



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени
Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчёт по лабораторной работе №7 по дисциплине "Проектирование экспертных систем"

Тема Алгоритм прямого логического вывода на обобщенных правилах продукции

Студент Варламова Е. А.

Группа ИУ7-33М

Оценка (баллы) _____

Преподаватели Русакова З.Н.

Москва — 2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1 Используемые структуры данных	4
2 Алгоритм прямого логического вывода на обобщенных правилах продукции	5
2.1 Алгоритм поиска по графу в ширину от данных	5
2.1.1 Основной метода поиска	5
2.1.2 Метод потомки	5
2.2 Метод унификации	6
3 Реализация	7
4 Пример работы	11
4.1 Программная реализация задачи	11
ЗАКЛЮЧЕНИЕ	15

ВВЕДЕНИЕ

Цель работы – реализовать алгоритм прямого логического вывода на обобщенных правилах продукции.

Для достижения поставленной цели потребуется:

- описать используемые структуры данных;
- описать алгоритм прямого логического вывода на обобщенных правилах продукции;
- привести реализацию алгоритма;
- привести примеры работы алгоритма.

1 | Используемые структуры данных

Разработаем класс переменной. Поля класса:

- имя;
- тип (переменная).

Разработаем класс константы. Поля класса:

- значение;
- тип (не переменная).

Разработаем класс вершины (атома). Поля класса:

- имя;
- список термов (терм – константа или переменная).

Разработаем класс подстановок. Поля класса:

- словарь переменных (ключ – имя переменной, значение – связанная переменная или константа);
- словарь ссылок (ключ – значение константы, значение – список переменных, имеющих значение этой константы).

Разработаем класс правила. Поля класса:

- список входных вершин;
- целевая вершина;
- номер правила;
- флаг правила – открыто/закрыто.

2 | Алгоритм прямого логического вывода на обобщенных правилах продукции

2.1 Алгоритм поиска по графу в ширину от данных

Вход:

- доказанные вершины;
- целевая вершина.

Выход:

- информация о решении.

2.1.1 Основной метода поиска

Отметить все доказанные вершины как закрытые. Поа флаги решения истинны, выполняем:

- вызвать метод потомки, который возвращает количество закрытых правил;
- если флаг решение найдено, то выход с сообщением об успешном поиске;
- если количество закрытых правил равно 0, то выход с сообщением о неудачном поиске;

2.1.2 Метод потомки

В цикле по списку правил выполнить.

- Если правило не открыто, то пропустить его.
- Если все выходные вершины правила покрываются закрытыми вершинами, то пометить правило закрытым, пометить выходную вершину правила как закрытую. Если выходная вершина равна целевой, то изменить флаг решение найдено. Увеличить количество закрытых правил.

2.2 Метод унификации

Вход:

- 2 атома;
- подстановка.

Выход:

- информация о возможности унификации.

Унификация выполняется:

1. Если термы константы, то они унифицируемы если совпадают.
2. Если в первом атоме терм переменная, а во втором константа, то они унифицируемы и переменная получает значение константы.
3. Если терм в первом атоме переменная и во втором тоже, то они унифицируемы и становятся связанными.
4. Если в первом атоме терм переменная, а во втором функция от переменных, то они унифицируемы и вместо переменной подставляется функция (x и $f(x)$ не унифицируемы).

3 | Реализация

Листинг 3.1: Структуры данных

```
1 class Label(Enum):
2     OPEN = 0
3     CLOSE = 1
4
5 class Constant:
6     def __init__(self, value):
7         self.value = value
8         self.variable = False
9
10    def __str__(self):
11        return self.value
12
13    def __repr__(self):
14        return self.__str__()
15
16
17 class Variable:
18     def __init__(self, name):
19         self.name = name
20         self.variable = True
21
22     def __str__(self):
23        return self.name
24
25     def __repr__(self):
26        return self.__str__()
27
28
29 class Table:
30     def __init__(self):
31         self.variables = dict()
32         self.links = dict()
33
34     def reset(self, other):
35         self.variables = other.variables
36         self.links = other.links
37
38     def val(self, var):
```

```

39         return self.variables[var.name]
40
41     def var_links(self, var):
42         return self.links[self.variables[var.name]]
43
44     def __str__(self):
45         res = ""
46         for const in self.links.keys():
47             res += str(self.links[const]) + ": " + str(const) + "\n"
48         return res
49
50 class Node:
51     def __init__(self, name, terminals):
52         self.name = name
53         self.terminals = terminals
54
55     def __str__(self):
56         strterms = ""
57         for term in self.terminals:
58             strterms += str(term) + ", "
59         return self.name + '(' + strterms.strip(", ") + ')'
60
61     def __repr__(self):
62         a = self.__str__()
63         return f"{a}"
64
65     def print(self):
66         a = self.__str__()
67         print(f"{a}", end = " ")

```

Листинг 3.2: класс поиска

```

1 class Search:
2     def __init__(self, rule_arr: [Rule]):
3         self.rule_arr = rule_arr
4
5         self.goal_node = None
6         self.table = Table()
7         self.solution_flg = 1
8         self.no_solution_flg = 1
9         self.closed_arr = []
10
11     def run(self, goal_node: Node, in_node_arr: [Node]):
12         self.goal_node = goal_node
13         self.set_nodes_closed(in_node_arr)
14
15         while self.solution_flg and self.no_solution_flg:
16             rule_cnt = self.parent_search()
17
18             if self.solution_flg == 0:

```



```

19         return
20
21     if rule_cnt == 0:
22         self.no_solution_flg = 0
23         print("                                ")
24
25
26 def parent_search(self):
27     cnt_rules = 0
28
29     for rule in self.rule_arr:
30         if self.solution_flg:
31             if rule.label != Label.OPEN:
32                 continue
33             print("\n")
34             print(rule, "                                ")
35             if self.close_goal_if_close_nodes_cover(rule.node_arr, rule.
36 out_node):
37                 print(f'                                {rule.number}:
38                                     ')
39                 rule.label = Label.CLOSE
40
41                 if unification(self.table, rule.out_node, self.goal_node
42 ):
43                     self.solution_flg = 0
44                     print(f'                                {rule.number}
45                                     ')
46
47                     cnt_rules += 1
48
49     else:
50         break
51
52     print(f'                                : ', end='')
53     self.print_closed_rules()
54     return cnt_rules
55
56 def close_goal_if_close_nodes_cover(self, in_node_arr: [Node], goal_node
57 ):
58     print("                                : ", self.closed_arr)
59     print("                                : ",
60 in_node_arr, " -> ", goal_node)
61
62     table = copy.deepcopy(self.table)
63     for node in in_node_arr:
64         found = False
65         for node_closed in self.closed_arr:
66             if unification(table, node, node_closed):
67                 found = True
68                 break

```

```

62         if not found:
63             print("
64                 node)
65             return False
66     self.table = table
67
68     new_terminals = []
69     for term in goal_node.terminals:
70         if term.variable:
71             new_terminals.append(self.table.variables[str(term)])
72         else:
73             new_terminals.append(term)
74     goal_node.terminals = new_terminals
75     self.closed_arr.append(goal_node)
76     print("
77         ;
78         : ", self.closed_arr)
79
80     return True
81
82 def set_nodes_closed(self, node_arr):
83     for node in node_arr:
84         self.closed_arr.append(node)

```

4 | Пример работы

4.1 Программная реализация задачи

```
c_N = Constant('N')
c_M1 = Constant('M1')
c_W = Constant('W')
c_A1 = Constant('A1')

v_x = Variable("x")
v_y = Variable("y")
v_z = Variable("z")
v_x1 = Variable("x1")
v_x2 = Variable("x2")
v_x3 = Variable("x3")

rule_arr = [
    Rule(1, Node("C", [v_x]), [Node("W1", [v_y]),
                                Node("A", [v_x]),
                                Node("S", [v_x, v_y, v_z]),
                                Node("H", [v_z])]),
    Rule(2, Node("S", [c_W, v_x1, c_N]), [Node("M", [v_x1]),
                                           Node("O", [c_N, v_x1])]),
    Rule(3, Node("W1", [v_x2]), [Node("M", [v_x2])]),
    Rule(4, Node("H", [v_x3]), [Node("E", [v_x3, c_A1])]),
]

facts = [Node("O", [c_N, c_M1]),
         Node("M", [c_M1]),
         Node("A", [c_W]),
```

```
Node("E", [c_N, c_A1]))  
Search(rule_arr).run(Node("C", [c_W]), facts)
```

Результат:

Правило #1: $[W1(y), A(x), S(x, y, z), H(z)] \rightarrow C(x)$ сейчас обрабатывается
доказанные факты: $[O(N, M1), M(M1), A(W), E(N, A1)]$
в процессе доказательства: $[W1(y), A(x), S(x, y, z), H(z)] \rightarrow C(x)$
Не удалось унифицировать: $W1(y)$

Правило #2: $[M(x1), O(N, x1)] \rightarrow S(W, x1, N)$ сейчас обрабатывается
доказанные факты: $[O(N, M1), M(M1), A(W), E(N, A1)]$
в процессе доказательства: $[M(x1), O(N, x1)] \rightarrow S(W, x1, N)$
Удалось унифицировать; доказанные факты сейчас:
 $[O(N, M1), M(M1), A(W), E(N, A1), S(W, M1, N)]$
Правило 2: все вершины закрыты

Правило #3: $[M(x2)] \rightarrow W1(x2)$ сейчас обрабатывается
доказанные факты: $[O(N, M1), M(M1), A(W), E(N, A1), S(W, M1, N)]$
в процессе доказательства: $[M(x2)] \rightarrow W1(x2)$
Удалось унифицировать; доказанные факты сейчас:
 $[O(N, M1), M(M1), A(W), E(N, A1), S(W, M1, N), W1(M1)]$
Правило 3: все вершины закрыты

Правило #4: $[E(x3, A1)] \rightarrow H(x3)$ сейчас обрабатывается
доказанные факты: $[O(N, M1), M(M1), A(W), E(N, A1), S(W, M1, N), W1(M1)]$
в процессе доказательства: $[E(x3, A1)] \rightarrow H(x3)$
Удалось унифицировать; доказанные факты сейчас:
 $[O(N, M1), M(M1), A(W), E(N, A1), S(W, M1, N), W1(M1), H(N)]$
Правило 4: все вершины закрыты
Закрытые правила сейчас: 2 3 4

Правило #1: $[W1(y), A(x), S(x, y, z), H(z)] \rightarrow C(x)$ сейчас обрабатывается
доказанные факты:
 $[O(N, M1), M(M1), A(W), E(N, A1), S(W, M1, N), W1(M1), H(N)]$
в процессе доказательства: $[W1(y), A(x), S(x, y, z), H(z)] \rightarrow C(x)$

Удалось унифицировать; доказанные факты сейчас:

$[O(N, M1), M(M1), A(W), E(N, A1), S(W, M1, N), W1(M1), H(N), C(W)]$

Правило 1: все вершины закрыты

Правило 1 имеет выходную вершину равную целевой

Закрытые правила сейчас: 1 2 3 4

ЗАКЛЮЧЕНИЕ

В результате работы цель была достигнута.

Для достижения поставленной цели потребовалось:

- описать используемые структуры данных;
- описать алгоритм прямого логического вывода на обобщенных правилах продукции;
- привести реализацию алгоритма;
- привести примеры работы алгоритма.