COMPUTING FACTS IN NON-HORN DEDUCTIVE SYSTEMS.

Eliezer L. Lozinskii

Institute of Mathematics and Computer Science
The Hebrew University, Jerusalem, Israel.

Abstract. Let S be a set of clauses (non-Horn as well as Horn ones), and q an atomic query. Consider the problem of deriving from S all ground unit clauses satisfying q, which we call *computing all facts* for q. It is shown how a *non-Horn* system S can be transformed into a set of *singleton-head-rules*, $SH(S)$, such that computing of all facts for a given query q in S is reduced to the query evaluation in a set of *Horn* clauses *relevant* to q which is a subset of $SH(S)$. The transformation is sound and complete.

1. Introduction.

Consider a deductive database DB as a set of *clauses* (either *Horn* or *non-Horn* ones) [7]. Let q be an atomic query regarding a relation P such that $q:\exists x(P(x))$ or $q:\exists x(\neg P(x))$, x is a vectors of terms. Consider ground atoms $P(a), P(b), P(c)$ (a, b, c are vectors of constants), and suppose that $P(a)$ is true and $P(b)$ is false in all models of DB, while $P(c)$ is true in some models but false in others. Then $P(a)$, $\neg P(b)$ must appear in an answer to q. As to $P(c)$, it is not determined definitely by DB, but we may assume its truth value according to one of well known approaches, like Closed World Assumption [15], Negation as Failure [5] (for Horn systems) or Generalized Closed World Assumption [13]. We say that $P(a)$ and $\neg P(b)$ are *facts* in DB, while regarding $P(c)$ we have got just a *belief*.

Any state of DB reflects our current knowledge (rather incomplete) about the real world. Suppose we assumed that $P(c)$ is false, but later on have discovered that $P(c)$ is actually true. In this case we must revise our belief about $P(c)$. This need for revising beliefs in the course of acquiring new information is responsible for *non-monotonicity* of databases [6,12,13,16,]. On the other hand, the set of facts is most stable and *monotonic* in the sense that no new data consistent with DB can disprove a fact, since all facts belong to all models of DB.

With this in mind, given a database DB and a unit clause C, we are interested in deriving from DB all facts satisfying C. Recently a number of efficient query evaluation methods have been developed for Horn databases [2,3,7-11,14,17-19,21], while the current research on non-Horn systems is less comprehensive. It is either concerned mainly with producing beliefs rather than facts [12,13,16,22], or applicable to particular systems, e.g., *stratified* databases [1,20].

In the sequel we present an efficient method of deriving facts from a general database (Horn as well as non-Horn one) by transforming it to a set of Horn clauses relevant to a given query, and then applying techniques developed for Horn-systems.

2. Clauses and sh-rules.

Consider a deductive system, S, and an atomic query $q: P(x_1,...,x_k)$, where for $i = 1,...,k$ x_i is either a variable or a constant. First, we would like to know what ground instances of P are facts in S such that $S \vdash P(a_1, \ldots, a_k)$ or $S \vdash \neg P(a_1, \ldots, a_k)$ ($a_1, \ldots, a_k$ are constants).

Definition 1. (a) Consider a clause $A: l_1 \vee l_2 \vee, \ldots, \vee l_k$ ($l_1,...,l_k$ are literals) that is equivalent to an implication $B: \neg l_1 \wedge, \ldots, \wedge \neg l_i \rightarrow l_{i+1} \vee, \ldots, \vee l_k$. We say that B presents the clause A in the form of a *rule* or an *axiom*, where $l_{i+1} \vee, \ldots, \vee l_k$ is the *head* of the axiom, and $\neg l_1 \wedge, \ldots, \wedge \neg l_i$ is its *body*. Where it is appropriate we omit connectives, and display clauses and axioms as sets of literals:

$A: \{ l_1, l_2, \ldots, l_k \}$,

$B: \{ \neg l_1, \ldots, \neg l_i \} \rightarrow \{ l_{i+1}, \ldots, l_k \}$,

This research is supported in part by Israel National Council for Research and Development, grant 2454-3-87.

$Head(B) = \{L_{i+1}, \ldots, L_k \mid L_j = l_j \text{ for } i+1 \le j \le k\}$,

$Body(B) = \{L_1, \ldots, L_i \mid L_j = \neg l_j \text{ for } 1 \le j \le i\}$.

(b) If the head of a rule, C, contains exactly one literal, then C is called a *singleton-head-rule* (*sh-rule*, for short).

(c) If a clause A contains k literals, then there are k different sh-rules *corresponding* to A:

$$A_i : \{\neg l_j \mid 1 \le j \le k \text{ and } j \ne i\} \rightarrow l_i, \text{ for } i=1,\ldots,k,$$

and a set $SH(A) = \{A_1, \ldots, A_k\}$ is a *full set* of sh-rules corresponding to A.

(d) Consider an sh-rule C containing in its body literals L_1, L_2 unifiable by a substitution θ_1, $L_1\theta_1 = L_2\theta_1$, while $Head(C)$ is not unifiable by θ_1 with any literal of C, which means that there is no literal $L \in Body(C)$ such that $Head(C)\theta_1 = \neg L\theta_1$. Then a factor $F_1 = C\theta_1$ is a *body-factor* of C. If there is a literal $L_3 \in Body(C)$ such that $Head(C)\theta_2 = \neg L_3\theta_2$ then $F_2 = C\theta_2$ is a *head-factor* of C such that

$Head(F_2) = Head(C)\theta_2$,

$Body(F_2) = (Body(C) - \{L_3\})\theta_2$.

(e) In the sequel we shall resolve two sh-rules only upon a literal that appears in the head of one of them and in the body of the other. Let C_1, C_2 be sh-rules, and suppose $L_i\theta = L_j\theta$, where

$$L_i \in Head(C_1), \quad L_j \in Body(C_2).$$

Then $Res(C_1, C_2)$ is a *resolvent* of C_1, C_2 (upon L_i, L_j) such that

$Head(Res(C_1, C_2)) = Head(C_2)\theta$,

$Body(Res(C_1, C_2)) = Body(C_1)\theta \cup (Body(C_2) - \{L_j\})\theta$.

Thus, $Res(C_1, C_2)$ denotes a resolvent of C_1, C_2 upon the head of C_1, preserving the head of C_2. ■

Example 1. Consider two clauses

$$A : \neg P(x,y) \vee Q(y,x),$$

$$B : \neg Q(t_1,t_2) \vee R(t_1,t_3) \vee R(t_3,t_2).$$

Then by Definition 1, sh-rules corresponding to A are $A_1 : \neg Q(y,x) \rightarrow \neg P(x,y)$, $A_2 : P(x,y) \rightarrow Q(y,x)$;

sh-rules corresponding to B are

$$B_1 : \neg R(t_1, t_3),\ \neg R(t_3, t_2) \rightarrow \neg Q(t_1, t_2),$$

$$B_2 : Q(t_1, t_2),\ \neg R(t_3, t_2) \rightarrow R(t_1, t_3),$$

$$B_3 : Q(t_1, t_2),\ \neg R(t_1, t_3) \rightarrow R(t_3, t_2).$$

Full sets of sh-rules are $SH(A) = \{A_1, A_2\}$, $SH(B) = \{B_1, B_2, B_3\}$. A body-factor of B_1 is $F_1 : \neg R(t_1, t_1) \rightarrow \neg Q(t_1, t_1)$. A head-factor of B_2 is $F_2 : Q(t_1, t_1) \rightarrow R(t_1, t_1)$.
A resolvent of A_2 and B_3 is $Res(A_2, B_3) : P(t_2, t_1),\ \neg R(t_1, t_3) \rightarrow R(t_3, t_2)$. ■

Lemma 1. Let $SH(DB)$ denote a full set of sh-rules corresponding to all clauses of a deductive database DB, $SH(DB) = \bigcup_{C \in DB} SH(C)$. Then

(a) C' is an instance of a clause of DB iff there are instances of rules of $SH(DB)$ that constitute a full set of sh-rules corresponding to C';

(b) F is a factor of a clause of DB iff there are factors of rules of $SH(DB)$ (head- or body-factors) that constitute a full set of sh-rules corresponding to F;

(c) Res is a resolvent of two clauses of DB iff there are resolvents of rules of $SH(DB)$ that constitute a full set of sh-rules corresponding to Res.

Proof. (a) $\Rightarrow$ Let C' be an instance of a clause $C \in DB$ such that $C = \{L_1, \ldots, L_k\}$, and $C' = C\theta = \{L_1\theta, \ldots, L_k\theta\}$ (θ is a substitution). $SH(DB)$ includes a full set of sh-rules corresponding to C:

$SH(C) = \{C_i \mid 1 \le i \le k \text{ and } Head(C_i) = L_i$

$\text{and } Body(C_i) = \{\neg L_j \mid L_j \in C \text{ and } j \ne i\}\}$.

Then a set of instances of rules of $SH(C)$,

$SH(C)\theta = \{C'_i \mid 1 \le i \le k \text{ and } Head(C'_i) = L_i\theta$

$\text{and } Body(C'_i) = \{\neg L_j\theta \mid L_j \in C \text{ and } j \ne i\}\}$,

is a full set of sh-rules corresponding to C', $SH(C') = SH(C)\theta$.

(a) $\Leftarrow$ Let T be a set of instances of rules of $SH(DB)$ such that T is a full set of sh-rules corresponding to a clause $B = \{L_1, \ldots, L_m\}$ (Fig. 1 illustrates the relationship among the clauses and rules mentioned in this proof):

$T = \{B_i \mid 1 \le i \le m \text{ and } Head(B_i) = L_i$

$\text{and } Body(B_i) = \{\neg L_j \mid L_j \in B \text{ and } j \ne i\}\}$.

$$\begin{tikzcd}
D \in SH(E) \arrow[d, shift right=2] \arrow[d, shift left=2, "\sigma"'] & E \arrow[l, "sh"'] \arrow[d, "\sigma"] \\
B_k \in T = SH(E') & E' = B \arrow[l, "sh"']
\end{tikzcd}$$

Fig. 1.

Let $B_k \in T$ be an instance of a rule $D \in SH(DB)$ such that $B_k = D\sigma$ (σ is a substitution). Denote

$Head(D) = M_k$,

$Body(D) = \{\neg M_j \mid 1 \le j \le m \text{ and } j \ne i\}$.

Then D is an sh-rule, E_k, corresponding to a clause $E \in DB$, $E = \{M_1, \ldots, M_m\}$. So, $SH(DB)$ must contain the full set of sh-rules corresponding to E:

$SH(E) = \{E_i \mid 1 \le i \le m$ and $Head(E_i) = M_i$

and $Body(E_i) = \{\neg M_j \mid 1 \le j \le m$ and $j \ne i\}\}$.

Since $B_k = D\sigma = E_k\sigma$, the set T is actually an instance of $SH(E)$:

$T = \{B_i \mid 1 \le i \le m$ and $Head(B_i) = M_i\sigma$

and $Body(B_i) = \{\neg M_j\sigma \mid 1 \le j \le m$ and $j \ne i\}\}$

$= SH(E)\sigma$.

Therefore, there is an instance of E, $E' = E\sigma = \{M_1\sigma, \ldots, M_m\sigma\}$, such that T is a full set of sh-rules corresponding to E'.

(b),(c) By an argument similar to that used in (a). For the sake of brevity some proofs are presented in a sketchy form. ■

Consider a sound and complete proof-procedure such as a deduction process of resolution [4] which we denote DPR. It consists of a sequence of steps such that each one derives either an instance or a factor of a clause, or a resolvent of two clauses of DB.

Lemma 1 implies the following

Theorem 1. A clause C is derivable from a deductive database DB iff a full set of sh-rules corresponding to C is derivable from $SH(DB)$.

Proof. $\Rightarrow$ Let a clause C be derivable from a set of clauses DB by a deduction process of resolution, DPR, in n steps. Denote by DB_i the state of DB after performing of i steps of DPR, and by $SH(DB_i)$ the full set of sh-rules corresponding to DB_i, so, $DB_0 = DB$. We show by induction on the number of steps of DPR that for every DB_i there is a set of sh-rules $SH_i(DB)$ derivable from $SH(DB)$ which is a full set of sh-rules corresponding to DB_i such that $SH_i(DB) = SH(DB_i)$.

Indeed, $SH_0(DB) = SH(DB) = SH(DB_0)$. Suppose that the claim holds for all $i \le k$, i.e., $SH_k(DB) = SH(DB_k)$. Let step $k+1$ of DPR derive a new clause D, so, $DB_{k+1} = DB_k \cup \{D\}$. Then by Lemma 1 a full set of sh-rules $SH(D)$ corresponding to D can be derived from $SH_k(DB)$, therefore $SH_{k+1}(DB) = SH_k(DB) \cup SH(D) = SH(DB_{k+1})$ is derivable from $SH(DB)$ and consists of a full set of sh-rules corresponding to DB_{k+1}. If a clause C is derivable from DB in n steps of DPR, i.e., $C \in DB_n$, then there is $SH_n(DB)$ derivable from $SH(DB)$ such that $SH_n(DB) = SH(DB_n)$, and hence $SH(C) \in SH_n(DB)$.

$\Leftarrow$ In the same way, by induction on the number of steps of a deduction process performed in $SH(DB)$, using the argument of Lemma 1. ■

A unit clause is at the same time a full set of its sh-rules. And DPR, being a sound and complete procedure for refutation, is so for deriving unit clauses. Hence, Theorem 1 yields the following

Corollary 1. A ground unit clause is a fact in DB iff it is a fact in $SH(DB)$. ■

3. Hornization of sh-rules.

Due to Theorem 1 and Corollary 1, the same set of facts can be produced by dealing with sh-rules instead of general clauses. Since the former is a special case of the latter, performing derivation on sh-rules may be simpler and more efficient than handling general clauses. This section shows how a further simplification of the derivation process can be achieved.

Let S be a set of sh-rules. Consider a rule $A \in S$ with a negated predicate in its head, $Head(A) = \neg P(\mathbf{x})$ ($\mathbf{x}$ is a vector of variables and constants), and a rule $B \in S$ containing $\neg P(\mathbf{y})$ in its body. Suppose that $P(\mathbf{x})\theta = P(\mathbf{y})\theta$, so A and B can be resolved upon the head of A producing $Res(A, B)$. Now, replace $\neg P$ by a new predicate symbol $\bar{P}$ getting new rules $\bar{A}, \bar{B}$, respectively. Still $\bar{A}, \bar{B}$ are resolvable upon the head of $\bar{A}$.

Definition 2. Let A be an sh-rule, and $\rho = \{\bar{P}_1/\neg P_1, \ldots, \bar{P}_k/\neg P_k\}$ stand for a substitution that renames negated predicate symbols, such that $P_i \ne P_j$ and $\bar{P}_i \ne \bar{P}_j$ for $i \ne j$. Then $A\rho$ is a *renaming* (or more specifically, ρ*-renaming*) of A. Since all predicate symbols appearing in ρ are distinct, we have $\rho\rho^{-1} = \varepsilon$, where ε is an empty substitution, and $\rho^{-1} = \{\neg P_i/\bar{P}_i \mid \bar{P}_i/\neg P_i \in \rho\}$. ■

Example 2. Consider sh-rules of Example 1. A renaming substitution for all negated predicates is $\rho = \{\bar{P}/\neg P, \bar{Q}/\neg Q, \bar{R}/\neg R\}$. Then the following is the ρ-renaming of the rules of $SH(A)$, $SH(B)$:

$\bar{A}_1 = A_1\rho : \bar{Q}(y, x) \to \bar{P}(x, y);$

$\bar{A}_2 = A_2\rho : P(x, y) \to Q(y, x);$

$\bar{B}_1 = B_1\rho : \bar{R}(t_1, t_3), \bar{R}(t_3, t_2) \to \bar{Q}(t_1, t_2);$

$\bar{B}_2 = B_2\rho : Q(t_1, t_2), \bar{R}(t_3, t_2) \to R(t_1, t_3);$

$\bar{B}_3 = B_3\rho : Q(t_1, t_2), \bar{R}(t_1, t_3) \to R(t_3, t_2).$ ■

Let ρ be a renaming substitution for *all* negated predicate symbols of S, $\rho = \{\bar{P}_i/\neg P_i \mid \neg P_i \in S\}$, and $\bar{S}$ denote the set of ρ-renamings of all sh-rules of S. Then, with respect to the predicate symbols appearing in $\bar{S}$ (the original as well as the new "barred" ones), all rules of $\bar{S}$ are Horn clauses, since there is no negated predicate in their heads or bodies. Because deduction by resolution on Horn clauses is much simpler and more efficient than that performed on general clauses or sh-rules with negation, we are interested in determining the conditions under which a derivation performed on ρ-renamed rules of $\bar{S}$ gives the same result as the one carried out on the original rules of S.

Lemma 2. Let S be a set of sh-rules, ρ — a renaming substitution for all negated predicate symbols of S, and $\bar{S}$ — the set of ρ-renamed rules of S. Then

(a) C' is an instance of a rule $C \in S$ iff there exists an instance D' of a rule $D \in \bar{S}$ such that D' is a ρ-renaming of C';

(b) $Res(A, B)$ is a resolvent of rules $A, B \in S$ iff there exists a resolvent $Res(C, D)$ of rules $C, D \in \bar{S}$ such that $Res(C, D)$ is a ρ-renaming of $Res(A, B)$;

(c) F is a body-factor of a rule $C \in S$ iff there exists a body-factor G of a rule $D \in \bar{S}$ such that G is a ρ-renaming of F;

(d) If H is a head-factor of a rule $C \in S$, then there may not exist a factor G of a rule of $\bar{S}$ such that G is a ρ-renaming of H.

Proof. (a) $\Rightarrow$ Let C' be an instance of a rule $C \in S$, $C' = C\theta$. Then $\bar{S}$ contains a ρ-renaming D of C, $D = C\rho$. Hence, an instance D' of D, $D' = D\theta = C\rho\theta$ is a ρ-renaming of C'; indeed, $C\rho\theta = C\theta\rho$ since θ and ρ are independent and distinct: the former is a substitution for variables, while the latter — for predicate symbols.

(a) $\Leftarrow$ Let D' be an instance of $D \in \bar{S}$, $D' = D\theta$, while D is a ρ-renaming of a rule $C \in S$, $D = C\rho$. Then there is an instance C' of C, $C' = C\theta$ such that D' is a ρ-renaming of C', since $D' = D\theta = C\rho\theta = C'\rho$.

(b),(c) By an argument similar to that used in (a).

(d) Let C be a rule in S such that $Head(C) = P(x)$, $\neg P(y) \in Body(C)$, and $P(x)\theta = P(y)\theta$. Then C has a head-factor H such that $Head(H) = P(x)\theta$ and $Body(H) = Body(C)\theta - \{\neg P(y)\theta\}$. Consider a ρ-renaming $\bar{C}$ of C, i.e., $\bar{C} = C\rho$ and $\bar{C} \in \bar{S}$. Since $Head(\bar{C}) = P(x)\rho = P(x)$, but $\neg P(y)\rho = \bar{P}(y)$, there is no literal in the body of $\bar{C}$ whose negation is unifyable with the head of $\bar{C}$. And there may not be any other rule in $\bar{S}$ with P in its head. The same holds regarding a rule $D \in S$ with a negated head such that $Head(D) = \neg R(x)$, $R(y) \in Body(D)$, and $R(x)\sigma = R(y)\sigma$. ■

Definition 3. Consider a set S of sh-rules. Suppose that a formula ϕ has been derived from S by performing a deduction process of resolution, $DPR(\phi)$. Let $HEADF(\phi)$ be a set of all head-factors computed in the course of $DPR(\phi)$. It has been pointed out in this section that if ρ is a renaming substitution for all negated predicate symbols of a set S of sh-rules then a ρ-renaming of any rule of S becomes a Horn clause. So, we call this transformation into Horn clauses a *hornization*. Let us call a set of renamed rules of S, $HORN(S, \phi)$, a *hornization of S with respect to a formula* ϕ if it includes renamings of all head-factors needed for deriving ϕ from S:

$$HORN(S, \phi) = \{C\rho \mid C \in (S \cup HEADF(\phi))\}. \quad ■$$

If we add all the rules of $HEADF(\phi)$ to S, then ϕ can be derived from $S \cup HEADF(\phi)$ without computing of any head-factor, and by Definition 3 ϕ can be derived from $S \cup HEADF(\phi)$ iff it is derivable from S. On the other hand, by Lemma 2, a formula $\bar{\phi}$ can be derived from a set $\bar{T}$ of ρ-renamed sh-rules of T iff a formula $\phi = \bar{\phi}\rho^{-1}$ is derivable from T without computing of any head-factor. This implies the following

Theorem 2. A formula ϕ is derivable from a set of sh-rules S iff its ρ-renaming is derivable from $HORN(S, \phi)$. ■.

Theorem 2 holds for recursive rules as well as for non-recursive ones since ρ-renaming preserves recursion, and the latter does not interfere with head-factoring. Indeed, if C is a recursive sh-rule such that $\neg P(x)$ appears in its head, and $\neg P(y)$ in its body, then $\bar{P}(x), \bar{P}(y)$ appear, respectively, in the head and body of $C\rho$, thus preserving the recursion. If the body of C contains also $P(z)$ that is unifiable with $P(x)$ in its head, then the corresponding head-factor F of C and its ρ-renaming $F\rho$ can be computed without interfering with the recursion between $\neg P(x)$ and $\neg P(y)$ (respectively, between $\bar{P}(x)$ and $\bar{P}(y)$).

Theorem 1 makes it possible that derivation of a formula from a general non-Horn system can be performed in a set of the corresponding sh-rules. An important practical consequence of Theorem 2 is that it determines conditions under which this process can be further reduced to handling of a set of hornized rules by employing numerous efficient techniques developed for Horn systems [2,3,7-11,14,17-19,21]. In particular, regarding a ground unit clause, Corollary 1 and Theorem 2 imply

Corollary 2. Consider a set of clauses S, a full set $SH(S)$ of sh-rules corresponding to S, a ground unit clause C, and a hornization of $SH(S)$ w.r.t. C, $HORN(SH(S), C)$. Then C is a fact in S iff $C\rho$ is a fact in $HORN(SH(S), C)$. ■

4. Relevant rules.

Consider a set of sh-rules S, and a unit clause C. Without loss of generality, let $C = P(x)$ (x is a vector of variables and constants). An instance of C can be derived *immediately* only from a rule with P in its head, so, denote by $S_0(P)$ a set of all such rules in S:

$$S_0(P) = \{A \mid A \in S \text{ and } P \in Head(A)\}.$$

Consider a rule B such that $Head(B) = P(y)$, $Body(B) = \{L_1, \ldots, L_m\}$. To produce an instance $P(y)\theta$ by using B we need to derive $L_i\theta$ for all $L_i \in Body(B)$. $L_i\theta$ can be produced immediately only by a rule with L_i in its head belonging to a set $S_0(L_i) = \{D \mid D \in S \text{ and } L_i \in Head(D)\}$. In the same way denote

$$S_1(P) = \bigcup_{B \in S_0(P)} \bigcup_{R \in Body(B)} S_0(R);$$

$$S_j(P) = \bigcup_{B \in S_{j-1}(P)} \bigcup_{R \in Body(B)} S_0(R). \qquad (1)$$

Definition 4. If L is a literal (a predicate symbol or its negation), and a rule A *may* be used in a derivation of an instance of a unit clause containing L, then A *is relevant to* L. Let $RELEV(L)$ denote a set of all rules of a system S that are relevant to L. Then $RELEV(L)$ is defined as follows:

(a) If $L \in Head(A)$ then $A \in RELEV(L)$;

(b) If $R \in Head(B)$ such that $R \in Body(D)$ and $D \in RELEV(L)$ then $B \in RELEV(L)$;

(c) No rule different from those of (a),(b) belongs to $RELEV(L)$.

Hence (cf. (1)),

$$RELEV(L) = \bigcup_{i \geq 0} S_i(L), \quad RELEV(L) \subseteq S. \ ■$$

By Definition 4 any rule not belonging to $RELEV(L)$ is useless with regard to derivation of an instance of L. Therefore all instances of L can be derived from $RELEV(L)$ which is only a subset of S. Hence, the following holds:

Theorem 3. A ground instance L' of a unit clause containing a literal L can be derived from a set of sh-rules S iff L' is derivable from a set of relevant rules $RELEV(L) \subseteq S$. ■

A set of sh-rules can be displayed by means of a *system graph* introduced in [10,11].

Definition 5. A *system graph*, $SG(S)$, of a set of sh-rules S is a directed bipartite graph consisting of a set of *ax-nodes* (*ax* for *axiom*) representing sh-rules, a set of *rel-nodes* (*rel* for *relation*) representing predicates, and a set of arcs, such that

(a) there is a distinct ax-node representing every sh-rule of S;

(b) there is a distinct rel-node for each predicate symbol P, and another one for $\neg P$;

(c) if there is a rule A in S such that $Body(A) = \{L_1, \ldots, L_k\}$, $Head(A) = L_{k+1}$ then there are arcs from rel-nodes representing $L_1, \ldots, L_k$ to the ax-node representing A, and an arc from the ax-node A to the rel-node representing L_{k+1}. ■

Example 3. Consider a set of clauses $S = \{A, B, C\}$:

$$A: \neg P(x,y) \vee \neg Q(x,y); \quad B: Q(v,w) \vee \neg R(v,w);$$

$$C: Q(t_1,t_2) \vee R(t_2,t_3) \vee \neg T(t_1,t_3).$$

The full set of sh-rules, $SH(S)$, corresponding to S is

$\{A_1: Q(x,y) \rightarrow \neg P(x,y); \quad A_2: P(x,y) \rightarrow \neg Q(x,y);$

$B_1: R(v,w) \rightarrow Q(v,w); \quad B_2: \neg Q(v,w) \rightarrow \neg R(v,w);$

$C_1: \neg R(t_2, t_3), T(t_1, t_3) \rightarrow Q(t_1, t_2);$

$C_2: \neg Q(t_1, t_2), T(t_1, t_3) \rightarrow R(t_2, t_3);$

$C_3: \neg Q(t_1, t_2), \neg R(t_2, t_3) \rightarrow \neg T(t_1, t_3)\}.$

The ρ-renaming of $SH(S)$ is

$\{\bar{A}_1: Q(x, y) \rightarrow \bar{P}(x, y); \quad \bar{A}_2: P(x, y) \rightarrow \bar{Q}(x, y);$

$\bar{B}_1: R(v, w) \rightarrow Q(v, w); \quad \bar{B}_2: \bar{Q}(v, w) \rightarrow \bar{R}(v, w);$

$\bar{C}_1: \bar{R}(t_2, t_3), T(t_1, t_3) \rightarrow Q(t_1, t_2);$

$\bar{C}_2: \bar{Q}(t_1, t_2), T(t_1, t_3) \rightarrow R(t_2, t_3);$

$\bar{C}_3: \bar{Q}(t_1, t_2), \bar{R}(t_2, t_3) \rightarrow \bar{T}(t_1, t_3)\}.$

The set of ρ-renamed rules relevant to $\bar{P}$:

$$RELEV(\bar{P}) = \{\bar{A}_1, \bar{A}_2, \bar{B}_1, \bar{B}_2, \bar{C}_1, \bar{C}_2\}.$$

(Note that $\bar{C}_3 \notin RELEV(\bar{P})$).

The system graph of $RELEV(\bar{P})$ is displayed in Fig. 2 in solid lines. Ax-nodes are round, rel-nodes — square. ■

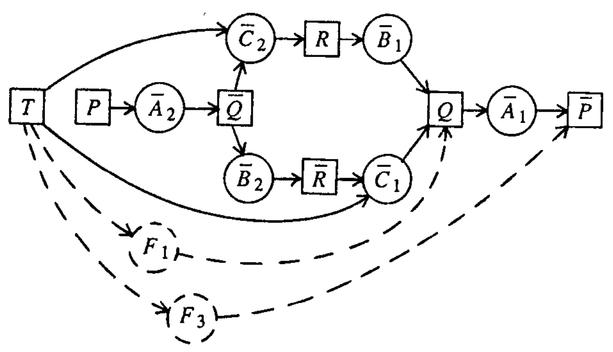

Fig. 2.

Definition 6. If there is a directed path (or an arc, in particular) in a system graph SG from a node v_1 to a node v_2, then v_1 is a *predecessor* (resp., an *immediate predecessor*) of v_2, and v_2 is a *successor* (resp., an *immediate successor*) of v_1. We use $pred(v_2)$, $impred(v_2)$, $succ(v_1)$, $imsucc(v_1)$ to denote these relationships between nodes of a system graph. ■

5. Computing facts.

Given a goal of computing all facts satisfying a unit clause $C = L(x)$, and having determined the set of sh-rules relevant to L, $RELEV(L)$, we should now produce its hornization, $HORN(RELEV(L), L)$ in order to perform evaluation of L entirely in a set of Horn clauses (cf. Theorem 2, Corollary 2). For this purpose we need a set $HEADF(L)$ of all head-factors useful for deriving L from $RELEV(L)$. The following Lemma is helpful:

Lemma 3. Let S be a set of sh-rules, and SG — its system graph. Then a rule with literals L_1 in its body and L_2 in its head can be derived from S iff node L_1 is a predecessor of node L_2 in SG. ■

This Lemma provides grounds for computing $HEADF(L)$:

Algorithm 1. (Given $RELEV(L)$, returns $HEADF(L)$)

1) Construct a system graph SG for $RELEV(L)$.
2) $HEADF(L) := \varnothing$. For every pair of rel-nodes L_1, L_2 such that one contains negation of the predicate symbol of the other, and $L_1 = pred(L_2)$ perform 2.1) — 2.3):

2.1) Compute a *path-resolvent*, $PRes(L_1, L_2)$ defined as follows. Let a directed path from L_1 to L_2 be $Path(L_1, L_2) = \{L_1, A_1, P_1, A_2, P_2, \ldots, P_{m-1}, A_m, L_2\}$ such that $P_1, \ldots, P_{m-1}$ are rel-nodes, $A_1, \ldots, A_m$ are ax-nodes. Compute first $Res(A_1, A_2)$ upon P_1, with P_2 in its head, then $Res(A_1, A_i) = Res(Res(A_1, A_{i-1}), A_i)$ for all $i = 3, \ldots, m-1$, and finally, $PRes(L_1, L_2) = Res(A_1, A_m)$ such that $Head(PRes(L_1, L_2)) = L_2\sigma$, and $L_1\sigma \in Body(PRes(L_1, L_2))$, where σ is the corresponding substitution.

2.2) If there is a substitution θ such that $L_1\sigma\theta = \neg L_2\sigma\theta$ then compute a head-factor, $F = PRes(L_1, L_2)\theta$.

2.3) $HEADF(L) := HEADF(L) \cup \{F\}$. ■

Example 4. Consider again Example 3 and the set of rules relevant to $\bar{P}$. In its system graph (Fig. 2, solid lines) we have:

$\bar{Q} = pred(Q)$, $P = pred(\bar{P})$;

$Path_1(\bar{Q}, Q) = \{\bar{Q}, \bar{C}_2, R, \bar{B}_1, Q\}$;

$Path_2(\bar{Q}, Q) = \{\bar{Q}, \bar{B}_2, \bar{R}, \bar{C}_1, Q\}$;

$Path_1(P, \bar{P}) = \{P, \bar{A}_2, \bar{Q}, \bar{C}_2, R, \bar{B}_1, Q, \bar{A}_1, \bar{P}\}$;

$Path_2(P, \bar{P}) = \{P, \bar{A}_2, \bar{Q}, \bar{B}_2, \bar{R}, \bar{C}_1, Q, \bar{A}_1, \bar{P}\}$.

So, the following path-resolvents and their head-factors can be computed:

$PRes_1(\bar{Q}, Q) : \bar{Q}(t_1, t_2), T(t_1, t_3) \to Q(t_2, t_3)$,

$F_1 : T(t_1, t_1) \to Q(t_1, t_1)$;

$PRes_2(\bar{Q}, Q) : \bar{Q}(t_2, t_3), T(t_1, t_3) \to Q(t_1, t_2)$,

$F_2 = F_1 : T(t_1, t_1) \to Q(t_1, t_1)$;

$PRes_1(P, \bar{P}) : P(t_1, t_2), T(t_1, t_3) \to \bar{P}(t_2, t_3)$,

$F_3 : T(t_1, t_1) \to \bar{P}(t_1, t_1)$;

$PRes_2(P, \bar{P}) : P(t_2, t_3), T(t_1, t_3) \to \bar{P}(t_1, t_2)$,

$F_4 = F_3 : T(t_1, t_1) \to \bar{P}(t_1, t_1)$.

Hence, $HORN(RELEV(\bar{P}), \bar{P}) = \{\bar{A}_1, \bar{A}_2, \bar{B}_1, \bar{B}_2, \bar{C}_1, \bar{C}_2, F_1, F_3\}$. The addition of F_1, F_3 is shown in Fig. 2 in broken lines. ■

Given a database DB (either Horn or non-Horn) and an atomic query $q : L(x)$, $L \in (P, \neg P)$, the following algorithm derives all facts satisfying q by reducing DB to a set of hornized rules relevant to q.

Algorithm 2. (given DB and q, returns $Facts(q)$)

1) Compute a full set $SH(DB)$ of sh-rules corresponding to all clauses of DB.
2) Compute a set $RELEV(q)$ of rules relevant to q.
3) By applying Algorithm 1 compute a set $HEADF(q)$ of head-factors for q.
4) Produce a set $HORN(RELEV(q), q)$ of Horn clauses by ρ-renaming of all the rules of $RELEV(q) \cup HEADF(q)$. (Denote $q' = q\rho$)
5) Apply to $HORN(RELEV(q), q)$ any query evaluation method that is sound and complete for Horn systems, producing all facts satisfying q' that are derivable from $HORN(RELEV(q), q)$.
6) Compute a set $Facts(q)$ of all facts satisfying q that are derivable from DB, by ρ^{-1}-renaming of all facts produced at step 5). ■

Theorem 4. Algorithm 2 is sound and complete in the sense that given a database DB and an atomic query q it produces all and only the facts that are derivable from DB and satisfy q.

Proof. By Corollary 1 a ground unit clause is a fact in DB iff it is a fact in $SH(DB)$ computed at step 1) of Algorithm 2. By Corollary 2 and Theorem 3 a ground instance C of q is a fact in $SH(DB)$ iff $C\rho$ is a fact in $HORN(RELEV(q), q)$ computed at step 4). The query evaluation method applied at step 5) is sound and complete for $HORN(RELEV(q), q)$, hence step 6) computes all and only the facts for q derivable from DB. ■

Conventionally, a database DB consists of an *intentional* part, IDB — a set of non-unit clauses (rules, axioms), and an *extensional* one, *EDB* — a set of ground unit clauses (facts). To estimate the complexity of Algorithm 2 denote by n the number of clauses in the *IDB*, by m — the maximum number of literals in a clause of *IDB*, by h — the number of head-factors computable in $RELEV(q)$ in the course of evaluating q. Let $HT(n, m)$ stand for *run-time complexity* of deriving all facts for an atomic query by an efficient algorithm in a Horn database, and $NHT(n, m)$ correspond to achieving of the same goal by Algorithm 2 in a non-Horn one. If both databases have the same volume of *EDB*, then $NHT(n, m) \le HT(m(n+h), m)$.

It should be noted that sets $RELEV(q)$ and $HEADF(q)$ are determined by the predicate symbol appearing in q, but not by the particular binding of its terms. Therefore steps 1) — 4) may and should be preprocessed at the system design stage for all (or most frequently queried) literals, so only steps 5),6) are to be performed at a query run-time.

Acknowledgments.

Many thanks to the anonymous referees for their very helpful comments.

References.

1. Apt, K., Blair, H., and Walker, A., Towards a theory of declarative knowledge. *Proc. Workshop on Foundations of Deductive Databases and Logic Programming,* Minker, J., (ed.), Washington, D.C., 1986, 546-629.
2. Bancilhon, F., Maier, D., Sagiv, Y., and Ullman, J.D., Magic sets and other strange ways to implement logic programs. *Proc. 5th ACM SIGACT-SIGMOD Symp. on Principles of Database Systems* (Cambridge, 1986), 1-15.
3. Beeri, C., and Ramakrishnan, R., On the power of magic. *Proceedings 6th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (San Diego, 1987).*
4. Chang, C.-L., and Lee, R. C.-T., *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, 1973.
5. Clark, K.L., Negation as failure. *Logic and Data Bases,* Gallaire, H., and Minker, J. (eds.), Plenum Press, New York, 1978, 293-322.
6. Gabbay, D.M., Theoretical foundations for non-monotonic reasoning in expert systems. *Logics and Models of Concurrent Systems,* Apt, K.R. (Ed.), NATO ASI Series, vol. F13, Springer-Verlag, 1985, 439-455.
7. Gallaire, H., Minker, J., and Nicolas, J.-M., Logic and databases: a deductive approach. *ACM Computing Surveys 16* (1984), no. 2, 153-186.
8. Gardarin, G., and De Mainderville, C., Evaluation of database recursive logic programs as recurrent function series. *Proceedings of the ACM SIGMOD Intl. Conf. on Management of Data* (Washington, D.C., 1986).
9. Henschen, L. J., and Naqvi, S. A., On compiling queries in recursive first-order databases. *J. ACM 31,* 1 (Jan. 1984), 47-85.
10. Kifer, M., and Lozinskii, E.L., Filtering data flow in deductive databases. *Lecture Notes in Computer Science,* v. 243, Springer-Verlag, 1986, 186-202.
11. Lozinskii, E.L., Evaluating queries in deductive databases by generating. *Proceedings 9th Intl. Joint Conference on Artificial Intelligence* (Los Angeles, 1985), 173-177.
12. McCarthy, J., Circumscription - a form of non-monotonic reasoning. *Artificial Intelligence 13* (1980), 27-39.
13. Minker, J., On indefinite databases and the closed world assumption. *Lecture Notes in Computer Science 138,* Springer- Verlag, 1982, 292-308.
14. Minker, J., and Nicolas, J.-M., On recursive axioms in deductive databases. *Information Systems 8,* 1 (1983), 1-13.
15. Reiter, R., On closed world data bases. *Logic and Data Bases,* Gallaire, H., and Minker, J. (Eds.), Plenum Press, New-York, 1978, 55-76.
16. Reiter, R., A logic for default reasoning. *Artificial Intelligence 13* (1980), 81-132.
17. Sacca, D., and Zaniolo, C., On the implementation of a simple class of logic queries for databases. *Proceedings 5th ACM SIGACT-SIGMOD Symposium on Principles of Databases Systems* (Cambridge, 1986), 16-23.
18. Ullman, J. D., Implementation of logical query languages for databases. *ACM Trans. Database Syst., 10* 3 (Sept. 1985), 289-321.
19. Van Gelder, A., Message passing framework for logical query evaluation. *Proceedings of the ACM SIGMOD Intl. Conference on Management of Data* (Washington, D.C., 1986), 155-165.
20. Van Gelder, A., Negation as failure using tight derivations for general logic programs. *Proc. Third IEEE Symp. on Logic Programming,* Salt Lake City, Utah, 1986.
21. Vieille, L., Recursive axioms in deductive databases: the query/subquery approach. *Proceedings 1st Intl. Conference on Expert Database Systems* (Charleston, 1986), 179-194.
22. Yahya, A., and Henschen, L.J., Deduction in non-Horn databases. *J. of Automated Reasoning 1* (1985), 141-160.