A Unit Resolution Approach to Knowledge Compilation

Arindama Singh and Manoj K Raut
Department of Mathematics
Indian Institute of Technology
Chennai-600036, India

Abstract : Knowledge compilation deals with the computational intractability of reasoning problems. To overcome this difficulty we provide three different compilation approaches to a first order knowledge base. The knowledge base Σ is preprocessed by unit resolution into an approximate knowledge base CKB(Σ) from which a subset of possible queries can be answered by unit refutation. The number of clauses obtained by each of the three methods is less than the number of clauses in the prime implicate set.

Keywords : First Order Logic, Resolution Restriction, Horn Clauses

1. Introduction

Answering a query in propositional knowledge base is a central issue in Artificial Intelligence. Determining whether or not a given query follows from a propositional knowledge base is intractable [3] since every algorithm takes exponential time in the worst case. To overcome such computational difficulties the knowledge base is mapped (compiled) during an off-line phase into an appropriate data structure with respect to which inference becomes tractable. The inference process is referred to as the on-line reasoning, and the off-line phase is termed as *knowledge compilation* [2]. Such type of knowledge compilation involves the mapping of Σ into a logical equivalent set of formulas $\Pi(\Sigma)$, the set of *prime implicates/implicants* [6, 7, 10, 11] of Σ, where answering queries with respect to $\Pi(\Sigma)$ is tractable. Another equivalence preserving compilation is given in [4, 9] where Σ is compiled into a logically equivalent database FPI(Σ) and the queries are answered by unit refutation in an on-line phase in polynomial time.

Most of the research works to date in knowledge compilation have been carried out in propositional knowledge base in spite of the higher expressing capability of first order knowledge base. In this paper we describe three knowledge compilation methods to map the given clausal data base Σ by unit resolution to an approximate data base CKB(Σ) with respect to which a subset of possible queries can be answered. These methods extend the corresponding propositional algorithms [1, 5, 8] to first order knowledge base. We will see that $\Sigma \equiv CKB(\Sigma)$, for all the three compilation methods. The compilation involves the derivation of all possible resolutions among clauses exploiting a resolution based prime implicate algorithm in first order knowledge base.

The paper is structured as follows. In Section 2, the definition of two types of restrictions of resolution are described and their relationship with unit resolution is established. All the three compilation methods are described in Section 3. We finally give some concluding remarks in Section 4.

2. Preliminaries

We assume the syntax and semantics of first order logic along with the usual notions of literals, disjunctive clauses, and SCNF or Skolem conjunctive normal form formulas. Two literals s and t are said to be *complementary* to each other iff there exists a substitution ξ(most general unifier) such that $s\xi = \neg t\xi$. Such a substitution is called a complementary substitution. For example, Qyb and $\neg Qax$ are complementary to each other with respect to the substitution $\xi = [y/a, x/b]$. We also refer a clause as *fundamental* if it does not contain a literal and its negation.

Let C_1 and C_2 be two clauses. C_2 is said to be *subsumed* by C_1 iff $C_1\sigma \subseteq C_2$ for a substitution σ and then C_1 subsumes C_2. For example, $\{Px, Qb\}$ subsumes $\{Pa, Qb, Rxc\}$ for the substitution $\sigma = [x/b]$.

A fundamental disjunctive clause C_1 is an *implicate* of a formula Σ iff $\Sigma \models C_1$. A clause C_1 is said to be a prime implicate of Σ if C_1 is said to be an implicate of Σ and there is no other implicate C_2 of Σ such that C_2 subsumes C_1.

Let C_1 and C_2 be two clauses in a knowledge base Σ. If there exists exactly one literal which occurs unnegated in one of C_1 or C_2 and negated in other and s and t be such a pair of complementary literals in C_1 and C_2, respectively with respect to a most general unifier σ then *resolution* of C_1 and C_2 is $C = ((C_1 - \{s\}) \cup (C_2 - \{t\}))\sigma$ which can also be written as $((C_1\sigma - \{l\} \cup (C_2\sigma - \{\neg l\}))$, where $s\sigma = l$ and $t\sigma = \neg l$ for a literal l. If C is obtained by the resolution of C_1 and C_2 with respect to a most general unifier σ then the *resolvent* C is said to be *associated* with σ. By default, each clause C_i in $\Sigma = \{C_1, \ldots, C_n\}$ is associated with the empty substitution ϵ. If C_i and C_j are two resolvent clauses associated with substitutions σ_1 and σ_2, respectively then their resolution with respect to σ is defined *provided* $\sigma_1\sigma = \sigma_2\sigma$. A *unit resolution* is a resolution in which one of the two clauses is a unit clause.

Recall that if C is the resolution of two clauses C_1 and C_2 then $C = ((C_1\sigma - \{l\}) \cup (C_2\sigma - \{\neg l\}))$. Moreover if $u \in C_1\sigma - \{l\}$ and $v \in C_2\sigma - \{\neg l\}$ and for some substitution ξ, $u\xi = v\xi$ then the *merge* literal is a literal which subsumes both u and v. We denote by M the set of all merge literals for all possible ξ. We call C a merge resolvent with M as the set of merge literals. For example, let $C_1 = \{Pxa, Qa, Rz\}$ and $C_2 = \{Pzy, \neg Rb\}$. The merge literal of the resolvent $C = \{Pxa, Qa, Pby\}$ is Pxy.

Let C be the resolution of two clauses C_1 and C_2. If $u_1\sigma \in C_1\sigma - \{l\}$, then the occurrence of $u_1\sigma$ is an *immediate descendant* of $u_1 \in C_1$ and u_1 is an *immediate ancestor* of $u_1\sigma$. Similarly, the definition can also be applied to C_2 instead of C_1. A *resolution deduction* of a clause D from a clausal database Σ is a sequence of clauses $D_1, \ldots, D_m$ such that each D_i is either a member of Σ or a resolvent of D_j and D_k preceding D_i and $D = D_m$. Let 'descendant' be the reflexive transitive closure of the relation 'immediate descendant' in a given resolution deduction.

A resolution deduction is called a *general resolution deduction* if every resolvent is obtained by resolution. A resolution deduction is called a *unit deduction* if every resolvent is obtained by unit resolution. A resolution deduction is said to be *weak no-merge (wnm) deduction* if no resolvent is a merge resolvent. A resolution deduction of a clause C from a set of clauses Σ is said to be *no-merge (nm) deduction* if no merge resolvent has any descendants of its merge literals resolved upon. Let $\Sigma \vdash C$, $\Sigma \vdash_u C$, $\Sigma \vdash_{wnm} C$ and $\Sigma \vdash_{nm} C$ denote respectively, the general, unit, weak no-merge and no-merge deductions. The wnm-deduction does not allow merges whereas no-merge keeps all merge resolvents provided no merge literal is resolved upon later in the deduction. Unit resolution is a special case of both wnm- and nm- resolution as merge does not occur in it.

Theorem 2.1 *Let Σ be a set of clauses and C be any clause. The following two statements are equivalent.*
(i) $\Sigma \vdash_{nm} D$ where $D\eta \subseteq C$, for some substitution η and some clause D
(ii) $\Sigma \cup \neg C \vdash_u \bot$

Proof Let $\Sigma = \{C_1, \ldots, C_k\}$ be the set of clauses. Let $D_1, \ldots, D_n$ be the nm-deduction of a clause D ($= D_n = \{d_1, \ldots, d_m\}$) from the set of clauses Σ. Rewrite the given clause C as $\{d_1^*, \ldots, d_m^*, d_{m+1}^*, \ldots, d_l^*\}$ where $d_1\eta = d_1^*, \ldots, d_m\eta = d_m^*$ for a substitution η so that $D\eta \subseteq C$. Then, $\Sigma \cup \neg C$ becomes $\{C_1, \ldots, C_k, \neg d_1^*, \ldots, \neg d_m^*, \neg d_{m+1}^*, \ldots, \neg d_l^*\}$. We show that $\Sigma \cup \neg C \vdash_u \bot$ by induction on the length n of the nm-deduction.

Suppose D is obtained by resolution in one step from Σ. In particular, let $D = ((C_1\sigma - \{s\}) \cup (C_2\sigma - \{t\}))$ where s and t are two complementary pair of literals. Then taking resolution (in fact, unit resolution) on $\Sigma \cup \neg C$ (where $\neg D\eta \subseteq \neg C$) all the literals of $\neg C$ will be resolved away with clauses of Σ except possibly, s and t. This two literals can be resolved by the substitution σ. Hence it upholds for $n = 1$.

Let it be true for all i such that $1 \leq i \leq n$. To show that it is true for $n + 1$, let the deduction be $D_1, \ldots, D_n, D_{n+1}$ of the clause $D = D_{n+1}$ and for some η, $D\eta \subseteq C$. We want to show that $\Sigma \cup \neg C \vdash_u \bot$. Suppose D is obtained by the resolution of two clauses D_n and D^* by a substitution σ which are associated with substitutions $\sigma_1 \ldots \sigma_n$ and $\eta_1 \ldots \eta_{n'}$ respectively and D^* comes from some other deduction. Let $\{d_1^{'}, \ldots, d_i^{'}, s\} = D_n$ and $\{d_j^{'}, \ldots, d_m^{'}, t\} = D^*$ be such that
$d_1, \ldots, d_i, l \in D_n\sigma$ and $d_j, \ldots, d_m, \neg l \in D^*\sigma$,
where $d_1^{'}\sigma = d_1, \ldots, d_i^{'}\sigma = d_i, d_j^{'}\sigma = d_j, \ldots,$
$d_m^{'}\sigma = d_m, s\sigma = l, t\sigma = \neg l$. Since D_n and D^* are obtained in at most n steps, by the induction hypothesis, $\Sigma \vdash_{nm} D_n$ for some $D_n\sigma \subseteq C_1^{'}$. This implies that $\Sigma \cup \neg C_1^{'} \vdash_u \bot$. And also, $\Sigma \vdash_{nm} D^*$ for some $D^*\sigma \subseteq C_2^{'}$ implies $\Sigma \cup \neg C_2^{'} \vdash_u \bot$. Since $s \in D_n$ and $t \in D^*$ are resolved to get D, s and t must not be merge literals as it is an nm-deduction. So we have exactly two clauses C_p and C_q in Σ such that $l_1 \in C_p$, $l_2 \in C_q$, and $l_1\sigma_1 \ldots \sigma_n = s$, $l_2\eta_1 \ldots \eta_{n'} = t$ which implies

$$l_1\sigma_1 \ldots \sigma_n\sigma = l, \quad l_2\eta_1 \ldots \eta_{n'}\sigma = \neg l \qquad (1)$$

When we take resolution in $\Sigma \cup \neg C$ (where $\neg D\eta \subseteq \neg C$), all the clauses will be resolved away with respect to the substitution $\sigma_1 \ldots \sigma_n \sigma\eta$ and $\eta_1 \ldots \eta_{n'} \sigma\eta$ due to unit refutation of D_n and D^* (by induction) leaving l_1 and l_2 which can then be resolved away with the substitutions $\sigma_1 \ldots \sigma_n \sigma$ and $\eta_1 \ldots \eta_{n'} \sigma$ by (1) to obtain a unit refutation in $\Sigma \cup \neg C$.

Conversely, since every unit refutation is an nm-refutation, $\Sigma \cup \neg C \vdash_u \bot$ implies $\Sigma \cup \neg C \vdash_{nm} \bot$. By soundness of resolution $\Sigma \vdash_{nm} C$. Taking $D = C$ we get $\Sigma \vdash_{nm} D$. This completes the proof. $\square$

Since every wnm-deduction is also an nm-deduction, we have the following:

Corollary 2.2 *Let Σ be a set of clauses and C be any clause. $\Sigma \vdash_{wnm} D$ for some $D \subseteq C$ implies $\Sigma \cup \neg C \vdash_u \bot$.*

We will use the following terminology. A unit (nm-, wnm-) deduction of $\bot$ from a set of clauses Σ is called a *unit (nm-, wnm-) refutation* of Σ. If there is a unit (nm-, wnm-) refutation of Σ, then we call Σ to be *unit (nm-, wnm-) refutable.*

Let Σ be a clausal knowledge base and C be any clause. Then Σ is *unit refutation complete (nm-refutation complete, wnm-refutation complete)* iff for any clause C, $\Sigma \models C$ iff $\Sigma \cup \neg C$ is unit (nm-, wnm-)refutable. we abreviate it to $uc(nmc, wnmc)$. Using Theorem 2.1, the following results can be proved.

Theorem 2.3 *Let Σ be a clausal knowledge base. Then Σ is nm-refutable iff it is wnm-refutable iff it is unit refutable. Moreover, Σ is uc iff it is nmc iff it is wnmc.*

The knowledge base Σ is preprocessed (compiled) into another knowledge base $CKB(\Sigma)$ from which a subset of possible queries can be answered. We compute the prime implicates for first order logic formulas by consensus-subsumption algorithm. Note that, if each clause in a CNF is assumed to be a sentence, i.e, each variable is universally quantified then the resolution principle that resolvent of two clauses is a logical consequence of the previous clause holds. Those implicates are added to Σ *in violation* of the nm or wnm restriction. We get *nmc* and *wnmc* which ensures *uc*.

3. wnm-, nm- resolution and horn compilation

We compute the implicates by a resolution based algorithm. Recall that we can compute an implicate C with respect to a substitution σ from two clauses C_1 and C_2 associated with σ_1 and σ_2 respectively, *provided* $\sigma_1\sigma = \sigma_2\sigma$. The set of clauses obtained by this algorithm is collected in $CKB_1(\Sigma)$, the compiled knowledge base of Σ, which is partially unit refutation complete.

Algorithm(wnmerge)

```
Input: $\Sigma$, the given set of clauses
Output: $CKB_1(\Sigma)$
begin
    $CKB_1 := \Sigma$;
    $\Pi := \Sigma$;
  While two clauses contain a pair of complementary
literals in $CKB_1$
    do
      compute the implicate $C$;
      if $C := \phi$;
        return $CKB_1$
      else
        if $C$ is subsumed by some clause
        in $\Pi$
          continue;
        else
          if $C$ subsumes any clause $D$
          from $\Pi$ and $CKB_1$
            $CKB_1 :=$
            $(CKB_1 - \{D\}) \cup \{C\}$;
            $\Pi = \Pi - \{D\}$;
          endif
          $\Pi = \Pi \cup \{C\}$;
          if $C$ is a merge resolvent
            $CKB_1 := CKB_1 \cup \{C\}$;
          endif
        endif
      endif
    od
    return $CKB_1(\Sigma)$
end
```

Let Σ be the the given set of clauses. Let $L(\Sigma)$ be the set of clauses obtained after the application of one step of the algorithm, i.e, after taking the resolution of two clauses. Similarly we can construct the sequence $\Sigma, L(\Sigma), L(L(\Sigma)), \ldots$, i.e, with $L^{n+1}(\Sigma)$ $= L(L^n(\Sigma))$, for $n \geq 0$ and $L^0(\Sigma) = \Sigma$, and we write $CKB_1(\Sigma) = \cup\{L^i(\Sigma) : i \in \mathbb{N}\}$. Does the sequence $\{L^n(\Sigma)\}$ terminate ? The following example shows that it may not. This is because, the process may not terminate for some inputs. The above algorithm reflects these possibilities in the step 'compute an implicate C'.

Example 3.1 Let $\Sigma = \{\neg Pxy \vee \neg Pyz \vee Pxz,\ \neg Pst \vee \neg Ptu \vee \neg Puw \vee Psw\}$. The consensus closure of both the clauses is infinite. $L(\Sigma) = \{\neg Pxy \vee \neg Pyz \vee Pxz,\ \neg Pst \vee \neg Ptu \vee \neg Puw \vee Psw,\ \neg Psy \vee \neg Pyt \vee \neg Ptu \vee \neg Puw \vee Psw\}$. We can see there is no $m > n$ such that $L^m(Z_1) = L^n(Z_1)$, i.e., none of the clauses subsumes any of the others. Hence the process does not terminate for transitivity axiom as input.

However, the following results about the weak no merge resolution compilation hold; proofs may be obtained using the results of Section 2.

Theorem 3.1 *If the algorithm wnmerge terminates, then it correctly computes the set CKB_1.*

Theorem 3.2 *Let Σ be a given clausal database and CKB_1 be the compiled database of Σ. Then $\Sigma \equiv CKB_1$.*

Theorem 3.3 *Let $CKB_1(\Sigma)$ be the database obtained by the algorithm wnmerge from the clausal database Σ. Then $\Sigma \models C$ iff $CKB_1(\Sigma) \cup \neg C \vdash_u \bot$, i.e., $CKB_1(\Sigma)$ is unit refutation complete.*

The knowledge base CKB_1 stores all merge resolvents whereas CKB_2 keeps merge resolvents provided merge literals are not resolved upon. We can see that CKB_2 does not produce more number of clauses than CKB_1. In the algorithm given below the set of merge literals of a resolvent C (implicate) obtained from C_1 and C_2 are collected in a set M. But when we collect all possible implicates, the set of merge literals of all implicates are updated in the set $M^{'}$ which equals $M^{'} \cup M$.

Algorithm(nmerge)

```
Input : $\Sigma$, a set of clauses
Output : $CKB_2(\Sigma)$
begin
    $CKB_2 := \Sigma$;
    $M^{'} := \phi$;
    while two clauses $C_i$ and $C_j$ in $CKB_2$
    contain a pair of
          complementary literals $r$ and $s$
    do
         if the complementary literals do not
         unify with any element of $M^{'}$
            compute an implicate $C$ from
            $C_i$ and $C_j$
            $M$:= set of merge literals of $C$;
            if $C := \phi$
                return $CKB_2$
                if $C$ is subsumed by some
                clause in $CKB_2$
                    continue;
                else
                    if $C$ subsumes some clause
                    $D$ in $CKB_2$ or $C$ is a
                    merge resolvent
                            $CKB_2 := CKB_2 - \{D\}$;
                            $CKB_2 := CKB_2 \cup \{C\}$;
                    endif
                endif
            endif
         else
            continue
         endif
         $M^{'} := M^{'} \cup M$;
    od
    return $CKB_2(\Sigma)$
end
```

In general, CKB_2 saves space as compared to CKB_1, see Example 4.1 below. Analogous to *wnmerge*, the algorithm *nmerge* need not terminate. We show its partial correctness. Moreover, the following results about the no merge resolution compilation hold.

Theorem 3.4 *If the algorithm nmerge terminates, then it correctly computes the set CKB_2.*

Theorem 3.5 *Let $CKB_2(\Sigma)$ be the knowledge base obtained by the algorithm nmerge from the knowledge base Σ. Then $\Sigma \models C$ iff $CKB_2(\Sigma) \cup \neg C \vdash_u \bot$,i.e., $CKB_2(\Sigma)$ is unit refutation complete.*

As another alternative approach, we take the clausal database Σ as the set of horn clauses and non-horn clauses. In the algorithm we avoid taking resolutions between horn clauses as it increases the search space. We collect the set of horn clauses in CKB^H and non-horn clauses in CKB^N and perform their union to obtain CKB_3. We compute C as the resolution of two clauses where at least one clause is non-horn due to [5].

Algorithm(horn)

```
Input : $\Sigma$, the set of given clauses
Output: $CKB_3(\Sigma)$
begin
    $H$:=horn clauses of $\Sigma$;
    $N$:=non horn clauses of $\Sigma$;
    $CKB^H := H$ and $CKB^N := N$;
    while two clauses $C_1$ and $D_1$ from
    $H \cup N$ and $N$ respectively contain
    a pair of complementary literals
    do
        compute $C$;
        if $C := \phi$;
            return $CKB_3$
        else
            if $C$ is subsumed by some clause
            $D \in H \cup N$
                continue;
            else
                if $C$ is horn
                    if $C$ subsumes some clause
                    $D$ from $H$,$N$,$CKB^H$
                    and $CKB^N$
                        $H := H - \{D\}$,
                        $N := N - \{D\}$,
                        $CKB^H :=$
                        $CKB^H - \{D\}$,
                        $CKB^N :=$
                        $CKB^N - \{D\}$;
                        $H := H \cup C$;
                    endif
                    if $C$ is a merge resolvent or
                    $C$ subsumes some clause
                    $D$ from $CKB^H \cup CKB^N$
                        $CKB^H := CKB^H \cup \{C\}$;
                    endif
                else
                    if $C$ subsumes any clause $D$
                    from $N$ and $CKB^N$
                        $N := N - \{D\}$,
                        $CKB^N :=$
                        $CKB^N - \{D\}$;
                        $N := N \cup C$;
                    endif
                    if $C$ is a merge resolvent or
                    $C$ subsumes some clause
                    $D \in CKB^N$
                        $CKB^N := CKB^N \cup C$;
                    endif
                endif
            endif
        endif
    od
    $CKB_3 := CKB^H \cup CKB^N$;
    return $CKB_3(\Sigma)$
end
```

Note that here also, the algorithm may not terminate as is evident form Example 3.1. However, the following results about the algorithm *horn* hold.

Theorem 3.6 *Let Σ be a knowledge base and C be any clause. Then $\Sigma \models C$ iff $CKB_3(\Sigma) \cup \neg C \vdash_u \bot$, i.e., $CKB_3(\Sigma)$ is unit refutation complete.*

Theorem 3.7 *If the algorithm horn terminates, then it correctly computes the set $CKB_3(\Sigma)$.*

We illustrate the above three algorithms in the following example.

Example 3.2 Let $\Sigma = \{\{Px, Qya, Rx\}, \{Pa, \neg Qbz, Sy\}, \{\neg Pb, Qbz, Ta\}, \{\neg Py, \neg Qxa, Uz\}, \{\neg Sb, Vy, Wb\}\}$. Taking resolution between $\{Px, Qya, Rx\}$ and $\{\neg Pb, Qbz, Ta\}$ we get $C = \{Qya, Rb, Qbz, Ta\}$ associated with the substitution $[x/b]$. As Qyz subsumes both Qya and Qbz, it is the merge literal in C. $M = \{Qyz\} = M'$. $CKB_1 = \Sigma \cup \{Qya, Rb, Qbz, Ta\}, CKB_2 = \Sigma \cup \{Qya, Rb, Qbz, Ta\}$. Taking resolution between $\{Pa, \neg Qbz, Sy\}$ and $\{\neg Py, \neg Qxa, Uz\}$ we get $C = \{\neg Qbz, Sa, \neg Qxa, Uz\}$ associated with a substitution $[y/a]$. $\neg Qxz$ is the merge literal in C as it subsumes both $\neg Qbz$ and $\neg Qxa$. $M = \neg Qxz$ and $M' = \{Qyz, \neg Qxz\}$. So $CKB_1 = CKB_1 \cup \{\neg Qbz, Sa, \neg Qxa, Uz\}$, $CKB_2 = CKB_2 \cup \{\neg Qbz, Sa, \neg Qxa, Uz\}$. Taking resolution between $\{Px, Qya, Rx\}$ and $\{Pa, \neg Qbz, Sy\}$ we get $C = \{Px, Rx, Pa, Sb\}$ associated with $[y/b, z/a]$. As the complementary literals Qya and $\neg Qbz$ of the above two clauses unify with M', C can be added to CKB_1 but not to CKB_2. $CKB_1 = CKB_1 \cup \{Px, Rx, Pa, Sb\}$ but CKB_2 remains as such. Similarly, we get $C = \{\neg Pb, Ta, \neg Py, Ua\}$ by taking resolution between $\{\neg Pb, Qbz, Ta\}$ and $\{\neg Py, \neg Qxa, Uz\}$ with respect to complementary literals Qbz and $\neg Qxa$. Since this two complementary literals unify with M', C is added to CKB_1 but not to CKB_2. Thus, $CKB_1 = \{\{Px, Qya, Rx\}, \{Pa, \neg Qbz, Sy\}, \{\neg Pb, Qbz, Ta\}, \{\neg Py, \neg Qxa, Uz\}, \{\neg Sb, Vy, Wb\}, \{Qya, Rb, Qbz, Ta\}, \{\neg Qbz, Sa, \neg Qxa, Uz\}, \{Px, Rx, Pa, Sb\}, \{\neg Pb, Ta, \neg Py, Ua\}\}$ and $CKB_2 = \{\{Px, Qya, Rx\}, \{Pa, \neg Qbz, Sy\}, \{\neg Pb, Qbz, Ta\}, \{\neg Py, \neg Qxa, Uz\}, \{\neg Sb, Vy, Wb\}, \{Qya, Rb, Qbz, Ta\}, \{\neg Qbz, Sa, \neg Qxa, Uz\}\}$. It can be verified that $CKB_1 = CKB_3$ and the set of prime implicates become $PI = \{\{Px, Qya, Rx\}, \{Pa, \neg Qbz, Sy\}, \{\neg Pb, Qbz, Ta\}, \{\neg Py, \neg Qxa, Uz\}, \{\neg Sb, Vy, Wb\}, \{Qya, Rb, Qbz, Ta\}, \{\neg Qbz, Sa, \neg Qxa, Uz\}, \{Px, Rx, Pa, Sb\}, \{\neg Pb, Ta, \neg Py, Ua\}, \{Pa, Sy, \neg Pb, Ta\}, \{Pa, \neg Qbz, Vb, Wb\}, \{Px, Rx, Pa, Vb, Wb\}, \{Vb, Wb, Pa, \neg Pb, Ta\}\}$. The resolutions between $\{\neg Pb, Qbz, Ta\}$ and $\{Px, Rx, Pa, Sb\}$, $\{\neg Py, \neg Qxa, Uz\}$ and $\{Pa, \neg Qbz, Vb, Wb\}$, $\{Px, Rx, Pa, Sb\}$ and $\{Pa, Sy, \neg, Ta\}$, etc. cannot be computed due to undefined composition of substitution, i.e, $\sigma_1\sigma \neq \sigma_2\sigma$.

Note that the number of clauses produced by the above three methods are smaller than the number of clauses in the set of prime implicates.

4. Conclusion

In this paper we have presented three algorithms to transform a knowledge base Σ into an approximate knowledge base $CKB(\Sigma)$. But first order compilation is having many issues such as semi-decidability of entailment and the problem of termination of the compilation algorithms, which are not visible in propositional knowledge base. This is the reason why we can not compute a logically equivalent database to Σ, in general. Since implicates and implicants are dual to each other one algorithm which computes the implicates of CNF can be used to compute implicants of a DNF. These algorithms can then be used to compute $CKB(\Sigma)$ of an $SDNF$ formula Σ.

Since the compilation of the data base can take a long time to be completed, it is desirable to ask queries before the compilation stops. In these methods queries can be asked to the database at any intermediate stage of compilation. They can be queried at any time during compilation to obtain the answer by unit refutation. Though all the queries can not be answered but as the compilation goes on the possibility of answering the number of queries increases. If we want any query to be answered and it is answered at any intermediate stage during compilation then we do not need to continue the compilation further as it is too much space consuming. So the off line computation can be avoided partially.

When the original knowledge base is updated or modified a little, we need an incremental method to answer queries rather than preprocessing the knowledge base from the very beginning. Compilation with respect to wnm and with respect to horn clauses are incremental whereas compilation based on nm is not. Special attention must be paid to find the incremental methods of the above knowledge compilation methods. Similarly, other knowledge compilation methods such as the transversal clauses method [10] may be explored for the first order knowledge base.

References

[1] Andrews, P. B.(1968), Resolution with merging, *Journal of the ACM*, **15**, pp. 367-381.

[2] Cadoli, M., & Donini, F. M. (1998), A survey on knowledge compilation. *AI Communications-The European Journal for Articial Intelligence*, **10**, pp. 137–150.

[3] Cook, S. A. (1971), The complexity of theorem proving procedure, In *Proc. of the 3rd Annual ACM Symposium on the Theory of Computing*, pp. 151-158.

[4] del Val, A. (1994), Tractable databases: How to make propositional unit resolution complete through compilation, In *Proceedings of Fourth International Conference on Principles of Knowledge Represantation and Reasoning*, pp. 551-561.

[5] Henschen, L. & Wos, L. (1974), Unit refutations and horn sets, *Journal of the ACM*, **21**, pp. 590-605.

[6] Kean, A., & Tsiknis, G. (1990), An incremental method for generating prime implicants / implicates, *J. of Symbolic Computation.* **9**, pp. 185-206.

[7] de Kleer, J. (1992), An improved incremental algorithm for computing prime implicants. In *Proceedings of AAAI-92*, San Jose, CA, pp. 780–785.

[8] Reiter, R. (1971), Two results on ordering for resolution with merging and linear format, *Journal of the ACM*, **18**, pp. 630-646.

[9] Selman, B. & Kautz, H.(1991), Knowledge compilations using horn approximations, In *Proceedings of AAAI-91*, pp. 904-909.

[10] Singh, A. (1999), Computing prime implicants via transversal clauses, *Int. J. Computer Math.*, **70**, pp. 417-427.

[11] Tison, P. (1967), Generalized consensus theory and application to the minimisation of boolean functions, *IEEE Trans. on Elec. Comp*, **EC-16** (4), pp. 446-456.