Indefinite Models and Parallel Positive Resolution for Disjunctive Stratified Logic Programs

Marek A. Suchenek
California State University – Dominguez Hills
Carson, CA 90747 U.S.A.
e-mail addr: Suchenek@csudh.edu

Abstract

The subject of this paper is the semantics of propositional disjunctive logic programs and indefinite deductive data bases. The concept of the indefinite model of a disjunctive logic program is introduced. The algorithm of computing the indefinite model and an algorithm of evaluation of clauses in the indefinite model, both using a parallel positive variant of resolution, are presented. The evaluation algorithm is extended to cover the stratified programs. The complexity of these algorithms and their scope of applicability is analyzed.

Key words: hierarchically-minimal semantics, indefinite deductive data bases, complexity of minimal entailment, prioritized circumscription,

1 Introduction

In this paper we investigate a resolution based method for evaluation of queries to propositional indefinite data bases by means of, what we call, an indefinite modeling. Our approach extends the well known results of research in semantics of negation, particularly the closed-world assumption, circumscription, and minimal model theory (cf.[Lif85a, Lif85b, YH85, Suc89, SS90, Suc90, Suc93, Suc94]). Its generalization over non-propositional data bases (i.e., with universal quantifier ∀) is the subject of forthcoming [Suc97].

Under the minimal-model semantics, a query q to a disjunctive deductive data base p is answered positively if q is true in all minimal models of p. This fact is usually articulated by $p \vdash_{\text{min}} q$ (p minimally entails q). The minimal-model semantics has the following neat characterization in terms of first-order provability (propositional, in this paper) of positive (i.e., negation-free) sentences:

$$p \vdash_{\text{min}} q \iff Cn_{\text{Pos}}(p \land q) \subseteq Cn(p) \quad (1)$$

where $Cn_{\text{Pos}}(X)$ is the set of all positive logical consequences of X, and $Cn$ is the set of all logical consequences of X (see [YH85] for the proof of the case of q being a disjunction of negative ground literals, and [Suc89] for the proof of the general case with universal quantifier). Intuitively, the right-hand side of (1) means that every positive consequence of $p \land q$ is provable from p itself. In particular, if q is a positive sentence then $p \vdash_{\text{min}} q$ is equivalent to $p \vdash q$. Consequently, characterization (1) yields the following corollary: p and p’ have the same positive consequences iff for every clause q, $p \vdash_{\text{min}} q$ and $p’ \vdash_{\text{min}} q$ are equivalent.

The above fact gives rise to the following scheme of deciding $p \vdash_{\text{min}} q$. If p’ is a formula equivalent to the conjunction of all positive consequences of p then decide $p’ \vdash_{\text{min}} q$ instead of $p \vdash_{\text{min}} q$. It is being hoped that, at least in certain cases, one can find p’ for which the decision problem $p’ \vdash_{\text{min}} q$ is easier than $p \vdash_{\text{min}} q$. Of course, p’ may be assumed to be a positive formula. In this paper we consider two forms of such alternate p’: a positive reduced disjunctive form (DF), and a positive reduced conjunctive form (CF). The former constitutes, in fact, the familiar set of minimal models of p, and allows for straightforward algorithm of deciding $p’ \vdash_{\text{min}} q$. We call the latter an indefinite model of p, and develop for it a method of deciding $p’ \vdash_{\text{min}} q$ which uses the criterion (1) without actually computing all the positive consequences of $p’ \land q$. This method involves a transformation of data base p into suitable DF or CF, using a parallel positive variant of resolution, complete with respect to positive conclusions. Although in many cases, computing an indefinite model p’ of p a program (also a CF-formula) seems quite easy, the problem $p’ \vdash_{\text{min}} q$ for $p’$ in a conjunctive form is far from being trivial.

Our method of indefinite modeling extends over strat-
ified data bases whose semantics is given by the hierarchically-minimal entailment $\vdash_{H\text{min}}$ of [SS90] (or equivalently, by prioritized circumscription of [Lif85a, Lif85b]). This stratified case and its semantics constitutes the central point of interest in this paper, and is briefly introduced below.

The very form of a **stratified** program $p$ (specifically, the occurrences of negation in the bodies of its clauses) defines a hierarchy $L$ of first-order languages $L_i$ which extend alphabets of one another, and partitions $p$ onto strata $\sigma_i$. The concept of hierarchically-minimal model of $p$ differs from the regular (that is, non-hierarchical) one in that its construction follows a series of model minimizations, one for each stratum $\sigma_i$. The “basis step” in this inductive process consists of the construction of a minimal model of the innermost stratum $\sigma_0$ of $p$ in its smallest language $L_0$. The “inductive step” proceeds as follows. Let the language $L_i'$ of stratum $\sigma_i'$ be the closest extension of the language $L_i$ of stratum $\sigma_i$ in the hierarchy $L$. The hierarchically-minimal model of $\sigma_i'$ in $L_i'$ is obtained by expansion of hierarchically-minimal model of $\sigma_i$ in $L_i$ to a minimal model of $\sigma_i \cup \sigma_i'$ in $L_i'$. This step requires minimization of a model of $\sigma_i \cup \sigma_i'$ in $L_i'$, where the only predicates that are allowed to vary are those in $L_i' \setminus L_i$, and is possible for every stratified program $p$. Finally, the hierarchically-minimal entailment $p \vdash_{H\text{min}} q$ is defined as the truthfulness of $q$ in all hierarchically-minimal models of $p$.

Computing a hierarchically-minimal model of a stratified logic program $p$ is, generally, much more complex than computing just a minimal model of $p$, and makes the direct verification of $q$ in all hierarchically-minimal models of $p$ impractical. In Section 5 we present an alternative decision algorithm for $p \vdash_{H\text{min}} q$ based on the concept of the weakest precondition which, in a way characteristic to abductive reasoning, allows for backward evaluation of $q$ in the indefinite model of $p$ without actually computing models for relevant strata of $p$. It turns out that our approach, by appropriate generalization of the concept of stratification, eliminates the need of clauses with negation in the body.

Since this paper describes a research in progress, the results and ideas presented here are preliminary and have no pretenses to completeness. Moreover, most of the contents of Sections 2 and 3 is known from elsewhere (albeit in a different notation) and has been included for comparison. New ideas and results are offered in Sections 4 and 5.

### 2 Notation, etc.

We confine ourselves to a first-order language $L$ without equality and variables (and, therefore, without quantifiers), with usual Boolean connectives $\lor, \land, \neg$ (we treat all other connectives as appropriate abbreviations), finitely many predicate symbols $A, B, ..., M$. We denote formulas of $L$ by $p, q, r, ...$. We call atomic formulas of $L$ propositional variables (or simply variables) and place their arguments in subscripts rather than in parentheses (e.g., we write $B_{a, b, c}$ instead of $B(3,7)$). $L$ has only finitely many distinct propositional variables; we denote their number by $N$. If $\varphi$ is a set of formulas then $\land \varphi$ stands for $\land_{\varphi \in \varphi} p$, with $\land 0$ interpreted as true, and dually for $\lor \varphi$, with $\lor 0$ interpreted as false. $\neg \varphi$ means $\{\neg \varphi \mid \varphi \in \varphi\}$. We extend the above convention over sets of sets of formulas in an obvious way; e.g. $\land \{\varphi_1, ..., \varphi_n\}$ means $\{\land \varphi_1, ..., \land \varphi_n\}$.

A **clause carrier** is a set of literals with every propositional variable occurring at most once. A program carrier is a set of clause carriers. We use small Greek letters to denote clause carriers, and capital Greek letters for program carriers. Both program and clause carriers are finite since $L$ has only finitely many distinct variables. A clause is a formula of the form $\lor \varphi$, where $\varphi$ is a clause carrier. A program (for simplicity we use this term instead of indefinite deductive data base) is a formula of the form $\lor \Phi$, where $\Phi$ is a program carrier, that is, a program is a finite set of clauses. Since the meaning of $\lor \Phi$ is the same as $\land \lor \Phi$, we will use $\lor \Phi$ and $\land \lor \Phi$ interchangeably. If $X$ is a carrier, or a program or a clause, $|X|$ denotes the size of $X$ measured as the number of occurrences of propositional variables in $X$, while $\text{card}(X)$ denotes the number of elements of $X$.

A **term** is a formula of the form $\land \varphi$, and is called a minterm if $\varphi$ contains occurrences of all propositional variables of $L$. A conjunctive form formula (abbr: $CF$) is a formula of the form $\land \land \Phi$. Similarly, a disjunctive form formula (abbr: $DF$) is a formula of the form $\lor \land \Phi$. If all elements of $\land \Phi$ are minterms then $\land \land \Phi$ is called a disjunctive normal form formula (abbr: $DNF$). A set $Y$ of subsets of $X$ is called an anti-chain in $X$ if no two distinct elements of $Y$ are subsets one of another. If $\Phi$ is an anti-chain then $\land \lor \Phi$ is called a reduced conjunctive form formula (abbr: $RCF$), and $\lor \land \Phi$ a reduced disjunctive form formula (abbr: $RDF$).

Adjective positive refers to formulas and carriers without occurrences of negation, negative refers to those equivalent to negated positive ones. For instance, $A \lor B$ is a positive formula, while $\neg (\neg A \land \neg B)$ and $A \lor \neg B$ are not ($A \lor \neg B$ is an abbreviation for $\neg A \lor B$). Therefore, in the sense of this convention, positive clauses (or programs) are elsewhere called bodiless clauses (or programs). Bodiless definite programs constitute the easy case in traditional logic programming; they coincide with their minimal Herbrand mod-
els. Indefinite bodiless programs are much less trivial, and will serve as indefinite models. If \( \varphi \) is a clause carrier then \( \varphi^{Pos} \) denotes the set of all positive literals of \( \varphi \). If \( \Phi \) is a program carrier then \( \Phi^{Pos} = \{ \varphi \in \Phi | \varphi \text{ is positive} \} \) and \( \Phi^{Neg} = \{ \varphi^{Pos} | \varphi \in \Phi \} \). For instance, \( \{(A, \neg B), \{C\}\}^{Pos} = \{\{C\}\} \), while \( \{(A, \neg B), \{C\}\}^{Neg} = \{\{A\}, \{C\}\} \). We use similar convention for \( \Phi^{Neg} \) and \( \Phi^{Neg} \).

Minterms \( \land \varphi \) may be interpreted as propositional models of formulas of \( L \). For instance, minterm \( \land \varphi \) is a model of clause \( \forall \psi \) if \( \varphi \cap \psi \) is non-empty, which, of course, is equivalent to \( \land \varphi \models \forall \psi \). Therefore, DNF-formulas \( \land \Phi \) constitute sets of propositional models of formulas of \( L \). Because every element of a DNF-formula \( \land \Phi \) is a minterm, \( \Phi \) is unambiguously determined by its positive fragment \( \Phi^{Pos} \); therefore we will think of \( \Phi^{Pos} \) as of the set of models. Perhaps abusing the notation, we use \( \text{CWA} \) to denote the function which reconstructs \( \Phi \) from \( \Phi^{Pos} \). For instance, if \( L \) contains only two variables \( A \) and \( B \), \( \text{CWA}(\{A\}, 0, \{A, B\}) = \{\{A, \neg B\}, \{\neg A, \neg B\}, \{A, B\}\} \). We denote by \( \text{Mod}(p) \) the set of all models of \( p \). If \( \Phi \) is the set of models of \( p \) (that is, \( \Phi \) is positive, \( \forall \land \text{CWA}(A, B) \) is a DNF-formula, and \( \land \land \text{CWA}(\Phi) \equiv p \) is a propositional tautology) then the set \( \text{min} \{\Phi\} \) of all \( \land \)-minimal elements of \( \Phi \) is the set of minimal models of \( p \), which we also denote by \( \text{Mod}_{min}(p) \). \( \text{Mod}_{min}(p) \) is a positive anti-chain, and for every positive anti-chain \( \Psi \), there is a formula \( p \) of \( L \) (e.g., \( \land \land \Psi \)) with \( \Psi = \text{Mod}_{min}(p) \). Therefore sets of minimal models may be identified with positive RDF-formulas. It follows from (1) that a positive anti-chain \( \Psi \) is the set of minimal models of formula \( p \) if \( \land \land \Psi \) is logically equivalent to the conjunction of all positive clauses which are logical consequences of \( p \) (there are only finitely many of them).

Relation of minimal entailment \( \models p \) is defined by: \( p \models q \) if and only if \( q \in \text{Mod}_{min}(p) \). It is true for every minimal model of \( p \) if \( \models q \). Question “Does \( p \models q \)” makes a formal articulation of query “\( q \)” to program \( p \). Of course, \( p \models \land \land \varphi \) if \( q \in \varphi \land \land \varphi \). Therefore rather than allowing arbitrary programs as queries, we will restrict ourselves to clauses. By the above observation, this will cover all reasonable forms of queries to deductive data bases, at least in the propositional case.

It follows that for every DNF-formula \( \land \land \Phi \), the following are equivalent: \( \land \land \Phi \models q \) for every \( q \in \Phi \land \land \varphi \). There are only finitely many of them. Therefore for deductive data bases in disjunctive form, deciding \( \models p \) is straightforward and can be carried out in polynomial time with respect to the size of a data base and its query.

Sets of positive RCF formulas \( \land \lor \land \Phi \) define indefinite models of formulas of \( L \): \( \forall \Phi \) is an indefinite model of formula \( p \) (notation: \( \text{Ind}(p) \)) if \( \land \lor \land \Phi \) is logically equivalent to the conjunction of all positive logical consequences of \( p \) (and therefore, to \( \land \lor \text{Mod}_{min}(p) \)). It follows from (1) that for any formulas \( p \) and \( q \),

\[
p \models q \iff \text{Ind}(p) \models q.
\]

3 Minimal Modeling

In this section we discuss some tractable cases of deciding \( p \models q \) by direct verification of \( q \) in all minimal models of \( p \). We call this verification process a minimal modeling.

The set \( \text{Mod}(p) \) of models of program \( p \) is a positive carrier \( \Phi \), such that the DNF-formula \( \land \land \text{CWA}(\Phi) \) is logically equivalent to \( p \). The set \( \text{Mod}_{min}(p) \) is the \( \square \)-dense anti-chain \( \Psi \) in \( \Phi \). Given a positive carrier \( \Phi \), verifying if all elements of \( \land \land \Phi \) are models of a clause \( \forall \psi \) is easy because of simplicity of \( \text{CWA}(\land \land \Phi) \): for any \( \varphi \in \Phi \), \( \varphi \) is a model of \( \forall \psi \) if \( \land \land (\varphi^{Pos} \land \varphi) \) is non-empty. (A straightforward generalization of this scheme can be used for equally easy verification of a more general case of \( \psi \) being a set of any clauses, not just a set of literals; recall that the problem \( \forall \land \text{CWA}(\land \land \Phi) \models q \) is in \( P \) for arbitrary positive carrier \( \Phi \) and formula \( q \).) Obviously, \( \text{Mod}_{min}(p) \subseteq \text{Mod}(p) \), which implies, in the case when \( p \models q \) holds, that the direct verification of \( p \models q \) is never more complex than the direct verification of \( p \models q \). If \( p \models q \) is not the case then in some cases deciding \( p \models q \) may be more costly than the direct verification of \( p \models q \), since one can (incidentally) hit a non-minimal model of \( p \) which is not a model of \( q \) at the very beginning of the verification process. However, the ratio of the average size of \( \text{Mod}(p) \) \( \text{Mod}_{min}(p) \) to the average size of \( \text{Mod}_{min}(p) \) is approx. \( \sqrt{2N-1} \). Therefore, hunting for an element of \( \text{Mod}(p) \) \( \text{Mod}_{min}(p) \) which is not a model of \( q \) is very likely to be more costly than checking if \( q \) is true in all elements of \( \text{Mod}_{min}(p) \).

This observation gives rise to a hope that \( \models p \) is more often feasibly verifiable than it is. For instance, a program \( \varphi = \{ A_1 \lor ... \lor A_n, B_1 \lor ... \lor B_m \} \), where \( A_i \)'s and \( B_j \)'s are distinct variables, has only \( n^2 \) minimal models, but it has \( (2^n - 1)^2 \) models. Thus the direct verification of \( \land \land \varphi \) is tractable in this case, while direct verification of \( \land \land \varphi \) in the worst case, is not. And, of course, any definite program has only one min-
imal model (the least fixed-point of the operator \( T_\varphi \)), which makes the direct verification of \( \land \varphi \vdash_{\text{min}} q \) even easier.

The process of verification of minimal entailment \( \vdash_{\text{min}} \) by minimal modeling consists of two phases. First, given program \( \forall \Phi \), the set \( \text{Mod}_{\text{min}}(\forall \Phi) \) has to be constructed. Second, given a query \( \forall \psi \), the non-emptyness of \( (\psi^{\text{pos}} \cap \varphi) \cup (\psi^{\text{neg}} \setminus \varphi) \) for every minimal model \( \varphi \in \text{Mod}_{\text{min}}(\forall \Phi) \) must be verified. The latter process can be done in \( O(|\text{Mod}_{\text{min}}(\forall \Phi)| \times |\forall \psi|) \) steps in the worst case. The former may be more costly; however the time spent on its execution will be spread over presumably a large number of queries to the same program \( \forall \Phi \). Below, we discuss two cases when this process leads to a feasible computation.

**Case 1.** If reducing of \( \forall \Phi \) to a DNF-formula \( \forall \land \text{Mod}(\forall \Phi) \) can be done quickly (in polynomial time), then \( \text{Mod}_{\text{min}}(\forall \Phi) \) may also be built in a polynomial time; having \( \text{Mod}(\forall \Phi) \) one can purge all the non-minimal elements of \( \text{Mod}(\forall \varphi) \), which obviously can be done in \( O(|\text{Mod}(\forall \Phi)|^2 \times N^2) \), (where \( N \) is the number of variables in \( L \)).

**Case 2.** If there are only few (polynomially many), say \( K \), structures for \( L \) which do not include minimal models of \( \forall \Phi \) then \( \text{Mod}_{\text{min}}(\forall \Phi) \) may be built in polynomial time, according to the following algorithm. Think of relation of inclusion between sets of variables of \( L \) as of a digraph \( G \) with the empty set on the top and the set of all variables on the bottom, with edges going top down, connecting sets with their least proper supersets. A straightforward modification of breath-first search traversal of \( G \) with concurrent checking if the current vertex \( v \) is a model of \( \forall \Phi \) (in \( O(N^2) \) worst-case time) and cutting of its children in the case \( v \) is a model of \( \forall \Phi \), will take \( K + |\text{Mod}_{\text{min}}(\forall \Phi)| \) steps. Because \( |\text{Mod}_{\text{min}}(\forall \Phi)| \leq 2K \), the entire process will stop in \( O(N^2 \times K) \) worst-case time, which of course is polynomial.

Minimal modeling has several obvious drawbacks. First of all, the biggest cardinality of \( \text{Mod}_{\text{min}}(p) \) is \( \left( \begin{array}{c} N \\ \left\lfloor \frac{N}{2} \right\rfloor \end{array} \right) \) (cardinality of the longest anti-chain in a power set of an \( N \)-element set), or approximately, \( \sqrt{2} \frac{N}{\pi} \times 2^{\frac{N}{\sqrt{\pi}}} \), which makes its computation infeasible except for small values of \( N \). For instance, let us consider a well known (cf. [CR79, Hak85, BT88]) case for resolution proofs, the pigeonhole principle \( (\text{PHP}) \). Let \( \text{PHP}^- \) be

\[
\{\{A_{1,1}, \ldots, A_{1,n}\}, \ldots, \{A_{n+1,1}, \ldots, A_{n+1,n}\}\}
\]

and \( \text{PHP}^+ \) be

\[
\{\{A_{1,1}, A_{2,1}\}, \{A_{1,1}, A_{1,1}\}, \ldots, \{A_{n,1}, A_{n+1,1}\}, \ldots, \{A_{n,1}, A_{2,n}\}, \{A_{1,n}, A_{3,n}\}, \ldots, \{A_{n,n}, A_{n+1,n}\}\}
\]

where all \( A_{i,j} \)'s are distinct variables with the intentional interpretation of "\( i \)-th pigeon sits in the \( j \)-th hole." Obviously, \( \land \text{PHP}^- \land \land \text{PHP}^+ \) holds: it states that if there are \( n + 1 \) pigeons occupying \( n \) pigeonholes then at least one pigeonhole must accommodate at least two pigeons. Because \( \text{PHP}^+ \) is a positive carrier, \( \land \text{PHP}^- \land \land \text{PHP}^+ \) and \( \land \text{PHP}^- \land \land \text{PHP}^+ \) are equivalent. Hence \( \land \land \text{PHP}^- \land \land \text{PHP}^+ \) holds as well. \( \land \land \text{PHP}^- \) has \( N = n \times (n+1) \) variables and roughly \( 2^N \) models, which makes a direct verification of \( \land \land \text{PHP}^- \land \land \text{PHP}^+ \) (as well as its proof by resolution) intractable. Number of minimal models of \( \land \land \text{PHP}^- \) is \( n^{n+1} \). Although a considerable improvement over \( \land \) (with ratio approximately \( \left(\frac{2}{\sqrt{\pi}}\right)^N \)), this size of \( \text{Mod}_{\text{min}}(\land \land \text{PHP}^-) \) leaves direct verifiability of \( \land \land \text{PHP}^- \land \land \text{PHP}^+ \) out of the question.

Second, even relatively easy queries will involve their verification in all minimal models of the program in question. For instance, a program \( \forall \Phi = \{A_1 \land \ldots \land A_n, B_1 \land \ldots \land B_n\} \) has \( n^2 \) minimal models, therefore direct verification of straightforward \( \forall \land \text{Ind}_{\text{min}} \land \land \Phi \) will involve \( \Omega(n^3) \) steps (plus the time to compute \( \text{Mod}_{\text{min}}(\Phi) \)).

Third, adding new clauses to a program \( \varphi \) may considerably change \( \text{Mod}_{\text{min}}(\varphi) \) (in fact, \( \text{Mod}_{\text{min}}(\varphi) \) and \( \text{Mod}_{\text{min}}(\varphi') \), where \( \varphi' \subseteq \varphi ' \), may be disjoint). In this case, the costly process of computation of \( \text{Mod}_{\text{min}}(\varphi) \) must be repeated.

### 4 Indefinite Modeling

As we have seen, converting a program \( \forall \Phi \) into its positive RDF-formula \( \forall \land \text{Mod}_{\text{min}}(\forall \Phi) \) may be very costly, as the size of \( \text{Mod}_{\text{min}}(\forall \Phi) \) can be exponential in the size of \( \forall \Phi \). It may also happen that converting a program \( \forall \Phi \) into an equivalent DNF-formula is an exponentially lengthy process, even if \( \text{Mod}_{\text{min}}(\forall \Phi) \) is only polynomial in \( |\forall \Phi| \). However, because a program is a CF-formula itself, converting it into a positive RCF-formula \( \land \land \text{Ind}(\forall \Phi) \) should be much easier, at least in some cases. For instance, if \( \forall \Phi \) is a positive program then \( \text{Ind}(\forall \Phi) \) is a result of purging from \( \forall \Phi \) all the clauses which are subsumed by other clauses of \( \forall \Phi \), which can be done in \( O(|\forall \Phi|^2 \times N^2) \) time. Existence of cases like this gives rise to use of indefinite modeling instead of minimal modeling. In this section we present and analyze algorithms for converting a
program $\forall \Phi$ into its positive RDF-formula Ind($\forall \Phi$), and for deciding $\text{Ind}(\forall \Phi) \vdash_{\text{min}} q$, Both algorithms are based on the following parallel positive variant of resolution:

$$q \lor \neg R_1 \lor \ldots \lor \neg R_n \mid p_1 \lor R_1 \mid \ldots \mid p_n \lor R_n,$$

where $n > 0$, $R_i$’s are propositional variables, and $q$ and $p_i$’s are distinct positive clauses with no occurrences of variables $R_1, \ldots, R_n$.

Both algorithms use the operator of direct positive consequence $C$, which assigns to a set of clauses $\forall \Phi$ the set $C(\forall \Phi)$ of all positive clauses which can be derived from $\forall \Phi$ by subsumption and a single application of an instance of (3). Given a program $\forall \Phi$, Algorithm 4.1 computes Ind($\forall \Phi$).

**Algorithm 4.1**

$$K := \text{min}(\forall \Phi); I := K^{\text{Pos}}; J := K \setminus I;$$

while $C(I \cup J)$ is not subsumed by $I$ do

$$I := \text{min}(I \cup C(I \cup J));$$

{At this point, all positive consequences }\{of $\forall \Phi$ are subsumed by $I$ }\}

{Therefore, $I = \text{Ind}(\forall \Phi)$ }\}

return($I$);

Here, $\text{min}(X)$ is the set of all clauses of $X$ which are not subsumed by other clauses of $X$. The following lemma guarantees partial correctness of Algorithm 4.1 (and, in fact, the completeness of the parallel positive variant of resolution with respect to positive consequences).

**Lemma 4.2** Let $\Phi$ be a program carrier. If there is a positive clause $q$ which is a logical consequence of $\forall \Phi$ but is not subsumed by any of the clauses of $\forall \Phi^{\text{Pos}}$, then $C(\forall \Phi)$ is not subsumed by $\forall \Phi^{\text{Pos}}$.

**Proof.** Let $\Psi = \Phi \setminus \forall \Phi^{\text{Pos}}$. Assume that $C(\forall \Phi)$ is subsumed by $\forall \Phi^{\text{Pos}}$, that is, for every $\psi \in \Psi$, $C(\forall \Phi^{\text{Pos}} \cup \{\psi\})$ is subsumed by $\forall \Phi^{\text{Pos}}$. Hence, $\text{Mod}(\forall \Phi^{\text{Pos}}) = \text{Mod}(\forall \Phi^{\text{Pos}} \cup \{\psi\} \cup C(\forall \Phi^{\text{Pos}} \cup \{\psi\}))$, that is, $\text{Mod}_{\text{min}}(\forall \Phi^{\text{Pos}}) = \text{Mod}_{\text{min}}(\forall \Phi^{\text{Pos}} \cup \{\psi\} \cup C(\forall \Phi^{\text{Pos}} \cup \{\psi\}))$. On the other hand, straightforward verification yields $\text{Mod}_{\text{min}}(\forall \Phi^{\text{Pos}} \cup \{\psi\} \cup C(\forall \Phi^{\text{Pos}} \cup \{\psi\}))$. From this we conclude that $\text{Mod}_{\text{min}}(\forall \Phi^{\text{Pos}}) = \text{Mod}_{\text{min}}(\forall \Phi^{\text{Pos}} \cup \{\psi\} \cup C(\forall \Phi^{\text{Pos}} \cup \{\psi\}))$, that is, $\forall \Phi^{\text{Pos}} \vdash_{\text{min}} \forall \Phi$. Therefore, by thm. 4.3 in [Suc89], $\forall \Phi$ is consistent, that is, by definition of $\text{cwa}_{\text{S}}$ (def. 3.1 in [Suc89]), all positive consequences of $\forall \Phi$ are provable from $\forall \Phi^{\text{Pos}}$.

To evaluate the complexity of Algorithm 4.1, let us consider its following variant which calculates an “unpurged” version $\text{Ind}^*(\forall \Phi)$ of $\text{Ind}(\forall \Phi)$.

$$K := \forall \Phi; I := K^{\text{Pos}}; J := K \setminus I;$$

while $C(I \cup J)$ is not subsumed by $I$ do

$$I := I \cup C(I \cup J);$$

return($I$);

We have: $\text{Ind}(\forall \Phi) = \text{min}(\text{Ind}^*(\forall \Phi))$. Obviously, the number of iterations of the while loop is the same for both algorithms. Because this number cannot be greater than the number of elements in $\text{Ind}^*(\Phi)$ (which is finite; actually, it can be shown that the number of iterations is not greater than $|\Phi|$), Algorithm 4.1 halts, and therefore, is totally correct, that is, it returns the indefinite model $\text{Ind}(\forall \Phi)$ of $\forall \Phi$.

Purging out non-minimal (with respect to subsumption) clauses form $I \cup C(I \cup J)$ can be done as a part of checking whether $I$ does not subsume $C(I \cup J)$, which takes $O(|I| \times |C(I \cup J)|)$ time in the worst case, that is, $O(|\text{Ind}^*(\forall \Phi)| \times |C(\forall \Phi \cup \text{Ind}^*(\forall \Phi))|)$. Therefore, the entire algorithm executes in $O(|\forall \Phi| \times |\text{Ind}^*(\forall \Phi)| \times |C(\forall \Phi \cup \text{Ind}^*(\forall \Phi))|)$ worst-case time. If the size of $C(\forall \Phi \cup \text{Ind}^*(\forall \Phi))$ is polynomial in the size of $\Phi$ (say, $O(|\Phi|^{c+1})$) then the worst-case running time of Algorithm 4.1 is polynomial in $|\Phi|$ too ($O(|\Phi|^{2c+1})$). There are two kinds of hard cases for Algorithm 4.1: when $|\text{Ind}(\Phi)|$ is large (e.g., exponential) in $|\Phi|$, and when $|\text{Ind}(\Phi)|$ is small (e.g., polynomial) in $|\Phi|$ but $|\text{Ind}^*(\forall \Phi)|$ is large in $|\Phi|$. For instance, $|\text{Ind}(\Phi_1 \lor \ldots \lor A_n, B_1 \lor \neg A_1, \ldots, B_n \lor \neg A_n)| = 2^n$, which illustrates the first case. Since Algorithm 4.1 is based on resolution and $|\text{Ind}(\forall \Phi^{PHP^+}) \lor \forall \Phi^{\neg \neg \neg PHP^+}| = 0$ (because $\forall \Phi^{PHP^+} \lor \forall \Phi^{\neg \neg \neg PHP^+}$ is inconsistent), $\text{PHP}$ necessarily constitutes the second kind of a hard case (otherwise Algorithm 4.1 would yield a resolution based proof of $\text{PHP}$ with only polynomially many steps, which is impossible; cf. [CR79]).

Now, we turn to the decision algorithm for $\vdash_{\text{min}}$.

**Theorem 4.3** For every positive program $\forall \Phi$ and every clause $q$, $\text{Ind}(\forall \Phi) \vdash_{\text{min}} q$ iff $\text{Ind}^*(\forall \Phi)$ subsumes $C(\text{Ind}(\forall \Phi) \cup \{q\})$.

**Proof.** We inter from Lemma 4.2 that $C(\text{Ind}(\forall \Phi) \cup \{q\})$ subsumes all positive clauses provable from $\forall \Phi \cup \{q\}$. By def 3.1 and thm. 4.3 in [Suc89], $\text{Ind}(\forall \Phi) \vdash_{\text{min}} q$ iff all positive clauses provable from $\text{Ind}(\forall \Phi) \cup \{q\}$ are subsumed by $\text{Ind}(\forall \Phi)$. From this we conclude the thesis. \qed

Having calculated $\text{Ind}(\forall \Phi)$, the problem of $\forall \Phi \vdash_{\text{min}} q$
reduces by (2) to \( Ind(\forall \Phi) \vdash_{\text{min}} q \), and by Theorem 4.3 to the following question:

Is \( C(Ind(\forall \Phi) \cup q) \) subsumed by \( Ind(\forall \Phi) \)? (4)

which may be answered in \( O(|Ind(\forall \Phi)| \times |C(Ind(\forall \Phi) \cup \{q\})|) \) worst-case running time.

We conclude this section with examples of easy cases for indefinite modeling. For instance, allowing for at most one negation per clause makes \( Ind(\forall \Phi) \) easy to calculate. Below is another example which allows for fast computation of \( Ind(\forall \Phi) \).

**Theorem 4.4** If the length of every positive clause of \( \forall \Phi \) is in \( O(1) \), and every literal occurring negatively in \( \Phi \) has the total number of occurrences in \( \Phi \) at most 2 then Algorithm 4.1 runs in polynomial worst-case time.

**Proof** follows from the fact that there may be at most \( 2^\lambda \times N \) applications of a positive instance of resolution during the execution of Algorithm 4.1, where \( \lambda \) is the maximum length of a positive clause in \( \forall \Phi \). \( \square \)

If \( \text{card}(Ind(\forall \Phi)) \) is in \( O(1) \) then simple application of distributive law yields \( \text{Mod}_{\text{min}}(\forall \Phi) \) and decision \( Ind(\forall \Phi) \) \( \vdash_{\text{min}} q \) in polynomial time. Analysis of the one-step parallel positive resolution yields a number of characterizations of other easy cases for \( \forall Ind(\forall \Phi) \) \( \vdash_{\text{min}} q \). For instance, for each variable \( R \), \( Ind(\forall \Phi) \) \( \vdash_{\text{min}} \neg R \) iff \( R \) does not occur in \( \Phi \). Here is another one.

**Theorem 4.5** If the number of negative literals of \( q \) having more than 1 positive occurrence in \( Ind(\forall \Phi) \) is \( O(1) \) then deciding (4) can be performed in \( O(|Ind(\forall \Phi)|^2 \times |q|) \) worst-case time.

**Proof.** Generating \( C(Ind(\forall \Phi) \cup \{q\}) \) takes \( O(|Ind(\forall \Phi)| \times |q|) \) in the worst-case (all possible applications of parallel positive instances of resolution involving only one non-positive formula \( q \)), and then checking subsumption takes \( O(|Ind(\forall \Phi)|^2 \times |q|) \). \( \square \)

Theorem 4.5 allows for speeding up Algorithm 4.1 for programs \( \Phi \) with \( O(1) \) occurrences of negated literals (which seems to be a typical case for deductive data bases).

**Algorithm 4.6**

\[
I := \min(\forall \Phi^{\text{Pos}}) ;
\]
\[
\text{for } \pi \in \text{perm}(\min(\Phi \setminus \Phi^{\text{Pos}})) \text{ do}
\]
\[
\quad \text{for } q \in \pi \text{ do}
\]
\[
\quad I := \min(I \cup C(I \cup q));
\]
\[
\text{return}(I);
\]

where \( \text{perm}(X) \) is the set of all enumerations of \( X \).

**Theorem 4.7** If \( \Phi \) has \( O(1) \) occurrences of negated literals then Algorithm 4.6 computes \( Ind(\Phi) \) in \( O(|Ind(\Phi)|^2) \) worst-case running time.

**Proof.** Running time is evaluated by application of Theorem 4.5, letting \( |q| \in O(1) \). Partial correctness follows from analysis of genealogical graph of any positive formula in \( Ind(\Phi) \) computed by Algorithm 4.1: it follows that positive clauses \( p \) which are matched in (3) with their non-negative ancestors produce children which are subsumed either by ancestors of \( p \) or by siblings of \( p \). \( \square \)

Obviously, if \( \Phi \) contains no negations and \( q \) is a positive clause then \( \Phi \vdash_{\text{min}} q \) can be decided in \( O(|\Phi| \times |\psi|) \) worst-case time. Below is a summary of easy cases.

i. Computation of \( Ind(\forall \Phi) \) is polynomial in \( |\Phi| \) if:

(a) every clause of \( \forall \Phi \) is \( O(1) \) long, and each of its negative literal occurs at most twice (positively or negatively);
(b) \( \Phi \) has \( O(1) \) occurrences of negative literals.

ii. Decision of \( Ind(\forall \Phi) \) \( \vdash_{\text{min}} q \) is polynomial in \( |Ind(\forall \Phi) \cup \{q\}| \) if:

(a) \( \text{card}(Ind(\forall \Phi)) \) is in \( O(1) \);
(b) there are only \( O(1) \) negative literals of \( q \) with more than one positive occurrence in \( Ind(\forall \Phi) \);
(c) (a special case) there are only \( O(1) \) negative literals in \( q \);
(d) (a special case) \( |q| \) is in \( O(1) \).

iii. Decision of \( Ind(\forall \Phi) \) \( \vdash_{\text{min}} q \) is polynomial in \( |\forall \Phi \cup \{q\}| \) if:

(a) \( \forall \Phi \cup \{q\} \) has \( O(1) \) occurrences of negative literals;
(b) any other combination of subcases in (i) and (ii) holds.

5 Stratification

The method of indefinite modeling generalizes nicely over stratified programs. Quite surprisingly, the concept of indefinite model does not need any modifications in this case. This is a considerable advantage of indefinite modeling over minimal modeling which requires recomputation of the set of minimal models after stratification has been imposed or changed.

In this Section we analyse the case of programs with two strata, which covers the “inductive step” of the
definition of hierarchically-minimal model (end of Section 1). Since the “basis step” of that definition reduces to the minimal-model semantics, a routine induction over any tree hierarchy $L$ of first-order languages extends our approach over programs with arbitrary finite number of strata. We distinguish a non-empty sublanguage $L'$ of $L$, which partitions every program $\forall \Phi$ onto two strata: $\forall \Phi'$ and $\forall \Phi''$, where $\Phi' = \Phi \cap \mathcal{P}(L')$, and $\Phi'' = \Phi \setminus \mathcal{P}(L')$. $\forall \Phi'$ consists of all clauses of $\forall \Phi$ which do not contain occurrences of literals from outside $L'$; $\forall \Phi''$ contains the remaining clauses of $\forall \Phi$. Stratum $\forall \Phi'$ is given a higher priority during model minimization than stratum $\forall \Phi''$. Imposing this priority implies that certain minimal models of $\forall \Phi$ are no longer considered minimal for stratified $\forall \Phi$ (the converse is true, though).

Rather then using a relatively strong assumption of stratifiability (cf. [Apt88]) of $\forall \Phi$, we content ourselves with a weaker postulate, requiring that $\forall \Phi$ be a minimally conservative extension of $\forall \Phi'$, that is, that every minimal model (in the usual, non-stratified sense) of $\forall \Phi'$ has an expansion to a minimal model of entire $\forall \Phi$. We call every such an expansion a hierarchically-minimal model of $\forall \Phi$, and the entailment $\forall \Phi \vdash_{\text{min}} q$ induced by hierarchically minimal semantics a hierarchically-minimal entailment. Any program which satisfies the above requirement is called a layered program. One can verify with ease that hierarchically-minimal models are exactly the models of prioritized circumscription. It is also easy to check that every stratified program is layered, but not vice versa.

For each clause $q$ of $L'$, $\forall \Phi \vdash_{\text{min}} q$ and $\forall \Phi \vdash_{\text{min}} q$ are equivalent, as they are for negated literals $q$ of $L$. For other clauses from $L \setminus L'$, $\forall \Phi \vdash_{\text{min}} q$ and $\forall \Phi \vdash q$ need not be equivalent. In particular, there are layered programs $\forall \Phi$ and positive clauses $q$ of $L \setminus L'$ with $\forall \Phi \vdash_{\text{min}} q$ but not $\forall \Phi \vdash q$. For example, if $R$ is a variable of $L'$ and $S$ is a variable of $L \setminus L'$ then $\forall \Phi \vdash_{\text{min}} q$ in terms of $\text{Ind}(\Phi)$, $\vdash_{\text{min}}$, and $q$. To that end, we need the following concept.

**Definition 5.1** Let $\forall \Phi$ be a program and let $q$ be a formula of $L$. The weakest precondition $WP(\forall \Phi, q)$ for $q$ relative to $\Phi$ in $L'$ is a set of clauses of $L'$ which satisfies two conditions:

i. $\forall \Phi \cup WP(\forall \Phi, q) \vdash q$;

ii. for every sentence $r$ of $L'$, if $\forall \Phi \cup \{r\} \vdash q$ then $\{r\} \vdash WP(\forall \Phi, q)$.

Intuitively, the weakest precondition is equivalent to the weakest formula of the smaller language $L'$ which together with $\forall \Phi$ implies $q$, that is, $\forall WP(\forall \Phi, q)$ is logically equivalent to $\forall r \in L' \mid \forall \Phi \cup \{r\} \vdash q$. In particular, if $q$ is a formula of $L'$ then $WP(\forall \Phi, q) = \{q\}$. The following Lemma relates $\forall \Phi \vdash_{\text{min}} q$. For every positive layered program $\forall \Phi$ and every positive formula $q$ of $L$, 

$$\forall \Phi \vdash_{\text{min}} q \text{ iff } \forall \Phi' \vdash_{\text{min}} \forall WP(\forall \Phi, q).$$

**Proof.** $\forall \Phi \vdash_{\text{min}} q$ iff $[\text{by [SS90], thm. 5.5]} q \in \text{cwa}\_S(\forall \Phi \cup \text{cwa}\_S(\forall \Phi'))$ iff $[\text{by [Suc89], thm. 4.3]} \forall \Phi \cup \text{cwa}\_S(\forall \Phi') \vdash_{\text{min}} q$ iff $[\text{by definition 5.1 of WP}] \text{cwa}\_S(\forall \Phi') \vdash \forall WP(\forall \Phi, q)$ iff $[\text{by [Suc89], thm. 4.3}] \forall \Phi' \vdash_{\text{min}} \forall WP(\forall \Phi, q)$. □

The following lemma extends applicability of Lemma 5.2 over non-positive formulas $q$.

**Lemma 5.3** For every positive program $\forall \Phi$ and every clause $q$,

$$\forall \Phi \vdash_{\text{min}} q \text{ iff } \forall \Phi \vdash_{\text{min}} \forall \text{cwa}(\forall \Phi \cup \{q\}).$$

**Proof.** $\Rightarrow$ is obvious. $\Leftarrow$ $\forall \Phi \vdash_{\text{min}} \forall \text{cwa}(\forall \Phi \cup \{q\})$ implies $[\text{by [SS90], thm. 5.5}] \forall \text{cwa}(\forall \Phi \cup \{q\}) \in \text{cwa}\_S(\forall \Phi \cup \text{cwa}\_S(\forall \Phi'))$ implies $[\text{by thm. 4.3 of [Suc89]}] \forall \Phi \cup \text{cwa}\_S(\forall \Phi') \vdash_{\text{min}} \forall \text{cwa}(\forall \Phi \cup \{q\})$ implies $[\text{by positiveness of $\forall \Phi$ and thm 2.7 of [SS90]}] \forall \Phi \cup \text{cwa}\_S(\forall \Phi') \vdash \forall \text{cwa}(\forall \Phi \cup \{q\})$ implies $[\text{by positiveness of $\forall \Phi$ and Lemma 4.2}] \forall \Phi \cup \text{cwa}\_S(\forall \Phi')$ proves all positive consequences of $\forall \Phi \cup \{q\}$ implies $[\text{by [Suc89], thm. 4.3}]$ for every minimal model $M$ of $\forall \Phi \cup \text{cwa}\_S(\forall \Phi')$ there exists a minimal model $N$ of $\forall \Phi \cup \{q\}$ with $N \subseteq M$ implies $[\text{by [SS90], thm 5.4}]$ every model of $\text{cwa}\_S(\forall \Phi')$ is a minimal model of $\forall \Phi'$, therefore, $M \models \forall \Phi'$ is a minimal model of $\forall \Phi'$; also, $N \models \forall \Phi \cup \{q\}$ and $N \models \forall \Phi'$ implies $\forall \Phi \cup \{q\}$.

$$\forall \Phi \vdash_{\text{min}} q \text{ iff } \text{Ind}(\forall \Phi) \vdash_{\text{min}} q. \quad (5)$$

For every program $\forall \Phi$ and clause $q$,

$$\forall \Phi \vdash_{\text{min}} q \text{ iff } \forall \Phi \vdash_{\text{min}} \text{Ind}(\forall \Phi) \vdash_{\text{min}} q.$$
This property follows from the fact, that if \( p \) and \( r \) have the same minimal models then they also have the same hierarchically-minimal models. Hence the following theorem.

**Theorem 5.4** For every program \( \forall \Phi \) and clause \( q, \forall \Phi \vdash_{H_{\min}} q \) iff \( \text{Ind}(\forall \Phi)' \) subsumes \( C(\text{Ind}(\forall \Phi)) \cup W P(\text{Ind}(\forall \Phi), \forall \Phi) \cup \{ q \} \).

**Proof.** \( \forall \Phi \vdash_{H_{\min}} q \) iff by (5) \( \text{Ind}(\forall \Phi)' \vdash_{H_{\min}} q \) iff by Lemma 5.3 \( \text{Ind}(\forall \Phi) \vdash_{H_{\min}} \forall \Phi \cup \{ q \} \) iff by Lemma 5.2 \( \text{Ind}(\forall \Phi)' \vdash_{H_{\min}} \forall \Phi \cup \{ q \} \). By Theorem 4.3 \( \text{Ind}(\forall \Phi)' \) subsumes \( C(\text{Ind}(\forall \Phi)' \cup W P(\text{Ind}(\forall \Phi), \forall \Phi) \cup \{ q \}) \).

Given a procedure for computing \( WP \), Theorem 5.4 yields a straightforward decision algorithm for \( \vdash_{H_{\min}} \). Because \( WP(\forall \Phi, p \land q) = WP(\forall \Phi, p) \cup WP(\forall \Phi, q) \), the following theorem indicates an equally straightforward algorithm to compute \( WP \) for positive programs and positive clauses.

**Theorem 5.5** Let \( \forall \Phi \) be a set of positive clauses, let \( q \) be a positive clause in \( \forall \Phi \), and let \( s_1 \lor p_1, \ldots, s_n \lor p_n \) be all the clauses of \( \forall \Phi \) such that each \( s_i \) is in \( \forall \Phi \), each \( p_i \) is in \( \forall \Phi \), and each \( s_i \) subsumes \( q \). Then

\[
WP(\forall \Phi, q \lor r) \equiv \{ \lnot p_1 \lor \ldots \lor \lnot p_n \lor r \}.
\]

(If \( n = 0 \) then, by convention, \( \lnot p_1 \lor \ldots \lor \lnot p_n \lor r = r \).)

**Proof.** Let us first prove that \( \vdash \lnot p_1 \lor \ldots \lor \lnot p_n \) is the weakest precondition for \( q \). Obiously, \( \forall \Phi \cup \{ \lnot p_1 \lor \ldots \lor \lnot p_n \} \vdash q \lor r \). Therefore, it suffices to demonstrate that for every ground clause \( x \) in \( \forall \Phi \cup \{ \lnot p_1 \lor \ldots \lor \lnot p_n \} \), \( \forall \Phi \cup \{ x \} \vdash q \lor r \). Suppose to the contrary, that \( \forall \Phi \cup \{ x \} \not\vdash q \lor r \). That is, \( \forall \Phi \cup \{ x \} \cup \{ p_1, \ldots, p_n \} \) is consistent. Because \( \{ p_1, \ldots, p_n \} \) subsumes \( \{ s_1 \lor p_1, \ldots, s_n \lor p_n \} \), \( \forall \Phi \cup \{ x \} \cup \{ p_1, \ldots, p_n \} \) had the same models as \( \forall \Phi' = \forall \Phi \setminus \{ s_1 \lor p_1, \ldots, s_n \lor p_n \} \cup \{ p_1, \ldots, p_n \} \). In particular, \( \forall \Phi' \) is consistent. Let \( M \) be a model of \( \forall \Phi' \). Because \( \forall \Phi' \) is positive, \( M' = M \cup \{ A | A \text{ is a positive literal of } \forall \Phi' \} \) is also a model of \( \forall \Phi' \). By the definition of \( \forall \Phi' \), all occurrences of \( q \) in \( \forall \Phi' \) are in positive clauses involving literals of \( \forall \Phi' \) not in \( q \), and all these literals are satisfied by \( M' \). Therefore, \( M'' = M' \setminus \{ B | B \text{ is a literal of } q \} \) is a model of \( \forall \Phi' \). Hence \( \forall \Phi' \cup \{ \lnot q \} \) is consistent, and therefore, \( \forall \Phi \cup \{ x \} \cup \{ p_1, \ldots, p_n \} \cup \{ \lnot q \} \) is consistent. From this, we conclude that \( \forall \Phi \cup \{ x \} \cup \{ \lnot q \} \) is consistent - a contradiction. Thus \( \forall \Phi \cup \{ x \} \vdash \{ p_1 \lor \ldots \lor \lnot p_n \} \), and \( \lnot p_1 \lor \ldots \lor \lnot p_n \) is the weakest precondition for \( q \).

Now, let \( \forall \Phi \cup \{ x \} \vdash q \lor r \). We get \( \forall \Phi \cup \{ x \lor \lnot r \} \vdash q \), that is, \( \forall \Phi \cup \{ x \lor \lnot r \} \vdash \lnot p_1 \lor \ldots \lor \lnot p_n, \) so \( \forall \Phi \cup \{ x \} \vdash \lnot p_1 \lor \ldots \lor \lnot p_n \lor r \), that is, \( \forall \Phi \cup \{ x \lor \lnot r \} \lor p_1 \lor \ldots \lor \lnot p_n \lor r \), which is the worst case performance of these methods, since there are programs \( \forall \Phi \) for which both \( \text{Mod}_{\min}(\forall \Phi) \) and

It follows from Theorem 5.5 that if \( \{ C(\text{Ind}(\forall \Phi) \cup \{ q \}) \} \) is polynomial in \( \{ C(\varphi) \} \), then \( \lnot \text{Mod}_{\min}(\text{Ind}(\forall \Phi), \forall \Phi) \cup \{ q \} \) is polynomial in \( \forall \Phi \). If, moreover, the number of occurrences of negation in \( \text{Mod}_{\min}(\text{Ind}(\forall \Phi), \forall \Phi) \cup \{ q \} \) is \( O(1) \), then the question \( \forall \Phi \vdash_{H_{\min}} q \) may be solved in polynomial time (proofs are straightforward). More general evaluation of its complexity requires further study.

**Example 5.6** Let \( L = \{ A, B \} \), \( L = \{ A, B, C, D \} \), \( \forall \Phi = \{ A \lor B, A \lor C \lor D, B \lor C \lor D \} \), and \( q = \{ C \lor D \} \). We have: \( \text{Ind}(\forall \Phi) = \forall \Phi; \forall \Phi' = \{ A \lor B \}; \) \( \forall \Phi \) is a minimal conservative extension of \( \forall \Phi' \); \( C(\text{Ind}(\forall \Phi) \cup \{ q \}) = \{ C \lor D \} \); \( W P(\forall \Phi, C \lor D) = \{ \lnot A \lor \lnot B \}; \) \( C(\forall \Phi' \cup \{ \lnot A \lor \lnot B \}) = B \lor A \). \( \forall \Phi' \) subsumes \( C(\forall \Phi' \cup \{ B \lor A \}) \). Hence,

\( \forall \Phi \vdash_{H_{\min}} q \).

Complexity of modeling of definite stratified programs was analyzed in [AB91].

**6 Speed-ups**

The pigeonhole principle PHP, as it was demonstrated in [CR79], leaves no hope for a complete and fast resolution-based derivation algorithm. Consequently, the resolution-based algorithms we presented so far must perform at least exponentially badly in the worst case. In this section we briefly discuss to what extent the indefinite modeling may be faster than the minimal modeling and vice versa.

First, let us notice that for every anti-chain \( \Phi \) in the set of sets of variables of \( L \), \( \forall \Phi \) is its own indefinite model, and \( \forall \Phi \) is the set of minimal models of \( \forall \land \Phi \). Therefore, if there are programs for which the indefinite model is smaller than the set of minimal models, there must be equally many others for which the converse is true. This fact suggests that a parallel application of both methods while deciding \( \forall \Phi \vdash_{\min} q \) may result in a considerable speed-up in some cases, while the slow-down factor, if any, will not exceed 2. For instance, as we noted in Section 3, \( \text{Mod}_{\min}(\text{Ind}(\forall \Phi \lor \forall \Phi')) \) is exponential in \( \forall \land \Phi \lor \forall \Phi \lor \forall \Phi' \), while \( \text{Ind}(\text{Ind}(\forall \Phi \lor \forall \Phi')) \) = \( \forall \lor \forall \Phi \lor \forall \Phi' \). Also the converse is true: there are programs \( \forall \Phi \) with \( \text{Mod}_{\min}(\forall \Phi) \) polynomial in \( \forall \Phi \) but \( \text{Mod}_{\min}(\forall \Phi) \) exponential in \( \forall \Phi \). (Even if both are polynomial, any resolution-based conversion of one onto another may require exponential number of steps). This parallel strategy, however, cannot improve the worst-case performance of these methods, since there are programs \( \forall \Phi \) for which both \( \text{Mod}_{\min}(\forall \Phi) \) and
Elementary calculations show that the average length \( l \) of the total length \( \sum_{i=1}^{\gamma} d_i \) is at least \( \frac{\log_{2} \left( n + \frac{2}{\gamma} \right) - 2}{\log_{2}(N + \frac{2}{\gamma})} \). Hence

\[
R_{avg}^{D,C} = \frac{1}{\kappa} \sum_{i=1}^{\kappa} \frac{d_i}{c_i} \geq \frac{1}{\kappa} \sum_{i=1}^{\kappa} \frac{d_i}{\left\lceil \frac{N}{2} \right\rceil \times \gamma} = \frac{1}{\left\lceil \frac{N}{2} \right\rceil \times \gamma} \sum_{i=1}^{\kappa} \frac{d_i}{\kappa} \geq \frac{\gamma - 2}{N \times \log_{2}(N + 2) \times \gamma} \approx \frac{1}{N \times \log_{2}N},
\]

where \( c_i \)'s and \( d_i \)'s are the sizes of all indefinite models and sets of their minimal models. Symmetric argument shows that the average \( R_{avg}^{C,D} \) of the converse ratio has the same lower bound. Therefore, on average, the cost of minimal modeling and indefinite modeling are very close to each other. Their closer relationship, for instance, the average ratio \( \frac{d}{c} \) with averaging restricted to cases when the smaller of \( c_i \), \( d_i \) is polynomial in \( N \), requires further study.

7 Credit to Others

A version of the right-hand side of criterion (1) for indefinite deductive data bases was proposed (in an equivalent form) by Minker [Min82] under the name of Generalized Closed-World Assumption. Liu and Sunderraman [LS87] introduced indefinite tuples in a relational model of a data base. The credit for stratification is due to Apt, Walker and Blair [ABW88], and Van Gelder [Gel88]. The idea of deciding \( p \vdash_{min} q \) by direct verification is Minker’s and his collaborators.

8 Acknowledgements

I would like to thank Henrietta Okeke and Johnny Chan forTEXing my hieroglyphic manuscript, and Selase Williams for his support of my research activities.

References


