Stratified Resolution

Anatoli Degtyarev[1] and Andrei Voronkov[2]

[1] Manchester Metropolitan University
a.degtiarev@doc.mmu.ac.uk
[2] University of Manchester
voronkov@cs.man.ac.uk

Abstract. We introduce a calculus of *stratified resolution*, in which special attention is paid to clauses that "define" relations. If such clauses are discovered in the initial set of clauses, they are treated using the rule of *definition unfolding*, i.e. the rule that replaces defined relations by their definitions. Stratified resolution comes with a new, previously not studied, notion of redundancy: a clause to which definition unfolding has been applied can be removed from the search space. To prove completeness of stratified resolution with redundancies we use a novel technique of *traces*.

1 Introduction

In this article we introduce two versions of *stratified resolution*, — a resolution calculus with special rules for handling hierarchical definitions of relations. Stratified resolution generalizes SLD-resolution for Horn clauses to a more general case, where clauses may be non-Horn but "Horn with respect to a set of relations".

EXAMPLE 1 Suppose we try to establish inconsistency of a set of clauses S containing a recursive definition of a relation *split* that splits a list of conferences into two sublists: of deduction-related conferences, and of all other conferences.

$$\begin{array}{l} \mathit{split}(\texttt{[}x\,|\,y\texttt{]}, \texttt{[}x\,|\,z\texttt{]}, u) \texttt{ :- } \mathit{deduction}(x), \mathit{split}(y,z,u). \\ \mathit{split}(\texttt{[}x\,|\,y\texttt{]}, z, \texttt{[}x\,|\,u\texttt{]}) \texttt{ :- } \neg\mathit{deduction}(x), \mathit{split}(y,z,u). \\ \mathit{split}(\texttt{[]}, \texttt{[]}, \texttt{[]}). \end{array}$$

Suppose that S also contains other clauses, for example

$$\neg\mathit{split}(x,y,z) \vee \mathit{conference\_list}(x).$$

If we use ordered resolution with negative selection (as most state-of-the-art systems would do), we face several choices in selecting the order and negative literals in clauses. For example, if we choose the order in which every literal with the relation *deduction* is greater than any literal with the relation *split*, then we must select either $\neg\mathit{deduction}(x)$ or $\neg\mathit{split}(y,z,u)$ in the first clause. It seems much more natural to select $\mathit{split}(\texttt{[}x\,|\,y\texttt{]}, \texttt{[}x\,|\,z\texttt{]}, u)$ instead, then we can use the first clause in the same way it would be used in logic programming. Likewise, if we always try to select a negative literal in a clause, the literal $\neg split(y, z, u)$ will be selected in the second clause, which is most likely a wrong choice, since then any resolvent with the second clause will give us a larger clause.

Let us now choose an ordering in which the literals $split(\texttt{[}x\,|\,y\texttt{]}, \texttt{[}x\,|\,z\texttt{]}, u)$ and $split(\texttt{[}x\,|\,y\texttt{]}, z, \texttt{[}x\,|\,u\texttt{]})$ are maximal in their clauses, and select these literals. Consider the fourth clause. If we select $\neg split(x, y, z)$ in it, we can resolve this literal with all three clauses defining *split*. It would be desirable to select $conference\_list(x)$ in it (if our ordering allows us to do so), since a resolvent upon $conference\_list(x)$ is likely to instantiate x to a nonvariable term t, and then the literal $\neg split(t, y, z)$ can be resolved with only two, one or no clauses at all, depending on the form of t.

In all cases, it seems reasonable to choose an ordering and selection function in such a way that the first three clauses will be used as a *definition* of *split* so that we *unfold* this definition, i.e. replace the heads of these clauses with their bodies. Such an ordering would give us the best results if we have a right strategy of negative selection which says: select $\neg split(t, r, s)$ only if t is instantiated enough, or if we have no other choice.

In order to implement this idea we have to be able to formalize the right notion of a "definition" in a set of clauses. Such a formalization is undertaken in our paper, in the form of a calculus of *stratified resolution*. Stratified resolution is based on the following ideas that can be tracked down to earlier ideas developed in logic programming.

1. Logic programming is based on the idea of using definite clauses as definitions of relations. Similar to the notion of definite clause, we introduce a more general notion of a set of clauses *definite w.r.t. a set of relations*. These relations are regarded as defined by this set of clauses.
2. In logic programming, relations are often defined in terms of other relations. The notion of *stratification* [5, 1, 8] allows one to formalize the notion "P is defined in terms of Q". We use a similar idea of stratification, but in our case stratification must be related to a reduction ordering $\succ$ on literals.

Consider another example.

Example 2 The difficult problem is to find automatically the right ordering that makes the atom in the head of a "definition" greater than atoms in the body of this definition. Consider, for example, clauses defining reachability in a directed graph, where the graph is formalized by the binary relation *edge*:

$reachable(x, y)$:- $edge(x, y)$.
$reachable(x, z)$:- $edge(x, y), reachable(y, z)$.

There is no well-founded ordering stable under substitutions that makes the atom $reachable(x, z)$ greater than $reachable(y, z)$. So the standard ordered resolution with negative selection cannot help us in selecting the "right" ordering. The theory developed in this paper allows one to select only the literal $reachable(x, z)$ in this clause despite that this literal is not the greatest.

In addition to intelligent selection of literals in clauses, stratified resolution has a new kind of redundancy, not exploited so far in automated deduction. To explain this kind of redundancy, let us come back to Example 1. Suppose we have a clause

$$\neg split(\texttt{[}cade, www, lpar\texttt{]}, y, z). \tag{1}$$

Stratified resolution can resolve this clause with the first two clauses in the definition of *split*, obtaining two new clauses

$$\begin{array}{l} deduction(cade), split(\texttt{[}www, lpar\texttt{]}, y, z); \\ \neg deduction(cade), split(\texttt{[}www, lpar\texttt{]}, y, z). \end{array}$$

In resolution-based theorem proving these two clauses would be *added* to the search space. We prove that they can *replace* clause (1) thus making the search space smaller.

When the initial set of clauses contains no definitions or cannot be stratified, stratified resolution becomes the ordinary ordered resolution with negative selection. However, sets of clauses which contain definitions and can be stratified in our sense are often met in practice, since they correspond to definitions (maybe recursive) of relations of a special form. For example, a majority of TPTP problems can be stratified.

This paper is organized as follows. In Section 2 we define the ground version of stratified resolution and prove its soundness and completeness. Then in Section 3 we define stratified resolution with redundancies, a calculus in which a clause can be removed from the search space after a definition unfolding has been applied to it. Then in Section 4 we define a nonground version of stratified resolution with redundancies.

In this paper we deal with first-order logic without equality, we will only briefly discuss equality in Section 6.

Related work

There are not so many papers in the automated deduction literature relevant to our paper. Our formal system resembles SLD-resolution [6]. When the initial set of clauses is Horn, our stratified resolution with redundancies becomes SLD-resolution. The possibility for arbitrary selection for Horn clauses, and even in the case of equational logic was proved in [7]. For one of our proofs we used a renaming technique introduced in [2, 3].

Acknowledgments

The idea of this paper has appeared due to discussions of the second author during CADE-16 with Harald Ganzinger and Tanel Tammet. Harald Ganzinger

explained that the SATURATE system, tries to use an ordering under which the heads of "definition-like" clauses are the greatest in their clauses, and hence can be selected. This allows one to use definitions in the way they are often used in mathematics. Tanel Tammet explained that he tried to implement in his prover GANDALF "rewriting on the clause level", again for clauses that look like definitions, though the implementation was not ready by CASC-16.

2 Stratified resolution: the ground case

As usual, we begin with a propositional case, lifting to the general case will be standard.

Throughout this paper, we denote by $\mathcal{L}$ a set of propositional atoms. The literal complementary to a literal L is denoted by $\overline{L}$. For every set of atoms $\mathcal{P}$, we call a $\mathcal{P}$*-atom* any atom in $\mathcal{P}$, and a $\mathcal{P}$*-literal* any literal whose atom is in $\mathcal{P}$. A *clause* is a finite multiset of literals. We denote literals by L, M and clauses by C, D, maybe with indices. The empty clause is denoted by $\square$. We use the standard notation for multisets, for example write C_1, C_2 for the multiset union of two clauses C_1 and C_2, and write $L \in C$ if the literal L is a member of the clause C. For two ground clauses C_1 and C_2, we say that C_1 *subsumes* C_2, if C_1 is a submultiset of C_2. This means, for example, that A, A does *not* subsume A.

Let $\succ$ be a total well-founded ordering on $\mathcal{L}$. We extend this ordering on literals in $\mathcal{L}$ in the standard way, such that for every atom A we have $\neg A \succ A$ and there is no literal L such that $\neg A \succ L \succ A$. If L is a literal and C is a clause, we write $L \succ C$ if for every literal $M \in C$ we have $L \succ M$. Usually, we will write a clause as a disjunction of its literals. As usual, we write $L \succeq L'$ if $L \succ L'$ or $L = L'$ and write $L \succeq C$ if for every literal $M \in C$ we have $L \succeq M$. In this paper we assume that $\succ$ always denotes a fixed well-founded ordering.

We will now define a notion of selection function which makes a slight deviation from the standard notions (see e.g. [3]. However, the resulting inference system will be standard, and several following definitions will become simpler. We call a *selection function* any function σ on the set of clauses such that (i) $\sigma(C)$ is a set of literals in C, (ii) $\sigma(C)$ is nonempty whenever C is nonempty, and (iii) if $A \in \sigma(C)$ and A is a positive literal, then $A \succeq C$. If $L \in \sigma(C)$, we say that L is *selected by* σ in C. When we use a selection function, we underline selected literals, so when we write $\underline{A} \vee C$, this means that A (and maybe some other literals) are selected in $A \vee C$.

In our proofs we will use the result on completeness of the inference system of ordered binary resolution with negative selection (but our main inference system will be different). Let us define this inference system.

DEFINITION 3 Let σ be a selection function. The *inference system* $R^{\succ}_{\sigma}$ consists of the following inference rules:

1. *Positive ordered factoring*:

$$\frac{C \vee A \vee \underline{A}}{C \vee A},$$

where A is a positive literal.
2. *Binary ordered resolution with selection:*

$$\frac{C \vee \underline{A} \quad D \vee \underline{\neg A}}{C \vee D},$$

where $A \succ C$.

The following theorem is well-known (for example, a stronger statement can be found in [3]).

THEOREM 4 *Let S be a set of clauses. Then S is unsatisfiable if and only $\square$ is derivable from S in $R^{\succ}_{\sigma}$.*

The following two definitions are central to our paper.

DEFINITION 5 Let $\mathcal{P} \subseteq \mathcal{L}$ be a set of atoms. A set S of clauses is *Horn with respect to $\mathcal{P}$*, if every clause of S contains at most one positive $\mathcal{P}$-literal. A clause D is called *definite with respect to $\mathcal{P}$* if $\mathcal{D}$ contains exactly one positive $\mathcal{P}$-literal. Let $P \vee L_1 \vee \ldots \vee L_n$ be a clause definite w.r.t. $\mathcal{P}$ such that $P \in \mathcal{P}$. We will sometimes denote this clause by P :- $\overline{L}_1, \ldots, \overline{L}_n$.

DEFINITION 6 (stratification) We call a $\succ$*-stratification of* $\mathcal{L}$ any finite sequence $\mathcal{L}_n \succ \ldots \succ \mathcal{L}_0$ of subsets of $\mathcal{L}$ such that

1. $\mathcal{L} = \mathcal{L}_0 \cup \ldots \cup \mathcal{L}_n$;
2. If $m > n$, $A \in \mathcal{L}_m$, and $B \in \mathcal{L}_n$, then $A \succ B$.

We will denote $\succ$-stratifications by $\mathcal{L}_n \succ \ldots \succ \mathcal{L}_0$ and call them simply stratifications.

From now on we assume a fixed stratification of $\mathcal{L}$ of the form

$$\mathcal{Q}_n \succ \mathcal{P}_n \succ \mathcal{Q}_{n-1} \succ \mathcal{P}_{n-1} \succ \ldots \succ \mathcal{Q}_1 \succ \mathcal{P}_1 \succ \mathcal{Q}_0. \tag{2}$$

We denote $\mathcal{P} = \mathcal{P}_n \cup \ldots \cup \mathcal{P}_1$ and $\mathcal{Q} = \mathcal{Q}_n \cup \ldots \cup \mathcal{Q}_0$ and use this notation throughout the paper. Atoms in $\mathcal{P}$ will be denoted by P (maybe with indices).

Let C be a clause definite w.r.t. $\mathcal{P}$. Then C contains a positive literal $P \in \mathcal{P}_i$, for some i. We say that C *admits stratification (2)*, if all atoms occurring in C belong to $\mathcal{P}_i \cup \mathcal{Q}_{i-1} \cup \ldots \cup \mathcal{P}_1 \cup \mathcal{Q}_0$. Note that every such clause has the form

$$P \text{ :- } \underbrace{P_1, \ldots, P_k}_{\substack{\text{atoms in} \\ \mathcal{P}_i \cup \ldots \cup \mathcal{P}_1}}, \underbrace{L_1, \ldots, L_l}_{\substack{\mathcal{Q}_{i-1} \cup \ldots \cup \mathcal{Q}_0\text{-} \\ \text{literals}}}.$$

EXAMPLE 7 Consider the set consisting of four clauses: $A \vee B$, $A \vee \neg B$, $\neg A \vee B$, and $\neg A \vee \neg B$. This set is Horn with respect to $\{A\}$ and also with respect to $\{B\}$, but not with respect to $\{A, B\}$. This set of clauses admits stratification $\emptyset \succ \{A\} \succ \{B\}$, in which A is considered as a relation defined in terms of B, but also admits $\emptyset \succ \{B\} \succ \{A\}$, in which B is considered as defined in terms of A. This example shows that it is hard to expect to find the "greatest" (in any sense) stratification.

Let us fix a well-founded order $\succ$, a selection function σ, and a $\succ$-stratification (2).

DEFINITION 8 (stratified resolution) The inference system of *stratified resolution*, denoted $\mathcal{SR}$, consists of the following inference rules.

1. *Positive ordered factoring*:

$$\frac{C \vee A \vee \underline{A}}{C \vee A},$$

where A is a positive literal, and C contains no positive $\mathcal{P}$-literals.
2. *Binary ordered resolution with selection:*

$$\frac{C \vee \underline{A} \quad D \vee \underline{\neg A}}{C \vee D},$$

where $A \succ C$ and C, D, A contain no positive $\mathcal{P}$-literals.
3. *Definition unfolding:*

$$\frac{C \vee P \quad D \vee \underline{\neg P}}{C \vee D},$$

where $P \in \mathcal{P}$ and D contains no positive $\mathcal{P}$-literals. Note that in this rule we do not require that P be selected in $C \vee P$.

THEOREM 9 (soundness and completeness of stratified resolution)
Let S be a set of clauses Horn w.r.t. $\mathcal{P}$. Let, in addition, every clause in S definite w.r.t. $\mathcal{P}$ admits stratification (2). Then S is unsatisfiable if and only if $\Box$ is derivable from S in the system of stratified resolution.

PROOF. Soundness is obvious. To prove completeness, we use a technique of [3], (see also [2]) for proving completeness of resolution with free selection for Horn sets. Let us explain the idea of this proof. The inference rules of stratified resolution reminds us of the rules of ordered resolution with negative selection, but with a nonstandard selection function: in any clause $P \vee C$ definite w.r.t. $\mathcal{P}$ the literal P is selected. We could use Theorem 4 on completeness of ordered resolution with negative selection if P was maximal in $P \vee C$, since we could then select P in $P \vee C$ using a standard selection function. However, P is not necessarily maximal: there can be other $\mathcal{P}$-literals in C greater than P. What we do is to "rename" the clause $P \vee C$ so that P becomes greater than C.

Formally, let $\mathcal{P}'$ be a new set of atoms of the form P^n, where $P \in \mathcal{P}$, and n is a natural number. Denote by $\mathcal{L}'$ the set of atoms $\mathcal{P}' \cup \mathcal{Q}$. We will refer to the set $\mathcal{L}'$ as the *new language* as opposite to the *original language* $\mathcal{L}$. Define a mapping $\rho : \mathcal{L}' \longrightarrow \mathcal{L}$ by (i) $\rho(P^n) = P$ for any natural number n, (ii) $\rho(A) = A$ if $A \in \mathcal{Q}$. Extend the mapping to literals and clauses in the new language in a natural way: $\rho(\neg A) = \neg\rho(A)$ and $\rho(L_1 \vee \ldots \vee L_n) = \rho(L_1) \vee \ldots \vee \rho(L_n)$.

For every clause C Horn w.r.t. $\mathcal{P}$ of the original language we define a set of clauses C^ρ in the new language as follows.

1. If C is definite w.r.t. $\mathcal{P}$, then C has the form $P \vee \neg P_1, \ldots, \vee \neg P_k \vee D$, where $k \geq 0$ and D has no $\mathcal{P}$-literals. We define C^ρ as the set of clauses
$$\{P^{1+n_1+\ldots+n_k} \vee \neg P_1^{n_1} \vee \ldots \vee \neg P_k^{n_k} \vee D \mid n_1, \ldots, n_k \text{ are natural numbers } \}.$$
2. Otherwise, C has the form $\neg P_1 \vee \ldots \vee \neg P_k \vee D$, where $k \geq 0$ and D has no $\mathcal{P}$-literals. We define C^ρ as the set of clauses
$$\{\neg P_1^{n_1} \vee \ldots \vee \neg P_k^{n_k} \vee D \mid n_1, \ldots, n_k \text{ are natural numbers } \}.$$

For any set of clauses S Horn w.r.t. $\mathcal{P}$ we define S^ρ as the set of clauses $\bigcup_{C \in S} C^\rho$. We prove the following:

(3) $\quad S$ is satisfiable if and only if so is S^ρ.

Suppose S is satisfiable, then some valuation $\tau : \mathcal{L} \longrightarrow \{true, false\}$ satisfies S. Define a valuation $\tau' : \mathcal{L}' \longrightarrow \{true, false\}$ such that $\tau'(A) = \tau(A)$, if $A \in \mathcal{Q}$, and $\tau'(P^n) = \tau(P)$. It is not hard to argue that τ' satisfies S^ρ.

Now we suppose that S is unsatisfiable and show that S^ρ is unsatisfiable, too. We apply induction on the number k of $\mathcal{Q}$-literals occurring in S.

1. *Case* $k = 0$. Then S is a set of Horn clauses. This case has been considered in [3]. (The idea is that P^n is interpreted as P has derivation by SLD-resolution in n steps).
2. *Case* $k > 0$. For any set of clauses T and literal L, denote by T/L the set of clauses obtained from T be removing all clauses containing L and removing from the rest of clauses all occurrences of literals $\overline{L}$. Note that if L is a $\mathcal{Q}$-literal occurring in T, then T/L contains less $\mathcal{Q}$-literals than T. It is not hard to argue that T is unsatisfiable if and only so are both T/L and $T/\overline{L}$. Take any $\mathcal{Q}$-literal L occurring in S. Since S is unsatisfiable, so are both S/L and $S/\overline{L}$. By the induction hypothesis, $(S/L)^\rho$ and $(S/\overline{L})^\rho$ are unsatisfiable, too. It is easy to see that $(S/L)^\rho = S^\rho/L$ and $(S/\overline{L})^\rho = S^\rho/\overline{L}$, so both S^ρ/L and $S^\rho/\overline{L}$ are unsatisfiable. But then S^ρ is unsatisfiable too.

The proof of (3) is completed. Let us continue the proof of Theorem 9. Define an order $\succ'$ on $\mathcal{L}'$ as follows:

1. $\succ'$ and $\succ$ coincide on $\mathcal{Q}$-literals.

2. If L is a $\mathcal{Q}$-atom and P is a $\mathcal{P}$-atom, then for every n we let $L \succ' P^n$ if and only if $L \succ P$.
3. $P_1^{n_1} \succ' P_2^{n_2}$ if $P_1 \in \mathcal{P}_i$, $P_2 \in \mathcal{P}_j$ and $i > j$.
4. $P_1^{n_1} \succ' P_2^{n_2}$ if P_1 and P_2 belong to the same set $\mathcal{P}_i$ and $n_1 > n_2$.
5. $P_1^n \succ' P_2^n$ if P_1 and P_2 belong to the same set $\mathcal{P}_i$ and $P_1 \succ P_2$.

Using the selection function σ on clauses of the original language we will now define a selection function σ' on clauses of the new language. We will only be interested in the behavior of σ' on clauses Horn w.r.t. $\mathcal{P}'$, so we do not define it for other clauses.

1. If a clause C is definite w.r.t. $\mathcal{P}'$, then σ' selects in C the maximal literal.
2. If C contains no positive $\mathcal{P}$-literals, then σ' selects a literal L in C if and only if σ selects $\rho(L)$ in $\rho(C)$.

Note that our ordering $\succ$ is defined in such a way that for any clause $(P^n \vee C) \in S^\rho$ definite w.r.t. $\mathcal{P}'$ we have $P^n \succ C$, and hence P^n is always selected in this clause.

It is not hard to argue that σ' is a selection function. To complete the proof we use Theorem 4 on completeness of the system $R_{\sigma'}^{\succ'}$ applied to S^ρ.

Consider a derivation of a clause D' from S^ρ with respect to $R_{\sigma'}^{\succ'}$. We show simultaneously that:

(4) there exists a derivation of the clause $D = \rho(D')$ from S in the system $\mathcal{SR}$;

(5) if $D' \notin S^\rho$, then D' contains no positive $\mathcal{P}'$-literal.

We apply induction on the length of the derivation of D'. If $D' \in S^\rho$, then $\rho(D') \in S$, so the induction base is straightforward. Assume now that the derivation of D' is obtained by an inference from $D_1', \dots, D_n'$. We will show that D can be obtained by an inference in $\mathcal{SR}$ from $\rho(D_1'), \dots, \rho(D_n')$, this will imply (4), since $\rho(D_1'), \dots, \rho(D_n')$ can be derived in $\mathcal{SR}$ from S by the induction hypothesis. Consider the following cases.

1. *Case: $D' = C' \vee A$ is derived from $C' \vee A \vee \underline{A}$ by positive ordered factoring.* Then $A \succeq' C'$. Note that A cannot be a $\mathcal{P}'$-atom: $C' \vee P^n \vee P^n \notin S^\rho$ because all clauses in S^ρ are Horn w.r.t. $\mathcal{P}'$, and $C' \vee P^n \vee P^n$ cannot be derived by the induction hypothesis. Therefore, A is a $\mathcal{Q}$-atom, and hence $\rho(C' \vee A \vee A) = \rho(C') \vee A \vee A$.
   Moreover, C' contains no positive literal P^n because in this case the clause $(\rho(C') \vee A \vee A) \in S$ is definite w.r.t. $\mathcal{P}$, so it admits the stratification, so $P \succ A$, which contradicts to $A \succ' P^n$.
   Therefore, $C' \vee A$ contains no positive $\mathcal{P}'$-literal. Then by our definition of σ' and $\succ'$, A is maximal and selected in $\rho(C') \vee A \vee A$, so we can apply positive ordered factoring and derive $\rho(C') \vee A$ in $\mathcal{SR}$. It is easy to see that $\rho(C') \vee A = \rho(C' \vee A)$ and that $\rho(C' \vee A)$ contains no positive $\mathcal{P}'$-literal.

2. *Case: $D = C'_1 \vee C'_2$ is obtained from $C'_1 \vee \underline{A}$ and $C'_2 \vee \neg\underline{A}$ by binary ordered resolution with selection.* Denote $C_1 = \rho(C'_1)$ and $C_2 = \rho(C'_2)$. Consider two cases.
   (a) *Case: $A \in \mathcal{Q}$.* Then $\rho(A) = A$. In this case we show that

   $$\frac{C_1 \vee \underline{A} \quad C_2 \vee \underline{\neg A}}{C_1 \vee C_2}$$

   is an inference in $\mathcal{SR}$ by binary ordered resolution with selection.
   Since $A \succ' C'_1$, then $A \succ C_1$. By our definition of σ', A $(\neg A)$ is selected by σ in $C_1 \vee A$ (in $C_2 \vee \neg A$) because A $(\neg A)$ is selected by σ' in $C_1 \vee A$ $(C_2 \vee \neg A)$. It remains to check that C_1, C_2 contain no positive $\mathcal{P}$-literals. This is done exactly as in case of positive ordered factoring.
   (b) *Case: $A = P^n$.* We prove that

   $$\frac{C_1 \vee P \quad C_2 \vee \underline{\neg P}}{C_1 \vee C_2}$$

   is an inference in $\mathcal{SR}$ by definition unfolding. Indeed, since $\neg P^n$ is selected by σ' in $C'_2 \vee \neg P^n$, then $\neg P$ is selected by σ in $C_2 \vee \neg P$.
   As in the previous cases, we can prove that C'_1, C'_2 contains no positive literal of the form P^m.

Now we can conclude the proof of Theorem 9. Suppose that S in unsatisfiable. By (3), S^ρ is unsatisfiable. Then by Theorem 4 the empty clause $\square$ is derivable from S^ρ in $R^{\succ'}_{\sigma'}$. Hence, by (4) $\rho(\square)$ is derivable from S in $\mathcal{SR}$. But $\rho(\square) = \square$, so $\square$ is derivable from S in $\mathcal{SR}$.

We will now show that the condition on clauses to admit stratification is essential.

EXAMPLE 10 This example is taken from [7]. Consider the following set of clauses:

$\neg Q \vee \underline{R}$	$\neg P \vee \underline{Q}$	$\neg R \vee \underline{\neg Q}$
$\neg Q \vee \underline{\neg P}$	$\neg P \vee \underline{\neg R}$	
$\neg R \vee P$	$R \vee Q \vee P$	

This clause is unsatisfiable and definite w.r.t. P. Consider the ordering $R \succ Q \succ P$. However, the empty clause cannot be derived from it, even if tautologies are allowed. Indeed, the conclusion of any inference by stratified resolution is subsumed by one of the clauses in this set. The problem is that the clause $R \vee P \vee Q$ admits no $\succ$-stratification.

3 Redundancies in stratified resolution

In this section we make stratified resolution into an inference system on sets of clauses and add two kinds of inferences that remove clauses from sets. Then we prove completeness of the resulting system using a new technique of *traces*.

The derivable objects in the inference system of stratified resolution with redundancies are sets of clauses. Each inference has exactly one premise S and conclusion S', we will denote such an inference by $S \triangleright S'$. We assume an ordering $\succ$, stratification and selection function be defined as in the case of $\mathcal{SR}$.

In this section we always assume that S_0 is an initial set of clauses Horn w.r.t. $\mathcal{P}$ and having the following property: for every $P \in \mathcal{P}$ S_0 contains only a finite number of clauses in which P is a positive literal.

DEFINITION 11 (stratified resolution with redundancies) The inference system of *stratified resolution with redundancies*, denoted $\mathcal{SRR}$ consists of the following inference rules.

1. Suppose that $(C \vee A \vee \underline{A}) \in S$, A is a positive literal, and C contains no positive $\mathcal{P}$-literals. Then

$$S \triangleright S \cup \{C \vee A\},$$

   is an inference by *positive ordered factoring*.
2. Suppose that $\{C \vee \underline{A}, D \vee \neg\underline{A}\} \subseteq S$, where $A \succ C$ and C, D, A contain no positive $\mathcal{P}$-literals. Then

$$S \triangleright S \cup \{C \vee D\}$$

   is an inference by *binary ordered resolution*.
3. Suppose $(C \vee \underline{\neg P}) \in S$, where $P \in \mathcal{P}$, and C contains no positive $\mathcal{P}$-literals. Furthermore, suppose that $P \vee D_1, \ldots, P \vee D_k$ are all clauses in S containing P positively. Then

$$S \triangleright (S - \{C \vee \underline{\neg P}\}) \cup \{C \vee D_1, \ldots, C \vee D_n\}$$

   is an inference by *definition rewriting*. Note that this inference *deletes* the clause $C \vee \underline{\neg P}$. Moreover, if S contains no clauses containing the positive literal P, then $C \vee \underline{\neg P}$ is deleted from the search space and not replaced by any other clause.
4. Suppose $\{C, D\} \in S$, $C \neq D$ and C subsumes D. Then

$$S \triangleright S - \{D\}$$

   is an inference by *subsumption*.

For the new calculus, we have to change the notion of derivation into a new one. We call a *derivation* from a set of clauses S_0 any sequence of inferences:

$$S_0 \triangleright S_1 \triangleright S_2 \triangleright \ldots ,$$

possibly infinite. The derivation is said to *succeed* if some S_i contains $\square$, and *fail* otherwise.

Consider a derivation $\mathbf{S} = S_0 \triangleright S_1 \triangleright \ldots$. The set $\bigcup_i \bigcap_{j \geq i} S_j$ is called the *limit* of this derivation and denoted by $\lim_{\mathbf{S}}$. The derivation is called *fair* if the following three conditions are fulfilled:

1. If $(C \vee A \vee \underline{A})$ belongs to the limit of $\mathbf{S}$ and $A \succ C$, then some inference in $\mathbf{S}$ is the inference by positive ordered factoring

$$S_i \triangleright S_i \cup \{C \vee A\}.$$

2. If $C \vee \underline{A}, D \vee \underline{\neg A}$ belong to the limit of $\mathbf{S}$, $A \succ C$ and C, D, A contain no positive $\mathcal{P}$-literals, then some inference in $\mathbf{S}$ is the inference by binary ordered resolution

$$S_i \triangleright S_i \cup \{C \vee D\}.$$

3. Let a clause $C \vee \underline{\neg P}$ belongs to the limit of $\mathbf{S}$ and $P \in \mathcal{P}$. Then some inference in $\mathbf{S}$ is the inference by definition rewriting

$$S_i \triangleright (S_i - \{C \vee \underline{\neg P}\}) \cup \{C \vee D_1, \ldots, C \vee D_n\}.$$

We call a selection function σ *subsumption-stable* if it has the following property. Suppose C subsumes D, L is a literal in D selected by σ and $L \in C$. Then L is also selected in C by σ. It is evident that subsumption-stable selection functions exist.

THEOREM 12 (soundness and completeness of $\mathcal{SRR}$) *Let S_0 be a set of clauses Horn w.r.t. $\mathcal{P}$ and every clause definite w.r.t. $\mathcal{P}$ in S_0 admits stratification (2). Let σ be a subsumption-stable selection function. Consider derivations in $\mathcal{SRR}$. (i) If any derivation from S_0 succeeds, then S_0 is unsatisfiable. (ii) If S_0 is unsatisfiable, then every fair derivation from S succeeds.*

PROOF. Soundness is easy to prove. Let a derivation $S_0 \triangleright \ldots \triangleright S_n$ succeed. It is easy to see that every clause in every S_i is a logical consequence of S_0. Then $\square$ is a logical consequence of S_0, and hence S_0 is unsatisfiable.

To prove completeness is more difficult. We will introduce several notions, prove some intermediate lemmas, and then come back to the proof of this theorem.

The main obstacle in the proof of completeness is that some clauses can be deleted from the search space, when we apply definition rewriting or subsumption. If we only had subsumption, the proof could be done by standard methods based on clause orderings. However, clause rewriting can rewrite clauses into larger ones. For example, consider a set S of clauses that contains two clauses definite w.r.t. $\mathcal{P}$: P_1 `:-` P_2 and P_2 `:-` P_1. Then the following is a valid derivation:

$$S \cup \{\neg P_1\} \triangleright S \cup \{\neg P_2\} \triangleright S \cup \{\neg P_1\} \triangleright \ldots,$$

so independently of a choice of ordering $\succ$ one of these inferences replaces a smaller clause by a bigger one.

To prove completeness we will use a novel technique of *traces*. This technique was originally introduced in [10], where it was used to prove completeness of a system of modal logic with subsumption. Intuitively, a trace is whatever remains of a clause when the clause is deleted. Unlike [10], in this paper a trace can be an infinite set of clauses.

Suppose that S_0 is the set of initial clauses, Horn w.r.t. $\mathcal{P}$. Consider any clause D that contains no positive $\mathcal{P}$-literals. Consider the following one-player game that builds a tree of clauses. Initially, the tree contains one root node D. We call a node C *closed* if either $C = \Box$ or the literal selected in C by σ is a $\mathcal{Q}$-literal. An *open node* is any leaf that is not closed. At every step of the game the player has two choices of moves:

1. *(Subsumption move).* Select any leaf C (either closed or open) and add to C as the child node any clause C' such that C' subsumes C and $C' \neq C$.
2. *(Expansion move).* Select an open leaf C. If the literal selected in C by σ is a negative $\mathcal{P}$-literal $\neg P$, then C has the form $\underline{\neg P} \vee C'$. (As we shall see later, no clause in the tree can contain a positive $\mathcal{P}$-literal, so in this case a negative $\mathcal{P}$-literal is always selected). Let all clauses in S_0 that contain the positive literal P be $P \vee D_1, \ldots, P \vee D_n$. Then add to this node as children all nodes $C' \vee D_1, \ldots, C' \vee D_n$.

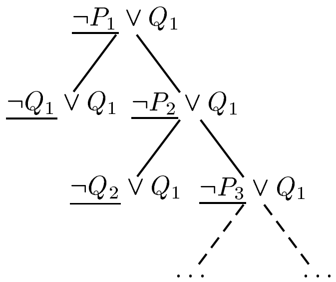
Using the property that S_0 is Horn w.r.t. $\mathcal{P}$ it is not hard to argue that no clause in the tree can contain a positive $\mathcal{P}$-literal, so an open leaf always contains a selected negative $\mathcal{P}$-literal.

A game is *fair* if every non-closed node is selected by the player. Let us call a *tree for D* any tree obtained as a limit of a fair game. We call a *cover for D* the set of all closed leaves in any tree T for D. We will say that the cover is *obtained from* the tree T and denote the cover by cl_T.

EXAMPLE 13 Suppose that the set of clauses S_0 is

$$\{P_i \vee \neg Q_i \mid i = 1, 2, \ldots\} \cup \{P_i \vee \neg P_{i+1} \mid i = 1, 2, \ldots\},$$

where all P_i's are $\mathcal{P}$-literals and all Q_i's are $\mathcal{Q}$-literals. Suppose that the selection function σ always selects a $\mathcal{P}$-literal in a clause that contains at least one $\mathcal{P}$-literal. Then one possible tree for $\underline{\neg P_1} \vee Q_1$ is as follows:



In this tree no subsumption move was applied. The cover for $\underline{\neg P_1} \vee Q_1$ obtained from this tree is the set of clauses

$$\{\underline{\neg Q_i} \vee Q_1 \mid i = 1, 2, \ldots\}.$$

Let us prove some useful properties of covers.

LEMMA 14 *Every cover for a clause C is also a cover for any clause $C \vee D$.*

PROOF. Let S be a cover for C and T be the tree from which this cover is obtained. Apply a subsumption move to $C \vee D$

$$\begin{array}{c} C \vee D \\ | \\ C \end{array}$$

and extend this tree by the tree T below C. Evidently, we obtain a tree for $C \vee D$ and the cover obtained from this tree is also S.

The following straightforward lemma asserts that it is enough to consider trees with no repeated subsumption moves.

LEMMA 15 *For every cover S for a clause C there exists a tree T for C such that $S = cl_T$ and no subsumption move in T was applied to a node obtained by a subsumption move.*

PROOF. Let T' be a tree for C such that $S = cl_{T'}$. Suppose that T' contains a node obtained by a subsumption move so that a subsumption move is applied to this node as well:

$$\begin{array}{c} D_1 \vee D_2 \vee D_3 \\ | \\ D_1 \vee D_2 \\ | \\ D_1 \end{array}$$

Evidently, these two subsumption moves can be replaced by one subsumption move:

$$\begin{array}{c} D_1 \vee D_2 \vee D_3 \\ | \\ D_1 \end{array}$$

In this way we can eliminate all sequences of subsumption moves and obtain a required tree T.

Let C be a clause and $\mathbf{S}$ be a derivation. We call a *trace* of C in $\mathbf{S}$ any cover S for C such that $S \subseteq \lim_{\mathbf{S}}$. When we speak about a trace of C in $\mathbf{S}$ we do not assume that C occurs in $\mathbf{S}$.

LEMMA 16 *Let* $\mathbf{S} = S_0 \triangleright S_1 \triangleright \ldots$ *be a fair derivation. Then every clause* D *occurring in any* S_i *has a trace in* $\mathbf{S}$.

PROOF. We will play a game which builds a tree T such that cl_T is a trace of D in $\mathbf{S}$. We will define a sequence of trees $T_1, T_2, \ldots$ such that each T_{j+1} is obtained from T_j by zero or more moves, and T is the limit of this sequence. The initial tree T_1 consists of the root node D. Each tree T_{j+1} is obtained from T_j by inspecting the inference $S_j \triangleright S_{j+1}$ as follows.

1. Suppose this inference is a subsumption inference
$$S_j \triangleright S_j - \{C \vee D\},$$
where $C \in S$ and $C \vee D$ is a leaf node in T_j. Then make a subsumption move for each such leaf, adding C to it as the child.
2. Suppose this inference is a definition rewriting inference
$$S_j \triangleright S_j - \{C \vee \underline{\neg P}\} \cup \{C \vee D_1, \ldots, C \vee D_n\},$$
such that $C \vee \underline{\neg P}$ is a leaf in T_j. Then make an expansion move for each such leaf, adding to it the clauses $C \vee D_1, \ldots, C \vee D_n$ as children.
3. In all other cases make no move at all.

Let us make several observations about the tree T and the game.

1. *Every leaf in any tree* T_j *is a member of some* S_k. This is proved by induction on j. For T_1 this holds by the definition of T_1, since it contains one leaf $D \in S_i$. By our construction, every leaf of any T_{j+1} which is not a leaf of T_j is a member of S_{j+1}.
It remains to prove that every node which is a leaf in both T_j and T_{j+1} belongs to S_{j+1}. Take any such node C, by the induction hypothesis it belongs to S_j. Suppose, by contradiction, $C \notin S_{j+1}$, then C has been removed by either subsumption or definition rewriting inference. But by our construction, in both cases C will become a nonleaf in T_{j+1}, so we obtain a contradiction.
2. *The game is fair.* Suppose, by contradiction, that an open leaf has never been selected during the game. Then this leaf contains a clause $C \vee \underline{\neg P}$. Let T_j be any tree containing this open leaf. Then the leaf belongs to all trees T_k for $k \geq j$. By the previous item, $C \vee \underline{\neg P} \in S_k$ for all $k \geq j$. Therefore, the clause $C \vee \neg P$ belongs to the limit of $\mathbf{S}$, but definition rewriting has not been applied to it. This contradicts to the fairness of $\mathbf{S}$.
3. *Every closed leaf* C *in* T *belongs to the limit of* $\mathbf{S}$. Let this leaf first appeared in a tree T_j. Then it belongs to all trees T_k for $k \geq j$. By the first item, $C \in S_k$ for all $k \geq j$, but then $C \in \lim_{\mathbf{S}}$.

Now consider cl_T. Since T was built using a fair game, cl_T is a cover for D. But we proved that every element of cl_T belongs to the limit of $\mathbf{S}$, hence cl_T is also a trace of D in $\mathbf{S}$.

LEMMA 17 *Let* $\mathbf{S} = S_0 \rhd S_1 \rhd \ldots$ *be a fair derivation. Further, let* E *be a clause derivable from* S_0 *in* $\mathcal{SR}$. *Then* D *has a trace in* $\mathbf{S}$.

PROOF. The proof is by induction on the derivation of E in $\mathcal{SR}$. When the derivation consists of 0 inferences, $E \in S_0$. By Lemma 16 every clause of every S_i has a trace in $\mathbf{S}$, so E has a trace.

When the derivation has at least one inference, we consider the last inference of the derivation. By the induction hypothesis, all premises of this inference have traces in $\mathbf{S}$. We consider three cases corresponding to the inference rules of stratified resolution. In all cases, using Lemma 15 we can assume that no tree contains a subsumption move applied to a node which itself is obtained by a subsumption move.

1. *The last inference is by positive ordered factoring.*

$$\frac{C \vee A \vee \underline{A}}{C \vee A} \; ,$$

   Consider a trace S of $C \vee A \vee \underline{A}$ in $\mathbf{S}$ and a tree T from which this trace was obtained. Since A is selected by σ in C, there are only two possibilities for T:

   (a) *Case: T consists of one node $C \vee A \vee A$.* Then $\{C \vee A \vee A\}$ is a trace, and hence $(C \vee A \vee A) \in \lim_{\mathbf{S}}$. Since $\mathbf{S}$ is fair, some S_i contains the clause $C \vee A$ obtained from $C \vee A \vee A$ by positive ordered factoring. Then by Lemma 16, $C \vee A$ has a trace in $\mathbf{S}$.

   (b) *Case: the first move of T is a subsumption move.* This subsumption move puts a clause C' which subsumes $C \vee A \vee A$ as the child to $C \vee A \vee A$. Consider two cases.

      i. *Case: C' has a form $C'' \vee A \vee A$, where C'' subsumes C.* Since the selection function is subsumption-stable, A is selected in $C'' \vee A \vee A$ by σ, so the tree T contains no node below $C'' \vee A \vee A$. Then $(C'' \vee A \vee A) \in \lim_{\mathbf{S}}$. Since $\mathbf{S}$ is fair, some S_i contains the clause $C'' \vee A$ obtained from $C'' \vee A \vee A$ by positive ordered factoring. By Lemma 16, $C'' \vee A$ has a trace in $\mathbf{S}$. But $C'' \vee A$ subsumes $C \vee A$, so by Lemma 14 this trace is also a trace of $C \vee A$.

      ii. *Case: C' subsumes $C \vee A$.* Note that cl_T is also a trace of C', so by Lemma 14 cl_T is also a trace of $C \vee A$.

2. *The last inference is by binary ordered resolution with selection:*

$$\frac{C \vee \underline{A} \quad D \vee \underline{\neg A}}{C \vee D} \; ,$$

   where $A \succ C$ and C, D, A contain no positive $\mathcal{P}$-literals. Consider traces S_1, S_2 of $C \vee A$ and $D \vee \neg A$, respectively and trees T_1, T_2 from which these traces were obtained. We consider two simple cases and then the remaining case

(a) If the first move of T_1 is a subsumption move that replaces $C \vee A$ by a clause C' that subsumes C, then C' has a trace in $\mathbf{S}$, but C subsumes $C \vee D$, so by Lemma 14 this trace is also a trace of $C \vee D$.

(b) Likewise, if the first move of T_2 is a subsumption move that replaces $D \vee \neg A$ by a clause D' that subsumes D, then D' has a trace in $\mathbf{S}$, so by Lemma 14 this trace is also a trace of $C \vee D$.

(c) If neither of the previous cases takes place, then either T_1 consists of one node or the top move in T_1 is a subsumption move:

$$\begin{array}{c} C \vee \underline{A} \\ | \\ C' \vee A \end{array}$$

such that C' subsumes C. Note that in the latter case, since the selection function is subsumption-stable, A is selected in $C' \vee A$. In both cases the limit of $\mathbf{S}$ contains a clause $C' \vee \underline{A}$ such that C' subsumes C (in the former case we can take C as C').
Likewise, we can prove that the limit of $\mathbf{S}$ contains a clause $D' \vee \underline{\neg A}$ such that D' subsumes D. Since $\mathbf{S}$ is fair, some S_i contains the clause $C' \vee D'$ obtained from $C' \vee \underline{A}$ and $D' \vee \neg \underline{A}$ by binary ordered resolution with selection. By Lemma 16, $C' \vee D'$ has a trace in $\mathbf{S}$. But $C' \vee D'$ subsumes $C \vee D$, so by Lemma 14 $C \vee D$ has a trace in $\mathbf{S}$.

3. *The last inference is by definition unfolding:*

$$\frac{C \vee P \quad D \vee \underline{\neg P}}{C \vee D} \ ,$$

where $P \in \mathcal{P}$. Consider a S of $D \vee \underline{\neg P}$, and a tree T from which this trace was obtained.

(a) If the first move of T is a subsumption move that replaces $D \vee \neg P$ by a clause D' that subsumes D, then D' has a trace in $\mathbf{S}$, so by Lemma 14 this trace is also a trace of $C \vee D$.

(b) If the previous case does not take place, there are two possibilities for the top move of T: it is either an expansion move or a subsumption move followed by an expansion move. We consider the former case, the latter case is similar. The top of the tree T has the form

$$\begin{array}{ccccc} & & D \vee \neg P & & \\ & / & | & \backslash & \\ \dots & & C \vee D & & \dots \end{array}$$

Denote the subtree of T rooted at $C \vee D$ by T'. Note that $cl_{T'}$ is a cover of $C \vee D$. But since T' is a subtree of T, we have $cl_{T'} \subseteq cl_T$. Since cl_T is a trace, we also have $cl_T \subseteq \lim_{\mathbf{S}}$, then $cl_{T'} \subseteq \lim_{\mathbf{S}}$, and hence $cl_{T'}$ is a trace. So $C \vee D$ has a trace in $\mathbf{S}$.

The proof is completed.

We can now easily complete the proof of completeness of stratified resolution with redundancies.

PROOF (of Theorem 12, continued). Suppose S_0 is unsatisfiable. Take any fair derivation $\mathbf{S} = S_0 \triangleright S_1 \triangleright \ldots$. By Theorem 9 there exists a derivation of $\square$ from clauses in S_0 by stratified resolution. By Lemma 17, $\square$ has a trace in $\mathbf{S}$, i.e. a cover whose members belong to the limit of $\mathbf{S}$. By the definition of a cover, $\square$ has only one cover $\{\square\}$. Then $\square$ belongs to the limit of $\mathbf{S}$, and hence $\mathbf{S}$ is successful.

The proof can be easily modified for a system with redundancies in which clauses containing positive $\mathcal{P}$-literals can also be subsumed and in which tautologies are deleted.

4 Nonground case

In this section we introduce the general, nonground, version of stratified resolution with redundancies.

Most of the definitions given below are obtained by a simple modification of the definition for the ground case. In this section we treat $\mathcal{L}$, $\mathcal{P}$, and $\mathcal{Q}$ as sets of *relations* but not atoms. To define stratification, we use a *precedence relation* $\succ_{\mathcal{L}}$, i.e. a total ordering on $\mathcal{L}$. We call a $\mathcal{P}$-literal any literal whose relation belongs to $\mathcal{P}$, and similar for other sets of relations instead of $\mathcal{P}$. The notions of *Horn set w.r.t. $\mathcal{P}$*, *definite clause w.r.t. $\mathcal{P}$*, *stratification*, and *clause admitting a stratification* are the same as in the ground case.

Now we have to define an ordering $\succ$ corresponding to our notion of stratification. We require $\succ$ to be well-founded ordering on atoms, stable under substitutions (i.e. $A \succ B$ implies $A\theta \succ B\theta$) and total on ground atoms. In addition, we require that $\succ$ is compatible with the precedence relation in the following sense: if $A_1 \succ_{\mathcal{L}} A_2$, then $A_1(t_1, \ldots, t_n) \succ A_2(s_1, \ldots, s_m)$ for all relations A_1, A_2 and terms $t_1, \ldots, t_n$ and $s_1, \ldots, s_m$. It is obvious how to modify recursive path orderings or Knuth-Bendix orderings so that they satisfy this requirement.

In the definition of selection function for nonground case we revert in fact to one of earlier notions given in [4]. We define a (nonground) *selection function* to be a function σ on the set of clauses such that (i) $\sigma(C)$ is a set of literals in C, (ii) $\sigma(C)$ is nonempty whenever C is nonempty, and (iii) at least one negative literal or otherwise all maximal literals must be selected. Similar to the ground case we assume that selection functions are subsumption-stable in the following sense: if a literal L is selected in $C \vee L$ and D is a submultiset of C, then L is selected in $D \vee L$.

Let us fix an order $\succ$, a selection function σ, and stratification (2).

DEFINITION 18 (stratified resolution with redundancies) The inference system of *stratified resolution with redundancies*, denoted $\mathcal{SRR}$ consists of the following inference rules.

1. *Positive ordered factoring* is the following inference rule:

$$S \triangleright S \cup \{(C \vee A_1)\theta\}$$

such that (i) S contains a clause $C \vee \underline{A_1} \vee \underline{A_2}$, (ii) A_1, A_2 are positive literals, (iii) C contains no positive $\mathcal{P}$-literals, and (iv) θ is a most general unifier of A_1 and A_2.

2. *Binary ordered resolution with selection* is the following inference rule:

$$S \triangleright S \cup \{(C \vee D)\theta\}$$

such that (i) S contains clauses $C \vee \underline{A_1}$ and $D \vee \underline{\neg A_2}$, (ii) $A_1\theta \notin C\theta$, (iii) C, D, A_1 contain no positive $\mathcal{P}$-literals, and (iv) θ is a most general unifier of A_1 and A_2.

3. *Definition rewriting*. Suppose $(C \vee \underline{\neg P}) \in S$, where $P \in \mathcal{P}$, and C contains no positive $\mathcal{P}$-literals. Furthermore, suppose that $P_1 \vee D_1, \ldots, P_k \vee D_k$ are all clauses in S such that P_i is unifiable with P. Then

$$S \triangleright (S - \{C \vee \neg P\}) \cup \{(C \vee D_1)\theta_1, \ldots, (C \vee D_n)\theta_n\},$$

where each θ_i is a most general unifier of P and P_i, is an inference by *definition rewriting*.

THEOREM 19 *Let S be a set of clauses Horn w.r.t. $\mathcal{P}$. Let, in addition, every clause in S definite w.r.t. $\mathcal{P}$ admit stratification (2). Then S is unsatisfiable if and only if every fair derivation from S in the system $\mathcal{SRR}$ contains the empty clause.*

The proof can be obtained by standard lifting with a special care of selection function.

5 How to select a stratification

Example 7 shows that a set of clauses may admit several different stratifications. How can choose a "good" stratification?

Suppose that a set of clauses S is Horn w.r.t. $\mathcal{P}$. Then we can always use the stratification

$$\emptyset \succ \mathcal{P} \succ \mathcal{Q},$$

in which any $\mathcal{P}$-atom is greater than any $\mathcal{Q}$-atom. Unfortunately, this stratification may be not good enough, since it gives us too little choice for selecting positive $\mathcal{Q}$-literals. Let us illustrate this for clauses of Example 1. Assume that $\mathcal{P}$ is $\{split\}$. We obtain the stratification

$$\emptyset \succ \{split\} \succ \{deduction, conference\_list\}.$$

This stratification does not allow us to select the literal $conference\_list(x)$ in

$$\neg split(x, y, z) \lor conference\_list(x),$$

while intuitively it should be the right selection.

This observation shows that it can be better to split the set of atoms into as many strata as possible. Then we will have more options for selecting positive $\mathcal{Q}$-atoms in clauses. In Example 1, a more flexible stratification would be

$$\{conference\_list\} \succ \{split\} \succ \{deduction\}.$$

We are planning experiments with the choice of stratification using our theorem prover Vampire [9].

6 Conclusion

We will mention several interesting open problems associated with stratified resolution.

In our proofs, we could not get rid of an annoying condition on selection function to be subsumption-stable.

PROBLEM 1 *Is stratified resolution with redundancies complete with selection functions that are not subsumption-stable?*

It is quite possible this can be done, but require a different proof technique.

PROBLEM 2 *Find new techniques for proving completeness of stratified resolution with redundancies.*

PROBLEM 3 *Find a powerful generalization of definite resolution for logic with equality.*

There is quite a simple generalization based on the following transformation of clauses definite w.r.t. $\mathcal{P}$: we can replace any clause $P(t_1, \ldots, t_n)$ `:-` C by $P(x_1, \ldots, x_n)$ `:-` $x_1 = t_1, \ldots, x_n = t_n, C$. Then we can define stratified resolution and prove its completeness in essentially the same way as before. However, in this case *any* clause containing $\neg P(s_1, \ldots, s_n)$ will be unifiable with the head of *any* clause defining P, so the gain of using definition rewriting is not obvious any more.

The standard semantics of stratified logic programs are based on nonmonotonic reasoning. Stratified resolution makes one think of a logic that combines nonmonotonic reasoning with the monotonic resolution-based reasoning. Such a logic, its semantics and ways of automated reasoning in it, should be investigated. So we have

PROBLEM 4 *Find a combination of stratified resolution with nonmonotonic logics.*

Stratified resolution is different from ordered resolution with negative selection in that it allows one to select heads of clauses, even when they are not maximal in their clauses. It is interesting to see if this can give us new decision procedures for decidable fragments of predicate calculus. So, we have

PROBLEM 5 *Find new decision procedures based on stratified resolution.*

7 Recent developments

Recently, Robert Nieuwenhuis proposed a new interesting method for proving completeness of stratified resolution. His method gives a positive answer to Problem 1: the restriction on selection functions to be subsumption-stable is not necessary. In a way, it also answers to Problem 2, since his technique is rather different from ours. He also proposed a more concise definition of stratification and clauses admitting stratification in terms of well quasi orderings.

References

1. K.R. Apt and H.A. Blair. Towards a theory of declarative knowledge. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 89–148. Morgan Kaufmann, 1988.
2. L. Bachmair and H. Ganzinger. A theory of resolution. Research report MPI-I-97-2-005, Max-Planck Institut für Informatik, Saarbrücken, Germany, 1997. To appear in Handbook of Automated Reasoning, edited by A. Robinson and A. Voronkov.
3. L. Bachmair and H. Ganzinger. A theory of resolution. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*. Elsevier Science and MIT Press, 2000. To appear.
4. L. Bachmair, H. Ganzinger, C. Lynch, and W. Snyder. Basic paramodulation. *Information and Computation*, 121:172–192, 1995.
5. A. Van Gelder. Negation as failure using tight derivations for general logic programs. In J. Minker, editor, *Deductive Databases and Logic Programming*, pages 149–177. Morgan Kaufmann, 1988.
6. R.A. Kowalski and D. Kuehner. Linear resolution with selection function. *Artificial Intelligence*, 2:227–260, 1971.
7. C. Lynch. Oriented equational logic is complete. *Journal of Symbolic Computations*, 23(1):23–45, 1997.
8. H. Przymusinski. On the declarative semantics of deductive databases and logic programs. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 193–216. Morgan Kaufmann, 1988.
9. A. Riazanov and A. Voronkov. Vampire. In H. Ganzinger, editor, *Automated Deduction—CADE-16. 16th International Conference on Automated Deduction*, volume 1632 of *Lecture Notes in Artificial Intelligence*, pages 292–296, Trento, Italy, July 1999.
10. A. Voronkov. How to optimize proof-search in modal logics: a new way of proving redundancy criteria for sequent calculi. Preprint 1, Department of Computer Science, University of Manchester, February 2000. To appear in LICS'2000.