dl2asp: Implementing Default Logic via Answer Set Programming

Yin Chen[1], Hai Wan[2], Yan Zhang[3], and Yi Zhou[3]

[1] Department of Computer Science, South China Normal University, Guangzhou, China, 510631
[2] School of Software, Sun Yat-Sen University, Guangzhou, China, 510275
[3] School of Computing and Information Technology, University of Western Sydney, Penrith South DC, NSW 1797, Australia

Abstract. In this paper, we show that Reiter's default logic in the propositional case can be translated into answer set programming by identifying the internal relationships among formulas in a default theory. Based on this idea, we implement a new default logic solver - dl2asp. We report some experimental results, in particular the application of dl2asp for solving the fair division problem in social choice theory.

1 Introduction

As a predominant approach for nonmonotonic reasoning, Default Logic (DL) [21] has attracted many researchers in the last three decades. Default logic is theoretically significant not only because of its elegant syntax and semantics, but also its expressive power to capture other nonmonotonic reasoning approaches, such as autoepistemic logic, defeasible reasoning, and so on [1,11,8,22].

However, despite the remarkable success on theoretical aspects, default logic has encountered huge difficulties from a practical viewpoint. Although many endeavors have been done [3,18,17], the implementation of default logic still remains unsatisfactory. Consequently, the practical value of default logic has been severely restricted.

This paper intends to address this issue by translating default logic into Answer Set Programming (ASP) [7], a promising approach that has been successfully implemented by a number of sophisticated solvers [6,19,12,13]. It is well-known that ASP is a special case of default logic by restricting the formulas in default theories to atoms/literals [8,22]. An interesting question arises whether the converse can also be done to some extent. In other words, is it possible to translate default logic back into answer set programming?

We answer this question positively. We show that default logic in the propositional case can be translated to answer set programming by identifying the internal relationships among formulas in a default theory. By internal relationships, we mean those implication rules whose head is a formula, whose body is a set of formulas occurred in the default theory, and the body entails the head in propositional logic.

This translation is not only theoretically interesting but also of practical relevance. Based on the translation, we implement a new solver for default logic, called dl2asp. We report some experimental results, which demonstrate that the performance of dl2asp is rather satisfactory.

The paper is organized as follows. Section 2 recalls some basic notions and definitions in default logic and answer set programming. Section 3 presents the translation from default logic to answer set programming, and discusses some related properties. Section 4 explains the implementation of dl2asp in detail. Section 5 reports some experiments, while Section 6 considers an application of dl2asp for solving the fair division problem in social choice theory. Finally, Section 7 concludes the paper.

2 Preliminaries

2.1 Reiter's Default Logic

We consider Reiter's default logic [21] in propositional case. A *default theory* Δ is a pair $\langle W, D\rangle$, where W is a set of propositional formulas and D is a set of *defaults* of the following form:

$$\alpha : \beta_1, \dots, \beta_n / \gamma, \tag{1}$$

where $\alpha, \beta_1, \dots, \beta_n, \gamma$ are propositional formulas. In addition, α is called the *prerequisite*, $\beta_1, \dots, \beta_n$ the *justifications*, and γ the *conclusion* of the default. Let Δ be a default theory. We use P_Δ, J_Δ and C_Δ to denote the sets of prerequisites, justifications and conclusions occurred in the default theory respectively.

Definition 1 (Extension [21]). *Let $\Delta = \langle W, D\rangle$ be a default theory and T a theory. We say that T is an* extension *of Δ if $T = \Gamma(T)$, where for any theory S, $\Gamma(S)$ is the minimal set (in the sense of set inclusion) satisfying the following three conditions:*

1. *$W \subseteq \Gamma(S)$.*
2. *$\Gamma(S)$ is a theory.*
3. *For any default rule $\alpha : \beta_1, \dots, \beta_n/\gamma \in D$, if $\alpha \in \Gamma(S)$ and $\neg\beta_i \notin S, (1 \leq i \leq n)$, then $\gamma \in \Gamma(S)$.*

Example 1. Consider the default theory $\Delta_1 = \langle W_1, D_1\rangle$, where $W_1 = \{\neg b \vee \neg c, c \vee d\}$ and D_1 contains the following four defaults:

$$: \neg b/a, \tag{2}$$

$$: \neg a, \neg c/b, \tag{3}$$

$$: a \wedge \neg b/\neg d, \tag{4}$$

$$\neg c : \neg a/\neg a. \tag{5}$$

It can be checked that Δ_1 has two extensions (under equivalence): $E_1 = Th(W_1 \cup \{a, \neg d\})$ and $E_2 = Th(W_1 \cup \{\neg a, b\})$.[1]

[1] We use $Th(S)$ to denote the deductive closure of a set S of formulas.

2.2 Answer Set Programming

An *answer set program* (*program* for short) is a set of *rules* of the following form:

$$a \leftarrow b_1, \dots, b_m, \mathsf{not}\ b_{m+1}, \dots, \mathsf{not}\ b_n, \tag{6}$$

where $0 \le m \le n$, a is either an atom or $\bot$, and $b_1, \dots b_n$ are atoms. In addition, a is called the *head* of the rule and $\{b_1, \dots, b_m, \mathsf{not}\ b_{m+1}, \dots, \mathsf{not}\ b_n\}$ the *body* of the rule. More specifically, $\{b_1, \dots, b_m\}$ is called the *positive body* of the rule, while $\{b_{m+1}, \dots, b_n\}$ is called the *negative body* of the rule. We call a rule a *constraint* if a is $\bot$, *fact* if $n = 0$, and *positive* if $m = n$.

Let X be a set of atoms. We say that X *satisfies* a rule of form (6) if X satisfies its head (i.e. $a \in X$) whenever X satisfies its body (i.e. $\{b_1, \dots, b_m\} \subseteq X$ and $\{b_{m+1}, \dots, b_n\} \cap X = \emptyset$). Hence, X satisfies a constraint iff it does not satisfy its body.

Definition 2 (Answer set [7]). *Let Π be a program and X a set of atoms. We say that X is an* answer set *of Π if X is the minimal set (in the sense of set inclusion) that satisfies Π^X, where Π^X is obtained as follows:*

- *delete all rules whose bodies are not satisfied by X;*
- *delete* not b_i *in the bodies of the remaining rules.*

Gelfond and Lifschitz [8] showed that answer set programming is a special case of default logic by simply rewriting a rule of form (6) to the following default:

$$b_1 \wedge \dots \wedge b_m : \neg b_{m+1}, \dots, \neg b_n / a. \tag{7}$$

Let Π be a program. By $DL(\Pi)$, we denote the default theory obtained from Π as above (Note that W in $DL(\Pi)$ is obtained by facts in Π). Then, the answer sets of Π and the extensions of $DL(\Pi)$ are one-to-one corresponded.

Theorem 1 (From ASP to DL [8]). *Let Π be a program and X a set of atoms. Then, X is an answer set of Π iff $Th(X)$ is an extension of $DL(\Pi)$.*

An interesting question arises whether the converse of Theorem 1 holds as well. In other words, is it possible to translate default logic back into answer set programming? In the next section, we answer it positively.

3 From Default Logic to Answer Set Programming

This section translates Reiter's default logic in propositional case to answer set programming. To begin with, let us take a closer look at the translation from ASP to DL, which. In fact, this translation indicates that the rule of form (6) in ASP plays the same role of the default (7) in DL. This means that, conversely, a specific kind of default, namely of form (7) plays the same role to an ASP rule, namely of form (6). Analogously, its suggests that a default of form (1) in DL, i.e.

$$\alpha : \beta_1, \dots, \beta_n / \gamma,$$

should play a similar role to

$$\gamma \leftarrow \alpha, \mathsf{not}\ \neg\beta_1, \dots, \mathsf{not}\ \neg\beta_n.$$

However, this is not exactly an ASP rule. To fix this problem, we can simply introduce a new atom p_α for each formula α. Then, analogous to the translation from ASP to DL, a default of form (1) in DL should play a similar role to the following rule in ASP:

$$p_\gamma \leftarrow p_\alpha, \mathsf{not}\ p_{\neg\beta_1}, \dots, \mathsf{not}\ p_{\neg\beta_n}. \tag{8}$$

Now, we have a naive translation from DL to ASP. Formally, let $\Delta = \langle W, D\rangle$ be a default theory. We introduce a set of new atoms p_α for each formula $\alpha \in W \cup P_\Delta \cup \neg J_\Delta \cup C_\Delta$.[2] Let r be a default of form (1), by $R(r)$, we denote the ASP rule of form (8). Let $\Delta = \langle W, D\rangle$ be a default theory, by $R(\Delta)$, we denote the program

$$\{p_\alpha \mid \alpha \in W\} \cup \{R(r) \mid r \in D\}.$$

Indeed, $R(\Delta)$ is the answer set program obtained from the default theory Δ by mapping each formula occurred in the default theory to a corresponding new atom.

Example 2. Recall Example 1. According to the above definition, $R(\Delta_1)$ contains the following rules:

$$\begin{array}{ll}
p_{\neg b\vee\neg c} \leftarrow & p_b \leftarrow \mathsf{not}\ p_a, \mathsf{not}\ p_c \\
p_{c\vee d} \leftarrow & p_{\neg d} \leftarrow \mathsf{not}\ p_{\neg(a\wedge\neg b)} \\
p_a \leftarrow \mathsf{not}\ p_b & p_{\neg a} \leftarrow p_{\neg c}, \mathsf{not}\ p_a.
\end{array}$$

It can be checked that $R(\Delta_1)$ has two answer sets: $M_1 = \{p_{\neg b\vee\neg c}, p_{c\vee d}, p_a, p_{\neg d}\}$ and $M_2 = \{p_{\neg b\vee\neg c}, p_{c\vee d}, p_b, p_{\neg d}\}$. Compared to the extensions of Δ_1, while M_1 exactly corresponds to E_1, M_2 and E_2 are not related.

Let us take a closer look at Examples 1 and 2. One may observe that there is no extension of Δ_1 containing b when $\neg d$ holds because $W_1 \cup \{\neg d\} \models c$, thus default (3) in Δ_1 will never be triggered. This is the reason why M_2 fails to correspond to any extension of Δ_1. Also, suppose that b is in an extension of Δ_1. Then, $\neg c$ must be in the extension as well because $W \cup \{b\} \models \neg c$. As such, default (5) in Δ_1 could be triggered so that $\neg a$ is also in the extension. This explains why there is no answer sets of $R(\Delta_1)$ corresponding to E_2.

In general, we can conclude that what is missing in $R(\Delta)$ are the internal relationships among formulas in Δ. By simply translating Δ to $R(\Delta)$, some internal relationships among formulas are lost, which might be crucial for computing a default theory's extensions, for instance, the entailment relationship $W_1 \cup \{\neg d\} \models c$ as discussed above.

Our main theoretical result in this paper is: together with the internal relationships, $R(\Delta)$ exactly captures the extensions of Δ. Formally, let Δ be a

[2] We use $\neg J_\Delta$ to denote the set of formulas $\{\neg\phi \mid \phi \in J_\Delta\}$.

default theory and $F_\Delta = W \cup P_\Delta \cup \neg J_\Delta \cup C_\Delta$. The set of *implication rules* of Δ, denoted by $I(\Delta)$, is the set of rules of the form

$$p_\phi \leftarrow p_{\phi_1}, \ldots, p_{\phi_n} \tag{9}$$

where $\{\phi, \phi_1, \ldots \phi_n\} \subseteq F_\Delta$, $\phi \notin \{\phi_1, \ldots, \phi_n\}$ and $\{\phi_1, \ldots, \phi_n\} \models \phi$.

Finally, let $AS(\Delta) = R(\Delta) \cup I(\Delta)$. The following theorem shows that $AS(\Delta)$ exactly captures the extensions of Δ. That is, the answer sets of $AS(\Delta)$ is one-to-one corresponding to the extensions of Δ.

Theorem 2 (From DL to ASP). *Let Δ be a default theory and T a consistent theory.*[3] *Then, T is an extension of Δ iff p_T is an answer set of $AS(\Delta)$, where $p_T = \{p_\alpha \mid \alpha \in F_\Delta, T \models \alpha\}$.*

Proof (sketch). [4] Firstly, it is observed that if T is an extension of Δ, then there exists $F \subseteq F_\Delta$ such that T is the deductive closure of $W \cup F$.

Now, suppose that T is an extension of Δ. Then, p_T satisfies all rules in $AS(\Delta)$ according to the definition. Thus, p_T satisfies $AS(\Delta)^{p_T}$. Assume that p_T is not an answer set of $AS(\Delta)$. Then, there exists $X \subset p_T$ such that X satisfies $AS(\Delta)^{p_T}$ as well. Let T' be the deductive closure of $\{\alpha \mid p_\alpha \in X\}$. Then, it can be checked that T' satisfies the conditions of the operator Γ with respect to T (note that X satisfies $I(\Delta)$). Hence, $\Gamma(T) \subseteq T'$. In addition, $T' \subset T$ since $X \subset p_T$. This shows that $\Gamma(T) \subseteq T' \subset T$, a contradiction.

On the other hand, suppose that p_T is an answer set of $AS(\Delta)$. Then, p_T satisfies each rule in $R(\Delta)$. Therefore, T satisfies the conditions of the operator Γ with respect to T itself. Hence, $\Gamma(T) \subseteq T$. Assume that $\Gamma(T) \subset T$. Then, $p_{\Gamma(T)}$ satisfies $R(\Delta)^{p_T}$ according to the definition of $R(\Delta)$. Also, $p_{\Gamma(T)}$ satisfies $I(\Delta)^{p_T}$ according to the definitions of $I(\Delta)$ and $p_{\Gamma(T)}$. Hence, $p_{\Gamma(T)}$ satisfies $AS(\Delta)^{p_T}$. In addition, $p_{\Gamma(T)} \subset p_T$ since $\Gamma(T) \subset T$. This shows that p_T is not an answer set of $AS(\Delta)$, a contradiction.

Although $AS(\Delta)$ exactly captures the extensions of Δ, the implication rules in $I(\Delta)$ could be a lot. Next, we propose several techniques to reduce the number of implication rules by the following observations:

- In ASP, suppose that there are two rules sharing the same head, the same negative body but one rule's positive body is a subset of another's. Then, the latter rule is "dummy" because these two rules are strongly equivalent to the former one [15]. Hence, we can only consider those "minimal" implication rules for a default theory Δ.
- In DL, given a default theory $\Delta = \langle W, D\rangle$, all the extensions must contain W. Hence, we can fix W for the implication rules. That is, we can only consider those internal relationships (i.e. implication rules) generated from the set $P_\Delta \cup \neg J_\Delta \cup C_\Delta$ under the context of W.

[3] Here, we only consider consistent extensions. Inconsistency can be easily checked in default logic [22].

[4] Due to a space limit, proofs in this paper, if given, are sketched.

- Suppose that there exists an inconsistent (unsatisfiable) set of formulas. Then, for any other ϕ, it generates an implication rule. Of course, this is not necessary. All we need is a constraint stating that these formulas (their corresponding atoms) cannot appear at the same time.
- Observed from Theorem 2.5 [21], all the extensions of a default theory Δ can be rewritten as $Th(W \cup C)$, where C is a subset of C_Δ. Hence, we only need to consider the implication rules whose heads are only from $P_\Delta \cup \neg J_\Delta$ and bodies are only from C_Δ.

Based on the above observations and discussions, we can simplify the implication rules as follows. Let $\Delta = \langle W, D \rangle$ be a default theory. The set of *modified implication rules* of Δ, denoted by $I^*(\Delta)$, is the set of rules of form (9), where

- either ϕ is $\bot$,[5] $\phi_i \in C_\Delta$, $W \cup \{\phi_1, \ldots, \phi_n\}$ is unsatisfiable, and $\{\phi_1, \ldots, \phi_n\}$ is the minimal set satisfying the above conditions,
- or $\phi \in P_\Delta \cup \neg J_\Delta$, $\phi_i \in C_\Delta \setminus \{\phi\}$, $W \cup \{\phi_1, \ldots, \phi_n\}$ is satisfiable, $W \cup \{\phi_1, \ldots, \phi_n\} \models \phi$, and $\{\phi_1, \ldots, \phi_n\}$ is the minimal set satisfying the above conditions.

Example 3. Recall Δ_1 discussed in Examples 1 and 2 again. According to the definition, $I^*(\Delta)$ is the set of the following rules:

$$\begin{array}{ll} \bot \leftarrow p_a, p_{\neg a} & p_c \leftarrow p_{\neg d} \\ \bot \leftarrow p_b, p_{\neg d} & p_{\neg(a \wedge \neg b)} \leftarrow p_b \\ p_{\neg c} \leftarrow p_b & p_{\neg(a \wedge \neg b)} \leftarrow p_{\neg a} \end{array}$$

It can be checked that $R(\Delta) \cup I^*(\Delta)$ has two answer sets: $\{p_{\neg b \vee \neg c}, p_{c \vee d}, p_a, p_{\neg d}\}$ and $\{p_{\neg b \vee \neg c}, p_{c \vee d}, p_{\neg a}, p_b\}$, which are exactly corresponding to the extensions E_1 and E_2 of Δ_1 respectively.

In general, let $AS^*(\Delta) = R(\Delta) \cup I^*(\Delta)$. The following theorem shows that $AS^*(\Delta)$ is enough to capture the extensions of Δ.

Theorem 3. *Let Δ be a default theory and T a consistent theory. Then, T is an extension of Δ iff p^*_T is an answer set of $AS^*(\Delta)$, where $p^*_T = \{p_\alpha \mid \alpha \in W \cup C_\Delta, T \models \alpha\}$.*

Clearly, the atoms used in $AS^*(\Delta)$ is linear with respect to the size of Δ. However, although the number of implication rules in $I^*(\Delta)$ is significantly reduced compared to $I(\Delta)$, there might be exponential number of such rules. It is well-known that checking whether a default theory in the propositional case has an extension is Σ_2^P complete [10], while checking whether a normal logic program has an answer set is NP complete [4]. Hence, the exponential size seems inevitable, providing some general assumptions in the complexity theory.

However, as we will show later in the experiments, the number of implication rules is not really explosive. This is also partially evidenced by the research of

[5] In this case, we define $p_\bot$ as $\bot$ for convenience.

minimal unsatisfiable subset. For instance, the experiments in [14] showed that the number of MUS (corresponding to minimal implication rules in $I^*(\Delta)$) is not big for many cases. Also, although simple, it is worth mentioning that if the default theory Δ itself is ASP-like (i.e. $W \cup C_\Delta \cup \neg J_\Delta \cup P_\Delta$ only contains atoms), then $I^*(\Delta)$ is empty.

A closely related work is due to Dao-Tran et al. [20] for translating default logic to so-called description logic knowledge bases, which is an extension of ASP with description logics. Restricted in the propositional case, while this work is to translate DL to an extension of ASP, ours directly translates DL to ASP itself.

4 Implementation

Based on Theorems 2 and 3, we have implemented a new solver for default logic, called dl2asp, which computes all extensions of a given default theory. More precisely, the key idea of dl2asp is to translate a given default theory Δ to $AS^*(\Delta)$. Then, by Theorem 3, the task of computing all the extensions of Δ turns into computing all the answer sets of $AS^*(\Delta)$.

A default theory is encoded in an ASCII file in dl2asp. For example, the default theory Δ_1 in Example (1) is encoded as follows:

```
-b | -c                          c | d
: -b / a                         : -a, -c / b
: a & -b / -d                    -c : -a / -a
```

Here, "-", "|" and "&" stand for the connectives $\neg$, $\vee$ and $\wedge$ respectively.

Figure 1 illustrates how dl2asp works. An input default theory is firstly translated to an answer set program by the *translator* in dl2asp. Then, an *ASP solver* is called to compute the answer sets of the program. Finally, the answer sets will be interpreted back to extensions of the original default theory by a *convertor*.

For the *ASP solver* module in dl2asp, we just use *clasp*[6]. The *convertor* in dl2asp is trivial. As demonstrated in Theorem 3, one can simply interpret each atom in the answer sets to their corresponding formulas. Hence, the main issue in dl2asp is the *translator*.

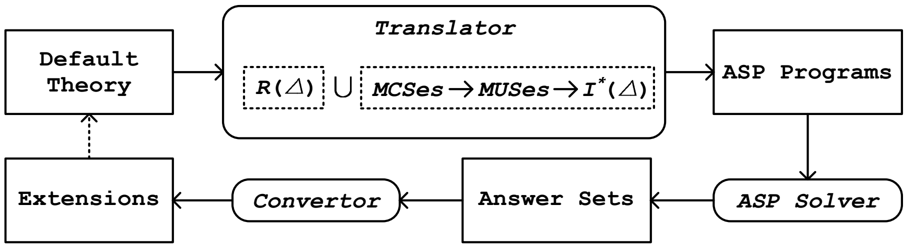

Fig. 1. Outline of dl2asp

[6] http://www.cs.uni-potsdam.de/clasp/

Let $\Delta = \langle W, D\rangle$ be a default theory. According to the construction, $AS^*(\Delta)$ contains two parts, namely $R(\Delta)$ and $I^*(\Delta)$. For $R(\Delta)$, we can first introduce a new atom p_α for each formula $\alpha \in F_\Delta$, then get a new fact p_α for each formula $\alpha \in W$ and get a new rule of form (8) for each default in D of form (1). However, $I^*(\Delta)$ is relatively difficult. The most technical part of dl2asp is to compute $I^*(\Delta)$ - the set of modified implication rules.

For this purpose, we borrow some ideas and techniques of computing minimal unsatisfiable subsets from [14]. Let S be a set of formulas. A subset S' of S is a *Minimal Unsatisfiable Subset (MUS)* if S' is unsatisfiable and for any $S'' \subset S'$, S'' is satisfiable. Liffiton and Sakallah [14] developed a sound and complete algorithm, called CAMUS[7], for computing all MUSes of a given set of clauses. However, instead of computing MUSes directly, they computed so-called MCSes first. Here, a subset S' of S is a *Maximal Correction Subset (MCS)* if $S \backslash S'$ is satisfiable and for any $S'' \subset S'$, $S \backslash S''$ is unsatisfiable. MCS and MUS are closely related. In fact, all the MUSes are exactly all the minimal hitting sets of the collection of all MCSes. Here, given a collection of sets Ω, a set H is a *minimal hitting set* of Ω iff for all $H_0 \in \Omega$, $H \cap H_0 \neq \emptyset$ and there is no $H' \subset H$ satisfying the above condition.

We now apply this method to compute all the modified implication rules (i.e. $I^*(\Delta)$) in our translation. Let Δ be a default theory. According to the construction, $I^*(\Delta)$ contains rules of the form $p_\phi \leftarrow p_{\phi_1}, \ldots, p_{\phi_n}$, where $\{p_{\phi_1}, \ldots, p_{\phi_n}\}$ is a minimal set such that $W \cup \{\phi_1, \ldots, \phi_n\} \models \phi$. That is, it is a minimal set inconsistent with $W \cup \{\neg\phi\}$. In other words, it is an MUS of the set $W \cup \{\neg\phi\} \cup C_\Delta$ by fixing W and $\{\neg\phi\}$. Hence, we can compute all the implication rules in $I^*(\Delta)$ as follows:

constraints compute all the MUSes of $W \cup C_\Delta$ by fixing W;
other implication rules for all $\phi_i \in \neg J_\Delta \cup P_\Delta$, compute all the MUSes of $W \cup \{\neg\phi_i\} \cup C_\Delta$ by fixing W and $\{\neg\phi_i\}$.

There are two major differences between this task and the one in [14] for computing all MUSes. Firstly, we need to fix some formulas, e.g. formulas in W. Secondly, we need to compute the MUSes for every ϕ in $P_\Delta \cup \neg J_\Delta$. Clearly, the task of computing constraints can be considered as a special case of the second case. In dl2asp, we first compute constraints, then the other implication rules, where the latter is implemented by Algorithm 1.

Let use take a closer look at Algorithm 1. Step 1 fixes W as a background formula set. Steps 2-4 (Steps 5-6) add a selector variable x_i (y_j) to the formula $\neg\psi_i$ (ϕ_j resp.). The *AtMost* constraint in Step 7 is provided by MiniSat [5] to restrict that at most one of $\{x_1, \ldots, x_m\}$ holds, and together with Step 8, exactly one of $\{\neg\psi_1, \ldots, \neg\psi_m\}$ holds. In fact, x_i is corresponding to those implication rules whose head is p_{ψ_i}. Steps 9-16 compute all the MCSes in a similar way to Figure 2 in [14] except that W and at most one of ψ_i are fixed. *SAT* in Step 10 and *IncrementalSAT* in Step 12 uses MiniSat's incremental solving ability to find a model of a formula. *BlockingClause* in Steps 14 and 15 is pro-

[7] http://www.eecs.umich.edu/~liffiton/camus/

vided by CAMUS to block the MCS that obtained before. Steps 17-20 compute all the implication rules in the same way to computing MUSes in [14], where *ConstructMUS* is provided by CAMUS for computing all minimal hitting sets, and *Rewrite* is the function to rewrite an MUS to an implication rule of form (9), and delete those self implication rules like $p_\phi \leftarrow p_\phi$ and those implication rules like $p_\phi \leftarrow p_{\phi_1}, \ldots, p_{\phi_n}$ when there is already a constraint $\bot \leftarrow p_{\phi_1}, \ldots, p_{\phi_n}$.

Algorithm 1. Computing Implication Rules

```
input  : A default theory $\Delta = \langle W, D \rangle$, where $C_\Delta = \{\phi_1, \ldots, \phi_n\}$ and
         $P_\Delta \cup \neg J_\Delta = \{\psi_1, \ldots, \psi_m\}$
output: the set of non-constraint implication rules of $\Delta$
$\Phi \leftarrow W$                                    // set $W$ as background theory
forall $1 \le i \le m$ do                            // add selector variables to $\neg\psi$
    $MCS(\neg\psi_i) \leftarrow \emptyset$
    $\Phi \leftarrow \Phi \cup \{\neg\psi_i \vee \neg x_i\}$
forall $\phi_j$ such that $1 \le j \le n$ do       // add selector variables to $\phi$
    $\Phi \leftarrow \Phi \cup \{\phi_j \vee \neg y_j\}$
$\Phi \leftarrow \Phi \cup AtMost(\{x_1, \ldots, x_m\}, 1)$        // at most one of $x_i$ holds
$\Phi \leftarrow \Phi \cup \{x_1 \vee \cdots \vee x_m\}$             // at least one of $x_i$ holds
$k \leftarrow 1$
while $(SAT(\Phi))$ do                                   // computing $MCSes$
    $\Phi_k \leftarrow \Phi \cup AtMost(\{\neg y_1, \ldots, \neg y_n\}, k)$
    while $(M = IncrementalSAT(\Phi_k))$ do
        $MCS(\neg\psi_i) \leftarrow MCS(\psi_i) \cup \{\phi_j \mid y_j \in M\}$
        $\Phi_k \leftarrow \Phi_k \cup BlockingClause(M)$
        $\Phi \leftarrow \Phi \cup BlockingClause(M)$
        $k \leftarrow k + 1$
$I^*(\Delta) \leftarrow \emptyset$
forall $1 \le i \le m$ do                            // computing implication rules
    $I^*(\Delta) \leftarrow I^*(\Delta) \cup Rewrite(ConstructMUS(\psi_i))$
return $I^*(\Delta)$
```

Example 4. Again, recall Δ_1 discussed in Examples 1 and 2, and consider the formula $\neg(a \wedge \neg b) \in \neg J_\Delta$. According to Algorithm 1, we have $MCS(\neg(a \wedge \neg b)) = \{\{b, \neg a\}\}$ and $MUS(\neg(a \wedge \neg b)) = \{\{b\}, \{\neg a\}\}$. So, we have two rules $p_{\neg(a \wedge \neg b)} \leftarrow p_b$ and $p_{\neg(a \wedge \neg b)} \leftarrow p_{\neg a}$ in $I^*(\Delta_1)$ corresponding to the result above.

Finally, we end up this section by showing how the rest parts of dl2asp work for the example. By calling *clasp*, $R(\Delta_1) \cup I^*(\Delta_1)$ has two answer sets: $\{p_{\neg b \vee \neg c}, p_{c \vee d}, p_a, p_{\neg d}\}$ and $\{p_{\neg b \vee \neg c}, p_{c \vee d}, p_{\neg a}, p_b\}$. Then, the *convertor* just rewrite them as: $\{\neg b \vee \neg c, c \vee d, a, \neg d\}$ and $\{\neg b \vee \neg c, c \vee d, \neg a, b\}$. In contrast with Example 1, these two formula sets are exactly the two extensions of Δ_1.

5 Experimental Results

In this section, we report some experimental results of dl2asp. First, we briefly review some existing solvers for default logic, including DeReS, XRay, and GADEL. Unfortunately, we are unable to run them properly because these solvers are out of date, and the versions of softwares they used are too old to be compatible with current versions.

Nevertheless, it is still necessary to review these approaches, particularly their test data. The task of all these approaches is to compute all extensions of a given default theory. For this purpose, DeReS [3] directly uses search algorithms to compute the extensions. In [3], some benchmarks in graph theory, e.g. finding Hamiltonian circuit, are tested. Instead, XRay [18] is implemented by using Prolog, and supporting local proof procedures. XRay tested the Hamiltonian circuit problem and some contextual default theories randomly generated. GADEL [16] applies genetic algorithms to compute the extensions. GADEL tested the Hamiltonian circuit problem as well as a handed-coded default theory about relationships among people (Example 4.1, [16]).

Similarly, dl2asp intends to find all extensions of a given default theory as well. First of all, it is observed that if the default theory is ASP-like (i.e. $W \cup C_\Delta \cup \neg J_\Delta \cup P_\Delta$ only contains atoms), then the set of implication rules is empty. Furthermore, if the default theory is disjunction-free (i.e. all the formulas occurred in the default theory are literals), then all the implication rules are constraints, and the total number is linear. This is because the implication rules can only be of the form $\bot \leftarrow p_l, \mathsf{not}\ p_{\neg l}$, where l is a literal occurred in the default theory and $\neg l$ is the complementary literal of l. In both cases, Algorithm 1 can find out all the implication rules immediately. Hence, it is not interesting to test such default theories for dl2asp.

However, most of the test benchmarks, e.g. the Hamiltonian circuit problem, belong to the above two categories. Therefore, in this section, we only report our experimental results for solving the people's relationship problem[8] (Example 4.1 in [16]), which is claimed difficult to be solved in default logic.

The system dl2asp is written in C++. The program is running on a machine with 4 processors($AMD\ Athlon^{tm}\ II\ X4$ 620) under Ubuntu 9.10 Linux operating system. We record two series of time in seconds, $time_t$ - the time for computing all implication rules and $time_{all}$ - the overall time, taking the average of 3 runs. num_{I^*} is the number of the rules in $I^*(\Delta_\mathcal{P})$. The experimental results of the people's relationship problem are summarized in Table 1.

Although we do not intend to compare dl2asp with other solvers on this particular instance because of fairness reasons, we can see that dl2asp performs rather satisfactory on this benchmark. As an example, for $woman \wedge student$, while dl2asp only takes 0.3 seconds, GADEL and DeRes takes 1202 seconds and more than 7200 seconds respectively under their test environments (see Table 2 in [16]).

[8] For more details about this particular default theory, please refer to [16].

Table 1. Experimental results of the people's relationship problem

	boy	*girl*	*man*	*woman*	*man* $\wedge$ *student*	*woman* $\wedge$ *student*
num_{I^*}	10	10	11	12	10	11
$time_t$	0.5134	0.5134	0.6427	0.7001	0.2453	0.2894
$time_{all}$	0.5227	0.5228	0.6534	0.7120	0.2520	0.2987

6 Application to Fair Division Problem

In this section, we apply dl2asp for solving the fair division problem, which is one of the central problems in social choice theory. The problem of fair division is: given a set of agents, a set of goods and the preference among goods for each agent, to obtain a "fair" solution for allocating the goods to the agents [2].

Formally, a *fair division problem* is a tuple $\mathcal{P} = \langle I, X, \mathcal{R} \rangle$, where $I = \{1, \ldots, N\}$ is a set of agents, $X = \{x_1, \ldots, x_p\}$ is a set of indivisible goods, and $\mathcal{R} = \{R_1, \ldots, R_N\}$ is a preference profile, where each R_i is a reflexive, transitive and complete relation on 2^X. An *(complete) allocation* for $\mathcal{P} = \langle I, X, \mathcal{R} \rangle$ is a mapping $\pi : I \to 2^X$ such that for all i and $j \neq i$, $\pi(i) \cap \pi(j) = \emptyset$, and for every $x \in X$ there exists an i such that $x \in \pi(i)$. An allocation π is *(Pareto-) efficient* iff there is no π' such that π' dominates π, where for two allocations π and π', π dominates π' iff for all i, $(\pi(i), \pi'(i)) \in R_i$, and there exists an i such that $(\pi'(i), \pi(i)) \notin R_i$. An allocation π is *envy-free* iff $(\pi(i), \pi(j)) \in R_i$ holds for all i and all $j \neq i$.

A preference R_i is *dichotomous* iff there exists a subset $Good_i$ of 2^X such that for all $A, B \subseteq X$, $(A, B) \in R_i$ iff $A \in Good_i$ or $B \notin Good_i$. A dichotomous preference can be naturally represented by a single propositional formula, where variables correspond to goods. Given a dichotomous preference R_i, we can always use formula $\phi_i = \bigvee_{A \in Good_i} (\bigwedge_{x \in A} x \wedge \bigwedge_{x \notin A} \neg x)$ to represent R_i. Thus, a fair division problem with dichotomous preference $\mathcal{P} = \langle I, X, \mathcal{R} \rangle$ can always be represented by $\langle \phi_1, \ldots, \phi_N \rangle$, and I, X and $\mathcal{R}$ are obviously determined from $\langle \phi_1, \ldots, \phi_N \rangle$. The following proposition shows that for fair division problem with dichotomous preference, it can be translated into default logic.

Proposition 1 (Proposition 3, [2]). *Let $\mathcal{P} = \langle \phi_1, \ldots, \phi_N \rangle$ be a fair division problem. Let $\Delta_\mathcal{P}$ be the default theory $\langle \Gamma_\mathcal{P}, \Phi_\mathcal{P} \cup \{\neg \Lambda_\mathcal{P} : / \bot\} \rangle$, where*

- $\Gamma_\mathcal{P} = \bigwedge_{x \in X} \bigwedge_{i \neq j} \neg(x_i \wedge x_j)$,
- $\Lambda_\mathcal{P} = \bigwedge_{i=1,\ldots,N} \left[\phi_i^* \vee \left(\bigwedge_{j \neq i} \neg \phi_{j|i}^* \right) \right]$, *and*
- $\Phi_\mathcal{P}$ *is the set of defaults of the form* $: \phi_i^* / \phi_i^*$, $i = 1, \ldots, N$,

where ϕ_i^* is obtained from ϕ_i by replace every variable x by a new symbol x_i, and $\phi_{j|i}^*$ is obtained from ϕ_i^* by replace every symbol x_i by x_j. Then, each extension of $\Delta_\mathcal{P}$ is corresponding to an efficient and envy-free allocation of $\mathcal{P}$.

Based on this result, we are able to use dl2asp to compute all efficient and envy-free allocations for a given fair division problem. Given the numbers of goods and agents, we randomly generate a fair division problem instance, then

Table 2. Experimental results of Fair Division Problem

$(goods, agents)$	$(3,4)$	$(4,4)$	$(4,6)$	$(4,8)$	$(4,10)$	$(4,15)$	$(4,20)$	$(4,25)$
num_{I^*}	1	1	1	1	14	21	410	460
$time_t$	0.0213	0.0267	0.0653	0.2213	0.4427	1.7534	21.8267	33.2407
$time_{all}$	0.0333	0.0547	0.1000	0.2787	0.5320	1.9148	22.2241	33.8248
EXT	0	0	0	0	5	13	170	177

compute all the extensions of the default theory according to Proposition 1. Our experimental results are shown in Table 2, where EXT is the number of extensions returned by dl2asp and num_{I^*}, $time_t$ and $time_{all}$ are the same as those in Table 1.

7 Conclusions and Future Work

This paper contributes the study of default logic both from a theoretical and a practical point of view. Theoretically, we showed that default logic can be translated into answer set programming by identifying the internal relationships (i.e. implication rules) among formulas (Theorem 2), and indeed, the number of such implication rules can be largely reduced (Theorem 3). Practically, based on the above translation, we developed a new solver - dl2asp - for implementing default logic via answer set programming. Our experimental results (Table 1) illustrated that the performance of dl2asp is rather satisfactory, which is further confirmed by applying dl2asp to solving the fair division problem (Table 2).

One of our future work is to consider the relationships between default logic and disjunctive answer set programming as they are on the same complexity level. For instance, an open problem is whether DL can be naturally translated to disjunctive ASP, or the other way around. If not, then an interesting question arises, for a particular application on the Σ_2^P level, whether it is better to be encoded in DL or in disjunctive ASP.

For other future directions, one important task is to develop more techniques for improving dl2asp, especially for computing the implication rules. Another work worth pursuing is to extend dl2asp for more expressive default logics, such as disjunctive default logic [9] and general default logic [22]. Last but not least, it is interesting to explore more applications of default logic by using dl2asp.

Acknowledgments

We would like to thank Jérôme Lang for his generous help on the fair division problem, and the anonymous reviewers for their valuable comments. This research is supported in part by an Australian Research Council Linkage Project grant LP0883646, and the first author is also partially supported by Natural Science Foundation of China under grant NSFC60705095 and Natural Science Foundation of GuangDong Province, China under grant GDSF07300237.

References

1. Bochman, A.: Default logic generalized and simplified. Ann. Math. Artif. Intell. 53(1-4), 21–49 (2008)
2. Bouveret, S., Lang, J.: Efficiency and envy-freeness in fair division of indivisible goods: Logical representation and complexity. JAIR 32, 525–564 (2008)
3. Cholewinski, P., Marek, V.W., Truszczynski, M., Mikitiuk, A.: Computing with default logic. Artificial Intelligence (AIJ) 112(1-2), 105–146 (1999)
4. Dantsin, E., Eiter, T., Gottlob, G., Voronkov, A.: Complexity and expressive power of logic programming. ACM Comput. Surv. 33(3), 374–425 (2001)
5. Eén, N., Sörensson, N.: An extensible SAT-solver. In: Giunchiglia, E., Tacchella, A. (eds.) SAT 2003. LNCS, vol. 2919, pp. 502–518. Springer, Heidelberg (2004)
6. Gebser, M., Kaufmann, B., Neumann, A., Schaub, T.: Conflict-driven answer set solving. In: IJCAI, pp. 386–392 (2007)
7. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: ICLP/SLP, pp. 1070–1080 (1988)
8. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Comput. 9(3/4), 365–386 (1991)
9. Gelfond, M., Przymusinska, H., Lifschitz, V., Truszczynski, M.: Disjunctive defaults. In: KR, pp. 230–237 (1991)
10. Gottlob, G.: Complexity results for nonmonotonic logics. Journal of Logic and Computation 2, 397–425 (1992)
11. Janhunen, T.: On the intertranslatability of autoepistemic, default and priority logics, and parallel circumscription. In: Dix, J., Fariñas del Cerro, L., Furbach, U. (eds.) JELIA 1998. LNCS (LNAI), vol. 1489, pp. 216–232. Springer, Heidelberg (1998)
12. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The dlv system for knowledge representation and reasoning. ACM Trans. Comput. Log. 7(3), 499–562 (2006)
13. Lierler, Y., Maratea, M.: Cmodels-2: Sat-based answer set solver enhanced to non-tight programs. In: Lifschitz, V., Niemelä, I. (eds.) LPNMR 2004. LNCS (LNAI), vol. 2923, pp. 346–350. Springer, Heidelberg (2003)
14. Liffiton, M.H., Sakallah, K.A.: Algorithms for computing minimal unsatisfiable subsets of constraints. Journal of Automated Reasoning (JAR) 40(1), 1–33 (2008)
15. Lin, F., Chen, Y.: Discovering classes of strongly equivalent logic programs. J. Artif. Intell. Res. (JAIR) 28, 431–451 (2007)
16. Nicolas, P., Saubion, F., Stéphan, I.: Gadel: a genetic algorithm to compute default logic extensions. In: ECAI, pp. 484–490 (2000)
17. Nicolas, P., Saubion, F., Stéphan, I.: Heuristics for a default logic reasoning system. IJAIT 10(4), 503–523 (2001)
18. Nicolas, P., Schaub, T.: The xray system: an implementation platform for local query-answering in default logics. In: Hunter, A., Parsons, S. (eds.) Applications of Uncertainty Formalisms. LNCS (LNAI), vol. 1455, pp. 354–378. Springer, Heidelberg (1998)
19. Niemelä, I., Simons, P.: Smodels - an implementation of the stable model and well-founded semantics for normal lp. In: Fuhrbach, U., Dix, J., Nerode, A. (eds.) LPNMR 1997. LNCS, vol. 1265, pp. 421–430. Springer, Heidelberg (1997)
20. Dao-Tran, M., Eiter, T., Krennwallner, T.: Realizing Default Logic over Description Logic Knowledge Bases. In: Sossai, C., Chemello, G. (eds.) ECSQARU 2009. LNCS, vol. 5590, pp. 602–613. Springer, Heidelberg (2009)
21. Reiter, R.: A logic for default reasoning. Artificial Intelligence (AIJ) 13(1-2), 81–132 (1980)
22. Zhou, Y., Lin, F., Zhang, Y.: General default logic. Annals of Mathematics and Artificial Intelligence (2010) (to appear)