

**Proposition.** For a finite set of sentences KB and a sentences  $\alpha$ ,  $KB \models \alpha$  iff  $\models KB \rightarrow \alpha$  iff  $KB \cup \{\alpha\}$  is unsatisfiable iff  $KB \cup \{\alpha\} \not\models \perp$ . Here  $\alpha$  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

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

Use the structural induction( $\{\neg, \vee, \wedge\}$  is complete):

base case: Any atom is a literal and thus a CNF

inductive case: suppose  $p = c_1 \wedge c_2 \wedge \dots \wedge c_n$ ,  $q$  are two CNFs, then  $p \wedge q$  and  $p \vee q$  are CNFs, and by moving  $\neg$  inward,

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

is a CNF. □

**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  $[]$ (so identical literals will be reduced to only one; this process is called **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  $\rho$ ,

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

or in clausal form

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

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,  $S \models \square$  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 LOGIC 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, and things become more complex: the general resolution is applied for the so-called PCNFs with only universal quantifiers.

**Definition.** (Prenex Conjunctive Normal Form, PCNF)

A first order logic formula is in prenex conjunctive normal form(PCNF) iff it's of the form

$$Q_1x_1 \dots Q_nx_nM$$

where the  $Q_i$  are quantifiers and  $M$  is a quantifier-free formula in CNF. The sequence  $Q_1x_1 \dots Q_nx_n$  is called the **prefix** and  $M$  is called the **matrix**.

The clausal form of a 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**.

**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*.

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 substitution of terms for variables is a set:

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

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  $\theta$  be a substitution. An instance  $E\theta$  of  $E$  is obtained by *simultaneously* replacing each occurrence of  $x_i$  in  $E$  by  $t_i$ .

Also note that for two terms  $t_1$  and  $t_2$  and a substitution  $\theta$ , 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  $\{\rho_1, \rho_2, \dots, \rho_n\}$ , a unifier  $\theta$  is a substitution s.t.

$$\rho_1\theta = \rho_2\theta = \dots = \rho_n\theta$$

is valid.

## 2.2 Resolution

we have the following so-called general resolution:

**Theorem.** (General resolution)

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

$$\frac{\{\forall_1(c_1 \vee \rho_1 \vee \dots \vee \rho_m), \forall_2(c_2 \vee \sigma_1 \vee \dots \vee \sigma_n)\}}{\forall_3(\{c_1 \vee c_2\}\theta)}$$

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

Though two sets of literals are considered in the general description of resolution, two literals are commonly used in practice. 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.

## 3 RESOLUTION INTRACTABILITY

**Definition.** (Resolution Derivation)

A Resolution derivation of a clause  $c$  from a set of clauses  $S$  is a sequence of clauses  $c_1, \dots, c_n$ , 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  $c_1, \dots, c_n$  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  $\theta_1$  is  $c_1(a)$ , for  $\theta_2$  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  $\theta_2$ , but not for  $\theta_1$ . Actually, when a stupid algorithm finds that  $\theta_1$  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

$$\begin{aligned}\theta &= \{x_1 \leftarrow t_1, \dots, x_n \leftarrow t_n\} \\ \sigma &= \{y_1 \leftarrow s_1, \dots, y_k \leftarrow s_k\}\end{aligned}$$

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

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

**Definition.** (Most General Unifier)

For a set of literals  $L$ , a most general unifier(MGU) for  $L$  is a unifier  $\mu$  such that any unifier  $\theta$  of  $L$  can be expressed as

$$\theta = \mu\lambda$$

for some substitution  $\lambda$ .

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  $\rho_1, \rho_2$  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  $\rho_1\theta = P(a, f(g(z), \dots))$  and  $\rho_2\theta = P(a, f(u) \dots)$ , then  $DS = (u, g(z))$ )
- (4) if  $DS$  exactly contains a variable  $v$  and a term  $t$  not containing  $v$ , then set  $\theta$  to  $\theta\{v \leftarrow t\}$ , and go to step 2; otherwise, fail.

## 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.