计算机应用
Journal of Computer Applications
ISSN 1001-9081,CN 51-1307/TP

《计算机应用》网络首发论文

题目：　析取回答集程序结构化测试方法
作者：　杨东，王以松
收稿日期：　2021-11-08
网络首发日期：　2022-05-17
引用格式：　杨东，王以松．析取回答集程序结构化测试方法[J/OL]．计算机应用.
https://kns.cnki.net/kcms/detail/51.1307.TP.20220516.2108.023.html

Journal of Computer Applications　ISSN 1001-9081
计算机应用　CODEN JYIIDU　http://www.joca.cn

DOI:10.11772/j.issn.1001-9081.2021111891

析取回答集程序结构化测试方法

杨　东,王以松*
(贵州大学 计算机科学与技术学院,贵阳 550025)
(*通信作者电子邮箱yswang@gzu.edu.cn)

摘　要:针对析取回答集程序的结构化测试基础理论匮乏的问题,系统化地提出了析取回答集程序结构化测试覆盖概念。首先,定义了针对析取回答集程序的测试用例,确立了析取回答集程序的主要测试实体为程序中的逻辑规则,而后通过对规则的头、规则的体、规则的集合等不同测试目标构建了规则覆盖、定义覆盖、环覆盖等基本概念来模拟结构化测试中的语句覆盖、分支覆盖等概念。其次,提出了析取回答集程序的测试覆盖率计算公式,并举例说明各种覆盖下的覆盖率计算方法,同时还讨论了析取回答集程序的部分特殊性质和关键指标,通过提出以上基本理论,为研究析取回答集程序的测试提供了理论基础。

关键词:回答集程序设计;测试理论;析取回答集程序;结构化测试方法;覆盖
中图分类号:TP301.6　　**文献标志码:**A

Structural testing methods of disjunctive answer set programs

YANG Dong, WANG Yisong*
(*College of Computer Science and Technology, Guizhou University, Guiyang Guizhou* 550025, *China*)

Abstract: Focused on the issue that lack of theories of structured testing of disjunctive answer set programs, concepts of structured test coverage for disjunctive answer set programs is proposed systematically. Firstly, the concepts of test cases of disjunctive answer set program is proposed and the test objects that logic rules in the program is established. Then the concepts of rule coverage, definition coverage, loop coverage and so on are constructed to simulate the concepts of statement coverage and branch coverage in structured testing. Besides, the formula of disjunctive answer set program test coverage, and different coverage calculation methods are illustrated. In the same time, some special natures of the disjunctive answer set program and key indicator are presented. By developing the above basic theory, a basic theory for testing of the disjunctive answer set programs is developed.

Key words: answer set programming; testing theory; disjunctive answer set program; structural testing method; coverage

0　引言

回答集程序设计(Answer Set Programming,ASP)是语法上基于传统逻辑程序设计(Prolog)而语义上是基于稳定模型语义的非单调的一种描述式程序设计范式[1]。在ASP中,搜索问题被简化为计算稳定模型,通过回答集求解器(用于生成稳定模型的程序)执行搜索得到问题的解。由于回答集程序的非单调特征、良好的表达能力以及各种有效的求解器(如clasp、dlv)的出现,ASP现在已经在众多的人工智能领域得到了广泛运用,包括规划[2]、机器人[3][4]、知识推理[5]、决策支持[6]、电路诊断以及硬件设计[7][8]、自动驾驶[9]等领域。

回答集程序的工程化(包括软件调试、软件测试等)历经多年的发展取得了一定的成果,JANHUNEN等[10]将过程式程序设计语言(包括C、Java等)的测试覆盖方法引入到正规回答集程序,提出正规回答集程序的测试理论后,JANHUNEN等[11]在此理论基础上实现了正规回答集程序的结构化测试方法,与随机测试实验结果的比较表明,虽然在测试结果上很难体现出随机测试和结构化测试的优劣性,但是在某些特定的代码场景下,结构化测试方法相较于随机测试方法有着显著的规则异常捕获优势。OETSCH等[12]提出了小范围内穷尽测试的测试方法测试正规回答集程序,在4类包括增删规则和文字、谓词和变量重命名等共20种回答集程序变异操作中,用不超过13个个体符号的测试能达到99%的置信度。FEBBRARO等[13]提出了一种用于在ASP程序中指定和运行单元测试的语言。该测试语言是在回答集编程集成开发环境(Integrated Development Environment for Answer Set Programming,ASPIDE)中实现的,支持回答集程序的全生命周期开发,将回答集程序的测试、调试、分析、求解器执行配置和输出等功能集成在一个对用户友好的图形化工具中。BUSONIU等[14]开发了一个回答集编程的集成开发环境SeaLion。SeaLion为Gringo和DLV(DataLog with Disjunction, where the logical disjunction symbol V is used)提供源代码编辑器,并提供如语法高亮显示、语法检查、代码完成、可视化程序和代码重构功能。通过以上功能,SeaLion显著提升了回答集程序开发人员的编程效率。GREBLER等[15]设计了正规回答集程序的测试工具Harvey,使用ASP规则定义测试输入空间,Harvey使用异或抽样实现测试输入对正规回答集程序进行随机测试。AMENDOLA等[16]提出了一种基于ASP的测试执行机制以及在ASP程序中进行内联测试的单元测试规范语言,并用该语言开发提供了一个支持测试驱动开发(Test-Driven Development,TDD)

收稿日期:2021-11-08;**修回日期:**2022-04-21;**录用日期:**2022-04-25。　**基金项目:**国家自然科学基金项目(61976065)。
作者简介:杨东(1987—),男,贵州六盘水人,硕士研究生,CCF会员,主要研究方向:人工智能、知识表示与推理;　王以松(1975—),男,贵州思南人,博士,教授,CCF会员,主要研究方向:人工智能、知识表示与推理。

ASP程序的编程环境。

析取回答集程序的表达能力严格强于正规情形[17]，判断一个析取回答集程序是否存在回答集是-Σ_2^p完全的，而对正规回答集程序时是NP-完全的，因而析取回答集程序比正规回答集程序更复杂，应用范围也更广。然而，目前国内外仍缺少对析取回答集程序测试的深入研究，既缺乏其测试的理论基础，更没有系统化地析取逻辑程序自动化测试工具。

本文通过初步构建析取回答集程序的测试理论，模拟结构化测试方法中的语句覆盖、分支覆盖等覆盖形式，并给出相应的覆盖率计算方法，为进一步实现析取回答集程序自动化测试提供理论支持。

本文的主要工作包括三个方面：

1)将回答集程序结构化测试的理论扩展到析取回答集程序，使得原有的测试理论和公式的应用范围进一步扩大；由析取回答集程序的规则可知，正规回答集程序的规则头满足只有一个原子的情形，也就是说正规回答集程序是析取回答集程序的特例。本文提出的析取回答集测试理论是正规回答集程序测试理论的泛化，即本文的测试理论不仅可以运用于析取回答集程序，也可运用于正规回答集程序，反之则不行。

2)对析取回答集程序的部分特殊性质和定理进行了提取、分析和证明。

3)提出了回答集程序的解释I的重要指标，通过该指标可以衡量不同解释I对回答集程序的规则满足性，从而判断解释I的有效性。

1 预备知识

下面主要介绍析取回答集程序的基础知识和定义，并给出析取回答集程序的实例和稳定模型的具体计算方法。

1.1 析取回答集程序

假设命题语言L是建立在原子集合A之上。一个析取回答集程序P是由一系列规则(rule)组成，这些规则的形式是：

$$a_1 \vee a_2 \cdots \vee a_k \leftarrow b_1, b_2, \cdots, b_m, \text{not } c_1, \text{not } c_2, \cdots, \text{not } c_n \tag{1}$$

其中$a_i(1 \leqslant i \leqslant k), b_j(1 \leqslant j \leqslant m), c_t(1 \leqslant t \leqslant n)$都是原子公式。对于回答集程序$P$中的任一规则，有：

1) $a_1 \vee a_2 \vee \cdots \vee a_k$表示该规则的头；

2) $b_1, b_2, \cdots, b_m, \text{not } c_1, \text{not } c_2, \cdots, \text{not } c_n$ 表示该规则的体。

特别地，当k=1时，析取逻辑规则转为正规逻辑规则，即$a_1 \leftarrow b_1, b_2, \cdots, b_m, \text{not } c_1, \text{not } c_2, \cdots, \text{not } c_n$。回答集程序中的否定形式为缺省否定，即对于一个原子集合S，$\text{not } S = \{\text{not } p | p \in S\}$，给定式(1)中的规则$r$，有如下定义：

$$hd(r) = \{a_1, a_2, \cdots, a_k\}$$

$$pos(r) = \{b_1, b_2, \cdots, b_m\}$$

$$neg(r) = \{c_1, c_2, \cdots, c_n\}$$

$$bd(r) = pos(r) \cup not\ neg(r)$$

通常规则r的表示式简写为：$hd(r) \leftarrow bd(r)$。析取回答集程序示例如下。

例1 图的着色问题。给定一个图G，有三种颜色可以用于对图着色，要求相邻两个端点的颜色不同而且每个节点只有一种颜色。用$node(X)$表示顶点，边$edge(X,Y)$表示X,Y两个顶点之间有边，颜色用$col(C)$表示，三种颜色分别是$\{red, green, blue\}$，某个点着色用$color(X,C)$表示，则表示着色的析取逻辑程序为：

r_1: $node(X) \leftarrow color(X, red) \vee clor(X, green) \vee clor(X, blue)$

r_2: $\leftarrow color(X, C) \wedge color(Y, C) \wedge edge(X, Y) \wedge col(C)$

r_3: $color(X, red) \leftarrow node(X) \wedge not\ color(X, green) \wedge not\ color(X, blue)$

r_4: $color(X, blue) \leftarrow node(X) \wedge not\ color(X, red) \wedge not\ color(X, green)$

r_5: $color(X, green) \leftarrow node(X) \wedge not\ color(X, red) \wedge not\ color(X, blue)$

规则r_1为析取规则，表示每一个节点可从三种颜色中选取一种颜色着色。规则r_2表示相邻两个节点的颜色不能为同一个颜色。规则r_3、r_4、r_5表示若点着了红蓝绿其中一种颜色后，就不能再着其他颜色。

1.2 析取回答集程序的回答集

命题语言L的解释是一函数$I: A \rightarrow \{0,1\}$，其中$I(p) = 1$，表示给命题p赋值为真，$I(p) = 0$表示命题p赋值为假。一个解释I是来自集合A的原子的集合。若某个原子包含在I中，则表示该原子被赋值为真，否则赋值为假。

例2：若$I = \{b\}$，$P = b \leftarrow a$，则表示b赋值为真，且b为真时，命题P也为真。

若命题公式φ在解释I下为真，则称$I \models \varphi$。若I满足命题理论Σ中的所有公式，则称I满足Σ；若I满足命题公式(集)则称I是该命题公式集的模型。解释I是逻辑程序P的模型(表示I满足P)，记为$I \models P$，其中符号满足($\models$)定义为：

1) 对于一个原子p：$I \models p$当且仅当$p \in I$；$I \models \text{not } p$当且仅当$I \models p$不成立；

2) 对于一个集合S：$I \models S$当且仅当$I \models s$对于每一个$s \in S$，其中S的元素可以是原子p，或其缺省否定$\text{not } p$，或规则；

3) 对于一条规则r：$I \models r$当且仅当$I \models bd(r)$蕴含$I \models hd(r)$。

一个程序P的模型I是P的极小模型，当且仅当不存在另外一个模型I'使得$I' \subset I$。给定一个程序P和一个解释I，$P^I = \{hd(r) \leftarrow pos(r) | r \in P, neg(r) \cap I = \varnothing\}$是$P$关于$I$的G-L(Gelfond-Lifschitz)规约；若解释I是P^I的极小模型，则称I是程序P的回答集[18]。对于一个逻辑程序P和一个原子$a \in A$，程序P定义原子a的规则集合，记为$def_P(a) = \{r \in P | a \in hd(r)\}$，也称$Def_P(a)$中的规则定义了原子$a$。$Def_P(a, I)$表示在解释$I$下程序$P$中定义原子$a$的规则集合，记为：$def_P(a, I) = \{r \in P | a \in I \cap hd(r)\}$。$P$关于$I$的支撑规则集合定义为：$\{r \in P | I \models bd(r)\}$，记为$SuppR(P, I)$。

2 测试理论

这里详细介绍软件工程中的测试方法，然后根据析取回答集程序的特点提出测试的思想和测试用例的概念，主要分为两个部分：一是阐述析取回答集程序结构化测试方法的技术路线；二是定义结构化测试方法以及测试用例的概念。

2.1 析取回答集程序结构化测试技术路线

软件测试是使用人工或自动的手段来运行或测定某个软件系统的过程，其目的在于检验它是否满足规定的需求或弄清预期结果与实际结果之间的差别[19]。软件工程领域的测试包括黑盒测试和白盒测试。黑盒测试是将软件看成一个盒子，测试人员无须考虑其内部逻辑结构，根据规格说明书设计测试用例。黑盒测试重点关注软件实现和规格说明书的一致性，根据输入/输出确定的逻辑关系设计测试数据，以检查其是否正确。白盒测试又称结构测试、透明盒测试、逻辑驱动测试或基于代码的测试。白盒测试是一种测试用例设计方法，盒子指的是被测试的软件，白盒指的是盒子是可视的，即清楚盒子内部的东西以及里面是如何运作的。"白盒"法全面了解程序内部逻辑结构、对所有逻辑路径进行测试[20]。

要采用白盒测试,首先就必须要设计合理的软件测试用例。软件测试用例是指对一项特定的软件产品进行测试任务的描述,体现测试方案、方法、技术和策略。测试用例是为某个特殊目标而编制的一组测试输入、执行条件以及预期结果,用于核实是否满足某个特定软件需求[21]。因此,对析取逻辑程序测试的思路是:首先对析取回答集程序定义测试用例(Test Case)的概念。通过构造析取回答集程序测试用例的输入、输出,然后利用软件测试的各项覆盖标准来实现析取回答集程序的测试覆盖,在此过程中设计和提出析取回答集程序的测试相关概念。

此外,还考虑到析取回答集程序的语法的两个特点。第一个特点是:规则(rule)是程序最重要的语法结构之一,测试理论的主要对象为析取回答集程序的规则。对于析取回答集程序的规则测试,又可以细分出对规则体、规则头、规则集合等测试对象。

第二个特点是:析取回答集程序不是顺序执行的特点,无法对析取回答集程序的某个断点进行测试。根据一个测试输入的解释I运算出来的回答集与标准回答集进行比对,以此来验证程序是否符合预期。

综上,结构化测试方法使得测试人员更加清楚地了解软件的内部结构和运行机制,从而更好地设计测试用例,为后续的测试实验设计提供方便。而且结构化测试方法通过统计代码覆盖率、路径覆盖率等技术指标对代码的测试更加明确。

2.2 析取回答集程序的测试用例

首先给出析取回答集程序的测试用例的定义。设P是一个析取回答集程序,I_P、O_P分别是程序P的输入原子集合字母表和输出原子集合字母表,假设$I \subseteq I_P$,$P[I]$是逻辑程序P在解释I下的回答集,表示为$P[I] = AS(P \cup I)|_{O_P}$。定义一个$P$的规约满足映射关系$\sigma_P$,即程序$P$从集合$I_P$到集合的集合$O_P$上的映射关系是$\sigma_P$,定义为:$\sigma_P$:$I_P \to 2^{O_P}$。则给定一个输入$I$后,$P$的正确输出为$\sigma_P(I)$。由此对测试用例的定义如下。

定义1 (测试用例)P是一个任意的析取回答集程序,σ_P是P的一个规约,P和σ_P定义的测试用例为一对序偶$T = < I, O >$,其中$I \subseteq I_P$,$O = \sigma_P(I)$。程序P通过测试用例的条件为$P[I] = O$。

例1 设析取回答集程序P有如下三条规则:

$r_1: a \vee b$ $r_2: a \leftarrow b$ $r_3: b \leftarrow a$

其中$I_P = O_P = \{a, b\}$和σ_P组成以下的映射关系:

$\sigma_P(\varnothing) = \{\{a\}, \{b\}\}$,$\sigma_P(\{a\}) = \{\{a\}\}$,

$\sigma_P(\{b\}) = \{\{b\}\}$,$\sigma_P(\{a, b\}) = \{\{a, b\}\}$

通过程序P及其映射σ_P,很容易验证程序P在每一个测试用例$T = < I, O >$下测试均通过。一个析取回答集程序P和σ_P的测试套件$\mathcal{S}$是P和σ_P的测试用例的集合。程序P测试通过测试套件$\mathcal{S}$的条件是P测试通过$\mathcal{S}$中的每一个测试用例。程序P关于σ_P正确的条件是:当且仅当P测试通过P和σ_P的每一个测试用例。

3 覆盖函数以及覆盖率

通过实例分析,确定相关的测试覆盖指标,以此衡量测试用例的覆盖情况。

3.1 结构化测试的覆盖形式

结构化测试在测试执行时需要每一行代码至少执行一次(语句覆盖)、遍历所有的程序分支(分支覆盖)或者其他的可能的分支,其覆盖标准有逻辑覆盖、循环覆盖和基本路径测试。根据析取回答集程序的语法规则和测试对象的差异性,给出相应的覆盖函数定义以及各覆盖条件下的函数,并给出具体的计算方法。

3.2 覆盖函数

定义2 (覆盖函数)逻辑程序P和测试输入集合的覆盖函数γ定义为$\gamma: I \times P \to [0, 1]$,对于每个程序$P$和每个关于$P$的输入集合$\mathcal{I} \subseteq 2^{I_P}$,以下性质成立:

1) $\gamma(\mathcal{I}, P) = 1$,当且仅当$\mathcal{I} = 2^{I_P}$;

2) $\gamma(\mathcal{I}', P) \leqslant \gamma(\mathcal{I}, P)$,当且仅当$P$的每一个输入集合$\mathcal{I}' \subseteq \mathcal{I}$.

在这里说$\mathcal{I}$是解释的集合,当$\gamma(\mathcal{I}, P) = 1$时,满足的条件为$\mathcal{I} = 2^{I_P}$,可以生成所有逻辑程序$P$的覆盖。其次,析取回答集程序的覆盖函数的形式虽然与正规回答集程序相似,但是二者的含义却有不同,析取回答集程序产生的输入、输出集合的涵盖范围比正规回答集程序的范围大。

$$\text{覆盖率} = \frac{\text{至少被执行一次的语句(判定)数}}{\text{语句(判定)的总数}}$$

在软件工程中,如果一个测试用例使得被测试的文件(类或者函数)中所有的语句(判定)至少被执行到一次,那么这种测试覆盖准则被称为语句(判定)覆盖。软件工程中测试覆盖率的计算方法[20]如下:

考虑到析取回答集程序中没有语句、分支的概念,因此假设在输入$\mathcal{I} \subseteq 2^{I_P}$的情况下,其规则、定义、环以及子程序的覆盖数量与全部输入案例产生的规则、定义、环以及子程序的覆盖数量比值作为覆盖率。析取回答集程序的覆盖率公式定义为:

$$\mathcal{C}_X(\mathcal{I}, P) = \begin{cases} \dfrac{covered_X(\mathcal{I}, P)}{covered_X(2^{I_P}, P)}, & covered_X(2^{I_P}, P) > 0 \\ 1, & \text{其他} \end{cases}$$

根据以上的覆盖率计算公式,可以计算出析取回答集程序的结构化测试方法中各覆盖形式下的覆盖率。

3.3 析取回答集程序的测试覆盖率

由于在析取回答集程序测试中,代码被逻辑规则替代,在回答集程序中,每行规则都会进行执行。因此需要测试回答集程序中的每一条规则。在析取回答集程序中,所谓的测试覆盖率是测试在给定的解释I下,规则r是否满足,而规则又可分为规则的头、规则的体、规则集合等不同的测试对象,根据测试对象的满足情况,得到析取回答集程序的以下各种覆盖。

在面向过程的程序测试方法中,路径覆盖法主要是通过测试用例的执行,覆盖程序中所有可能的路径。由于在析取回答集程序中,规则与规则不会产生分支执行,因此使用部分规则组成的集合作为一个子程序,通过对子程序的覆盖来模拟路径覆盖的概念。

定义3 (子程序覆盖)设P是一个析取回答集程序,$I \subseteq I_P$且$P' \subseteq P$。当$P' = SuppR(P, X)$对某回答集$X \in AS(P \cup I)$成立时,I覆盖P'。

例2 设析取回答集程序P有例1中的三条规则:

$r_1: a \vee b$ $r_2: a \leftarrow b$ $r_3: b \leftarrow a$

其中$I_P = O_P = \{a, b\}$,$I = \{a, b\}$,有$X \in AS(P \cup I) = \{\{a, b\}\}$,$P' = \{r_1, r_2, r_3\}$,可知$P' = SuppR(P, X)$,$I$覆盖$P'$。按照计算覆盖率的方法,很容易知道对于任意的$I \subseteq I_P$都有$AS(P \cup I) = \{\{a, b\}\}$,即对于任意的$I \subseteq I_P$都覆盖了同一个$SuppR(P, X)$,根据上面的计算式有:

$$C_P(\mathcal{I}, P) = \frac{covered_P(\mathcal{I}, P)}{covered_P(2^{I_P}, P)} = \frac{1}{1} = 1$$

在结构化测试方法中,由于面向过程的编程语言并非每行代码都会执行,只能通过执行过的代码块占总代码的比率,判断测试用例的好坏。语句覆盖是结构化测试方法中的一个重要指标,要求测试过程中,测试用例尽可能地测试到每一条代码。以此衡量测试用例是否真正完全覆盖了应用程序代码中的各种可能以及在运行这些测试用例时执行了多少代码。所以通过测试规则体的是否满足来模拟语句覆盖。

定义4 (规则覆盖)给定一个析取回答集程序P,$r \in P$,$I \subseteq I_P$。如果对某个回答集$X \in AS(P \cup I)$时,有$X \models bd(r)$成立,则规

则$r \in P$被I正覆盖。如果对回答集$X \in AS(P \cup I)$,有$X \models bd(r)$,则r被I负覆盖。此外,r被测试用例T正覆盖,当且仅当r被I正覆盖。

例3 设析取回答集程序P为例1中的程序,其中$I_P = O_P = \{a, b\}$,$I_1 = \{a, b\}$和$I_2 = \{a\}$,则有$AS\left(P \cup I_1\right) = \{\{a, b\}\}$,可知$I_1$覆盖了规则$r_1, r_2, r_3$;对于$AS\left(P \cup I_2\right) = \{\{a, b\}\}$,可知$I_2$覆盖规则$r_1, r_2, r_3$。

在计算覆盖率方面,用$covered_{R^+}(\mathcal{I}, P)$表示程序$P$中解释$I \in \mathcal{I}$下正覆盖的规则数量(相应的$covered_{R^-}(\mathcal{I}, P)$表示程序$P$中解释$I \in \mathcal{I}$下所负覆盖的规则数量),易知下面的计算式是成立的:$covered_R(\mathcal{I}, P) = covered_{R^+}(\mathcal{I}, P) + covered_{R^-}(\mathcal{I}, P)$,然后再根据计算式$\dfrac{covered_X(\mathcal{I}, P)}{covered_X(2^{I_P}, P)}$计算出覆盖率:对于任意的$I \subseteq I_P$可得其规则正负覆盖函数分别是$covered_{R^+}(I, P) = 3$,$covered_{R^-}(I, P) = 0$,而$covered_R(I, P) = 3$。因此有:

$$C_R(\mathcal{I}, P) = \frac{covered_R(\mathcal{I}, P)}{covered_R\left(2^{\mathbb{I}_P}, P\right)} = \frac{3}{3} = 1$$

引理1 设P是一个析取回答集程序,$r \in P$,$X \in \{P, R\}$,$\mathcal{I} \subseteq 2^{I_P}$,$J \in I$,并且$AS(P \cup I) = M$对于每个$I \in \mathcal{I}$成立,即$P \cup I$对于每个$I \in \mathcal{I}$都有相同的唯一回答集$M$。则:

1) r被$I \in \mathcal{I}$正覆盖当且仅当r被J正覆盖。

2) $covered_X(\mathcal{I}, P) = covered_X(I, P)$对于每一个$I \in \mathcal{I}$成立。

证明 r被$I \in \mathcal{I}$正覆盖,当且仅当$M \models bd(r)$对于某个回答集$M \in AS(P \cup I)$成立。又$M \in AS(P \cup I)$,则$M \models bd(r)$。对于某个回答集$M \in AS(P \cup I)$来说,因为$AS(P \cup J) = M$,,$M \models bd(r)$,所以r被J正覆盖。同理可证2)。

推论1 设P是一个析取回答集程序,如果$P \cup I$对任意I都有唯一的回答集,则对于任何非空的$I \subseteq 2^{I_P}$和$X \in \{P, R\}$,有$C_X(I, P) = 1$。

证明 设$X = R$并且有$M = AS(P \cup I)$对于任意的$I \subseteq 2^{I_P}$成立,假设$covered_R(2^{I_P}, P) > 0$,则:

$$C_R(\mathcal{I}, P) = \frac{\text{cov}\,ered_R(\mathcal{I}, P)}{covered_R(2^{I_P}, P)} = \frac{1}{1} = 1$$

其中$I \subseteq 2^{I_P}$,又根据引理1的2)可知:

$$\mathcal{C}_R(I, P) = \frac{covered_R(\mathcal{I}, P)}{covered_R\left(2^{I_P}, P\right)} =$$

$$\frac{covered_{R^+}(\mathcal{I}, P) + covered_{R^-}(\mathcal{I}, P)}{covered_{R^+}\left(2^{I_P}, P\right) + covered_{R^-}\left(2^{I_P}, P\right)} =$$

$$\frac{covered_{R^+}(I, P) + covered_{R^-}(I, P)}{covered_{R^+}(I, P) + covered_{R^-}(I, P)} = 1$$

在关注规则的满足性时主要用到的是规则覆盖,但是在计算不同规则头之间是否递归满足时,需要用到环覆盖的概念。

定义5 (环覆盖)设P是一个析取回答集程序,$I \subseteq I_P$。L_P是P的一个环当且仅当有某个回答集$X \in AS(P \cup I)$,使得原子a由$r \in SuppR(P, X)$所定义,且对于定义的每个原子a都有$a \in L_P$,则P的环L_P被I正覆盖。若有一个规则$r \in SuppR(P, X)$定义了a,但是对于某个回答集$X \in AS(P \cup I)$,$def_P(a, X) \neq \varnothing$,若没有规则$SuppR(P, X)$定义$a$,则$L_P$被$I$负覆盖。

例4 以例1中的程序为例,其中$I_P = O_P = \{a, b\}$,该程序的回答集是$AS(P \cup I) = \{\{a, b\}\}$,该回答集对应的环除单环$\{a\}$,$\{b\}$外,还有环$L_P = \{a, b\}$。同理,在计算环覆盖时用如下方法:

参照例1的程序,知道对于任意的$I \subseteq I_P$都有$AS(P \cup I) = \{\{a, b\}\}$,设$I = \{a, b\} \subseteq I_P$,程序$P$存在环$L_P = \{a, b\}$,则$covered_{L^+}(I, P) = 1$,$covered_{L^-}(I, P) = 0$,根据上面的计算式计算环覆盖率,有:

$$C_L(\mathcal{I}, P) = \frac{covered_L(\mathcal{I}, P)}{covered_L(2^{I_P}, P)} = \frac{1}{1} = 1$$

与面向过程的编程语言不同,析取回答集程序的规则中包含了规则的头(head),为了测试每条规则的头是否满足,给出定义覆盖的概念。

定义6 (定义覆盖)设P是一个析取回答集程序,$a \in A$,$I \subseteq I_P$。若对于某个回答集$X \in AS(P \cup I)$,存在规则r满足$r \in SuppR(P, X)$定义了原子a,记为$def_P(a, X) = \{r \in P | a \in X \cap hd(r)\}$,称原子$a$被$I$正覆盖。若$def_P(a, X) \neq \varnothing$且不存在规则$r \in SuppR(P, X)$定义的$a$,则称$a$被$I$负覆盖。

例5 以例1中程序P为例,其中$I_P = O_P = \{a, b\}$,该程序的回答集是$AS(P \cup I) = \{\{a, b\}\}$,该回答集对应的规则正覆盖了原子$a$,原子$b$。同理,对于任意的$I \subseteq I_P$,可得其规则正覆盖函数$covered_{D^+}(I, P) = 2$,负覆盖函数$covered_{D^-}(I, P) = 0$,而$covered_D(\{I\}, P) = 2$,因此有:

$$C_D(\mathcal{I}, P) = \frac{covered_D(\mathcal{I}, P)}{covered_D(2^{I_P}, P)} = \frac{2}{2} = 1.$$

环覆盖和定义覆盖有一定的相似性,原因是在于两者都是为了测试规则头的满足性,但是两者测试的规则头又有不同的特点。定义覆盖是针对单个的规则头,即单个的原子满足性。而环覆盖则是为了测试规则头的集合,在这个规则头集合中,原子之间存在着一定的递归满足关系。

由以上的各覆盖测试条件,设$covered_*(\mathcal{I}, P)$表示规则(定义、环、子程序)的正覆盖,$covered_{*^-}(\mathcal{I}, P)$表示规则(定义、环、子程序)的负覆盖,总结得到覆盖公式的计算式为:

$covered_*(\mathcal{I}, P) = covered_{*^+}(\mathcal{I}, P) + covered_{*^-}(\mathcal{I}, P)$,其中$* \in \{R, D, L, P\}$。

性质1 $covered_*(\varnothing, P) = 0$对任意的逻辑程序$P$成立,其中$* \in \{R, D, L, P\}$。

推论2 P是析取回答集程序,若$P \cup I$对任意$I \subseteq I_P$都有唯一的回答集,则有$SuppR(P, X) = P$对于任意的$I \subseteq 2^{I_P}$和$X \in \{P, R\}$成立。

证明采用反证法证明。假设$SuppR(P, X) \neq P$且$P \cup I$有唯一的回答集,对于任意的$I \subseteq I_P$成立。假设存在某条规则$r \in P$,并且使得对于回答集$X \in AS(P \cup I)$有$r \notin SuppR(P, X)$。由于$AS(P \cup I)$有唯一的回答集,则根据推论1可知$\mathcal{C}_R(I, P) = 1$。若此时$r \notin SuppR(P, X)$,则有$covered_X\left(2^{I_P}, P\right) > covered_X(I, P)$,且$\mathcal{C}_R(I, P) \neq 1$。这与假设矛盾,证毕。

定义7 (支撑规则比值函数)函数α是指对于某个析取回答集程序P和它的解释$I \subseteq I_P$,其支撑规则集合的势与程序规则组成的集合的势之比。其中,支撑规则集合的势作为分子,析取回答集程序P的规则组成的集合的势作为分母。记为:

$$\alpha = \frac{\left|\left|SuppR(P, X)\right|\right|}{||P||} * 100\%$$

α是一个重要的指标,主要用于衡量和计算一个析取回答集程序中,其解释I对于某个回答集$X \in AS(P \cup I)$中支撑规则与程序规则的势之比,由此可以评估出逻辑程序P在解释I的非支撑规则的冗余数量,此外还可以评估解释I的有效性。

例6 设逻辑程序P有如下规则:

$r_1: e \leftarrow d, not\, f$; $r_2: d \leftarrow a, b, not\, c$; $r_3: f \leftarrow c, not\, e$;

$I = \{a, b\}$,对该程序求解回答集得到回答集:$AS(P \cup I) =$

$\{\{d,b,a,e\}\}$,其支撑集合的势为$\left|\left|SuppR(P,X)\right|\right|=4$,$||P||=5$,$\alpha=80\%$。

推论3 P是一个析取回答集程序,若$P\cup I$对任意I都有唯一的回答集,则对于任何非空的$I\subseteq 2^{I_P}$,则有$\alpha=100\%$。

证明 由推论2可知,$SuppR(P,X)=P$,则有$||SuppR(P,X)||=||P||$。根据支撑规则函数的计算式有:

$$\alpha=\frac{\left|\left|SuppR(P,X)\right|\right|}{||P||}*100\%=1*100\%=100\%$$

4 结语

本文推广了命题正规逻辑程序结构化测试基本思想到命题析取逻辑程序,提出了命题析取逻辑程序规则覆盖、子程序覆盖、定义覆盖、环覆盖等基本的结构化测试评价准则,并探讨了它们的一些基本性质。接下来将进一步推广这些基本概念到带变元、带聚合函数(sum、min、max等)、以及抽象约束原子的一般逻辑程序,并进一步实现一般回答集程序的自动结构化测试。

参考文献(References)

[1] GELFOND M , LIFSCHITZ V . Classical negation in logic programs and deductive databases [J]. New Generation Computing, 1991, 9(3):365-385.

[2] NGUYEN V, VASILEIOU S L, SON T C, et al. Explainable planning using answer set programming [C]// Proceedings of the 2020 17th International Conference on Principles of Knowledge Representation and Reasoning. California: IJCAI,2020: 662-666.

[3] RIZWAN M, PATOGLU V, ERDEM E. Human robot collaborative assembly planning: an answer set programming approach [J]. Theory and Practice of Logic Programming, 2020, 20(6): 1006-1020.

[4] FALKNER A, FRIEDRICH G, SCHEKOTIHIN K , et al. Industrial applications of answer set programming [J]. KI-Künstliche Intelligenz, 2018(2):1-12.

[5] IZMIRLIOGLU Y, ERDEM E. Reasoning about cardinal directions between 3-dimensional extended objects using answer set programming [J]. Theory and Practice of Logic Programming, 2020, 20(6): 942-957.

[6] 徐珩僭,王以松,冯仁艳. 一种用于Slater与Kemeny选举求解的ASP方法[J]. 计算机工程,2019,45(09):198-203.(XU H J, WANG Y S, FENG R Y. An ASP method for calculating slater and kemeny voting [J]. Computer Engineering, 2019, 45 (09): 198-203.)

[7] EITER T, FABER W, LEONE N, et al. The diagnosis frontend of the dlv system[J]. AI Communications, 1999, 12(1/2): 99-111.

[8] ERDEM E, WONG M D F. Rectilinear steiner tree construction using answer set programming [C]// Proceedings of the 2004 International Conference on Logic Programming. Berlin: Springer, , 2004: 386-399.

[9] SUCHAN J, BHATT M, VARADARAJAN S. Out of Sight But Not Out of Mind: An Answer Set Programming Based Online Abduction Framework for Visual Sensemaking in Autonomous Driving [C]// Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI 2019), Macao, China, August 10-16, 2019. ijcai. org, 2019: 1879-1885.

[10] JANHUNEN T, NIEMELÄ I, OETSCH J, et al. On Testing Answer-Set Programs[C]//Proceedings of the 2010 conference on ECAI 2010: 19th European Conference on Artificial Intelligence. 2010: 951-956.

[11] JANHUNEN T, NIEMELÄ I, OETSCH J, et al. Random vs. structure-based testing of answer-set programs: An experimental comparison[C]// Proceedings of the 2011 International Conference on Logic Programming and Nonmonotonic Reasoning. Berlin: Springer , 2011: 242-247.

[12] OETSCH J, PRISCHINK M , PÜHRER J, et al. On the small-scope hypothesis for testing answer-set programs [C] //Proceedings of the 2012 Thirteenth International Conference on Principles of Knowledge Representation and Reasoning. Palo Alto, CA: AAAI Press:2012: 43-53.

[13] FEBBRARO O, LEONE N, REALE K, et al. Unit testing in ASPIDE [M]//Applications of Declarative Programming and Knowledge Management. Berlin:Springer, 2011: 345-364.

[14] BUSONIU P A, OETSCH J, PÜHRER Jet al. SeaLion: An eclipse-based IDE for answer-set programming with advanced debugging support [J]. Theory and Practice of Logic Programming, 2013, 13(4-5): 657-673.

[15] GREBLER A, OETSCH J, TOMPITS H. Harvey: a system for random testing in ASP[C]// Proceedings of the 2017 International Conference on Logic Programming and Nonmonotonic Reasoning. Springer, Cham, 2017: 229-235.

[16] AMENDOLA G, BEREI T, RICCA F. Testing in ASP: revisited language and programming environment [C]// Proceedings of the 2021 European Conference on Logics in Artificial Intelligence. Springer, Cham,2021: 362-376.

[17] LEONE N, RULLO P, SCARCELLO F. Disjunctive Stable Models: Unfounded Sets, Fixpoint Semantics, and Computation [J]. Information and Computation, 1997, 135(2): 69-112.

[18] GELFOND M. The Stable Model Semantics for Logic Programming [C]//Proceedings of the 5th International Conference and Symposium on Logic Programming. 1989: 1070-1080.

[19] 张涛. 软件技术基础实验教程[M]. 西安:西北工业大学出版社, 2015: 111-112.(ZHANG T. Basic Experiment Tutorial of Software Technology [M]. Xi'an: Northwestern Polytechnical University Press, 2015, 111-112)

[20] 蔡立志. 软件测试导论[M]. 北京:清华大学出版社, 2016:27-89.(CAI L Z . Introduction to Software Testing [M]. Beijing: Tsinghua University Press, 2016:27-89.)

[21] 李香菊. 软件工程课程设计教程[M]. 北京:北京邮电大学出版社, 2016: 72-90.(LI X J. Design Tutorial of Software Engineering [M]. Beijing: Beijing University of Posts and Telecommunications Press, 2016:72-90.)

This work is partially supported by National Natural Science Foundation of China (61976065).

YANG Dong, born in 1987, M. S candidate. His research interests include artificial intelligence, knowledge representation and reasoning.

WANG Yisong, born in 1975, Ph.D., professor. His research interests include artificial intelligence, knowledge representation and reasoning.