A Procedural Semantics for Disjunctive Closed World Assumption*

WANG Ke-wen, ZHOU Li-zhu, FENG Jian-hua

(*Department of Computer Science and Technolgy, Tsinghua University, Beijing* 100084, *China*)

E-mail: {kewen,dcszlz,fengjh}@tsinghua.edu.cn

http://www.tsinghua.edu.cn

Received January 10, 2000; accepted February 7, 2001

Abstract: Recently there has been an increasing interest in representing disjunctive information. DCWA (disjunctive closed world assumption) is a skeptical semantics for disjunctive deductive databases (DDBs) (with default negation) and extends the well-founded model for normal logic programs. DCWA also provides an approximation for the generalized closed world assumption (GCWA) and supports argumentation. This paper presents a top-down procedure for DCWA and proves its soundness and completeness.

Key words: disjunctive deductive databases; closed world assumption; semantics

Recently there has been increasing interest in representing disjunctive information. The obvious reason is that we are often required to deal with disjunctive information in representing our common knowledge information. The extension of deductive databases by introducing disjunction in the heads of database rules (i.e. disjunctive deductive databases) has been widely accepted as a promising tool for representing incomplete knowledge. The closed world reasoning in databases is actually a kind of default negation. However, the task of defining a precise meaning for default negation (and disjunction) is proved to be difficult. Reiter's[1] closed world assumption (CWA) provides such an excellent semantics for performing closed world reasoning in non-disjunctive databases. Minker's GCWA[2] is a natural generalization of CWA for disjunctive deductive databases without default neagtion and there are also some other proposals to extend CWA.[3~6] However, all of these proposals bear their own drawbacks and do not support argumentation. In Ref. [7], we defined a form of closed world reasoning DCWA for general disjunctive databases (denoted there as $GCWA^G$).

However, DCWA lacks a suitable procedural semantics. The main contribution of this paper is to incorporate default negation into traditional SLI-resolution[8] and to provide a top-down procedure for DCWA. We prove that this procedure is sound and complete with respect to DCWA. The rest of this paper is arranged as follows: in Section 1 we specify our notations and briefly recall DCWA; Some definitions related to SLIN-resolution are included in Section 2; The completeness and soundness of the SLIN-resolution with respect to DCWA is proved in Section

* Supported by the National Natural Science Foundation of China under Grant Nos. 69883008, 69773027 (国家自然科学基金); the National Grand Fundamental Research 973 Program of China under the Grant No. G1999032704 (国家重点973基础研究发展规划)

WANG Ke-wen is an associate professor in the Department of Computer Science Technology at Tsinghua University. His research interests are in logic programming, knowledge representation and database theory. **ZHOU Li-zhu** is head and professor of the Department of Computer Science Technology at Tsinghua University. His research has spanned many aspects of database systems. **FENG Jian-hua** is an associate professor in the Department of Computer Science Technology at Tsinghua University. His research interests are in database systems.

3; Section 4 is the conclusion.

1 CWA for General Disjunctive Databases

A general disjunctive deductive database (simply, disjunctive database) P is defined as a set of rules of the form: $p_1 \vee \ldots \vee p_r \leftarrow p_{r+1}, \ldots, p_s, \sim p_{s+1}, \ldots, p_n$.
Here, $n \geqslant s \geqslant r > 0$ and p_i's are atoms for $i=1,\ldots,n$. $\vee$ and not denote non-classical disjunction and default negation, respectively.

If $n=s$, the above rule is said to be positive. P is a positive disjunctive database if each rule of P is positive.

B_P is the Herbrand base of P (the set of all atoms in P for propositional P).

A default literal (or, negative literal) is of form $\sim p$ where $p \in B_P$ and is also called an assumption of P. A hypothesis Δ of P is a set of its assumptions. The set of all hypotheses of P is written as $H(P)$.

For simplicity, we also express a rule C in P as the form of $\Sigma \leftarrow \Pi_1, \sim \Pi_2$,
where Σ is a disjunction of atoms in B_P, Π_1 an unordered sequence of finite atoms of B_P denoting a conjunction of atoms, and $\sim \Pi_2 = \{\sim q : q \in \Pi_2\}$ for $\Pi_2 \subseteq B_P$ denoting a conjunction of negative literals.

Denote $\sim \Sigma = \sim p_1 \wedge \ldots \wedge \sim p_t$ if $\Sigma = p_1 \vee \ldots \vee p_t$.

Definition 1.1. Let Δ be a hypothesis, then

(1) For each rule C in P, delete all the negative literals in the body of C that belong to Δ. The resulting disjunctive database is denoted as P_Δ;

(2) The positive disjunctive database consisting of all the positive rules of P_Δ is denoted as P_Δ^+, and is said to be the generalized GL-transformation of P.

Example 1.1. Consider the disjunctive database P:

$$p \vee q \leftarrow \sim u$$
$$v \leftarrow p, \sim q$$

If $\Delta = \{\sim u, \sim p\}$, then P_Δ^+ is the positive disjunctive database:

$$p \vee q \leftarrow$$

Definition 1.2. Let Δ be a hypothesis of disjunctive database P and Σ a disjunction of atoms. $\Delta \vdash_P \Sigma$ if there is a disjunction Σ' of atoms such that $P_\Delta^+ \vdash \Sigma'$ and $\Sigma' = \Sigma \vee \Sigma''$ for some disjunction Σ'', where '$\vdash$' is the inference in the first order logic.

In Example 1.1, if $\Delta = \{\sim u, \sim p\}$, then $P_\Delta^+ \vdash q$.

Originally, DCWA is defined as the least alternating fixpoint but it also possesses the following equivalent characterization.

Definition 1.3. Let P be a disjunctive database, Δ and Δ' be two hypotheses of P. If there exists an assumption $\sim q \in \Delta'$ such that $\Delta \vdash_P q$, then we say Δ attacks Δ', written $\Delta \rightsquigarrow_P \Delta'$. In particular, Δ is said to be an attacker of an assumption $\sim q$ if $\Delta \rightsquigarrow_P \{\sim q\}$.

If Δ is an attacker of an assumption $\sim p$ and there is no attacker Δ' of $\sim p$ such that $\Delta' \subset \Delta$, then we say Δ is a minimal attacker of $\sim p$.

For instance, if we take $\Delta = \{\sim u, \sim p\}$ and $\Delta' = \{\sim q\}$ in Example 2.1. Then $\Delta \rightsquigarrow_P \Delta'$. That is, Δ is an attacker of $\sim q$ and is also a minimal one. Another attacker of $\sim q$ is the hypothesis $\{\sim v\}$.

Definition 1.4. Let Δ be a hypothesis of P. An assumption $\sim p$ is acceptable with respect to Δ if $\Delta \rightsquigarrow_P \Delta'$ for any attacker Δ' of $\sim p$.

In Example 1.1, the assumption $\sim q$ is acceptable wrt. $\Delta = \{\sim \mu, \sim q\}$.

Notice that, if "attacker" is replaced by "minimal attacker" in Definition 2.4, we shall get an equivalent definition of A_P. This observation will be used in the proof of subsequent results.

Set $A_P(\Delta)=\{\sim p:\sim p$ is acceptable with respect to $\Delta\}$. Then A_P defines an operator from $H_P\to H_P$. This operator is monotonic but not necessarily continuous. Thus, the A_P has the least fixpoint $A_P\uparrow\gamma$, where γ is an ordinal.

Definition 1.5. Let P be a disjunctive database. Then the closed world assumption DCWA for P is defined as $\mathrm{DCWA}(P)=A_P\uparrow\gamma$, where γ is an ordinal.

If P is the disjunctive database in 2.1, then $\mathrm{DCWA}(P)=\{\sim v,\sim p\}$.

DCWA is no stronger than the semantics WFDH defined in Ref. [9]: $\mathrm{DCWA}(P)\subseteq\mathrm{WFDH}(P)$ for any disjunctive database P. The following example shows that WFDH is strictly strong than DCWA.

Example 1.2. Consider the disjunctive database P:

$$p\vee q\leftarrow\sim u$$

$$q\vee u\leftarrow$$

We know that $\mathrm{WFDH}(P)$ contains $\sim p$ but $\sim p\notin\mathrm{DCWA}(P)$.

2 SLIN-Resolution

In this section, we shall provide a top-down procedure for DCWA which is an extension of SLI-resolution[8] by incorporating default negation. From now on, we assume that P is a finite propositional database.

A goal G is of the form: $\leftarrow l_1,\ldots,l_r,\sim a_1,\ldots,\sim a_s$, where each l_i is an atom a or its negation $\neg a$ for $i=1,\ldots,r$ and, a_is are atoms. Distinct from default literals, l is said to be a classic literal if $l=a$ or $l=\neg a$.

A goal is negative if it is of the form $\leftarrow\neg a_1,\ldots,\neg a_r,\sim a_{r+1},\ldots,\sim a_n$ where a_i is an atom and $r\leqslant n$.

In our resolution- like procedure, given database rule $C:\Sigma\leftarrow\Pi_1,\sim\Pi_2$, we transform C to the goal $gt(C):\leftarrow\neg\Sigma,\Pi_1,\sim\Pi_2$ and call it the goal transformation of C. Since our resolution is to resolve literals in both heads and bodies of database rules, this transformation allows a unified and simple approach.

The special goal $\leftarrow$ is called an empty goal. The empty goal $\leftarrow$ is also written as the familiar symbol $\square$. The non-empty goal of form $\leftarrow\neg\Sigma,\sim\Pi$ is said to be a negative goal.

Given a disjunctive database P, set $gt(P)=\{gt(C):C\in P\}$. The traditional goal resolution can be generalized to $gt(P)$ as follows.

Goal Resolution (GR):

If $G:\leftarrow l,\neg\Sigma,\Pi_1,\sim\Pi_2$ and $G':\leftarrow l',\neg\Sigma',\Pi'_1,\sim\Pi'_2$ are two goals such that classic literals l and l' are complementary, then the GR-resolvant of G with G' on selected literal l is the goal $\leftarrow\neg\Sigma,\neg\Sigma',\Pi_1,\Pi'_1,\sim\Pi_2,\sim\Pi'_2$.

GR is actually a variant of the SLI-rule defined in Ref. [8].

Definition 2.1. An SLIN-derivation from G in $gt(P)$ is a sequence of goals: $G_0,G_1,\ldots,G_n$, where $G_0=G$ and, for $i=0,\ldots,n-1$, G_{i+1} is obtained from one of the following two steps:

1. G_{i+1} is the GR-resolvant of G_i with a goal in $gt(P)\cup\{G_0,\ldots,G_i\}$ on selected literal, or

2. If G_i is the goal: $\leftarrow\neg\Sigma^{(i)},\Pi_1^{(i)},\sim p,\sim\Pi_2^{(i)}$ such that there exists a success positive tree T_p^+ for the goal $\leftarrow p$. Then G_{i+1} is the goal $\leftarrow\neg\Sigma^{(i)},\Pi_1^{(i)},\sim\Pi_2^{(i)}$.

An SLIN-refutation for G is an SLIN-derivation: $G_0,G_1,\ldots,G_n$ such that $G_0=G$ and $G_n=\square$.

Let P be a disjunctive database and G a goal. A positive tree T_G^+ for G is defined as follows:

1. The root of T_G^+ is G.

2. For each node $G':\leftarrow\neg\Sigma',p,\Pi'_1,\sim\Pi'_2$, and each goal G_i in $gt(P)$, if G'_i is the GR-resolvant of G' with G_i on p and G'_i is different from all nodes in the branch of G', then G' has a child G'_i.

We distinguish three types of leaves in a positive tree (there may be other leaves):

1. Empty leaves which are labeled by the empty rule.

2. Dead leaves which contain atoms that cannot be resolved with any goal in $gt(P)$ by GR-rule.

3. Failure leaves which are goals of the form $\leftarrow\neg p_1,\ldots,\neg p_n,\sim p_{n+1},\ldots,\sim p_m (m\geqslant n)$ such that, for some i $(1\leqslant i\leqslant n)$, $\leftarrow p_i$ has an SLIN-refutation in $gt(P)$.

T_G^+ is success if T_G^+ contains only two types of leaves: dead leaves and failure leaves.

Our SLIN-resolution consists of SLIN-derivations and positive trees.

If the goal G is of form $\leftarrow p$, its positive tree T_G^+ is also written as $T^+(p)$.

Let us illustrate the SLIN-resolution with an example.

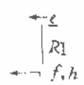
Example 2.1. Let P_2 consist of the following rules:

$$a\vee b\leftarrow c,\sim d$$
$$b\leftarrow\sim e,\sim c$$
$$e\vee f\leftarrow h$$
$$a\vee f\leftarrow c$$

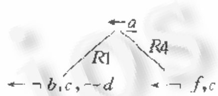
$gt(P_2)$ is as follows:

$$R1:\leftarrow\neg a,\neg b,c,\sim d$$
$$R2:\leftarrow\neg b,\sim e,\sim c$$
$$R3:\leftarrow\neg e,\neg f,h$$
$$R4:\leftarrow\neg a,\neg f,c$$

The sequence of goals $G_0=\leftarrow b, G_1=\leftarrow\sim e,\sim c, G_2=\leftarrow\sim c, G_3=\leftarrow$ is an SLIN-refutation for the goal $\leftarrow b$, since both $\leftarrow e$ and $\leftarrow c$ have success positive trees. In fact, the positive tree of the goal $\leftarrow c$ is itself and the positive tree of the goal $\leftarrow e$ is as follows (these two trees have only dead leaves):



The goal $\leftarrow a$ has the following success positive tree $T^+(a)$ since its two leaves $\leftarrow\neg b,c,\sim d$ and $\leftarrow\neg f,c$ are both dead:



It is not hard to see that $\sim a\in \mathrm{DCWA}(P_2)$ and $\mathrm{DCWA}(P_2)\vdash_{P_2} b$.

3 Completeness and Soundness of SLIN-Resolution

In general, we have the following soundness and completeness theorem.

Theorem 3.1 (Soundness and Completeness of SLIN-resolution).

Let P be a disjunctive database and $p\in B_P$.

1. $\sim p\in \mathrm{DCWA}(P)$ if and only if there exists a success positive tree for the goal $G:\leftarrow p$.

2. $\mathrm{DCWA}(P)\vdash_P p$ if and only if there exists an SLIN-refutation for the goal $G:\leftarrow p$.

Notice that the notions of SLIN-refutation and success positive trees are defined recursively and we can make them more mathematically precise according to their ranks: for an SLIN-derivation D and a positive tree T_G^+,

rank$(D)=0$ if there is no positive tree involved in the definition of D.

rank$(T_G^+)=1$ if there is no SLIN-derivation involved in the definition of failure leaves of T_G^+.

For $k\geqslant 1$, rank$(D)=k$ if D is defined through positive trees whose ranks are no more than k and at least one positive tree having rank k is involved.

For $k \geqslant 2$, rank$(T_G^+) = k$ if, for each failure leaf L: *leftarrow* $\neg p_1, \ldots, \neg p_n, \sim p_{n+1}, \ldots, \sim p_m, m > n$, there exists i $(1 \leqslant i \leqslant n)$ such that $\leftarrow p_i$ has an SLIN-refutation in $gt(P)$ whose rank is no more than $k-1$ and at least one such rank is $k-1$.

The proof of Theorem 3.1 is directly derived from the following two lemmas.

Lemma 3.1. For any disjunctive database P and a goal G:

1. If there exists a success positive tree T_G^+ for $G: \leftarrow p$, then $\sim p \in$ DCWA(P).

2. If there exists an SLIN-refutation R for $G: \leftarrow p$, then DCWA$(P) \vdash_P p$.

Proof. It suffices to prove that the following two items hold:

(1) $A_P^k(\varnothing) \vdash_P p$ if $rank(R) = k$ for $k \geqslant 0$ and

(2) $\sim p \in A_P^k(\varnothing)$ if $rank(T_G^+) = k$ for $k \geqslant 1$.

First, for $k=0$, by induction on the length of R, we have that (1) holds.

It remains to prove that both (1) and (2) hold for $k \geqslant 1$ by using simultaneous induction on the rank k of SLIN-refutation and positive tree:

Basis: $k=1$. If positive tree T_G^+ has rank 1, then the assumption $\sim p$ has no attackers. Thus $\sim p$ is admissible wrt $\varnothing$, i.e. $\sim p \in A_P(\varnothing)$. Further, if R has rank 1, then all selected literals of form $\sim q$ in R are in $A_P(\varnothing)$. Thus $A_P(\varnothing) \vdash_P p$.

Induction: Assume that both (1) and (2) above hold for rank $k-1$. We want to show that (1) and (2) hold for rank k:

To prove (2): if rank$(T_G^+) = k$: Suppose that G has t failure leaves $L_1, \ldots, L_t$ in T_G^+. Then, for each failure leaf $L_i: \leftarrow \sim q_1, \ldots, \sim q_r, \sim q_{r+1}, \ldots, \sim q_s$, it corresponds to a minimal attacker $\Delta_i = \{\sim q_1, \ldots, \sim q_s\}$ of the assumption $\sim p$. Moreover, $\Delta_1, \ldots, \Delta_t$ are all minimal attackers of $\sim p$. For each Δ_i, L_i is a failure leaf and thus there is some $q_j (1 \leqslant j \leqslant s)$ such that the goal $\leftarrow q_j$ has an SLIN-refutation R' in $gt(P)$. Notice that $rank(R') \leqslant k-1$. Thus, by induction assumption, $A_P^{k-1}(\varnothing) \vdash_P q_j$. Thus, $A_P^{k-1}(\varnothing) \twoheadrightarrow_P \{\sim q_1, \ldots, \sim q_s\}$. This implies that $A_P^{k-1}(\varnothing) \twoheadrightarrow_P \Delta'$ for any minimal attacker Δ' of $\sim p$. Thus, $\sim p \in A_P(A_P^{k-1}(\varnothing))_P = A_P^k(\varnothing)$.

To prove (1): if $rank(R) = k$: by induction assumption and the definition of SLIN-resolution, all selected literals of form $\sim q$ are in $A_P^k(\varnothing)$ and thus, (1) holds for $rank(R) = k$. □

Lemma 3.2. For any disjunctive database P and a goal G:

1. If DCWA$(P) \vdash_P p$, then there is an SLIN-refutation for the goal $G: \leftarrow p$.

2. If $\sim p \in$ DCWA(P), then there is a success positive tree for the goal $G: \leftarrow p$.

Proof. It is enough to show that both of the following C1 and C2 hold:

C1. For $k \geqslant 0$, if $A^k(\varnothing) \vdash_P p$, then there exists an SLIN-refutation R for goal $G: \leftarrow p$ such that $rank(R) \leqslant k$.

C2. For $k \geqslant 1$, if $\sim p \in A_P^k(\varnothing)$, then the positive tree T_G^+ for $G: \leftarrow p$ is success and $rank(T_G^+) \leqslant k$.

For $k=0$: if $\varnothing \vdash_P p$, then $P_\varnothing^+ \vdash p$, where $\vdash$ is the inference relation in the classic logic and $P_\varnothing^+$ is the generalized GL-transformation of P wrt. the empty hypothesis. By the completeness of the SLI-resolution[8], there is an SLI-refutation R for G. Notice that R is also SLIN-refutation R for G and $rank(R) = 0$.

We prove that C1 and C2 hold for $k \geqslant 1$ by using simultaneous induction on rank k.

basis. For $k=1$: that is, if $\sim p \in A_P(\varnothing)$, suppose that T_G is the positive tree for G whose negative leaves (negative goals and empty goal) are $L_1, \ldots, L_t$, where $L_i: \leftarrow \neg\Sigma^{(i)}, \sim\Pi^{(i)}$. Write $\Delta'_i = \{\sim q : q \in atoms(\Sigma^{(i)}) \cup atoms(\Pi^{(i)})\}$. Here, $atoms(E)$ denotes the set of atoms appearing in the expression E. Obviously, $\Delta'_i \vdash_P p$ for $i = 1, \ldots, t$ and $\Delta_1, \ldots, \Delta_t$ are all minimal attackers of the assumption $\sim p$. By $\sim p \in A_P(\varnothing)$, it should be the case $\varnothing \twoheadrightarrow_P \Delta'_i$ (Notice that this also means Δ'_i is not empty). Thus, for any $i = 1, \ldots, t$, there is an assumption $\sim q_i \in \Delta'_i$, $\varnothing \vdash_P q_i$. By the case of $k=0$ just proved above, this implies that there exists an SLIN-refutation R'_i for $\leftarrow q_i$

such that $rank(R'_i)=0$. Thus, we have shown: (1) the rank of T^i_G is no more than 1; (2) there is no empty leave; (3) every negative leaves is a failure one. Therefore, C2 holds for $k=1$.

If $A(\varnothing)\vdash_{\neg}p$, similar to the proof of C1 for $k=0$, we can see that the goal $G:\leftarrow p$ has an SLIN-refutation $R'_1:D'_1,\ldots,D'_t$ in $A_P(\varnothing)^1$. R'_1 corresponds to an SLIN-derivation R_1 in P (not necessarily a SLIN-refutation in P): $D_1,\ldots,D_t$ satisfying that D_t is a negative goal: $\leftarrow\sim d_1,\ldots,\sim d_r$, where $D_i\in A_P(\varnothing)$. Define $D_{t+i}:\leftarrow\sim d_i,\ldots,\sim d_r$ for $i=1,\ldots,r$. Then D_{t+r} is the empty goal. By the conclusion of C1 for $k=0$, $R:D_1,\ldots,D_t,D_{t+1},\ldots,D_{t+r}$ is an SLIN-refutation for G and the rank of R is no more than 1.

Induction. Assume that both C1 and C2 hold for any natural number that is not more than k, similar to the proof of the case for $k=1$, we can show that C1 and C2 hold for $k+1$. □

4 Conclusion

In this paper, we have proposed a procedural semantics for DCWA and shown that it is complete and sound. The basic idea of our top-down procedure is to incorporate default negation into traditional SLI-resolution. Our procedure has been implemented in Prolog by Peter Baumgartner of Koblenz University. It should be pointed out that the disjunctive database can also be preprocessed by the fixpoint transformation *Lft* introduced in Ref. [9]. *Lft* transforms each disjunctive deductive database P into a negative database $Lft(P)$ (i.e. without atoms in bodies of the rules of P). Thus the given disjunctive database can be optimized before it is evaluated by SLIN-resolution. For limitation of space, we shall not discuss this issue in detail. We are also working on providing such a top-down procedure for WFDH in Ref. [8] and the credulous semantics in Ref. [10].

References:

[1] Reiter, R. On the closed world databases. In: Gallaire, H., Minker, J., eds. Logic and Data Bases. New York: Plenum Press, 1978. 119~140.

[2] Minker, J. On indefinite databases and the closed world assumption. LNCS 138, 1982. 292~308.

[3] Brass, S., Dix, J. Semantics of disjunctive logic programs based on partial evaluation. Journal of Logic Programming (to appear), 1998. Extended abstract appeared as disjunctive semantics based upon partial and bottom-up evaluation. In: Proceedings of the 12th International Logic Programming Conference. MIT Press, 1995. 199~213.

[4] Baral, C., Lobo, J., Minker, J. Generalized disjunctive well founded semantics for logic programs: declarative semantics. In: Proceedings of the 5th International Symposium on Methodologies for Intelligent Systems. Knoxville, TN, 1990. 465~473.

[5] Rajasekar, A., Minker, J. On stratified disjunctive programs. Annals of Mathematics and Artificial Intelligence, 1(1~4):339~357.

[6] Sakama, C., Inoue, K. Negation in disjunctive logic programs. In: Proceedings of the 10th International Conference on Logic Programming (ICLP'93). MIT Press, 1993. 703~719.

[7] Wang, K., Lin, F. Closed world reasoning and query evaluation in disjunctive deductive databases. In: Proceedings of the 1999 International Conferenceon Applications of Prolog. Japan, 1999.

[8] Lobo, J., Minker, J., Rajasekar, A. Foundations of Disjunctive Logic Programming. MIT Press, 1992.

[9] Wang, K. Argumentation-based abduction in disjunctive logic programming. Journal of Logic Programming, 2000,45(1-3):105~141.

[10] Wang, K., Chen, H., Wu, Q. Credulous argumentation with the disjunctive stable semantics. Science in China (Series E), 1998,41(3):330~336.

析取封闭世界假设的一种过程语义

王克文, 周立柱, 冯建华

(清华大学 计算机科学与技术系,北京 100084)

摘要:析取信息的表示是一个重要的研究问题.DCWA(析取封闭假设)为一般演绎数据库提供了一种谨慎语义,并且扩充了标准的良基语义.同时DCWA支持争论推理,为广义封闭世界假设提供了一种逼近.基于此,提出了DCWA的过程语义,并证明了它的可靠性和完备性.

关键词:演绎数据库;封闭世界假设;语义

中图法分类号:TP18 **文献标识码**:A