NORTH-HOLLAND

PARTIAL DEDUCTION IN DISJUNCTIVE LOGIC PROGRAMMING

CHIAKI SAKAMA AND HIROHISA SEKI

▷ This paper presents a partial deduction method in disjunctive logic programming. Partial deduction in normal logic programs is based on unfolding between normal clauses, hence it is not applicable to disjunctive logic programs in general. Then we introduce a new partial deduction technique, called disjunctive partial deduction, which preserves the minimal model semantics of positive disjunctive programs and the stable model semantics of normal disjunctive programs. From the procedural side, disjunctive partial deduction is combined with a bottom-up proof procedure of disjunctive logic programs, and top-down partial deduction is introduced for query optimization. Disjunctive partial deduction is also applied to optimizing abductive logic programs and compiling propositional disjunctive programs. © Elsevier Science Inc., 1997 ◁

1. INTRODUCTION

Partial deduction or *partial evaluation* is known as one of the optimization techniques in logic programming. Given a logic program, partial deduction derives a more specific program through performing deduction on a part of the program, while preserving the meaning of the original program. Such a specialized program is usually more efficient than the original program when executed.

Partial deduction in logic programming was first introduced by Komorowski [10] and has been investigated by several researchers in various aspects. (For an introduction and bibliographies, see [11] and [27], for example.) From the semantic viewpoints, Lloyd and Shepherdson [12] formalized partial evaluation for normal logic programs and provided conditions to assure the correctness with respect to Clark's program completion semantics. In the context of the unfold/fold transfor-

Address correspondence to Chiaki Sakama, Department of Computer and Communication Sciences, Wakayama University, Sakaedani, Wakayama 640, Japan, E-mail: sakama@sys.wakayama-u.ac.jp.
Received February 1995; accepted August 1996.

655 Avenue of the Americas, New York, NY 10010
0743-1066/97/$17.00
PII S0743-1066(96)00120-3

mation of logic programs, Tamaki and Sato [28] showed that partial deduction preserves the least Herbrand model semantics of definite logic programs. The result was extended to the perfect model semantics for stratified logic programs [14, 25], and the stable model semantics and the well-founded semantics for normal logic programs [24, 26].

Recent studies of logic programming extended its framework to include indefinite information in a program. *Disjunctive logic programming* [13] is such an extension of logic programming, and due to its expressiveness, it has been given increasing attention over the past few years. However, the expressiveness of disjunctive logic programming implies the difficulty of realizing efficient procedures. In fact, it is known that the computation of disjunctive logic programs is generally harder than that of normal logic programs [4]. In order to make disjunctive logic programming practical, it is necessary to develop optimization techniques such as partial deduction for disjunctive logic programs. Partial deduction in normal logic programs is based on unfolding between normal clauses, so that some extension is needed to apply the technique to disjunctive logic programs.

In this paper, we develop partial deduction techniques for disjunctive logic programming. We first extend the unfolding operation in normal logic programs to the one which supplies unfolding between disjunctive clauses. Then we introduce a new partial deduction method, called *disjunctive partial deduction*, for disjunctive logic programs. We prove that disjunctive partial deduction preserves the minimal model semantics of positive disjunctive programs, and the stable model semantics of normal disjunctive programs. On the procedural side, disjunctive partial deduction is combined with a bottom-up proof procedure for disjunctive logic programs, and top-down partial deduction is introduced for query optimization. Disjunctive partial deduction is also applied to optimizing abductive logic programs and compiling propositional disjunctive programs.

The rest of this paper is organized as follows. In Section 2, we provide basic terminologies which shall be used in this paper. In Section 3, we introduce disjunctive partial deduction for positive disjunctive programs, and show that it preserves the minimal model semantics. Section 4 extends the result to normal disjunctive programs and shows that the stable model semantics is preserved by disjunctive partial deduction. A connection between normal and disjunctive partial deduction is also addressed. In Section 5, disjunctive partial deduction is combined with a proof procedure of disjunctive logic programs, and top-down partial deduction is introduced. Section 6 presents applications of disjunctive partial deduction for abductive logic programs and propositional disjunctive programs. Section 7 discusses related issues and Section 8 summarizes the paper.

2. PRELIMINARIES

We first introduce the framework of disjunctive logic programming which is standard in the literature.

A *normal disjunctive program* is a set of clauses of the form

$$A_1 \vee \cdots \vee A_l \leftarrow B_1 \wedge \cdots \wedge B_m \wedge \mathit{not}\ B_{m+1} \wedge \cdots \wedge \mathit{not}\ B_n \quad (l \geq 0, n \geq m \geq 0),$$

where A_i's and B_j's are atoms and *not* is the *negation-as-failure* operator. The left-hand side of the clause is the *head*, while the right-hand side is the *body*. The

clause is called *disjunctive* (resp. *normal*) if $l > 1$ (resp. $l = 1$). The clause is called an *integrity constraint* if the head is empty ($l = 0$). A normal disjunctive program with no occurrences of *not* is called a *positive disjunctive program*, while a program containing no disjunctive clause is called a *normal logic program*. A *ground* clause/program is a clause/program containing no variables. In this paper, when we write $A \vee \Sigma \leftarrow \Gamma$, Σ denotes a (possibly empty) disjunction in the head, and Γ denotes a (possibly empty) conjunction in the body.[1]

A *substitution* is a mapping from variables to terms $\sigma = \{x_1/t_1, \ldots, x_n/t_n\}$, where each x_i is a variable, each t_i is a term distinct from x_i, and the variables $x_1, \ldots, x_n$ are distinct. The substitution σ is called a *ground substitution* if all t_i's are ground. A substitution σ is a *unifier* of two atoms A and B if $A\sigma = B\sigma$. A unifier σ of two atoms A and B is a *most general unifier* (mgu) if for any unifier θ of A and B, $\theta = \lambda\sigma$ holds for some substitution λ.

An *interpretation* of a program is a subset of the Herbrand base of the program. For a positive disjunctive program P, an interpretation I is called a *minimal model* of P if I is a minimal set satisfying the program. A positive disjunctive program is *consistent* if it has a minimal model; otherwise, a program is *inconsistent*. The *minimal model semantics* [16] of a positive disjunctive program P is defined as the set of all minimal models of P (denoted by $\mathcal{MM}_P$).

Given a normal disjunctive program P and an interpretation I, a ground positive disjunctive program P^I is defined as follows. A clause $A_1 \vee \cdots \vee A_l \leftarrow B_1 \wedge \cdots \wedge B_m$ is in P^I iff $A_1 \vee \cdots \vee A_l \leftarrow B_1 \wedge \cdots \wedge B_m \wedge not\ B_{m+1} \wedge \cdots \wedge not\ B_n$ is a ground instance of a clause in P and $\{B_{m+1}, \ldots, B_n\} \cap I = \emptyset$. Then, I is called a *stable model* of P if I coincides with a minimal model of P^I. A normal disjunctive program can have none, one, or multiple stable models in general. A program having no stable model is called *incoherent*. The *stable model semantics* [6, 17] of a normal disjunctive program P is defined as the set of all stable models of P (denoted by $\mathcal{SM}_P$). The stable model semantics coincides with Gelfond and Lifschitz's stable model semantics [5] in normal logic programs.

3. PARTIAL DEDUCTION IN POSITIVE DISJUNCTIVE PROGRAMS

3.1. Normal Partial Deduction

Partial deduction considered in this paper is defined as *unfolding* between clauses in a program.[2] For a normal logic program P, partial deduction is formally presented as follows.

Given a normal clause C from P,

$$C: H \leftarrow A \wedge \Gamma,$$

suppose that $C_1, \ldots, C_k$ are all of the clauses in P such that

$$C_i: A_i \leftarrow \Gamma_i \quad (1 \le i \le k),$$

where $A\sigma_i = A_i\sigma_i$ holds with an mgu σ_i for each i.

[1] When we write a clause as $A \vee \Sigma \leftarrow \Gamma$, it does not necessarily mean that A should be the left-most atom in the head of the clause. That is, any two clauses are identified modulo the permutation of disjuncts/conjuncts in their heads/bodies.

[2] Partial deduction (also called partial evaluation) is defined in terms of SLDNF procedure in [12], but here we prefer to define it independently of particular procedures.

Then the *normal partial deduction* of P (with respect to C on A) is defined as the program $\pi^N_{\{C;A\}}(P)$ (called a *residual program*) such that

$$\pi^N_{\{C;A\}}(P) = (P \setminus \{C\}) \cup \{C'_1, \ldots, C'_k\},$$

where each C'_i is defined as

$$C'_i: (H \leftarrow \Gamma \wedge \Gamma_i)\sigma_i.$$

When any particular clause C and an atom A are not important in the context, we simply write $\pi^N(P)$ instead of $\pi^N_{\{C;A\}}(P)$.

Example 3.1. Let P be the program consisting of clauses

$$C_1: p(x) \leftarrow q(x),$$
$$C_2: q(y) \leftarrow r(y),$$
$$C_3: q(a) \leftarrow .$$

Then the normal partial deduction of P with respect to C_1 on $q(x)$ becomes $\pi^N_{\{C_1;q(x)\}}(P) = \{C_2, C_3, C'_2, C'_3\}$, where

$$C'_2: p(y) \leftarrow r(y),$$
$$C'_3: p(a) \leftarrow .$$

In the context of the unfold/fold transformation of logic programs, Tamaki and Sato [28] showed that normal partial deduction preserves the least Herbrand model semantics of definite logic programs. The result was later extended to the stable model semantics and the well-founded semantics of normal logic programs [24, 26]. In disjunctive logic programming, however, a program possibly contains disjunctive clauses and normal partial deduction presented above is not applicable in general.

Example 3.2. Let P be the program consisting of clauses

$$C_1: p(a) \vee p(b) \vee q(c) \leftarrow ,$$
$$C_2: p(x) \leftarrow q(x),$$
$$C_3: r(y) \leftarrow p(y).$$

Considering partial deduction with respect to C_3 on $p(y)$, C_3 is unfolded by C_2. In addition, the disjunctive clause C_1 contains disjuncts unifiable with $p(y)$; then C_3 must be unfolded by C_1.

So our first task is to extend normal partial deduction to the one which supplies unfolding for disjunctive clauses.

3.2. Disjunctive Partial Deduction

Partial deduction in positive disjunctive programs is defined as follows.

Definition 3.1. Let P be a positive disjunctive program and C a clause in P of the form

$$C: \Sigma \leftarrow A \wedge \Gamma. \tag{1}$$

Suppose that $C_1, \ldots, C_k$ are all of the clauses in P such that

$$C_i\colon A_i \vee \Sigma_i \leftarrow \Gamma_i \quad (1 \leq i \leq k), \tag{2}$$

where $A_i\sigma_i = A\sigma_i$ holds with an mgu σ_i for each i.

Then a *disjunctive partial deduction* of P (with respect to C on A) is defined as a *residual program* $\pi^D_{\{C;\,A\}}(P)$ such that

$$\pi^D_{\{C;\,A\}}(P) = \begin{cases} P \cup \{C'_1, \ldots, C'_k\}, & \text{if there is a clause } C_i \text{ in } P \text{ such that } \Sigma_i \text{ contains an atom unifiable with } A; \\ (P \setminus \{C\}) \cup \{C'_1, \ldots, C'_k\}, & \text{otherwise,} \end{cases}$$

where

$$C'_i\colon (\Sigma \vee \Sigma_i \leftarrow \Gamma \wedge \Gamma_i)\sigma_i. \tag{3}$$

A simplified notation $\pi^D(P)$ is also used hereafter. In the above transformation, the clause C'_i is produced by resolution in the residual program. In addition, the original clause C is kept in the residual program if C_i contains multiple disjuncts unifiable with A. This is because in this case C would be used for unfolding between A and other unifiable atoms in Σ_i. Otherwise, if each C_i has at most one disjunct unifiable with A, C is removed in the residual program.

Note that in contrast to normal partial deduction, a residual program depends on the choice of the disjunct A_i in C_i if C_i contains multiple disjuncts unifiable with A. In the absence of disjunctive clauses, disjunctive partial deduction coincides with normal partial deduction in the previous section.

Example 3.3. Let P be the program of Example 3.2. Then a disjunctive partial deduction $\pi^D_{\{C_3;\,p(y)\}}(P)$ becomes $\{C_1, C_2, C_3, C'_1, C'_2\}$, where

$$C'_1\colon r(a) \vee p(b) \vee q(c) \leftarrow \quad (\text{by } C_3 \text{ and } C_1 \text{ with the mgu } \{y/a\}),$$
$$C'_2\colon r(x) \leftarrow q(x) \quad (\text{by } C_3 \text{ and } C_2 \text{ with the mgu } \{y/x\}).$$

On the other hand, $\pi^D_{\{C_2;\,q(x)\}}(P)$ becomes $\{C_1, C_3, C''_1\}$, where

$$C''_1\colon p(a) \vee p(b) \vee p(c) \leftarrow \quad (\text{by } C_2 \text{ and } C_1 \text{ with the mgu } \{x/c\}).$$

Now we show that disjunctive partial deduction preserves the minimal model semantics of positive disjunctive programs. We first present a preliminary lemma.

Lemma 3.1. Let P be a positive disjunctive program and M its minimal model. Then an atom A is in M iff there is a ground clause $A \vee \Sigma \leftarrow \Gamma$ from P such that $M \setminus \{A\} \models \Gamma$ and $M \setminus \{A\} \not\models \Sigma$.

PROOF. ($\Rightarrow$) Suppose that for some atom A in M, there is no ground clause $A \vee \Sigma \leftarrow \Gamma$ from P such that $M \setminus \{A\} \models \Gamma$ and $M \setminus \{A\} \not\models \Sigma$. Then, for each ground clause C of the form $A \vee \Sigma \leftarrow \Gamma$, $M \setminus \{A\} \not\models \Gamma$ or $M \setminus \{A\} \models \Sigma$; hence is holds that $M \setminus \{A\} \models \Gamma$ implies $M \setminus \{A\} \models \Sigma$. In this case, $M \setminus \{A\}$ satisfies each clause C and becomes a model of P, which contradicts the assumption that M is a minimal model. Hence the result follows.

($\Leftarrow$) Assume that A is not in M. Then $M \setminus \{A\} = M$, and for a ground clause $A \vee \Sigma \leftarrow \Gamma$ in P, $M \models \Gamma$ and $M \not\models \Sigma$ imply $A \in M$, contradiction. □

Theorem 3.2. Let P be a positive disjunctive program and $\pi^D(P)$ any residual program of P. Then $\mathcal{MM}_P = \mathcal{MM}_{\pi^D(P)}$.

PROOF. Let us consider a disjunctive partial deduction of P with respect to C of the form (1) on the atom A. First consider the case that there is a clause C_i of the form (2) in P such that Σ_i contains an atom unifiable with A. In this case, $\pi^D_{\{C;A\}}(P) = P \cup \{C'_1, \ldots, C'_k\}$. Since P and $\pi^D_{\{C;A\}}(P)$ are logically equivalent, the result immediately holds. Next consider the case that for any clause C_i of the form (2) in P, Σ_i contains no atom unifiable with A.

($\subseteq$) Let M be a minimal model of P. Since the clause (3) is a resolvent of the clauses (1) and (2) in P, M satisfies each clause (3) in $\pi^D_{\{C;A\}}(P)$. Thus M is a model of $\pi^D_{\{C;A\}}(P)$. Assume that there is a minimal model N of $\pi^D_{\{C;A\}}(P)$ such that $N \subset M$. Since N is not a model of P, N does not satisfy the clause (1). Then, for some ground substitution θ such that $(\Sigma \leftarrow A \wedge \Gamma)\theta$ is a ground clause, it holds that $N \models \Gamma\theta$, $N \models A\theta$, and $N \not\models \Sigma\theta$. As the minimal model N of $\pi^D_{\{C;A\}}(P)$ includes $A\theta$, it follows from Lemma 3.1 that there is a ground instance of a clause E of the form (2) or (3) from $\pi^D_{\{C;A\}}(P)$ such that it contains $A\theta$ in the head. Let $E\psi$ be such a ground clause for some substitution ψ. Put $\theta' = \theta\psi$. Then $E\psi = E\theta'$ contains $A\theta' = A\theta$ in the head. Also, by $N \models \Gamma\theta$, $N \models A\theta$, and $N \not\models \Sigma\theta$, it holds that $N \models \Gamma\theta'$, $N \models A\theta'$, and $N \not\models \Sigma\theta'$. (i) Suppose first that $E\theta'$ is a ground instance of the clause of the form (2) such that $A_i\theta' = A\theta'$. Then $N \models A\theta'$ implies $N \setminus \{A\theta'\} \models \Gamma_i\theta'$ and $N \setminus \{A\theta'\} \not\models \Sigma_i\theta'$ (by Lemma 3.1). Here $N \setminus \{A\theta'\} \models \Gamma_i\theta'$ implies $N \models \Gamma_i\theta'$. Since Σ_i contains no atom unifiable with A, $\Sigma_i\theta'$ does not contain $A\theta'$. Thus $N \setminus \{A\theta'\} \not\models \Sigma_i\theta'$ also implies $N \not\models \Sigma_i\theta'$. Hence, N does not satisfy the ground clause $(\Sigma \vee \Sigma_i \leftarrow \Gamma \wedge \Gamma_i)\theta'$. But since $A\theta' = A_i\theta'$ and σ_i is an mgu of A and A_i, it holds that $\theta' = \sigma_i\lambda$ for some λ. Then, the above clause, which is not satisfied in N, is an instance of the clause (3). This contradicts the assumption that N is a model of $\pi^D_{\{C;A\}}(P)$. (ii) Next suppose that $E\theta'$ is a ground instance of the clause of the form (3) such that $(\Sigma \vee \Sigma_i)\theta'$ contains $A\theta'$. In this case, it holds that $\theta' = \sigma_i\lambda$ for some λ; then $A\sigma_i = A_i\sigma_i$ implies $A\theta' = A_i\theta'$. As $\Sigma_i\theta'$ does not contain $A_i\theta'$ in $C_i\theta'$, $\Sigma\theta'$ contains $A\theta'$. Then $N \models A\theta'$ implies $N \models \Sigma\theta'$, which contradicts the fact $N \not\models \Sigma\theta'$. Therefore, M is also a minimal model of $\pi^D_{\{C;A\}}(P)$.

($\supseteq$) Let M be a minimal model of $\pi^D_{\{C;A\}}(P)$. If M is not a model of P, M does not satisfy the clause (1). In this case, $M \not\models \Sigma\theta$, $M \models A\theta$, and $M \models \Gamma\theta$ hold for the ground clause $(\Sigma \leftarrow A \wedge \Gamma)\theta$ with a ground substitution θ. Since the minimal model M of $\pi^D_{\{C;A\}}(P)$ includes $A\theta$, it follows from Lemma 3.1 that there is a ground instance of the clause E of the form (2) or (3) in $\pi^D_{\{C;A\}}(P)$ such that it contains $A\theta$ in the head. Let θ' be a ground substitution defined in the same manner as above. When E is of the form (2), $M \models A\theta'$ implies $M \models \Gamma_i\theta'$ and $M \not\models \Sigma_i\theta'$ (by the same argument as (i) presented above). Thus M does not satisfy the corresponding clause (3), which contradicts the assumption that M is a model of $\pi^D_{\{C;A\}}(P)$. Else when E is of the form (3), by the same argument as (ii) above, it holds that $M \models A\theta'$ implies $M \models \Sigma\theta'$, which contradicts the fact $M \not\models \Sigma\theta'$. Hence M is a model of P. Next assume that there is a minimal model N of P such that $N \subset M$. By ($\subseteq$), N is also a minimal model of $\pi^D_{\{C;A\}}(P)$, but this is impossible since M is a minimal model of $\pi^D_{\{C;A\}}(P)$. □

Corollary 3.3. Let P be a positive disjunctive program. Then P is inconsistent iff $\pi^D(P)$ is inconsistent.

Example 3.4. If is easy to see that both $\pi^D_{\{C_3;\, p(y)\}}(P)$ and $\pi^D_{\{C_2;\, q(x)\}}(P)$ in Example 3.3 have the minimal models $\{p(a), r(a)\}$, $\{p(b), r(b)\}$, and $\{p(c), q(c), r(c)\}$, which are exactly the same as $\mathcal{MM}_P$.

4. PARTIAL DEDUCTION IN NORMAL DISJUNCTIVE PROGRAMS

4.1. Disjunctive Partial Deduction in Normal Disjunctive Programs

Disjunctive partial deduction is defined as unfolding between positive subgoals and disjunctive heads, so the presence of negation as failure in clause bodies does not affect the process of partial deduction. Therefore, disjunctive partial deduction in normal disjunctive programs is defined in the same manner as Definition 3.1, except that in this case each clause possibly contains negation as failure. Then we show that disjunctive partial deduction preserves the stable model semantics of normal disjunctive programs. In the following, $not\Gamma$ means the conjunction of negation-as-failure formulas.

Theorem 4.1. Let P be a normal disjunctive program. Then $\mathcal{SM}_P = \mathcal{SM}_{\pi^D(P)}$.

PROOF. Let M be a stable model of P. Then M is a minimal model of P^M. Since P^M is a positive disjunctive program, M is a minimal model of P^M iff M is a minimal model of $\pi^D(P^M)$ (by Theorem 3.2). Now let us consider the clauses

$$C\colon \Sigma \leftarrow A \wedge \Gamma \wedge not\,\Gamma'$$

and

$$C_i\colon A_i \vee \Sigma_i \leftarrow \Gamma_i \wedge not\,\Gamma_i' \quad (1 \le i \le k)$$

in P, where $A\sigma_i = A_i\sigma_i$ for some mgu σ_i. Let θ be any ground substitution which makes both C and C_i ground and $A\theta = A_i\theta$.[3] In this case, σ_i is an mgu of A and A_i, so $\theta = \sigma_i\lambda$ holds for some substitution λ.

(i) If $M \not\models \Gamma'\theta$ and $M \not\models \Gamma_i'\theta$ for some i $(1 \le i \le k)$, the clauses

$$E\theta\colon (\Sigma \leftarrow A \wedge \Gamma)\theta$$

and

$$E_i\theta\colon (A_i \vee \Sigma_i \leftarrow \Gamma_i)\theta$$

are in P^M. From these clauses, disjunctive partial deduction generates the clause

$$E_i'\theta\colon (\Sigma \vee \Sigma_i \leftarrow \Gamma \wedge \Gamma_i)\theta$$

in $\pi^D_{\{C;\, A\}}(P^M)$. On the other hand, from C and C_i in P, there is the clause

$$C_i'\colon (\Sigma \vee \Sigma_i \leftarrow \Gamma \wedge \Gamma_i \wedge not\,\Gamma' \wedge not\,\Gamma_i')\sigma_i$$

in $\pi^D_{\{C;\, A\}}(P)$. Since $M \not\models \Gamma'\theta$ and $M \not\models \Gamma_i'\theta$ hold for $\theta = \sigma_i\lambda$, $C_i'\theta$ becomes $E_i'\theta$ in $\pi^D_{\{C;\, A\}}(P)^M$.

(ii) Else if $M \models \Gamma'\theta$ or $M \models \Gamma_i'\theta$ for any i $(1 \le i \le k)$, the clause $E\theta$ or $E_i\theta$ is not included in P^M. Then the clause $E_i'\theta$ is not produced in $\pi^D_{\{C;\, A\}}(P^M)$. On the other hand, each clause $C_i'\theta$ from $\pi^D_{\{C;\, A\}}(P)$ is eliminated in $\pi^D_{\{C;\, A\}}(P)^M$.

[3] Such a ground substitution is obtained by combining a ground substitution for C and one for C_i with suitable variable renaming.

Thus, there is a one-to-one correspondence between the clauses in $\pi^D_{\{C;A\}}(P^M)$ and the clauses in $\pi^D_{\{C;A\}}(P)^M$; hence, $\pi^D_{\{C;A\}}(P^M) = \pi^D_{\{C;A\}}(P)^M$. Therefore, M is a minimal model of a $\pi^D(P^M)$ iff M is a minimal model of $\pi^D(P)^M$ and a stable model of $\pi^D(P)$. □

Corollary 4.2. Let P be a normal disjunctive program. Then P is incoherent iff $\pi^D(P)$ is incoherent.

The above theorem also implies that in normal logic programs, normal partial deduction preserves Gelfond and Lifschitz's stable model semantics.

Corollary 4.3. Let P be a normal logic program. Then $\mathcal{SM}_P = \mathcal{SM}_{\pi^N(P)}$.

The above corollary justifies the corresponding result of [24].

4.2. Connection between Normal and Disjunctive Partial Deduction

Next we present a method of computing disjunctive partial deduction in terms of normal partial deduction.

Let C be a clause of the form $A \vee \Sigma \leftarrow \Gamma$. Then, we define $Sft(C, A)$ as the clause

$$\tilde{C}\colon A \leftarrow \Gamma \wedge \tilde{\Sigma}$$

where $\tilde{\Sigma} = \tilde{A}_1 \wedge \cdots \wedge \tilde{A}_l$ for $\Sigma = A_1 \vee \cdots \vee A_l$, and each $\tilde{A}_j$ is a new atom introduced for each A_j. In particular, $C = \tilde{C}$ if Σ is empty. Conversely, the function Sft^{-1} is defined as the transformation which transforms any clause $\tilde{C}$ to C in a program. Given a normal disjunctive program P, $\tilde{P}$ is a normal logic program in which any disjunctive clause C in P is replaced with some normal clause $\tilde{C}$.

Theorem 4.4. Given a normal disjunctive program P, let C be a clause of the form (1) *and C_i a clause of the form* (2) *such that $A\sigma_i = A_i\sigma_i$ with an mgu σ_i. Then,*

$$\pi^D_{\{C;A\}}(P) = \begin{cases} Sft^{-1}\left(\pi^N_{\{\tilde{C};A\}}(\tilde{P} \cup \{A \leftarrow A\})\right), & \text{if there is a clause } C_i \text{ in } P \text{ such that } \Sigma_i \text{ contains an atom unifiable with } A; \\ Sft^{-1}\left(\pi^N_{\{\tilde{C};A\}}(\tilde{P})\right), & \text{otherwise,} \end{cases}$$

where $\tilde{P}$ is a normal logic program in which C is replaced with some $\tilde{C}$ and each C_i is replaced with $Sft(C_i, A_i)$.

PROOF. Let $\tilde{C} = H_1 \leftarrow A \wedge \Gamma \wedge \Xi$, where $\Sigma = H_1 \vee \cdots \vee H_l$ and $\Xi = \tilde{H}_2 \wedge \cdots \wedge \tilde{H}_l$. When each C_i contains at most one disjunct unifiable with A, by $\tilde{C}$ and $Sft(C_i, A_i) = A_i \leftarrow \Gamma_i \wedge \tilde{\Sigma}_i$, $\pi^N_{\{\tilde{C};A\}}(\tilde{P})$ produces clauses

$$\tilde{C}'_i\colon \left(H_1 \leftarrow \Gamma \wedge \Xi \wedge \Gamma_i \wedge \tilde{\Sigma}_i\right)\sigma_i,$$

which become the disjunctive clause (3) by Sft^{-1}. Therefore, $Sft^{-1}(\pi^N_{\{\tilde{C};A\}}(\tilde{P})) = \pi^D_{\{C;A\}}(P)$. Else when some C_i contains more than one disjunct unifiable with A, by unfolding $\tilde{C}$ with $A \leftarrow A$, $\pi^N_{\{\tilde{C};A\}}(\tilde{P} \cup \{A \leftarrow A\})$ produces $\tilde{C}$ as well as $\tilde{C}'_i$. Therefore, $Sft^{-1}(\pi^N_{\{\tilde{C};A\}}(\tilde{P} \cup \{A \leftarrow A\}))$ coincides with $\pi^D_{\{C;A\}}(P)$. □

Example 4.1. Let P be the program

C_1: $p(a) \vee p(b) \leftarrow$,

C_2: $q(x) \leftarrow p(x)$.

Then, a disjunctive partial deduction $\pi^D_{\{C_2;\, p(x)\}}(P)$ becomes $\{C_1, C_2, C_1'\}$, where

C_1': $p(b) \vee q(a) \leftarrow$.

On the other hand, $Sft(C_1, p(a))$ is

$\tilde{C}_1$: $p(a) \leftarrow \tilde{p}(b)$,

and $\tilde{P} = \{\tilde{C}_1, C_2\}$. Since C_1 has two disjuncts unifiable with $p(x)$, by putting $C_3 = p(x) \leftarrow p(x)$, $\pi^N_{\{C_2;\, p(x)\}}(\tilde{P} \cup \{C_3\}) = \{\tilde{C}_1, C_2, C_3, C_4\}$, where

C_4: $q(a) \leftarrow \tilde{p}(b)$.

C_4 becomes C_1' by Sft^{-1}, so $Sft^{-1}(\pi^N_{\{C_2;\, p(x)\}}(\tilde{P} \cup \{C_3\})) = \{C_1, C_2, C_3, C_1'\}$. Here C_3 is a tautology; hence, $Sft^{-1}(\pi^N_{\{C_2;\, p(x)\}}(\tilde{P} \cup \{C_3\}))$ is equivalent to $\pi^D_{\{C_2;\, q(x)\}}(P)$.

The above theorem shows that disjunctive partial deduction can be computed using normal partial deduction together with suitable program transformations.

5. COMPUTING PARTIAL DEDUCTION

5.1. Bottom-Up Proof Procedure

Computation of minimal and stable models of a disjunctive program is achieved in a bottom-up fashion. The following procedure, which generates a set of models of a positive disjunctive program, is a standard one in the literature [7, 13, 15].

Given a positive disjunctive program P, let $\mathcal{S}_P^0 = \{\emptyset\}$. Then for $i \geq 0$ do:

1. For any $I \in \mathcal{S}_P^i$, there is an integrity constant in P of the form

   $$\leftarrow B_1 \wedge \cdots \wedge B_m$$

   such that $I \models (B_1 \wedge \cdots \wedge B_m)\sigma$ for some ground substitution σ, then remove I from $\mathcal{S}_P^i$.
2. For any $I \in \mathcal{S}_P^i$ and for every clause in P of the form

   $$A_1 \vee \cdots \vee A_l \leftarrow B_1 \wedge \cdots \wedge B_m \quad (l \geq 1)$$

   such that $I \models (B_1 \wedge \cdots \wedge B_m)\sigma$ and $I \not\models (A_1 \vee \cdots \vee A_l)\sigma$ for some ground substitution σ, put $I \cup \{A_j\sigma\}$ into $\mathcal{S}_P^{i+1}$ for every $j = 1, \ldots, l$.
3. Iterate the above two steps until it reaches the fixpoint $\mathcal{S}_P^{n+1} = \mathcal{S}_P^n$.

In step 1, the procedure prunes interpretations which do not satisfy integrity constraints in the program. In step 2, the procedure generates the new set of interpretations $\mathcal{S}_P^{i+1}$ from $\mathcal{S}_P^i$ by performing forward-reasoning based on hyper-resolution and case-splitting on non-unit derived clauses.

The soundness of case-splitting is guaranteed in *range-restricted* programs,[4] while the termination of the procedure is assured in function-free programs. The next lemma presents that the above procedure is used to compute the minimal model semantics.

[4] Any variable in a clause has an occurrence in a positive atom in the body.

Lemma 5.1 [7, 13, 15]. *Let P be a range-restricted function-free positive disjunctive program and $\mathcal{I}_P^\omega$ the fixpoint of the above procedure. Then, $\mathcal{MM}_P = \min(\mathcal{I}_P^\omega)$, where* $\min(\mathcal{I}) = \{I \in \mathcal{I} \mid \nexists J \in \mathcal{I}$ *such that* $J \subset I\}$.

Now we show that disjunctive partial deduction can be combined with the procedure presented above.

Lemma 5.2. Let P be a range-restricted positive disjunctive program. Then any residual program $\pi^D(P)$ is also range-restricted.

PROOF. Since P is range-restricted, any variable occurring in Σ_i in (2) also occurs in Γ_i. On the other hand, any variable occurring in Σ of (1) occurs in either Γ or A. If it occurs in Γ, then the clause (3) is range-restricted. Else if it occurs in A, each variable x_j in A is unified with the corresponding term t_j in A_i, where variables, if any, in t_j occur in Γ_i. Thus, the clause (3) is range-restricted. Hence, the residual program $\pi^D_{\{C;\,A\}}(P)$ is range-restricted. □

By Lemmas 5.1 and 5.2 and Theorem 3.2, the following result holds.

Theorem 5.3. Let P be a range-restricted function-free positive disjunctive program and $\pi^D(P)$ any residual program of P. Then,

$$\mathcal{MM}_P = \min(\mathcal{I}_P^\omega) = \min(\mathcal{I}_{\pi^D(P)}^\omega).$$

Computation of stable models and partial deduction in normal disjunctive programs are similarly achieved using the program transformation in [7].

5.2. *Top-Down Partial Deduction*

Top-down partial deduction specializes a program with respect to a given atom, which is useful to optimize programs for query-answering in deductive databases.

Given a normal disjunctive program P, an atom A is *true* in P under the stable model semantics if for every stable model I of P, there is a substitution σ such that $A\sigma$ is included in I. Else if for some stable model I of P there is σ such that $A\sigma$ is included in I, A is *possibly true* in P. Otherwise, if there is no such substitution, A is *false* in P.

Top-down partial deduction in normal disjunctive programs is defined as follows.

Definition 5.1. Given a normal disjunctive program P and an atom A, let us define

$$\Pi_A^0(P) = P,$$

$$\Pi_A^{i+1}(P) = \pi^D_{\{C,\,B_j\}}(\Pi_A^i(P)),$$

where C is any clause of the form $A' \vee \Sigma \leftarrow \Gamma$ in $\Pi_A^i(P)$ such that A' is unifiable with A, and $B_j \in \Gamma$. Then *top-down partial deduction* with respect to A is defined as any $\Pi_A^i(P)$ $(i > 0)$.

By definition, $\Pi_A^i(P)$ is defined as a residual program which is obtained from P by iteratively performing disjunctive partial deduction with respect to any clause containing an atom unifiable with A in the head. As a result, deduction steps starting from the target atom A are reduced in residual programs when queried.

Since top-down partial deduction is defined as an iterative application of disjunctive partial deduction, the next result immediately follows from Theorem 4.1.

Theorem 5.4. Let P be a normal disjunctive program. Then, A is true (resp. possibly true, false) in P iff A is true (resp. possibly true, false) in $\Pi_A^i(P)$ $(i > 0)$.

Example 5.1. Let P be the program

$$C_1: q(x) \leftarrow p(x),$$
$$C_2: p(x) \leftarrow r(x),$$
$$C_3: r(a) \vee s(b) \leftarrow .$$

Then, $\Pi^2_{q(x)}(P) = \{C_2, C_3, C_4\}$, where

$$C_4: q(a) \vee s(b) \leftarrow ,$$

and $q(x)$ is possibly true.

6. APPLICATIONS

6.1. Optimizing Abductive Logic Programs

Abductive logic programming [9] is an extension of logic programming which has important applications in common-sense reasoning and knowledge representation. Inoue and Sakama [8, 19] showed that abductive logic programs are expressed in terms of disjunctive logic programs. This fact implies that disjunctive partial deduction is also used to optimize abductive logic programs. In this section, we address an application of disjunctive partial deduction to abductive logic programming.

An abductive logic program is defined in terms of a normal disjunctive program as follows.

Definition 6.1 [8, 19]. Let P be a normal disjunctive program and $\mathscr{A}$ a set of atoms called *abducibles*. Then, an *abductive disjunctive program* is defined as a normal disjunctive program

$$P_A = P \cup \{A \vee \tilde{A} \leftarrow \mid A \in \mathscr{A}\},$$

where $\tilde{A}$ is a newly introduced atom associated with each A.[5]

In an abductive disjunctive program, an abductive hypothesis is either assumed or not. The situation is encoded in the program by the augmented disjunctive clause. Namely, the disjunct A is chosen if an abducible A is assumed, while $\tilde{A}$ is chosen otherwise.

Let P_A be an abductive disjunctive program and O a ground atom which represents an *observation*.[6] Then, a set $E \subseteq \mathscr{A}$ is an *explanation* of O in P_A if there

[5] In [8], a negation-as-failure formula *not* A is used instead of $\tilde{A}$.

[6] If an observation contains variables $O(x)$, we introduce a clause $O' \leftarrow O(x)$ to a program and consider the newly introduced ground atom O' as an observation.

is a stable model M of P_A such that $O \in M$ and $E = M \cap \mathscr{A}$.[7] An explanation E is *minimal* if no other explanation F exists such that $F \subset E$.

Example 6.1. Let P be the program

C_1: $o \leftarrow p$,

C_2: $p \leftarrow a$,

C_3: $p \leftarrow not\, b$,

with the abducibles $\mathscr{A} = \{a, b\}$. Then the abductive disjunctive program P_A includes the additional clauses

C_4: $a \vee \tilde{a} \leftarrow$,

C_5: $b \vee \tilde{b} \leftarrow$.

Suppose that o is an observation. Then P_A has three stable models containing o: $\{o, p, \tilde{a}, \tilde{b}\}$, $\{o, p, a, \tilde{b}\}$, $\{o, p, a, b\}$. Hence, $\emptyset$, $\{a\}$, and $\{a, b\}$ are explanations of o, while $\emptyset$ is the minimal explanation.

Now we apply top-down partial deduction to optimize abductive logic programs.

Theorem 6.1. Let P_A be an abductive disjunctive program and O an observation. Then, E is a (minimal) explanation of O in P_A iff E is a (minimal) explanation of O in $\Pi_O^i(P_A)$ $(i > 0)$.

PROOF. The result follows from the definition of explanations and Theorem 5.4. □

Thus, top-down partial deduction optimizes abductive logic programs by reducing inference steps from an observation to explanations.

Example 6.2. In Example 6.1, $\Pi_o^2(P_A) = \{C_2, C_3, C_4, C_5, C_6, C_7\}$, where

C_6: $o \vee \tilde{a} \leftarrow$,

C_7: $o \leftarrow not\, b$.

Then $\Pi_o^2(P_A)$ has three stable models containing o, which are equivalent to those of P_A. Hence, all the (minimal) explanations are preserved.

Sakama and Inoue [21] also introduce another method of partial deduction for abductive logic programs.

6.2. *Compiling Propositional Disjunctive Programs to Normal Logic Programs*

In propositional disjunctive programs, disjunctive partial deduction is used to compile disjunctive logic programs to normal logic programs.

A ground clause is called a *conditional fact* if it has no positive subgoals (i.e., $\Sigma \leftarrow not\, \Gamma$). A propositional normal disjunctive program is called a *normal form* if every clause in the program is a conditional fact. A ground clause $\Sigma \leftarrow \Gamma$ is a

[7] Here, $\mathscr{A}$ is identified with its ground instances.

tautology if $\Sigma \cap \Gamma \neq \emptyset$. Brass and Dix [3] showed that the iterative applications of disjunctive partial deduction and the elimination of tautology transform any propositional normal disjunctive program to a semantically equivalent disjunctive program in a normal form.

Example 6.3. Let P_0 be the program

C_1: $p \leftarrow q \wedge notr$,

C_2: $q \leftarrow p$,

C_3: $r \leftarrow s$,

C_4: $p \vee q \leftarrow$.

First, $\pi^D_{\{C_1, q\}}(P_0)$ becomes $P_1 = \{C_2, C_3, C_4, C_5, C_6\}$, where[8]

C_5: $p \leftarrow p \wedge notr$,

C_6: $p \leftarrow notr$.

Here, C_5 is a tautology, hence removed. Put $P_2 = P_1 \setminus \{C_5\}$. Then, $\pi^D_{\{C_2, p\}}(P_2)$ becomes $P_3 = \{C_3, C_4, C_6, C_7, C_8\}$, where

C_7: $q \leftarrow$,

C_8: $q \leftarrow notr$.

Finally, $\pi^D_{\{C_3, s\}}(P_3)$ just removes the clause C_3 from the program, and the resulting program $P_4 = \{C_4, C_6, C_7, C_8\}$ is a normal form.

A *dependency graph* of a propositional normal disjunctive program P is a directed graph in which its nodes are ground atoms from P and there is an edge from A to B iff there is a ground clause C from P such that A appears in the head and B appears positively in the body of C. A propositional normal disjunctive program P is a *head-cycle-free* [1] if its dependency graph contains no cycle which goes through two atoms appearing in the head of the same disjunctive clause.

Let P be a propositional normal disjunctive program. Then the normal logic program $n(P)$ is obtained from P by replacing each disjunctive clause

$$A_1 \vee \cdots \vee A_l \leftarrow \Gamma$$

with l normal clauses

$$A_i \leftarrow \Gamma \wedge notA_1 \wedge \cdots \wedge notA_{i-1} \wedge notA_{i+1} \wedge \cdots \wedge notA_l \quad (1 \leq i \leq l).$$

We call the above transformation *shifting*. Then the following result holds.

Lemma 6.2 [1]. *Let P be a head-cycle-free proposition normal disjunctive program. Then M is a stable model of P iff M is a stable model of $n(P)$.*

Thus any head-cycle-free propositional normal disjunctive program is transformed to a semantically equivalent normal logic program.

Any disjunctive program in a normal form contains no positive subgoal; hence, it is always head-cycle-free. This fact, together with Lemma 6.2, implies the following result.

[8] In C_6, $p \vee p \leftarrow notr$ is identified with $p \leftarrow notr$. Such merging is also done in C_7.

Theorem 6.3. Let P be a propositional normal disjunctive program. Then, P is transformed to a semantically equivalent (under the stable model semantics) normal logic program by the transformation sequences of disjunctive partial deduction and tautology elimination, and shifting.

Example 6.4. In Example 6.3, by shifting, P_4 is transformed to the program $P_5 = \{C_6, C_7, C_8, C_9, C_{10}\}$, where

C_9: $p \leftarrow not\, q$,

C_{10}: $q \leftarrow not\, p$.

Then, P_0 and P_5 have the same stable model $\{p, q\}$.

It is known that the computation of stable models in propositional normal disjunctive programs is generally harder than that in propositional normal logic programs. Namely, the computational complexity of stable models in propositional normal disjunctive programs is at the second level of the polynomial hierarchy, while in propositional normal logic programs it is at the first level [4]. Therefore, there is no polynomial-time transformation from normal disjunctive programs to normal logic programs in general, unless the polynomial hierarchy collapses. In this sense, the above transformation costs exponential computation in the worst case, but it might be useful as a compilation technique from disjunctive to normal logic programs. Note that in a predicate disjunctive program, disjunctive partial deduction and tautology elimination cannot transform the program to a normal form in general; hence, such compilation is impossible.[9]

7. DISCUSSION

Partial deduction in logic programming has been widely investigated for normal logic programs, while few results are known for disjunctive logic programs. Sakama and Seki [22] introduced disjunctive partial deduction for propositional disjunctive programs. Brass and Dix [2, 3] also independently developed partial deduction for propositional disjunctive programs which is equivalent to [22]. In [2, 3], the authors investigate several abstract properties of disjunctive logic programs and characterize various semantics in terms of partial deduction. Procedurally, [2] introduces a bottom-up procedure to compute a fixpoint of conditional facts in a function-free and range-restricted normal disjunctive program.

Disjunctive partial deduction reduces deduction steps by unfolding clauses in a program, while it generally introduces new disjunctions in the program. For example, given the program

P: $p \vee q \leftarrow$,

$r \leftarrow q$,

disjunctive partial deduction generates the program

P': $p \vee q \leftarrow$,

$p \vee r \leftarrow$.

[9] For instance, the positive subgoal in $p(f(x)) \leftarrow p(x)$ cannot be eliminated by disjunctive partial deduction and tautology elimination.

In other words, disjunctive partial deduction reduces the depth of a *model tree* [13], while introducing additional branches as a trade-off. By contrast, *folding*, which is a dual operation of unfolding, often reduces the number of disjunctions in a program. In the above example, folding P' will generate P. Thus, folding will also be useful when one wants to reduce the number of branches in a model tree. A general framework of folding in disjunctive logic programming is not known and is an interesting topic to be investigated.

In the previous section, we presented some applications of disjunctive partial deduction. Further applications of disjunctive partial deduction are as follows. First, positive disjunctive programs are identified with first-order theories; therefore, disjunctive partial deduction can be used as an optimization technique for first-order theorem provers. Sato [23] proposes unfold/fold transformation systems for first-order programs, but he does not treat disjunctive clauses in a program. Second, disjunctive partial deduction is also directly applicable to disjunctive logic programs containing classical negation [6]. This is because the answer set semantics of extended disjunctive programs can be translated into the stable model semantics of normal disjunctive programs by viewing negative literals as new atoms [6]. Third, disjunctive logic programming is closely related to other nonmonotonic formalisms in AI [4, 18], so that the partial deduction technique presented in this paper has potential application to nonmonotonic reasoning systems.

Finally, it is worth noting that the resolution-based disjunctive partial deduction does not always preserve the syntax-dependent logic programming semantics. Among others, semantics for *inclusive disjunctions* such as the *possible model semantics* [20] are not preserved in general. For instance, in the program presented above, $\{p, q\}$ and $\{p, r\}$ are possible models of P', which are not possible models of P. This is because models for inclusive disjunctions are usually *weakly supported*,[10] and weakly supported models are not preserved by disjunctive partial deduction in general [3].

8. CONCLUSION

This paper presented partial deduction techniques in disjunctive logic programming. We introduced disjunctive partial deduction for disjunctive logic programs, which is a natural extension of normal partial deduction. It was shown that disjunctive partial deduction preserves the minimal model semantics of positive disjunctive programs, and the stable model semantics of normal disjunctive programs. Disjunctive partial deduction was combined with a bottom-up proof procedure of disjunctive logic programs, and top-down partial deduction was introduced for query optimization. We also addressed applications of disjunctive partial deduction to optimizing abductive logic programs and compiling propositional disjunctive programs.

The potential importance of disjunctive logic programming in artificial intelligence and knowledge representation is recognized these days. Disjunctive logic programming has rich expressive power but its computation is generally expensive. In this respect, partial deduction makes disjunctive logic programming more practical by providing a method for optimizing disjunctive logic programs and disjunctive deductive databases.

[10] A model M of a program P is weakly supported if for each $A \in M$, there is a clause $A \vee \Sigma \leftarrow \Gamma$ from P such that $M \models \Gamma$ [3].

We thank Katsumi Inoue for useful discussion on the subject of this paper. The result of Section 6.2 was suggested by Jia-Huai You. Thanks also to the anonymous referee for detailed comments on the earlier draft of this paper.

REFERENCES

1. Ben-Eliyahu, R. and Dechter, R., Propositional Semantics for Disjunctive Logic Programs, in: *Proceedings of the Joint International Conference and Symposium on Logic Programming*, MIT Press, Cambridge, MA, 1992, pp. 813–827.
2. Brass, S. and Dix, J., Disjunctive Semantics Based upon Partial and Bottom-Up Evaluation, in: *Proceedings of the 12th International Conference on Logic Programming*, MIT Press, Cambridge, MA, 1995, pp. 199–213.
3. Brass, S. and Dix, J., Characterizations of the Stable Semantics by Partial Evaluation, in: *Proceedings of the 3rd International Conference on Logic Programming and Nonmonotonic Reasoning, LNAI 928*, Springer-Verlag, Berlin, 1995, pp. 85–98.
4. Eiter, T. and Gottlob, G., Complexity Results for Disjunctive Logic Programming and Application to Nonmonotonic Logics, in: *Proceedings of the International Logic Programming Symposium*, MIT Press, Cambridge, MA, 1993, pp. 266–278.
5. Gelfond, M. and Lifschitz, V., The Stable Model Semantics for Logic Programming, in: *Proceedings of the 5th International Conference and Symposium on Logic Programming*, MIT Press, Cambridge, MA, 1988, pp. 1070–1080.
6. Gelfond, M. and Lifschitz, V., Classical Negation in Logic Programs and Disjunctive Databases, *New Generation Computing* 9:365–385 (1991).
7. Inoue, K., Koshimura, M., and Hasegawa, R., Embedding Negation as Failure into a Model Generation Theorem Prover, in: *Proceedings of the 11th International Conference on Automated Deduction, LNAI 607*, Springer-Verlag, Berlin, 1992, pp. 400–415.
8. Inoue, K. and Sakama, C., On Positive Occurrences of Negation as Failure, in: *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, Los Altos, CA, 1994, pp. 293–304.
9. Kakas, A. C., Kowalski, R. A., and Toni, F., Abductive Logic Programming, *J. Logic and Computation* 2:719–770 (1992).
10. Komorowski, J., A Specification of an Abstract Prolog Machine and Its Application to Partial Evaluation, Technical Report LSST 69, Linköping University, 1981.
11. Komorowski, J., An Introduction to Partial Deduction, in: *Proceedings of the 3rd International Workshop on Meta-Programming in Logic, LNCS 649*, Springer-Verlag, Berlin, 1992, pp. 49–69.
12. Lloyd, J. W. and Shepherdson, J. C., Partial Evaluation in Logic Programming, *J. Logic Programming* 11:217–242 (1991).
12. Lobo, J., Minker, J., and Rajasekar, A., *Foundations of Disjunctive Logic Programming*, MIT Press, Cambridge, MA, 1992.
14. Maher, M., A Transformation System for Deductive Database Modules with Perfect Model Semantics, *Theoretical Computer Science* 110:377–403 (1993).
15. Manthey, R. and Bry, F., SATCHMO: A Theorem Prover Implemented in Prolog, in: *Proceedings of the 9th International Conference on Automated Deduction, LNCS 310*, Springer-Verlag, Berlin, 1988, pp. 415–434.
16. Minker, J., On Indefinite Data Bases and the Closed World Assumption, in: *Proceedings of the 6th International Conference on Automated Deduction, LNCS 138*, Springer-Verlag, Berlin, 1982, pp. 292–308.
17. Przymusinski, T. C., Stable Semantics for Disjunctive Programs, *New Generation Computing* 9:401–424 (1991).
18. Sakama, C. and Inoue, K., Relating Disjunctive Logic Programs to Default Theories, in: *Proceedings of the 2nd International Workshop on Logic Programming and Nonmonotonic Reasoning*, MIT Press, Cambridge, MA, 1993, pp. 266–282.

19. Sakama, C. and Inoue, K., On the Equivalence between Disjunctive and Abductive Logic Programs, in: *Proceedings of the 11th International Conference on Logic Programming*, MIT Press, Cambridge, MA, 1994, pp. 489–503.
20. Sakama, C. and Inoue, K., An Alternative Approach to the Semantics of Disjunctive Logic Programs and Deductive Databases, *J. Automated Reasoning* 13:145–172 (1994).
21. Sakama, C. and Inoue, K., The Effect of Partial Deduction in Abductive Reasoning, in: *Proceedings of the 12th International Conference on Logic Programming*, MIT Press, Cambridge, MA, 1995, pp. 383–397.
22. Sakama, C. and Seki, H., Partial Deduction of Disjunctive Logic Programs: A Declarative Approach, in: *Proceedings of the 4th International Workshop on Logic Program Synthesis and Transformation, LNCS 883*, Springer-Verlag, Berlin, 1994, pp. 170–182.
23. Sato, T., Equivalence-Preserving First Order Unfold/Fold Transformation Systems, *Theoretical Computer Science* 105:57–84 (1992).
24. Seki, H., A Comparative Study of the Well-Founded and the Stable Model Semantics: Transformation's Viewpoint, in: *Proceedings of the Workshop on Logic Programming and Nonmonotonic Logic*, Association for Logic Programming and Mathematics Sciences Institute, Cornell University, 1990, pp. 115–123.
25. Seki, H., Unfold/Fold Transformation of Stratified Programs, *Theoretical Computer Science* 86:107–139 (1991).
26. Seki, H., Unfold/Fold Transformation of General Logic Programs for the Well-Founded Semantics, *J. Logic Programming* 16:5–23 (1993).
27. Sestoft, P. and Zamulin, A. V., Annotated Bibliography on Partial Evaluation and Mixed Computation, *New Generation Computing* 6:309–354 (1988).
28. Tamaki, H. and Sato, T., Unfold/Fold Transformation of Logic Programs, in: *Proceedings of the 2nd International Conference on Logic Programming*, 1984, pp. 127–138.