Курс лекций «Введение в ИИ.». Часть I. От психофизиологической проблемы до экспертных систем. Лекция 10. Экспертные системы. Часть III.

О.Г. Чанышев

Содержание

1 Введение: ЭС реального времени

Среди ЭС наиболее значимы экспертные системы реального времени, (ЭС РВ) или динамические экспертные системы. На их долю приходится 70 процентов рынка систем, основаных на знаниях. Значимость ЭС РВ определяется, в первую очередь, тем, что только с помощью подобных средств создаются стратегически значимые приложения. Классы задач, решаемых экспертными системами реального времени, таковы:

- мониторинг,
- управление верхнего уровня,
- обнаружение неисправностей,

- диагностика,
- составление расписаний,
- планирование,
- оптимизация,
- советчики оператора,
- проектирование.

Требования, предъявляемые к ЭС РВ:

- Представлять изменяющиеся во времени данные, поступающие от внешних источников, обеспечивать их хранение и анализ.
- Планировать в соответствии с приоритетами обработку поступивших в систему процессов.
- Обеспечивать механизм рассуждения при ограниченных ресурсах (время, память). Реализация этого механизма предъявляет требования к высокой скорости работы системы, способности одновременно решать несколько задач (т.е. операционные системы UNIX, VMS, Windows NT, но не MS-DOS).
- Обеспечивать "предсказуемость"поведения системы, т.е. гарантию того, что каждая задача будет запущена и завершена в строгом соответствии с временными ограничениями.
- Моделировать "окружающий мир", рассматриваемый в данном приложении.
- Протоколировать свои действия и действия персонала, обеспечивать восстановление после сбоя.
- Обеспечивать наполнение базы знаний для приложений с минимальными затратами времени и труда.
- Обеспечивать настройку системы на решаемые задачи (проблемная/предметная ориентированность).
- Обеспечивать создание и поддержку пользовательских интерфейсов для различных категорий пользователей.
- Обеспечивать уровень защиты информации (по категориям пользователей) и предотвращать несанкционированный доступ.

Инструментарий для создания экспертных систем реального времени впервые выпустила фирма Lisp Machine Inc в 1985 году. Этот продукт предназначался для символьных ЭВМ Symbolics и носил название Picon. Его успех привел к тому, что группа ведущих его разработчиков образовала фирму Gensym, которая, значительно развив идеи, заложенные в Picon, выпустила в 1988 году инструментальное средство под названием G2. В настоящий момент работает его третья версия и подготовлена четвертая [1,7]. С отставанием от Gensym на два-три года ряд других фирм начал создавать (или пытаться создавать) свои инструментальные средства. Назовем ряд из них: RT Works (фирма Talarian, США), COMDALE/C (Comdale Techn.,

Канада), COGSYS (SC, США), ILOG Rules (ILOG, Франция). Сравнение двух наиболее продвинутых систем, G2 и RT Works, которое проводилось путем разработки одного и того же приложения двумя организациями, NASA (США) и Storm Integration (США) [10], показало значительное превосходство первой.

2 Архитектура экспертной системы реального времени

Специфические требования, предъявляемые к экспертной системе реального времени, приводят к тому, что их архитектура отличается от архитектуры статических систем. Не вдаваясь в детали, отметим появление двух новых подсистем - моделирования внешнего окружения и сопряжения с внешним миром (датчиками, контроллерами, СУБД и т.п.) - и значительные изменения, которым подвергаются оставшиеся подсистемы. Для того, чтобы понять, что представляет из себя среда для создания экспертных систем реального времени, опишем ниже жизненный цикл такой системы, а также ее основные компоненты. Описание оболочки экспертной системы реального времени приведем на примере средства G2, поскольку в нем полностью реализованы возможности, которые считаются необходимыми и уместными в подобных программных продуктах.

2.1 Жизненный цикл приложения

Жизненный цикл приложения в G2 состоит из ряда этапов.

2.1.1 Разработка прототипа приложения

Разработчиком обычно является специалист в конкретной области знаний. Он в ходе обсуждений с конечным пользователем определяет функции, выполняемые прототипом. При разработке прототипа не используется традиционное программирование (оболочка!). Создание прототипа обычно занимает от одной до двух недель (при наличии у разработчика опыта по созданию приложений в данной среде. Прототип, как и приложение, создается на структурированном подмножестве естественного языка, с использованием объектной графики, иерархии классов объектов, правил, динамических моделей внешнего мира. Многословность языка сведена к минимуму путем введения операции клонирования, позволяющей размножить любую сущность базы знаний.

2.1.2 Расширение прототипа до приложения

Конечный пользователь предлагает этапность проведения работ, направления развития базы знаний, указывает пропуски в ней. Разработчик может расширять и модифицировать базу знаний в присутствии пользователя даже в тот момент, когда приложение исполняется. В ходе этой работы прототип развивается до такого состояния, что начинает удовлетворять представлениям конечного пользователя. В крупных приложениях команда разработчиков может разбить приложение на отдельные модули, которые интегрируются в единую базу знаний.

Возможен и альтернативный подход к созданию приложения. При этом подходе каждый разработчик имеет доступ к базе знаний, находящейся на сервере, при помощи средства, называемого Telewindows, обычно расположенного на компьютере-клиенте. В этом случае разработчики могут иметь различные авторизованные уровни доступа к приложению. Приложение

может быть реализовано не только на различных ЭВМ, но и с использованием нескольких взаимодействующих оболочек G2.

2.1.3 Тестирование приложения на наличие ошибок

В G2 ошибки в синтаксисе показываются непосредственно при вводе конструкций (структур данных и исполняемых утверждений) в базу данных. Вводятся только синтаксически правильные конструкции. Разработчик освобожден и от необходимости знать детальный синтаксис языка G2, так как при вводе в базу знаний некоторой конструкции ему в виде подсказки сообщается перечень всех возможных синтаксически правильных продолжений.

Для выявления ошибок и неопределенностей реализована возможность "Inspect", позволяющая просматривать различные аспекты базы знаний, например, "показать все утверждения со ссылками на неопределенные сущности (объекты, связи, атрибуты)", "показать графически иерархию заданного класса объектов", "показать все сущности, у которых значение атрибута Notes не OK". (Данный атрибут есть у всех сущностей, представимых в языке G2; его значение - либо OK, когда нет претензий к сущности, либо описание реальных или потенциальных проблем, например, ссылка на несуществующий объект, несколько объектов с одним именем и т.п.)

2.1.4 Тестирование логики приложения и ограничений

Блок динамического моделирования позволяет при тестировании воссоздать различные ситуации, адекватные внешнему миру. Конечный пользователь может принять непосредственное участие в тестировании благодаря управлению цветом (т.е. изменение цвета при наступлении заданного состояния или выполнения условия) и анимации (т.е. перемещение/вращение сущности при наступлении состояния/условия). Благодаря этому он сможет понять и оценить логику работы приложения, не анализируя правила и процедуры, а рассматривая графическое изображение управляемого процесса, технического сооружения и т.п. Для проверки выполнения ограничений используется возможность "Meters", вычисляющая статистику по производительности и используемой памяти.

Полученное приложение полностью переносимо на различные платформы в среду UNIX (SUN, DEC, HP, IBM и т.д.), VMS (DEC VAX) и Windows NT (Intel, DEC Alpha). База знаний сохраняется в обычном ASCII-файле, который однозначно интерпретируется на любой из поддерживаемых платформ. Перенос приложения не требует его перекомпиляции и заключается в простом перемещении файлов. Функциональные возможности и внешний вид приложения не претерпевают при этом никаких изменений. Приложение может работать как в "полной"(т.е. предназначенной для разработки) среде, так и под runtime, которая не позволяет модифицировать базу знаний.

2.1.5 Сопровождение приложения

Не только сам разработчик данного приложения, но и любой пользователь может легко его понять и сопровождать, так как все объекты/классы, правила, процедуры, функции, формулы, модели хранятся в базе знаний в виде структурированного естественного языка и в виде графических объектов. Для ее просмотра используется возможность "Inspect". Сопровождение упрощается за счет того, что различным группам пользователей выдается не вся информация, а только ее часть, соответствующая их потребностям.

2.2 Основные компоненты

Экспертная система реального времени состоит из базы знаний, машины вывода, подсистемы моделирования и планировщика.

2.2.1 Базы знаний

Все знания в G2 хранятся в двух типах файлов: базы знаний и библиотеки знаний. В файлах первого типа хранятся знания о приложениях: определения всех объектов, объекты, правила, процедуры и т.п. В файлах библиотек хранятся общие знания, которые могут быть использованы более, чем в одном приложении, например, определение стандартных объектов. Файлы баз знаний могут преобразоваться в библиотеки знаний и обратно. В целях обеспечения повторной используемости приложений реализовано средство, позволяющее объединять с текущим приложением ранее созданные базы и библиотеки знаний. При этом конфликты в объединяемых знаниях выявляются и отображаются на дисплее. Знания структурируются: предусмотрены иерархия классов, иерархия модулей, иерархия рабочих пространств. Каждую из них можно показать на дисплее.

2.2.2 Сущности и иерархия классов

В системе G2 последовательно реализована объектно-ориентированная технология. Структуры данных представляются в виде классов объектов. Классы наследуют атрибуты от суперклассов и передают свои атрибуты подклассам. Каждый класс (исключая корневой) может иметь конкретные экземпляры класса.

Все, что хранится в базе знаний и с чем оперирует система, является экземпляром того или иного класса. Более того, все синтаксические конструкции G2 являются классами. Для сохранения общности даже базовые типы данных - символьные, числовые, булевы и истинностные значения нечеткой логики - представлены соответствующими классами. Описание класса включает ссылку на суперкласс и содержит перечень атрибутов, специфичных для класса.

2.2.3 Иерархия модулей и рабочих пространств

Для структуризации G2-приложений используются "модули"и "рабочие пространства". Несмотря на схожесть функций этих конструкций, между ними есть существенные различия.

Различие между "модулями"и "рабочими пространствами"состоит в следующем. Модули разделяют приложение на отдельные базы знаний, совместно используемые в различных приложениях. Они полезны в процессе разработки приложения, а не его исполнения. Рабочие пространства, наоборот, выполняют свою роль при исполнении приложения. Они содержат в себе различные сущности и обеспечивают разбиение приложения на небольшие части, которые легче понять и обрабатывать.

Рабочие пространства можно устанавливать (вручную или действием в правиле/процедуре) в активное или неактивное состояние (при этом сущности, находящиеся в этом пространстве и в его подпространствах, становятся невидимыми для механизма вывода). Данный механизм используется, например, при наличии альтернативных групп правил, когда активной должна быть только одна из них.

Кроме того, рабочие пространства используются для определения пользовательских ограничений, определяющих разное поведение приложения для различных категорий пользователей.

2.2.4 Структуры данных

Сущности в базе знаний представлены как пассивными структурами, так и активными (процедурами).

Примерами первых являются объекты и их классы, связи (connection), отношения (relation), переменные, параметры, списки, массивы, рабочие пространства.

Примерами вторых - правила, процедуры, формулы, функции. Выделяют объекты (и классы), встроенные в систему и вводимые пользователем. При разработке приложения, как правило, создаются подклассы, отражающие специфику данного приложения. Среди встроенных подклассов объектов наибольший интерес представляет подкласс данных, включающий подклассы переменных, параметров, списков и массивов. Особая роль отводится переменным. В отличие от статических систем переменные делятся на три вида: собственно переменные, параметры и простые атрибуты. Параметры получают значения в результате работы машины вывода или выполнения какой-либо процедуры. Переменные представляют измеряемые характеристики объектов реального мира и поэтому имеют специфические черты: время жизни значения и источник данных. Время жизни значения переменной определяет промежуток времени, в течение которого это значение актуально, по истечении этого промежутка переменная считается не имеющей значения.

Выполняемые утверждения

Основу выполняемых утверждений баз знаний составляют правила и процедуры. Кроме того, есть формулы, функции, действия и т.п. Правила в G2 имеют традиционный вид: левая часть (антецедент) и правая часть (консеквент). Кроме if-правила, используется еще четыре типа правил: initially, unconditionally, when и whenever. Каждое из типов правил может быть как общим, т.е. относящимся ко всему классу, так и специфическим, относящимся к конкретным экземплярам класса.

Несмотря на то что продукционные правила обеспечивают достаточную гибкость для описания реакций системы на изменения окружающего мира, в некоторых случаях, когда необходимо выполнить жесткую последовательность действий, например, запуск или остановку комплекса оборудования, более предпочтителен процедурный подход. Язык программирования, используемый в G2 для представления процедурных знаний, является достаточно близким родственником Паскаля. Кроме стандартных управляющих конструкций язык расширен элементами, учитывающими работу процедуры в реальном времени: ожидание наступления событий, разрешение другим задачам прерывать ее выполнение, директивы, задающие последовательное или параллельное выполнение операторов. Еще одна интересная особенность языка - итераторы, позволяющие организовать цикл над множеством экземпляров класса.

2.2.5 Машина вывода, подсистема моделирования и планировщики

С точки зрения ЭС РВ, полный перебор возможных к применению правил - непозволительная роскошь. В связи с тем, что G2 ориентирована на приложения, работающие в реальном времени, в машине вывода должны быть средства для сокращения перебора, для реакции на непредвиденные события и т.п. Для машины вывода G2 характерен богатый набор способов возбуждения правил. Предусмотрено девять случаев:

1. Данные, входящие в антецедент правила изменились (прямой вывод - forward chaining).
2. Правило определяет значение переменной, которое требуется другому правилу или процедуре (обратный вывод - backward chaining).

3. Каждые n секунд, где n - число, определенное для данного правила (сканирование - scan).
4. Явное или неявное возбуждение другим правилом - путем применения действий фокусирования и возбуждения (focus и invoke).
5. Каждый раз при запуске приложения.
6. Входящей в антецедент переменной присвоено значение, независимо от того, изменилось оно или нет.
7. Определенный объект на экране перемещен пользователем или другим правилом.
8. Определенное отношение между объектами установлено или уничтожено.
9. Переменная не получила значения в результате обращения к своему источнику данных.

Если первые два способа достаточно распространены и в статических системах, а третий хорошо известен как механизм запуска процедур-демонов, то остальные являются уникальной особенностью системы G2. В связи с тем, что G2-приложение управляет множеством одновременно исполняемых задач, необходим планировщик. Хотя пользователь никогда не взаимодействует с ним, планировщик контролирует всю активность, видимую пользователем, и активность фоновых задач. Планировщик определяет порядок обработки задач, взаимодействует с источниками данных и пользователями, запускает процессы и осуществляет коммуникацию с другими процессами в сети.

Подсистема моделировоания

Подсистема моделирования G2 - достаточно автономная, но важная часть системы. На различных этапах жизненного цикла прикладной системы она служит достижению различных целей. Во время разработки подсистема моделирования используется вместо объектов реального мира для имитации показаний датчиков.

На этапе эксплуатации прикладной системы процедуры моделирования выполняются параллельно функциям мониторинга и управления процессом, что обеспечивает следующие возможности: - верификация показаний датчиков во время исполнения приложения; - подстановка модельных значений переменных при невозможности получения реальных значений (выход из строя датчика или длительное время реакции на запрос).

Как видим, играя роль самостоятельного агента знаний, подсистема моделирования повышает жизнеспособность и надежность приложений. Для описания внешнего мира подсистема моделирования использует уравнения трех видов: алгебраические, разностные и дифференциальные (первого порядка).

3 Заключение

Одним из основных направлений в этой области инеллектуальных систем являются экспертные системы реального времени.

G2 представляет самодостаточную среду для разработки, внедрения и сопровождения приложений в широком диапазоне отраслей.

G2 объединяет в себе как универсальные технологии построения современных информационных систем

- стандарты открытых систем,

- архитектура клиент/сервер,
- объектно-ориентированное программирование,
- использование ОС, обеспечивающих параллельное выполнение в реальном времени многих независимых процессов, так и специализированные методы рассуждения, основанные на правилах
- рассуждения, основанные на динамических моделях, или имитационное моделирование,
- процедурные рассуждения,
- активная объектная графика,
- структурированный естественный язык для представления базы знаний,

а также интегрирует технологии систем, основанных на знаниях с технологией традиционного программирования (с пакетами программ, с СУБД, с контроллерами и концентраторами данных и т.д.).

Все это позволяет с помощью данной оболочки создавать практически любые большие приложения значительно быстрее, чем с использованием традиционных методов программирования, и снизить трудозатраты на сопровождение готовых приложений и их перенос на другие платформы.

Список литературы

[1] F. Hayes-Roth, N. Jacobstein. The State of Enowledge-Based Systems. Communications of the ACM, March, 1994, v.37, n.3, pp.27-39.

[2] P. Harmon. The Size of the Commercial AI Market in the US. Intelligent Software Strategies. 1994, v.10, n.1, pp. 1-6.

[3] Expert system saves 20 million L on pipeline management. C&I July, 1994, p.31.

[4] P. Harmon. The Market for Intelligent Software Products. Intelligent Sopware Strategies 1992, v.8, n.2, pp.5-12.

[5] D.R Perley. Migrating to Open Systems: Taming he Tiger. McGraw-Hill, 1993, p.252.

[6] P. Harmon. The AI Tools Market The Market for Intelligent Software Building Tools. Part I. Intelligent Softwane Strategies, 1994, v 10, n.2, pp.1-14.

[7] P. Harmon. The market for intelligent software pnducts Intelligent Software Strategies, 1992, v.8, n.2, pp.5-12.

[8] B.R. Clements and F. Preto. Evaluating Commencial Real Time Expert System Software for Use in the Process Industries. C&I, 1993, pp. 107-114.

[9] B. Moore et al. Questions and Answers about G2. 1993. Gensym Corporation. pp.26-28.

[10] B. Moore. Memorandum. 1993, April. Gensym Corparation.

[11] Р. Богатырев. "Этот странный придуманный мир". Компьютерра. С.30-33. 1996 год.

Следующая лекция

Лекция 11. Введение в распознавание образов