THE CLAUSAL FORM OF LOGIC

Introduction
The clausal form of logic is a restricted subset of the standard form of logic. It has the advantage that it bears greater resemblance to other formalisms used for databases and programming. Moreover, simple, efficient, and reasonably natural resolution theorem provers have been developed for it.

The arrow of clausal form "←"is written in the opposite direction to that normally used in the standard form of logic i.e.

A ← B (A if B)

instead of

B → A (if B then A).

The notation "A ← B" is used in order to draw attention to the conclusion of the clause.

In the clausal form of logic all the expressions are clauses.

A *clause* is an expression of the form

A1,...,Am ← B1,...,Bn

where A1,...,Am, B1,...,Bn are atomic formulae, $n \geq 0$ and $m \geq 0$.
The atomic formulae B1,...,Bn are the joint conditions of the clause and A1,...,Am are the alternative conclusions.

There are no symbols used in clausal representation other than the arrow (←) and the comma(,). However, information that was conveyed by the other logical symbols AND, OR, and NOT are implied based on the position in the clause. Every atomic formula to the left of the arrow is assumed to be separated by an OR. Every atomic formula to the right of the arrow is assumed to be separated by an AND. If there are only atoms to the right or left rather than both, then implication arrow is not represented in the standard logic form. The following are examples of some similar constructs between standard and clausal form:

A, B ←	is the same as	$A \vee B$
D, E, F ←	is the same as	$D \vee E \vee F$
Y ← X	is the same as	$X \rightarrow Y$
Z ← X, Y	is the same as	$X \,\&\, Y \rightarrow Z$
D, E ← A, B, C	is the same as	$A \,\&\, B \,\&\, C \rightarrow D \vee E$
Y ←	is the same as	Y
← X	is the same as	$\neg X$

If the clause contains the variables x1,...,xk then interpret it as stating that

for all x1,...,xk
A1 or ... or Am if B1 and ... and Bn

That is, all the variables are universally quantified.

If n=0 then interpret it as stating unconditionally that

for all x1,...,xk
A1 or ... or Am.

If m=0 then interpret it as stating that

for all x1,...,xk
it is not the case that
B1 and ... and Bn.

If m=n=0 then write it as $\square$ and interpret it as a sentence which is always false.

An atom (or atomic formula) is an expression of the form
$P(t_1,...,t_q)$
where P is an q-place predicate symbol, $t_1,...,t_q$ are terms and $q \geq 1$.
The atom is interpreted as asserting that the relation called P holds among the individuals called $t_1,...,t_q$.

A term is a variable, a constant symbol, or an expression of the form
$f(t_1,...,t_q)$
where f is an q-place function symbol, $t_1,...,t_q$ are terms and $q \geq 1$.

The sets of predicate symbols, function symbols, constant symbols and variables are any mutually disjoint sets.

Note
Function symbols denote mappings from elements of a domain (or tuples of elements of domains) to elements of a domain. For instance, weight is a function that maps objects to their weight: weight(Tom) = 150.
Therefore the predicate greater_than(weight(Bob), 100) means that the weight of Bob is greater than 100. The arguments of a function may themselves be functions.

Examples
Roman(x1) ← Pompeian(x1) ; x1 is a roman if x1 is a pompeian
man(Marcus) ← ; Marcus is a man
loyalto(x2,Caesar), hate(x2,Caesar) ← Roman(x2) ; x2 is loyal to Caesar or x2 hates Caesar if x2 is a roman
← person(x4), ruler(y1), tryassassinate(x4,y1), loyalto(x4,y1) ; it cannot be true that x4 is a
; person, y1 is a ruler, x4 tried
; to assasinate y1 and x4 was
; loyal to y1

Conversion to Clausal Form
Any sentence in standard form can be converted to clausal form. The rules for converting to clausal form can be expressed more simply if implications and equivalences are re-expressed in terms of negation, conjunction, and disjunction, by using the equivalences:

$[X \rightarrow Y] \leftrightarrow \neg X \vee Y$
$[X \leftrightarrow Y] \leftrightarrow [X \rightarrow Y] \,\&\, [Y \rightarrow X]$ i.e. $[X \leftrightarrow Y] \leftrightarrow [\neg X \vee Y] \,\&\, [\neg Y \vee X]$

Once implications and equivalences have been rewritten, the rest of the conversion consists of

i). Moving negations inside the sentence past conjunctions, disjunctions and quantifiers, until they stand only in front of atomic formulae,
ii). Moving disjunctions inside the sentence past conjunctions and quantifiers, until they connect only atoms or negated atoms,
iii). Eliminating existential quantifiers
iv). Renaming the variables so that no two disjunctions of the form
$A1 \vee ... \vee Am \vee \neg B1 \vee ... \vee \neg Bn$ use the same variables, and eliminate the universal quantifiers

v). Re-express disjunctions

$A1 \vee ... \vee Am \vee \neg B1 \vee ... \vee \neg Bn$ of atoms and their negations as clauses

$A1,...,Am \leftarrow B1,...,Bn$

Negations can be moved in front of atoms by repeatedly applying the following equivalences:

$\neg[X \ \& \ Y] \quad \leftrightarrow \quad \neg X \vee \neg Y$

$\neg[X \vee Y] \quad \leftrightarrow \quad \neg X \ \& \ \neg Y$

$\neg \exists u \ X \quad \leftrightarrow \quad \forall u \ \neg X$

$\neg \forall u \ X \quad \leftrightarrow \quad \exists u \ \neg X$

$\neg \neg X \quad \leftrightarrow \quad X$

where X and Y are any formulae and u is any variable.

Disjunctions can be moved inside a sentence until they connect only atoms and their negations by using the equivalences:

$X \vee [Y \ \& \ Z] \quad \leftrightarrow \quad [X \vee Y] \ \& \ [X \vee Z]$

$X \vee \exists u \ Y \quad \leftrightarrow \quad \exists u \ [X \vee Y]$

$X \vee \forall u \ Y \quad \leftrightarrow \quad \forall u \ [X \vee Y]$

where the variable u does not occur in X.

Given a conjunction of sentences S, in order to eliminate existential quantifiers from S it is necessary to eliminate them from sentences of the form:

$\forall u_1 \ \forall u_2 \ ... \ \forall u_n \ \exists w \ X$

belonging to S.

Such a sentence can be replaced by the new sentence

$\forall u_1 \ \forall u_2 \ ... \ \forall u_n \ X'$

where X' is obtained from X by replacing all free occurrences of w in X by the term $f(u_1,u_2...,u_n)$ where f is a function symbol which does not occur in S.

If n=0 the term $f(u_1,u_2...,u_n)$ reduces to a constant symbol.

In order to transform sentences belonging to S into correct form, it is useful to move universal quantifiers inside conjunctions

$\forall u \ [X \ \& \ Y] \quad \leftrightarrow \quad \forall u \ X \ \& \ \forall u \ Y$

Repeated applications of the preceding rules will convert any conjunction of sentences in standard form into a conjunction of sentences, each of which has a form:

$\forall u_1 \ \forall u_2 \ ... \ \forall u_k \ [A1 \vee ... \vee Am \vee \neg B1 \vee ... \vee \neg Bn]$

Rename the variables so that different clauses contain different variables, and drop the quantifiers (all variables are universally quantified)

$A1 \vee ... \vee Am \vee \neg B1 \vee ... \vee \neg Bn$

which is equivalent to a clause

$A1,...,Am \leftarrow B1,...,Bn$

Example

Convert the following expressions to clause form

1) Everyone makes mistakes

$\forall x \exists y$ [human(x) $\rightarrow$ does(x, y) & mistake(y)]

Solution
does(x1, m(x1)) $\leftarrow$ human(x1)
mistake(m(x2)) $\leftarrow$ human(x2)

Exercise
If a person x is the parent of another person y, or a parent of a predecessor of y then x is a predecessor of y

RESOLUTION
The resolution rule of inference

The *chain rule(resolution rule)* is ((P $\rightarrow$ Q) and (Q $\rightarrow$ R)) |-- (P $\rightarrow$ R)
The equivalent (clause form) is ((R $\leftarrow$ Q) and (Q $\leftarrow$ P)) |-- (R $\leftarrow$ P)

and may be written as:

(R $\leftarrow$ Q)
(Q $\leftarrow$ P)
(R $\leftarrow$ P)

There is an apparent cancellation of the Q from the left of $\leftarrow$ with the Q from the right of $\leftarrow$.

The new clause (R $\leftarrow$ P) is called the *resolvent* of the clauses (R $\leftarrow$ Q) and (Q $\leftarrow$ P).

In general

clause1:	A1,...,Am	$\leftarrow$	B1,...,Bn,X
clause2:	C1,...,Cp,X	$\leftarrow$	D1,...,Dq
resolvent:	A1,...,Am,C1,...,Cp	$\leftarrow$	B1,...,Bn,D1,...,Dq

Let us consider the two clauses

$\leftarrow$ X (i.e. $\neg$X) and X $\leftarrow$ (i.e. X)

that cannot be true in the same time, and apply the resolution rule:

$\leftarrow$ X
X $\leftarrow$
$\square$

By resolving these two clauses one gets the empty clause $\square$.

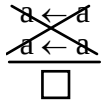
Therefore, whenever $\square$ results from resolution, it means that the initial clauses contain a contradiction.

Notice that only 1 term could be cancelled when resolving two clauses. *Canceling more than one term is not correct.*

This is illustrated by the following examples.

Let us consider the two identical clauses "(a $\leftarrow$ a)" and "(a $\leftarrow$ a)". Obviously they are not contradictory and they can be correctly resolved in one of the following two ways:

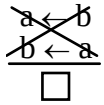
It is not correct to cancel all the terms. Doing so would result in a contradiction, although the two clauses are not contradictory:



This resolution is wrong

As another example consider the clauses "(a ← b)" and "(b ← a)" that are also not contradictory. Indeed, such two clauses are true for all equivalent propositions: a<-->b.
The two clauses can be correctly resolved in one of the following two ways:

Canceling both "a" and "b" would produce a contradiction, which is a wrong result:



This resolution is wrong

The resolution method
The resolution method is a method for demonstrating that a certain logical expression (a theorem) follows from a set of axioms.
If a theorem does follow from its axioms then the axioms and the negation of the theorem cannot all be true. Therefore, the axioms and the negated theorem must lead to a contradiction.
The resolution method is a form of proof by contradiction that involves producing new clauses, called resolvents, from the union of the axioms and the negated theorem, by using the resolution rule. These resolvents are then added to the set of clauses from which they were derived, and new resolvents are derived. This process continues, recursively, until it produces a contradiction. *Resolution is guaranteed to produce a contradiction if the theorem follows from the axioms.*

i). Resolution in propositional logic
In the propositional logic there are no variables, predicates, quantifiers or functions. There are only propositions that may be true or false.

For instance:
P may represent the fact "It is raining", and
Q may represent the fact "The streets are wet".
In this case Q← P represents "The streets are wet If it is raining".

Algorithm: Propositional resolution
Let F be a set of axioms and P the theorem to prove.

1. Convert all the propositions of F and ¬P to clause form.

2. Repeat until either a contradiction is found or no progress can be made

(i.e. no new clauses may be generated):

(a) Select two clauses

(b) Resolve them together, by applying the resolution rule.

(c) If the resulting clause (the resolvent) is the empty clause (i.e. contains no element), then a contradiction has been found. Otherwise, add it to the set of clauses available to the algorithm.

Example

Consider the following axioms:

$(A \rightarrow C \vee D) \ \& \ (A \vee D \vee E) \ \& \ (A \rightarrow \neg C)$

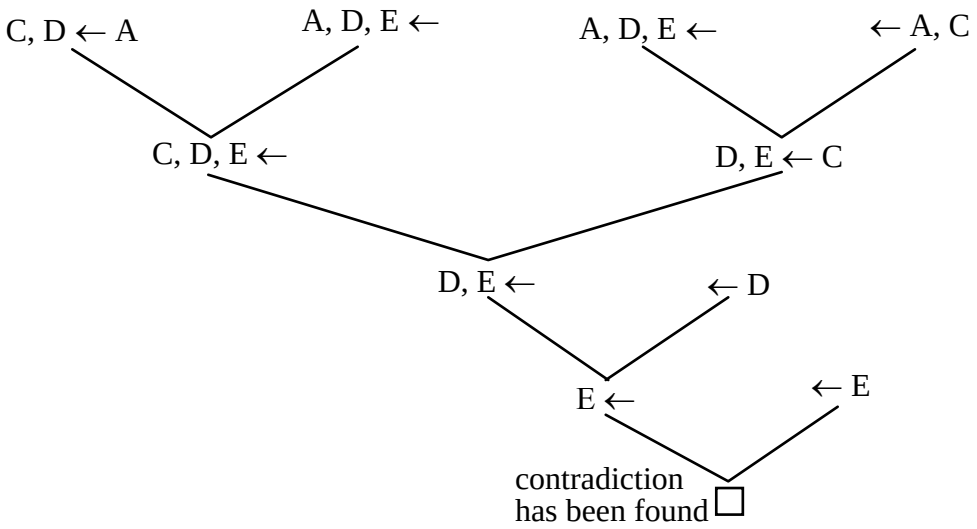
Prove that $(D \vee E)$ follows from them.

Solution

1. Convert the axioms and the negation of the theorem to clause form:

C, D ← A
A, D, E ←
← A, C
← D
← E

2. Resolve the clauses to produce the following resolution tree:



ii). Resolution in predicate calculus

a) Substitution and unification

A *substitution* has the following form: $\sigma = \{x_1=t_1, \ldots , x_n=t_n\}$

Each x_i (i=1,...,n) is a variable and each t_i (i=1,...,n) is a term. All the variables x_i are distinct.

A substitution can be regarded as a function which maps variables to terms.

If E is an expression (term, atom, or clause) then the result of applying the substitution $\theta = \{x_1=t_1, \ldots , x_n=t_n\}$ to E is an expression $E\theta$.

$E\theta$ is identical to E except that for every component $x_i=t_i$ which belongs to θ, whenever E contains an occurrence of x_i, $E\theta$ contains an occurrence of t_i.

The new expression Eθ is said to be an instance of E.

A substitution σ unifies the two expressions E_1 and E_2 if it makes them identical, i.e.

$$E_1\sigma = E_2\sigma$$

$E_1\sigma$ is the common instance of E_1 and E_2 determined by σ.

Example
Consider the following expressions:

E_1 =P(f(a),x)

E_2 =P(y, g(z))

A substitution that unifies these expressions (atoms) is σ = {x=g(b), y=f(a), z=b}.
Indeed, $E_1\sigma = E_2\sigma$ = P(f(a), g(b))

One says that θ *matches* E_1 and E_2 (or, that θ is *the most general unifier* of E_1 and E_2) if

1) θ unifies E_1 and E_2 and
2) The common instance E1σ determined by any other unifier σ of E_1 and E_2 is an instance of the common instance $E_1\theta$ determined by θ. Thus $E_1\sigma = (E_1\theta)\lambda$ for some substitution λ.

Example
Let us consider again the expressions:

E_1 =P(f(a),x)

E_2 =P(y, g(z))

The most general unifier is θ = {x=g(z), y=f(a)}

$E_1\theta = E_2\theta$ = P(f(a), g(z))

Another unifier is σ = {x=g(b), y=f(a), z=b}

$E_1\sigma = E_2\sigma$ = P(f(a), g(b))

One can see that

$E_1\sigma = (E_1\theta)\lambda$

where λ = {z=b}

b) A unification algorithm
Let A1 and A2 be two atoms to be unified (matched).
Let θ = {}.

We regard A1 and A2 as being two strings of symbols and move left-to-right examining the corresponding symbols until a "disagreement" is found. The terms in this position form a disagreement set.

If none of the terms in the disagreement set consists of a variable by itself, we give up because the set of atoms cannot be unified by any substitution.

Otherwise, we convert the variable into a term by adding an element of the form vp=tp to the substitution, where vp is the variable from the disagreement set, and tp is the corresponding term.

To add an element vp=tp to a substitution θ = {v1=t1, v2=t2,...,vk=tk} it must be the case that vp is not equal to any vi, i = 1,...,k.

We first apply the substitution {vp=tp} to each ti, i=1,...,k, and to A1, A2, and then we insert it into the resulting set.

We continue to add such substitutions and simultaneously perform them on the atoms until either the disagreement set is no longer a disagreement set, or no variables remain (in this case we also give up).

Once the disagreement has been taken care of, symbol examination is resumed (including the examination of symbols recently inserted by substitution).

When and if the matching reaches the right end of all the literals, a most general unifier θ has been found.

Note
The algorithm could easily be extended to find the most general unifier of a set of atoms A1,...,An.

Example

A1 = G(x, f(y))
A2 = G(a, f(g(z))) θ = {}

A1 = G(x, f(y))
A2 = G(a, f(g(z))) θ = {x=a}
↑

A1 = G(a, f(y))
A2 = G(a, f(g(z))) θ = {x=a, y=g(z)}
↑

A1 = G(a, f(g(z)))
A2 = G(a, f(g(z))) θ = {x=a, y=g(z)}
↑

c) Resolution of predicate calculus expressions
In applying the resolution procedure to predicate calculus expressions, one has first to unify the corresponding atoms from the clauses to be resolved.

Suppose we have the clauses:
R(a) ← Q(a)
Q(x) ← P(x)

These two clauses could be resolved if we could unify Q(a) in the first clause with Q(x) in the second. The unifier is θ = {x=a}. By applying this unifier to the two clauses they become:
R(a) ← Q(a)
Q(a) ← P(a)
and their resolvent is
R(a) ← P(a)

This shows that, when applying the resolution rule to clauses that contain predicates with variables and functions, one has to first unify the clauses and then to resolve them.

Example

$P(f(a), x), Q(x) \leftarrow$
$R(g(b)) \leftarrow P(y, g(b))$

After unification of P in these two clauses, one obtains
$P(f(a), g(b)), Q(g(b)) \leftarrow$
$R(g(b)) \leftarrow P(f(a), g(b))$
and their resolvent is
$R(g(b)), Q(g(b)) \leftarrow$

Example
Let us consider again the example from section 2.

Given: 1. 1. man(Marcus)
2. Pompeian(Marcus)
3. $\forall x$ (Pompeian(x) $\rightarrow$ Roman(x))
4. ruler(Caesar)
5. $\forall x$ (Roman(x) $\rightarrow$ loyalto(x,Caesar) $\vee$ hate(x,Caesar))
6. $\forall x \exists y$ (person(x) $\rightarrow$ person(y) & loyalto(x,y))
7. $\forall x \forall y$ (person(x) & ruler(y) & tryassassinate(x,y) $\rightarrow \neg$loyalto(x,y))
8. tryassassinate(Marcus,Caesar)
9. $\forall x$ (man(x) $\rightarrow$ person(x))

Prove that: $\neg$loyalto(Marcus, Caesar)

First the axioms and the negation of the theorem have to be converted to clause form:

The axioms 1, 2, 4, 8 are already in clause form.
The other axioms are converted as follows:

3. $\forall x$ (Pompeian(x) $\rightarrow$ Roman(x))

Roman(x1) $\leftarrow$ Pompeian(x1)

5. $\forall x$ (Roman(x) $\rightarrow$ loyalto(x,Caesar) $\vee$ hate(x,Caesar))

loyalto(x2,Caesar), hate(x2,Caesar) $\leftarrow$ Roman(x2)

6. $\forall x \exists y$ person(x) $\rightarrow$ person(y) & loyalto(x,y)

Eliminate $\rightarrow$:
$\forall x \exists y$ ($\neg$person(x) $\vee$ (person(y) & loyalto(x,y)))

Move disjunction inside the sentence:
$\forall x \exists y$ (($\neg$person(x) $\vee$ person(y)) & ($\neg$person(x) $\vee$ loyalto(x,y)))

Eliminate existential quantifier:

$\forall x ((\neg person(x) \vee person(f1(x))) \& (\neg person(x) \vee loyalto(x,f1(x))))$

Move universal quantifier inside conjuncts:

$\forall x (\neg person(x) \vee person(f1(x))) \&$
$\forall x (\neg person(x) \vee loyalto(x,f1(x)))$

Rename variables and drop the universal quantifiers:

$(\neg person(x31) \vee person(f1(x31)))$
$(\neg person(x32) \vee loyalto(x32,f1(x32)))$

Write in clause form:

$person(f1(x31)) \leftarrow person(x31)$

$loyalto(x32,f1(x32)) \leftarrow person(x32)$

7. $\forall x \forall y (person(x) \& ruler(y) \& tryassassinate(x,y) \rightarrow \neg loyalto(x,y))$

Eliminate $\rightarrow$:

$\forall x \forall y (\neg(person(x) \& ruler(y) \& tryassassinate(x,y)) \vee \neg loyalto(x,y))$

Reduce the scope of $\neg$ to a single term:

$\forall x \forall y (\neg person(x) \vee \neg ruler(y) \vee \neg tryassassinate(x,y) \vee \neg loyalto(x,y))$

Rename variables and drop the universal quantifiers:

$\neg person(x4) \vee \neg ruler(y1) \vee \neg tryassassinate(x4,y1) \vee \neg loyalto(x4,y1)$

Write in clause form:

$\leftarrow person(x4), ruler(y1), tryassassinate(x4,y1), loyalto(x4,y1)$

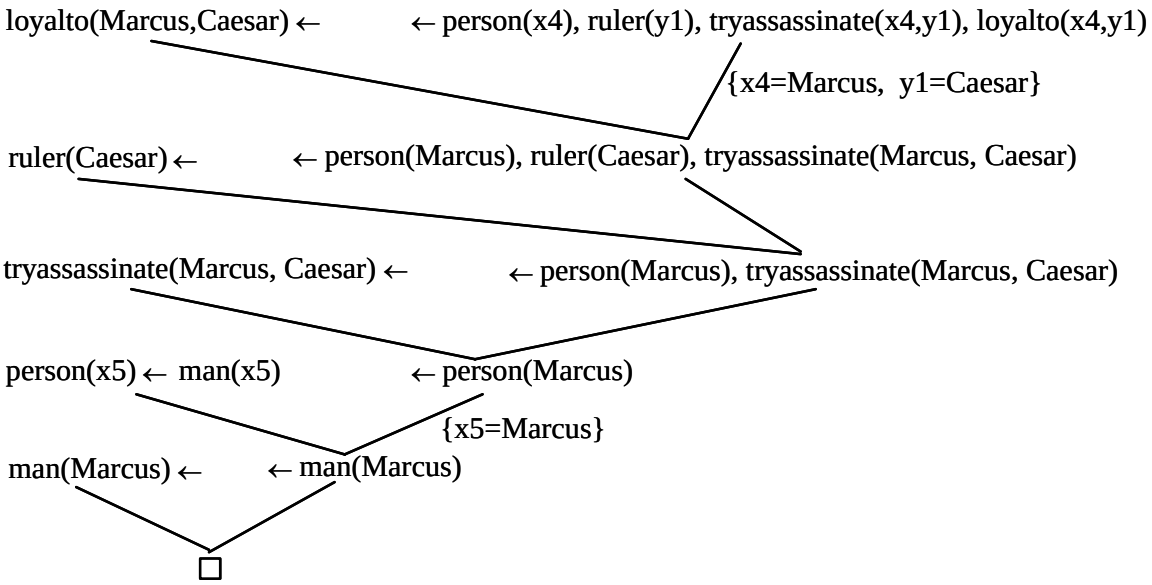
9. $\forall x (man(x) \rightarrow person(x))$

$person(x5) \leftarrow man(x5)$

The negation of the theorem is already in clause form:

loyalto(Marcus, Caesar)

The following is a proof by resolution:



The resolution is *complete* for first-order logic, i.e. can prove all the theorems of it.
The resolution is *sound*, i.e. will not indicate that nontheorems are true.
It can be extremely time-consuming.

d) Strategies for improving the efficiency of resolution

Several strategies have been proposed to minimize the number of the clauses to which the resolution rule is applied.

Set-of-support strategy
At least one parent of each resolvent is chosen from the negation of the theorem or from the set of clauses that are derived from it.
This strategy restricts the number of clauses that can be resolved at any given time and is usually more efficient than a breadth-first search.

Linear-input-form strategy
Always one parent of each resolvent is chosen from the set of the original clauses.
It is more efficient than the previous strategy but it is no longer complete. That is, there are cases in which it will not find a contradiction when one exists.