

Constrained Default Logic Programming

Shutao Zhang¹, Zhizheng Zhang¹ and Jun Shen¹

¹*Southeast University, School of Computer Science and Engineering, Nanjing, Jiangsu, China*

Abstract

This paper develops a new formalism CDLP by combining ASP and constrained default logic to facilitate modeling questions with incomplete information, such that both Reiter's defaults and constraint defaults can be represented directly. After that, a method to split the CDLP programs is proposed by extending the notion of splitting sets to accommodate its property of constraint nonmonotonicity. Finally, we design a primary algorithm for solving the CDLP programs.

Keywords

answer set programming, constrained default logic, logic program splitting

1. Introduction

Answer Set Programming (ASP) [1, 2, 3, 4, 5] and Default Logic (DL) [6, 7] are two well-known paradigms for knowledge representation and nonmonotonic reasoning with incomplete information. *Negation as failure* represents incomplete information in ASP programs based on the stable model semantics of logic programs. In default logic, default rules extend the classical first-order logic to model some cases without complete information. ASP and DL have been widely researched for the last decades, and several variants have been developed [8, 7]. For example, Epistemic Logic Program [9], P-log [10], and LP_{MLN} [11] are extensions for ASP. Justified Default Logic [12], Rational Default Logic [13], Disjunctive Default Logic [14], and Constraint default logic [15] are variants of Reiter's default logic.

The relationship between ASP and DL has attracted much attention and related studies [16, 5, 17]. Some focus on converting default rules in Reiter's DL to ASP rules. Lifschitz [16] proposed translating a default rule into an ASP rule without disjunction and constraints. Gelfond et al. [5] proposed a method to represent defaults in ASP programs with additional literals for abnormal individuals. However, so far as we know, no ASP approach is proposed to represent defaults defined in Reiter's DL variants, which are considered to have some natural features such as joint consistency. During our practice, we found that the domain specialists can hardly understand some subtle features of ASP, for example, the usage of constraint rules, negation as failure, and strong negations. The specialists believe that joint consistency is a natural property of the description of their domain knowledge. As a result, they tend to use constraint rules to deny some beliefs rather than a model filtering mechanism. We illustrate it by a *Multi-sport Athlete* problem below.

15th Workshop on Answer Set Programming and Other Computing Paradigms July 31, 2022

✉ shutao_zhang@seu.edu.cn (S. Zhang); seu_zzz@seu.edu.cn (Z. Zhang); junshen@seu.edu.cn (J. Shen)

🆔 0000-0003-0853-9281 (S. Zhang); 0000-0001-9851-6184 (Z. Zhang); 0000-0002-7307-2887 (J. Shen)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

For example, an athlete plans to run both the marathon and the relay race in a game if possible. It can be described by the following set D_1 of default rules:

$$\left\{ \begin{array}{l} : \textit{marathon} \\ \textit{marathon} \end{array} \right\}, \left\{ \begin{array}{l} : \textit{relay} \\ \textit{relay} \end{array} \right\}$$

The default theory (D_1, \emptyset) has a consistent extension $Th(\{\textit{marathon}, \textit{relay}\})$. The following ASP program Π_1 is obtained from D_1 by the translation in [18] and [17].

$$\textit{marathon} \leftarrow \neg \sim \textit{marathon}. \quad (r_1)$$

$$\textit{relay} \leftarrow \neg \sim \textit{relay}. \quad (r_2)$$

Π_1 has a unique answer set $\{\textit{marathon}, \textit{relay}\}$. However, this method leads to the non-existence of stable models if the inconsistency is not explicitly declared by *strong negations* [19]¹. For example, assume the athlete has been informed that ‘‘The marathon and the relay race are scheduled simultaneously. Hence, he can not run both of them.’’ Then we naturally add a formula into the default theory (D_1, W_1) such that $W_1 = \{\textit{marathon} \wedge \textit{relay} \supset \perp\}$. Similarly, we add a new constraint rule $r_c : \leftarrow \textit{marathon}, \textit{relay}$. into the program Π_1 . The classical default theory (D_1, W_1) has no consistent extensions, and the corresponding ASP program $\Pi_1 \cup \{r_c\}$ is unsatisfiable. An alternative method for ASP modeling is adding the *Closed World Assumption* (CWA) rules. For example, the additional rules for CWA of Π_1 are

$$\sim \textit{marathon} \leftarrow \neg \textit{marathon}. \quad (r_3)$$

$$\sim \textit{relay} \leftarrow \neg \textit{relay}. \quad (r_4)$$

However, the knowledge in r_3 and r_4 is beyond the original description of this problem, and it is hard for programmers to determine if the closed-world assumption is appropriate for the scenario. In summary, constraints in ASP programs are aimed at filtering models. Before filtering, the other rules must generate the desired models in this program, which may lead to extra work and unexpected results for programmers.

This paper introduces an extension of ASP, CDLP (Constrained Default Logic Programming), such that constrained defaults can be handled in logic programming. Specifically, a new unary operator \mathbf{C} is added to ASP to represent defaults, such that $\mathbf{C}l$ intuitively means l is assumed to be consistent. By a new operator \mathbf{C} of assumption, we can create new desired models with constraint rules while the rest constraints can still work as filters. The rest of this paper is organized as follows. Firstly, we introduce the syntax and semantics of CDLP in Section 3. Secondly, we investigate the splitting of CDLP programs in Section 4. We illustrate that the semantics of CDLP has the property of constraint nonmonotonicity. To accommodate this property, we extend the notion of splitting sets to CDLP programs and present the splitting theorem for CDLP. After that, we propose a primary algorithm for solving CDLP programs using the generate-and-test approach in Section 5. Then, we analyze the relationship between CDLP and constrained default logic in Section 6. Finally, the related works of belief and assume operators are surveyed in Section 7.

¹By abuse of notation, in this paper, we let \neg and \sim denote negation as failure and strong negation, respectively.

2. Preliminaries

2.1. Constrained Default Logic

Constrained default logic (CDL) [15] is a variant of Default Logic [6]. The primary motivation of constrained default logic is to guarantee joint consistency of default justifications. A default rule is of the form

$$\frac{\alpha : \beta_1, \dots, \beta_n}{\gamma} \quad (1)$$

where $\alpha, \beta_1, \dots, \beta_n$ and γ are first-order sentences. α is called the prerequisite, β_i s the justifications, and γ the consequent.

Let (D, W) be a default theory, where D is a set of default rules, W a set of first-order sentences. The *constrained extension* of (D, W) is a pair of set of sentences $\langle E, C \rangle$, where E and C can be obtained by

1. $E_0 = C_0 = W$, and
2. for $i \geq 0$
 - $E_{i+1} = Th(E_i) \cup \{\gamma \mid \frac{\alpha : \beta_1, \dots, \beta_n}{\gamma} \in D, \alpha \in E_i, C \cup \{\beta_1, \dots, \beta_n, \gamma\} \not\vdash \perp\}$
 - $C_{i+1} = Th(C_i) \cup \{\gamma \wedge \beta_1 \wedge \dots \wedge \beta_n \mid \frac{\alpha : \beta_1, \dots, \beta_n}{\gamma} \in D, \alpha \in E_i, C \cup \{\beta_1, \dots, \beta_n, \gamma\} \not\vdash \perp\}$
3. $\langle E, C \rangle = \langle \bigcup_{i=0}^{\infty} E_i, \bigcup_{i=0}^{\infty} C_i \rangle$.

For a default rule $\frac{\alpha : \beta_1, \dots, \beta_n}{\gamma} \in D$ and a constrained extension $\langle E, C \rangle$, if $\alpha \in E$, and $\gamma, \beta_1, \dots, \beta_n \in C$, then $\gamma \in E$. As a result, E is the set of consequences of reasoning, C is the corresponding set containing all justifications that support E .

The difference between CDL and Reiter's DL is that CDL requires *joint consistency*, while it is not necessary for Reiter's default theories. It means that in a classical default theory, two applied default rules to a consistent extension may have contradicted justifications, and the justification of an applied default rule should be consistent with the result of reasoning. In a word, in a constrained default theory, the set of formulas in W , all justifications and consequents of applied rules to a constrained extension should be consistent, i.e.

$$W \cup \{\gamma, \beta_1, \dots, \beta_n \mid \frac{\alpha : \beta_1, \dots, \beta_n}{\gamma} \text{ is applied to } (E, C)\} \not\vdash \perp$$

For example, consider the default theory (D_1, W_1) in Section 1 as a constrained default theory. Since applying both default rules in D_1 would cause inconsistency, (D_1, W_1) has two CDL extensions, $\langle Th(\{marathon\}), Th(\{marathon\}) \rangle$ and $\langle Th(\{relay\}), Th(\{relay\}) \rangle$, which is precisely the result we expected for this problem.

2.2. Answer Set Programming

An ASP program [2] is a collection of rules of the form

$$l_1 \mid \dots \mid l_k \leftarrow l_{k+1}, \dots, l_m, \neg l_{m+1}, \dots, \neg l_n. \quad (2)$$

where l_i s ($1 \leq i \leq n$) are ground literals, \neg is the operator of negation as failure, $|$ is the operator of epistemic disjunction. Intuitively, $\neg l$ means “it is not believed that l is true,” and $l_1|l_2$ means “ l_1 is believed to be true or l_2 is believed to be true”. The left side of a rule r is called the head of r , and the right side is called the body of r . An ASP rule is called a *fact* if its body is empty and a *constraint* if its head is empty.

A constraint rule r_c means that the literals in $body(r_c)$ can not be satisfied by a model at the same time. For example, “there is no penguin that is capable of flying” means an entity can not be a penguin while it can fly, which can be represented by the following rule.

$$\leftarrow penguin(X), fly(X).$$

There are three possible causes of inconsistency in an ASP program [20, 21]:

1. *constraint rules*, which may eliminate all stable models of the program;
2. *incoherence*, which means every stable model contains at least a pair of opposite literals of the form p and $\sim p$;
3. *odd loops*, which means a loop in the dependency graph with an odd number of NAF on the edges that no consistent stable model can satisfy all rules in this loop.

The constraints can be used to eliminate strong negations in an ASP program. For an ASP program Π , a literal $\sim l$ can be eliminated by following steps:

1. replace all occurrences of $\sim l$ with a new literal l^- ,
2. adding the following constraint r_c into program Π

$$\leftarrow l, l^-.$$

An *odd loop* of negations [22] that depends on no other rules can also represent inconsistency and can be transformed into constraint rules [5]. For example, the following rules

$$\begin{aligned} p &\leftarrow \neg q, s. \\ q &\leftarrow \neg u, t. \\ u &\leftarrow \neg p. \end{aligned}$$

are equivalent to $\{\leftarrow \neg p, \neg q, \neg u, s, t.\}$.

3. Constrained Default Logic Programming

A CDLP program is a finite collection of rules of the form

$$l_1 | \dots | l_k \leftarrow e_1, \dots, e_m, d_1, \dots, d_n. \quad (3)$$

where l_i s are classical literals with or without strong negations, e_i s are extended literals of the form l or $\neg l$, d_i s are default literals of the form Ce or $\neg Ce$. The body of a rule r , denoted by $body(r)$, is defined as $\bigcup e_i \cup \bigcup d_i$, while the head of r , denoted by $head(r)$, is defined as $\bigcup l_i$. For a CDLP program Π , we use $defaults(\Pi)$ to denote the set of all default literals of the form

Ce in the language of Π , and $literals(\Pi)$ to denote the set of classical literals that occur in Π , i.e. $l \in literals(\Pi)$ iff. $\exists r \in \Pi$ such that $\{l, \neg l, Cl, \neg Cl, C\neg l, \neg C\neg l\} \cap body(r) \neq \emptyset$. The default literals provide an approach to express the constrained default theories in ASP programs.

Example 1 illustrates that default literals can be falsified by the inconsistency of both incoherence and constraints.

Example 1. Consider following ASP rules with defaults.

$$fly(X) \leftarrow bird(X), Cfly(X). \quad (r_1)$$

$$\sim fly(X) \leftarrow penguin(X). \quad (r_2)$$

$$\leftarrow fly(X), ostrich(X). \quad (r_3)$$

Intuitively speaking, rules r_2 and r_3 are two different ways to represent the negation of default literals $Cfly(X)$ with strong negations and constraints, respectively. Consider an instance tweety of penguin and bird, rule r_2 generates $\sim fly(tweety)$, and will cause inconsistency if $Cfly(tweety)$ is true. Additionally, consider an instance plucky of ostrich and bird, the constraint rule r_3 forbids any model containing $fly(plucky)$; thus $Cfly(plucky)$ will cause inconsistency. As a result, both $Cfly(tweety)$ and $Cfly(plucky)$ are falsified, and both $fly(tweety)$ and $fly(plucky)$ are false in the models of this program.

The satisfiability of a CDLP program is defined as follows.

Definition 1 (Satisfiability). A default interpretation of program Π is a pair of consistent literal sets $\langle X, Y \rangle$, where $X \subseteq Y \subseteq literals(\Pi)$. Let $\langle X, Y \rangle$ be a default interpretation of Π ,

- $\langle X, Y \rangle \models l$ iff $l \in X$ where l is a literal;
- $\langle X, Y \rangle \models \neg e$ iff $X \not\models e$ where e is an extended literal;
- $\langle X, Y \rangle \models Cl$ iff $l \in Y$;
- $\langle X, Y \rangle \models C\neg e$ iff $Y \not\models e$;
- $\langle X, Y \rangle \models \neg Ce$ iff $\langle X, Y \rangle \not\models Ce$;
- $\langle X, Y \rangle \models r$ iff $\exists e \in head(r) : \langle X, Y \rangle \models e$ or $\exists \phi \in body(r) : \langle X, Y \rangle \not\models \phi$, where r is a rule in Π , e is an extended literal in the head of r , ϕ is an extended literal or default literal;
- $\langle X, Y \rangle \models \Pi$ iff $\forall r \in \Pi : \langle X, Y \rangle \models r$.

A default interpretation of a CDLP program contains two parts X and Y , where X is the consequence of reasoning, and Y contains all the assumptions that X is based on.

Example 2. Consider following program Π_2 .

$$p \leftarrow s. \quad (r_1)$$

$$q \leftarrow Cs. \quad (r_2)$$

The assumption of s will not cause inconsistency because it does not mean s is proved. Thus a model of Π_2 that satisfies Cs does not necessarily satisfy s . A default interpretation of Π_2 is $M = \langle \{q\}, \{q, s\} \rangle$, and $M \models Cr$ while $M \not\models s$.

d	$\langle X, Y \rangle \models d$	$\langle X, Y \rangle \not\models d$
Cl	remove d	delete the rule
$C\neg l$	remove d	delete the rule
$\neg Cl$	replace d with $\neg l$	delete the rule
$\neg C\neg l$	replace d with l	delete the rule

Table 1

Obtain Π_X^M from Π by eliminating default literals in Π , where $l \in \text{literals}(\Pi)$.

With the definition of satisfiability, we can define the semantics of ASP programs with defaults as follows.

Definition 2 (Default Reduct). *Let Π be an ASP program with defaults, $M = \langle X, Y \rangle$ be a default interpretation of Π . The default reduct of Π w.r.t $\langle X, Y \rangle$ is a pair of ASP programs $\Pi^M = \langle \Pi_X^M, \Pi_Y^M \rangle$, where Π_X^M is obtained from Π by eliminating all default literals as Table 1, and Π_Y^M is obtained from Π_X^M by adding fact rules for every literal in Y , i.e., $\Pi_Y^M = \Pi_X^M \cup \{l \mid l \in Y\}$.*

For a default interpretation $M = \langle X, Y \rangle$ of Π , Π_X^M and Π_Y^M are two ASP programs without defaults, while Π_X^M eliminates all default literals according to the assumptions in Y . As a result, a consequence $\langle X, Y \rangle$ of program Π , or a *default model*, should be a default interpretation that X is concluded by Π_X^M , and Π_Y^M is satisfiable.

Definition 3 (Default Models). *Consider a CDLP program Π , a default interpretation $M = \langle X, Y \rangle$, $\Phi(Y, \Pi)$ be a set of default literals of the form Ce in Π that is satisfied by Y , i.e., $\Phi(Y, \Pi) = \{Ce \mid Ce \in \text{defaults}(\Pi), Y \models Ce\}$. $\langle X, Y \rangle$ is a default model of Π iff. it satisfies the following conditions.*

1. X is an answer set of Π_X^M ,
2. Π_Y^M is satisfiable,
3. $Y = X \cup \{l \mid Cl \in \Phi(Y, \Pi)\}$,
4. $\nexists \langle X', Y' \rangle \in DM(\Pi)$ s.t. $\Phi(Y, \Pi) \subsetneq \Phi(Y', \Pi)$, where $DM(\Pi)$ denotes the set of all default models of Π .

The first condition of Definition 3 means X is a minimal model w.r.t the assumption Y . The second condition means Y is consistent with Π_X^M . Additionally, the third and fourth conditions mean all literals in $Y \setminus X$ are assumptions with corresponding default literals, and Y contains as many assumptions as possible.

Example 3 (Continue Example 2). *For the program Π_2 , $M = \langle X = \{q\}, Y = \{q, s\} \rangle$ is a default interpretation. The default reduct Π^M is a pair $\langle \Pi_X^M, \Pi_Y^M \rangle$, where Π_X^M is*

$$p \leftarrow s. \qquad q.$$

and Π_Y^M is

$$p \leftarrow s. \qquad q. \qquad s.$$

Apparently, X is an answer set of Π_X^M , and Π_Y^M is satisfiable. Meanwhile, $Y \setminus X$ contains the only literal r in $\Phi(Y, \Pi_2)$ and Π_2 , which means Y contains maximal assumptions. Therefore, M is a default model of Π_2 .

By the fourth condition in Definition 3, CDLP also provides a method to distinguish the negation of assumption ($\neg Cl$) and the assumption of negation ($C\neg l$).

Example 4. Consider following CDLP programs Π_3

$$q \leftarrow C\neg p.$$

and Π_4

$$q \leftarrow \neg Cp.$$

Π_3 represents an assumption of negation, which means q is true if $\neg p$ is consistent. Obviously, Π_3 has a unique default model $\langle \{q\}, \{q\} \rangle$. Meanwhile, by Π_4 , q is true if p is not consistent, which is not justified by any fact or rule. Therefore, $\langle \emptyset, \{p\} \rangle$ is the only default model of Π_4 .

As a result, we can describe the athlete's problem in Section 1 as follows.

Example 5 (Athlete's Dilemma). The athlete's dilemma in Section 1 can be described with a CDLP program Π_5 :

$$\text{marathon} \leftarrow C\text{marathon}. \quad (r_1)$$

$$\text{relay} \leftarrow C\text{relay}. \quad (r_2)$$

$$\leftarrow \text{marathon}, \text{relay}. \quad (r_3)$$

Program Π_5 has two default models,

$$M_1 = \langle \{\text{marathon}\}, \{\text{marathon}\} \rangle \text{ and}$$

$$M_2 = \langle \{\text{relay}\}, \{\text{relay}\} \rangle.$$

4. Program Splitting

4.1. Constraint Nonmonotonicity

The *constraint monotonicity* is an important property for ASP logic programs. A semantics \mathcal{S} satisfies *constraint monotonicity* if, for any ASP program Π and a constraint rule r_c , there does not exist a stable model A of $\Pi \cup \{r_c\}$ such that A is not a stable model of Π . However, CDLP obviously does not satisfy this property.

Corollary 1 (Constraint Nonmonotonicity). *The semantics of CDLP is not constraint monotonic.*

This corollary can be proved by the following example.

Example 6. Consider following program Π_6 : $\{q \leftarrow Cp.\}$ and a constraint rule r_c : $\leftarrow q$. Π_6 has a unique default model $M = \langle \{q\}, \{p, q\} \rangle$, while $\Pi_6 \cup \{r_c\}$ has a unique default model $M' = \langle \emptyset, \emptyset \rangle$.

An observation of Example 6 is that the constraint rule r_c does not entail $M' \neq \mathbf{C}p$ directly. It rejects the default model produced by Π_6 , and brings out a new guess of default literals that $\mathbf{C}p$ is not satisfied. It means that whether a default literal $\mathbf{C}p$ is satisfied is not only affected by the rules p depends, and also by the rules depends p directly or indirectly.

4.2. Strong Splitting Set

In a standard ASP program Π , constraint rules are always in the top part $t_U(\Pi)$ with regard to a splitting set U , because $head(r)$ of a constraint rule is always empty. Since the semantics of CDLP does not satisfy constraint monotonicity, the literals in constraint rules should be calculated first. We can define the splitting set of CDLP by modifying the splitting set of ASP such that all constraint rules belong in the bottom part.

Definition 4 (Strong Splitting Set). *Let Π be a CDLP program without odd loops. A splitting set of Π is a set $U \in literals(\Pi)$ such that for every rule $r \in \Pi$, if $head(r) \cup U \neq \emptyset$, or $head(r) = \emptyset$, then $literals(r) \in U$. The set of rules $r \in \Pi$ such that $literals(r) \subseteq U$ is called the bottom of Π w.r.t. U , and is denoted by $b_U(\Pi)$. The set of rules $\Pi \setminus b_U(\Pi)$ is called the top of Π w.r.t. U , and is denoted by $t_U(\Pi)$.*

Example 7. *Consider following program Π_7 :*

$$\begin{aligned} q &\leftarrow \mathbf{C}p. & (r_1) \\ &\leftarrow q. & (r_2) \\ s &\leftarrow \mathbf{C}q. & (r_3) \end{aligned}$$

Let $U = \{q, p\}$ be a splitting set of Π_7 . Thus we have $b_U(\Pi_7) = \{r_1, r_2\}$, and $t_U(\Pi_7) = \{r_3\}$.

Let M_b be a default model of $b_U(\Pi)$. For every rule r , if for every extended literal (or default literal) t in $body(r)$, $literals(t) \subseteq U \rightarrow M_b \models t$, then we say r is not falsified by M_b . We can obtain a rule r' from every $r \in t_U(\Pi)$ by following steps if r is not falsified:

- $head(r') = head(r)$, and
- if there is an extended literal or default literal t that $literals(t) \in U$, then remove t from $body(r')$.

We denote the set of obtained rules by $e_U(\Pi \setminus b_U(\Pi), M_b)$.

Definition 5 (Solution). *Let Π be a CDLP program without odd loops, U be a splitting set of Π . A solution to Π with regard to U is a pair $\langle M_b, M_e \rangle$ such that*

1. M_b is a default model of $b_U(\Pi)$, and
2. M_e is a default model of $e_U(\Pi \setminus b_U(\Pi), M_b)$, and
3. for $M_b = \langle X_b, Y_b \rangle$ and $M_e = \langle X_e, Y_e \rangle$, $X_b \cup X_e$ and $Y_b \cup Y_e$ are consistent respectively.

Theorem 1 (Strong Splitting Theorem). *Let Π be a CDLP program without odd loops, U be a strong splitting set of Π . $\langle X, Y \rangle$ is a default model of Π , if and only if there exists a solution $\langle \langle X_b, Y_b \rangle, \langle X_e, Y_e \rangle \rangle$ w.r.t. U such that $X = X_b \cup X_e$ and $Y = Y_b \cup Y_e$.*

4.3. Splitting Set

Although a strong splitting set can be used to split a CDLP program, Definition 4 is too strong for many conditions. For example, for CDLP programs with odd loops, a strong splitting set can not split these programs correctly.

Example 8. Consider following program Π_8 with an odd loop.

$$q \leftarrow \mathbf{C}p. \quad (r_1)$$

$$s \leftarrow \neg s, p. \quad (r_2)$$

If we extend Definition 4 to CDLP programs with odd loops, $U = \{p\}$ was a strong splitting set of Π_8 . Therefore, there was a solution $\langle M_b, M_e \rangle$ of Π_8 , where $M_b = \langle \{q\}, \{p, q\} \rangle$ and $M_e = \langle \emptyset, \emptyset \rangle$. However, r_2 in this program is a rule with an odd loop, which implies that p will lead to a contradiction, and $\mathbf{C}p$ should not be satisfied by any default models. As a result, $M = \langle \emptyset, \emptyset \rangle$ is the unique default model of Π_8 . Apparently, $\langle M_b, M_e \rangle$ is not a solution corresponding to M .

On the other hand, if a program consists of two sets Π' and Π'' of rules with disjoint literals, then Π' and Π'' can both contain constraint rules.

Example 9. Consider following program Π_9 :

$$\begin{array}{ll} p \leftarrow \mathbf{C}p. & \leftarrow p. \\ q \leftarrow \mathbf{C}q. & \leftarrow \mathbf{C}q. \end{array}$$

Π_9 should have two splitting sets $\{p\}$ and $\{q\}$.

Further observation shows that if a literal p belongs in the bottom part $b_U(\Pi)$, literals in a rule r that depends on $\mathbf{C}p$ should also belong in $b_U(\Pi)$.

Example 10. Consider following program Π_{10} :

$$p|u. \quad (r_1)$$

$$s \leftarrow \neg p. \quad (r_2)$$

$$q \leftarrow \mathbf{C}p. \quad (r_3)$$

$$\leftarrow q. \quad (r_4)$$

$$t \leftarrow u, \neg t. \quad (r_5)$$

Figure 1 is the support graph of Π_{10} . In Figure 1, both r_4 and r_5 are used to represent inconsistency. It shows that r_4 is indirectly dependent on $\mathbf{C}p$. As a result, every literal that p depends on should also belong in U . Thus Π_{10} has a splitting set $U = \{p, q, u, s\}$, and $b_U(\Pi_{10}) = \{r_1, r_2, r_3, r_4\}$.

Definition 6 (Splitting Set). Let Π be a program, U and U^- be two sets of literals in $\text{literals}(\Pi)$. U is a splitting set if for every rule $r \in \Pi$,

- if $\text{head}(r) \cap U \neq \emptyset$, then $\text{literals}(r) \in U$, and

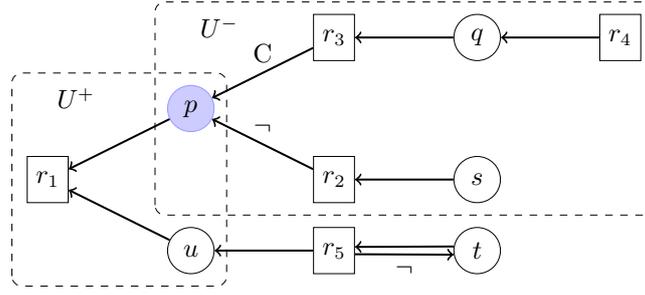


Figure 1: The support graph with constraints of Π_{10} . U^- contains all rules and literals that depending p , while p belongs in U^+ , which is a subset of U . Therefore, the literals in r_2 , r_3 and r_4 should belong in U .

d	$Ce \in g$	$Ce \notin g$
Ce	remove Ce	delete the rule
$\neg Ce$	replace Ce with \bar{e}	delete the rule

Table 2

Obtain Π_X^g from Π by eliminating default literals in Π , where e is an extended literal, and \bar{e} is its counterpart.

- for every literal $l \in defaults(\Pi)$, if $l \in U$, then there exists a set U^- of literal such that
 - $l \in U^-$, and
 - if $literals(body(r)) \cap U^- \neq \emptyset$, then $literals(r) \subset U^-$, and
 - $U^- \subseteq U$.

According to the definition above, the splitting set U of Π_{10} in Example 10 is $U^+ \cup U^-$ in Figure 1, where U^+ contains all literals in r_1 , and U^- contains all literals in r_3 and r_4 .

Theorem 2 (Splitting Theorem). Let Π be a CDLP program, U a splitting set of Π . $\langle X, Y \rangle$ is a default model of Π , if and only if there exists a solution $\langle \langle X_b, Y_b \rangle, \langle X_e, Y_e \rangle \rangle$ w.r.t U such that $X = X_b \cup X_e$ and $Y = Y_b \cup Y_e$.

5. A Primary Solving Algorithm

We propose a primary algorithm for solving CDLP programs based on the idea of generate-and-test.

From Example 3, we can find that the Π_X^M in a default reduct is only affected by Y in the interpretation $M = \langle X, Y \rangle$, or more specifically, by $\Phi(Y, \Pi)$. Therefore, for a guess about which default literals are satisfied, we can build a collection of default interpretations satisfying Condition 1 to 3 in Definition 3, which is called *candidate default models* and denoted by $CDM(\Pi)$.

For a guess g of a CDLP program Π , where $g \subseteq defaults(\Pi)$, the candidate default models w.r.t g is obtained by following steps:

1. generate Π_X^g according to Table 2, which is a variation of Table 1.
2. obtain the X parts by solving the ASP program Π_X^g .
3. for a possible X , obtain Y by union X and the literals in g and check if the Y part satisfies $\Pi_Y^{\langle X, Y \rangle}$.

Once we calculated all candidate default models of a program Π , we can get the default models of Π by comparing the $\Phi(Y, \Pi)$ of these candidate default models. Since the guesses contain all default literals in function Φ , we can also compare the subset relation between guesses. Additionally, by sorting the guess sequence, we can make sure that whenever we are testing guess g_j , the super sets of g_j are already tested. Thus we only need to check whether there is a default model with a tested guess g_i , where $g_j \subset g_i$.

The algorithm above is formally described in Algorithm 1.

Algorithm 1: solving a CDLP program

Input: Π , a CDLP program
Output: DM, the set of all default models of Π

```

1 DM  $\leftarrow$   $\emptyset$ 
  // obtain sorted consistent guesses s.t.
  //  $\forall i, j : (g_i \supset g_j) \rightarrow (i < j)$ 
2 Guess = GuessDefault ( $\Pi$ )
3 foreach  $g \in$  Guess do
4   if  $\nexists \langle X', Y' \rangle \in$  DM s.t.  $g \subset \Phi(Y', \Pi)$  then
5     Calculate  $\Pi_X^g$  according to Table 2
6     Solve  $AS(\Pi_X^g)$  with existing ASP solvers
7     foreach  $X \in AS(\Pi_X^g)$  do
8        $Y \leftarrow X \cup \{l \mid Cl \in g\}$ 
9       // check if  $\langle X, Y \rangle$  is a default model
10      Generate  $\Pi_Y^{\langle X, Y \rangle}$ 
11      if Satisfiable( $\Pi_Y^{\langle X, Y \rangle}$ ) then
          DM  $\leftarrow$  DM  $\cup$   $\{\langle X, Y \rangle\}$ 

```

Theorem 3 (Soundness and Completeness of Algorithm 1). *Let a CDLP program Π be the input of Algorithm 1, and DM is the set of default interpretations output by Algorithm 1. A default interpretation M is a default model of Π , if and only if $M \in DM$.*

For a CDLP program Π and a guess $g \in defaults(\Pi)$, the calculation of default reduct Π_X^g is polynomial, while the calculation of X and Π_Y^g is as difficult as solving an ASP program, which is in Σ_2^P [23]. Therefore, the solving of CDLP programs is in Σ_3^P .

6. Relation with CDL

We can establish a correspondence between CDLP programs and constrained default theories. Let Π be a CDLP program without disjunction heads or negation as failure, then its corresponding CDL theory $\tau(\Pi) = (D_\tau, W_\tau)$ is obtained by following steps. For a rule r in a program Π with following form:

$$l_0 \leftarrow l_1, \dots, l_m, \mathbf{C}l_{m+1}, \dots, \mathbf{C}l_n. \quad (4)$$

1. if l_0 is a literal and $defaults(r)$ is not empty, its corresponding default $\tau(r) \in D_\tau$ is

$$\frac{l_1 \wedge \dots \wedge l_m : l_{m+1}, l_n}{l_0} \quad (5)$$

2. otherwise, its corresponding formula $\tau(r) \in W_\tau$ is

$$l_1 \wedge \dots \wedge l_m \supset l_0 \quad (6)$$

where l_0 may be a literal or \perp .

Theorem 4. *Let Π be a CDLP program without disjunction and negation as failure, and $\tau(\Pi)$ the corresponding constrained default theory.*

- if $\langle X, Y \rangle$ is a default model of Π , then $\langle Th(X), Th(Y) \rangle$ is a constrained extension of $\tau(\Pi)$.
- If $\langle E, C \rangle$ is an extension of $\tau(\Pi)$ with maximal(w.r.t. inclusion) justifications and Π is satisfiable, then there is a default model $\langle X, Y \rangle$ of Π such that $E = Th(X)$ and $Th(Y)$.

Example 11. *Consider the following program Π_{11}*

$$c \leftarrow \mathbf{C}a. \quad d \leftarrow \mathbf{C}b. \quad \leftarrow a, b.$$

The disjunctive constrained default theory $\tau(\Pi_{11})$ is

$$\left(D = \left\{ \frac{:a}{c}, \frac{:b}{d} \right\}, W = \{a \wedge b \supset \perp\} \right)$$

Apparently, the program Π_{11} has two default models $\langle \{c\}, \{a, c\} \rangle$ and $\langle \{d\}, \{b, d\} \rangle$, while the default theory $\tau(\Pi_{11})$ has two constrained extensions, $E_1 = \langle Th(\{c\}), Th(\{a, c\}) \rangle$ and $E_2 = \langle Th(\{d\}), Th(\{b, d\}) \rangle$.

7. Related Works

There are many existing works on the consistency operators in logic programs. Logic GK of minimal knowledge and justified assumption [24] is a logic with two modal operators, \mathcal{K} for “known” and \mathcal{A} for “assumed”, while $\mathcal{A}F$ is true in a preferred model only if the assumption of F is justified. MBNF [25] is a simplified version of [24]. It uses believe operator \mathcal{B} (\mathcal{K} in the earlier version [26]) and adapts \neg as another operator for nonmonotonicity. AELB [27] is a

variation of autoepistemic logic (AEL) with an additional operator \mathcal{B} , but the meaning of operator \mathcal{B} is different from MBNF. $\mathcal{B}F$ intuitively means formula F is believed to be true, and formally means F is true by some nonmonotonic formalism. For example, F is true in some minimal models based on circumscription. ASP with default logic [28] and dl2asp [17] combine answer set programming and Reiter’s default logic by forming defaults into ASP rules with default operators. The language of epistemic specifications [9, 29] extends ASP with modal operators \mathbf{K} and \mathbf{M} , where \mathbf{M} is the dual operator of \mathbf{K} . $\mathbf{M}l$ is defined as $\neg\mathbf{K}\neg l$ and means literal l “may be true.” All these works are closely related to default logic, autoepistemic logic, and answer set programming [30, 31, 32, 33]. A belief formula with consistency operators in these logics is true in a model only if it is justified in all or some possible worlds. By the principle of rational or minimal knowledge, an assumption of literal l is not true if l is not justified. Thus these logics do not have the property of joint consistencies when they are used to represent defaults.

Some logic paradigms that support the joint consistency of assumptions, such as deontic logic and DL-semantics for answer sets. Deontic logic [34, 35] is a modal logic with two operators, \mathbf{O} for obligations and \mathbf{P} for permissions. A standard deontic logic theory does not allow any normative conflicts, for example, $\mathbf{O}p$ and $\mathbf{O}\sim p$ [36], which is similar to the joint inconsistencies in constrained default logic and CDLP. Shen and Eiter [37] state that the additional constraints can lead to believing some non-minimal models of origin programs in a relaxation of answer set semantics, which seems a more intuitive interpretation of constraints for domain specialists.

8. Conclusion

This paper proposes a new ASP paradigm CDLP concerning the constraint default logic defined on a popular idea of joint consistency in commonsense reasoning. Unlike the existing default logic variants defined in the classical first-order logic, negation as failure is allowed in CDLP. It thus provides a more powerful tool to deal with incomplete information. The splitting theorem and an algorithm to solve the CDLP program are given.

In the future, we plan to investigate more properties of CDLP, for example, the Kripke semantics for CDLP. Moreover, we also plan to develop a parallel algorithm for solving CDLP programs based on the splitting set and the primary algorithm presented in this paper and develop a parallel solver. In addition, we plan to model some real-world applications with CDLP and solve these problems with the solver.

References

- [1] M. Gelfond, V. Lifschitz, The stable model semantics for logic programming, in: R. A. Kowalski, K. A. Bowen (Eds.), *Logic Programming, Proceedings of the Fifth International Conference and Symposium, Seattle, Washington, USA, August 15-19, 1988* (2 Volumes), MIT Press, 1988, pp. 1070–1080.
- [2] M. Gelfond, V. Lifschitz, Classical negation in logic programs and disjunctive databases, *New Gener. Comput.* 9 (1991) 365–386.
- [3] V. W. Marek, M. Truszczyński, Stable models and an alternative logic programming paradigm, in: K. R. Apt, V. W. Marek, M. Truszczyński, D. S. Warren (Eds.), *The Logic*

- Programming Paradigm - A 25-Year Perspective, Artificial Intelligence, Springer, 1999, pp. 375–398.
- [4] G. Brewka, T. Eiter, M. Truszczynski, Answer set programming at a glance, *Commun. ACM* 54 (2011) 92–103.
 - [5] M. Gelfond, Y. Kahl, Knowledge Representation, Reasoning, and the Design of Intelligent Agents, Cambridge University Press, 2014.
 - [6] R. Reiter, A logic for default reasoning, *Artificial Intelligence* 13 (1980) 81–132.
 - [7] G. Antoniou, A tutorial on default logics, *ACM Comput. Surv.* 31 (1999) 337–359.
 - [8] W. Faber, An introduction to answer set programming and some of its extensions, in: M. Manna, A. Pieris (Eds.), Reasoning Web. Declarative Artificial Intelligence - 16th International Summer School 2020, Oslo, Norway, June 24-26, 2020, Tutorial Lectures, volume 12258 of *Lecture Notes in Computer Science*, Springer, 2020, pp. 149–185.
 - [9] M. Gelfond, Strong introspection, in: T. L. Dean, K. R. McKeown (Eds.), Proceedings of the 9th National Conference on Artificial Intelligence, Anaheim, CA, USA, July 14-19, 1991, Volume 1, AAAI Press / The MIT Press, 1991, pp. 386–391.
 - [10] C. Baral, M. Gelfond, J. N. Rushton, Probabilistic reasoning with answer sets, *Theory Pract. Log. Program.* 9 (2009) 57–144.
 - [11] J. Lee, Y. Wang, Weighted rules under the stable model semantics, in: C. Baral, J. P. Delgrande, F. Wolter (Eds.), Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR 2016, Cape Town, South Africa, April 25-29, 2016, AAAI Press, 2016, pp. 145–154.
 - [12] W. Lukaszewicz, Considerations on default logic: an alternative approach, *Comput. Intell.* 4 (1988) 1–16.
 - [13] A. Mikitiuk, M. Truszczynski, Constrained and rational default logics, in: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes, Morgan Kaufmann, 1995, pp. 1509–1517.
 - [14] M. Gelfond, H. Przymusinska, V. Lifschitz, M. Truszczynski, Disjunctive defaults, in: Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91). Cambridge, MA, USA, April 22-25, 1991, Morgan Kaufmann, 1991, pp. 230–237.
 - [15] T. Schaub, On Constrained Default Theories, in: B. Neumann (Ed.), 10th European Conference on Artificial Intelligence, ECAI 92, Vienna, Austria, August 3-7, 1992. Proceedings, John Wiley and Sons, 1992, pp. 304–308.
 - [16] V. Lifschitz, Answer set programming and plan generation, *Artif. Intell.* 138 (2002) 39–54.
 - [17] Y. Chen, H. Wan, Y. Zhang, Y. Zhou, dl2asp: Implementing default logic via answer set programming, in: T. Janhunen, I. Niemelä (Eds.), Logics in Artificial Intelligence - 12th European Conference, JELIA 2010, Helsinki, Finland, September 13-15, 2010. Proceedings, volume 6341 of *Lecture Notes in Computer Science*, Springer, 2010, pp. 104–116.
 - [18] V. Lifschitz, H. Turner, Splitting a logic program, in: P. V. Hentenryck (Ed.), Logic Programming, Proceedings of the Eleventh International Conference on Logic Programming, Santa Marherita Ligure, Italy, June 13-18, 1994, MIT Press, 1994, pp. 23–37.
 - [19] D. Nelson, Constructible falsity, *J. Symb. Log.* 14 (1949) 16–26.
 - [20] T. Syrjänen, Debugging inconsistent answer set programs, in: The 11th International Workshop on Nonmonotonic Reasoning, Low Wood hotel, Lake District, England, UK, 30

May-1 June, 2006, 2006, pp. 77–83.

- [21] N. Madrid, M. Ojeda-Aciego, Measuring Inconsistency in Fuzzy Answer Set Semantics, *IEEE Trans. Fuzzy Syst.* 19 (2011) 605–622.
- [22] F. Lin, X. Zhao, On Odd and Even Cycles in Normal Logic Programs, in: D. L. McGuinness, G. Ferguson (Eds.), *Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence*, July 25-29, 2004, San Jose, California, USA, AAAI Press / The MIT Press, 2004, pp. 80–85.
- [23] T. Eiter, W. Faber, M. Fink, S. Woltran, Complexity results for answer set programming with bounded predicate arities and implications, *Ann. Math. Artif. Intell.* 51 (2007) 123–165.
- [24] F. Lin, Y. Shoham, A logic of knowledge and justified assumptions, *Artif. Intell.* 57 (1992) 271–289.
- [25] V. Lifschitz, Minimal belief and negation as failure, *Artificial Intelligence* 70 (1994) 53–72.
- [26] V. Lifschitz, Nonmonotonic Databases and Epistemic Queries, in: *Proceedings of the 12th International Conference on Artificial Intelligence- Volume 1*, 1991, pp. 381–386.
- [27] T. C. Przymusiński, Autoepistemic Logic of Knowledge and Beliefs, *Artif. Intell.* 95 (1997) 115–154.
- [28] V. W. Marek, J. B. Remmel, Answer set programming with default logic, in: J. P. Delgrande, T. Schaub (Eds.), *10th International Workshop on Non-Monotonic Reasoning (NMR 2004)*, Whistler, Canada, June 6-8, 2004, Proceedings, 2004, pp. 276–284.
- [29] P. Kahl, R. Watson, E. Balai, M. Gelfond, Y. Zhang, The language of epistemic specifications (refined) including a prototype solver, *J. Log. Comput.* 30 (2020) 953–989.
- [30] K. Konolige, On the relation between default and autoepistemic logic, *Artificial Intelligence* 35 (1988) 343–382.
- [31] M. Truszczyński, Modal Interpretations of Default Logic, in: *Proceedings of the 12th International Joint Conference on Artificial Intelligence*. Sydney, Australia, August 24-30, 1991, Morgan Kaufmann, 1991, pp. 393–398.
- [32] V. W. Marek, M. Truszczyński, More on modal aspects of default logic, *Fundam. Informaticae* 17 (1992) 99–116.
- [33] G. Gottlob, Translating Default Logic into Standard Auto-epistemic Logic, *Journal of the ACM (JACM)* 42 (1995) 711–740.
- [34] G. H. von Wright, Deontic logic, *Mind* 60 (1951) 1–15.
- [35] L. Åqvist, *Deontic Logic*, Springer Netherlands, Dordrecht, 2002, pp. 147–264.
- [36] J. Horty, Deontic Modals: Why Abandon the Classical Semantics?, *Pacific Philosophical Quarterly* 95 (2014) 424–460.
- [37] Y.-D. Shen, T. Eiter, Determining inference semantics for disjunctive logic programs, *Artif. Intell.* 277 (2019).