same universe as before:

- Leo: a lion
- Jan: an octopus with all 8 tentacles
- Bill: an octopus with only 7 tentacles

And the same propositional functions:

- $P(x)$ = "x is an octopus"
- $Q(x)$ = "x has 8 limbs"

$$\forall x \ (P(x) \rightarrow Q(x))$$

Q: Is the proposition true or false?

Universal Quantifier

A: False. The proposition is equivalent to
$(P(\text{Leo}) \rightarrow Q(\text{Leo})) \wedge (P(\text{Jan}) \rightarrow Q(\text{Jan})) \wedge (P(\text{Bill}) \rightarrow Q(\text{Bill}))$

Bill is a ***counterexample***, i.e. a value making an instance (and therefore the whole universal quantification) false.

P(Bill) is true because Bill is an octopus, while Q(Bill) is false because Bill only has 7 tentacles, not 8. Thus the conditional $P(\text{Bill}) \rightarrow Q(\text{Bill})$ is false since T→F gives F, and the conjunction is false.

Note: Leo is <u>not</u> a counterexample.

Multivariate Quantification

Quantification involving only one variable is fairly straightforward. Just a bunch of OR's or a bunch of AND's.

When two or more variables are involved each of which is bound by a quantifier, the order of the binding is important, and the meaning may require more thought.

Parsing Multivariate Quantification

When evaluating an expression such as

$$\exists x \, \forall y \, \exists z \;\; P(x,y,z)$$

translate the proposition in the same order to English:

There exists an x such that for all y there exists a z such that $P(x,y,z)$.

Order matters

Set the universe of discourse to be all natural numbers {0, 1, 2, 3, ... }.

Let $R(x,y)$ = "$x < y$".

Q1: What does $\forall x \exists y\, R(x,y)$ mean?

Q2: What does $\exists y \forall x\, R(x,y)$ mean?

Order matters

$R(x,y)$ = "$x < y$"

A1: $\forall x \exists y\, R(x,y)$:

"All numbers x admit a bigger number y"

A2: $\exists y \forall x\, R(x,y)$:

"Some number y is bigger than all x"

Q: Is each statement *true* or *false*?

Order matters

A: First is true and second is false.

$\forall x \exists y\, R(x,y)$: "All numbers x admit a bigger number y" (just choose $y = x + 1$)

$\exists y \forall x\, R(x,y)$: "Some number y is bigger than all numbers x" (y is never bigger than itself, so setting $x = y$ is a counterexample)

Q: What if we have two quantifiers of the same kind? Does order still matter?

Order matters – but not always

A: No! If we have two quantifiers of the same kind, the order is irrelevant.

$\forall x \forall y$ is the same as $\forall y \forall x$ because these are both interpreted as "for every combination of x and y..."

$\exists x \exists y$ is the same as $\exists y \exists x$ because these are both interpreted as "there is a pair x , y..."

Logical Equivalence with Formulas

DEF: Two logical expressions possibly involving propositional formulas and quantifiers are said to be ***logically equivalent*** if no matter what universe and what particular propositional formulas are plugged in, the expressions always have the same truth value.

EX: $\forall x \exists y\, Q(x,y)$ and $\forall y \exists x\, Q(y,x)$ are equivalent (names of variables don't matter).

EX: $\forall x \exists y\, Q(x,y)$ and $\exists y \forall x\, Q(x,y)$ are <u>not</u> equivalent!

DeMorgan's Laws Revisited

Recall DeMorgan's laws:

- Conjunctional negation:

$$\neg(p_1 \wedge p_2 \wedge \ldots \wedge p_n) \Leftrightarrow (\neg p_1 \vee \neg p_2 \vee \ldots \vee \neg p_n)$$

- Disjunctional negation:

$$\neg(p_1 \vee p_2 \vee \ldots \vee p_n) \Leftrightarrow (\neg p_1 \wedge \neg p_2 \wedge \ldots \wedge \neg p_n)$$

Since the quantifiers are the same as taking a bunch of AND's ($\forall$) or OR's ($\exists$) we have:

- Universal negation:

$$\neg\, \forall x\, \mathrm{P}(x) \Leftrightarrow \exists x\, \neg \mathrm{P}(x)$$

- Existential negation:

$$\neg\, \exists x\, \mathrm{P}(x) \Leftrightarrow \forall x\, \neg \mathrm{P}(x)$$

Negation Example

Simplify: $\neg \forall x \exists y \ (x > y)$

In English, we are trying to find the opposite of "for every x there exists a y such that $x > y$". The opposite is that "for some x there is no y such that $x > y$".

Algebraically, one just flips all quantifiers from $\forall$ to $\exists$ and vice versa, and negates the interior propositional function. In our case we get:

$$\exists x \forall y \ \neg(x > y) \ \Leftrightarrow \ \exists x \forall y \ (x \leq y)$$

Mathematical induction

Suppose the universe of discourse is the natural numbers {0, 1, 2, 3, ... }. Also let $S(n) = n + 1$ denote the successor function.

To prove $(\forall n)\ P(n)$ by mathematical induction, we instead prove that:

$$P(0) \wedge (\forall n)\ (P(n) \rightarrow P(S(n))).$$

This works because it can be shown that:

$$(\forall n)\ P(n) \iff P(0) \wedge (\forall n)\ (P(n) \rightarrow P(S(n))).$$

[Proof is omitted here.]

Logic programming

Logic programming is a declarative programming paradigm in which the set of attributes that a solution should possess are specified, rather than a set of steps to obtain such a solution.

It makes use of pattern-directed invocation of procedural plans from assertions and/or goals.

(Prolog is a logic programming language that starts only from goals, and uses a backtracking control structure so that only one possible computation path must be stored at a time.)

Logic programming with propositional logic

A ***clause*** has the form:

$$p_1 \vee p_2 \vee \ldots \vee p_n \vee \neg q_1 \vee \neg q_2 \vee \ldots \vee \neg q_m$$

This is logically equivalent to:

$$q_1 \wedge q_2 \wedge \ldots \wedge q_m \rightarrow p_1 \vee p_2 \vee \ldots \vee p_n$$

It can be written as a ***rule*** as follows:

$$p_1 , p_2 , \ldots , p_n \leftarrow q_1 , q_2 , \ldots , q_m$$

A logic program is essentially a ***database*** that contains a set of such rules.

Resolution

The following implication is a tautology which is the principle of ***resolution***:

$$(p_1 \vee \ldots \vee p_n \vee \neg q_1 \vee \ldots \vee \neg q_m \vee \neg r) \wedge (r \vee s_1 \vee \ldots \vee s_j \vee \neg t_1 \vee \ldots \vee \neg t_k) \rightarrow (p_1 \vee \ldots \vee p_n \vee \neg q_1 \vee \ldots \vee \neg q_m \vee s_1 \vee \ldots \vee s_j \vee \neg t_1 \vee \ldots \vee \neg t_k)$$

Equivalently, here is resolution in the rule format:

If $p_1, \ldots, p_n \leftarrow q_1, \ldots, q_m, r$

and $r, s_1, \ldots, s_j \leftarrow t_1, \ldots, t_k$

then $p_1, \ldots, p_n, s_1, \ldots, s_j \leftarrow q_1, \ldots, q_m, t_1, \ldots, t_k$

Resolution example

Suppose the database contains these rules:

1. $q , r \leftarrow p$
2. $s \leftarrow q$
3. $s \leftarrow r$
4. $w \leftarrow s , u$
5. $u \leftarrow t$

Our goal is to prove:

$$w \leftarrow p , t$$

Resolution: deduction proof

Continue as follows:

A. Resolve 1 with 2: $r , s \leftarrow p$

B. Resolve A with 3: $s \leftarrow p$

C. Resolve B with 4: $w \leftarrow p , u$

D. Resolve C with 5: $w \leftarrow p , t$

This proves the previously stated goal.

Resolution: contradiction proof

We want to prove the goal $w \leftarrow p, t$
which is equivalent to $(w \vee \neg p \vee \neg t)$.
So assume the opposite:

$$\neg(w \vee \neg p \vee \neg t) \Leftrightarrow (\neg w \wedge p \wedge t).$$

That is, add these new rules to the database:

6. $\leftarrow w$
7. $p \leftarrow$
8. $t \leftarrow$

Resolution: contradiction proof

Now we want to use rules 1, 2, ..., 8 to reach a contradiction. What does a contradiction look like?

The empty clause evaluates to *false*, because it's the identity element of the OR operation. Equivalently, the empty rule is $\leftarrow$, which is our new goal.

Resolution: contradiction proof

The database now contains these rules:

1. $q , r \leftarrow p$
2. $s \leftarrow q$
3. $s \leftarrow r$
4. $w \leftarrow s , u$
5. $u \leftarrow t$
6. $\leftarrow w$
7. $p \leftarrow$
8. $t \leftarrow$

Our goal is to prove: $\leftarrow$

Resolution: contradiction proof

A'. Resolve 1 with 7: $q , r \leftarrow$

B'. Resolve 4 with 6: $\leftarrow s , u$

C'. Resolve 5 with 8: $u \leftarrow$

D'. Resolve A' with 2: $r , s \leftarrow$

E'. Resolve D' with 3: $s \leftarrow$

F'. Resolve E' with B': $\leftarrow u$

G'. Resolve F' with C': $\leftarrow$

Logic programming with predicate logic

All variables that appear within rules are assumed to be universally quantified (∀).

Predicates introduce a new complication: Variables can be ***substituted*** with other expressions by a process known as ***unification***.

Rather than give a formal definition here, we provide some examples on the next few pages.

Unification example 1

Suppose these rules are in the database:

1. $A(x, y) \leftarrow B(y)\ ,\ C(x)$
2. $C(w)\ ,\ D(z) \leftarrow E(w, z)$

To resolve 1 with 2, we must unify $C(w)$ with $C(x)$. Hence substitute $w := x$, which yields:

$$A(x, y)\ ,\ D(z) \leftarrow B(y)\ ,\ E(x, z)$$

Unification example 2

Suppose:

1. $A(h(y), x) \leftarrow B(f(x), y)$
2. $B(w, g(z)) \leftarrow C(z, h(w))$

To resolve, we must unify $B(f(x), y)$ with $B(w, g(z))$. Substitute $w := f(x)$ and $y := g(z)$ to obtain:

$$A(h(g(z)), x) \leftarrow C(z, h(f(x)))$$

Unification example 3

Suppose:

1. $A(x) \leftarrow B(f(x), x)$
2. $B(y, g(y)) \leftarrow C(y)$

To resolve, we must unify $B(f(x), x)$ with $B(y, g(y))$). Substitute $y := f(x)$ and $x := g(y)$. This circularity cannot be satisfied, so unification and resolution do not succeed here.

Unification example 4

Suppose:

1. $A(x) \leftarrow B(x, g(x))$
2. $B(h(y), x) \leftarrow C(x, y)$

The x in rule 1 has different scope from the x in rule 2 . To avoid conflict, replace rule 2 by:

2'. $B(h(y), z) \leftarrow C(z, y)$

To resolve 1 with 2', we must unify $B(x, g(x))$ with $B(h(y), z)$. Substitute $x := h(y)$ and $z := g(x) := g(h(y))$. Therefore:

$$A(h(y)) \leftarrow C(g(h(y)), y)$$

Logic programming in Prolog

To keep things manageable, Prolog only allows ***Horn clauses***, i.e., at most one non-negated literal appears in each clause, or at most one term on left side of rule.

Clause form: $p \vee \neg q_1 \vee \neg q_2 \vee \ldots \vee \neg q_m$

Rule form: $p \leftarrow q_1 , q_2 , \ldots , q_m$

Also, Prolog has several syntax differences from what we have seen so far.