CHAPTER 4

COMPLETENESS OF ORDERING REFINEMENTS

As soon as the resolution method was proposed in [Rob 65] and the analogous 'inverse method' in [Mas 64], it was noticed that for the completeness of these methods it is not always required that all the literals in a clause are resolved upon. Prohibiting resolution on some literals in a clause immediately means a reduction of the branching factor for the search tree. This reduction often leads to a slower growth of the set of derived clauses, which in turn can lead to finding the empty clause faster. For some classes of first order predicate logic formulas, it is possible to find such refinements of resolution that the search space turns out to be finite, thus giving a decision algorithm for these classes.

DEFINITION 4.1.: A resolution refinement is called complete iff the following holds: if a set of clauses is unsatisfiable, an empty clause is derivable from the set by the resolution refinement.

Quite obviously, we are mainly interested in complete refinements. However, sometimes the refinement we are looking for is generally incomplete, hence does not guarantee finding an empty clause for all unsatisfiable clause sets, yet is complete for a certain class of sets of clauses . In this case we can use the refinement, provided that the set of clauses we are investigating belongs to this class.

During the current chapter a clause is assumed to be a list rather than a set; that is, a clause can contain two syntactically equal literals. Consider some clause $C = (L_1, L_2, \ldots, L_n)$ and such a restriction or refinement of binary resolution that the literals $L_1, L_2, \ldots, L_k$ in C are allowed to be resolved upon and the literals $L_{k+1}, \ldots, L_n$ are not allowed to be resolved upon. We can say that this refinement is defined by a two-place predicate '>' defined on literals in the same clause with the meaning that whenever $(L_i > L_j)$ is true for some literals L_i and L_j in the clause, L_i is preferred over L_j, that is, L_j is not allowed to be resolved upon. The question whether L_i is allowed

to be resolved upon, depends on whether the given clause contains any literal L_q such that ($L_q > L_i$) holds: if this is the case, then L_i cannot be resolved upon in this clause, otherwise it can be resolved upon.

EXAMPLE 4.1.: In clause C for any two literals L_i, L_j the predicate ($L_i > L_j$) holds iff $i \leq k$ And $j > k$.

In order that the restriction were completely defined for a class of formulas, the predicate '>' must be defined so that for any clause C derivable from any set of clauses S in the given class by the restriction defined by '>', '>' were computable for any pair of literals in C. Generally '>' might not depend on the syntactic form of literals only; the derivation tree of the literal could also be taken into account (lock-resolution is a prime example of taking the derivation into account).

4.1 PROVING COMPLETENESS BY LIFTING

One of the usual methods for proving the completeness of any variant of the resolution method consists of the following steps (see [CL 73]):
(From now on we will consider completeness proofs of such kind and only for binary resolution, where the resolution and factoring steps are performed separately).

Ground case: It is proved that the variant is complete for the ground case, that is, for the Herbrand expansion of the set of clauses. Instead of using semantic trees we will use the completeness proof for ground lock-resolution (see [Boy 71], [CL 73]), where an induction on the k-parameter is used (k-parameter is the difference between the sum of lengths of clauses and the number of clauses).

Lifting: Let A and B be two non-ground clauses: $A = (A_1, \ldots, A_n)$, $B = (B_1, \ldots, B_m)$. Let A' and B' be any corresponding ground clauses from the Herbrand expansion of A and B. Lifting the ground completeness proof to the

variable-containing case means showing that whenever a new clause C' can be inferred from A' and B' by some complete refinement of resolution, such a clause C can be inferred from A and B so that C' is a ground instance of C.

Thus clauses with variables can be seen as a 'shorthand' for a possibly infinite set of ground clauses with the same structure. A resolution step for non-ground clauses can be seen as a shorthand for a possibly infinite number of resolution steps for subsets of corresponding ground clauses.

In chapter 5 C.Fermüller investigates resolution variants, the completeness of which he proves by using the method of semantic trees introduced in [KH 69]; in several cases the last method turns out to have some special properties, simplifying the completeness proof.

It is clear that for a completeness proof for any class of formulas it suffices to take into account only those ground clauses which can be obtained from the clausal form of some formula in the class. E.g., for the Monadic Class it suffices to consider clauses consisting of one-place literals only.

4.2 SOME IMPORTANT PROPERTIES OF THE '$\rhd$'-PREDICATES

4.2.1 Completeness for the ground case

When proving the completeness of any refinement of resolution given by a certain '$\rhd$'-predicate, the first step is to prove completeness for the ground case (remember that in this chapter we consider only proofs with the structure presented in section 4.1).

Not all possible '$\rhd$'-predicates yield a complete refinement for the ground case. Obviously, the first thing to note is that generally it is needed that in every clause there is at least one literal which is allowed to be resolved upon:

(A) for each clause $C = (L_1, \ldots, L_n)$ there is at least one i such that $L_j \rhd L_i$ holds for each j, $1 \leq j \leq n$.

Notice that since for deriving an empty clause from any unsatisfiable set of clauses we have to use clauses with a single literal (assuming we treat factorization as a separate rule), the fact (L ⟩ L) for all literals L follows from condition (A).

As demonstrated by the following example, the property (A) does not suffice for the general completeness of a '⟩'-refinement for ground clauses.

EXAMPLE 4.2.: (taken from [Boy 71]). Consider the following unsatisfiable set of ground clauses:

1: (P, R)
2: (¬R, P)
3: (R, ¬P)
4: (¬P, ¬R)

Let us allow to resolve only upon the first literal of each clause. On the first level we can thus get only the following two clauses, which both are tautologies:

1,4 give 5: (R, ¬R)
2,3 give 6: (P, ¬P)

We allow all literals of these tautologies to be resolved upon. Thus we get:

6,1 give 7: (P, R)
5,2 give 8: (¬R, P)
5,3 give 9: (R, ¬P)
6,4 give 10: (¬P, ¬R)

The set we got on the second level is identical to the input set. If we again allow to resolve only upon the first literals, no new clauses can be derived, thus the empty clause cannot be derived.

The problem with the '⟩'-refinement used in the example is that the '⟩' predicate is cyclical. Consider the definition of the '⟩'-predicate we used in terms of the syntax of literals:

(P ⟩ R) and (R ⟩ ¬P) and (¬P ⟩ ¬R) and (¬R ⟩ P)

The cycle starts with 'P' and ends with 'P'. Since the '>'- predicate defined in the above way is not transitive (e.g, although (P > R) and (R > ¬P) holds, (P > ¬P) does not hold), condition (A) holds for clauses which do not contain all the literals P, R,¬P, ¬R .

If the given '>'-predicate is transitive, that is, the following condition holds:

(B) for all literals X, Y, Z: (X > Y) and (Y > Z) implies (X > Z)

then for every literal X in the cycle the fact (X > X) must hold. In this case the condition (A) cannot be fulfilled. Thus, if we have a '>'-predicate for which both condition (A) and transitivity (condition (B)) hold, then '>' cannot form cycles.

The opposite is not true, however: although acyclicity guarantees (A), it does not guarantee (B). For example, the >'-predicate defined by ((P > R) and (R > ¬R)) is acyclical, but not transitive, as (P > ¬R) does not hold.

Nevertheless, every acyclical '>'-predicate can be strenghtened to a new '>'-predicate for which both, (A) and (B), hold. By strengthening the '>'-predicate we mean forming the transitive closure of the '>'-predicate, that is, adding new pairs of literals for which '>'-holds, while retaining all the old pairs. E.g., the '>'-predicate defined by ((P > R) And (R > ¬R)) is strengthened to a transitive predicate by adding (P > ¬R).

In the following we will only investigate such '>'- predicates, for which both conditions, (A) and (B), hold for the ground case. We will call such '>'-predicates orderings.

As shown by S. Maslov (see [Mas 68], [Mas 71]), the acyclicity of '>' (therefore also conditions (A) and (B)) is enough for the resolution with a '>'-refinement to be complete for ground formulas.

Let us consider some orderings. We will say that the ordering is complete iff the following condition holds:

(C) for all ground literals X and Y with X ≠ Y either (X > Y) or (Y > X) holds.

EXAMPLE 4.3.: (P > R > ¬P > ¬R) is a complete ordering for a clause set containing only atoms R and P.

As an example of an incomplete ordering we could define:

$$P > \neg P,\ P > \neg R,\ R > \neg P,\ R > \neg R.$$

Then none of

$$P > R,\ R > P,\ \neg P > \neg R,\ \neg R > \neg P$$

holds and in the clauses (P, R) and (¬P, ¬ R) both literals are allowed to be resolved upon.

We will write (X >< Y) to denote the incomparability of any literals X and Y. By the definition of orderings (X >< X) holds for any literal X and any ordering. In the last example of '>' we have (P >< R) and (¬P >< ¬R).

An important class of orderings is formed by lock-resolution (introduced in [Boy 71]). These orderings are not defined in terms of syntax of literals, but in terms of a derivation tree. Namely, for each literal in the input set of clauses a certain numerical index is given and the ordering is defined in terms of these indices:

$$X > Y \quad \text{iff} \quad \text{index}(X) < \text{index}(Y)$$

We will denote the fact that the literal A has index i by writing A{i}. Literals in the derived clauses retain the same index they had in the parent clauses. We will not consider an index to be a part of the literal; instead we consider it to be a part of a deduction tree.

Thus syntactically equal literals in different clauses are treated as different objects from the standpoint of lock resolution. The fact that a lock-ordering is indeed an ordering (that is, enjoys properties (A) and (B)) follows from the fact that the usual ordering '<' of natural numbers has properties (A) and (B).

Since we know that lock-resolution is complete, it is simple to prove that resolution for ground clauses with any syntactically defined acyclical '>-'predicate ordering is complete. We start by strengthening the acyclical '>'-predicate to the corresponding ordering by adding new pairs of literals for which '>' holds, as shown earlier. The restriction given by this ordering is stronger than the one given by the original '>'- predicate, so whenever the ordering gives a complete resolution refinement, the original '>'-predicate also does. The ordering we get may contain several trees and branches of comparable literals, literals in

different chains and different branches being incomparable. When we here speak of "trees", we mean oriented graphs which do not contain oriented cycles, although they may contain unoriented cycles.

EXAMPLE 4.4.:

(4.1) ((P > R) and (¬P > ¬R))

(4.2) ((P > R) and (R > ¬ P) and (P > ¬ P))

(4.3) ((P > R) and (R > ¬P) and (P > ¬P) and (R > ¬R) and (P > ¬R))

are such orderings, the first consisting of two incomparable trees, the second containing an unoriented cycle and the third containing two incomparable branches.

We are going to give numerical indexes to all literals, such that the corresponding lock-strategy is stronger (or of exactly the same strength) than the strategy given by ordering. The indexing method is the following:

- all syntactically equal literals get the same index.
- trees are indexed one-by-one, that is, all indexes in the next tree are bigger than the indexes in the previous tree.

EXAMPLE 4.5.:

((P{1} > R{2}) and (¬P{3} > ¬R{4})).

- we divide all vertexes between levels, so that vertexes at the next level have the same index, which is bigger than the index for the previous level.

EXAMPLE 4.6.: ((P{1} > R{2}) and (R{2} > ¬P{3}) and (P{1} > ¬P{3})) and
((P{1} > R{2}) and (R{2} > ¬P{3}) and
(P{1} > ¬P{3}) and (R{2} > ¬R{4}) and (P{1} > ¬R{4}))

Such an indexing is possible for all orderings (recall properties (A) and (B)) and for all finite sets of literals. So, the fact that any acyclical '>'-ordering gives a complete resolution refinement for finite sets of clauses is proved. Notice that we never use infinite sets of clauses. Although the Herbrand expansion of a non-ground formula may be infinite, whenever this formula is unsatisfiable, the result can be shown using the first N clauses in the Herbrand

expansion for some N. Generally we do not know N beforehand, but we know that iff the formula is unsatisfiable, the required finite Herbrand expansion can be formed.

An important point to notice is the difference between orderings defined in terms of syntax and in terms of deduction tree. This becomes noticeable when we investigate the treatment of tautological clauses and subsumed clauses.

DEFINITION 4.2.: A clause is called tautological iff it contains some literal X and its negation X^d. A ground clause C subsumes a ground clause D iff the set of literals in C is a subset of the set of literals in D.

During the process of deriving new clauses by resolution we want to avoid any unnecessary derivations. As it is the case that for most of practically interesting formulas most of the derived clauses are subsumed by older clauses or subsume some older clauses, it is very important that we could avoid using any subsumed clauses. We have to consider the following question: in which cases does the resolution with ordering strategy allow us to discard subsumed clauses ?

It is useful to consider 'back subsumption' and 'forward subsumption' separately. By 'back subsumption' we mean the following restriction: whenever a newly derived clause C subsumes some older clause D, we eliminate D (that is, from now on we will not use D for any resolution steps). By 'forward subsumption' we mean the following: whenever a newly derived clause C is subsumed by some older clause D, we eliminate C.

We also consider separately the cases where a subsuming clause is syntactically equal to the subsumed clause (up to renaming variables) and the case where the subsuming clause is more general (for ground case this means to be shorter) than the subsumed clause. The first kind of subsumption we call 'trivial subsumption' and the second kind 'proper subsumption'.

By the compatibility of some kind of subsumption with some ordering we will mean that the combination of the resolution refinement defined by the ordering

with the clause deletion strategy (using the kind of subsumption) is a complete resolution variant.

THEOREM 4.1.: Proper back subsumption is compatible with all orderings, including lock resolution.

Proof:
Recall that we use resolution as a process of iteratively generating new (without subsumption, strictly bigger) conjunctions of clauses. Consider an unsatisfiable set of clauses $S = \{A_1, A_2, ..., A_n\}$. Then the set $S' = \{A'_1, A_2, ..., A_n\}$ such that A'_1 subsumes A_1, is also unsatisfiable. Suppose A'_1 is used to properly back subsume A_1. Then at the next level the obtained set remains unsatisfiable. As the subsumption application was of a proper kind, the chain defined by replacing clauses by subsuming ones cannot be infinite: at every such step we get a more general clause, up to a one-literal clause (where in the non-ground case all the arguments of the predicate are different variables).

Q.E.D.

For forward subsumption the proof does not work in the general case since even the proper forward subsumption can result in a cycle: although the set of clauses remains unsatisfiable, resolution may give a set of clauses which is identical to the set found at the last level.

However, both forward and backward subsumption are compatible with syntactically defined orderings (and acyclical '>'- predicates also) for the ground case, as shown in [Mas 68]. Again, this becomes immediately visible when we investigate subsumption for lock resolution.

As demonstrated in [Boy71], general subsumption is uncompatible with lock resolution even for the ground case:

1: (P{1}, R{2})
2: (¬R{3}, P{4})
3: (R{5}, ¬P{6})
4: (¬P{7}, ¬R{8})

At the first level we can thus only get the following two clauses, which both are tautologies:

1,4 give 5: (R{2}, ¬R{8})

2,3 give 6: (P{4}, ¬P{6})

Both clauses obtained at the next step are forward subsumed by older clauses:

5,2 give 7: (¬R{8}, P{4})

6,4 give 8: (¬P{6}, ¬R{8})

If we do not eliminate subsumed clauses resolution will proceed as follows:

7,8 give

9: (¬R{8})

6,8 give 10: (¬P{6}, ¬R{8})

9,3 give 11: (¬P{6})

11,1 give 12: (R{2})

9,12 give □

We will examine how we can restrict forward subsumption so that it becomes compatible with lock resolution.

THEOREM 4. 2.: Forward subsumption with the following restriction is compatible with lock-resolution: A is defined to subsume B in the restricted sense iff A subsumes B in the ordinary sense and the corresponding literals in A and B have the same indices.

Proof:

Consider the deduction tree of an empty clause from some set of clauses by lock-resolution. Replace every occurrence of clause B by clause A in this tree. Consider two cases:

- A is syntactically equal to B (trivial subsumption). Then, since the indices are the same for A and B, the proof tree is unchanged and obviously all the steps remain permissible.
- A is more general than B (for the ground case this means that A is shorter than B). If A is shorter than B, then we have to cut off branches in the old tree corresponding to literals present in B, but not in A. For clauses in the tree for which B was an ancestor, we have also to cut off these literals

which were introduced by the branches we have cut off. Then, due to the restriction on the indices all the steps remain permissible lock-resolution steps. The essential possible change from the old tree is that the empty clause may be derived earlier in the rebuilt tree.

For non-ground clauses we note that replacing A by a more general clause permits all unifications in the rebuilt tree. Since indices do not depend on the syntax of the clause, the previous argument holds.

Q.E.D.

Consider some syntactically defined ordering for the ground case. As we know, this ordering can be modelled by lock resolution. Now, whenever some clause A subsumes a clause B, all the literals in A are also in B, thus their indices are equal (since the indices of syntactically equal literals are the same). Thus the restriction for indices in theorem 2 is always fulfilled and full subsumption is compatible with any syntactically defined ordering for the ground case.

In the light of subsumption the treatment of tautological clauses is trivial: we can always eliminate tautological clauses, provided that any clause derived with a tautology as one of the parents is forward subsumed by its second parent. For syntactically defined orderings for the ground case we can therefore eliminate all tautologies. This is not the case for lock-resolution, since, for example, (P{1}, R{2}) and (¬P{3}, P{3}) give (P{3}, R{2}), which has indices different from the parent clauses (see also the previous example).

4.2.2 Properties of orderings for the non-ground case

As a clause with variables can be considered to be a shorthand for a possibly infinite set of ground clauses (Herbrand expansion), the most important thing is to consider a relation among an ordering on a clause with variables and the ordering on all ground instances of this clause.
As said before, all the results of the previous chapter which hold for finite input sets of clauses, remain valid for the possibly infinite Herbrand expansion of any non-ground set of clauses.

For unrestricted binary resolution the lifting of a completeness proof to the non-ground level is possible due to the existence of a most general unifier for any two literals. For a resolution refinement it might happen that a resolution step which is possible for ground instances of clauses is not any more possible for the corresponding non-ground case, thus destroying lifting.

EXAMPLE 4.7.: Consider the following syntactically defined ordering for the general case:

$(A >_{ig} B)$ iff A is ground and B is non-ground.
Then only the first literals of the following clauses are allowed to be resolved upon:

$(P(a), R(x))$, $(\neg R(a), P(x))$.

Therefore these two clauses cannot be used to derive any new clauses. However, for the ground case corresponding to these clauses the derivation of new clauses is possible:

$(P(a), R(a))$ and $(\neg R(a), P(a))$ give $(P(a), P(a))$.

Indeed, if lifting is not preserved, we have no guarantee for the completeness of the corresponding ordering strategy. Consider the same ordering $>_{ig}$ and the following unsatisfiable set of clauses:

1: $(P(a), R(x))$
2: $(\neg R(a), P(x))$
3: $(R(a), \neg P(x))$
4: $(\neg P(a), \neg R(x))$

At the first level we can only get the following two clauses, which both are tautologies:

1, 4 give 5: $(R(x), \neg R(x))$
2, 3 give 6: $(P(x), \neg P(x))$

We allow all literals of these tautologies to be resolved upon. The set we get at the second level is identical to the input set, therefore the empty clause is never derived.

We want to separate a class of orderings which preserve lifting. Since conditions (A) and (B) hold for orderings, the following condition is what we need:

(D) for any literals X, Y and any ground substitution σ the following implication holds: $(X > Y) \Rightarrow (X\sigma > Y\sigma)$.

The following does not suffice:

(Dweak) $(X\sigma > Y\sigma) \Rightarrow \neg(Y > X)$.

The counterpositive of (Dweak) is $(Y > X) \Rightarrow \neg(X\sigma > Y\sigma)$, and this obviously holds for the ordering in our last example, since it follows from $(Y >_{ig} X) \Rightarrow (X\sigma >\!<_{ig} Y\sigma)$. But in our example all the ground literals were incomparable w.r.t. the ordering $>_{ig}$.
Given any ordering obeying (D), it is possible to define the resolution refinement for this ordering in two distinct ways, so that lifting is preserved for both.

The first way is to order any clause with the given ordering and prohibit resolution on non-maximal literals, exactly as was done for the ground case. We will call such refinements "apriori refinements".

The second way, introduced in [Joy76], relies on noticing that in order to preserve lifting, it suffices to consider clauses under application of m.g.u.s. Consider two clauses,
$C_1 = (L_1, L_2, \ldots, L_n)$ and
$C_2 = (G_1, G_2, \ldots, G_m)$, such that L_1 and G_1^d are unifiable with the substitution $\sigma = m.g.u(L_1, G_1^d)$. Given any ordering $>_X$ we can define a liftable refinement of binary resolution in the following way:
The clause $(L_2, \ldots, L_n, G_2, \ldots, G_m)\sigma$ is allowed to be derived by binary resolution iff there is no L_i in C_1 such that $(L_i\sigma >_X L_1\sigma)$, and no G_i in C_2 such that $(G_i\sigma >_X G_1\sigma)$.

The last kind of resolution refinements we will call "aposteriori refinements". As shown by C. Fermüller in [Fer91] and in chapter 5, in several cases aposteriori refinements can be much more powerful and convenient than apriori refinements. On the other hand, using aposteriori refinements in the actual proof search is connected with a noticeable computational overhead. Therefore, in case there exists an apriori refinement with the same restrictive power as the aposteriori refinement, it is better, because more efficient, to use the apriori refinement.

The fact that any syntactically defined acyclical '>'-predicate with property (D) gives us a generally complete resolution refinement was first proved by S.Maslov (see [Mas 68], [Mas 71]; the proof is presented in [Mas 64]). S. Maslov called such predicates "π- orderings". He also demonstrated that subsumption is compatible with π-orderings. As in the current chapter we have reserved the name 'ordering' for '>'-predicates with the properties (A) and (B) only, π-ordering is a wider class of predicates than the orderings in our sense. Yet we prefer not to drop the name 'π-orderings', so the reader is asked to keep

in mind that a 'π-ordering', differently from simply 'ordering', need not satisfy (B) (acyclicity suffices instead of (B)).

Due to (D) we can lift the completeness and subsumption-compatibility result for resolution with any syntactically defined acyclical '>'- predicate (see section 4.2.1). This gives the completeness and subsumption-compatibility proof for resolution with any π-ordering.

Notice that property (D) also holds for lock-resolution. Thus we get completeness for general (that is, not ground-only) lock-resolution by lifting. Theorems 1 and 2 were already formulated for the general case, so we know that these types of subsumption are compatible with general lock-resolution. An important point to notice is that, differently from the ground case, it is not possible to use lock-resolution to model any syntactically defined ordering with the property (D). We will give an example later.

Independently from S. Maslov, R. Kowalski and P. Hayes introduced orderings called A-orderings in the paper [KH 69]. An A-ordering is a '>'-predicate on atoms (not on literals!) which satisfies (A), (B) and (D). As A-orderings form a subclass of π-orderings, the completeness of a resolution refinement defined by an A-ordering follows from the completeness of resolution with π-orderings. We refer to chapter 5 for a more detailed investigation of A-ordering.

4.2.3 Factoring

In the ground case the notion of factoring is of no special importance, as the clauses are normally considered to be sets of ground literals and therefore the factoring operation is, so to say, performed implicitly by definition. This is not completely true for non- syntactically defined orderings, like lock-resolution, as syntactically equal ground literals can have different indices. For lock resolution it is therefore defined explicitly that in case a clause C contains two syntactically equal literals A_i and B_j with different indices i and j, C should be replaced by a clause where B_j is removed and the index i in A is replaced by the max(i, j).

For non-ground clauses the notion of factoring gains explicit importance, as set formation is not enough any more:

e.g, the clause (P(f(x)), P(y)) has a ground instance (P(f(a)), P(f(a))), which gives a one-element set P(f(a)). Obviously, in order to use lifting for completeness proofs, factoring has to be performed.

As noted in [Lei 89] and in chapter 2, in theorem-proving literature factoring is treated in one of the two ways: Robinson's resolution method requires factoring during a binary resolution step only, and the set of factorized literals in a clause has to contain the literal which is resolved upon. The other way to treat factorization is to use it as a separate rule - this can be useful from the standpoint of implementation, as using factorization during a binary resolution step is connected with a certain computational overhead.

For ordering refinements of Robinson's resolution the set of factored literals in a clause always contains some literal L such that for no R in the clause $R >_0 L$. The same restriction can therefore be introduced to factoring as a separate rule: for any ordering $>_0$ it is required that any set of factored literals in a clause must contain a literal L such that for no other literal R in the same clause $R >_0 L$ holds.

4.3 POSSIBILITIES OF USING π-REFINEMENTS AS DECISION ALGORITHMS

By "π-refinement" we will mean an apriori resolution refinement based on some π-ordering.

We will present the following ordering to be used later:

DEFINITION 4.3.1: $(X >_{sub} Y)$ iff all arguments of the literal Y are proper subterms of some arguments of the literal X.

Then $P(f(y)) >_{sub} R(y,y)$, $R(g(f(y)),y) >_{sub} R(y, f(y))$,
$P(g(f(a)), a) >_{sub} R(a, f(a))$, $P(x, y) \not>_{sub} P(x)$, to give some few examples.

As shown by N. Zamov and V. Sharonov (e.g. in [ZS 74]), and independently by W. Joyner in [Joy 76], the apriori restriction strategy based on this ordering is a decision algorithm for clause forms of formulas from the Ackermann Class (prefix $\forall\exists \ldots \exists$) and Monadic Class (formulas with one-place predicates only). Notice that both of these classes are defined for the predicate logic without function symbols; the only function symbols appearing in clause forms of such formulas are the ones produced by Skolemization.

In this chapter we concentrate on completeness, so the reader is suggested to look at the following chapters to see the decision proofs. All such proofs show that the set of clauses derivable by the investigated strategy is finite for a clause form of any formula in these classes. This is achieved by showing that the term depth does not grow and due to condensing the length of derived clauses is also bounded (we call the factors subsuming their parents 'condensings') . So, if resolution does not produce any more clauses and it has not derived an empty clause, we can conclude (due to completeness of the strategy) that the input set of clauses must be satisfiable.

Let us investigate the properties of the $>_{sub}$-ordering. We can easily notice that this ordering relation has both, properties (A) and (B), thus is really an ordering. Therefore it is complete for the ground case. As $(P(a) >\!<_{sub} P(b))$, (C) does not hold. Consider property (D). Whenever $(X >_{sub} Y)$ for some literals X and Y, $(X\sigma >_{sub} Y\sigma)$ also holds. Thus we have proved that the given ordering is indeed a π-ordering.

We give an example of using the apriori restriction with a given ordering $>_{sub}$ as a decision strategy. Consider the following set of clauses:

1: (P(f(x)), P(x))

2: (¬P(f(x)), ¬P(x))

Ordering $>_{sub}$ allows to resolve only on the first literals, thus we can produce a single clause, (P(x), ¬P(x)), which is tautological and due to the fact that we used a π-refinement, we can eliminate it. As no more clauses can be derived, we know that the input set is satisfiable.

As one can easily check out, unrestricted binary resolution will deduce an infinite amount of clauses from the given set. As can be almost as easily checked, the same happens for lock- resolution with any indexing of the input clauses (recall that tautologies cannot be eliminated for lock-resolution). So, π- refinements cannot be modelled by lock-resolution.

4.4 BEYOND π-REFINEMENTS

As noticed by N. Zamov, V. Sharonov and W. Joyner, π- refinements (or, in Joyner's case, refinements based on A-orderings), cannot be used to decide the Gödel's Class, the class of formulas with the prefix ∀∀∃. The problem here is that the clause form contains clauses like (P(x, y), P(x, x)), which are left unordered by any π- ordering. The free use of such clauses produces clauses like (P(f(x,y), z), P(x, y)), which start producing terms with unbounded depth.

EXAMPLE 4.8.: From (P(x, x), P(x, y)) and (¬P(f(x,y), f(x,y)), ¬P(x, y)) we get (P(f(x,y), z), ¬P(x, y)). From the last one and from (¬P(x, f(x,y)), P(f(x,y), f(x,y))) we get: (¬P(x, y), P(f(f(x,y), y'), f(f(x,y), y'))). In this way the depth of terms is going to grow unbounded.

It is easily seen that no π-ordering can order the clause (P(x,y), P(x, x)) so that the first literal is preferred. If an ordering prefers P(x, y) to P(x, x), then it does not satisfy (D), since in order to satisfy (D) it must prefer the first literal of (P(a, a), P(a, a)), the ground instance of the last clause. Then it either is not syntactical or does not satisfy (A).

4.4.1 Amending lifting by an additional saturation rule

In the papers [Zam 72, 89, 89a] the problem is overcame by defining an ordering which does not generally satisfy (D), and by introducing a special saturation rule to resolution, so that for resolution augmented with this saturation rule, lifting is possible. We will define an ordering which essentially captures the idea of the orderings for the Minimal Gödel Class defined in [Zam 89a].

DEFINITION 4.4.1: $(X >_{sv} Y)$ iff one of the following holds:

(1) each argument of the literal Y is a proper subterm of some argument of the literal X.

(2 neither X nor Y contain functional terms with arguments, the set of variables in Y is a proper subset of variables in X and the set of variables in Y is non-empty.

The proof that this ordering indeed gives a decision strategy for the Minimal Gödel's Class can be found in the paper [Zam 89a] and in chapter 6 (indeed, the orderings defined there are somewhat different, but for the Minimal Gödel Class they give exactly the same restrictions).

Obviously, the last ordering $>_{sv}$ is syntactically defined and satisfies conditions (A) and (B). Notice that the condition (1) in the '$>_{sv}$'-ordering is exactly the same as the '$>_{sub}$'-ordering given in the previous example.

The main problem, as we saw already, lies in condition (2), which destroys lifting. One way to overcome this, previously suggested by N. Zamov and V. Sharonov, consists of adding a new derivation rule, saturation, to resolution. Saturation allows to derive certain substitution instances of clauses. Understandably the idea is that saturation always allows to derive only a finite amount of substitution instances, and generally, the less the better. The following saturation rule suggested by N. Zamov makes resolution with the $>_{sv}$-ordering complete for Minimal Gödel's Class:

(SAT1) If resolving upon some literal L in some clause C is prohibited only due to the fact that C contains some literal M with $(M > L)$ by condition (2), then derive all substitution instances $C\sigma$ of this clause such that $\sigma = \{y/x\}$, for $x \in V(M) - V(L)$, $y \in V(L)$.

EXAMPLE 4.9.: From (P(x,y), R(y,y)) (SAT1) derives (P(y,y), R(y,y)). From (A(f(x, y)), P(x,y), R(y,y)) (SAT1) derives nothing, since resolution upon R(y,y) is prohibited due to condition (1) already.

Let us see why our last ordering $>_{SV}$ for resolution with the additional derivation rule (SAT1) preserves lifting in case of the Minimal Gödel Class. We have to show that whenever some literal L' in the ground instance C' of any clause C is allowed to be resolved upon, the corresponding literal L in C is also allowed to be resolved upon. For resolution without (SAT1) this does not hold because for some pairs of ground clauses $A\sigma$ and $B\sigma$ we have $(A\sigma >\!<_{SV} B\sigma)$ (e.g, (P(a, a) $>\!<_{SV}$ P(a, a)), but there can be corresponding clauses A and B with variables such that $(A >_{SV} B)$ (eg, P(x,y) $>_{SV}$ P(y,y)) because of condition (2). Recall that the clause form of any formula in the Minimal Gödel Class does not contain constants. Thus the only literals in Herbrand expansion which do not contain functional terms with arguments, have a single new constant symbol (e.g, 'a') on all the argument places (e.g, S(a,a,a)). So, lifting is not preserved on these formulas only. Thus, whenever lifting is not preserved for some ground formula derived from Herbrand expansion, this ground formula is a substitution instance of a formula derived by (SAT1) for the variable-containing case. Now, the formula derived by (SAT1) "cures" lifting, as literals in it are not ordered.

An unpleasant consequence of using such a saturation rule is that it becomes unclear how to deal with subsumption. Obviously, we cannot use general subsumption, since then the clause derived by the saturation rule would be subsumed, as if it had never been derived. The following example demonstrates that full subsumption is not compatible with the last ordering. During the example we do not use (SAT1), since by full subsumption the derived clauses would be subsumed.

Consider the following unsatisfiable set of four clauses:

1: (P(x,y), R(x,x))
2: (¬R(x,y), P(x,x))
3: (R(x,y), ¬P(x,x))
4: (¬P(x,y), ¬R(x,x))

The $>_{sv}$-ordering allows to resolve upon the first literals only, using condition (2). At the first level we can only get the following two clauses, which both are tautologies.

1, 4 give 5: (R(x, x), ¬R(x, x))

2, 3 give 6: (P(x, x), ¬P(x, x))

Obviously we have to use these tautologies to get the refutation. On the second level we get:

5,2 give 7: (P(x, x), ¬R(x, x))

6,4 give 8: (¬P(x, x), ¬R(x, x))

5,3 give 9: (¬P(x, x), R(x, x))

6,1 give 10: (P(x, x), R(x, x))

Notice that all these are subsumed by clauses 1-4. Any clause derivable on the second level is obviously subsumed by clauses 1-4 since clauses 5 and 6 are tautologies. Thus we have to use the subsumed clauses in order to get the refutation:

7, 8 give 11: ¬R(x, x)

9, 10 give 12: R(x, x)

11, 12 give the empty clause.

As noted by N. Zamov, the use of saturation rule for any class decidable with the corresponding resolution variant can be postponed until no new clauses can be derived without saturation. The reason is that the amount of derivable clauses is finite (this follows from the proof of decidability).

4. 4. 2 Amending lifting by replacing clauses with certain instances

Whereas the method considered in the previous section consisted in using an additional derivation rule to derive certain instances of clauses, in [Joy76] the problem of amending lifting is overcome by replacing clauses with sets of instances. That is, in case some derived clause C fulfils certain properties, a new set of instances inst(C) is derived by a special saturation rule and C itself is deleted (compare also sections 5. 3 and 5. 4).

Any such a resolution variant using saturation combined with deletion of clauses needs a special completeness proof to show why the certain resolution variant is complete for a certain investigated class. The useful property of such resolution variants, when compared to variants decribed in the previous section, is that subsumption does not have to be restricted in any way.

Resolution variants of the "replacement" kind have been investigated by C. Fermüller (see [Fer91]), and are explained in detail in chapter 5 of the current monograph.

4. 4. 3 Amending lifting by defining a separate ordering for the ground case

N. Zamov has shown that although the ordering $>_{SV}$ cannot be lifted, the apriori refinement using this ordering is nevertheless complete, for Maslov's Class K, (which contains Gödel's Class) without any saturation (although this refinement won't be compatible with full subsumption). The proof of this fact can be inferred from the analogous proof for a very close ordering, given in chapter 6 dealing with Maslov's Class K.

The general method (originating from N. Zamov and V. Sharonov) for proving the completeness of the strategy based on some such ordering $>_0$ which cannot be lifted from the ground case, is to take some stronger ordering than $>_0$, say $>_0'$, for the ground case, such that $>_0'$ can be lifted to $>_0$. That is, the following modified version of condition (D) must hold for the input set and for all derived clauses and the corresponding Herbrand instances (not just ground instances of the input clauses):

(D') for any literals X, Y in some clause and corresponding Herbrand instances X' and Y' s.t. X', Y' occur in a ground clause derived from Herbrand expansion holds $(X >_0 Y) \Rightarrow (X' >_0' Y')$.

A good candidate for such an $>_0'$ is some non-syntactically defined ordering (eg, lock-resolution with the suitable indexing schema) for the ground case. Such a technique was originally used by N.Zamov for several completeness proofs, including the one in [Zam89].

The consequence of using a special underlying lock-resolution ordering is that full forward subsumption and tautology-elimination are not allowed. They are allowed only in cases correponding to π-ordering.

EXAMPLE 4.10.: For any t, P(x, y) is not allowed to subsume P(x, x){t/x}. For any t, the tautology {P(x, x), ¬P(x, x)}{t/x} cannot be removed. However, P(x, y) is allowed to subsume P(x, f(x, y)) and the tautology {P(x, y), ¬P(x, y)} may be removed.

4.5 THE CLASS E^+

In this section we will investigate some subclasses of the class called E^+. We will prove completeness of a certain refinement for a subclass of the class E^+. The ordering used here is much "less liftable" than the previously presented ordering $>_{SV}$, thus the method of using some strong lock-indexing for the ground case is somewhat more important.

In chapter 5 C. Fermüller investigates the same class E^+ using methods different from the current chapter: whereas we use a certain apriori refinement based on a generally non-liftable ordering, he uses an aposteriori refinement based on a certain A-ordering.

DEFINITION 4.5.: A functional term is called weakly covering iff for all subterms s of t that properly contain variables V(s) = V(t). An atom or literal A is called weakly covering iff each argument of A is either a ground term or a weakly covering term s.t. V(t) = V(A). (This definition appears in chapter 5).

For important properties of weakly covering terms and literals see chapter 5, section 5.1.3.

DEFINITION 4.6.: We say that a clause set S belongs to the Class E^+ iff for all clauses C in S the following holds:

(i) each literal in C is weakly covering.
(ii) for all literals L, M in C either V(L) = V(M) or $V(L) \cap V(M) = \emptyset$

The class just presented contains the clausal form of the class of formulas with function symbols where every atom contains only one variable; this class has been called N_1 and E (if given in terms of derivability, formulas in E can generally be assumed to be of the form $(\exists x)\, M$). Previously, the solvability of E has been shown in various ways. E contains, for example, the Skolemized version of Initially-extended Ackermann Class (the class of formulas without function symbols and with the prefix $(\exists x_1) \ldots (\exists x_n)(\forall y)(\exists z_1) \ldots (\exists z_m)$).

EXAMPLE 4.11.: The following clause set belongs to class E^+:

$\{P(f(f(a)), g(c, g(y, x))),\ P(u, f(u))\}$

$\{\neg P(f(h(x, y, z)), z),\ G(b),\ G(v),\ \neg G(u)\}$

$\{E(g(h(z, x, y), f(b)))\}$

$\{R(g(v, g(v, u)), g(u, v)),\ R(f(b), g(f(f(a)), g(u, v)))\}$.

In the paper [Tam 90] it is shown that in case the apriori v-refinement given below is complete for the Class E^+, it is also a decision strategy for E^+. However, so far we have not been able to prove the completeness of this refinement for E^+, nor have we found any counterexamples to this.

As to the decidability of E^+, as shown by C. Fermüller in chapter 5, a simple saturation rule added to the v-refinement will guarantee completeness of the procedure. Then a simple modification of the theorem 4 below gives decidability of E^+. However, from the proof-search aspect the saturation rule is very unwelcome, and we would still like to find a provably complete resolution refinement without having to use saturation.

In the following we will give a completeness proof of the apriori v-refinement for the clause form of the class E, which, as previously noted, is a subclass of E^+.

4.5.1 Completeness of a v-refinement for the class E

DEFINITION 4.7.: Let A and B be literals. We say that $A >_v B$ iff the following is true:

$V(A) = V(B)$ and for all y in V(A) holds $\tau_{max}(y, B) < \tau_{max}(y, A)$.

The v-refinement is defined by the apriori restriction on a resolution rule: only the maximal (in the sense of new ordering $>_V$) literals in a clause are allowed to be resolved upon.

EXAMPLE 4.12.: In the clause A(f(f(y,x),y),a), B(b,g(x,y)), C(y,b,x) only the literal A(f(f(y,x),y),a) is allowed to be resolved upon.

The proof presented in [Tam90] and in section 4.5.3 of the fact that only a finite number of clauses can be derived by the v-refinement from any clause set in the Class E^+, has the following structure: it is shown that the v-refinement has this property for the Class $E^{+'}$, which is the same as the Class E^+, except for the property (ii), which in $E^{+'}$ is replaced by a property (ii'):

(ii') for all literals L, M in C holds V(L) = V(M).

If the v-refinement decides $E^{+'}$, it is clear that v- refinement with splitting decides the class E^+, since every formula in E^+ can be split into a finite set of formulas in $E^{+'}$.

Notice that instead of using splitting, the decision algorithm for E^+ is obtained from the same algorithm for $E^{+'}$ by either using an aposteriori refinement of Robinson's resolution or using a corresponding restriction on factorization.

4.5.2 Completeness proof of the v-refinement for a class E'

DEFINITION 4.8.: We say that a clause set S belongs to the Class E' iff the following properties hold for all clauses C in S:

1. Each literal in C is weakly covering.
2. Either C is ground or V(C) = {x} for some variable x and for each literal L in C holds V(L) = {x}.

We will now present a proof of the completeness of the v- refinement for the class E'. The general completeness of this refinement is still an open problem.

The structure of the proof: first we define a certain indexing for all literals in the ground instances of the given clauses. Then, given that the set of ground clauses is unsatisfiable, we can find a lock-resolution proof with the defined indices. We will show that the ordering of literals in the ground clauses given by indices can be lifted to the $>_V$ -ordering for clauses with variables; the lifting is immediate for the original set of clauses and can be shown to hold for all new clauses generated by resolution with the refinement from a clause set in Class E'.

Indexing

Let S be a clause set from the Class E'. Consider the Herbrand expansion H of S: Each clause C' in H is an instance of some clause C from S: C' = C{t/x} where x is a single variable in C, t is a ground term from the Herbrand universe of S. We will give each occurrence of the literal $L_j = L_j\{t/x\}$ in C' the following index: $\tau(t) + \tau_V(L_j)$, where $\tau_V(L)$ is the depth of the deepest occurence of a variable in L.

EXAMPLE 4.13.: The following is a sample indexing:

P(x, ffa)	P(fa, x)	P(fa, fx)	
P(a,ffa) :0	P(fa,a) :0	P(fa,fa) :1	| a/x
P(fa,ffa) :1	P(fa,fa) :1	P(fa,ffa) :2	| fa/x
P(ffa,ffa) :2	P(fa,ffa) :2	P(fa,fffa) :3	| ffa/x
P(fffa,ffa) :3	P(fa,fffa):3	P(fa,ffffa):4	| fffa/x
...	...	...	...

As we can see, syntactically equal literals in the Herbrand expansion can have different indexes, since the index of an occurrence of a literal in the expansion depends on a corresponding literal in the unexpanded clause set.

LEMMA 4.1.: Let $(L_1', \ldots, L_n')$ and $(R_1', \ldots, R_m')$ be two clauses in the Herbrand expansion of S, corresponding to $(L_1, \ldots, L_n)$ and $(R_1, \ldots, R_m)$ in S. Let $L_1' = R'^{d}_1$ Then, if the clause $C = (L_2, \ldots, L_n, R_2, \ldots, R_m)\sigma$, where $\sigma = m.g.u(L_1, R_1^d)$ is not ground, the index of L_1' must be the same as the index of R_1'.

Proof:

Suppose that C is not ground. Then, due to property (B), both L_1 and R_1 contain variables. Let x be the variable in L_1, y be the variable in R_1. Consider the substitution σ = m.g.u(L_1, R_1^d). Since C is not ground, there must be at least one variable (either x or y) which is not bound to a ground term by σ . Suppose that x is a variable which is not bound to a ground term. Then we have three possibilities for binding of x in σ :

(1) y/x. Then the maximal depth of x is the same as the maximal depth of y and, since $L_1'=R_1'$, the term substituted for x in L_1' is the same as the term substituted for y in R_1'.

(2) f[y]/x, where f[y] is some term containing y in R_1. Then the depth of the term substituted for x in L_1' is the same as the depth of the term substituted for variable y in R_1' minus the maximal depth of y in f[y].

(3) g[x]/x, where g[x] contains x. This cannot happen by definition of a m.g.u.

Thus we get that the index of L_1' must be the same as the index of R_1'.

Q.E.D.

THEOREM 4.3.: The v-refinement of resolution is complete for E'.

Proof:

Induction base: Lifting of the given indexing ordering to the $>_V$-ordering for the originally given set of clauses is immediate.

Induction step: Let lifting of the ordering hold for non-ground clauses A and B. Let resolution generate a clause C from clauses A and B. If C is ground, we can, for example, use splitting (or some restrictions on C based on splitting). So the important case is where C is not ground. From lemma 1 we get that for any ground instances A' and B' in the Herbrand expansion the literals resolved upon have the same index. Then it is easy to see that ordering of the clause C', given by resolution from A' and B' is also lifted, that is, corresponds exactly to the $>_V$-ordering of a non-ground clause C.

Q.E.D.

As we proved completeness by using a lock-strategy for the ground case, we cannot eliminate tautologies and cannot use unrestricted subsumption, since lock-strategy is incompatible with these. However, we can do partial subsumption, where the ordering of the literals in the clause is also taken into consideration.

EXAMPLE 4.13.: Consider the following refutable set of four clauses from E':

1: (P(f(a),f(x)), R(x,f(f(a))))

2: (¬ R(a,f(x)), P(x,x))

3: (R(a,f(x)), ¬ P(x,x))

4: (¬P(f(a),f(x)), ¬ R(x,f(f(a))))

By the $>_V$-ordering only the first literals in these four clauses are allowed to be resolved upon. At the first level we can derive two new clauses, both of them tautologies:

1,4 give 5: (R(x, f(f(a))), ¬R(x, f(f(a))))

2,3 give 6: (P(x, x), ¬P(x, x))

The clauses derived at the next level are subsumed by clauses 1-4. However, when subsumption is restricted by the ordering, these clauses are not subsumed and the refutation follows easily.

4.5.3 Termination of the v-refinement for the Class $E^{+'}$

The following presents a proof that the v-refinement terminates on any clause set from the Class $E^{+'}$. Although the completeness of the v-refinement for Class $E^{+'}$ is still an open problem, we decided to present the termination proof for the following reasons: first, the techniques used in the proof could be useful elsewhere; second, since we have not found any counterexamples even to the general completeness of the v-refinement, it could be the case that v-refinement is indeed complete, and should the completeness proof be found in the future, the following termination proof can be put to an immediate use. Lemmas 4.2, 4.3 and 4.4 are analogous to the subcases of the lemma 5.5 in chapter 5.

LEMMA 4.2.: If clauses D_1 and D_2 both have the property (ii') (in the sense of the definition of Class E^+') and every literal in D_1 and D_2 is weakly covering, then all clauses inferred from D_1 and D_2 using v-refinement also have the property (ii') and every literal in them is weakly covering.

Proof:

1) Property (ii'): If (ii') holds for C, then (ii') holds for $C\sigma$, where C is any clause and σ is any substitution. If the clause C is obtained from clauses D_1 and D_2 with property (ii') by v-refinement, resolving upon literal L_1 in D_1 and literal L_2 in D_2, then L_1 contains all variables in D_1 and L_2 contains all variables in D_2.

2) Weakly covering: Obviously factorization preserves weakly covering (the value of the m.g.u, used for factorizing D_1, cannot have elements of the kind t/x, where t contains variables, since then by weakly covering and (ii') t must also contain x). Next we will consider a binary resolution step.

   In order that weakly covering were not preserved, such a substitution σ must be applied to some literal L in D_1 (let $\bar{x} = \{x_1, ..., x_k\} = V(L)$) that σ contains t/x_i $(1 \leq i \leq k)$, t contains variables and L contains some variable $x_j \neq x_i$ such that $L\sigma$ also contains x_j and t does not contain x_j. $\sigma = m.g.u(L_1, L_2^d)$, where by property (ii') and v - refinement L_1 contains variables $x_1, ..., x_k$. Let $y = \{y_1, ..., y_r\} = V(L_2)$.

   Suppose that σ contains $t[\bar{y}']/x_i$ $(1 \leq i \leq k)$ (where $\bar{y}'$ is a subset of variables in L_2) and L contains a variable $x_j = x_i$. We will examine whether $L\sigma$ can contain x_j. For this we will investigate possible cases for some term d in L_2 which corresponds to some occurrence of x_j in L_1.

   Consider the following cases:

   - x_j occurs in some term $g[\bar{x}]$ in L_1'. This cannot happen by definition of a m.g.u.

   - $d = y_n$ for some y_n in $\bar{y}$. Then either $t\sigma$ contains x_j or $L\sigma$ does not contain x_j.

   - d is a ground term. Then $L\sigma$ does not contain x_j.

   - $d = g[\bar{y}]$ for some non-ground term $g[\bar{y}]$. Then $L\sigma$ does not contain x_j.

Q.E.D.

LEMMA 4.3.: If clauses D_1 and D_2 both have the property (ii') and every literal in them is weakly covering, then for any clause D inferred from D_1 and D_2 with a factorization or binary resolution step of v-refinement $\tau_V(D) \leq \max(\tau_V(D_1), \tau_V(D_2))$.

Proof:
Let D be a clause inferred from D_1 and D_2 using the v - refinement. D can contain some variable deeper than any variable in D_1 and D_2 only in case a substitution used for inferring D contains such an element $t[\bar{y}]/x_i$ (where $\bar{y}$ is some set of variables) that some literal $L\sigma$ in D is such that x_i occurs in L (L is the original of $L\sigma$ in D_1 or D_2) in some term g and σ does not bind all variables in y to ground terms. We will show that this is impossible.

1. Inference by factorization. Since (ii') holds for D_1 and every literal in D_1 is weakly covering, factorization cannot give such a substitution σ that contained $t[\bar{y}]/x_i$ (then $\bar{y}$ should contain x_i, which is impossible).

2. Inference by resolution. Let $L, L_1 \in D_1$, $L_2 \in D_2$.
   $V(D_1) = \{x_1, \ldots, x_k\}$, $V(D_2) = \{y_1, \ldots, y_r\}$.
   Let $\sigma = m.g.u(L_1, L_2^d)$.

Since we use the v - refinement, the variable x_i cannot occur deeper in L than in L_1. Let $f[x_i]$ be the term in L_1 where x_i has its deepest occurence in L_1. We can assume that D_2 does not contain variables deeper than D_1. Due to the v-refinement, $f[x_i]$ has the deepest occurrences of variables in D_1, D_2.

Suppose that σ contains the element $t[\bar{y}]/x_i$. Then during the computation of σ the term $f[x_i]\{t[\bar{y}]/x_i\} = f[t[\bar{y}]]$ must have been unified with some term d in L_2. Consider the following cases:

1) d is a variable. Then d cannot be unified with $f[t[\bar{y}]]$, since $\bar{y}$ contains d.

2) d is a ground term. Then every variable in $\bar{y}$ is bound to some ground term, thus σ cannot contain an element $t[\bar{y}]/x_i$ s.t. variables in $\bar{y}$ were not bound to some ground term.

3) d = f[g], g is either a variable or a ground term and g occurs in the term f[g] on the same place as $t[\bar{y}]$ first occurs in $f[t[\bar{y}]]$. If g is a variable, then d cannot be unified with $f[t[\bar{y}]]$ since $\bar{y}$ contains g. If g is a ground term, then every variable in $\bar{y}$ is bound to some ground term, thus σ cannot contain an element $t[\bar{y}]/x_i$ s.t. variables in $\bar{y}$ were not bound to some ground term.

4) d = $f[t'[\bar{y}]]$, where $t'[\bar{y}]$ occurs in $f[t'[\bar{y}]]$ on the same place as $t[\bar{y}]$ first occurs in $f[t[\bar{y}]]$. Suppose that $t[\bar{y}]$ and $t'[\bar{y}]$ are unifiable. In case $t[\bar{y}]$ contains non-ground functional terms as proper subterms, then iterate our analysis (notice that if some literal L is weakly covering, then any subterm of L is weakly covering) until such a term $r[\bar{y}]$ is reached which does not contain non-ground functional terms as proper subterms.

Therefore, in order that σ existed and $L\sigma$ contained some variables in $\bar{y}$, L_2 must contain such a term $f[t'[\bar{y}]]$ that the maximal depth of variables in $f[t'[\bar{y}]]$ were not smaller than the maximal depth of variables in $f[t[\bar{y}]]$. Since $f[x_i]$ is assumed to contain the deepest occurrence of variables in D_1 and D_2, $f[t[\bar{y}]]$ contains variables deeper than the maximal depth of variables in D_1 and D_2, which is impossible. Thus no variable in $D\sigma$ may occur deeper than the depth of the deepest variable in D_1, D_2.

Q.E.D.

Corollary 4.1.: v-refinement of the resolution method cannot infer a clause D from the clause set S in class E+' such that any variable occurred deeper in D than the maximal depth of variables in S.

Proof:
immediate from lemmas 3 and 4.

Q.E.D.

Let S be some clause set in Class E^+'. Let T be a set of all non-ground functional terms in S (if T is empty, the proof of the forthcoming theorem 4.4 is trivial). Let L be some literal in S. Since T is a finite set, there must be a finite set L^* of such literals which can be obtained from L by applying substitutions of the kind $\{t_1/x_1, \ldots, t_k/x_k\}$ (where $\bar{x} = \{x_1, \ldots, x_k\}$ is some set

of variables, and t_i denote arbitrary elements of T) to L so that no variable occurs deeper in L^* than $\tau_V(S)$. (T and L^* are defined up to renaming the variables).

EXAMPLE 4.14.: $S = \{\{A(f(x)),\ L(x, f(a))\},\ \{\neg P(f(g(u, y)), u)\}\}$
Then $\tau_V(S) = 2$, $T = \{f(x),\ f(g(u,y)),\ g(u,y)\}$.

If $L = A(f(x))$, then
$L^* = \{A(f(x)),\ A(f(g(x,y))),\ A(f(f(x)))\}$. If $L = \neg P(f(g(u,y)), u)$, then
$L^* = \{\neg P(f(g(u,y)),u)\}$.

DEFINITION 4.9.: Let L be some literal such that we can compute L^*. VNmax(L) denotes the maximal number of different variables in literals in L^*.

EXAMPLE 4.15.: Let S be the same as in the previous example.

$$\begin{aligned} VNmax(A(f(x)) &= 2,\\ VNmax(A(f(f(x))) &= 1,\\ VNmax(A(f(a))) &= 0,\\ VNmax(\neg P(f(g(u,y)), u)) &= 2,\\ VNmax(A(f(f(a)))) &= 0,\\ VNmax(A(x)) &= 4 \ \{\text{Notice } A(g(g(u'', y''), g(u', y')))\}. \end{aligned}$$

LEMMA 4.4.: Let S be a clause set in E+'. The v-refinement of resolution cannot infer a clause $(L_1', \ldots, L_n')$ from S such that for some $L_i'(1 < i < n)$; $VNmax(L_i') > VNmax(L_i)$, where L_i is the original of L_i'.

Proof:
Immediate from corollary 1 and the definition of VNmax.

Q.E.D.

LEMMA 4.5.: Let clauses D_1 and D_2 have the property (ii') and each literal in D_1, D_2 be weakly covering. Let the v-refinement infer a new clause $D = (L_1', \ldots, L_n')$ from D_1 and D_2 such that D contains a term deeper than the deepest term in

D_1, D_2. Then:

$$\forall_i (1 < i < n) \begin{cases} VNmax(L_i') = 0 & \text{if } VNmax(L_i) = 0 \\ VNmax(L_i') < VNmax(L_i) & \text{if } VNmax(L_i) \neq 0 \end{cases}$$

where L_i the original of L_i' in D_1 or D_2, VNmax is computed in respect to the original formula in E+' from which D_1, D_2 were inferred.

Proof:

Due to lemma 4.4 the only possibility for some L_i' to contain a term deeper than the deepest term in D_1 and D_2 is to substitute some ground term t for such a variable x_i in L_i such that x_i occurs in some functional term $f[x_i]$ in L_i. Then x_i will occur in no term in D, thus the VNmax of all literals in D_1 which contain variables (VNmax = 0) is bigger than VNmax of the corresponding literals in D.

Next we will consider the clause D_2, which did not contain x_i. Let x_i have a deepest occurrence in L_1 in a term $g[x_i]$. Notice that after applying a substitution $\{t/x_i\}$ to the literal L_1 in D_1 (assume L_1 is resolved upon in D_1) the term $g[x_i]$ will turn into $g[t]$, which by assumption is deeper than the deepest one in D_1, D_2 (since we use v-refinement, x_i has a deepest occurrence in L_1) . Let d be the term which occurs in R_1 (assume R_1 is resolved upon in D_2) on the place of the first occurence of $g[x_i]$ in L_1. Consider the following cases:

- d = g[t]. This is impossible, since g[t] is deeper than the deepest term in D_1, D_2.

- d = $g[y_j]$. Then literals in D which correspond to literals in D_2 will contain a ground term on place of a variable y_j, thus VNmax of these literals in D will be less than VNmax of their originals in D_2.

- d is a variable. Then literals in D which correspond to literals in D_2 will contain a ground term on place of a variable d, thus VNmax of these literals in D will be less than VNmax of their originals in D_2.

Q.E.D.

THEOREM 4. 4.: The v-refinement of the resolutiom method infers a finite number of clauses from any clause set in the Class $E^{+'}$.

Proof:
Let S be a clause set in the class $E^{+'}$. Due to corollary 4. 1 no clause inferred from S with the v-refinement method with V-strategy can contain variables deeper than $\tau_V(S)$. All literals in S have a certain VNmax. Lemmas 4. 4 and 4. 5 demonstrate that if a depth of a literal becomes bigger than a certain given bound for S, then VNmax of a literal will be correspondingly smaller. By lemma 4. 4 VNmax of a literal cannot grow. Thus such a bound on the depth of terms must exist that all literals of this depth have a VNmax equal to 0, thus they do not contain variables and their depth cannot grow.

Therefore by v-refinement one cannot infer literals deeper than the before-mentioned depth bound. Thus the number of inferrable literals (up to renaming variables) is also bounded. Then, due to lemma 4. 3, the length of the inferrable clauses is bounded and thus the number of inferrable clauses.

Q.E.D.