Proposition. For a finite set of sentences KB and a sentences α, KB $\vDash \alpha$ iff $\vDash$ KB $\to \alpha$ iff KB $\cup\{\neg\alpha\}$ is unsatisfiable iff KB $\cup\{\neg\alpha\} \vDash \bot$. Here α is also called a **query**.

As shown in this proposition, we can check an entailment via a check of unsatisfiability, which can be done by a single inference rule called resolution(though check for satisfiability is NP-complete).

Definition. A literal is an atomic formula or the negation of an atomic formula. A clause is a disjunction of literals.

Since the definition of atomic formulas differs from propositional logic to first-order logic, the definition of literals and clauses also varies.

Definition. (Conjunctive Normal Form, CNF)
A formula is said to be in conjunctive normal form iff it's a conjuncture of clauses. Such formulas are also called CNFs.

Note that a CNF in first-order logic may have free variables and thus is not a sentence.

1 PROPOSITIONAL CASE

1.1 Preliminaries

Proposition. Every propositional formula is logically equivalent to a CNF.

Proof. Note that

$$\bigvee_{j=1}^{n}(l_{j,1}\wedge l_{j,2}\wedge\cdots\wedge l_{j,a_j}) \equiv \bigwedge_{i_1,i_2,\ldots,i_n}\bigvee_{j=1}^{n} l_{j,i_j}$$

Use the structural induction($\{\neg,\vee,\wedge\}$ is complete):
base case: Any atom is a literal and thus a CNF
inductive case: suppose $p = c_1\wedge c_2\wedge\cdots\wedge c_n, q$ are two CNFs, then $p\wedge q$ and $p\vee q$ are CNFs, and by moving $\neg$ inward,

$$\neg p \equiv \bigvee_{j=1}^{n}(\neg l_{j,1}\wedge\neg l_{j,2}\wedge\ldots\neg\wedge l_{j,a_j}) \equiv \bigwedge_{i_1,i_2,\ldots,i_n}\bigvee_{j=1}^{n}\neg l_{j,i_j}$$

is a CNF. □

In implementation, we can store propositional formula in a binary tree and sink $\vee$ downward as much as possible (we can either parse formulas by ourselves or using grammar parsing tools like `pyparsing`).

Definition. (Clausal Form)
The clausal form of a CNF is a notation where a clause is represented as a set of literals occurring in it surrounded by square brackets []($\square$)(so identical literals will be reduced to only one; this process is a special case of **factoring**). An empty [], called an empty clause, is used to represent a contradiction.
The clausal form of a set of clauses or a CNF is a set of the clausal forms of each clause(Note that a CNF can actually be viewed as a set of clauses). An empty {} is used to present an empty set of clauses; i.e. a tautology.

For example, the clausal form of the clause $x \vee f(a)$ is $[x, f(a)]$, and the clausal form of a CNF $(x \vee f(a)) \wedge (\neg y \vee g(b))$ is $\{[x, f(a)], [\neg y, g(b)]\}$.

1.2 Resolution

Definition. (Resolution for Propositional Logic)
For two clauses c_1, c_2 and a literal ρ,

$$\{c_1 \vee \rho, c_2 \vee \neg\rho\} \vDash c_1 \vee c_2$$

or in clausal form

$$\{c_1 \cup \{\rho\}, c_2 \cup \{\neg\rho\}\} \vDash c_1 \cup c_2$$

where $c_1 \vee c_2$ is called a **resolvent** of the two input clauses c_1, c_2.

Note. In this section, the symbol $\vdash$ means a deduction only using the resolution rule.

Definition. (Resolution Closure)
The resolution closure of a set of clauses S, denoted as $RC(S)$, is the set of all clauses derivable by repeated application of the resolution rule to clauses in or their derivatives.

Properties.

(1) Any two of the input clauses and the resolvent can determine the rest element.

(2) Resolution is refutation-sound and refutation-complete; i.e. For a set of clauses S where there are only finitely many distinct atoms, $S \vDash \square$ iff $S \vdash \square$. An informal proof of this can be seen in [1].

(3) For a set of clauses S with only finitely many symbols, checking $S \vdash \square$ is of finite steps: each clause added to the set is a resolvent of previous clauses, and so contains only literals mentioned in the original set S. There are only finitely many clauses with just these literals.

To use resolution for logical entailment in KB, we first need to convert KB to CNF once and when we want to add new facts to KB, we add clauses in it without affecting others. In this way, the KB is more extensible and easier to manipulate.

2 FIRST-ORDER CASE

In this note, constants in first-order logic are viewed as 0-ary functions.

2.1 Preliminaries

As we can see in the propositional case, the premises of resolution rule are two CNFs. But in first-order logic, quantifiers are introduced. To express FOL formulas in a similar form of clauses in propositional logic, we need t o do the so-called skolemization.

The generalization of CNF to first-order logic is the following:

Definition. (Prenex Conjunctive Normal Form, PCNF)
A first-order logic formula is in prenex conjunctive normal form(PCNF) iff it's of the form

$$Q_1 x_1 \dots Q_n x_n M$$

where the Q_i are quantifiers and M is a quantifier-free formula in CNF. The sequence $Q_1 x_1 \dots Q_n x_n$ is called the **prefix** and M is called the **matrix**.

Note that according to this definition, M can contain free variables.

The clausal form of a **closed** PCNF with only universal quantifiers and a matrix M is a set of clausal forms of the clauses of M; e.g. the clausal form of

$$\forall y \forall z([f(y) \vee z] \wedge [\neg y \vee g(z)])$$

is $\{[f(y), z], [\neg y, g(z)]\}$. Thus, **a set of clauses is implicitly universally quantified**.
Note that the notation here is a bit ambiguous; in that when we use curly brackets without square brackets, curly brackets mean disjunction, but when we denote CNF, the (outer) curly brackets denote conjunction while the square brackets denote disjunction. That is, $\{a, b\}$ means $a \vee b$ while $\{[a, b], [c]\}$ means $(a \vee b) \wedge c$.

Theorem. (Skolem)
Let A be a closed formula of FOL. Then there exists a PCNF A' s.t. A is satisfiable iff A' is satisfiable.

Therefore, for a set of clauses S(like $KB \cup \{\neg\alpha\}$), to check the satisfiability of S is converted to check the satisfiability of the PCNF of S.

Algorithms that transform A into A' can be found on chapter 9 of [2]. A basic one containing the following steps:

- Rename bound variables so that no variable appears in two quantifiers.
- Eliminate all binary Boolean operators other than $\vee$ and $\wedge$.

- push $\neg$ inward, collapsing multiple negations, until they apply to atomic formulas only.
- Extract quantifiers from the matrix. Always choose an outermost quantifier first.
- Use the distributive laws to transform the matrix into CNF. The formula is now in PCNF.
- a process called *Skolemization*, which eliminates existential quantifiers by introducing new function symbols called *Skolem functions*, making sure that dependencies of existential variables on universal variables are reflected in the Skolem functions argument structure. For example, the clausal form of $\forall x \exists y P(x, y)$ should be something like $[P(x, f(x))]$.

Finally, the formula can be written in clausal form by dropping the (universal)quantifiers and writing the matrix as sets of clauses.

To better illustrate the resolution, we need the concept of substitution:

Definition. (Substitution)
A substituion of terms for variables is a set:

$$\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$$

where each x_i is a distinct variable and each t_i is an arbitrary term. The empty substitution is the empty set.

Definition. (Substitution Instance)
An expression is a term, a literal, a clause or a set of clauses. Let E be an expression and let θ be a substitution. An instance $E\theta$ of E is obtained by *simultaneously* replacing each ocurrence of x_i in E by t_i.

Also note that for two terms t_1 and t_2 and a substitution θ, the following formulas are valid:

$$\begin{aligned}
(\neg t_1)\theta &= \neg(t_1\theta) \\
(t_1 \wedge t_2)\theta &= t_1\theta \wedge t_2\theta \\
(t_1 \vee t_2)\theta &= t_1\theta \vee t_2\theta \\
(t_1 \rightarrow t_2)\theta &= (t_1\theta) \rightarrow (t_2)\theta
\end{aligned}$$

So it still makes sense to use the "ambiguous" notation like $\neg t_1\theta$(the priority of substitution or negation doesn't matter).

Example. (p187 of [2])
Consider the expression $E = \{p(x), q(f(y))\}$ and a substitution $\theta = \{x \leftarrow y, y \leftarrow f(a)\}$. The instance obtained by performing the substitution is:

$$E\theta = \{p(y), q(f(f(a)))\}$$

The word *simultaneously* in the above definition means that one does not substitute y for x in E to obtain:

$$\{p(y), q(f(y))\}$$

and then substitute $f(a)$ for y to obtain:

$$\{p(f(a)), q(f(f(a)))\}$$

Definition. (Unifier)
For a set of literals $\{\rho_1, \rho_2, \ldots, \rho_n\}$, a unifier θ is a substitution s.t.

$$\rho_1\theta = \rho_2\theta = \cdots = \rho_n\theta$$

is valid.

2.2 Resolution

For a set of literals $L = \{l_1, \ldots, l_n\}$, let L^c denote $\{\neg l_1, \ldots, \neg l_n\}$. We have the following so-called general resolution:

Theorem. (General resolution)
Let c_1, c_2 be clauses with no variables in common. Suppose $L_1 = \{\rho_1, \rho_2, \ldots, \rho_m\} \subset c_1$ and $L_2 = \{\sigma_1, \sigma_2, \ldots, \sigma_n\} \subset c_2$ be subsets of literals such that $L_1\theta = L_2^c\theta$ is valid for a unifier $\theta = \{x_i \leftarrow t_i \, (i = 1, 2, \ldots, d)\}$. Then the clause $(c_1\theta - L_1\theta) \cup (c_2\theta - L_2\theta)$ can be inferred; to express this in terms of first-order formulas, resolution is a rule of inference such that

$$\frac{\forall_1.c_1, \forall_2.c_2}{\forall_3(c_1\theta - L_1\theta) \cup (c_2\theta - L_2\theta)}$$

$\forall_1$, $\forall_2$, $\forall_3$ are sequences of universal quantifiers, where $\forall_3$ involves all the remaining variables in $(c_1\theta - L_1\theta) \cup (c_2\theta - L_2\theta)$.

Though two sets of literals are considered in the general description of resolution, one literal per cause is commonly used in practice (this is called **narrow resolution**). Some examples can be seen in p58 of [3].

Properties.

(1) Even for a set of clauses with finite symbols, the first-order resolution procedure might not terminate; for example, suppose P is a unary predicate, f a function symbol, a a constant, and consider $S = \{[P(f(x)), \neg P(x)], [P(a)]\}$. Currently there is no general way to detect all such infinite branches.

(2) first-order resolution is also refutation-complete and refutation-sound.

2.3 Factoring

This is a process of simplifying a clause obtained by resolution via unifying variables occurring in itself. For example, we can simplify

$$P(x, y) \vee P(u, v)$$

to

$$P(x, y)$$

Generally speaking, if we only use narrow resolution, we must need to use factoring. An example demonstrating its necessity and more details can be found in [4].

2.4 Fill-in-the-Blank Questions

A fill-in-the-blank question is a FOL atom with the desired variables existentially quantified. For example, for the question "Who is the parent of John?", its question form would be $\exists x.P(x)$. To obtain what x is while doing resolution, we could form a disjunction from $\neg\exists x.P(x) \equiv \forall x.\neg P(x)$ and the literal $answer(x)$. In this way, when the resolution generates only a literal of $answer$, the term it contains is the answer.
This process can be extended for multiple variables.

3 RESOLUTION INTRACTABILITY

Definition. (Resolution Derivation)
A Resolution derivation of a clause c from a set of clauses S is a sequence of clauses $c_1, \ldots, c_n$, where the last clause, c_n, is c, and where each c_i is either an element of S or a resolvent of two earlier clauses in the derivation.

Even for propositional case, Armin Haken has proved that there are unsatisfiable propositional clauses $c_1, \ldots, c_n$ such that the shortest derivation of the empty clause has on the order of 2^n steps.

3.1 Most General Unifiers

The most important way of avoiding needless search in a first-order derivation is to keep the search as general as possible. For example, consider two PCNF formulas $\forall x(P(f(x)) \vee c_1(x))$ and $\forall y \neg P(y)$(P is a predicate and f is a function). The literals $P(f(x))$ and $P(y)$ can be unified either by

$$\theta_1 = \{x \leftarrow a, y \leftarrow f(a)\}$$

or

$$\theta_2 = \{y \leftarrow f(x)\}$$

(a is a constant)
But their resolvent is different; the resolvent for θ_1 is $c_1(a)$, for θ_2 is $\forall x c_1(x)$.

Now if the KB = $\{\forall x(P(f(x)) \vee c_1(x))\}$ and the query is $c_1(b) \vee \exists y P(y)$($b$ is another constant), we can obtain the empty clause with one more resolution using θ_2, but not for θ_1. Actually, when a stupid algorithm finds that θ_1 doesn't work, it may try another substitutions with a different constant, which will never work either. The problem is that these substitutions are too specific.

Definition. (Composition of Substitutions)
Let

$$\theta = \{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$$
$$\sigma = \{y_1 \leftarrow s_1, \ldots, y_k \leftarrow s_k\}$$

be two substitutions and let $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_k\}$ be the sets of variables substituted for in θ and σ , respectively. The composition of θ and σ, $\theta\sigma$, is a substitution such that

$$\theta\sigma = \{x_i \leftarrow t_i\sigma \,|\, x_i \in X\} \cup \{y_j \leftarrow s_j \,|\, y_j \in Y, y_j \notin X\}$$

Definition. (Most General Unifier)
For a set of literals L, a most general unifier(MGU) for L is a unifier μ such that any unifier θ of L can be expressed as

$$\theta = \mu\lambda$$

for some substitution λ.

Example.
Consider two atoms $P(A, y)$ and $P(x, y)$; $\theta = \{x \leftarrow A, y \leftarrow B\}$ is a unifier and $\mu = \{x \leftarrow A, y \leftarrow y\}$ is the mgu. Then

$$\{x \leftarrow A, y \leftarrow B\} = \{x \leftarrow A, y \leftarrow y\}\{y \leftarrow B\}$$

while μ can't be expressed as the form of $\theta\lambda$, since B is a constant and can never be substituted back to a variable.

The key fact is that we can limit the resolution rule to MGUs without loss of refutation completeness. And there are linear time algorithms for calculating an MGU. A simple algorithm called the Robinson's unification algorithm which takes exponential time to calculate an MGU for two literals ρ_1, ρ_2 is presented below(p72 of [3]):

(Robinson's unification algorithm)

(1) start with $\theta = \{\}$;

(2) exit if $\rho_1\theta = \rho_2\theta$;

(3) calculate the disagreement set DS, the pair of terms at the first place where $\rho_1\theta$ and $\rho_2\theta$ disagree; (e.g. if $\rho_1\theta = P(a, f(g(z)), \ldots))$ and $\rho_2\theta = P(a, f(u)\ldots)$, then $DS = (u, g(z))$)

(4) if DS exactly contains a variable v and a term t not containing v, then set θ to $\theta\{v \leftarrow t\}$, and go to step 2; otherwise, fail.

4 Paramodulation

For formuals involving equality, applying the axioms of equality is slow and we need to avoid infinite intermediate results (= is reflexive). Instead of using them, we can only use one rule called **paramodulation** to deal with this case. The paramodulation says that for two clauses c_1 and c_2,

$$\frac{c_1 \cup \{t = s\}, c_2 \cup \{\rho[t']\}}{(c_1 \cup c_2 \cup \rho[s])\theta}$$

where t, s, t' are terms and $t\theta = t'\theta$.
Let's prove it:

1.	$c_1 \cup \{t = s\}$	premise
2.	$c_2 \cup \{\rho[t']\}$	premise
3.	$t\theta = t'\theta$	premise
4.	$c_1\theta \cup \{t\theta = s\theta\}$	$\forall_e, 1$
5.	$c_2\theta \cup \{\rho[t'\theta]\}$	$\forall_e, 2$
6.	$t\theta = s\theta$	assumption
7.	$t'\theta = s\theta$	$=, 3, 6$
8.	$c_2\theta \cup \{\rho[s\theta]\}$	$=_e, 5, 7$
9.	$(c_1 \cup c_2 \cup \rho[s])\theta$	$\vee_i, 8$
10.	$c_1\theta$	assumption
11.	$(c_1 \cup c_2 \cup \rho[s])\theta$	$\vee_i, 10$
12.	$(c_1 \cup c_2 \cup \rho[s])\theta$	$\exists_e, 4, 6-9, 10-11$

Furthermore, it can be showed that by using only paramodulation, we can infer all the axioms of equality.

References

[1] Russell, Stuart J., and Peter Norvig. *Artificial intelligence: a modern approach.* Malaysia; Pearson Education Limited,, 2016. p256

[2] Ben-Ari, Mordechai. *Mathematical logic for computer science.* Springer Science & Business Media, 2012.

[3] Levesque, Hector J., and Ronald, Brachman J.. *Knowledge representation and reasoning.* Elsevier, 2004.

[4] Schubert, Lenhart. Lecture note 7 for 2019 Fall CSC444: Knowledge Representation.