MAX-PLANCK-INSTITUT FÜR INFORMATIK

Minimal Resolution

Christoph Weidenbach

MPI–I–94–227 December 1994

INFORMATIK

Im Stadtwald
D 66123 Saarbrücken
Germany

Author's Address

Christoph Weidenbach (weidenb@mpi-sb.mpg.de),
Max–Planck–Institut für Informatik
Im Stadtwald
D-66123 Saarbrücken
Germany

Publication Notes

This report has been submitted for publication elsewhere and will be copyrighted if accepted.

Acknowledgements

Thanks to Ulrich Hustadt and Andreas Nonnengart for their comments on earlier versions of this report.

Abstract

Minimal resolution restricts the applicability of resolution and factorization to minimal literals. Minimality is an abstract criterion. It is shown that if the minimality criterion satisfies certain properties, minimal resolution is sound and complete. Hyper-resolution, ordered resolution and lock resolution are known instances of minimal resolution. We also introduce new instances called list resolution and tuple resolution. In addition, we investigate the compatibility of some important redundancy criteria with minimal resolution.

1 Introduction

The idea of minimal resolution is to restrict resolution steps and factorization steps to steps on "minimal" literals. Minimality is defined with respect to clauses. We can think of minimal resolution as a selection strategy for resolution. There are several of these strategies known, including hyper-resolution [13], ordered resolution [2, 3], lock resolution [5] and many others (see [6, 10]). Hyper-resolution and ordered resolution can be proved complete by using an ordering on the literals. A literal is minimal in a clause if there is no smaller literal in the clause with respect to the ordering. For hyper-resolution negative literals are always smaller than positive literals and for ordered resolution an arbitrary total (well-founded) ordering on ground literals is needed. For lock resolution the situation is different. Lock resolution assigns integers as indexes to literals and a literal is minimal in a clause if it has lowest index. This allows the minimality criterion to distinguish between literals which are syntactically equal.

Minimal resolution unifies and refines all these approaches. Therefore additional structure on the literals is needed, because we must be able to distinguish literals which are syntactically equal. We call this additional structure *literal occurrences*. A literal occurrence is a pair (*literal*, *natural number*). Then a clause is a multiset of literal occurrences. Now we assume a quasi-ordering $\preceq$ on occurrences. An occurrence is minimal in a clause if there is no smaller occurrence in the clause with respect to $\preceq$. Minimal resolution and minimal factorization are restricted to minimal literals. Soundness of the calculus is guaranteed by mapping a multiset of occurrences to the respective multiset of literals. Completeness of the resulting calculus is proved by ground completeness and lifting. In order to prove ground completeness, induction on the k-parameter [1] is used. In the induction step of the k-parameter proof, the current clause is split into two disjoint clauses. Then the proofs for the corresponding clause sets are combined to a proof including the current clause. For minimal resolution this combination is not a priori possible. The splitting has to be compatible with the minimality criterion and the minimal resolution and minimal factorization rule. Lifting is also not a priori possible, because $\preceq$ must be stable with respect to ground instantiations of occurrences. If $\preceq$ is a stable quasi-ordering, then minimal resolution is complete. The required properties on $\preceq$ are minimal in the sense that if $\preceq$ does not satisfy any of these properties minimal resolution is not complete.

Now the paper is organized as follows. In section 2 we introduce the needed notions and prove soundness and completeness of minimal resolution. Then in section 3 we show that most of the known ordering refinements are instances of this result. Although some of these approaches are defined with respect to a maximality criterion, for better readability we present them with respect to a minimality criterion. We also correct some flaws in existing literature and give some new insights in known results. Section 4 is devoted to instances of minimal resolution which are new or refinements of results presented in section 3. For many of the known strategies only completeness results have been presented. In section 5 we investigate the compatibility of several reduction criteria with minimal resolution. Subsumption and tautology deletion in the usual [6] sense are not compatible with minimal resolution. However, a restricted version of subsumption is compatible with minimal resolution. We end the paper with a discussion of the achieved results, section 6.

2 Minimal Resolution

A *multiset* over a set A is a function m from A to the natural numbers. Intuitively, $m(a)$ specifies the number of occurrences of a in m. We say that a is an element of m, $a \in m$, if $m(a) > 0$. A multiset m is *finite* if $m(a) \neq 0$ only for finitely many $a \in A$. Two multisets m_1, m_2 are equal, $m_1 = m_2$, if $m_1(a) = m_2(a)$ for all $a \in A$. The union, intersection, difference, and subset relationship of multisets are defined by the identities $m_1 \cup m_2 := m_1(x) + m_2(x)$, $m_1 \cap m_2 := \min(m_1(x), m_2(x))$, $m_1 \setminus m_2 := \max(0, m_1(x) - m_2(x))$, and $m_1 \subseteq m_2$ if $m_1(a) \leq m_2(a)$ for all $a \in A$. The cardinality of a multiset is given by $|m| := \sum m(x)$. Specific multisets are written in a set like notation, e.g. $\{a, a\}$ is the multiset m with $m(a) = 2$ and $m(b) = 0$ for all $b \neq a$.

A reflexive, transitive relation $\preceq$ over a set A is called a *quasi-ordering*. If $a \preceq b$ or $b \preceq a$ for all $a, b \in A$ it is called *total* over A.

The standard syntax and semantics of first-order logic are used. Terms, literals, formulae and substitutions are defined in the usual way. $\mathcal{L}$ is the set of all first-order literals. We call a substitution σ *ground for some literal* L, if $L\sigma$ is ground. We call a substitution σ *in the variables of some literal* L, if the domain of and the codomain variables of σ are contained in the variables of L. These two definitions can be naturally extended to literal occurrences, clauses (see below).

Now we define literal occurrences. A *literal occurrence* o is a pair (L, n) where $L \in \mathcal{L}$ and $n \in \mathbb{N}$. $\mathcal{L}^* = \mathcal{L} \times \mathbb{N}$ is the set of all occurrences. The two functions *lit*, *id* defined by $\mathit{lit}((L, n)) = L$ and $\mathit{id}((L, n)) = n$ map an occurrence to its literal and natural number, respectively. If $\mathit{id}(o) = n$ then we say that o has *identification* n. Two occurrences o, p are equal, $o = p$, if $\mathit{lit}(o) = \mathit{lit}(p)$ and $\mathit{id}(o) = \mathit{id}(p)$. Instantiation of occurrences by a substitution σ is given by $(L, n)\sigma = (L\sigma, n)$. An occurrence o is called *ground* if $\mathit{lit}(o)$ is ground. Two occurrences are called complementary (unifiable) if their respective literals are complementary (unifiable).

We assume a quasi-ordering $\preceq$ over $\mathcal{L}^*$. If $o \preceq p$ we say that o is *smaller* than p. We define $o \prec p$ for $o, p \in \mathcal{L}^*$ if $o \preceq p$ and not $p \preceq o$. Clauses are finite multisets over $\mathcal{L}^*$. For $C = \{\}$ we write $\Box$. If $|C| = 1$ we call C a *unit* clause[1]. C^* is the set of all clauses over $\mathcal{L}^*$. Instantiation of clauses is defined in the usual way. Since clauses are multisets instantiation cannot cause mergings between occurrences. The function *lit* can be extended to clauses in the usual way. Thus $\mathit{lit}(C)$ is the multiset m over $\mathcal{L}$ with $m(L) = \sum C(o)$ where $\mathit{lit}(o) = L$. An occurrence o is called *minimal* in a set (multiset) A of occurrences if there is no occurrence $p \in A$ with $p \prec o$. o is called *maximal* in A if there is no occurrence $p \in A$ with $o \prec p$. A quasi-ordering $\preceq$ over $\mathcal{L}^*$ is called *stable*, if $o \prec p$ implies $o\sigma \prec p\sigma$ for any two occurrences $o, p \in \mathcal{L}^*$ and any ground substitution σ for o, p.

Assumption 2.1 From now on we assume that $\preceq$ is a stable quasi-ordering over $\mathcal{L}^*$

Lemma 2.2 (Properties of $\preceq$)
Let σ be a ground substitution for a clause C. If $o\sigma$ is minimal in $C\sigma$, then o is minimal in C.

Proof: By contradiction. Assume $o\sigma$ is minimal in $C\sigma$, but o is not minimal in C. Then by the definition of minimality there is a $q \in C$ with $q \prec o$. As $\preceq$ is stable, Assumption 2.1, we have $q\sigma \prec o\sigma$ which contradicts that $o\sigma$ is minimal in $C\sigma$. $\Box$

Definition 2.3 (Minimal Resolution and Factorization) The rules are

[1] Remember that clauses are multisets and we defined the cardinality of multisets.

$$\text{Resolution} \qquad \frac{\{o_1,\dots,o_n\} \quad \{p_1,\dots,p_m\}}{\{o_2,\dots,o_n\}\sigma \cup \{p_2,\dots,p_m\}\sigma}$$

where σ is the mgu such that $o_1\sigma$ and $p_1\sigma$ are complementary, $o_1\sigma$ is minimal in $\{o_1,\dots,o_n\}\sigma$, $p_1\sigma$ is minimal in $\{p_1,\dots,p_m\}\sigma$.

We define two variants of factorization. One variant which deletes the minimal occurrence in the parent clause and one which deletes the occurrence unified with the minimal occurrence.

$$\text{Factorization I} \qquad \frac{\{o_1,\dots,o_n\}}{\{o_2,\dots,o_n\}\tau}$$

$$\text{Factorization II} \qquad \frac{\{o_1,\dots,o_n\}}{\{o_1,\dots,o_{j-1},o_{j+1},\dots,o_n\}\tau}$$

where τ the mgu of o_1 and o_j $(2 \leq j \leq n)$, $o_1\tau$ is minimal in $\{o_1,\dots,o_n\}\tau$. Any variant of the factorization rule is called *strict*, if both $o_1\tau$ and $o_j\tau$ are minimal in $\{o_1,\dots,o_n\}\tau$.

Lemma 2.4 (Soundness) Minimal resolution and minimal factorization are sound.
Proof: By definition, *lit* is a homomorphic embedding into the usual definitions of resolution and factorization. $\Box$

For completeness the choice of one variant of the factorization rule (strict or not) is sufficient. This will be proved in the following. The two versions of the factorization rule are introduced, because the ordering of occurrences in the factor may be different depending on the deleted literal. As usual, if R is a minimal resolvent (or minimal factor) of some clauses C, D, then C (or D) is called a *parent* of R.

A *minimal derivation* of a clause C^n with respect to a clause set CS is a finite sequence of clauses $C^1,\dots,C^n$ such that:

1. For all i either C^i is a minimal factor of some clause $C \in (CS \cup \{C^j \mid j < i\})$ or C^i is a minimal resolvent of some clauses C_1, $C_2 \in (CS \cup \{C^j \mid j < i\})$.

2. For all $i < n$, C^i is a parent clause of some C^j, $j > i$.

3. For all i, $C^i \notin CS$.

We call a minimal derivation $C^1,\dots,C^n$ *strict* if in addition to the above requirements we have for all $1 \leq i \leq n$:

4. Only strict factorization is used.

5. No C^i is the strict factor of some clause $C \in CS$.

6. If C^i is the strict factor of some clause C^j, $j < i$, using the occurrences $o, p \in C^j$, where wlog. o is removed from C^j, then the next step applied to C^i is either a strict factorization step using p or is a minimal resolution step using p.

7. If a sequence of strict factorization steps is applied to a clause C^i using the occurrences $o_1,\dots,o_m \in C^i$, then all o_i are minimal in C^i.

A (strict) *minimal refutation* is a (strict) minimal derivation of $\Box$.

Lemma 2.5 (Ground Derivations) Let CS be a set of ground clauses and let $C^1, \ldots, C^n$ be a minimal derivation of C^n with respect to CS. Let o be a maximal occurrence with respect to all clauses $D \in CS$ with $|D| \geq 2$ and let o be maximal with respect to all occurrences of some clause $C \in CS$. If we define $C' = C \cup \{o\}$, $CS' = (CS \setminus \{C\}) \cup \{C'\}$, then there is a minimal derivation $C'^1, \ldots, C'^n$ from CS' such that $C'^n = C^n \cup \{o, \ldots, o\}$. In addition, if $C^1, \ldots, C^n$ is a strict derivation, then $C'^1, \ldots, C'^n$ is strict, too.
Proof: First, we show that the derivation of C^n can be repeated using CS' instead of CS. This can be proved by an induction argument on the length n of the derivation. Since o is maximal with respect to C, if some occurrence p is minimal in C then p is minimal in C'. It remains to show that if an occurrence p is minimal in some clause C^i then p is also minimal in the clause C'^i. But this is obvious, because the extra occurrence o is also maximal for all clauses $D \in CS$ with $|D| \geq 2$ and all other unit clauses D play no role, because their occurrences disappear after the application of minimal resolution. Thus we can repeat the derivation of C^n using C' instead of C. It is easy to see that the resulting derivation is a minimal derivation, that $C'^i = C^i \cup \{o, \ldots, o\}$ and that if $C^1, \ldots, C^n$ is strict, then $C'^1, \ldots, C'^n$ is strict, too. □

Lemma 2.6 (Ground Completeness) If CS is a finite unsatisfiable set of ground clauses, then there is a minimal refutation from CS.
Proof: By induction on the k-parameter, $k(CS) := \sum\{(|C| - 1) \mid C \in CS\}$. If $k(CS) = 0$ then there are two clauses C_1, C_2 with $C_1 = \{o_1\}$, $C_2 = \{o_2\}$, $lit(o_1) = \neg P(t_1, \ldots, t_n)$ and $lit(o_2) = P(t_1, \ldots, t_n)$. By definition o_1 is minimal in C_1 and o_2 is minimal in C_2. Thus the minimal resolution step between C_1 and C_2 yields a minimal derivation of the empty clause.

If $k(CS) > 0$ then there exists at least one clause $C \in CS$ with $|C| \geq 2$. Now we select a clause C, $|C| \geq 2$ such that C contains a maximal occurrence o with respect to all occurrences in clauses $D \in CS$ with $|D| \geq 2$. Now we split C into the clauses $C_1 = C \setminus \{o\}$ and $C_2 = \{o\}$ and CS into the clause sets $CS_1 = (CS \setminus \{C\}) \cup \{C_1\}$ and $CS_2 = (CS \setminus \{C\}) \cup \{C_2\}$. By induction hypothesis there are minimal refutations for CS_1 and CS_2, because $k(CS_1) < k(CS)$ and $k(CS_2) < k(CS)$. These are combined to a minimal refutation for CS. First, the refutation of CS_1 is performed using C instead of C_1. This is possible by lemma 2.5. The result is either the empty clause or a clause $D = \{o, \ldots, o\}$. If the result is the empty clause, we are done. Otherwise we apply any variant of the factorization rule $D(o) - 1$ times[2] and get the clause $\{o\}$. Second, the refutation of CS_2 is repeated yielding a minimal refutation for CS. □

Corollary 2.7 (Strict Ground Completeness) If CS is a finite unsatisfiable set of ground clauses, then there is a strict, minimal refutation from CS.
Proof: In the proof of lemma 2.6 we actually constructed a strict, minimal refutation. The base case is solved by a minimal resolution step between clauses in CS, thus yielding a strict, minimal refutation. In the induction step, factorization is only used to derive a clause $\{o\}$ from some clause $D = \{o, \ldots, o\}$. All o are minimal in D and afterwards o is used by a minimal resolution step. □

Lemma 2.8 (Lifting) Let C_1, C_2 be two clauses with no common variables, σ be a ground substitution in the variables of C_1, C_2, $o\sigma$ minimal in $C_1\sigma$, $p\sigma$ minimal in $C_2\sigma$, $o\sigma$ and $p\sigma$ complementary. Let C_3 be a clause and τ be a ground substitution in the variables

[2] Remember that clauses are multisets. Thus $D(o)$ denotes how often o occurs in D.

of C_3, $r_1\tau$ minimal in $C_3\tau$, $r_2\tau \in C_3\tau$ and $lit(r_1\tau) = lit(r_2\tau)$. Then there exists a minimal resolvent R of C_1 and C_2 and a minimal factor F (F') of C_3 such that

1. $(C_1 \setminus \{o\})\sigma \cup (C_2 \setminus \{p\})\sigma)$ is an instance of R

2. $(C_3 \setminus \{r_1\})\tau$ is an instance of F ($(C_3 \setminus \{r_2\})\tau$ is an instance of F')

Proof: 1. By lemma 2.2 we have o is minimal in C_1 and p is minimal in C_2. There exists an mgu λ in the variables of C_1 and C_2 such that $o\lambda$ and $p\lambda$ are complementary. Now we select $R = (C_1 \setminus \{o\})\lambda \cup (C_2 \setminus \{p\})\lambda$. As λ is an mgu and λ, σ are substitutions in the variables of C_1 and C_2, we have $\sigma = \lambda\sigma$. Therefore $(C_1 \setminus \{o\})\sigma \cup (C_2 \setminus \{p\})\sigma$ is an instance of R by σ.

2. The proof of the second case is a variant of the first case. F is the result of an application of Factorization I and F' is the result of an application of Factorization II. Note that if $r_2\tau$ is minimal in $C_3\tau$, then r_2 is minimal in C_3 (lemma 2.2). Therefore strict factorization steps can be lifted, too. □

Theorem 2.9 (Minimal Resolution is Complete) If CS is an unsatisfiable clause set, then there exists a minimal refutation from CS.
Proof: As CS is unsatisfiable, there exists a finite, unsatisfiable set of ground clauses. Now lemma 2.6 and lemma 2.8 imply the existence of a minimal refutation from CS. □

Corollary 2.10 (Strict Minimal Resolution is Complete) If CS is an unsatisfiable clause set, then there exists a strict, minimal refutation from CS.
Proof: As CS is unsatisfiable, there exists a finite, unsatisfiable set of ground clauses. Now corollary 2.7 and lemma 2.8 imply the existence of a strict, minimal refutation from CS. □

3 Known Ordering Refinements of Resolution

In this section we show that various refinements of resolution are instances of minimal resolution. If we define $\preceq$ to be the trivial quasi-ordering $o \preceq p$ for all $o, p \in \mathcal{L}^*$, even standard resolution [14] is an instance of minimal resolution.

Some of the calculi presented here are based on a maximality criteria, i.e. resolution and factorization are restricted to maximal literals in a clause. In order to avoid confusion these calculi are translated into a form where resolution and factorization are defined with respect to minimal literals. Of course, the two formulations are equivalent.

In examples, we use a specific format to present resolution and factorization steps, for example

$$[(2)1, R, (9)2] \quad (11) \quad [R, \neg P]$$

names a minimal resolution step between the first literal of the second clause and the second literal of the ninth clause, yielding clause number eleven which is $[R, \neg P]$.

3.1 Ordered Resolution

For ordered resolution [2, 3] a stable, reduction ordering $\prec^r$ on atoms which is total on ground atoms (or an ordering which can be completed to a total ordering on ground atoms) is required. This ordering is lifted to literals. Literals are first compared with respect to their atoms and if their atoms are syntactically equal and the literals have different signs, the negative literal is smaller than the positive literal.

This ordering can be simulated by occurrences. We define $id(o) = 1$ if $lit(o)$ is positive and $id(o) = 0$ if $lit(o)$ is negative. Assume a function *atom* which maps a literal to its atom. Then we define $o \preceq p$ if $atom(lit(o)) \prec^r atom(lit(p))$ or $atom(lit(o)) = atom(lit(p))$ and $id(o) \leq id(p)$. Since $\prec^r$ is a reduction ordering, $\preceq$ is a stable quasi-ordering.

The ordered factorization rule is exactly the minimal factorization rule. But the ordered resolution rule is slightly more restrictive than the minimal resolution rule, because the ordered resolution rule requires the positive literal of the resolution step to be *strictly* minimal. A literal is *strictly* minimal in a clause, if there is no smaller literal and no syntactically equal literal. Of course, several syntactically equal minimal literals can be merged to one literal by factorization steps with identity substitutions. Then the remaining literal is strictly minimal. As all substitutions are identity substitutions the factorization rule can always be prefered to the resolution rule, not affecting completeness. Therefore minimal resolution is also complete with this refinement.

Theorem 3.1 (Ordered resolution is sound and complete) If CS is an unsatisfiable clause set, then there is a ordered refutation from CS.
Proof: By theorem 2.9, since $\preceq$ is a stable quasi-ordering and the above argumentation on strict minimal literals shows that minimal resolution can simulate ordered resolution. □

3.2 Lock Resolution

The idea of lock resolution [5] is to attach a natural number as an index to each literal of a clause. Then the minimal literals of a clause are the literals of lowest index. The literals in resolvents inherit their indices from their parent clauses.

Lock resolution is an instance of minimal resolution. Lock resolution uses the second variant of factorization. The index of a lock literal is the identification of the respective literal occurrence. Then we define $p \preceq r$ if $id(p) \leq id(r)$.

Theorem 3.2 (Lock resolution is sound and complete) If CS is an unsatisfiable clause set, then there is a lock refutation from CS.
Proof: Assumption 2.1 is satisfied. □

3.3 Hyper-Resolution

Hyper-resolution was introduced by Robinson [13, 6]. In contrast to minimal resolution, hyper-resolution is not a binary rule, i.e. a hyper-resolvent has more than one parent clause, in general. Nevertheless we can show that minimal resolution can simulate hyper-resolution in a binary way. This gives completeness of hyper-resolution.

Definition 3.3 (Hyper-Resolution) Let $C_1, \ldots, C_n$ be positive clauses, C'_i a factor of C_i, $K_i \in C'_i$,

$$\text{Hyper-Resolution} \quad \frac{C_1, \ldots, C_n, \{L_1, \ldots, L_m\}}{C'_1 \setminus \{K_1\} \cup \ldots C'_n \setminus \{K_n\} \cup \{L_{n+1}, \ldots, L_m\}\sigma}$$

Wlog. we assume σ the mgu such that each K_i and L_i are complementary and the L_i are exactly the negative literals of $\{L_1, \ldots, L_m\}$ ($1 \leq i \leq n$).

For hyper-resolution the identification of occurrences plays no role, e.g. we choose $id(o) = 1$ for all $o \in \mathcal{L}^*$. The ordering $\preceq$ is $o \preceq q$ if $lit(o)$ is negative and $lit(p)$ is positive. It is easy to verify that $\preceq$ is a quasi ordering and satisfies Assumption 2.1. Note that factorization of positive clauses is not restricted by the minimality criteria. Minimal resolution with respect to $\preceq$ is exactly what Robinson [13] called P_1-resolution. Thus the completeness of P_1-resolution is an instance of minimal resolution. The completeness of hyper-resolution is a straightforward consequence of the completeness of P_1-resolution. In fact, P_1-resolution simulates hyper-resolution in a binary way.

Theorem 3.4 (Hyper-Resolution) Hyper-resolution is complete.
Proof: For every unsatisfiable set of clauses there exists a minimal refutation (where $\preceq$ is defined as above) $C_1, \ldots, C_n$ with $C_n = \Box$. We define that C_n has depth 0, the parent clauses of C_n have depth 1 and so on. Using an induction argument it is sufficient to show that there is always a clause in the refutation which can be obtained by hyper-resolution. Each positive clause in the refutation derived by resolution and which has maximal depth can be obtained by hyper-resolution. $\Box$

3.4 Semantic Resolution

Sematic resolution was proposed by Slagle [15]. It is a generalization of hyper-resolution [13] (see Section 3.3).

Definition 3.5 (Semantic Resolution) Let $\mathcal{I}$ be an interpretation. Let $\mathbf{A}$ be an ordering of predicate symbols. A finite set of clauses $\{E_1, \ldots, E_n, N\}$, $n \leq 1$, is called a *semantic clash* with respect to $\mathbf{A}$ and $\mathcal{I}$ if and only if $E_1, \ldots, E_n$ and N satisfy the following conditions:

1. $E_1, \ldots, E_n$ are false in $\mathcal{I}$
2. Let $R_1 = N$. For each $i = 1, \ldots, n$ there is a resolvent R_{i+1} of R_i and E_i or a resolvent of factors R'_i and E'_i of R_i and E_i, respectively.
3. The literal in E_i, which is resolved upon, contains the largest predicate symbol in E_i (E'_i), $1 \leq i \leq n$
4. R_{n+1} is false in $\mathcal{I}$

R_{n+1} is called a semantic resolvent of the semantic clash $\{E_1, \ldots, E_n, N\}$.

In general, minimal resolution cannot simulate semantic resolution. But if $\mathcal{I}$ has the property that if some clause C is true in $\mathcal{I}$ then there is a literal $L \in C$ which is true in $\mathcal{I}$, then semantic resolution can be simulated by minimal resolution. In this case we define $id(o) = 0$ if $lit(o)$ is true in $\mathcal{I}$ and $id(o) = 1$ otherwise, for any occurrence o. The ordering $\preceq$ is given by $o \preceq q$ if $id(o) < id(q)$ or $id(o) = id(q)$ and the predicate of $lit(o)$ is smaller than the predicate of $lit(p)$ with respect to $\mathbf{A}$. Now it is easy to verify that $\preceq$ is a stable quasi-ordering (Assumption 2.1). It is stable because if $lit(o)$ is true in $\mathcal{I}$, then $lit(o\sigma)$ is also true in $\mathcal{I}$ for any σ. Similar to hyper-resolution this proves the completeness of a binary simulation of semantic resolution. However, this result can again be lifted.

Theorem 3.6 (Semantic Resolution) Semantic resolution is complete.

Proof: For every unsatisfiable set of clauses there exists a minimal refutation (where $\preceq$ is defined as above) $C_1, \ldots, C_n$ with $C_n = \Box$. We define that C_n has depth 0, the parent clauses of C_n have depth 1 and so on. Using an induction argument it is sufficient to show that there is always a clause in the refutation which can be obtained by semantic resolution. Each clause in the refutation which is false, derived by resolution and which has maximal depth can be obtained by semantic resolution. The main reason is that true literals are always smaller than false literals. Therefore the first false literal is not accessible until all true literals are resolved away. In addition, false literals are accessed with respect to **A**. Thus a clause obtained by minimal resolution and which is false is the result of a semantic resolution step. $\Box$

3.5 Π-Orderings

Π-Orderings have been introduced by Maslov [11, 12]. Clauses are lists of literals. Maslov assumes a decidable predicate Π on clauses, which is true for at least one permutation of the literals in the clause. Clauses which satisfy Π are called Π-*ordered.* Π is called *acyclic* if for every Π-ordered clause C and every clause C' and substitution σ such that $C'\sigma$ is a sublist of C, the clause C' is Π-ordered. The minimal literal of a Π-ordered clause C is the first literal in C. Now resolution and factorization are restricted to the first literal of a Π-ordered clause. Maslov stated that if Π is acyclic, the resulting calculus is complete [12]. He calls the resulting calculus the Π-*strategy.* Unfortunately, this result is wrong. Even if Π is acyclic the Π-strategy is not complete. Consider the following unsatisfiable set of propositional clauses.

Example 3.7 For the following set of clauses we use list notation.

$$
\begin{array}{rl}
(1) & [P, Q] \\
(2) & [Q, R] \\
(3) & [R, W] \\
(4) & [\neg R, \neg P] \\
(5) & [\neg W, \neg Q] \\
(6) & [\neg Q, \neg R] \\
(7) & [R, \neg R] \\
(8) & [W, \neg P] \\
(9) & [\neg P, \neg Q] \\
(10) & [Q, \neg Q]
\end{array}
$$

There are no two clauses which contain the same literals. All clauses consist of exactly two literals. Thus there exists an acyclic Π which is true on exactly the above permutations for the respective literals of the clauses. However, no factorization step is possible and all possible resolution steps result in existing clauses.

$$\begin{array}{rrl}
[(1)1, \Pi\mathrm{R}, (9)1] & (10) & [Q, \neg Q] \\
[(2)1, \Pi\mathrm{R}, (6)1] & (7) & [R, \neg R] \\
[(3)1, \Pi\mathrm{R}, (4)1] & (8) & [W, \neg P] \\
[(4)1, \Pi\mathrm{R}, (7)1] & (4) & [\neg R, \neg P] \\
[(5)1, \Pi\mathrm{R}, (8)1] & (9) & [\neg P, \neg Q] \\
[(6)1, \Pi\mathrm{R}, (10)1] & (6) & [\neg Q, \neg R]
\end{array}$$

Of course, the Π-strategy is not an instance of minimal resolution. The problem is that the ordering defined by $o \preceq p$ if the clause $[\mathit{lit}(o), \mathit{lit}(p)]$ is Π-ordered, is not transitive, even if Π is acyclic. $\preceq$ is defined on the literal part of the occurrences. Thus we can choose arbitrary identifications for occurrences, e.g. $\mathit{id}(o) = 1$ for all $o \in \mathcal{L}^*$. Then the Π-ordered clauses (4), (6), (9) introduce the cycle $(\neg Q, 1) \preceq (\neg R, 1) \preceq (\neg P, 1) \preceq (\neg Q, 1)$. That means with respect to the transitive closure of $\preceq$, all literals in the clauses (4), (6), (9) are minimal. This allows for a proof.

$$\begin{array}{rrl}
[(2)1, R, (9)2] & (11) & [R, \neg P] \\
[(11)1, R, (4)1] & (12) & [\neg P, \neg P] \\
[(12)1, F, (12)2] & (13) & [\neg P] \\
[(13)1, R, (1)1] & (14) & [Q] \\
[(14)1, R, (6)1] & (15) & [\neg R] \\
[(15)1, R, (3)1] & (16) & [W] \\
[(16)1, R, (5)1] & (17) & [\neg Q] \\
[(17)1, R, (14)1] & (18) & \Box
\end{array}$$

Note that the clauses (4), (7) plus transitivity implies $(R, 1) \preceq (\neg P, 1)$.

Loveland [10] introduced an ordering concept similar to that of Maslov. He assumes an ordering rule O determining the ordering of the literals of a clause. An *O-clause* is a list of literals with the list order in agreement with a given ordering rule O. Resolution and factorization are restricted with respect to that ordering. Then Loveland introduced assumptions about O which allow to lift ground O-refutations to general O-refutations. However, these assumptions don't guarantee ground completeness. Many of the refinements presented in this chapter are also instances of O orderings. But ground completeness has to be proved for each refinement separately. This was done by Loveland [10].

Tammet [7, Chapter 4] gives a reformulation of Maslov's Π-orderings that is complete and an instance of minimal resolution. Clauses are multisets of literals and a decidable predicate $<$ is assumed, defined between literals, which satisfies the properties (A), (B) and (D):

(A) For each clause $C = \{L_1, \ldots, L_n\}$ there is at least one i such that $L_j \not< L_i$ holds for all $j \neq i$, $1 \leq j \leq n$.

(B) For all literals L, K, D: $L < K$ and $K < D$ implies $L < D$

(D) For any literals L, K and any ground substitution σ, if $L < K$ then $L\sigma < K\sigma$

Now we show that the completeness of a resolution strategy using $<$ is an instance of minimal resolution. As $<$ is defined on literals again all occurrences have same identification, e.g. $id(o) = 1$ for all $o \in \mathcal{L}^*$. Then we define $o \preceq p$ if $o = p$ or $lit(o) < lit(p)$. By definition and property (B), $\preceq$ is a quasi-ordering. Property (D) implies Assumption 2.1. It remains to show that a literal is minimal in a clause C with respect to $\preceq$ if there is no literal in $lit(C)$ which is smaller with respect to $<$. This follows by the definition of $\preceq$ and the definition of minimality. Therefore condition (A) is superfluous. The problem is that the fact that there is no literal which is smaller is confused with the fact that there is a literal which is greater or equal.

3.6 A-orderings

A-orderings were suggested by Slagle, Kowalski and Hayes [15, 9]. Given a set of clauses CS an **A**-*ordering* for CS is a total ordering $\leq$ on some subset of the set of literals $\{L\sigma \mid L \in C \text{ for some clause } C \in CS\}$ such that

(i) if $L < K$ then $L\sigma < K\sigma$ for all substitutions σ

(ii) if L and K are alphabetic variants or complements then $L \leq K$ and $K \leq L$

Now resolution is restricted to minimal literals with respect to $\leq$. In order to get the completeness of resolution with respect to **A**-orderings the identification of occurrences are not needed, e.g. $id(o) = 1$ for all $o \in \mathcal{L}^*$. Then we define $o \preceq p$ if $lit(o) \leq lit(p)$. As $\leq$ is a stable ordering $\preceq$ is a quasi-ordering and satisfies Assumption 2.1.

Theorem 3.8 Resolution with respect to **A**-orderings is complete. □

3.7 Sequencing

Sequencing is a refinement not based on an ordering but on the structure of the clauses in a given clause set. It was first suggested by Genesereth and Nilsson [8]. Clauses are sequences of literals. Then only the first literal of a clause can be used for a factorization or resolution inference step.

Definition 3.9 (Sequencing) Let $[L_1, \ldots, L_n]$, $[K_1, \ldots, K_m]$ and $[D_1, \ldots, D_h]$ be three clauses.

Positive Sequencing	$\dfrac{[L_1, \ldots, L_n] \; [K_1, \ldots, K_m]}{[L_2, \ldots, L_n, K_2, \ldots, K_m]\sigma}$
Negative Sequencing	$\dfrac{[L_1, \ldots, L_n] \; [K_1, \ldots, K_m]}{[K_2, \ldots, K_m, L_2, \ldots, L_n]\sigma}$
Factorization	$\dfrac{[D_1, \ldots, D_h]}{[D_1, \ldots, D_{j-1}, D_{j+1}, \ldots, D_h]\lambda}$

L_1 is a positive literal, σ the mgu such that $L_1\sigma$ and $K_1\sigma$ are complementary, λ the mgu such that $D_1\lambda$ and $D_j\lambda$ are equal.

The difference between positive and negative sequencing is the generation of the resolvent. Positive strict sequencing appends the literals of the clause with the leading positive literal with the clause with the leading negative literal and negative strict sequencing just the other way round. A *rigid* sequencing calculus consists of one of the rules positive, negative sequencing plus factorization. The sequencing calculus consists of all three rules. Genesereth and Nilsson stated that rigid sequencing is not complete in general, but is complete on horn clauses and that sequencing is complete. They didn't give any proof or counter example as justification for these statements. Example 3.7 is a counterexample for the general completeness of rigid sequencing. The clause set is saturated under positive sequencing just as it is saturated under the Π-ordering introduced there. Indeed, rigid sequencing is complete for Horn clauses. This result cannot be obtained by instantiating minimal resolution. However, we give an extra proof for that.

Theorem 3.10 Rigid sequencing is complete for Horn clauses.
Proof: We show by k-parameter induction the ground completeness of positive sequencing. Of course, the completeness of negative sequencing is a consequence of this result. Lifting is straightforward because instantiation does not change the ordering of a clause. The base case for the k-parameter induction is trivial. For the induction step we select a Horn-clause $C = [K_1, \ldots, K_m]$ such that $m > 1$. We split C into the clauses $C_1 = [K_1]$ and $C_2 = [K_2, \ldots, K_m]$. By induction hypothesis the two resulting clause sets are refutable with positive sequencing. It remains to show that the two refutations using C_1 and C_2, respectively, can be combined to a refutation using C.

If K_1 is negative, the refutation using C_1 is repeated with C. This is possible because all clauses are Horn, C_1 contains exactly one negative literal and positive sequencing appends the clause with the negative leading literal to the clause with the positive leading literal. The result of the repetition is not the empty clause, but C_2 which is refuted by repeating the second refutation.

If K_1 is positive, the combination of the two refutations is more complicated. We need an additional induction argument on the number of times C_1 is used in the proof. The first refutation is repeated until C_1 is used. Assume it is resolved with a clause $[L_1, \ldots, L_n]$, where L_1 is negative and K_1 and L_1 are complementary. Then the resolvent using C instead of C_1 is $[K_2, \ldots, K_m, L_2, \ldots, L_n]$. Now the second refutation is repeated. This is possible, because all K_i are negative, we only consider horn clauses and by the definition of positive sequencing. The result is $[L_2, \ldots, L_n]$ and now the first refutation is continued. By interleaving the two refutations this way (this may require several repetitions of the second refutation) the two refutations are combined to a refutation using C. □

4 New Ordering Refinements on Resolution

4.1 Incrementally Build Orderings

The idea of incrementally built orderings is not to fix $\preceq$ from the beginning but build it by the way of generating new clauses such that $\preceq^k$ is extendable to a stable quasi-ordering (Assumption 2.1). Therefore $\preceq$ is always the reflexive, transitive, stable closure of the current $\preceq^k$. We start with the initial clause set CS^0. In order to be as general as possible all occurrences in CS^0 have different identifications and the initial ordering $\preceq^0$ does not relate occurrences having the same identification. Even with this restriction all occurrences in CS^0 can be totally ordered such that there is exactly one minimal occurrence in each clause. Now the resolution rule of definition 2.3 and one variant of the factorization rule define as usual the rules of the calculus. However, we want to restrict the extension of the ordering such that the minimal literals of already generated clauses do not change by the way new clauses are generated and the ordering is extended. Note that the ordering of a factor is the same than for its parent, because $\preceq$ is always the stable extension of $\preceq^k$.

Definition 4.1 (List Resolution and List Factorization) Let $C_1 = \{o_1, \ldots, o_n\}$, $C_2 = \{p_1, \ldots, p_m\}$ be two clauses in CS^k and $R = \{o_2, \ldots, o_n\}\sigma \cup \{p_2, \ldots, p_m\}\sigma$ a minimal resolvent of C_1 and C_2 with respect to $\preceq^k$. Then we extend CS^k, $\preceq^k$ by

$$\begin{array}{rcl} CS^{k+1} & := & CS^k \cup \{R\} \\ \preceq^{k+1} & := & \preceq^k \cup \preceq^R \end{array}$$

where $\preceq^R$ is a minimal extension to $\preceq^k$ such that R is totally ordered by $\preceq^{k+1}$ and if $o_i\sigma \preceq^R p_j\sigma$ ($p_j\sigma \preceq^R o_i\sigma$) then there are no occurrences o, p such that $o\tau = o_i$, $p\lambda = p_j$ for two substitutions τ, λ and $p \preceq o$ ($o \preceq p$) with respect to $\preceq^k$.

For any variant of the factorization rule CS^k is extended by the factor and $\preceq^{k+1} = \preceq^k$ because the factor is already totally ordered with respect to $\preceq^k$.

Lemma 4.2 (Properties of $\preceq$ for List Resolution) If $o \prec p$ with respect to $\preceq^k$ then $o \prec p$ with respect to all $\preceq^j$, $j > k$.
Proof: By contradiction. Assume $o \prec p$ with respect to $\preceq^k$ and there is some minimal $j > k$ such that $p \preceq q$ with respect to $\preceq^j$. As j is minimal, $\preceq^j = \preceq^{j-1} \cup \preceq^R$ for some resolvent R and $o \prec p$ with respect to $\preceq^{j-1}$. □

Theorem 4.3 (List resolution is sound and complete) If CS is an unsatisfiable clause set, then there is a list refutation from CS.
Proof: It is sufficient to show that list resolution is an instance of minimal resolution, i.e. we must verify the assumptions of Assumption 2.1. They are all satisfied by definition of list resolution. Note that $\preceq$ can always be completed to a total quasi-ordering on ground occurrences. □

We will now give an example for list resolution. First, we show the example in an intuitive way, i.e. writing clauses as sequences. Then we show the simulation by minimal resolution. The example clause set is

$$\begin{array}{ll} (1) & [\neg R(a)] \\ (2) & [\neg R(b)] \\ (3) & [R(x), P(x), Q(x)] \\ (4) & [\neg P(a), \neg P(b)] \\ (5) & [\neg Q(a)] \\ (6) & [\neg Q(b)] \end{array}$$

The minimal literal is the first literal of a sequence. Resolvents are build by appending the sequence of the positive complementary literal to the sequence of the negative complementary literal. We call this selection strategy *strict sequencing*. In order to refer to clauses and literals the clause number and the position of the literal in the clause are used. For example (3)2 refers to the literal $P(x)$ of clause (3). The following strict sequencing (SS) refutation is possible:

$$\begin{array}{rcl} [(1)1, RS, (3)1] & (7) & [P(a), Q(a)] \\ [(2)1, RS, (3)1] & (8) & [P(b), Q(b)] \\ [(7)1, RS, (4)1] & (9) & [\neg P(b), Q(a)] \\ [(9)1, RS, (8)1] & (10) & [Q(a), Q(b)] \\ [(10)1, RS, (5)1] & (11) & [Q(b)] \\ [(11)1, RS, (6)1] & (12) & [] \end{array}$$

Now the refutation is simulated by list resolution. Instead of writing pairs, we just index the literals. The clause set CS^0 is:

$$
\begin{array}{ll}
(1) & \{\neg R(a)_1\} \\
(2) & \{\neg R(b)_2\} \\
(3) & \{R(x)_3, P(x)_4, Q(x)_5\} \\
(4) & \{\neg P(a)_6, \neg P(b)_7\} \\
(5) & \{\neg Q(a)_8\} \\
(6) & \{\neg Q(b)_9\}
\end{array}
$$

Now $\preceq$ is defined as follows:

$$\preceq^0 := \{R(x)_3 \preceq P(x)_4, P(x)_4 \preceq Q(x)_5, \neg P(a)_6 \preceq \neg P(b)_7\}$$

The list refutation is (LR means list resolution):

$$
\begin{array}{rrll}
[(1)1, LR, (3)1] & (7) & \{P(a)_4, Q(a)_5\} & \preceq^1 := \preceq^0 \\
[(2)1, LR, (3)1] & (8) & \{P(b)_4, Q(b)_5\} & \preceq^2 := \preceq^1 \\
[(7)1, LR, (4)1] & (9) & \{\neg P(b)_7, Q(a)_5\} & \preceq^3 := \preceq^2 \cup \{\neg P(b)_7 \preceq Q(a)_5\} \\
[(9)1, LR, (8)1] & (10) & \{Q(a)_5, Q(b)_5\} & \preceq^4 := \preceq^3 \cup \{Q(a)_5 \preceq Q(b)_5\} \\
[(10)1, LR, (5)1] & (11) & \{Q(b)_5\} & \preceq^5 := \preceq^4 \\
[(11)1, LR, (6)1] & (12) & \square &
\end{array}
$$

Note that we achieved an exact simulation of the strict sequencing refutation. In every clause there is always exactly one minimal literal. The example cannot be simulated by lock resolution because in clause (10) the literals $Q(a)$ and $Q(b)$ will have the same index. Therefore both will be minimal using lock resolution. Whence list resolution (minimal resolution) is a refinement of lock resolution. In Section 3.2 we have shown that lock resolution is an instance of minimal resolution. List resolution does not perfectly simulate rigid sequencing. This is not surprising, because rigid sequencing is not complete in general. Consider the unsatisfiable propositional example clause set [6, p. 116]:

$$
\begin{array}{ll}
(1) & [P, Q] \\
(2) & [Q, R] \\
(3) & [R, W] \\
(4) & [\neg R, \neg P] \\
(5) & [\neg W, \neg Q] \\
(6) & [\neg Q, \neg R]
\end{array}
$$

Rigid sequencing derives the clauses

$$
\begin{array}{rrl}
[(2)1, RS, (6)1] & (7) & [R, \neg R] \\
[(3)1, RS, (4)1] & (8) & [W, \neg P] \\
[(7)1, RS, (4)1] & (9) & [\neg R, \neg P] \\
[(8)1, RS, (5)1] & (10) & [\neg P, \neg Q] \\
[(10)1, RS, (1)1] & (11) & [Q, \neg Q]
\end{array}
$$

After the generation of clause (11) the set is saturated by rigid sequencing. The possible resolution step $[(11)1, RS, (6)1]$ results in (6) and the resolution step $[(9)1, RS, (7)1]$ results in (9).

We refute the clause set by list resolution:

$$
\begin{array}{ll}
(1) & \{P_1, Q_2\} \\
(2) & \{Q_3, R_4\} \\
(3) & \{R_5, W_6\} \\
(4) & \{\neg R_7, \neg P_8\} \\
(5) & \{\neg W_9, \neg Q_{10}\} \\
(6) & \{\neg Q_{11}, \neg R_{12}\}
\end{array}
$$

We start with the ordering

$$\preceq^0 := \{P_1 \preceq Q_2, Q_3 \preceq R_4, R_5 \preceq W_6, \neg R_7 \preceq \neg P_8, \neg W_9 \preceq \neg Q_{10}, \neg Q_{11} \preceq \neg R_{12}\}$$

$$
\begin{array}{rcll}
[(2)1, LR, (6)1] & (7) & \{R_4, \neg R_{12}\} & \preceq^1 := \preceq^0 \cup \{R_4 \preceq \neg R_{12}\} \\
[(3)1, LR, (4)1] & (8) & \{W_6, \neg P_8\} & \preceq^2 := \preceq^1 \cup \{W_6 \preceq \neg P_8\} \\
[(7)1, LR, (4)1] & (9) & \{\neg R_{12}, \neg P_8\} & \preceq^3 := \preceq^2 \cup \{\neg R_{12} \preceq \neg P_8\} \\
[(8)1, LR, (5)1] & (10) & \{\neg P_8, \neg Q_{10}\} & \preceq^4 := \preceq^3 \cup \{\neg P_8 \preceq \neg Q_{10}\} \\
[(10)1, LR, (1)1] & (11) & \{Q_2, \neg Q_{10}\} & \preceq^5 := \preceq^4 \cup \{Q_2 \preceq \neg Q_{10}\}
\end{array}
$$

Now exactly the clauses of the saturated rigid sequencing set are derived. The list resolution step $[(9)1, LR, (7)1]$ results in (9) because $\neg R_{12} \preceq \neg P_8$ is introduced in $\preceq^3$. But the list resolution step $[(11)1, LR, (6)1]$ produces $\{\neg R_{12}, \neg Q_{10}\}$, because $\neg R_{12} \preceq \neg P_8 \preceq \neg Q_{10}$. This is the clause needed to refute the clause set.

$$
\begin{array}{rcll}
[(11)1, LR, (6)1] & (12) & \{\neg R_{12}, \neg Q_{10}\} & \preceq^6 := \preceq^5 \\
[(12)1, LR, (3)1] & (13) & \{W_6, \neg Q_{10}\} & \preceq^7 := \preceq^6 \\
[(13)1, LR, (5)1] & (14) & \{\neg Q_{10} \neg Q_{10}\} & \preceq^8 := \preceq^7 \\
[(14)1, LF, (14)2] & (15) & \{\neg Q_{10}\} & \\
[(15)1, LR, (2)1] & (16) & \{R_4\} & \\
[(16)1, LR, (4)1] & (17) & \{\neg P_8\} & \\
[(17)1, LR, (1)1] & (18) & \{Q_2\} & \\
[(18)1, LR, (15)1] & (19) & \square &
\end{array}
$$

The example shows that list resolution is very close to the border of complete calculi. In fact, we have not been able to find a refinement of list resolution which we could prove complete.

4.2 Tuple Resolution

The idea of tuple resolution is to separate $\mathcal{L}$ into a disjoint partition $\mathcal{L} = \mathcal{L}_1 \cup \ldots \cup \mathcal{L}_n$ such that each $\mathcal{L}_i$ $(1 \leq i \leq n)$ is closed under instantiation. Then clauses are n-tuples of multisets, $C = (C_1, \ldots, C_n)$. A literal L is minimal in such a clause C, if $L \in C_i$

and $C_j = \emptyset$ for $1 \leq j < i$, $1 \leq i \leq n$. Resolution between two tuples is performed by removing the complementary literals from the parent tuples and then building the union of the respective parts.

Tuple resolution can be precisely simulated by minimal resolution. As for list resolution occurrences are pairs of literals and natural numbers. An occurrence $o \in \mathcal{L}^*$ is a tuple $o = (L, m)$ where $L \in \mathcal{L}_m$. The ordering $\preceq$ is defined by $o \preceq p$ if $id(o) \leq id(p)$. As the $\mathcal{L}_i$ are closed under instantiation, $\preceq$ is stable. Thus tuple resolution is complete.

Theorem 4.4 (Tuple resolution is sound and complete) If CS is an unsatisfiable clause set, then there is a tuple refutation from CS.

As for hyper-resolution all negative occurrences (occurrences with a negative literal) are always smaller than positive occurrences, but neither negative nor positive occurrences are comparable to each other (except they are ground) hyper-resolution can be further refined. For example we extend $\preceq$ according to list resolution for the positive occurrences. This means in each clause containing negative occurrences all negative occurrences are minimal. If a clause consists of positive occurrences only the positive occurrences with respect to list resolution are minimal.

Corollary 4.5 (List Hyper-Resolution is sound and complete) If CS is an unsatisfiable clause set, then there is a list hyper refutation from CS.
Proof: As the sets of positive and negative occurrences are disjoint and closed under substitution soundness and completeness follows from theorem 4.4 and theorem 4.3. $\Box$

We refute the following clause set by list hyper-resolution.

$$
\begin{array}{ll}
(1) & \{R(x)_1, Q(a)_2\} \\
(2) & \{\neg Q(x)_3, R(x)_4\} \\
(3) & \{\neg R(a)_5, \neg S(a)_6\} \\
(4) & \{S(x)_7\}
\end{array}
$$

According to the definition of hyper-resolution we have $\preceq^0 = \{\neg Q(x)_3 \preceq R(x)_4\}$. Thus (1)1 and (1)2 are minimal in (1), (2)1 is minimal in (2), (3)1 and (3)2 are minimal in (3) and (4)1 is minimal in (4). Extending $\preceq$ according to list resolution for positive occurrences we get $\preceq^0 = \{\neg Q(x)_3 \preceq R(x)_4, R(x)_1 \preceq Q(a)_2\}$. Now only (1)1 is minimal in (1). We get the following unique refutation:

$$
\begin{array}{llllll}
(3) & \text{LHR} & (1)1(4)1 & (5) & \{[Q(a)]_8\} \\
 & & \preceq^1 & := & \preceq^0 \\
(2) & \text{LHR} & (5)1 & (6) & \{[R(a)]_9\} \\
 & & \preceq^2 & := & \preceq^1 \\
(3) & \text{LHR} & (6)1(4)1 & (7) & \Box
\end{array}
$$

Applying hyper-resolution to the above clauses the hyper-resolvent (2)HR(1)2 is also possible. Thus list hyper-resolution is a refinement of hyper-resolution.

4.3 New Combinations of Known Refinements

There are several possibilities to define new, complete refinements. As an example we define *ordered lock resolution* by a combination of lock resolution and ordered resolution. The quasi-ordering $\preceq$ is given by $o \preceq q$ if $id(o) < id(p)$ (lock part) or $id(o) = id(p)$ and $atom(lit(o)) \prec^r atom(lit(p))$ (ordered part) or $o = p$. $\preceq$ is a stable quasi-ordering.

Theorem 4.6 (Ordered Lock Resolution is Sound and Complete) If CS is an unsatisfiable clause set, then there is a ordered lock refutation from CS.

5 Redundancy

For many of the calculi which we showed to be instances of minimal resolution only completeness results have been known so far. Here we show that a specific form of subsumption is compatible with minimal resolution. The standard notion of subsumption which is only based on the literals of occurrences is not compatible with minimal resolution. Consider the following example [5]:

$$\begin{array}{ll} (1) & \{P_1, R_2\} \\ (2) & \{\neg R_3, P_4\} \\ (3) & \{R_5, \neg P_6\} \\ (4) & \{\neg P_7, \neg R_8\} \end{array}$$

If we refute this example with lock resolution, i.e. we define $o \preceq p$ if $id(o) \leq id(p)$, we generate the clauses

$$\begin{array}{lll} [(1)1, R, (4)1] & (5) & \{R_2, \neg R_8\} \\ [(2)1, R, (3)1] & (6) & \{P_4, \neg P_6\} \\ [(5)1, R, (2)1] & (7) & \{P_4, \neg R_8\} \\ [(6)1, R, (4)1] & (8) & \{\neg P_6, \neg R_8\} \end{array}$$

Now the clauses (5) and (6) are tautologies and the clauses (7) and (8) are standard subsumed by the clauses (2) and (4), respectively. Thus we get the following corollary:

Corollary 5.1 Minimal resolution is not compatible with tautology deletion and deletion of standard subsumed clauses.

If we don't delete tautologies and standard subsumed clauses, the empty clause can be derived:

$$\begin{array}{rll} [(7)1, R, (8)1] & (9) & \{\neg R_8, \neg R_8\} \\ [(9)1, F, (9)2] & (10) & \{\neg R_8\} \\ [(10)1, R, (3)1] & (11) & \{\neg P_6\} \\ [(11)1, R, (1)1] & (12) & \{R_2\} \\ [(12)1, R, (10)1] & (13) & \square \end{array}$$

Nevertheless if we restrict standard subsumption to occurrences having equal identities, we get a version of subsumption which is compatible with minimal resolution.

Definition 5.2 (Minimal Subsumption) A clause C *subsumes* a clause D, if there exists a substitution λ such that $C\lambda \subseteq D$.

We will show that minimal subsumption is compatible with minimal resolution. The basic idea for the proof is that if C subsumes D, then in any refutation where D is used, C can be used instead. However, there are some problems with this idea as the following lock resolution example shows:

$$(1) \quad \{P_1, R_2, P_3\}$$
$$(2) \quad \{R_2, P_3\}$$

The clause (2) subsumes clause (1). But if we build a lock factor of (1):

$$[(1)1, F, (1)3] \quad (3) \quad \{P_1, R_2\}$$

this factor cannot be simulated with (2), because the occurrence P_1 is not contained in (2). Unfortunately, (2) does not subsume (3), because the two P occurrences have different index. Even more seriously, (2) and (3) contain exactly the same literals but have different minimal occurrences. Therefore refutations where such factorization steps occur cannot be simulated. Tammet was not aware of this problem. Therefore his completeness proof of forward subsumption is wrong [7, p. 69, theorem 4.2]. Now we will present a correct proof. The basic idea is to strengthen the completeness result by showing that subsumption is compatible with minimal resolution even if we only consider strict, minimal refutations. As lifting causes no problems we concentrate on ground refutations. Nevertheless, for some intermediate clauses in the simulated strict refutation we will need a weaker notion of subsumption.

We say that a clause $C = \{o\} \cup C_1$ *weakly subsumes* a clause $C' = \{p\} \cup C'_1$, if there exists a substitution λ, such that C_1 subsumes C'_1 with substitution λ and $lit(o)\lambda = lit(p)\lambda$.

Lemma 5.3 Let C, C', D, D' be ground clauses such that C subsumes C' and D subsumes D'.

1. If F' is a strict factor of C', then either there is a strict factor F of C such that F subsumes F' or C subsumes F' or C weakly subsumes F'.

2. If R' is a resolvent of C' and D', then either there is a resolvent R of C and D such that R subsumes R' or C subsumes R' or D subsumes R'.

Proof: 1. Let F' is the strict factor of some literals $o, p \in C'$ where wlog. p is removed from C'. We distinguish three cases. Firstly, if $o, p \in C$, then o and p are minimal in C and it is straightforward to show that there exists a strict factor F of C using the literals o and p, such that F subsumes F'. Secondly, if $o \in C$ but $p \notin C$ then C subsumes F'. Thirdly, if $p \in C$ but $o \notin C$ then C weakly subsumes F'.
2. Let R' be the resolvent of some $p \in C'$ and $o \in D'$. Here we distinguish four cases. Firstly, if $p \in C$, $o \in D$, then the resolvent R between C, D using the occurrences p, o subsumes R'. Secondly, if $p \in C$, but $o \notin D$, then D subsumes R'. Thirdly, if $p \notin C$, but $o \in D$, then C subsumes R'. Eventually, if $p \notin C$ and $o \notin D$, then both C and D subsume R'. □

Lemma 5.4 Let $C = \{o_1\} \cup C_1$ and $C' = \{o'_1\} \cup C'_1$ be ground clauses such that C_1 subsumes C'_1, $lit(o_1) = lit(o'_1)$, and o_1 is minimal in C and o'_1 is minimal in C'. Let $D = \{p\} \cup D_1$ and $D' = \{p'_1\} \cup D'_1$ be ground clauses such that D_1 subsumes D'_1, $lit(p_1) = lit(p'_1)$, and p_1 is minimal in D and p'_1 is minimal in D'. Let F' be a strict factor of C' using o'_1 and some other occurrence $o \in C'_1$. If $o \in C$ we assume o minimal in C.

1. There is a strict factor F of C such that F (weakly) subsumes F' or C (weakly) subsumes F'.

2. If R' is the resolvent of C', D' using the occurrences o'_1, p'_1, then the resolvent R of C, D using the occurrences o_1, p_1, subsumes R'.

Proof: 1. We distinguish two cases. Firstly, if $o \in C$ and o is minimal in C, then if o'_1 is not in F', the strict factor F from C by removing o_1 subsumes F' and otherwise if o is not in F', the strict factor F from C by removing o weakly subsumes F'. Secondly, if $o \notin C$, then C weakly subsumes F'.
2. Since C_1 subsumes C'_1 and D_1 subsumes D'_1 we can conclude R subsumes R'. □

Theorem 5.5 (Subsumption) Let CS be an unsatisfiable clause set. There exists a strict, minimal refutation from CS, even if subsumed clauses are deleted.
Proof: It is sufficient to show that if we have any strict, minimal refutation, we can construct a new one where subsumed clauses are deleted. This is done by an induction argument on the number of subsumed clauses, showing that a refutation using a subsumed clause D can be simulated by a refutation there D is no longer used. The simulation is shown by an induction argument on the length of the refutation. It is sufficient to consider ground refutations, because lifting causes no problems. We only show the non-trivial case that a strict, minimal ground refutation $C'^1, \ldots, C'^n$ of a ground clause set CS using a ground clause C' can be simulated with a strict, minimal ground refutation $C^1, \ldots, C^m$, $m \leq n$, where C' is no longer used, if there is a ground clause $C \in CS$ such that C subsumes C'.
We show by induction on the length n of the derivation, that the refutation can be simulated, such that for each clause C'^i generated by minimal resolution, there is a clause C^k such that C^k subsumes C'^i and for each clause C'^j generated by strict factorization, there is a clause C^l such that C^l subsumes C'^j or C^l weakly subsumes C'^j.
If $n = 1$ then lemma 5.3.2 guarantees the existence of a strict, minimal refutation using C instead of C'.
For $n > 1$, let C'^i be some arbitrary clause in the derivation, generated by minimal resolution. By induction hypothesis, there is a clause C^k which subsumes C'^i. If the next step applied to C'^i is a minimal resolution step, we distinguish two cases. Firstly, if the second used clause C'^j is subsumed by some clause C^l, then lemma 5.3.2 guarantees the existence of a clause which subsumes the resolvent between C'^i and C'^j. Secondly, if the second used clause C'^j is weakly subsumed by some clause C^l, then lemma 5.4.2 guarantees the existence of a resolvent between C^k and C^l which subsumes the resolvent between C'^i and C'^j or C^k subsumes the resolvent between C'^i and C'^j. Note, that in lemma 5.4 we explicitly use that $C'^1, \ldots, C'^n$ is a strict, minimal refutation. If the next step applied to C'^i is a strict factorization step, then lemma 5.3.1 guarantees the existence of a clause which (weakly) subsumes the factor built of C'^i.
If C'^i is generated by strict factorization, by induction hypothesis there is a clause C^k which (weakly) subsumes C'^i. Now if the next step applied to C'^i is again a strict factorization step, lemma 5.3.1 and lemma 5.4.1 guarantee the existence of a clause which

(weakly) subsumes the strict factor of C'^i. Again these lemmata are only applicable because $C'^1, \ldots, C'^n$ is a strict, minimal refutation. The case that a minimal resolution step is applied to C'^i is symmetric to a case where C'^i was generated by resolution.
The simulation results in a derivation of the empty clause where C' is not used. Now we remove from this derivation all clauses, except the empty clause, which are not the parent clause of some other clause. This yields a strict, minimal refutation $C^1, \ldots, C^m$, where C' is no longer used. This together with the a induction on the number of subsumed clauses proves the theorem. $\square$

For unit clauses we may refine our notion of subsumption. A unit clause $C = \{o\}$ subsumes a clause D if there exists a substitution λ such that $\mathit{lit}(o) \in \mathit{lit}(D\lambda)$. It is obvious that C can be substituted for D in any proof, because o is necessarily minimal in C.

Corollary 5.6 (Refined Subsumption) Let CS be an unsatisfiable clause set. There exists a strict, minimal refutation from CS, even if subsumed clauses are deleted with respect to the refined subsumption criterion.

6 Discussion

We have defined minimal resolution. Minimal resolution is sound and complete if $\preceq$ is a stable quasi-ordering. Hyper-resolution, ordered resolution, lock resolution, Π-orderings, **A**-orderings are instances of minimal resolution. List resolution and tuple resolution are new calculi which cannot be simulated by known selection strategies for resolution. We also discussed ordering refinements which cannot be simulated by minimal resolution, semantic resolution and sequencing. Semantic resolution cannot be simulated in general, because the ordering refinement is based on a semantic criterion. Sequencing is the only calculus I know which is based on a syntactic ordering refinement which is not an instance of minimal resolution. However, there is no completeness proof for sequencing in the existing literature. I tried to find either a completeness proof or a counter example for several weeks, however, without success. Therefore this remains an open question.

We also showed that a specific form of subsumption is compatible with minimal resolution. This was not known before for some of the instances, as for instance lock resolution.

There are at several applications for minimal resolution. My initial motivation of minimal resolution was a variant of tuple resolution which allowed to order resolution proofs in a special way. These ordered proofs can then be transformed into proofs of a sorted calculus [16]. In addition, several authors now use refined resolution calculi to prove the decidability of subclasses of first-order logic [4, 7].

References

[1] R. Anderson and W.W. Bledsoe. A linear format for resolution with merging and a new technique for establishing completeness. *Journal of the ACM*, 17:525–534, July 1970.

[2] L. Bachmair and H. Ganzinger. On restrictions of ordered paramodulation with simplification. In *10th International Conference on Automated Deduction, CADE-10*, volume 449 of *LNCS*, pages 427–441. Springer, 1990.

[3] L. Bachmair and H. Ganzinger. Rewrite-based equational theorem proving with selection and simplification. MPI-Report MPI-I-91-208, Max-Planck-Institut für Informatik, Saarbrücken, September 1991.

[4] L. Bachmair, H. Ganzinger, and U. Waldmann. Superposition with simplification as a decision procedure for the monadic class with equality. In Georg Gottlob, Alexander Leitsch, and Daniele Mundici, editors, *Computational Logic and Proof Theory, Third Kurt Gödel Colloquium*, volume 713 of *LNCS*, pages 83–96. Springer, August 1993.

[5] R.S. Boyer. *Locking: A Restriction of Resolution.* PhD thesis, University of Texas at Austin, August 1971.

[6] C.L. Chang and R.C.T. Lee. *Symbolic Logic and Mechanical Theorem Proving.* Computer Science and Applied Mathematics. Academic Press, 1973.

[7] C. Fermüller, A. Leitsch, T. Tammet, and N. Zamov. *Resolution Methods for the Decision Problem*, volume 679 of *LNAI.* Springer, 1993.

[8] M.R. Genesereth and N.J. Nilsson. *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann, 1987.

[9] R. Kowalski and Hayes P.J. Semantic trees in automatic theorem proving. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 87–101. Edingburgh University Press, 1969.

[10] D. Loveland. *Automated Theorem Proving: A Logical Basis*, volume 6 of *Fundamental Studies in Computer Science.* North-Holland, 1978.

[11] S.J. Maslov. An inverse method for establishing deducibility in the classical predicate calculus. *Dokl.Akad.Nauk SSSR*, 159:1420–1424, 1964.

[12] S.J. Maslov. Proof-search strategies for methods of the resolution type. In B. Meltzer and D. Michie, editors, *Machine Intelligence 6*, chapter 6, pages 77–90. Edingburgh University Press, 1971.

[13] J.A. Robinson. Automatic deduction with hyper-resolution. *International Journal of Computer Mathematics*, 1:227–234, 1965.

[14] J.A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, January 1965.

[15] J.R. Slagle. Automatic theorem-proving with renamable and semantic resolution. *Journal of the ACM*, 14:687–697, 1967.

[16] C. Weidenbach. Extending the resolution method with sorts. In *Proc. of 13th International Joint Conference on Artificial Intelligence, IJCAI-93*, pages 60–65. Morgan Kaufmann, 1993.

Below you find a list of the most recent technical reports of the research group *Logic of Programming* at the Max-Planck-Institut für Informatik. They are available by anonymous ftp from our ftp server `ftp.mpi-sb.mpg.de` under the directory `pub/papers/reports`. Most of the reports are also accessible via WWW using the URL `http://www.mpi-sb.mpg.de`. If you have any questions concerning ftp or WWW access, please contact `reports@mpi-sb.mpg.de`. Paper copies (which are not necessarily free of charge) can be ordered either by regular mail or by e-mail at the address below.

Max-Planck-Institut für Informatik
Library
attn. Regina Kraemer
Im Stadtwald
D-66123 Saarbrücken
GERMANY
e-mail: `kraemer@mpi-sb.mpg.de`

MPI-I-95-2-005	F. Baader, H.-J. Ohlbach	A Multi–Dimensional Terminological Knowledge Representation Language
MPI-I-95-2-002	H. J. Ohlbach, R. A. Schmidt	Functional Translation and Second–Order Frame Properties
MPI-I-95-2-001	S. Vorobyov	Proof normalization and subject reduction in extensions of Fsub
MPI-I-94-261	P. Barth, A. Bockmayr	Finite Domain and Cutting Plane Techniques in CLP($\mathcal{PB}$)
MPI-I-94-257	S. Vorobyov	Structural Decidable Extensions of Bounded Quantification
MPI-I-94-254		Report and abstract not published
MPI-I-94-252	P. Madden	A Survey of Program Transformation With Special Reference to *Unfold/Fold* Style Program Development
MPI-I-94-251	P. Graf	Substitution Tree Indexing
MPI-I-94-246	M. Hanus	On Extra Variables in (Equational) Logic Programming
MPI-I-94-241	J. Hopf	Genetic Algorithms within the Framework of Evolutionary Computation: Proceedings of the KI-94 Workshop
MPI-I-94-240	P. Madden	Recursive Program Optimization Through Inductive Synthesis Proof Transformation
MPI-I-94-239	P. Madden, I. Green	A General Technique for Automatically Optimizing Programs Through the Use of Proof Plans
MPI-I-94-238	P. Madden	Formal Methods for Automated Program Improvement
MPI-I-94-235	D. A. Plaisted	Ordered Semantic Hyper-Linking
MPI-I-94-234	S. Matthews, A. K. Simpson	Reflection using the derivability conditions
MPI-I-94-233	D. A. Plaisted	The Search Efficiency of Theorem Proving Strategies: An Analytical Comparison
MPI-I-94-232	D. A. Plaisted	An Abstract Program Generation Logic
MPI-I-94-230	H. J. Ohlbach	Temporal Logic: Proceedings of the ICTL Workshop
MPI-I-94-229	Y. Dimopoulos	Classical Methods in Nonmonotonic Reasoning