
Journal of Intelligent Information Systems, 16, 229–253, 2001
© 2001 Kluwer Academic Publishers. Manufactured in The Netherlands.

An Extension to GCWA and Query Evaluation for Disjunctive Deductive Databases

KEWEN WANG* kewen@haiti.cs.uni-potsdam.de
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, People's Republic of China; Institut für Informatik, Universität Potsdam, Germany

LIZHU ZHOU
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, People's Republic of China

Received April 21, 2001; Accepted May 8, 2001

Abstract. We present a simple and intuitive extension GCWA^G of the generalized closed world assumption (GCWA) from positive disjunctive deductive databases to general disjunctive deductive databases (with default negation). This semantics is defined in terms of unfounded sets and possesses an argumentation-theoretic characterization. We also provide a top-down procedure for GCWA^G, which is sound and complete with respect to GCWA^G. We investigate two query evaluation methods for GCWA^G: database partition, and database splitting. The basic idea of these methods is to divide the original deductive database into several smaller sub-databases and the query evaluation in the original database is transformed into the problem of query evaluation in smaller or simplified components. We prove that these two methods of query evaluation are all sound with respect to GCWA^G.

Keywords: disjunctive deductive databases, closed world assumption, semantics, query evaluation, argumentation

1. Introduction

In general, negative information is not explicitly represented in databases and thus a meta-rule is often employed to derive negative information from deductive databases. The advantage of this approach is that deductive databases can be liberated from excessive amounts of explicit negative information and as close to natural discourse as possible. Reiter's (1978) closed world assumption (CWA) provides such an excellent mechanism for non-disjunctive deductive databases. It is well-known that many advanced applications of deductive databases require the efficient representation and reasoning of incomplete information in the form of disjunctions in the data, e.g., diagnostic reasoning, legal reasoning and reasoning in spatial databases. As first observed by Reiter in his original CWA paper (Reiter, 1978), however, CWA becomes inconsistent for disjunctive databases and, thus, Minker (1982) proposed the generalized closed world assumption (GCWA) for inferring negative information in positive disjunctive deductive databases. This

*To whom all correspondence should be addressed.

rule has now become one of the most important nonmonotonic mechanisms in deductive databases.

On the other hand, argumentation constitutes a major component of human's intelligence, such as defending one's opinion, persuading and dialogue. It is also widely used in fields of artificial intelligence, including legal reasoning (Prakken and Sartor, 1996), diagnostic reasoning, cooperation and negotiation of multi-agents (Karacapilidi and Papadias, 1998; Kraus et al., 1998). Intuitively, the idea of argumentation is that a hypothesis (or explanation) is acceptable if it can be argued successfully against attacking arguments. Argumentation has recently proved to be a unifying mechanism for most of the existing nonmonotonic formalisms (Bondarenko et al., 1997; Dung, 1995; Kakas, 1998; Wang, 2000; Wang and Chen, 1998). In this approach, negative default literals are taken as assumptions and thus each nonmonotonic theory can be translated into an argumentation-theoretic framework. This approach provides an elegant way of deciding acceptable assumptions for given nonmonotonic theories (Bondarenko et al., 1997). The relation of argumentation to the stable semantics and the well-founded semantics is well-understood (Dung, 1995; Kakas, 1998; You et al., 2000). As we have noticed, both GCWA and argumentation can be used to nonmonotonically infer negative default literals though they are distinct mechanism and have quite different intuition behind them. What is the relationship between GCWA and argumentative reasoning? How can argumentation be performed with GCWA? These are not only interesting theoretical problems but they are also important for applications of databases and knowledge-based systems. For example, suppose that we have the following two rules:

R1: If one eats less food than necessary, he may feel hungry or be anorectic.
R2: If one has not enough food, he has to eat less food than necessary.

These two rules can be expressed as a disjunctive database consisting of two rules:

$$Hungry \vee Anorectic \leftarrow EatLess$$
$$EatLess \leftarrow \sim EnoughFood$$

If we have observed that one is hungry, the intuitive explanation for this observation is that he has not enough food to eat and is not anorectic. Since the GCWA has already been implemented, one possible and promising way of computing this kind of explanations in deductive database systems is to employ the GCWA. However, we found that the problem of performing argumentation with GCWA in (disjunctive) deductive databases are rarely explored. The deep relation of GCWA to argumentation is also unclear.

Another interesting problem is how to extend GCWA to general disjunctive databases (with default negation). It is proven that default negation can greatly enhance the expressing power of deductive databases. For example, normally, if one is not dead then he can walk. This rule is expressed in deductive databases as $walking \leftarrow \sim dead$. But its contrapositive version is not true in general, and thus this rule can not be replaced simply by the rule $walking \vee dead$. However, Minker's GCWA is defined only for positive disjunctive databases (without default negation) although argumentation is generally used to derive assumptions in nonmonotonic theories with default negation. Thus, it is an interesting problem

to investigate whether argumentation can be employed to define a suitable generalization of GCWA. Although a number of proposals have been suggested to extend GCWA for disjunctive databases with default negation, for instance, GCWA$^{\neg}$ (Sakama and Inoue, 1993), SCWA (Minker and Rajasekar, 1990), D-WFS (Brass and Dix, 1999) and the static semantics (Przymusinski, 1995). As far as we know, these extensions of GCWA are not designed to compute acceptable hypotheses, in particular, not to perform argumentation.

In this paper, by employing the notion of unfounded sets, we first present a simple and intuitive extension GCWA^G of the generalized closed world assumption (GCWA) from positive disjunctive deductive databases to general disjunctive deductive databases (with default negation). This definition allows a natural iterative procedure for GCWA^G and the computation of this semantics requires $O(n^2)$ satisfiability tests on the underlying first order theory. For instance, the satisfiability for non-disjunctive Datalog (i.e., without function symbols) is polynomial-time. This result shows an attractive feature of our semantics since it implies that there may exist wider range applications for which reasoning with GCWA^G can be practical. The suitability of our semantics is also justified by many illustrating examples. We prove that GCWA^G possesses an argumentation-theoretic characterization. This fact is meaningful in several senses since (1) it establishes a close relationship between GCWA and argumentation; (2) it convinces the suitability of our semantics from a view point of commonsense reasoning; (3) it implies that some forms of argumentation (e.g., dialogue-like argumentation) can be performed in a disjunctive database system that implements GCWA. We also provide a resolution-like procedure for GCWA^G, which is sound and complete with respect to GCWA^G. We investigate two query evaluation methods for GCWA^G: *database partition*, and *database splitting*. The basic idea of these methods is to divide the original deductive database into several smaller sub-databases and the query evaluation in the whole databases is transformed into the problem of query evaluation in smaller or simplified components. We prove that these two methods of query evaluation are all sound with respect to GCWA^G.

The rest of this paper is arranged as follows. We shall first define a generalization GCWA^G of GCWA to general disjunctive deductive databases in Section 2. In particular, a fixpoint definition for GCWA^G is given. We also illustrate the suitability of our semantics by some examples. In Section 3, an approach of performing argumentation with GCWA^G is presented by providing an argumentation-theoretic interpretation for GCWA^G. Section 4 shows that GCWA^G is really a natural extension of the GCWA. In Section 5, we provide a top-down procedure for GCWA^G and prove that this procedure is sound and complete with respect to our GCWA^G. In Section 6, we investigate a method *database partition* of query evaluation for disjunctive databases with default negation by dividing the original disjunctive database into several irrelevant sub-databases (i.e., these databases share no common atoms), which was studied previously for only positive deductive databases by Fernandez (1994), and we prove this method is sound with respect to GCWA^G. However, this method is not always effective since, in some extreme cases, the original disjunctive database can not be divided into smaller irrelevant components. Thus, in Section 7, another method of query evaluation called *database splitting* is introduced. The database splitting is still to divide the original disjunctive database into smaller sub-databases but does not require these sub-databases completely irrelevant. In practical applications, these two methods can be used

alternatively. In Section 8, we compare our approach to other related work. Section 9 is the concluding remarks. Due to the limitation of space, proofs of the theorems are not included in this paper. They are contained in the full version of this paper.

2. GCWA^G: An Extension of GCWA

In this section, we first briefly review most of the basic notions used throughout this paper and then define GCWA^G for general disjunctive databases.

2.1. Basic definitions and notation

A comprehensive treatment of disjunctive deductive databases (logic programming) is given in Lobo et al. (1992), to which the reader is referred for unexplained concepts. We assume the existence of an arbitrary, but fixed propositional language, generated from selected propositional symbols (atoms). An expression with variables is understood as an abbreviation for the set of its all grounded instances.

A *general disjunctive deductive database* (simply, *disjunctive database*) P is defined as a set of disjunctive rules of the form:

$$p_1 \vee \cdots \vee p_r \leftarrow p_{r+1}, \ldots, p_m, \sim p_{m+1}, \ldots, \sim p_n.$$

Here, $n \geq m \geq r > 0$ and p_i's are atoms for $i = 1, \ldots, n$. $\vee$ and $\sim$ denote non-classical disjunction and default negation, respectively.

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

The informal meaning of the above rule is that "if $p_{r+1}, \ldots, p_m$ are true and $p_{m+1}, \ldots, p_n$ are all not provable, then one of $p_1, \ldots, p_r$ is true". For example, $male(greg) \vee female(greg) \leftarrow animal(greg), \sim ab(greg)$ means, informally, that if *greg* is an animal and it is not provable that *greg* is abnormal, then *greg* is either male or female.

For a disjunctive rule C: $p_1 \vee \cdots \vee p_r \leftarrow p_{r+1}, \ldots, p_m, \sim p_{m+1}, \ldots, \sim p_n$, $h(C) = \{p_1, \ldots, p_r\}$, $pos(C) = \{p_{r+1}, \ldots, p_m\}$, $neg(C) = \{p_{m+1}, \ldots, p_n\}$, $atoms(C) = h(C) \cup pos(C) \cup neg(C)$. We also write $p \in h(C)$ to denote that p appears in $h(C)$.

B_P is the Herbrand base of P (the set of all atoms in P for propositional P). If $I \subseteq B_P$, then $\sim I = \{\sim q \mid q \in I\}$.

A *negative default literal* (simply, *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 $\mathbf{H}(P)$.

A *positive* (*negative*) disjunction is a disjunction of atoms (negative literals) in P. A *pure disjunction* is either a positive or negative disjunction. For a disjunctive database P, its semantics is defined by model states. In this paper, a model state M is a pair $\langle M^+;\ M^-\rangle$ where M^+ and M^- are set of positive disjunctions and set of negative disjunctions, respectively. For brevity, we assume that M contains all implications of disjunctions in it. For instance, let $M = \langle\{p\};\ \{\sim q \vee \sim w\}\rangle$, then $p \vee q$ and $\sim p \vee \sim q \vee \sim w$ are also assumed to be in M.

The body of rule C is true in M, denoted $M \models body(C)$, if $pos(C) \subseteq M^+$ and $\sim neg(C) \subseteq M^-$. Otherwise, we say the body of rule C is false in M.

2.2. *Closed world assumption and unfounded sets*

The notion of unfounded sets provides the first definition of the well-founded model (Van Gelder et al., 1988) and, more importantly, it constitutes a powerful and intuitive tool for defining semantics for deductive databases (logic programs). This notion has also been generalized to characterize stable semantics for disjunctive logic programs in Leone et al. (1997); Eiter et al. (1997). We will define the semantics GCWA^G for general disjunctive databases in terms of unfounded sets and give examples to illustrate our semantics.

The following definition is a generalization of the unfounded sets defined in Leone et al. (1997). The major difference is that we now consider model states rather than only interpretations.

Definition 2.1. Let M be a model state of disjunctive database P, a set X of ground atoms is an unfounded set for P w.r.t M if, for each $p \in X$ and each rule $C \in P$ such that $p \in h(C)$, at least one of the following conditions holds:

1. the body of C is false in M;
2. $X \cap \text{pos}(C) \neq \emptyset$, that is, some body atom belongs to X;
3. if $M \models \text{body}(C)$, then an atom in $(h(C) - X) \in M$ where $h(C) - X$ is the disjunction obtained by deleting all atoms appearing in X from $h(C)$.

Example 2.1. Let P consist of the following disjunctive rules:

$$\begin{aligned} p \vee v &\leftarrow q \\ q &\leftarrow \sim v \\ u \vee u' \vee p &\leftarrow \\ u \vee u' &\leftarrow v \\ v &\leftarrow \end{aligned}$$

Then $\{p, q\}$ is an unfounded set of P w.r.t. $M = \langle \{v, u \vee u'\}; \{\emptyset\} \rangle$.

The greatest unfounded set of P w.r.t. a model state M is denoted $\mathcal{U}_P(M)$ if it exists.

Intuitively, unfounded set specifies negative information derived from P. The following operator $\mathcal{T}_P(M)$, which is actually the immediate consequence of P under M, defines how to derive positive information from P.

Definition 2.2. Let P be a disjunctive database, the operator $\mathcal{T}_P$ is defined as, for any model state M,

$$\begin{aligned} \mathcal{T}_P(M) = \{ & a_1 \vee \cdots \vee a_r | \text{ there is a rule } C \in P : \\ & a_1 \vee \cdots \vee a_r \vee a_{r+1} \vee \cdots \vee a_s \leftarrow \text{ body}(C) \text{such that} \\ & S \models \text{ body}(C) \text{ and } \sim a_{r+1}, \ldots, \sim a_s \in M \}. \end{aligned}$$

Notice that $\mathcal{T}_P(M)$ is a set of positive disjunctions rather than just a set of atoms.

Definition 2.3. Let P be a disjunctive database, the operator $\mathcal{W}_P$ is defined as, for any model state M,

$$\mathcal{W}_P(M) = \mathcal{T}_P(M) \cup \sim\mathcal{U}_P(M).$$

Given a disjunctive database P, we can define a sequence of model states $\{W_k\}_{k\geq 0}$ where $W_0 = \emptyset$ and $W_k = \mathcal{W}_P(W_{k-1})$ for $k > 0$.

Parallel to Proposition 5.6 in Leone et al. (1997), we can prove that $\{W_k\}_{k\geq 0}$ is monotonic and its limit is the least fixpoint of $\mathcal{W}_P$.

Definition 2.4. For any disjunctive deductive database P, its GCWA^G-semantics is defined by the least fixpoint $lfp(\mathcal{W}_P)$ of $\mathcal{W}_P$.

The generalized closed world assumption of P is the set of all negative disjunctions in $lfp(\mathcal{W}_P)$ and, written as GCWA$^G(P)$.

Notice that the GCWA^G-semantics of P is determined by GCWA$^G(P)$ because the set of positive disjunctions derived from the GCWA^G-semantics of P is GCWA$^G(P)^+ = \bigcup_{i\geq 0} \mathcal{T}_P^i(\text{GCWA}^G(P))$.

Consider again Example 2.1, we have that the GCWA^G-semantics is the model state $\langle\{v, u \vee u'\}; \{\sim p, \sim q\}\rangle$ and GCWA$^G(P) = \{\sim p, \sim q\}$. Notice that u and u' are unknown.

To see this, observe that

$$\begin{aligned}
&W_0 = \emptyset,\\
&W_1 = \mathcal{W}_P(W_0) = \mathcal{T}_P(W_0) \cup \sim\mathcal{U}_P(W_0) = \{v\} \cup \emptyset = \{v\},\\
&W_2 = \mathcal{W}_P(W_1) = \mathcal{T}_P(W_1) \cup \sim\mathcal{U}_P(W_1) = \{v, u \vee u'\} \cup \{\sim q\} = \{v, u \vee u', \sim q\},\\
&W_3 = \mathcal{W}_P(W_2) = \mathcal{T}_P(W_2) \cup \sim\mathcal{U}_P(W_2) = \{v, u \vee u'\} \cup \{\sim p, \sim q\}\\
&\quad = \{v, u \vee u', \sim p, \sim q\}\\
&W_4 = W_3.
\end{aligned}$$

Similar to Theorem 5.11 in Leone et al. (1997), we can prove that GCWA^G generalizes the well-founded model for non-disjunctive deductive databases (Van Gelder et al., 1988).

Theorem 2.1. *Let P be a non-disjunctive database and $WFM(P)$ be the well-founded model of P. Then*

$$\text{GCWA}^G(P) = WFM(P).$$

Notice that, for database $P' = \{p \vee q \vee u \leftarrow;\ u \leftarrow v\}$, GCWA$^G(P') = \{\sim v\}$. Thus, rule $p \vee q \leftarrow\sim u$ is not equivalent to the rule $p \vee q \vee u \leftarrow$. That is, a suitable semantics for deductive databases should be able to distinguish these two rules. The next example adapted

from (Wang and Chen, 1998) shows the difference of GCWA^G from GDWFS (Baral et al., 1990).

Consider a variant of Poole's broken-hand example (1989).

Example 2.2 (You et al., 2000). Suppose that we know either the left hand is broken or the right hand is broken, and in general, a hand is usable if not broken. We also know the left hand being usable leads to the use of it that results in moving a block; and the use of the right hand leads to moving the table. The given information is incomplete as we do not know which hand is broken and which is not (perhaps both could have been broken).

$$\begin{aligned} \text{lhBroken} \vee \text{rhBroken} &\leftarrow \\ \text{lhUsable} &\leftarrow \sim\text{lhBroken} \\ \text{rhUsable} &\leftarrow \sim\text{rhBroken} \\ \text{moveBlock} &\leftarrow \text{lhUsable} \\ \text{moveTable} &\leftarrow \text{rhUsable} \end{aligned}$$

Now suppose we observe that the block is moved from its original location (and suppose we cannot see any operations): moveBlock $\leftarrow$. This database is denoted as P and then,

$$\text{GCWA}^G(P) = \{\sim\text{lhBroken}, \sim\text{moveTable}, \sim\text{rhUsable}\}.$$

Then GCWA^G explains our observation and predicts that it is the right hand that is broken since both moveBlock and rhBroken are derivable from GCWA$^G(P)$.

The relation of our semantics to other approaches will be further explained in Section 8.

3. Argumentation-based interpretation of GCWA^G

In this section, we shall provide a way of performing argumentative reasoning with GCWA^G by defining an argumentative interpretation for GCWA^G.

A deductive database often contains incomplete information and thus, many conclusions are made on assumption. Different hypotheses are available but some of them most probably be conflict. The main task of argumentative reasoning is to specify the set of acceptable hotheses based upon the attack relation among hypotheses. In many forms of argumentative reasoning, an acceptable hypothesis is often obtained by the following simple and intuitive principle:

A hypothesis is acceptable if it can attack any hypothesis that attacks it.

In the following, we will formulate this principle in the setting of disjunctive databases. For this, we need to define the inference relation $\vdash_P$ as follows. Roughly speaking, $\vdash_P$ is the classical resolution augmented by a special resolution rule for default negation:

$$\vdash_P \equiv \text{SLI-resolution} \cup NR.$$

The SLI-resolution (Linear Resolution with Selection function for Indefinite clauses) was developed by Minker and Zanon (1982), Lobo et al. (1992). Hayes (1971) also presents an inference system similar to SLI-resolution. It should be noted that these forms of SLI-resolution are designed only for positive disjunctive databases (i.e., without default negation). Given a disjunctive database P with default negation and a hypothesis Δ, we can easily generalize the SLI-resolution in Lobo et al. (1992) to the following rule for P:

SLI:
$\Sigma \leftarrow a, \Pi_1, \sim\Pi_2$
$a \vee \Sigma' \leftarrow \Pi'_1, \sim\Pi'_2$

$\Sigma \vee \Sigma' \leftarrow \Pi_1, \Pi'_1, \sim\Pi_2, \sim\Pi'_2$

The resolution rule NR consists of two rules:

NR1:
$\Sigma \leftarrow \Pi_1, \sim\Pi_2, \sim q$
$\sim q$

$\Sigma \leftarrow \Pi_1, \sim\Pi_2,$

NR2:
$q \vee \Sigma \leftarrow \Pi_1, \sim\Pi_2$
$\sim q$

$\Sigma \leftarrow \Pi_1, \sim\Pi_2$

NR1 is designed to resolve default literals in the body of a rule in P while NR2 is designed to resolve default literals in the head of a rule in P.

Definition 3.1. Let Δ be a hypothesis of disjunctive database P. For any disjunction α of atoms in P, $\Delta \vdash_P \alpha$ if and only if the rule $\alpha \leftarrow$ can be derived from $P \cup \Delta$ by the resolution rules SLI, NR1 and NR2. If $\Delta \vdash_P \alpha$ does not hold, it is denoted $\Delta \vdash_P \alpha$.

Consider the disjunctive database $P = \{p \leftarrow q;\ q \vee q' \leftarrow \sim p'\}$. Let $\Delta = \{\sim p', \sim q'\}$, then $\Delta \vdash_P p$.

Notice that, if $\Delta \vdash_P \alpha$, we can first apply only SLI-rules, then apply NR-rules. This special order of applying database rules will not change the result of derivation.

Definition 3.2. 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 $\sim p$.

Example 3.1. Let P be the following disjunctive databases:

$$\begin{aligned} p \vee q &\leftarrow u, \sim v \\ u &\leftarrow \sim w \\ w' &\leftarrow \sim p, \sim q \end{aligned}$$

If $\Delta_1 = \{\sim v, \sim w\}$ and $\Delta_2 = \{\sim p, \sim q\}$, then $\Delta_1 \rightsquigarrow_P \Delta_2$, but $\Delta_2 \not\rightsquigarrow_P \Delta_1$.

Definition 3.3. 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 $\sim p$.

That is, Δ supports $\sim p$ if any attacker of $\sim p$ is attacked.

Consider again the database in Example 3.1, by Definition 3.3, $\sim w'$, $\sim v$ and $\sim w$ are acceptable w.r.t. $\Delta = \{\sim v, \sim w\}$ but $\sim p$ and $\sim q$ are not.

Set $\mathbf{A}_P(\Delta) = \{\sim p : \sim p \text{ is acceptable with respect to } \Delta\}$. Then $\mathbf{A}_P$ defines an operator from $\mathbf{H}_P \rightarrow \mathbf{H}_P$. This operator is monotonic but not necessarily continuous. Hence, by Tarski's Lemma, the least fixpoint of $\mathbf{A}_P$ is the limit $\mathbf{A}_P \uparrow \gamma$ of the increasing sequence of hypotheses:

$$\emptyset,\ \mathbf{A}_P(\emptyset), \ldots, \mathbf{A}_P^n(\emptyset), \ldots$$

Notice that, if "attacker" is replaced by "minimal attacker" in Definition 3.2, we shall get an equivalent definition of $\mathbf{A}_P$. This observation will be used in Section 5.

The following theorem relates our GCWA^G to argumentation.

Theorem 3.1. *Let P be a disjunctive database. Then*

$$\text{GCWA}^G(P) = \mathbf{A}_P \uparrow \gamma,$$

for some ordinal γ.

This theorem implies that, in some cases, argumentation can be performed by GCWA^G. For positive disjunctive database, its GCWA^G has a quite simple characterization.

Proposition 3.1. *Let P be a positive disjunctive database. Then*

$$\text{GCWA}^G(P) = \mathbf{A}_P(\emptyset).$$

This proposition shows that, for a positive disjunctive database, its GCWA^G is just the set of assumptions that are acceptable with respect to the trivial hypothesis $\emptyset$. This result will also be useful in proving the results in next section.

4. Relation to the GCWA

As noticed in previous sections, GCWA is only defined for the class of positive disjunctive databases but GCWA^G is well-defined for all disjunctive databases. The main result of this section is to show that GCWA^G is really a generalization of GCWA. Before doing this, we first review the definition of GCWA.

Definition 4.1 (Minker, 1982) (Semantic definition of the GCWA). Let P be a positive disjunctive database. The generalized closed world assumption (GCWA) is given by

$$\text{GCWA}(P) = \{\sim p \mid p \in B_P \text{ and } p \text{ is not in any minimal Herbrand model of } P\}.$$

Now we can state the main result in this section.

Theorem 4.1 (GCWA^G extends *GCWA*)**.** *For any positive disjunctive database P, its GCWA semantics and* GCWA^G *coincide*:

$$\text{GCWA}^G(P) = \mathit{GCWA}(P).$$

This result is interesting because (1) GCWA^G naturally extends GCWA; (2) two quite different nonmonotonic mechanisms (the standard closed world assumption and argumentation) are related. However, most of the existing extensions of the GCWA do not support argumentation.

We can also prove that GCWA^G possesses some other major properties of GCWA: consistency and stability. The first property guarantees that no direct contradiction will be drived while the second one denotes that any negative information added by the GCWA^G will not change the positive information that can be derived from the database.

Theorem 4.2. *The following two items hold*:

1. (*Consistency*) *Let P be a disjunctive database. Then* GCWA$^G(P)$ *is consistent in the sense that there exists no assumption $\sim q \in$* GCWA$^G(P)$ *such that* GCWA$^G(P) \vdash_P q$.
2. (*Stability*) *Let P be a positive disjunctive database. Then for any disjunction Σ of atoms in P, $\emptyset \vdash_P \Sigma$ if and only if* GCWA$^G(P) \vdash_P \Sigma$.

It should be pointed out that GCWA^G is no longer stable for disjunctive databases with default negation. This is natural and desired for disjunctive databases with default negation since the default negation can not be characterized only by the semantics of the standard logic. For example, if $P = \{q \vee s \leftarrow \sim p\}$, then GCWA$^G = \{\sim p\}$. $\emptyset \vdash_P q \vee s$ does not hold while GCWA$^G \vdash_P q \vee s$.

Thus, the results in this section convince that GCWA^G really provides a suitable extension for GCWA.

5. Computation of GCWA^G

In this section, we shall provide a top-down procedure for GCWA^G and, prove this procedure is sound and complete. From now on, we assume that P is a finite propositional database.

For simplicity, we also express a (disjunctive) database rule C in P as the form of

$$\Sigma \leftarrow \Pi_1, \sim\Pi_2,$$

where Σ is a disjunction of atoms, Π_1 an unordered sequence of finite atoms denoting a conjunction of atoms, and $\sim\Pi_2 = \{\sim q : q \in \Pi_2\}$ denoting a conjunction of negative literals.

A goal G is of the form: $\leftarrow l_1, \ldots, l_r, \sim a_1, \ldots, \sim a_n$, where each l_i is an atom a or its negation $\neg a$ for $i = 1, \ldots, r$ and, a_is are atoms. To distinguish from default literals, we shall say that l is 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 each a_i is an atom and $r \leq n$.

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 unifying and simple approach.

The special goal $\leftarrow$ is called an *empty goal*. The empty goal $\leftarrow$ is also written as the familiar symbol $\Box$. 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 Section 2.

Definition 5.1. 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 $G_n = \Box$.

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 $\leftarrow \neg p_1, \ldots, \neg p_n, \sim p_{n+1}, \ldots, \sim p_m$ $(m \geq n)$ such that, for some i $(1 \leq i \leq n)$, $\leftarrow p_i$ has an SLIN-refutation in $gt(P)$.

The intuition behind the empty leaves and dead leaves is not hard to see. For instance, in the positive tree for a goal G: $\leftarrow p$, each branch represents a possible set of necessary conditions for the atom p. Whether a leaf is dead or failure, one can not make it true (a dead leaf means that at least one atom in the premises for p can not be proven; a failure leaf means that it is not a dead leaf and at least one negative literal in the premises for p is false). Note that, in addition to the three classes of leaves defined above, there may exist other leaves in a positive tree. That is, a negative leaf may not be a failure leaf. For example, if P is the deductive database $\{q \leftarrow \sim q\}$, then the positive tree T_P^+ of the goal $\leftarrow q$ is as follows. Notice that the selected literals will be underlined.

Thus, the goal $\leftarrow \sim q$ is a negative leaf of T_P^+ but it is not a failure leaf.

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 $T^+(p)$.

We now illustrate our SLIN-resolution by some examples.

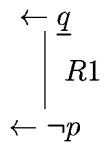
Example 5.1. First consider a positive disjunctive database P_1 as follows:

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

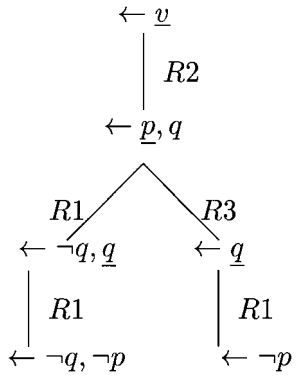
Then the rules of P_1 can be transformed into the set $gt(P_1)$ of goals:

$R1$: $\leftarrow \neg p, \neg q$
$R2$: $\leftarrow \neg v, p, q$
$R3$: $\leftarrow \neg p$

Since the goal $\leftarrow p$ has the obvious SLIN-refutation $\leftarrow p; \square$, the goal $\leftarrow q$ has a success, positive tree $T^+(q)$:



$\leftarrow v$ has the following success, positive tree $T^+(v)$:



It can also be verified that $\text{GCWA}^G(P_1) = \{\sim q, \sim v\}$ and $\text{GCWA}^G(P_1)^+ = \{p\}$.

Now we consider a disjunctive database that contains default negation.

Example 5.2. Let P_2 consist of the following rules:

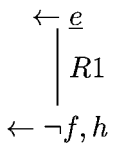
$$
\begin{aligned}
a \vee b &\leftarrow c, \sim d \\
b &\leftarrow \sim e, \sim c \\
e \vee f &\leftarrow h \\
a \vee f &\leftarrow c
\end{aligned}
$$

$gt(P_2)$ is as follows:

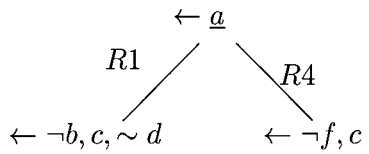
$$
\begin{aligned}
&R1\text{:} \leftarrow \neg a, \neg b, c, \sim d \\
&R2\text{:} \leftarrow \neg b, \sim e, \sim c \\
&R3\text{:} \leftarrow \neg e, \neg f, h \\
&R4\text{:} \leftarrow \neg a, \neg f, c
\end{aligned}
$$

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 all dead:



It is not hard to see that $\sim a \in \text{GCWA}^G(P_2)$ and $\text{GCWA}^G(P_2) \vdash_{P_2} b$.

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

Theorem 5.1 (Soundness and Completeness of SLIN-resolution)**.** *Let P be a disjunctive database and $p \in B_P$ (i.e., p is an atom in P).*

1. $\sim p \in \text{GCWA}^G(P)$ *if and only if there exists a success positive tree for the goal* G: $\leftarrow p$.
2. $\text{GCWA}^G(P) \vdash_P p$ *if and only if there exists an SLIN-refutation for the goal* G: $\leftarrow p$.

For SLIN-resolution, the disjunctive database with default negation can be preprocessed by the fixpoint transformation Lft introduced in Wang and Chen (1998). Lft transforms each disjunctive deductive database P into a negative database $Lft(P)$ (i.e., without atoms in bodies of the rules of P). Moreover, we can prove that $\text{GCWA}^G(P) = \text{GCWA}^G(Lft(P))$ Notice that the query evaluation under SLIN-resolution procedure for negative database is an easy task since the Goal Resolution will never be used in this case. According to the above theorem, we can first transform the given deductive database P into the negative database $Lft(P)$ and then use SLIN-resolution to answer query against $Lft(P)$. For the limitation of space, we will not go into details here.

6. Evaluating query by partition

Although disjunctive deductive databases greatly enhance the expressive ability of ordinary deductive databases, it is well-known that query evaluation in disjunctive deductive databases is in general computationally hard. When a database is large, given a query Q, it is often the case that most of the rules in the database are irrelevant to Q and thus, the query evaluation of Q can be restricted to the sub-database consisting of those rules that are relevant to Q. Two sets of rules are *irrelevant* if they have no common atoms. Otherwise, they are *relevant*. In this and the next sections, we will provide some intuitive ways to simplify the computation of GCWA^G.

Database partitioning means that a relatively large database P is divided into a collection of sub-databases or clusters (Fernandez, 1994; Yahya and Minker, 1994). This method can simplify the query answering process by transforming the query evaluation in the original database into that in a cluster. Since the storage requirements for the clausal representation of a database is the same as that for all of its components, database partitioning is economical for simplifying query evaluation in large disjunctive databases.

Most of the definitions in this section are generalizations of their counterparts for positive disjunctive databases in Fernandez (1994) and Yahya and Minker (1994).

Definition 6.1. A subset S of a disjunctive database P is a cluster of P if $C \in S$ for any rule C in P such that $\text{atoms}(C) \cap \text{atoms}(S) \neq \emptyset$, where $\text{atoms}(E)$ is the set of all atoms appearing in an expression E.

A cluster S is minimal if there is no cluster S' such that $S' \subset S$.

Informally, a cluster of P is a maximal collection of rules in P that are relevant.

For any disjunctive database P, we associate a graph $G(P)$ whose nodes are rules of P and whose edges are the set $\{\langle C_1, C_2\rangle : \mathit{atoms}(C_1) \cap \mathit{atoms}(C_2) \neq \emptyset\}$. Then a minimal cluster S of P just corresponds to a maximal connected subgraph of $G(P)$ and, a cluster S corresponds to a union of some maximal connected subgraph of $G(P)$. Thus, the task of constructing a cluster or minimal cluster for disjunctive database P is reduced to that of finding maximal connected subgraphs of $G(P)$.

Definition 6.2. Let $P_1, P_2, \ldots, P_n$ be clusters of disjunctive database P. The collection $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ is a partition of P if the following two conditions are satisfied:

1. $P_i \cap P_j = \emptyset$ for any $i \neq j$, $1 \leq i, j \leq n$;
2. $P = P_1 \cup P_2 \cdots \cup P_n$.

We do not require that P_i is minimal in the above definition.

Example 6.1. Let P consist of the following rules:

$$
\begin{aligned}
C_1&: p \vee q \leftarrow \sim u\\
C_2&: \quad u \leftarrow v, \sim p\\
C_3&: \quad w \leftarrow \sim w
\end{aligned}
$$

It is trivial that P itself is a cluster. Two nontrivial clusters of P are $P_1 = \{C_1, C_2\}$ and $P_2 = \{C_3\}$. P_1 and P_2 are also minimal clusters. Moreover, $\mathcal{P} = \{P_1, P_2\}$ is partition of P.

Theorem 6.1. *Let $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ be a partition of disjunctive database P. Then, for any atom $p \in B_P$,*

1. $\sim p \in \text{GCWA}^G(P)$ *if and only if* $\sim p \in \text{GCWA}^G(P_i)$ *for all* $i = 1, \ldots, n$.
2. $\text{GCWA}^G(P) \vdash_P p$ *if and only if* $\text{GCWA}^G(P_i) \vdash_P p$ *for some* i, $1 \leq i \leq n$.

This theorem translates the task of answering query Q in disjunctive database P into that of answering query Q in components if a partition of P is given. If $p \in B_{P_{i_0}}$, then $\sim p \in \text{GCWA}^G(P_j)$ for any $j \neq i_0$. That is, the truth or falsity of p in P can be decided in P_{i_0}. Therefore, we have the following useful corollary.

Corollary 6.1. *Let $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ be a partition of disjunctive database P and p be an atom that appears in P_{i_0}. Then p is true (or false) in P if and only if p is true (or false) in P_{i_0}.*

Example 6.2. Consider the disjunctive database P in Example 6.1. For query $Q = \sim u$, we have $\sim u \in \text{GCWA}^G(P)$ since $\sim u \in \text{GCWA}^G(P_1)$.

Notice that this example also shows that the partition method is not applicable to query answering of the stable semantics (Przymusinski, 1991) for disjunctive databases. For instance, in Example 6.1, $\sim v$ is true in P_1 under the stable semantics but the stable semantics of P is undefined since P has no stable model.

Given a database partition, the sizes of its components are often smaller than the original database and thus, query answering in a component is more efficient than in the entire database. Query evaluation by database partition makes it possible for us to use the same reasoning but to limit the total search effort. Ordering of partition components also provides us some possible ways of optimizing query answering, for example, we can carry out query evaluation by first starting with those partition components that have better chances of producing answer fast. Further, such ordering of partition components can be based on their sizes, the results of previous evaluations or any other relevant parameters. In particular, in some cases, the components of a database partition are significantly smaller than the entire database and therefore, the method of partitioning databases for query evaluation will be more powerful.

The partition technique also makes some kind of parallel query answering possible since the separate query evaluations in clusters can be performed in parallel. In the worst cases, the entire database has only one cluster, that is, the database has no non-trivial partition. Thus, the partition technique will not work for these databases. In the next section, we shall provide another method of query evaluation for disjunctive databases.

7. Evaluating query by splitting

As pointed out at the end of Section 6, a major drawback of query evaluation by database partition is that it may not work well when in cases the sizes of components of a partition are large. The main reason is that the notion of partitioning is a little strong (it requires that there be no connection between two clusters. In many applications, most of the rules in databases are relevant and thus, we cannot expect the databases always possess such partitions. In this section, we present another method of query evaluation by weakening the requirement in the definition of clusters. Similar to the database partition technique, we shall also divide a given database into smaller components but among these components there may exist some kind of dependent relations. For any two such components P_1 and P_2,

we allow one component to depend upon another but they cannot depend upon each other. This method can be used to design a query database in a hierarchical fashion.

Example 7.1. We consider a variant of the example in Eiter et al. (1997) that expresses the 3-colorability of a graph:

$$
\begin{aligned}
R1\text{: } & \mathrm{Red}(x) \vee \mathrm{Green}(x) \vee \mathrm{Blue}(x) \leftarrow \\
R2\text{: } & \mathrm{Notcolored} \leftarrow E(x, y),\ \mathrm{Red}(x),\ \mathrm{Red}(y) \\
R3\text{: } & \mathrm{Notcolored} \leftarrow E(x, y),\ \mathrm{Green}(x),\ \mathrm{Green}(y) \\
R4\text{: } & \mathrm{Notcolored} \leftarrow E(x, y),\ \mathrm{Blue}(x),\ \mathrm{Blue}(y) \\
R5\text{: } & \mathrm{Colored} \leftarrow {\sim}\mathrm{Notcolored}
\end{aligned}
$$

For this database, we may take P_1 as either $\{R1\}$ or $\{R1, R2, R3, R4\}$.

This idea is not quite a new one in deductive databases and logic programming since this is just the idea of defining the stratified logic programs (Apt et al., 1988). It is also exploited by Przymusinski (1988) to define the locally stratified programs, by Schlipf (1992) to define the notion of stratified pairs, by Dix (1992) to define "relevance" and "modularity" of logic programs. Lifschitz and Turner (1994) also use this idea to study the computation of answer sets of a logic program. More recently, Eiter et al. (1997) consider the problem of using this method to evaluate disjunctive Datalog under the stable semantics. To distinguish this method from database partition in the last section, we name the method of dividing databases in this section as *database splitting*. This terminology "splitting" is borrowed from Lifschitz and Turner (1994). In a certain sense, these discussions are all concerned with only two-valued models. However, our GCWA^G is essentially a three-valued semantics and thus, the situation becomes a little different and difficult as we shall see later on. The next definition formulates the idea above in the setting of disjunctive databases.

Definition 7.1. A sub-database S of a disjunctive database P is a splitting sub-database of P if $C \in S$ for any rule C of P such that $h(C) \cap \mathrm{atoms}(S) \neq \emptyset$. Here we abuse $h(C)$ to denote $\mathrm{atoms}(h(C))$ for simplicity.

The condition in Definition 7.1 guarantees that the answering for query in the Herbrand base B_S depends upon only the rules in S. The splitting sub-database S corresponds to the bottom stratification in stratified logic program and the minus set $T = P - S$ is just an upper stratification.

The existence of a splitting set also relates to the notion of modularity of programs. As shown in the example above, following the guess and check paradigm, we can first design a basic module P_1 that generates a set of possible solution candidates, and a module P_2 that takes the output of P_1 (the possible solutions) and eliminates from them those which violate the given criteria.

In the example above, both $\{R1\}$ or $\{R1, R2, R3, R4\}$ are splitting sub-databases of the disjunctive database.

For a 2-valued semantics, the splitting technique is relatively simple: given a splitting set S of a deductive database P, we can first evaluate S (say, we have a model M_0 for S) and then

we shall obtain models of P by evaluating only a simpler database $P' = (P - S) \cup \{p \leftarrow : p \in M_0\}$.

Example 7.2. Let P be the deductive database:

$R1$: $p \leftarrow u$
$R2$: $q \leftarrow \sim u$
$R3$: $v \leftarrow \sim u$

Then $S = \{R3\}$ is a splitting sub-database of P and $M_0 = \{v\}$ is a stable model of S. Then $P' = \{R1; R2; v \leftarrow\}$ and P' has the stable model $M = \{q, v\}$.

However, our semantics GCWA^G is essentially 3-valued and we are concerned with only inferring negative information, the situation becomes a little different in defining the partner of P' above.

Let P^S denote the disjunctive database simplified from P by the following three steps:

1. delete all rules in P whose bodies contain at least one atom p such that $\sim p \in$ GCWA$^G(S)$;
2. for any remaining rule, delete all atoms in the head whose complement literals are in GCWA$^G(S)$, delete all negative literals in the body that are in GCWA$^G(S)$;
3. delete all rules that have empty heads in the rules obtained by step 2.

Notice that P^S is similar to the GL-transformation (Gelfond and Lifschitz, 1988), but heads of the rules are further simplified here. If S is a splitting sub-database of disjunctive database P, then S is regarded as a disjunctive database on B_S and P^S as a disjunctive database on $B_P - atoms(\text{GCWA}^G(S))$.

Theorem 7.1. *Let S be a splitting sub-database of disjunctive database P and P^S is defined as above. Then*

$$\text{GCWA}^G(P) = \text{GCWA}^G(S) \cup \text{GCWA}^G(P^S).$$

This theorem suggests another method of optimizing query evaluation in some disjunctive databases: Given a splitting sub-database S of P, if a query is in S, we need only answer it against S; if a query is not in S and S is evaluated in advance, then the query answering can be performed in a simplified database P^S.

Example 7.3. Let P be the deductive database:

$R1$: $p \vee q \leftarrow$
$R2$: $p \leftarrow$
$R3$: $w \leftarrow \sim v$
$R4$: $v \leftarrow \sim p$

$S = \{R1, R2\}$ is a splitting sub-database of P and $\text{GCWA}^G(S) = \{\sim q\}$. Then P^S is the following disjunctive database:

$$p \leftarrow$$
$$w \leftarrow \sim v$$
$$v \leftarrow \sim p$$

We have that $\text{GCWA}^G(P^S) = \{\sim v\}$. Therefore, $\text{GCWA}^G(P) = \{\sim q, \sim v\}$.

The problem of constructing a splitting sub-database for a disjunctive database P can also be reduced to that of finding certain subgraph of a graph associated with P. Given a disjunctive database P, we associate a directed graph $DG(P)$, whose nodes are all rules of P and whose edges are the set $\{\langle C_1, C_2\rangle : h(C_1) \cap atoms(C_2) \neq \emptyset\}$.

Let S be a sub-database of disjunctive database P and $C_1 \in P - S$. If there is a rule C_2 in S such that the ordered pair $\langle C_1, C_2\rangle$ is an edge of $DG(P)$, then we say that C_1 is an unfounded node of $DG(S)$. From Definition 7.1, we have the following graph-based characterization for splitting sub-databases.

Proposition 7.1. *Let P be a disjunctive database and S a sub-database of P. Then S is a splitting sub-database of P if and only if the subgraph $DG(S)$ has no unfounded nodes in $DG(P)$.*

In general, more than one splitting sub-databases for a disjunctive database can be constructed, and thus we can generalize Theorem 7.1 to monotone sequences of splitting sub-databases.

Definition 7.2. A splitting sequence of a disjunctive database P is a sequence of splitting sub-databases of $P : P_1, P_2, \ldots, P_n$ such that $P_i \subset P_{i+1}$ for $i = 1, \ldots, n-1$.

Notice that P_i is also a splitting sub-database of P_{i+1} in Definition 7.2

By Theorem 7.1, it is an easy induction on the length of the splitting sequence to prove the following theorem.

Theorem 7.2. *Let $P_1, P_2, \ldots, P_n$ be splitting sequence of a disjunctive database P. Set $P_{n+1} = P$ and $P_0 = \emptyset$. Then*

$$\text{GCWA}^G(P) = \bigcup_{i=1}^{n+1} \text{GCWA}^G\left(P_i^{P_{i-1}}\right).$$

Therefore, if we have a splitting sequence for disjunctive database P, then the evaluation of P can be performed on smaller sub-databases sequentially. In general, this method seems not so efficient as the method of partitioning databases and also can not be carried out in parallel. However, it can be applied to a wider class of disjunctive databases than database partition.

8. Comparison to related work

The GCWA is initially introduced only for positive disjunctive databases (without default negation). It can, in fact, be seen as a form of skeptical reasoning and is in particular interesting to the community of (deductive) databases. Extensions of GCWA often fall into three main categories:

1. Extensions that are only interested in deriving negative information: This is a major feature of GCWA and our GCWA^G can be suitably put in this category. Some other related approaches in this category include SCWAS (Rajasekar and Minker, 1990) for stratified disjunctive databases, and GCWA$^\neg$ (Sakama and Inoue, 1993). Based upon the stable semantics (Gelfond and Lifschitz, 1988; Przymusinski, 1991), Sakama and Inoue (1993) provide a closed word reasoning form GCWA$^\neg$ for general disjunctive databases and GCWA$^\neg$ extends the GCWA. Unfortunately, this form of non-monotonic reasoning bears the same drawback as the stable semantics: GCWA$^\neg$ for some deductive databases may be undefined. We consider the Barber's Paradox example in You et al., (2000), which is a modification of the deductive database in Dung (1995).

Example 8.1.

$$\begin{aligned} \text{shave}(\text{bob}, x) &\leftarrow \sim \text{shave}(x, x) \\ \text{payCash}(y, x) \vee \text{payByCredit}(y, x) &\leftarrow \text{shave}(x, y) \\ \text{accepted}(x, y) &\leftarrow \text{payCash}(x, y) \\ \text{accepted}(x, y) &\leftarrow \text{payByCredit}(x, y) \end{aligned}$$

The first rule of this database says that bob shaves those who do not shave themselves; the second rule says that y may pay x by cash or credit; the last two rules says that either way of paying is accepted.

We further assume that greg is the mayor (another person different from bob). Then GCWA$^\neg$ is undefined for this disjunctive database since it has no stable models in this case. However, under the GCWA^G, we can infer ∼shave(greg, greg), shave(bob, greg and accepted(greg, bob).

2. Extensions that are interested in deriving both negative and positive information, but not interested in deriving sentences: Most generalizations of the well-founded semantics (Van Gelder et al., 1988) fall into this category, such as the generalized disjunctive well-founded semantics (GDWFS) (Baral et al., 1990), the static semantics (Przymusinski, 1995), the stationary semantics (i. e. partial stable models) (Przymusinski, 1991), WFDS (Wang, 2000) and D-WFS (Brass and Dix, 1999). Let us consider the following example, which has been used by many authors to show the suitability of their semantics.

Example 8.2. Assume that our database P consists of three rules:

$$\begin{aligned} p \vee q &\leftarrow \\ w &\leftarrow \sim p \\ w &\leftarrow \sim q \end{aligned}$$

Since $\text{GCWA}^G(P) = \emptyset$, w can not be inferred from P under GCWA^G. Some other approaches including D-WFS and GDWFS also do not allow w to be inferred. But under $\text{GCWA}^\neg$, stationary semantics and static semantics, w can be derived from the above P.

As an abstract program, we cannot say whether or not w should be inferred since we can enumerate many applications for which w should not be concluded as well as applications for which w should be concluded. Moreover, this problem has no much relation to whether a nonmonotonic semantics is skeptical. The real secret is the ways of interpreting the nonclassical disjunction '$\vee$'. For instance, the interpretation of the disjunctions *walking* $\vee$ *singing* and *working* $\vee$ *tired* is *inclusive*. The interpretation of the disjunctions *black* $\vee$ *white* and *up* $\vee$ *down* is *exclusive*.

In general, if the interpretation of '$\vee$' is inclusive, then w should not be concluded from the above P. Otherwise, w should be concluded.

Example 8.3. Suppose that we have an incomplete knowledge base *KB* containing three rules:

(1) If one is not excellent in research, he will be fired.
(2) If one is not excellent in teaching, he will be fired.
(3) We only know the fact that John is excellent at least in one of research and teaching.

Now, we ask: will John be fired? Intuitively, the correct answer should be unknown. That is, one can neither say that John will be fired nor say that John will not be fired, since the knowledge at hand is not enough to make a prediction about John's tenure status.

Let $t =$ ExcellentInTeaching, $r =$ ExcellentInResearch and $f =$ Fired, then this knowledge base *KB* can be expressed as the following disjunctive database P:

$$\begin{aligned} t \mid r &\leftarrow \\ f &\leftarrow \sim t \\ f &\leftarrow \sim r \end{aligned}$$

Under GCWA^G and D-WFS, f can not be inferred from *KB*. Thus, GCWA^G and D-WFS are the right semantics for this application. But the stationary semantics, static semantics and $\text{GCWA}^\neg$ are not.

GDWFS is defined through fixpoints but it seems a little complicated and less intuitive. In particular, this semantics interprets every disjunctive database into a positive one. For example, under GDWFS, the database P in Example 8.2 is equivalent to the positive database $P' = \{p \vee q \leftarrow;\ w \vee p \leftarrow; w \vee q \leftarrow\}$. Thus, it is not our intuition on P.

The above examples illustrate that GCWA^G is different from the existing extensions of GCWA and is the correct semantics for applications that interpret the disjunction inclusively.

Brass and Dix's D-WFS is relatively a new one, and has a quite different intuition from our GCWA^G. D-WFS is defined through a series of abstract semantic properties and does not intend to perform argumentation. In addition, GCWA^G also infers different literals from D-WFS as the following example shows.[1]

Example 8.4. According to the ranking of an European journal, the 20 top riches are all from four places: North America, Europe, Hong Kong and Japan. Assume that we say a person is rich if he/she is among the 20 ones. That is, if one is not from any of the above four places, then he/she has less money. Moreover, we know that Tom is from one of the four places (but we do not know exactly which). This knowledge base can be expressed as the following disjunctive deductive database P:

$$\begin{aligned} \text{lessMoney}(x) &\leftarrow \sim\text{eu}(x), \sim\text{na}(x), \sim\text{hk}(x), \sim\text{jp}(x) \\ \text{eu(Tom)} \vee \text{na(Tom)} \vee \text{hk(Tom)} \vee \text{jp(Tom)} &\leftarrow \end{aligned}$$

Intuitively, we can not say that Tom is rich or not since there are many persons in the above four places who are not rich according to the criteria above. That is, neither lessMoney(Tom) (he is not rich) nor $\sim$ lessMoney(Tom) (he is rich) can be inferred from P. It can be verified that $\text{GCWA}^G(P) = \emptyset$ and thus, $\text{GCWA}^G(P) \nvdash_P \text{lessMoney(Tom)}$ and $\sim$lessMoney(Tom) $\notin \text{GCWA}^G(P)$.

This is exactly our intuition on P. Under D-WFS, $\sim$lessMoney(Tom) is derivable from P.

Notice that some of the approaches in the first category can also be put in this category by including the positive information they derive. For example, given any disjunctive databases P, our GCWA^G is, in fact, a generalization of the well-founded semantics.

3. Extensions that are interested in arbitrary sentences: this line has been pursued by many researchers, such as Gelfond et al. (1989), Lifschitz (1985), and Lifschitz (1995). In these efforts, a knowledge base consists of arbitrary formulas and different circumscription policies are applied. But Minker restricts his attention to universal theories and their Herbrand models. Another feature of Minker's definition is that only one minimality of models is used. For this reason, Minker's GCWA is more simple than the circumscription-based approaches and may be easier to be implemented in deductive databases. But Minker's GCWA does not possess so strong expressive power as McCarthy's approach. Although both McCarthy's circumscription (McCarthy, 1980) and Minker's GCWA are based on minimal models, the deep relation between their uses of minimal models are quite subtle. Approaches in this category are less relevant to our GCWA^G, so we shall not discuss them in detail. Further discussions can be found in some previous literatures (for instance, Lifschitz, 1985, 1995).

9. Conclusion

Our aim in this paper was to understand the relation of the closed world assumption to argumentation (abduction) in disjunctive deductive databases. To do so, we first extend the well known GCWA to the GCWA^G for general disjunctive databases (with default negation) and a novel argumentation-based interpretation for GCWA^G is provided. Some interesting properties of GCWA^G are shown and these properties convince GCWA^G is really a suitable and natural extension of GCWA. We provide a complete and sound procedure for GCWA^G. It is well known that the query evaluation of disjunctive databases is a hard problem. We also present two methods (database partitioning and database splitting) to optimize the process of query evaluation. In particular, these two methods can be combined

in practical applications. For example, given a deductive database P, we can first partition it into clusters and then split each cluster. This combination of partitioning and splitting exploits the advantages of both two methods. As indicated in the previous sections, GCWA^G is a promising nonmonotonic mechanism in deductive databases for its simplicity and expressive power. An important direction of further research is to investigate the application of GCWA^G in commonsense reasoning. As the first step, the relation of GCWA^G to other formalisms of nonmonotonic reasoning should be clarified. Recently, Peter Baumgartner at Koblenz University worked out a prototype implementation of GCWA^G in ECLIPSE prolog, which is a restart model elimination prover (Baumgartner et al., 1997) augmented by default negation. It is possible to design a PTTP-based prover, or, to extend the PROTEIN prover. Thus, the efficient implementation of our semantics in disjunctive databases is also another issue to be further explored. GCWA^G is polynomial-time for the class of finite non-disjunctive propositional databases because GCWA^G coincides with the well-founded model. There may also exist some other classes of disjunctive databases that are both moderately expressive and possess efficient procedures. Our SLIN-resolution is designed only for query answering of propositional disjunctive databases. Another topic to be further pursued is how to extend this procedure to deal with query-answering at the first order level.

Acknowledgments

We would like to thank the anonymous referees for helpful comments on this paper. This work was supported in part by the Natural Science Foundation of China (No. 69883008, No. 69773027), the NKBRSF of China (No. G61999032704), and the German Science Foundation (DFG) within Project "Nichtmonotone Inferenzsysteme zur Verarbeitung konfligierender Regeln" under grant FOR 375/1-1, TP C.

Note

1. The difference of GCWA^G from D-WFS is pointed out by Jürgen Dix

References

Apt, K., Blair, H., and Walker, A. (1988). Towards a Theory of Declarative Knowledge. In *Foundations of Deductive Databases and Logic Programming* (pp. 89–148).

Brass, S. and Dix, J. (1999). Semantics of Disjunctive Logic Programs Based on Partial Evaluation, *Journal of Logic Programming*, 38(3), 167–312.

Baumgartner, P., Furbach, U., and Stolzenburg, F. (1997). Computing Answers with Model Elimination, *Artificial Intelligence*, 90(1–2), 135–176.

Bondarenko, A., Dung, P. M., Kowalski, R., and Toni, F. (1997). An Abstract, Argumentation-theoretic Framework for Default Reasoning, *Artificial Intelligence*, 93(1–2), 63–101.

Baral, C., Lobo, J., and Minker, J. (1990). Generalized Disjunctive Well-founded Semantics for Logic Programs: Declarative Semantics. In *Proceedings of the 5th International Symposium on Methodologies for Intelligent Systems*, Knoxville, TN (pp. 465–473).

Dix, J. (1992). Classifying Semantics of Disjunctive Logic Programs (extended abstract). In *Proc. Joint International Conference and Symposium on Logic Programming* (pp. 798–812).

Dung, P. (1995). An Argumentation-theoretic Foundation for Logic Programming, *J. Logic Programming*, 24, 151–177.

Eiter, T., Leone, N., and Sacca, D. (1997). On the Partial Semantics for Disjunctive Deductive Databases, *Annals of Math. and AI.*, 19(1–2), 59–96.

Eiter, T., Gottlob, G., and Mannila, H. (1997). Disjunctive Datalog, *ACM Transaction on Database Systems*, 22(3), 364–418.

Fernandez, J. (1994). Disjunctive Deductive Databases, PhD Thesis, University of Maryland.

Gelfond, M. and Lifschitz, V. (1988). The Stable Model Semantics for Logic Programming. In R. Kowalski and K. Bowen (Eds.), *Logic Programming: Proc. of the Fifth International Conference and Symposium* (pp. 1070–1080).

Gelfond, M., Przymusinska, H., and Przymusinski, T. (1989). On the Relationship Between Circumscription and Negation as Failure, *Artificial Intelligence*, 38(1), 75–94.

Hayes, P. (1971). Semantic Trees. PhD Thesis, Edinburgh University.

Kakas, A., Kowalski, R., and Toni, F. (1998). The Role of Abduction in Logic Programming. In D.M. Gabbay, C.J. Hogger and J.A. Robinson (Eds.), *Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 5* (pp. 235–324). Oxford, UK: Oxford University Press.

Karacapilidis, N. and Papadias, D. (1998). A Computational Approach for Argumentative Discourse in Multi-agent Decision Making Environments, *AI Communications*, 11(1), 21–33.

Kraus, S., Sycara, K., and Evenchik, A. (1998). Reaching Agreements through Argumentation: A logical Model and Implementation. *Artificial Intelligence*, 104, 1–69.

Leone, N., Rullo, P., and Scarcello, F. (1997). Disjunctive Stable Models: Unfounded Sets, Fixpoint Semantics, and Computation, *Information and Computation*, 135(2), 69–112.

Lifschitz, V. (1985). Closed-world Data bases and Circumscription, *Artificial Intelligence*, 27(2), 229–235.

Lifschitz, V. (1995). ECWA Made Easy, *Annals of Mathematics and Artificial Intelligence*, 14, 269–274.

Lobo, J., Minker, J., and Rajasekar, A. (1992). *Foundations of Disjunctive Logic Programming*, Cambridge, MA: MIT Press.

Lifschitz, V. and Turner, H. (1993). Splitting a Logic Program. In *Proc. of International Conference on Logic Programming* (pp. 567–585).

McCarthy, J. (1980). Circumscription—A form of Non-monotonic Reasoning, *Artificial Intelligence*, 13, 27–39.

Minker, J. (1982). On Indefinite Databases and the Closed World Assumption. In *LNCS 138*, pp. 292–308.

Minker, J. and Rajasekar, A. (1990). A Fixpoint Semantics for Disjunctive Logic Programs, *Journal of Logic Programming*, 9(1), 45–74.

Minker, J. and Zanon, G. (1982). An Extension to Linear Resolution with Selection Function, *Information Processing Letters*, 14(3), 191–194.

Poole, D. (1989). What the Lottery Tells us About Default Reasoning. In *Proc. of the 1st International Conference on Principles of Knowledge Representation and Reasoning*, Toronto, Canada (pp. 333–340).

Przymusinski, T. (1988). On the Declarative Semantics of Deductive Databases and Logic Programs. In *Foundations of Deductive Databases and Logic Programming* (pp. 193–216).

Przymusinski, T. (1991). Semantic of Disjuctive Logic Programs and Deductive Databases. In C. Delobel, M. Kifer, and Y. Masunaga (Eds.), *Proceedings of the Second International Conference on Deductive and Object-Oriented Databases* (*DOOD'91*) (pp. 85–107). Germany: Springer Verlag.

Przymusinski, T. (1991). Stable Semantics for Disjunctive Programs, *New Generation Computing*, 9, 401–424.

Przymusinski, T. (1995). Static Semantics for Normal and Disjunctive Logic Programs. *Annals of Mathematics and Artificial Intelligence*, 14, 323–357.

Prakken, H. and Sartor, G. (1996). Special Issue on Logical Models of Argumentation, *Artificial Intelligence and Law Journal*, 4(3–4).

Rajasekar, A. and Minker, J. (1990). On Stratified Disjunctive Programs, *Annals of Mathematics and Artificial Intelligence*, 1(1–4), 339–357.

Reiter, R. (1978). On the Closed World Databases. In H. Gallaire and J. Minke (Eds.), *Logic and Data Bases*. (pp. 119-140). New York: Plenum Press.

Schlipf, J. (1992). Formalizing a Logic for Logic Programming, *Annals of Mathematics and Artificial Intelligence*, 5, 279–302.

Sakama, C. and Inoue, K. (1993). Negation in Disjunctive Logic Programs. In *Proceedings of the 10th International Conference on Logic Programming* (ICLP'93) (pp. 703–719). Cambridge, MA: MIT Press.

Van Gelder, A., Ross, K., and Schlipf, J. (1988). Unfounded Sets and Well-founded Semantics for General Logic Programs. In *Proceedings of the 7th ACM Symposium on Principles Of Database Systems*. (pp. 221–230). Full version in *J. ACM*, 38, 620–650, 1992.

Wang, K. (2000). Argumentation-based Abduction in Disjunctive Logic Programming, *Journal of Logic Programming*, 45(1–3), 105–140.

Wang, K. and Chen, H. (1998). Abduction, Argumentation and Bi-disjunctive Logic Programs. In J. Dix, L. Pereira and T. Przymusinski (Eds.), *Logic Programming and Knowledge Representation* (*LNAI 1471*), Proc. LPKR'97 (pp. 139–163).

Yahya, A. and Minker, J. (1994). Query Evaluation in Partitioned Disjunctive Deductive Databases, *International Journal of Intelligent and Cooperative Systems*, 3(4), 385–413.

You, J., Yuan, L., and Gobel, R. (2000). Abductive Logic Programming with Disjunctive Logic Programs, *Journal of Logic Programming*, 44(1–3), 101–127.