第 42 卷 第 6A 期
2015 年 6 月
计 算 机 科 学
Computer Science
Vol. 42 No. 6A
June 2015

一种改进的处理不一致的回答集程序的方法

路芳芳 王 洁
(北京工业大学计算机学院多媒体与智能软件技术北京市重点实验室 北京 100124)

摘 要 回答集程序设计是一种描述性的程序设计范例,目前成为逻辑程序设计领域中一个很重要、很活跃的研究课题。在实际应用中,由于知识的不一致性,使得程序没有回答集。为解决该问题,朱涛等人提出了基于最小原理的思想,它可以很好地处理不一致的回答集程序,但是该方法会删除对用户来说很重要的知识,并且无法根据用户自己的偏好找到最满意的解。针对该问题,以加权逻辑程序为基础,使用加权定量的方法来处理不一致的回答集程序。通过为每一个规则分配权值,权值表示废除该规则所需的代价,废除规则代价之和最小的作为最优解,方便而简洁地解决了在无解情况下求解最满意解的问题。最后,与相关工作进行比较。

关键词 回答集程序,知识表示,不一致,最小原理,加权定量
中图法分类号 TP301 文献标识码 A

Improved Method to Process Inconsistent Answer Set Program

LU Fang-fang WANG Jie
(Key Laboratory of Multimedia and Intelligent Software, Department of Computer Science, Beijing University of Technology, Beijing 100124, China)

Abstract Answer Set Programming(ASP) is a kind of declarative programming. At present, ASP has become an important and active research in the field of logic programming. In practical applications, one of the possibles in answer set programming is the absence of any solutions in case of inconsistent information. To remedy this, Zhu Tao presented a minimal principle based method to process inconsistency in ASP. However, this method will delete the most important information for users and can't find the most satisfactory solution according to their own preferences. In order to solve this problem, based on weight logic program and using a weight quantitative method that associates a weight with each rule in a program, these weights define the "cost" of deleting a rule, the solution is preferred if it minimizes the sum of the weights of its deleted rules and after that, we compared this method with related work.

Keywords ASP, Knowledge representation, Inconsistency, Minimal principle, Weight quantitative

1 引言

回答集程序[1]是一种说明性的程序设计范例,起源于逻辑程序设计[2]和非单调性推理。目前,回答集程序设计是知识表示[3]的有效方法,并成为非单调性推理和逻辑程序设计研究领域的热点[4,5]。

回答集程序的基本思想是用逻辑程序来描述给定的问题,程序的回答集对应了原问题的解决途径[6,7]。但在实际应用中,一个可能的问题是由于存在不一致的知识使得该程序没有回答集或者程序的回答集中存在互补的文字,即回答集程序是不一致的。在 ASP 中,对不一致的处理为不一致的知识推理提供了一种方式。

近年来,为了修复不一致的回答集程序,使其成为一致的程序,国内外许多学者对不一致的回答集程序进行了研究[8-10]。文献[8]中,Balduccini 等提出了 CR-Prolog 语言处理不一致的回答集程序[8],通过引入一致性恢复规则达到一致性。文献[9]中,相关学者提出了一种新的扩展的回答集语义来处理不一致的回答集程序[9]。文献[10]中,相关学者提出了基于最小原理的思想来处理不一致性[10]。然而,这些方法存在以下缺点:首先,当 CR-Prolog 语言中的严格规则出现不一致时,通过引入一致性恢复规则无法达到一致性。其次,扩展的回答集语义的方法无法处理由约束规则引起的不一致性。最后,最小原理的基本思想是对于同时存在严格规则和缺省规则的不一致的回答集程序,当严格规则一致时,优先删除缺省规则,最大限度地使用严格规则进行推理。但是这种处理方式很有可能删除决策者不希望删除的信息,并且无法根据决策者自己的需求选择最适合自己的解决方案。

考虑如下不一致的回答集程序 P:

r_1 : rice←

r_2 : steak←rice

r_3 : ¬steak← rice, not pepper

很明显,该程序是不一致的。根据最小原理的方法,优先删除缺省规则达到一致性,即得到 P 的一个回答集为{rice, steak},删除的规则为 r_3。

路芳芳(1989—),女,硕士生,主要研究方向为回答集程序设计、知识表示和推理,E-mail:luxfbeyond@163.com;王 洁(1972—),女,副教授,主要研究方向为不确定推理、面向 agent 的程序设计。

但在实际生活中，这种删除方式并不具有合理性。例如，一个来自南方的人，他个人的喜好是在吃牛排的时候希望有辣椒，如果不确定有辣椒时，他宁愿选择不吃牛排。根据他的喜好程度，r_3 的价值更大，他更倾向于删除 r_2。然而使用最小原理的方法强制删除了他不希望删除的信息。最小原理的方法虽然能够处理不一致的程序，但是无法根据用户自己的需求选择最适合自己的解决方案。

因此，为了解决这一问题，本文借鉴 Nieuwenborgh 等人提出的“加权回答集”的思想[11]，采用加权定量的方法，根据决策者的偏好为每一个规则分配权值，权值表示废除该规则所需的代价，废除规则代价之和最小的称为优化解，该方法可以方便而简洁地根据用户的偏好关系找到满意的解。

2 回答集程序设计的相关知识

本部分将对回答集程序设计的语法、语义进行简单的介绍。

规则具有如下的形式：

$$r: L_0 \leftarrow L_1, \cdots, L_m, \text{not } L_{m+1}, \cdots, \text{not } L_n (n \geqslant m \geqslant 0) \quad (1)$$

其中，$L_0, L_1, \cdots, L_n$ 称为文字，规则左边为规则头部，记作 $head(r)$，右边为规则体部，记作 $body(r)$。并且定义 $body^+(r) = \{L_1, \cdots, L_m\}$，$body^-(r) = \{L_{m+1}, \cdots, L_n\}$。头部为空的称为约束，体部为空的称为事实。一个文字可以是否定的，前面带有符号 $\neg$ 的称为否定文字，另一种否定形式是缺省非(naf)，记作 not。

规则可被分为以下两种形式：

严格规则：如果 $body^-(r) =$ 空集，即 $r: L_0 \leftarrow L_1, \cdots, L_m$ 则称规则 r 是严格规则。

缺省规则：如果 $body^-(r) \neq$ 空集，即 $r: L_0 \leftarrow L_1, \cdots, L_m, \text{not } L_{m+1}, \cdots, \text{not } L_n (n \geqslant m \geqslant 0)$ 则称规则 r 为缺省规则。

由上面的规则集组成的程序 P 为回答集程序。根据规则的种类，程序可被分为简单的逻辑程序(SLP)和扩展的逻辑程序(ELP)，其中 SLP 由严格规则组成，ELP 中至少有一个缺省规则。

若程序 P 为简单的逻辑程序(SLP)，S 表示原子集，Lit 表示程序中的所有基文字集，$S \subseteq Lit$ 是 P 的一个回答集如果 S 是满足 P 中所有规则的最小集合。若程序 P 为扩展的逻辑程序(ELP)，P 的 reduct 为 $P^S = \{head(r) \leftarrow body^+(r) \mid body^-(r) \cap S =$ 空集$\}$，很明显，P^S 是一个 SLP。S 是 P 的回答集如果 S 是 P^S 的一个回答集。如果回答集程序 P 至少有一个回答集，则称程序 P 是一致的，否则 P 是不一致的。

3 处理不一致的回答集程序

3.1 不一致的回答集程序的扩展

处理不一致的回答集程序的方法是通过删除某些规则，使其成为一致的回答集程序。本文首先对不一致的回答集程序进行扩展，定义如下：

定义 1(不一致的 ASP 的扩展) 程序 P 为不一致的回答集程序，程序 P 的扩展 $E(P)$ 包含如下规则：对于每一条规则 $r: head(r) \leftarrow body(r)$，其中 $r \in P$，且 $head(r) \neq$ 空，则替换为如下规则形式 $r': head(r) \leftarrow body(r), not \neg head(r)$。

程序 P 的一致性扩展回答集是任何一个解释 I，使得 I 是 $E(P)$ 的回答集。

例 1：考虑如下不一致的回答集程序 P：

r_1：penguins←

r_2：bird← penguins

r_3：¬fly ← penguins

r_4：fly ←bird

程序 P 的扩展形式 $E(P)$ 如下：

r_1'：penguins← not¬penguins

r_2'：bird← penguins，not¬bird

r_3'：¬fly ← penguins，not fly

r_4'：fly ←bird，not¬fly

由 ASP 求解器很容易得出 $E(P)$ 的回答集为：$S_1 = \{$penguins，bird，fly$\}$，$S_2 = \{$penguins，bird，¬fly$\}$，即程序 P 的一致性扩展回答集为 S_1 和 S_2。

不一致性处理的方法是找到合适的被删除的规则集(记作 Pd)，使得 P 去除 Pd 后是一致的。因此，定义规则废除的概念：

定义 2(规则废除) 对于程序 P 的一个一致性扩展回答集 S，如果 P 中存在规则 $r: head(r) \leftarrow body(r)$，使得当 $body^+(r) \subseteq S$，$body^-(r) \cap S = \emptyset$ 时 $head(r) \notin S$，则这条规则应被废除。对于一条约束规则，如果 $body^+(r) \subseteq S$，$body^-(r) \cap S = \emptyset$，则这条约束规则应被废除。$P$ 关于 S 的废除规则集记作 Pd。

如例 1 所示，程序 P 有两个一致性扩展回答集 $S_1 =$ $\{$penguins，bird，fly$\}$ 和 $S_2 = \{$penguins，bird，¬fly$\}$。对于 S_1，程序 P 中存在一条规则 r_3，使得 penguins$\subseteq S_1$，且¬fly$\notin S_1$，r_3 不被 S_1 所满足，应被废除。即 P 中关于 S_1 被废除的规则为 r_3，记作 $Pd_1 = \{r_3\}$。同理，关于 S_2 被废除的规则为 r_4，记作 $Pd_2 = \{r_4\}$。

一致性扩展回答集并不总是等同优先的。如上面的不一致的回答集程序 P，根据常识可知，企鹅是一种鸟，但是它不会飞，所以废除规则 r_4 优于 r_3。所以，为了进一步优化问题，找到更优的解，我们借鉴“加权回答集”的思想，采用加权定量的方法，根据决策者的偏好为每一个规则分配权值，权值表示废除该规则所需的代价，废除规则代价之和最小的称为最优解。

3.2 加权定量的方法

加权定量方法的基本思想是为程序中的每一个规则指派一个权值，权值表示废除这条规则所需的代价。一个一致性扩展回答集是最优的，如果关于它的被废除的规则集的代价之和最小。

首先给出代价规则及加权回答集程序的定义：

定义 3(代价规则) 一条带有权值的规则称为代价规则，它具有如下形式的规则：$a \leftarrow \beta \langle w \rangle$，其中 $\{a\} \cup \beta$ 是有限的文字集，w 表示权值，是非负整数，表示这条规则的价值，$w(r)$ 表示规则 r 的权值。权值越高，价值越大，即废除这条规则付出的代价越大。

由带有权值的规则集组成的程序称为加权的回答集程序。加权的回答集程序的一致性扩展回答集与去除权值后的回答集程序的一致性扩展回答集是一致的。

例 2：例 1 中的程序 P 通过为每一个规则分配权值扩展为加权的回答集程序 P'，即：

r_1：penguins← ⟨1⟩

r_2：bird← penguins ⟨1⟩

r_3：¬fly ← penguins ⟨1⟩

r_4:fly ←bird 〈0〉

这个程序依然有两个一致性扩展回答集 S_1 和 S_2。然而 S_1、S_2 并不是等同优先的,我们希望能够找到付出代价最小的一致性扩展回答集作为最优解,下面给出关于一致性回答集 S 的代价及最优解的定义。

定义 4(一致性回答集 S 的代价) 关于加权的回答集程序 P 的一个一致性扩展回答集 S 的代价被定义为:$\Phi p(S)=\sum_{r\in pd} w(r)$,即关于一致性扩展回答集 S 被废除的规则的权值之和。

根据一致性扩展回答集之间的代价关系,可得到一致性扩展回答集之间的优先关系,称为最优解,定义如下:

定义 5(最优解) 对于两个一致性扩展回答集 S_1 和 S_2,定义 $S_1 \prec S_2$(符号 $\prec$ 表示优先关系,S_1 优先于 S_2)当且仅当 $\Phi p(S_1)\leqslant\Phi p(S_2)$ 且不存在 $\Phi p(S_2)\leqslant\Phi p(S_1)$,$P$ 的最优解是 P 的所有的一致性扩展回答集中关于 $\prec$ 最小的一致性扩展回答集。

如例 2 中 P' 的一致性扩展回答集为 $S_1=\{$penguins, bird, fly$\}$, $S_2=\{$penguins, bird, ¬fly$\}$, $Pd_1=\{r_3\}$, $Pd_2=\{r_4\}$,则 $\Phi p(S_1)=w(r_3)=1$, $\Phi p(S_2)=w(r_4)=0$,由优化解定义可知,S_1 是程序 P 的最优解。

下面举一个实例介绍本文提出的方法在实际生活中的应用。

例 3:小明为了决定晚饭吃什么,根据他的需求得到如下回答集程序 P:

r_1:rice ←

r_2:¬fries ← steak

r_3:steak ← not fries

r_4:fries ← not steak

r_5:¬rice← not ¬steak

然而该程序 P 是不一致的,为了找到最接近小明满意的解,首先根据他的喜好程序为规则分配相应的权值,得到 P':

r_1:rice ← 〈2〉

r_2:¬fries ← steak 〈2〉

r_3:steak ← not fries 〈1〉

r_4:fries ← not steak 〈1〉

r_5:¬rice← not ¬steak 〈3〉

根据定义 1 对不一致的回答集程序 P 进行扩展得到 $E(P)$,由 ASP 求解器很容易得出 $E(P)$ 的回答集为 $S_1=\{$fries, rice$\}$, $S_2=\{$steak, ¬fries, rice$\}$, $S_3=\{$fries, ¬rice$\}$, $S_4=\{$steak, ¬fries, ¬rice$\}$。即原程序 P 的一致性扩展回答集为 S_1, S_2, S_3, S_4。

根据废除规则的定义,求得一致性扩展回答集 S_1, S_2, S_3, S_4 对应的被废除的规则分别为:$Pd_1=r_5$, $Pd_2=r_5$, $Pd_3=r_1$, $Pd_4=r_1$,进而得出一致性回答集 S_1, S_2, S_3, S_4 的代价分别为:$\Phi p(S_1)=3$, $\Phi p(S_2)=3$, $\Phi p(S_3)=2$, $\Phi p(S_4)=2$。根据最优解的定义可知,S_3、S_4 是程序 P 的最优解,即废除的规则 r_1 优于 r_5。

加权定量的方法可以解决在无解情况下,根据用户的偏好求解用户最满意解的问题。对于不一致的回答集程序,首先通过规则替换得到一致的回答集程序,求出合理的一致性扩展回答集。根据用户分配的权值,求得一致性扩展回答集的代价,删除规则代价最小的作为最优解。这种加权定量的方法对原有的程序语法并未做太大的改变,计算过程简单、规范。

4 与相关工作比较

当回答集程序出现不一致时,如何处理不一致的回答集程序?针对这一问题,本文提出了一种新的处理方法,这种方法能克服之前的方法存在的缺陷,本部分将与相关工作进行比较。

Balduccini 和 Gelfond 等人提出了 CR-Prolog 语言[8]处理不一致的回答集程序。该方法的核心思想是:当回答集程序不一致时,添加一致性恢复规则达到一致性。该方法的好处是并未改变原有的回答集程序,但是当严格规则出现不一致时该方法无法通过添加一致性恢复规则得到回答集。

例如:考虑如下程序 P:

r_1:a ←

r_2:d ← a

r_3:f ← not d

r_4:b ← c

r_5:¬b ← a,not f,not e

r_6:c ←¬b,not f

CR-Prolog 引入一致性恢复规则 $head(r)\xleftarrow{\pm} body(r)$。这些规则具有如下特点:即使这些规则体部被满足也可能不被适用,只有当 P 不一致时才适用。上面的程序 P 是不一致的。添加一致性恢复规则 $e\xleftarrow{\pm}$,可得到回答集$\{a, d, e\}$。但是,当加入新规则 r_7:¬d ← a 到 P 中时,则 $PU\{r\}$ 的回答集推出¬d 和 d,也是不一致的,因为在 CR-Prolog 语言中,严格规则是不允许被废除的。

本例使用本文方法,首先分配相应的权值:

r_1:a ← 〈2〉

r_2:d ← a 〈2〉

r_3:f ← not d 〈1〉

r_4:b ← c 〈3〉

r_5:¬b ← a,not f,not e 〈2〉

r_6:c ← ¬b,not f 〈3〉

r_7:¬d ← a 〈1〉

然后,得出该程序的一致性扩展回答集及其相应的代价为:$S_1=\{a, \neg d, f\}$, $pd_1=\{r_2\}$, $\Phi p(S_1)=2$, $S_2=\{a, d\}$, $pd_2=\{r_5, r_7\}$, $\Phi p(S_2)=3$, $S_3=\{a, \neg b, d\}$, $pd_3=\{r_6, r_7\}$, $\Phi p(S_3)=4$。由此,S_1 是最优解,删除规则 r_2 代价最小。

Nieuwenborgh 和 Vermeir 等人提出了一种新的扩展的回答集语义[9],通过引入“被击败”规则来处理不一致的回答集程序。该方法的基本思想是:允许通过废除一些规则来达到一致性,但是,被废除的规则必须存在一条相应的被适用的规则。具体细节参考文献[9]。然而,这种方法并不能处理由约束规则引起的不一致性。

例如:考虑如下程序 P:

r_1:a ←

r_2:¬b ← a

r_3:¬c ← a

r_4:← ¬b, ¬c

由前 3 个规则可推出 a, ¬b, ¬c,但约束 r_4 限制¬b, ¬c 同时存在,所以程序 P 是不一致的。根据扩展的回答集语义,这条约束规则不可能被废除,因为不存在与它相对应的可被适用的规则,且废除其他规则也不存在与之对应的可被适用的规则。所以 Nieuwenborgh 等人提出的扩展语义无法处

理这种情况。

然而，使用本文方法，假如决策者分配的权值如下：

r_1：a 〈3〉

r_2：$\neg b \leftarrow a$ 〈2〉

r_3：$\neg c \leftarrow a$ 〈2〉

r_4：$\leftarrow \neg b, \neg c$ 〈1〉

首先，得出该程序的一致性扩展回答集及其相应的代价为：$S_1=\{a,\neg b\}$，$pd_1=\{r_3\}$，$\Phi p(S_1)=2$，$S_2=\{a,\neg c\}$，$pd_2=\{r_2\}$，$\Phi p(S_2)=2$，$S_3=\{a,\neg b,\neg c\}$，$pd_3=\{r_4\}$，$\Phi p(S_3)=1$。

根据最优解的定义，最终得出的优化解为：$S_3=\{a,\neg b,\neg c\}$，$pd_3=\{r_4\}$。即根据决策者的偏好，删除规则 r_4 付出的代价最小，是决策者最满意的解决方案。

朱涛等人提出了基于最小原理的方法来处理不一致的回答集程序[10]。最小原理的基本的思想是：对于一个不一致的回答集程序，当严格规则一致时，优先删除缺省规则，最大限度地保留严格规则，具体细节参考文献[10]。然而这种方法不能根据用户自己的需求选择最适合自己的解决方案。

如引言中的程序 P：

r_1：rice←

r_2：steak← rice

r_3：¬steak← rice，not pepper

根据最小原理的思想，严格规则 r_1，r_2 是一致的，不存在冲突，所以通过删除 r_3 达到一致性。这种处理方法无法保证删除的规则满足代价最小性，即不能根据用户的偏好关系选择自己最满意的解决方案。通过本文的方法，可以根据用户喜好程度分配不同的权值，优先选择删除规则代价之和最小的。

r_1：rice← 〈1〉

r_2：steak← rice 〈2〉

r_3：¬steak← rice，not pepper 〈3〉

该程序的一致性扩展回答集及其相应的代价为：

$S_1=\{\text{rice},\neg\text{steak}\}$，$pd_1=\{r_3\}$，$\Phi p(S_1)=3$，$S_2=\{\text{rice},\text{steak}\}$，$pd_2=\{r_2\}$，$\Phi p(S_2)=2$，$\Phi p(S_2)<\Phi p(S_1)$，即 S_2 是最优解，删除 r_2 付出的代价最小。

结束语　在实际应用中，由于知识库中存在不一致的知识，导致该回答集程序没有回答集或推出相互矛盾的结果。本文以加权逻辑程序为基础，提出了一种加权的定量方法来处理不一致的回答集程序。相比之前的方法，该方法通过引入代价最小性，方便而简洁地找到了决策者最满意的解，能更好地处理不一致性问题，具有重要的现实意义。

参 考 文 献

[1] Eiter T, et al. Answer Set Programming: A Primer [J]. Reasoning Web: Semantic Technologies for Information Systems, 2009, 5689: 40-110

[2] Gelfond M, Leone N. Logic programming and Knowledge representation. The A-Prolog perspective[J]. Artificial In telligence, 2002, 138: 3-38

[3] Baral C. Knowledge representation, reasoning and declarative problem solving[M]. Cambridge Univ Press, 2003

[4] 赵岭忠，张超，钱俊彦. 基于 ASP 的 CSP 并发系统验证研究[J]. 计算机科学，2012，39(12)：133-136

[5] Deng Wen-jun, Liang Yin-wen. Reason on UML Diagrams with Answer Set Programming[C]// Proc. of International Conf. on Computer Science and Software Engineering. 2008(1): 205-209

[6] Šefránek J. Updates of argumentation frameworks [C]// Proceedings of the 14th International Workshop on NonMonotonic Reasoning. 2012: 1-9

[7] luukkala V, Niemel I. Enhancing a smart space with answer set programming [M]// Semantic Web Rules. Springer Berlin Heidelberg, 2010: 89-103

[8] Balduccini M, Gelfond M. Logic programs with consistency-restoring rules[C]// International Symposium on Logical Formalization of Commonsense Reasoning, AAAI 2003 Spring Symposium Series. The AAAI press, 2003: 9-18

[9] Van Nieuwenborgh D, Vermeir D. Preferred answer sets for ordered logic programs[J]. Theory and Practice of Logic Programming, 2006, 6: 107-167

[10] Zhu Tao, Zhang Zhi-zheng. A Processing Method for Inconsistent Answer Set Programs Based on Minimal Principle[M]. 2012: 270-274

[11] D Nieuwenborgh V, Heymans S, Vermeir D. Weighted-Answer Sets and Applications in Intelligence Analysis[C] // MProc of the 11th Int. l Conf on Logic for Programming. ArtificialIntelligence, and Reasoning, 2006: 169-183

(上接第 515 页)

整个测试过程共运行测试用例 60110 个，其中通过 59943 个，失败 171 个。基于二进制代码生成的反汇编语言，对 TI C6701 编译器错误产生原因进行分类统计，并按照其对系统可靠性的影响进行分类。其中最严重优先级 bug 11 个，次严重优先级 bug 17 个，普通优先级 bug 有 36 个，次普通优先级 bug 有 8 个，最低优先级没有。

通过上述测试，可以发现：用户在不加限制地使用 TI C6701 编译器时，可能遇到编译器的正确性问题，进而影响整个系统的正确性。

结束语　本文的测试结果表明 TI C6701 编译器存在正确性错误，对于可靠性要求较高的领域的 DSP 应用开发有一定的参考价值。相关领域应用在进行软件开发时，程序员能够有意识地规避编译器中存在的正确性错误，从而提高整个应用的可靠性。

参 考 文 献

[1] Kocher P, Lee R, McGraw G, et al. Security as a New Dimension in Embedded System Design[C] // Proceedings of the Design Automation Conference. ACM Press, 2004(6): 735-760

[2] TMS320C62x Code Composer IDE Help (SPRH197), Parallel Debugging Manager: An Introduction[EB/OL]. http://www.ti.com, 2005-06

[3] Popovic M, Kovacevic V, Temerinac M. Software testing concept used for MAS/C-compiler[C] // Proceedings of the 26th Euromicro Conference. Maastricht. IEEE Computer Society, 2000 (2): 224-229

[4] 何群. C 编译器自动测试工具(Ctcgen)的剖析与移植(A)[J]. 计算机工程，2004，30(20)：95-97

[5] Plum Hall Inc. The Plum Hall Validation Suite for CTM[EB/OL]. http://www.plumhall.com/stecl.html

[6] ANSI X3.159-1989. Programming Language-C[OL]. http://www.freestd.us/soft/109648.html