A Classification of Non-liftable Orders for Resolution

Hans de Nivelle,
Centrum voor Wiskunde en Informatica,
PO BOX 94079, 1090 GB Amsterdam,
the Netherlands,
email: nivelle@cwi.nl

Abstract

In this paper we study the completeness of resolution when it is restricted by a non-liftable order and by weak subsumption. A *non-liftable order* is an order that does not satisfy $A \prec B \Rightarrow A\Theta \prec B\Theta$. Clause c_1 *weakly subsumes* c_2 if $c_1\Theta \subseteq c_2$, and Θ is a renaming substitution. We show that it is natural to distinguish 2 types of non-liftable orders and 3 types of weak subsumption, which correspond naturally to the 2 types of non-liftable orders. Unfortunately all natural combinations are not complete. The problem of the completeness of resolution with non-liftable orders was left open in ([Nivelle96]). We will also give some good news: Every non-liftable order is complete for clauses of length 2, and can be combined with weak subsumption.

1 Introduction

Resolution is one of the most successful methods for automated theorem proving in first order classical logic. It was introduced in ([Robins65]). Shortly after its discovery it was realized that resolution could be improved by adding so called *refinements*. Refinements are restrictions of the resolution rule. Refinements may improve the resolution process in two possible ways. In the case that a proof exists, a proof will be found more efficiently, i.e. less computer time and space will be used. Also in the case that a proof does not exist refined resolution will behave better than unrefined resolution. In particular the search process may terminate instead of continuing for ever. This results in resolution based *decision procedures*. ([Zamov72], [Joy76], [FLTZ93]).

A large class of refinements consists of *ordering refinements*. In ordered resolution the resolution rule is restricted by choosing an order on literals, and then allowing only the maximal literals in a clause to be resolved upon. Before

([Nivelle95]), the completeness of orders satisfying $A \prec B \Rightarrow A\Theta \prec B\Theta$ (the *liftable orders*) was known, and little was known about non-liftable orders. (The main result was [Tammet94]). In ([Nivelle95]) the completeness of a large class of non-liftable orders was proven, solving some open problems from ([FLTZ93]).
It was proven there that liftability can be replaced by two conditions: Invariance under renaming, ($A \prec B \Rightarrow A\Theta \prec B\Theta$, for all renaming substitutions Θ), and descending under substitution. ($A\Theta \prec A$, whenever $A\Theta$ is not a renaming of A) Subsumption had to be weakened, when combined with non-liftable orders. When $c_1\Theta \subseteq c_2$, the substitution Θ must be a renaming substitution.
In the case that all literals in each clause have exactly the same set of variables the second condition of 'descending under substitution' can be dropped.
The motivation for dropping the liftability property is the fact that the liftability property makes many literals incomparable. For example no liftable order can compare the literals $p(X)$ and $p(s(Y))$. Suppose $p(X) \prec p(s(Y))$, then it would follow that $p(s(0)) \prec p(s(1))$, by the substitution $\Theta_1 = \{X := s(0), Y := 1\}$. In the same way it is necessary that $p(s(1)) \prec p(s(0))$, by the substitution $\Theta_2 = \{X := s(1), Y := 0\}$. It is impossible that $p(s(0)) \prec p(s(1)) \prec p(s(0))$, because $\prec$ is an order, and so $p(X) \prec p(s(Y))$ is not possible. In exactly the same way $p(s(Y)) \prec p(X)$ is impossible and hence $p(X)$ and $p(s(y))$ are incomparable.
With the results of ([Nivelle95]), it is possible to construct orders which are total in the following sense: If A and B are not renamings of each other, then either $A \prec B$, or $B \prec A$. This is the theoretical optimum, since an order which compares literals that are renamings of each other is ill-defined, because the theorem prover can freely rename literals.
However after ([Nivelle95]) several important questions remained: Is it possible to combine full subsumption with non-liftable orders? What happens when the order is not descending under substitution? The proof method, that was based on resolution games, suggests that both are not complete.
Counterexamples for the combination of full subsumption with non-liftable orders can be easily found, but it turned out not easy to construct an unsatisfiable set of clauses, and a non-liftable order $\prec$ such that resolution using this order and combined with weak subsumption does not derive the empty clause. This led us at some moment to the belief that resolution with every order that is invariant under renaming is complete. This belief was increased by the proof that this is true for sets of clauses with length ≤ 2, and by experiments with a theorem prover. However in this paper we show that (probably) resolution with non-liftable orders and weak subsumption is not complete.
In this paper we will distinguish 2 types of non-liftable orders, weakly and strongly non-liftable orders. In order to be compatible with non-liftable orders, subsumption also has to be weakened. We will show that there is one notion of subsumption corresponding to weakly non-liftable orders, and two types of non-liftable orders corresponding to strongly non-liftable orders. For most of the combinations we have counterexamples now. For the remaining combinations

we have a candidate counterexample, which is very hard to check, and which at this point is not yet fully checked. In this paper we will do the following: **(1)** We will give counterexamples to the various combinations of non-liftable orders and subsumption. **(2)** We will give the right notion of subsumption that is compatible with the descending orders of ([Nivelle95]). **(3)** We will give the proof for the completeness of resolution with non-liftable orders for sets of clauses with length ≤ 2.

2 Introductory Definitions

In this section we give some basic definitions. We define clauses, the most general unifier, and we define ordered resolution.

Definition 2.1 Let P be a set of predicate symbols. Let F be a set of function symbols. The set of *terms* over F is the set of objects that can be obtained by finitely often applying the following rules: A variable is a term. If $f \in F$, and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term. If $t_1, \ldots, t_n$ are terms, and $p \in P$, then $p(t_1, \ldots, t_n)$ and $\neg\, p(t_1, \ldots, t_n)$ are *literals* over P and F. If the literal is of the form $p(t_1, \ldots, t_n)$, then the literal is *positive.* If the literal is of the form $\neg\, p(t_1, \ldots, t_n)$, then the literal is *negative.* A *clause* is a finite set of literals.

The intended meaning of a clause $c = \{A_1, \ldots, A_p\}$ is $\forall V_1 \cdots V_n (A_1 \vee \cdots \vee A_p)$, where $V_1, \ldots, V_n$ are the variables that occur in c. The meaning of a clause is independent of the variables that are actually used. Because of this we will assume that variables in clauses can be replaced freely by other variables.

Definition 2.2 The *complexity* of a term t, which we will write as $\#t$, is recursively defined as follows: For a variable V, the complexity $\#V$ equals 1. For a compound term $f(t_1, \ldots, t_n)$, the complexity $\#f(t_1, \ldots, t_n)$ equals $2 + \#t_1 + \cdots + \#t_n$.

Definition 2.3 A *substitution* is a list which specifies how variables should be replaced. A substitution has the form

$$\Theta = \{V_1 := t_1, \ldots, V_n := t_n\}.$$

This form prescribes that simultaneously all variables V_i have to be replaced by their corresponding t_i. In order to be meaningful it is necessary that $V_i = V_j \Rightarrow t_i = t_j$.

If A_1 and A_2 are literals, then A_1 is called an *instance* of A_2 if there is a substitution Θ, such that $A_1 = A_2\Theta$. The literals A_1 and A_2 are called *renamings* of each other if A_1 is an instance of A_2, and A_2 is an instance of A_1.

If Θ_1 and Θ_2 are substitutions, then the *composition* of Θ_1 and Θ_2 is defined as follows:

$$\Theta_1 \cdot \Theta_2 = \{V := (V\Theta_1)\Theta_2 \mid (V\Theta_1)\Theta_2 \neq V\}.$$

If A_1 and A_2 are literals/atoms/terms, then a *unifier* is a substitution Θ, for which $A_1\Theta = A_2\Theta$. A *most general unifier, mgu* is a unifier Θ, such that for every unifier Σ there exists a substitution Ξ, such that $\Sigma = \Theta \cdot \Xi$.

A substitution Θ is called a *renaming* if for every variable V, the result $V\Theta$ is a variable, and $V_1\Theta = V_2\Theta$ implies $V_1 = V_2$.

If two literals have a unifier then they have a most general unifier. There exists a total algorithm which computes a most general unifier, if one exists. This was first proven in ([Robins65]). Next we will define an order, and then ordered resolution and factorization.

Definition 2.4 An order is a relation $\prec$, with the following properties: Never $d \prec d$. If $d_1 \prec d_2$, and $d_2 \prec d_3$, then $d_1 \prec d_3$.

If S is a set then an element $s \in S$ is *maximal* in S if there is no $s' \in S$, with $s \prec s'$. If $\prec$ is an order then we will write $d_1 \preceq d_2$ for $d_1 \prec d_2$ or $d_1 = d_2$.

Definition 2.5 Let $\prec$ be an order on literals. We define ordered resolution and factorization:

Resolution Let $c_1 = \{A_1\} \cup R_1$ and $c_2 = \{\neg A_2\} \cup R_2$ be clauses (renamed, such that c_1 and c_2 have no overlapping variables), such that **(1)** A_1 is maximal in c_1, and $\neg A_2$ is maximal in c_2, and **(2)** A_1 and A_2 are unifiable, **(3)** Θ is the mgu of A_1 and A_2. Then the clause $R_1\Theta \cup R_2\Theta$ is an *ordered resolvent* of R_1 and R_2.

Factorization Let $c = \{A_1, A_2\} \cup R$ be a clause such that **(1)** A_1 is maximal in c, **(2)** A_1 and A_2 are unifiable, **(3)** Θ is the mgu of A_1 and A_2. Then $c\Theta = \{A_1\Theta\} \cup R\Theta$ is an *ordered*-factor of c.

In ([Robins65]) resolution was defined slightly different, in such a manner that the factorization and the resolution rules are combined: If $\{A_1, \ldots, A_n\} \cup R_1$ and $\{\neg B_1, \ldots, \neg B_m\} \cup R_2$ are clauses, one of the A_i is maximal, and one of the $\neg B_j$ is maximal, and the A_i and B_j have a simultaneous unifier Θ, then derive $R_1\Theta \cup R_2\Theta$. This resolution rule includes the factorization rule. See ([Leitsch88]) for a discussion of this.

The resolution and factoring rule are applied as follows: If one wants to prove that a formula G is a logical consequence of formulae $F_1, \ldots, F_p$ then the formula

$F_1 \wedge \cdots \wedge F_p \wedge \neg G$ is unsatisfiable. The formula $F_1 \wedge \cdots \wedge F_p \wedge \neg G$ can be transformed into a finite set of clauses $C = \{c_1, \ldots, c_n\}$, on which resolution can be applied.

3 Types of Orders

In this section we will introduce the various types of non-liftable orders, and the corresponding types of subsumption. In order to make the table complete, we will also include liftable orders, and full subsumption. Before we can define the orders we need a definition:

Definition 3.1 Let A_1 and A_2 be two literals. We call A_1 a *weak renaming* of A_2 if A_1 can be obtained from A_2 by replacing variables by variables.

So it is not necessary that the replacements are consistent. $p(X, Y, Y)$ is a weak renaming of $p(X, X, Y)$ and of $p(X, X, X)$, but not of $p(X, X, 0)$.

Definition 3.2 We distinguish the following type of orders:

L Order $\prec$ is *liftable* if $A \prec B \Rightarrow A\Theta \preceq B\Theta$, for all substitutions Θ.

WNL Order $\prec$ is *weakly non-liftable* if $A \prec B \Rightarrow A_1 \prec B_1$ for all weak renamings A_1 of A, and B_1 of B.

SNL Order $\prec$ is *strongly non-liftable* if $A \prec B \Rightarrow A_1 \prec B_1$, for all renamings A_1 of A, and B_1 of B.

DESC Order $\prec$ is descending if $A_1 \prec A$, for all instances A_1 of A, such that A_1 is not a renaming of A.

There are different possible notions of SNL. Another would be: SNL2: $A \prec B \Rightarrow A\Theta \prec B\Theta$, for all renaming substitutions Θ. This appears to be a weaker condition, but it is proven in ([Nivelle95b]), Theorem 7.2.5, that every order satisfying SNL2 is included in an order satisfying SNL, so for $\subseteq$-maximal orders SNL and SNL2 coincide.
We will mostly describe non-liftable orders by summing them up in the following manner: $A_1 \prec A_2 \cdots \prec A_n$. Writing this we mean the smallest order of the intended type, that satisfies the given inequalities.
All types of orders have their own type of subsumption associated with them. In order to define the type of subsumption that corresponds to descending orders, we need to define an order on clauses:

Definition 3.3 Let $\prec$ be an order. We define the relation $\preceq\preceq$. Let c_1 and c_2 be clauses. Then $c_1 \preceq\preceq c_2$ if the following holds: Let d_1 be the set of maximal literals of c_1, let d_2 be the set of maximal literals of c_2. Then it must be the case that for every literal A, the number of renamings of A in d_1 must be less than or equal to the number of renamings of A in d_2.

The ordering $\preceq\preceq$ can be seen as the standard multiset set order, but ignoring all non-maximal elements.

Definition 3.4 Clause c_1 *subsumes* clause c_2 in one of the manners below if the following holds: There is a substitution Θ, such that $c_1\Theta \subseteq c_2$, and $\mid c_1 \mid \ \leq \ \mid c_2 \mid$, together with the special conditions that belong to each type:

SUBS(L) There are no more conditions.

SUBS(WNL) For each literal $A \in c_1$, the literal $A\Theta$ must be a weak renaming of A.

SUBS(SNL1) For each literal $A \in c_1$, the literal $A\Theta$ must be a renaming of A.

SUBS(SNL2) The substitution Θ must be a renaming substitution.

SUBS(DESC) $c_2 \preceq\preceq c_1$.

The combination of orders of type DESC with subsumption of type SUBS(DESC) is complete. This we will show in the next section. For the remaining combinations there is the following table:

Table 3.5

	SUBS(L)	SUBS(WNL)	SUBS(SNL1)	SUBS(SNL2)
L	yes	yes	yes	yes
WNL	no (5.2)	no (5.3)	no (5.4)	unknown(no?)
SNL	no (5.2)	no (5.3)	no (5.4)	unknown(no?)

The first row consists of classical results. The problems in the lower right corner will be discussed in Section 7.

4 Completeness of DESC with SUBS(DESC)

We will prove that the combination of DESC with SUBS(DESC) is complete. The proof is based on the completeness proof in ([Nivelle95]), for orders of type DESC with subsumption of type SUBS(SNL1), which was based on the resolution game. The completeness proof here is a rather straightforward adaptation of that proof. In fact subsumption of type DESC is strongly suggested by the resolution game, and naturally occurs when the proof is read carefully.

Definition 4.1 Let C be a clause set. A *resolution game* is obtained by taking an attribute set $\mathcal{A}$, and a transitive reflexive relation $\sqsubseteq$ on $\mathcal{A}$. We define from $\sqsubseteq$ the relations: a_1 is *equivalent* with a_2 if $a_1 \sqsubseteq a_2$, and $a_2 \sqsubseteq a_1$. $a_1 \sqsubset a_2$ iff $a_1 \sqsubseteq a_2$ and not $a_2 \sqsubseteq a_1$. This $\sqsubset$ must be well-founded. To each literal in a clause of C, an attribute from $\mathcal{A}$ is attached. (Like in lock-resolution [Boyer71]).

Resolution If $c_1 = \{a{:}A_1\} \cup R_1$ and $c_2 = \{\neg\ a{:}A_2\} \cup R_2$ are clauses, such that $a{:}A_1$ is $\sqsubset$-maximal in c_1, and $\neg\ a{:}A_2$ is $\sqsubset$-maximal in c_2, then $R_1 \cup R_2$ is a resolvent of c_1 and c_2.

Factorization If $c = \{a{:}A_1, a{:}A_2\} \cup R$ is a clause, such that $a{:}A_1$ is $\sqsubset$-maximal in c, then $\{a{:}A_1\} \cup R$ is a factor of c. (a may be a negative literal)

Reduction Let c_1 be a clause. A clause c_2 is a reduction of c_1 if the following holds: Let d_1 be the set of $\sqsubset$-maximal literals in c_1. Let d_2 be the set of $\sqsubset$-maximal literals in c_2. Then, for every indexed literal $a{:}A$ the number of equivalent literals in d_1 must be not less than the number of literals that is equivalent with $a{:}A$ in d_2.

So in fact the definition of reduction strongly suggests subsumption of type DESC.

Definition 4.2 The resolution game is played by two players, the *defender*, and the *opponent*. The opponent tries to derive the empty clause using the factorization and resolution rule. The defender tries to disturb the opponent by replacing clauses by reductions.

The game starts with the defender. He may initially assign attributes from $\mathcal{A}$ to the literals. After that the opponent may derive new clauses, using the factorization and resolution rule. However, every time when the opponent derives a new clause, the defender has the right to replace it by a reduction, before the opponent can use it.

The resolution game can be seen as resolution with changing orders.

Theorem 4.3 There exists a winning strategy for the opponent if and only if the initial clause set is unsatisfiable. (A complete proof is in [Nivelle95]).

Theorem 4.4 Orders of type DESC can be combined with subsumption of type DESC without losing completeness.

Proof: (Sketch) The main idea is to construct a resolution game, such that the DESC-ordered deduction can be simulated by a strategy of the defender, and then to apply Theorem 4.3.

Let C be a clause set that is unsatisfiable. Let $\overline{C}$ be a set of ground clauses, such that each $c \in \overline{C}$ has a clause $c \in C$, such that $\bar{c}$ is an instance of c. (We will say that c *represents* $\bar{c}$) Construct the following resolution game:

- $\mathcal{A}$ is defined as the set of literals A such that A is an instance of a literal in C, and A has an instance in $\overline{C}$. (The set of *in-between literals*)
- The order $\sqsubseteq$ is defined as follows: $A_1 \sqsubseteq A_2$ iff $A_1 \prec A_2$, or A_1 is a renaming of A_2. (Note that $\sqsubset$ is well-founded, because the set of in-between literals that are not renamings of each other, is finite)
- The initial clause set of the resolution game is obtained by replacing each clause $\{a_1, \ldots, a_p\}$ represented by $\{A_1, \ldots, A_p\}$, by the indexed clause $\{a_1{:}\, A_1, \ldots, a_p{:}\, A_p\}$.

Now it can be checked that a resolution derivation, based on C, using the order $\prec$ can be simulated by the following strategy of the defender:
Every time a new indexed clause is derived, reduce the attributes in such a manner that the result is the clause that would be derived from C, using $\prec$.
If the result is replaced by a subsuming clause, then reduce to this subsuming clause.
Because of Theorem 4.4, the empty clause will be derived using this strategy. Then from this derivation of the empty clause a $\prec$-ordered refutation of C can be extracted by replacing each indexed clause $\{a_1{:}\, A_1, \ldots, a_p{:}\, A_p\}$ by $\{A_1, \ldots, A_p\}$.

5 Counter Examples

In this section we will present a series of counterexamples, showing that various combinations of non-liftable orders and subsumption are not complete. The counterexamples in this section are all based on the following propositional example:

Example 5.1 Take the following clause set:

$$\{p, q\}, \{q, r\}, \{r, p\}, \{\neg\, p, \neg\, q\}, \{\neg\, q, \neg\, r\}, \{\neg\, r, \neg\, p\}, \{\neg\, p, p\}, \{\neg\, q, q\}, \{\neg\, r, r\}$$

If a refinement prefers the first literal in the first three clauses, and it prefers negative over positive literals, then the empty clause can not be derived. Nevertheless the first 6 clauses together are inconsistent.

Example 5.2 We will show that orders of type WNL are not compatible with SUBS(L), even if the order satisfies DESC, and the clauses have exactly the same set of variables. Let the clause set be:

$$\{p(X), q(X)\}, \{q(X), r(X)\}, \{r(0), p(0)\},$$

$$\{\neg\, p(X), \neg\, q(X)\}, \{\neg\, q(X), \neg\, r(X)\}, \{\neg\, r(X), \neg\, p(X)\}.$$

Let the order $\prec$ be defined from: If A is a positive literal and B is a negative literal, then $A \prec B$. Among positive literals $\prec$ is defined from the following:

$$q(0) \prec p(0) \prec r(0) \prec r(V) \prec q(V) \prec p(V).$$

The only new positive clause that can be derived equals $\{p(0), q(0)\}$. However this clause is subsumed SUBS(L) by the first clause, and is not kept. It is easily checked that the clause set is unsatisfiable by substituting 0 for X.

Example 5.3 The combination of orders of type SNL with SUBS(WNL) is not complete.

$$\{p(X,Y), q(X,Y)\}, \{q(X,Y), r(X,Y)\}, \{r(0,Y), p(Y,Y)\},$$

$$\{\neg\, p(X,Y), \neg\, q(X,Y)\}, \{\neg\, q(X,Y), \neg\, r(X,Y)\}, \{\neg\, p(X,X), \neg\, r(Y,X)\}.$$

Let $\prec$ be defined from: If A is a positive literal, and B is a negative literal, then $A \prec B$. Among positive literals $\prec$ is defined as follows:

$$r(V,V) \prec q(V,V) \prec p(V,V) \prec r(0,V).$$

The only new positive clause that can be derived is $\{p(X,X), q(X,X)\}$. However this clause is SUBS(WNL)-subsumed by the first clause.

The following example proves that the combination of an SNL-order with SUBS(SNL2) is not complete. It is obtained from the previous example by replacing every positive literal $s(X,Y)$ by a pair $s_1(X,A), s_2(A,X)$.

Example 5.4 The combination of an SNL-order with SUBS(SNL1) is incomplete.

$$\{p_1(X,A), p_2(A,Y), q_1(X,B), q_2(B,Y)\}, \{p_2(X,Y), q_1(X,B), q_2(B,Y)\},$$

$$\{q_1(X,A), q_2(A,Y), r_1(X,B), r_2(B,Y)\}, \{q_2(X,Y), r_1(X,B), r_2(B,Y)\},$$

$$\{r_1(0,A), r_2(A,Y), p_1(Y,B), p_2(B,Y)\}, \{r_2(0,Y), p_1(Y,B), p_2(B,Y)\},$$

$$\{\neg p_1(X,X)\}, \{\neg q_1(X,X)\}, \{\neg r_1(X,X)\},$$

$$\{\neg\, p_2(A,B), \neg\, q_2(A,B)\}, \{\neg\, q_2(A,B), \neg\, r_2(A,B)\}, \{\neg\, p_2(X,X), \neg\, r_2(A,X)\}$$

The order is defined from: If P is a positive literal, and Q is a negative literal, then $P \prec Q$. Among positive literals $\prec$ is defined from: $r_2(X,Y) \prec r_1(X,Y) \prec q_2(X,Y) \prec q_1(X,Y) \prec$
$p_2(X,Y) \prec p_1(X,Y) \prec r_2(0,Y) \prec r_1(0,X)$.

6 Clauses Consisting of Two Literals

In this section we prove that orders of type SNL can be combined with subsumption of type SUBS(SNL1), when the clause set consists of clauses with length at most 2. We will call such clauses *2-clauses*. The proof is almost the same as the completeness proof for clause sets where the literals in each clause have exactly the same set of variables. The proof is based on the fact that it is possible to attach a measure to each clause. When two clauses resolve the new clause will receive a measure that is not higher than each of the parent clauses. When a literal changes, it changes at the moment that it is copied as a side literal into a clause with a lower measure. Using this fact, a resolution game can be constructed, by including this measure into the attributes. Then, when a literal appears to increase under the order, it in fact decreases, because the measure decreases. First note that the set of 2-clauses is closed under resolution and factorization:

Lemma 6.1 Let c_1 and c_2 be clauses with $\mid c_1 \mid \leq 2$, and $\mid c_2 \mid \leq 2$. If c_1' is a factor of c_1, then $\mid c_1' \mid \leq 2$. If c' is a resolvent of c_1 and c_2, then $\mid c' \mid \leq 2$.

We will define the measure for clauses:

Definition 6.2 Let c be a 2-clause, representing $\overline{c}$. The *connectivity* of $\overline{c}$ is defined as follows:

1. If c is of the form $\{A_1, A_2\}$ and $\overline{c}$ is of the form $\{a_1, a_2\}$, the connectivity of $\overline{c}$ is obtained as follows. Let Θ be a substitution, such that $A_1\Theta = a_1$ and $A_2\Theta = a_2$. Let $V_1, \ldots, V_n$ be the variables that are shared by A_1 and A_2. The connectivity of c equals $\#V_1\Theta + \cdots + \#V_n\Theta$.
2. If c is of the form $\{A_1, A_2\}$, and $\overline{c}$ is of the form $\{a\}$, then the connectivity of $\overline{c}$ is defined as 0.
3. If c is of the form $\{A\}$, and $\overline{c}$ is of the form $\{a\}$, then the connectivity of c is also defined as 0.

Example 6.3 If $\{p(X,Y), q(Y,Z)\}$ represents $\{p(0, s(0)), q(s(0), s(s(0)))\}$, then its connectivity equals $\#s(0) = 4$. The connectivity of $\{p(X,Y), q(Z,T)\}$ equals 0, independent of the clause that is represented. The connectivity of $\{p(X,Y), q(X,Y)\}$, representing $\{p(0,0), q(0,0)\}$ equals $\#0 + \#0 = 4$.

We will show that this measure decreases through the derivation. First we prove that it decreases in ordinary resolution, when both clauses are of length 2.

Lemma 6.4 Let $\overline{c}_1 = \{a, b_1\}$, and let $\overline{c}_2 = \{\neg\, a, b_2\}$. Let $c = \{A_1, B_1\}$ represent $\overline{c}_1$, and let $c_2 = \{A_2, B_2\}$ represent $\overline{c}_2$. Then the connectivity of $\{b_1, b_2\}$, represented by $\{B_1, B_2\}$ is not more than the connectivity of c_1, and not more than the connectivity c_2.

Proof: Assume that c_1 and c_2 have no overlapping variables, without loss of generality. Let Θ be the most general unifier of $\neg A_1$ and A_2. Let Σ be a substitution for which $B_1\Sigma = b_1$, $B_2\Sigma = b_2$, and $A_1\Theta\Sigma = a$. The construction of the resolvent takes place in two steps:

1. First $\{A_1, B_1\}$ is replaced by $\{A_1\Theta, B_1\Theta\}$, and $\{A_2, B_2\}$ is replaced by $\{A_2\Theta, B_2\Theta\}$.
2. Then the resolvent $\{B_1, B_2\}$ is constructed.

We show that, for both $i \in \{1,2\}$, the connectivity of $\{A_i\Theta, B_i\Theta\} \leq$ the connectivity of $\{A_i, B_i\}$, representing $\{\pm a, b_i\}$.
Let $V_1, \ldots, V_n$ be the variables that are shared by A_i and B_i. Let $W_1, \ldots, W_m$ be the variables that are shared by $A_i\Theta$ and $B_i\Theta$. We have $W_1\Sigma + \cdots + W_m\Sigma \leq V_1\Theta\Sigma + \cdots + V_n\Theta\Sigma$, because of the following argument: Every W_i must occur in one of the $V_j\Theta$, otherwise Θ would not be most general. The different W_i must occur at different positions in the $V_j\Theta$. Because of this the $W_i\Sigma$ are disjoint subterms of the $V_j\Theta$. For this reason the complexity of the $W_i\Sigma$ together must be lower than the complexity of the $V_j\Theta\Sigma$ together.
Now we prove 2. Let $X_1, \ldots, X_p$ be the variables that are shared by $B_1\Theta$ and $B_2\Theta$. For both $i \in \{1,2\}$, these variables must be shared by $A_i\Theta$ and $B_i\Theta$, since otherwise Θ would not be most general. Then $\#X_1\Sigma + \cdots + \#X_p\Sigma \leq \#Y_1\Sigma + \cdots + \#Y_q\Sigma$, because every X_i is a Y_j.
End of proof

Lemma 6.5 Let $c_1 = \{A_1, A_2\}$ and $c_2 = \{B_1, B_2\}$ be clauses representing the same ground clause $\{b_1, b_2\}$. If c_1 subsumes SUBS(SNL1) c_2, then the connectivity of c_1 is not more than the connectivity of c_2.

Proof: Let Θ be a substitution such that $A_1\Theta = B_1$, and $A_2\Theta = B_2$, Let Σ be a substitution such that $B_1\Sigma = b_1$, and $B_2\Sigma = b_2$. Let $V_1, \ldots, V_n$ be the variables that are shared by A_1 and A_2. Every variable V_i is replaced by a variable $V_i\Theta$, because all V_i occur in A_1, and $A_1\Theta$ is a renaming of A_1.
There are no V_i and V_j, such that $i \neq j$, and $V_i\Theta = V_j\Theta$. This is because V_i and V_j occur in A_1, and $A_1\Theta$ is a renaming of A_1.
Then because all the $V_i\Theta$ are separate variables, shared by $A_1\Theta$ and $B_1\Theta$, the connectivity cannot increase.
End of proof
The previous lemma is definitely not true for SUBS(WNL). On this fact Example 5.3 is based. The clause $\{p(X,Y), q(X,Y)\}$ (representing $\{p(0,0), q(0,0)\}$) subsumes SUBS(WNL) $\{p(X,X), q(X,X)\}$, but the connectivity of the first clause is 4, and the connectivity of the second clause is 2.

Theorem 6.6 Resolution, using an order of type SNL is complete for 2-clauses, and compatible with subsumption of type SUBS(SNL1).

Proof: Let C be an initial clause set that is unsatisfiable. Let $\prec$ be an order of type SNL. There is a set of ground clauses $\overline{C}$, that is unsatisfiable, and such that each $\overline{c} \in \overline{C}$ is represented by a clause $c \in C$. Construct the following resolution game:

- $\mathcal{A}$ is defined as the set of pairs (n, A), where n is a natural number, and where A is a literal, such that A is an instance of a literal occurring in C, and A has a literal occurring in $\overline{C}$ as an instance.
- The order $\sqsubseteq$ is defined as follows: If $n_1 < n_2$, then $(n_1, A_1) \sqsubseteq (n_2, A_2)$. If $n_1 = n_2$ and $A_1 \prec A_2$, then $(n_1, A_1) \sqsubseteq (n_2, A_2)$. If $n_1 = n_2$, and A_1 is a renaming of A_2, then $(n_1, A_1) \sqsubseteq (n_2, A_2)$.
- The initial clause set of the resolution game is obtained by replacing every clause $\{a_1, a_2\}$ in $\overline{C}$, represented by $\{A_1, A_2\}$, by a clause $\{a_1{:}\,(n, A_1), a_2{:}\,(n, A_2)\}$, where n is the connectivity of $\{A_1, A_2\}$.

The relation $\sqsubset$ is well-founded on $\mathcal{A}$, because it is the composition of two well-founded orders: $<$ on the natural numbers, and $\prec$ on the set of literals that have an instance in $\overline{C}$, and are instance of a literal in C. The latter is true because this set is finite modulo renaming. Similar to the situation in Theorem 4.4 it is possible to define a strategy of the defender, such that a $\prec$-ordered refutation of C can be extracted from it.
End of proof
At this point we can explain the examples of Section 5. SUBS(L) can not be combined with non-liftable orders, because replacing a 2-clause by a SUBS(L)-subsuming clause may increase the connectivity. (See Example 5.2). Replacing a 2-clause by a SUBS(WNL)-subsuming clause may also increase the connectivity. (See Example 5.3).

7 The Combination of SNL and SUBS(SNL2)

In this section we discuss the problems in the lower right corner of Table 3.5. We do not know for certain whether or not the combination of orders of type SNL with subsumption of type SUBS(SNL2) is complete, but we consider Examples 7.3 and 7.4 likely candidates for counter examples. The proof in Section 6 essentially needs the fact that the clauses are of length 2. The point where the proof fails is in factorization. The problem there is that it may increase the connectivity in clauses that have a length greater than 2. We will explain the problems using Example 7.2. First we give the propositional basis for Example 7.2.

Example 7.1 Consider the following clause set:

$$\{a_1, b_1\}, \{b_1, a_2\}, \{a_2, b_2\}, \{b_2, b_1\}, \{\neg\, a_1, a_2\}, \{\neg\, b_1, b_2\}, \{\neg\, a_1, \neg\, a_2\}, \{\neg\, b_1, \neg\, b_2\}$$

If a refinement always prefers the first literal in the positive clauses, and always prefers negative literals over positive literals, then the empty clause will not be derived. However the first positive clause together with the negative clauses, are unsatisfiable.

Example 7.2 Consider the following clause set:

$$\{a_1, b_1\}, \{b_1, a_2(X), p(X)\}, \{p(X), p(Y), a_2(X), b_2(Y)\}, \{a_2(X), b_2(X)\}, \{b_2(0), b_1\},$$

$$\{\neg\, a_1, p(X), a_2(X)\}, \{\neg\, b_1, p(Y), b_2(Y)\}, \{\neg\, a_1, \neg\, a_2(0)\}, \{\neg\, b_1, \neg\, b_2(0)\}, \{\neg p(X)\}.$$

The order $\prec$ is defined as follows: If A is a positive literal, and B is a negative literal, then $A \prec B$. Among positive literals $\prec$ is defined as follows:

$$b_2(X) \prec a_2(X) \prec p(X) \prec b_1 \prec a_1 \prec b_2(0) \prec a_2(0).$$

If we are forced to factor $p(X)$ and $p(Y)$ in the clause $\{p(X), p(Y), a_2(X), b_2(Y)\}$, then the empty clause can not be derived. However it is possible not to factor, and to derive $\{a_2(X), b_2(Y)\}$ instead of $\{a_2(X), b_2(X)\}$.

Example 7.2 shows that factorization can increase the connectivity. In the example the connectivity between $a_2(X)$ and $b_2(Y)$ is increased by factorization. Here factorization can be easily avoided. However it is possible to construct examples in which this is not possible, and the need for factorization brings us into real problems: They are obtained by adding a copy of the initial clause set, replacing a by c, b by d, and p by q. Then a clause $\{\neg\, p, \neg\, q\}$ is added. In order to derive a clause without p and q it is necessary at some moment to factor on p, or on q.

Example 7.3 The clause set is:

$$\{a_1(X), b_1(X)\}, \{a_2(X, A), b_2(X, A)\}, \{c_1(X), d_1(X)\}, \{c_2(X, A), d_2(X, A)\},$$

$$\{\neg\, a_1(X), a_2(X, A), p(A)\}, \{\neg\, b_1(X), b_2(X, B), p(B)\},$$

$$\{\neg\, c_1(X), c_2(X, A), q(A)\}, \{\neg\, d_1(X), d_2(X, B), q(B)\}, \{\neg\, p(X), \neg\, q(X)\},$$

$$\{\neg\, a_1(X), \neg\, a_2(0, X)\}, \{\neg\, b_1(X), \neg\, b_2(0, X)\},$$

$$\{\neg\, c_1(X), \neg\, c_2(0, X)\}, \{\neg\, d_1(X), \neg\, d_2(0, X)\}$$

The order is defined from: $A \prec B$ if A is positive, and B is negative, and

$$d_2(X, Y) \prec c_2(X, Y) \prec b_2(X, Y) \prec a_2(X, Y) \prec q(X) \prec p(X) \prec d_1(X) \prec$$

$$c_1(X) \prec b_1(X) \prec a_1(X) \prec d_2(0, X) \prec c_2(0, X) \prec b_2(0, X) \prec a_2(0, X).$$

Example 7.4

$$\{a_1(X), b_1(X)\}, \{a_2(X,A), b_2(X,A)\}, \{c_1(X), d_1(X)\}, \{c_2(A,X), d_2(A,X)\},$$
$$\{\neg\, a_1(X), a_2(X,A), p(X,A)\}, \{\neg\, b_1(X), b_2(X,B), p(X,B)\},$$
$$\{\neg\, c_1(X), c_2(A,X), q(A,X)\}, \{\neg\, d_1(X), d_2(B,X), q(B,X)\}, \{\neg\, p(X,Y), \neg\, q(X,Y)\},$$

$$\{\neg\, a_1(X), \neg\, a_2(X,0)\}, \{\neg\, b_1(X), \neg\, b_2(X,0)\},$$
$$\{\neg\, c_1(X), \neg\, c_2(0,X)\}, \{\neg\, d_1(X), \neg\, d_2(0,X)\}.$$

The order is defined from: $A \prec B$ if A is positive, and B is negative, and

$$d_2(X,A) \prec c_2(X,A) \prec b_2(X,A) \prec a_2(X,A) \prec q(X,A) \prec p(X,A) \prec d_1(X) \prec$$
$$c_1(X) \prec b_1(X) \prec a_1(X) \prec d_2(0,X) \prec c_2(0,X) \prec b_2(X,0) \prec a_2(X,0).$$

Both examples are based on the same principle: In order to derive the empty clause it is necessary to eliminate the p and q literals. In order to do this they have to be factored, and this generates a situation similar to Example 7.2.

We have made some tests with a theorem prover, and collected app. 100000 minutes of CPU-time for both examples, without coming near a situation from which the empty clause could be derived, so this can be taken as evidence that the examples are indeed counter examples, but we do not posses a conclusive argument that the empty clause cannot be derived from these examples.

8 Conclusions

The combination of the resolution game and connectivity is the right approach for dealing with non-liftable orders, because we can explain the examples with them: If the connectivity decreases then the refinement is complete, if the connectivity can increase then the refinement is not complete.
If at some moment it would turn out the combination of orders of type SNL with SUBS(SNL2)-subsumption is complete after all, then this would be of little practical value, because these orders do obviously not increase the efficiency of the search process.
Checking Examples 7.3, and 7.4 with a strong theorem prover has high priority. A problem here is that the standard theorem provers do not support weak subsumption and non-liftable orders.

References

[Boyer71] R.S. Boyer, Locking: A Restriction of Resolution, Ph. D. Thesis, University of Texas at Austin, Texas 1971.

[ChangLee73] C-L. Chang, R. C-T. Lee, Symbolic logic and mechanical theorem proving, Academic Press, New York 1973.

[FLTZ93] C. Fermüller, A. Leitsch, T. Tammet, N. Zamov, Resolution Methods for the Decision Problem, Springer Verlag, 1993.

[Joy76] W.H. Joyner, Resolution Strategies as Decision Procedures, J. ACM 23, 1 (July 1976), pp. 398-417.

[Leitsch88] A. Leitsch, On Some Formal Problems in Resolution Theorem Proving, Yearbook of the Kurt Gödel Society, pp. 35-52, 1988.

[Lovelnd78] D. W. Loveland, Automated Theorem Proving, a Logical Basis, North Holland Publishing Company, Amsterdam, New York, Oxford, 1978.

[Nivelle95] Resolution Games and Non-Liftable Resolution Orderings, in CSL 94, pp. 279-293, Springer Verlag, Kazimierz, Poland, 1994,

[Nivelle95b] H. de Nivelle, Ordering Refinements of Resolution, Ph. D. thesis, Delft University of Technology, 1995.

[Nivelle96] Resolution Games and Non-Liftable Resolution Orderings, the Annals of the Kurt Gödel Society, 1996.

[Reynolds66] J. Reynolds, Unpublished Seminar Notes, Stanford University, Palo Alto, California, 1966.

[Robins65] J.A. Robinson, A Machine Oriented Logic Based on the Resolution Principle, Journal of the ACM, Vol. 12, pp 23-41, 1965.

[Tammet94] T. Tammet, Seperate Orderings for Ground and Non-Ground Literals Preserve Completeness of Resolution, unpublished, 1994.

[Zamov72] N.K. Zamov: On a Bound for the Complexity of Terms in the Resolution Method, Trudy Mat. Inst. Steklov 128, pp. 5-13, 1972.