Бюджетное учреждение высшего образования

Ханты-Мансийского автономного округа – Югры

**«Сургутский государственный университет»**

Отчет по лабораторной работе № 4

**«ОБРАТНАЯ ЦЕПОЧКА РАССУЖДЕНИЙ»**

**Выполнил:**

Студент 3-ого курса

Гр. 607-91

Тунян Э.Г.

**Проверил:**

Шайторова И. А.

**2022 г.**

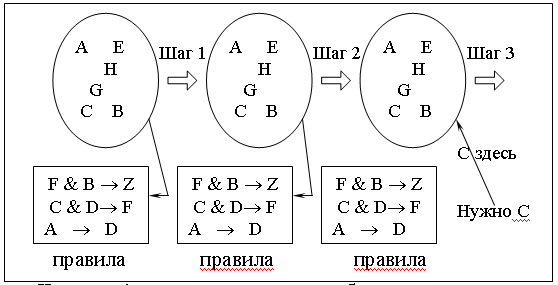
**Цель работы:**

Изучение способа вывода новых знаний с помощью обратной цепочки рассуждений. Научиться строить дерево целей и разрабатывать алгоритм.

**1.Теоретическая часть**

Прямой метод рассуждений имеет следующий недостаток. При большом количестве правил, чтобы найти информацию, связанную с Z, нужно выполнить много правил, не связанных с Z. При этом метод оказывается напрасной тратой времени и денег.

В таких ситуациях более рентабельной является обратная цепочка рассуждений.



*Рис.4 Пример реализации обратной цепочки рассуждений*

При этом методе система начинает с того, что нужно доказать, например, что ситуация Z существует, и нужно выполнить только те правила, которые относятся к установлению этого факта.

На шаге 1 системе говорится, чтобы она установила, что ситуация Z существует. Она ищет Z в базе, а если его нет, будет искать правило, приводящее к установлению Z. Она находит правило

F & B → Z

и решает, что надо установить F и B.

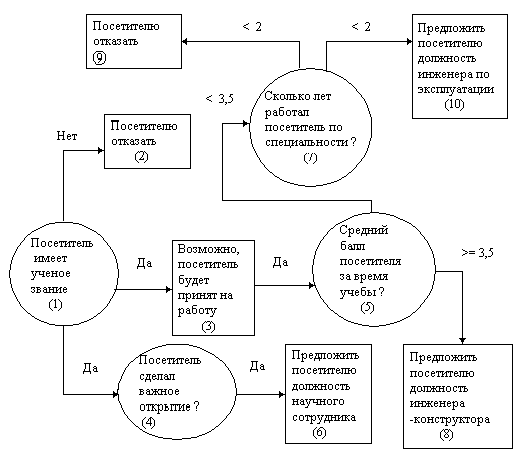
На шаге 2 система пытается найти факт F или в базе данных или среди правил. Находит правило C & D→ F и решает, что необходимо установить существование фактов C и D.

На шагах 3-5 система находит C, затем находит А прежде, чем получит заключение о D.

На шагах 6-8 система выполняет третье правило, чтобы установить D, затем исполняет второе правило, чтобы установить F и наконец – первое правило, чтобы установить основную цель – факт существования Z.

Теперь нужно наглядно её представить. Для описания подобных задач обычно используются диаграммы, которые называются ***деревьями решений.*** Деревья решений дают необходимую наглядность и позволяют проследить ход рассуждений.

Диаграммы называются деревьями решений потому, что, подобно настоящему дереву, имеют ветви. Ветви деревьев решений заканчиваются логическими выводами. Для рассматриваемого примера вывод заключается в том, предложит ли директор должность поступающему на работу, и если да, то какую. Многие задачи сложны, и их непросто представить (или для их решения не собираются использовать ЭС). Дерево решений помогает преодолеть эти трудности.



*Рис. 5. Дерево решений для выбора должности*.

На рис.5 показано дерево решений для примера с приёмом на работу. Видно, что диаграмма состоит из кружков и прямоугольников, которые называются вершинами. Каждой вершине присваивается номер. На вершины можно ссылаться по этим номерам. Линии, соединяющие вершины, называются дугами или ветвями. Кружки, содержащие вопросы, называются вершинами решений. Прямоугольники содержат цели диаграммы и означают логические выводы. Линии показывают направление диаграммы. Многие вершины имеют сразу по несколько ветвей, связывающих их с другими вершинами. Выбор выходящей из вершины ветви определяется проверкой условия, содержащегося в вершине.

Например, вершина 5 (см. рис.5) содержит вопрос, на который есть два возможных ответа, и поэтому у неё два пути в зависимости от среднего балла посетителя за время учёбы, то есть возможен выбор одной из двух ветвей. Если средний балл равен 3.1, то будет выбран первый путь, так как 3.1 меньше 3.5. В программе под средний балл сначала отводится переменная, а затем ей присваивается значение. Можно сказать, что вершины содержат переменные, а пути - это условия, в соответствии с которыми переменным присваиваются значения. После того как для проблемной области сформулированы правила, эти условия становятся условными частями (ЕСЛИ) правила. Прямоугольники содержат частные или общие выводы. Например, прямоугольник на рис.1 может содержать ответ на вопрос, будет ли посетителю предложена работа. Общая цель системы, в которой реализованы обратные рассуждения, - получить окончательный ответ. Локальной целью может быть содержащийся в прямоугольнике на рис.8 ответ на вопрос, будет ли посетителю предложена должность. Однако эта вершина имеет и исходящие ветви, и, следовательно, через неё может проходить путь к следующему логическому выводу. В последнем случае, поскольку исходящая ветвь не содержит условия, и она только одна, говорят, что вершина содержит локальный вывод для другой цели. Локальный вывод — это также составляющая условной части правила.

**Обобщённый алгоритм работы системы с обратной цепочкой выводов.**

Система, реализующая обратную цепочку рассуждений, должна выполнять следующие шаги:

1. Определить переменную логического вывода.
2. В списке логических выводов искать первое вхождение этой переменной. Если переменная найдена, в стек логических выводов поместить номер соответствующего правила и установить номер условия равным 1. Если переменная не найдена, сообщить пользователю, что ответ найти невозможно.
3. Присвоить значения всем переменным условия из данного правила.
4. Если в списке переменных указано, что какой-либо переменной условия не присвоено значение и её нет среди переменных логического вывода (её нет в списке логических выводов), запросить её значение у пользователя.
5. Если какая-либо переменная условия входит в переменные логического вывода, поместить в стек номер правила, в логический вывод которого она входит, и вернуться к шагу 3.
6. Если из правила нельзя определить значение переменной, удалить соответствующий ему элемент из стека и в списке логических выводов продолжить поиск правила с этой переменной логического вывода.
7. Если такое правило найдено, перейти к шагу 3.
8. Если переменная не найдена ни в одном из оставшихся правил в логическом выводе, правило для предыдущего вывода не верно. Если предыдущего вывода не существует, сообщить пользователю, что ответ получить невозможно. Если предыдущий вывод существует, вернуться к шагу 6.
9. Определить значение переменной из правила, расположенного в начале стека; правило из стека удалить. Если есть ещё переменные логического вывода, увеличить значение номера условия и для проверки оставшихся переменных вернуться к шагу 3. Если больше нет переменных логического вывода, сообщить пользователю окончательный вывод.

**Переменные:**

ПК – персональный компьютер (*запускается; не запускается*);

ПР – проводка (*есть проблемы; нет проблем*)

В – вывод (*исправен; не исправен*)

ОС – операционная система (*начинает загрузку; не начинает загрузку*)

ПСЖ – проблема с железом (*есть; нет*)

ОЗУ – оперативно запоминающее устройство (*исправен; не исправен*)

ВК – видеокарта (*исправен; не исправен*)

ЦП – центральный процессор (*исправен; не исправен*)

СЭ – синий экран (*есть; нет*)

СОС – статус операционной системы (*установлен; отсутствует*)

**Правила:**

1) ЕСЛИ ОС = *не начинает загрузку*, ТО ПСЖ = *есть*

2) ЕСЛИ ОС = *начинает загрузку*, ТО ПСЖ = *нет*

3) ЕСЛИ ПК = *не запускается* И ПР = *нет проблем*, ТО ПСЖ = *есть*

4) ЕСЛИ ПСЖ = *нет* И СОС = *установлен* ТО В = *исправен*

5) ЕСЛИ ПСЖ = *есть* И ОЗУ = *исправен*, ТО В = *исправен*

6) ЕСЛИ ПСЖ = *есть* И ОЗУ = *исправен* И ВК = *не* *исправен*, ТО В = *не* *исправен*

7) ЕСЛИ ПСЖ = *есть* И ОЗУ = *исправен* И ВК = *исправен*, ТО В = *исправен*

8) ЕСЛИ ПСЖ = *есть* И ОЗУ = *исправен* И ВК = *исправен*, И ЦП = *не исправен* ТО В = *не исправен*

9) ЕСЛИ ПСЖ = *есть* И ОЗУ = *исправен* И ВК = *исправен*, И ЦП = *исправен* ТО В = *исправен*

10) ЕСЛИ ПСЖ = *нет* И ОС = *не начинает загрузку* И СЭ – *нет*, ТО СОС = *отсутствует*

11) ЕСЛИ ПСЖ = *есть* И ОС = *не начинает загрузку* И СЭ – *есть*, ТО СОС = *установлен*

12) ЕСЛИ ПСЖ = *есть* И ОЗУ = *исправен* И ВК = *исправен*, И ЦП = *исправен* ТО В = *исправен*

13) ЕСЛИ ПК = *не запускается* И ПР = *есть проблемы*, ОС = *не начинает загрузку*, ТО ПСЖ = *есть*

***Общая схема алгоритма обратной цепочки рассуждений:***

1. Определить переменную логического вывода.
2. Найти правило, заключительная часть которого содержит переменную вывода.
3. Если такое правило не найдено, то сообщить пользователю, что вывод невозможен, и перейти к шагу 14, иначе поместить правило в стек.
4. Если переменная, соответствующая номеру условия правила в вершине стека, определена, то увеличить номер условия на 1 и перейти к шагу 8.
5. Найти правило, в заключительной части которого встречается переменная, соответствующая номеру условия.
6. Если правило не найдено или предыдущий вывод неверен (см. шаг 9), то запросить значение переменной у пользователя, увеличить номер условия на 1 и перейти к шагу 8.
7. Поместить найденное правило в стек и вернуться к правилу 4.
8. Если номер условия меньше или равен числу фактов в условной части правила, то вернуться к шагу 4.
9. Если условная часть истинна, то присвоить значение переменной из заключительной части правила, иначе предыдущий вывод неверен.
10. Удалить правило из стека.
11. Если переменная вывода определена, то перейти к шагу 13.
12. Если стек пуст, то вернуться к шагу 2.
13. Сообщить пользователю окончательный вывод.
14. Конец алгоритма.

Алгоритм прекращает работу, если выполняется одно из двух условий:

1. значение переменной вывода определено
2. при полученных значениях переменных значение переменной вывода получить невозможно.

***Пошаговый вывод, полученный с помощью разработанной экспертной системы:***

Работа алгоритма начинается с задания пользователем переменной логического вывода в форме запроса:

***Диагноз?***

Следовательно, ЭС необходимо определить значение переменной СОС. В БЗ производится поиск правил, в заключительной части которых находится переменная СОС. Первое такое правило П10 помещается в стек правил:

|  |  |
| --- | --- |
| П1 | 1(ПСЖ) |

Определяем переменные из условной части правила П10. Первая переменная – ПСЖ (проблемы с железом). В БЗ производим поиск правил, в заключительной части которых встречается переменная ПСЖ. Таким правилом является правило П1. Помещаем его на вершину стека правил:

|  |  |
| --- | --- |
| П1 | 1(ОС) |
| П2 | 1(ПСЖ) |

Переменная ОС из условной части правила П1 не определена и не встречается в заключительных частях правил БЗ. Поэтому ЭС запрашивает значение переменной ОС у пользователя:

***Операционная система имеется?***

Допустим, пользователь ответил: *не начинает загрузку*. Условная часть правила П1 признается истинной, поэтому переменной ПСЖ присваивается значение *есть*, и правило П1 удаляется из стека.

***Имеем ОС = не начинает загрузку***

***ПСЖ = есть***

Условная часть П10 признается ложно, удаляется из стека и вместо него помещаем П11, продолжаем.

Номер условия правила П11 увеличиваем на 1, что соответствует переменной ОС, так как она у нас определена увеличиваем еще на 1, что соответствует переменной СЭ (синий экран).

|  |  |
| --- | --- |
| П11 | 3(СЭ) |

Переменная СЭ из условной части правила П11 не определена и не встречается в заключительных частях правил БЗ. Поэтому ЭС запрашивает значение переменной СЭ у пользователя:

***Синий экран есть?***

Допустим, пользователь ответил: *есть*. Переменной СЭ присваивается значение *есть.*

***Имеем ОС = не начинает загрузку***

***ПСЖ = есть***

***СЭ = есть***

Условная часть П11 истина, поэтому переменной СОС присваиваем значение *установлен.*

Значение переменной СОС определено, следовательно, обратная цепочка рассуждения закончена.

***Факты, которые определены пользователем:***

ОС (операционная система) = не начинает загрузку

СЭ (синий экран) = есть

***Факты, которые выведены из правил:***

ПСЖ (проблема с железом) = есть

СОС (статус операционной системы) = установлен