К.т.н. Т.В. Зайцева, к.с.н. С.В. Игрунова, Н.П. Путивцева, О.П. Пусная, М.Ю. Манзуланич (Белгородский ГУ)

T.V. Zajtseva, S.V. Igrunova, N.P. Putivtseva, O.P. Pusnaja, M.Ju. Manzulanich

КОМПЬЮТЕРНАЯ ТЕХНОЛОГИЯ ГЕНЕРАЦИИ ПРАВИЛ ДЛЯ ГИБРИДНЫХ ПРОДУКЦИОННО-ФРЕЙМОВЫХ ЭКСПЕРТНЫХ СИСТЕМ

COMPUTER TECHNOLOGY OF RULES' GENERATION FOR HYBRID PRODUCTS-FRAME EXPERT SYSTEMS

Представлена технология генерации правил для экспертной системы, которая является актуальной, т.к. позволяет решить проблему автоматизации выбора классов и признаков, а также реализовать процедуру автоматического составления правил и установления наибольшего соответствия для принадлежности к классу. Новизна подхода обусловлена применением гибридной системы, объединяющей продукционные и фреймовые виды представления знаний в базе знаний. Приведен конкретный пример реализации описанного подхода для выбора модема.

Keywords: hybrid expert system, base of knowledge, rule-products, frame representation of knowledge, generator of rules

Часто во многих сферах человеческой деятельности возникают такие задачи, для решения которых не существует строгих подходов или методов, их могут решить лишь эксперты, специализирующиеся в этих областях знаний. Это диагностика аппаратуры, планирование рабочей недели, прогнозирование операции, обучение работе на аппаратуре и др. Как правило, подобные задачи возникают в таких областях как диагностика, планирование, прогнозирование, обучение, моделирование, а также в таких сферах как медицина, юриспруденция, различные отрасли науки и техники, экономики, экологии, в военном деле и многих других [1,2].

Для решения такого рода задач существуют специальные комплексы программ - экспертные системы (ЭС). Создание и использование ЭС является одним из концептуальных этапов развития информационных технологий. В основе интеллектуального решения проблем в некоторой предметной области лежит принцип воспроиз-

ведения знаний опытных специалистов - экспертов. Они позволяют получать решения задач с нечеткой постановкой благодаря обращению к специальным базам знаний (БЗ), в которых содержатся сведения той области, к которой принадлежит решаемая задача. Решения ЭС находят сами. Также ЭС обычно позволяют экспертам создавать собственные БЗ, изменять или дополнять уже существующие и управлять ими. Как правило, все ЭС, существующие на рынке ПО очень дороги, при этом наиболее дорогостоящая часть - БЗ.

Ключевым понятием ЭС является БЗ. Традиционно выделяют [1] четыре вида представления знаний в БЗ: фреймовое, логическое, сетевое и продукционное. В настоящее время почти невозможно найти БЗ, в которой использовался только одна модель представления знаний. Все чаще при разработке ЭС используют гибридные модели. Наибольшей популярностью пользуются ЭС, основанные на продукционно-фреймовой модели представления знаний, в которой статические знания о предметной области представляются в виде фреймовой иерархии, а в качестве динамических знаний о переходах между состояниями используются продукционные правила прямого и обратного вывода, сгруппированные вокруг соответствующих фреймов и слотов [3].

Фрейм рассматривается как набор слотов, каждый из которых может содержать значение заранее определенного или произвольного типа. Основными типами являются скалярный, который подразделяется на числовой (целый или с плавающей точкой), логический и строковый подтипы, списковый (содержащий произвольное количество элементов любого типа, в т.ч. спискового) и ссылочный (содержащий ссылку на фрейм или слот).

Следует отметить, что выбранное фреймовое представление знаний универсально, т.к.:

1. Фреймовое представление знаний очень похоже на традиционные - объектный и компонентный подходы, что позволяет использовать фреймовую модель как некий "общий знаменатель" при создании гибридных систем, сочетающих декларативные знания и императивные компоненты. Таким образом объекты и компоненты открытых систем в форме Java-классов, (Enterprise) JavaBeans, CORBA- и COM-объектов могут быть представлены как фреймы в единой иерархии и, наоборот, любой фрейм иерархии может использоваться как объект, вызываемый из внешней программной

системы.

2. Фреймы могут эффективно использоваться для доступа к реляционным базам данных, а также к другим типам структуризованной информации (например, для анализа сетей веб-страниц и др.).

3. Фреймовое представление знаний предоставляет естественный способ кластеризации знаний, в особенности динамических правил прямого и обратного вывода, вокруг соответствующих фреймов в виде процедур-демонов и процедур-запросов, что, в свою очередь, обеспечивает естественное распределение знаний между различными узлами. Представление знаний в виде иерархических структур находит отражение в виде моделей онтологий и распределенной фреймовой иерархии как один из способов реализации таксономических онтологий, совмещенных с традиционной моделью логического вывода и представления знаний. Методология создания распределенных БЗ на основе таксономической онтологии заслуживает отдельного рассмотрения, в частности, с позиций адаптации существующих объектных методологий проектирования программных систем.

При разработке ЭС эксперт и инженер по знаниям сталкиваются со множеством проблем. К ним можно отнести выбор классов, признаков, но, пожалуй, самой трудной, а точнее трудоемкой, частью работы является составление правил и установление наибольшего соответствия тому или иному классу.

На данный момент не существует программных средств, способных в совершенстве решить существующую проблему. Разрабатываемая технология генераций правил помогает решить эти проблемы с помощью автоматического генерирования фреймов по заданным параметрам и признакам.

Целью является - повышение эффективности работы разработчиков ЭС за счет автоматизации процесса создания правил ЭС.

В вычислительной технике структура данных - это программная единица, позволяющая хранить и обрабатывать множество однотипных и/или логически связанных данных. Для добавления, поиска, изменения и удаления, данных структура данных предоставляет некоторый набор функций, составляющих интерфейс структуры данных. Структура данных часто является реализацией какого-либо абстрактного типа данных.

При разработке ПО большую роль играет проектирование хранилища данных и представление всех данных в виде множества связанных структур данных. Хорошо спроектированное хранилище данных оптимизирует использование ресурсов (таких как время выполнения операций, используемый объём оперативной памяти, число обращений к дисковым накопителям), требуемых для выполнения наиболее критичных операций. Экспертная система состоит из двух основных частей генератора правил и оболочки ЭС (рис. 1).

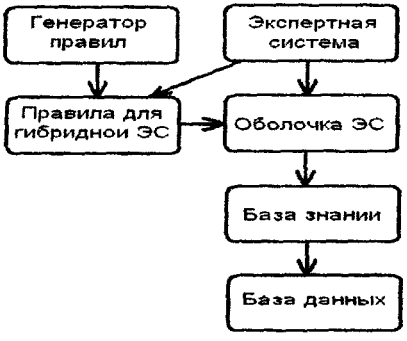

Рис.1
Схема работы ЭС

Генератор правил является приложением с открытым кодом, в котором будут существовать два хранилища данных, представленные матрицами. Первая матрица, размерностью A*B, где A - наибольшее число параметров одного признака, а B - количество признаков, заполняется параметрами, из которых в последствии формируются правила методом перебора.

Вторая матрица, размерностью (B+1)*C, где C - количество классов, заполняется классами с соответствующими им признаками.

Далее рассмотрены заполненные матрицы на примере ЭС выбора модема. На рис. 2. приведен пример заполненной матрицы параметров.

```
AnsiString mass[8][4]={
      {"ADSL", "VDSL", "SDSL", "SHDSL"},
      {"Да", "Нет", "0", "0"},
      {"Да", "Нет", "0", "0"},
      {"Да", "Нет", "0", "0"},
      {"Да", "Нет", "0", "0"},
      {"Да", "Нет", "0", "0"},
      {"Да", "Нет", "0", "0"},
      {"Да", "Нет", "0", "0"},

};
```

Рис. 2
Матрица параметров

На рис. 3. представлен пример заполненной матрицы классов.

```
int kol=15;
Ansistring modem[15][9]={
    {"Dynamix UM-S", "SHDSL", "Да", "Да", "Да", "Да", "Да", "Нет", "Да"},
    {"NSGate qBRIDGE-105", "SDSL", "Нет", "Нет", "Нет", "Нет", "Нет", "Нет", "Нет"},
    {"Zyxel P-660RU EE", "ADSL", "Да", "Нет", "Да", "Нет", "Да", "Нет", "Да"},
    {"NetGear DG834", "ADSL", "Да", "Да", "Да", "Да", "Да", "Нет", "Да"},
    {"TRENDnet TW100-BRM504", "ADSL", "Да", "Да", "Да", "Да", "Да", "Нет", "Нет"},
    {"Planet GRT-101", "SHDSL", "Да", "Нет", "Да", "Нет", "Да", "Нет", "Да"},
    {"D-link DSL-2520U", "ADSL", "Да", "Нет", "Да", "Да", "Да", "Нет", "Да"},
    {"ASUS AM604", "ADSL", "Да", "Да", "Да", "Нет", "Да", "Нет", "Да"},
    {"Planet GRT-501", "SHDSL", "Да", "Нет", "Да", "Да", "Да", "Нет", "Да"},
    {"ADC PairGain Megabit Modem 300S", "SDSL", "Нет", "Нет", "Нет", "Нет", "Да", "Нет", "Нет"},
    {"Aviv SDSL-165", "SDSL", "Нет", "Нет", "Нет", "Нет", "Нет", "Нет", "Нет"},
    {"Planet VC-200M", "VDSL", "Да", "Да", "Нет", "Нет", "Да", "Нет", "Да"},
    {"Zyxel Prestige 841 EE", "VDSL", "Нет", "Нет", "Нет", "Нет", "Нет", "Да", "Нет"},
    {"D-link DEV-301M", "VDSL", "Нет", "Нет", "Нет", "Нет", "Нет", "Да", "Нет"},
    {"Planet VC-1025", "VDSL", "Да", "Нет", "Нет", "Нет", "Нет", "Да", "Да"},

};
```

Рис. 3
Матрица классов

Генерация правил в программе производится путем полного перебора всех возможных вариантов. Из матрицы параметров выбирается сочетание параметров таким образом, что сочетания не повторяются. После формирования параметров и получения класса он сопоставляется с данными, внесенными в матрицу классов. Затем подбирается наиболее подходящий и, для него, рассчитывается коэффициент доверия. Следующим шагом создается правило в заданном виде, со всеми подобранными параметрами, при этом в правиле формулируется несколько возможных вариантов выбора класса.

Далее приведен алгоритм генерации правил, рассмотренный на примере выбора модема.

Первым шагом из матрицы параметров, по каждому из пунктов, выбирается первое значение (рис. 4).

Далее выбранное сочетание значений матрицы: «ADSL, Да, Да, Да, Да, Да, Да, Да», сопоставляется с матрицей приведенной на рис. 5. Как видно, выбранным параметрам соответствует четвертое значение NetGear DG834.

```
AnsiString mass[8][4]={
        {"ADSL", "VDSL", "SDSL", "SHDSL"},
        {"Да", "Нет", "0", "0"},
        {"Да", "Нет", "0", "0"},
        {"Да", "Нет", "0", "0"},
        {"Да", "Нет", "0", "0"},
        {"Да", "Нет", "0", "0"},
        {"Да", "Нет", "0", "0"},
        {"Да", "Нет", "0", "0"},
};
```

Рис. 4
Первый выбор значений из матрицы

```
int kol=15;
AnsiString modem[15][9]={
    {"Dynamix UM-S", "SHDSL", "Да", "Да", "Да", "Да", "Да", "Нет", "Да"},
    {"NSGate qBRIDGE-105", "SDSL", "Нет", "Нет", "Нет", "Нет", "Нет", "Нет", "Нет"},
    {"Zyxel P-660RU EE", "ADSL", "Да", "Нет", "Да", "Нет", "Да", "Нет", "Да"},
    {"NetGear DG834", "ADSL", "Да", "Да", "Да", "Да", "Да", "Нет", "Да"},
    {"TRENDNet TW100-BRM504", "ADSL", "Да", "Да", "Да", "Да", "Да", "Нет", "Нет"},
    {"Planet GRT-101", "SHDSL", "Да", "Нет", "Да", "Нет", "Да", "Нет", "Да"},
    {"D-link DSL-2520U", "ADSL", "Да", "Нет", "Да", "Да", "Да", "Нет", "Да"},
    {"ASUS AM604", "ADSL", "Да", "Да", "Да", "Нет", "Да", "Нет", "Да"},
    {"Planet GRT-501", "SHDSL", "Да", "Нет", "Да", "Да", "Да", "Нет", "Да"},
    {"ADC PairGain Megabit Modem 300S", "SDSL", "Нет", "Нет", "Нет", "Нет", "Да", "Нет", "Нет"},
    {"Aviv SDSL-165", "SDSL", "Нет", "Нет", "Нет", "Нет", "Нет", "Нет", "Нет"},
    {"Planet VC-200M", "VDSL", "Да", "Да", "Нет", "Нет", "Да", "Нет", "Да"},
    {"Zyxel Prestige 841 EE", "VDSL", "Нет", "Нет", "Нет", "Нет", "Нет", "Да", "Нет"},
    {"D-link DEV-301M", "VDSL", "Нет", "Нет", "Нет", "Нет", "Нет", "Да", "Нет"},
    {"Planet VC-102S", "VDSL", "Да", "Нет", "Нет", "Нет", "Нет", "Да", "Да"},

};
```

Рис. 5
Соответствие значения

Затем в соответствии с правилами построения модели формируется правило.

Из полей Label, расположенных непосредственно на основной форме, берутся статические данные, общие для каждого правила (рис. 6).

```
Rule

=(Исходные данные Тип модема,
=(Исходные данные Наличие маршрутизатора,
=(Исходные данные Наличие коммутатора,
=(Исходные данные Наличие DHCP-сервера,
=(Исходные данные Поддержка VPN,
=(Исходные данные Поддержка WEB-интерфейса,
=(Исходные данные Наличие встроенного сплиттера,
=(Исходные данные Адаптация для России,

Do
=(Цель Выбрать модем, Под ваш выбор подходит
EndR
```

Рис. 6
Статическая информация для составления правил

```
Rule 1
=(Исходные данные.Тип модема; ADSL)
=(Исходные данные.Наличие маршрутизатора; Да)
=(Исходные данные.Наличие коммутатора; Да)
=(Исходные данные.Наличие DHCP-сервера; Да)
=(Исходные данные.Поддержка VPN; Да)
=(Исходные данные.Поддержка WEB-интерфейса; Да)
=(Исходные данные.Наличие встроенного сплиттера; Да)
=(Исходные данные.Адаптация для России; Да)
Do
=(Цель.Выбрать модем; Под ваш выбор подходит NetGear DG834)
EndR
```

Рис. 7
Сформированное правило 1

Сгенерированное правило заносится в файл rule.txt. После чего начинается генерация второго правила.

Для создания второго правила матрицы параметров, по каждому из пунктов, за исключением последнего, выбирается первое значение, для последнего - выбирается второе значение (рис. 8).

```
{"ADSL", "VDSL", "SDSL", "SHDSL"},
{"Да", "Нет", "С", "С"},
{"Да", "Нет", "С", "С"},
{"Да", "Нет", "С", "С"},
{"Да", "Нет", "С", "С"},
{"Да", "Нет", "С", "С"},
{"Да", "Нет", "С", "С"},
{"Да", "Нет", "С", "С"},
```

Рис 8
Второй выбор значений из матрицы

Далее из матрицы классов выбирается соответствующее значение и генерируется правило 2 (рис. 9).

```
Rule 2
=(Исходные данные.Тип модема; ADSL)
=(Исходные данные.Наличие маршрутизатора; Да)
=(Исходные данные.Наличие коммутатора; Да)
=(Исходные данные.Наличие DHCP-сервера; Да)
=(Исходные данные.Поддержка VPN; Да)
=(Исходные данные.Поддержка WEB-интерфейса; Да)
=(Исходные данные.Наличие встроенного сплиттера; Да)
=(Исходные данные.Адаптация для России; Нет)
Do
=(Цель.Выбрать модем; Под ваш выбор подходит TRENDnet TW100-BRM504)
EndR
```

Рис 9
Сформированное правило 2

Затем все правила формируются аналогичным образом - путем полного перебора.

Ниже в блок-схеме приведена технология генерации правил (рис. 10).

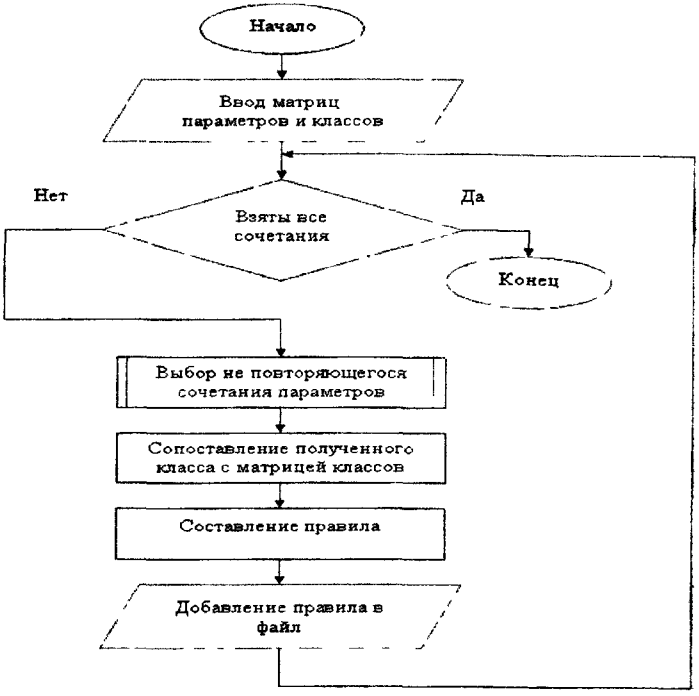

Рис. 10
Алгоритм генерации

При использовании данной технологии скорость разработки правил возрастает в разы (табл. 1).

Таблица 1

Скорость набора правил

Количество правил	Ручной набор правил	Генерация
До 20	15-20 мин	15-20 мин
20-100	0.9-1 мин/правило	0.3-0.35 мин/правило
100-500	1-1.2 мин/правило	0.15-0.2 мин/правило
500-1000	1.2-1.5 мин/правило	0.1-0.05 мин/правило
>1000	Проблематично	0.05-0.02 мин/правило

Из вышесказанного можно сделать следующие выводы:

1. Предлагаемая технология генерации сокращает скорость разработки правил;

2. При использовании генератора формируются правила для любого возможного сочетания параметров, при этом, если в матрице классов не находится однозначно определяемый класс, подбирается оптимальный по количеству схожих признаков,

3. В генератор легко добавляются как новые признаки, так и новые объекты класса или классы, переформирование правил занимает не более 20-30 мин.;

4 Программа позволяет формировать правила не только для конкретной гибридной ЭС, но и для любой ЭС, основанной на продукционно-фреймовом представлении знаний, например, GURU;

5. В генератор правил можно вводить дополнительные параметры, например, устанавливать приоритет одного из признаков или учитывать только те признаки, которые принимают определенное значение.

Литература

1. Гаврилова Т.А. и Хорошевский В.Ф. Базы знаний интеллектуальных систем. СПб., Питер, 2000.

2. Жиляков Е.Г. и Барсук А.А. О компьютерной реализации автоматической вариационной классификации объектов на спутниковых фотографиях земной поверхности. – "Вопросы радиоэлектроники", сер. ЭВТ, 2010, вып.1, с. 166-177.

3. Сошников Д.В. Построение распределенных интеллектуальных систем на основе распределенной фреймовой иерархии. – В сб.: Тезисы докл. Междунар. науч.-практич. конф. Информационные технологии в образовании. Шахты., 2001:

Статья поступила 12 10 2010