Национальный исследовательский
университет
“Московский энергетический институт”

Выпускная работа бакалавра

По направлению подготовки бакалавра
прикладной математики и информатики

Исследование и разработка механизма пересматриваемых рассуждений для систем поддержки принятия решений

Автор: студент группы А-13-08 Ганишев В. А.
Руководитель работы: д.т.н., проф. Вагин В.Н.

Москва, 2012
год

Цели и задачи

Целью данной дипломной работы является исследование и разработка механизма пересматриваемых рассуждений для систем поддержки принятия решений. Для достижений указанной цели необходимо решить следующие задачи.

- Исследование теории пересматриваемой аргументации, на базе которой будет построен механизм пересматриваемых рассуждений.
- Исследование и разработка алгоритма пересматриваемого вывода.
- Интеграция алгоритма вывода на основе прецедентов.
- Разработка программного комплекса.
- Тестирование разработанной программы.

Обоснование актуальности задачи

Задачи, стоящие перед реальными системами поддержки принятия решений, предполагают организацию правдоподобного вывода на наборах непостоянных или неполных данных. Средств классической логики высказываний для реализации данных систем недостаточно.

Механизм пересматриваемых рассуждений показал себя как мощный инструмент для реализации правдоподобного вывода и решает гораздо более широкий класс задач. Пересматриваемый вывод не является монотонным. Поэтому один довод не является подтверждением истинности предположения. А вот превосходство количество доводов «за» над доводами «против» позволяет судить об истинности предположения.

Теория аргументационных систем

Абстрактная аргументационная система – это тройка (L , R, $\preccurlyeq$), где L- язык(в данной работе используется язык логики высказываний), R – набор правил вывода, $\preccurlyeq$ – рефлексивный и транзитивный порядок аргументов.

Формулы языка называются *предложениями*.

Строгое правило вывода – это формула вида $\varphi_1, \dots, \varphi_n \rightarrow \varphi$, где $\varphi_1, \dots, \varphi_n$ – конечная, возможно пустая, последовательность предложений в языке и φ – предложение.

Пересматриваемое правило вывода – это формула вида $\varphi_1, \dots, \varphi_n \Rightarrow \varphi$, где $\varphi_1, \dots, \varphi_n$– конечная, возможно пустая, последовательность предложений в языке и φ – предложение.

Правила вывода в реализованной системе для удобства представлены не в виде “слева посылки, а справа заключение”, а “слева заключение, а справа посылки”.

Аргумент – это пара, состоящая из множества посылок и заключения. Посылки аргумента ,в свою очередь ,тоже могут являться аргументомами, что ведет к рекурсивному определению аргумента.

Рассматривается система аргументации, предложенная Ж. Вреесвийком.

Виды противоречий

В системе рассматривается 2 типа противоречий.

- *Опровержение* возникает, когда система принимает противоречащие заключения: P и ~P.
- *Подрывающий довод* оспаривает связь между посылками и заключениями. Если имеется довод, подрывающий связь P и Q, то он является причиной утверждать, что, если система поддерживает P, она не обязана поддерживать так же и Q.

Необходимо отметить, что данные противоречия не возможны в классическом монотонном выводе.

Механизм вывода на основе прецедентов.

Рассуждения на основе прецедентов (Case-based reasoning)являются процессом решения новых задач, основывающимся на решениях похожих задач в прошлом. Необходимо отметить, что они встречаются в повседневном человеческом поведении в виде “личного опыта”. Теория рассуждений на основе прецедентов является одной из наиболее глубоко изученных в когнитивной науке.

В данной системе прецедент – это последовательность посылок с решением, и в отличие от правила, в нем представлена часть рассуждений в цепи вывода, который уже был совершен.

Рассуждения на основе прецедентов носят в системе совещательный характер. Если степень достоверности прецедента выше, чем указанный пользователем порог достоверности, то система рекомендует выбрать это решение. Степень достоверности принимает значения от 0 до 1 и вычисляется следующим образом:

$$s(A,B) = \frac{2*|A \cap B|}{(|A|+|B|)} \quad (1)$$

s(A,B) – степень достоверности прецедента, A – множество фактов в системе, B – множество посылок в прецеденте, |A| - мощность множества A, $|A \cap B|$ - мощность пересечения множеств A и B.

Алгоритм вывода на основе прецедентов

procedure Debate

Входные параметры:

факт q – выбранный пользователем факт, о достоверности которого требуется принять решение.

ПОРОГ_ДОСТОВЕРНОСТИ – вещественное число, принимающее значение от 0 до 1, задается пользователем в программе.

Выходные параметры: выдается сообщение о возможности принять решение.

Переменные в процедуре:
ДОВОДЫ_ЗА и ДОВОДЫ_ПРОТИВ – целые числа.

Шаг 1. ДОВОДЫ_ЗА = 0.

Шаг 2. ДОВОДЫ_ПРОТИВ = 0.

Цикл, условие останова – в базе нет непроверенных прецедентов.

Шаг 3. Извлечь прецедент p из базы.

Шаг 4. Если решение p = q, то:

если ДОСТОВЕРНОСТЬ p > ПОРОГ_ДОСТОВЕРНОСТИ, то увеличить ДОВОДЫ_ЗА на единицу.

Шаг 5. Если решение p = ~q, то:

если достоверность p > ПОРОГ_ДОСТОВЕРНОСТИ, то увеличить ДОВОДЫ_ПРОТИВ на единицу.

Конец цикла.

Шаг 6. Если ДОВОДЫ_ЗА > ДОВОДЫ_ПРОТИВ, то выводится сообщение о том, что решение рекомендуется, иначе оно не рекомендуется.

Конец процедуры.

Procedure ДОСТОВЕРНОСТЬ p вычисляется по формуле (1).

Алгоритм пересматриваемого вывода

procedure Reasoning

Входные параметры: q – интересующий факт, о возможности принять который необходимо заключить.
Выходные параметры: сообщение о возможности принять факт.
Переменные в процедуре: ФЛАГ – целое, 0 или 1.

Шаг 1. Если q присутствует в базе фактов, то он уже принят и **конец процедуры**, иначе **Шаг 2**.
Шаг 2.Если ~q присутствует в системе, то он не может быть принят и **конец процедуры**, иначе **Шаг 3**.

Цикл, условие останова – в базе нет непроверенных правил.

Шаг 3. Извлечь правило p из базы

Шаг 4. Если заключение r = q или r = ~q, то **Шаг 5.**

Шаг 5. В цикле перебираются посылки правила p.

Шаг 6.Если посылка r присутствует в базе фактов, то ФЛАГ = ИСТИНА, иначе **Шаг 7.**

Шаг 7. Если ~r присутствует в базе фактов, то ФЛАГ = ЛОЖЬ, иначе **Шаг 8.**

Шаг 8. ***Reasoning*** r.
Шаг 9. Если r = ~q, то ФЛАГ = ~ФЛАГ.
Шаг 10. Если ФЛАГ = ИСТИНА, то r записывается в базу со статусом “не поражен”.

Конец цикла.

Шаг 11. ***Defeat.***

Шаг 12. Если q имеет статус “строгие”, то p принимается системой и **конец процедуры**, иначе **Шаг 12**

Шаг 13. Если ~q имеет статус “строгие”, то q не принимается системой и **конец процедуры**, иначе **Шаг 13.**

Шаг 14. Если КОЛИЧЕСТВО_ФАКТОВ q, имеющих статус “не поражен” > КОЛИЧЕСТВО_ФАКТОВ ~q, имеющих статус “поражен”, то p принимается системой, иначе p отвергается системой.

Конец процедуры.

КОЛИЧЕСТВО_ФАКТОВ вычисляется в таблице с помощью агрегатной функции COUNT(*)

Алгоритм процедуры поражения

procedure Defeat

Входные параметры: нет.

Выходные параметры: процедура меняет статусы аргументов в базе.

Цикл по всем фактам, принятым во время вывода.

Шаг 1. Построить множество фактов, поражающих p и имеющих статус “не поражен”.

Шаг 2. Проверить наличие самопоражения в базисе p, если таковое имеет место, то присвоить факту статус “поражен”.

Шаг 3. Проверить множество поражающих p фактов на множественное поражение. Если такое имеется, то присвоить временные статусы “пораженный” всем фактам.

Шаг 4. Иначе, если множество фактов, поражающих p не пусто, то присвоить p статус “поражен”, всем фактам, выводящимся по правилам, где p – посылка присвоить статус “поражен”.

Шаг 5. Если p имеет статус “не поражен” , правило его вывода – строгое, и все посылки этого правила находятся в начальной базе фактов или имеют статус “строгий”, то присвоить p статус “строгий”.

Конец цикла.
Выход из процедуры.

Программная реализация

Данный программный комплекс реализован в виде двух отдельных модулей (базы данных и приложения, работающего с ней). Данный подход выбран в силу того, что, во-первых, база данных является удобным и мощным транзакционным хранилищем. Во-вторых, для того, чтобы учесть возможность организации одновременной работы нескольких пользователей в будущем. База данных, содержащая все необходимые для работы данные не обязана находиться на том же компьютере, где запущена копия программы, а может располагаться на другом доступном компьютере в локальной сети. Связь приложения с базой данных выполняется с помощью инструментов библиотеки SMO языка C#.

На компьютере, где располагается база данных должна быть установлена копия MS SQL Server, и база данных должна быть подключена к нему. Компьютер, на котором запущен экземпляр программы должен работать под управлением операционной системы Windows XP или её более поздних версий. Все необходимые файлы библиотек находят

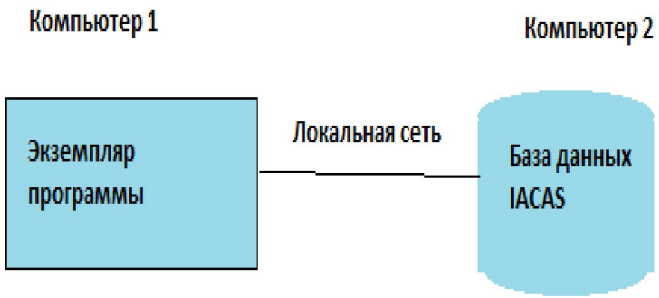

Рис. Пример организации работы приложения

Разработка базы данных

База данных приложения состоит из 9 таблиц, четыре из них доступны для редактирования пользователем. На них установлены правила целостности и триггеры для обеспечения целостности и корректности данных. Пять таблиц являются вспомогательными, они необходимы для работы приложения и недоступны для редактирования пользователем. Все таблицы базы данных находятся в третьей нормальной форме.

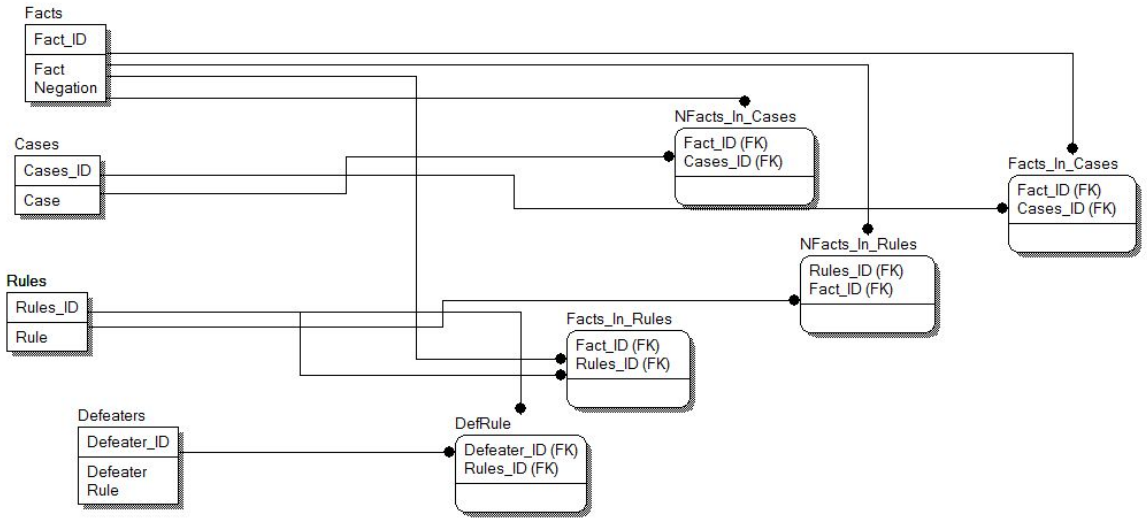

Рис. Модель базы данных приложения, созданная
с помощью CASE-средства ERWin DATA Modeler

Разработка программы

Программа написана на языке программирования C# с помощью среды разработчика MS Visual Studio 2010.

В программе реализована возможность поиска сервера баз данных на локальной машине или в локальной сети, а так же выбора базы данных для подключения. Подключение к базе данных реализовано с помощью средств SMO.

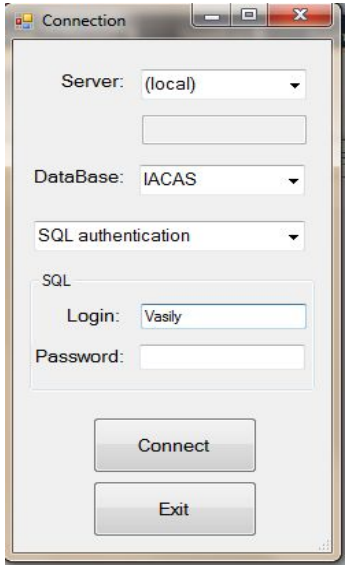

Рис. Окно подключения к базе данных

Пример 1

Ромб Никсона.

Интересующий нас факт – является ли Никсон пацифистом? Известно, что Никсон – квакер, а все квакеры – пацифисты. Но так же известно, что он республиканец, а среди республиканцев пацифистов нет.

Факты:
QUAKER
REPUBLICAN
Правила:
PACIFIST <= QUAKER
~PACIFIST <= REPUBLICAN

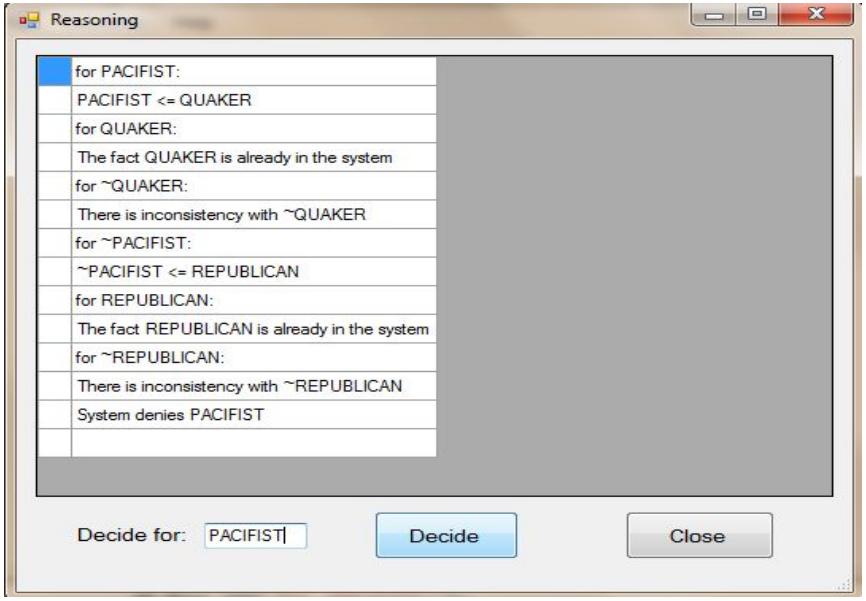

Рис. Работа программы на примере 1

Система заключила, что не может принять факт, что Никсон – пацифист. Но система также не подтвердит факт того, что Никсон не пацифист. В силу того, что имеется явный конфликт, и нет никакой дополнительной информации, система не может поддержать ни одно из решений.

Пример 2

Имеем следующую ситуацию:

Шумные автомобильные автомобильные двигатели обычно имеют высокие обороты.
Двигатели с высокими оборотами обычно малолитражные.
Шумные двигатели обычно большого размера.
Автомобили с большими колесами обычно устойчивы.
Устойчивость обычно означает изысканность.
Автомобили с большими колесами обычно не изысканные.
Все автомобили с большим двигателем и не изысканные – мощные.
Все автомобили с маленьким двигателем и изысканные – не мощные.
У Роберта шумная машина с большими колесами.
Интересующий нас факт: Машина Роберта мощная?

Факты:
NOISY_CAR
BIG_WHEELS

Правила:
HIGH_SPEED_ENGINE <= NOISY_CAR
~BIG_ENGINE <= HIGH_SPEED_ENGINE
BIG_ENGINE <= NOISY_CAR
STABILITY <= BIG_WHEELS
REFINEMENT <= STABILITY
~REFINEMENT <= BIG_WHEELS
POWERFUL_CAR <- BIG_ENGINE ~REFINEMENT
~POWERFUL_CAR <- ~BIG_ENGINE REFINEMENT

Поражающие доводы:
NOISY_CAR @ (REFINEMENT <= STABILITY)
BIG_WHEEL @ (~BIG_ENGINE <= HIGH_SPEED_ENGINE)

Работа системы:
for POWERFUL_CAR:
POWERFUL_CAR <- BIG_ENGINE ~REFINEMENT
for BIG_ENGINE:
BIG_ENGINE <= NOISY_CAR
for NOISY_CAR:
The fact NOISY_CAR is already in the system
for ~NOISY_CAR:
There is inconsistency with ~NOISY_CAR
for ~BIG_ENGINE:
~BIG_ENGINE <= HIGH_SPEED_ENGINE
for HIGH_SPEED_ENGINE:
HIGH_SPEED_ENGINE <= NOISY_CAR
for NOISY_CAR:
The fact NOISY_CAR is already in the system
for ~NOISY_CAR:
There is inconsistency with ~NOISY_CAR
for ~REFINEMENT:
~REFINEMENT <= BIG_WHEELS
for BIG_WHEELS:
The fact BIG_WHEELS is already in the system
for ~BIG_WHEELS:
There is inconsistency with ~BIG_WHEELS
for REFINEMENT:
REFINEMENT <= STABILITY
for STABILITY:
STABILITY <= BIG_WHEELS
for BIG_WHEELS:
The fact BIG_WHEELS is already in the system
for ~BIG_WHEELS:
There is inconsistency with ~BIG_WHEELS
for ~STABILITY:
System accepts POWERFUL_CAR

В результате система заключила, что машина Роберта мощная.

Пример 3

Пусть в системе имеются следующие факты:

a, ~b, d, ~e, h, I.

А также следующие прецеденты:

p -< a, c, d, g ,~k,
p -< ~a c f h I j,
p -< ~a c e f g h j,
~p -< a ~b d e ~h,
~p -< b ~d e g h I j,
~p -< ~d e f g h j ~k.

Порог достоверности положим равным 0.80.

Требуется заключить о возможности принять решение p.

Работа системы:
p -< a, c, d, g ,~k
denied
p -< ~a c f h I j
denied
p -< ~a c e f g h j
denied
~p -< a ~b d e ~h
established
~p -< b ~d e g h I j
established
~p -< ~d e f g h j ~k
established
as a result: DENIED

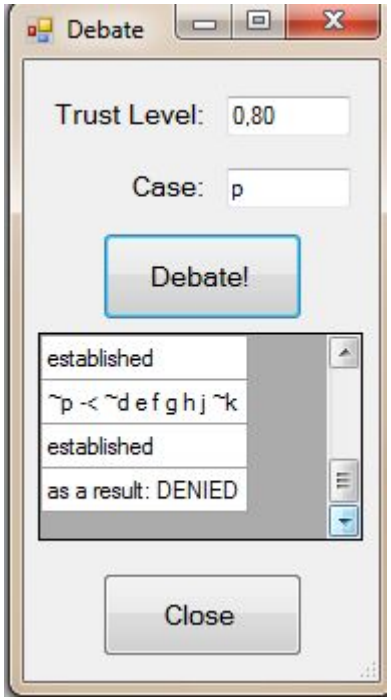

Рис. Пример работы программы на примере 3

В результате система заключила, что решение p не рекомендуется принимать.

Заключение

В результате выполнения работы было сделано следующее.

- ✔ Был проанализирован механизм пересматриваемых рассуждений.
- ✔ Разработаны и исследованы механизмы пересматриваемого вывода и вывода на основе прецедентов.
- ✔ Разработан программный комплекс, реализующий указанные выше типы вывода в виде двух отдельных модулей.
- ✔ Корректность работы системы проверена на ряде задач для систем автоматического доказательства теорем.
- ✔ Учтены возможности для организации многопользовательского доступа к системе в будущем.

Спасибо за внимание!