Тема3.

Лекция 5: Продукционные и логические модели представления знаний.

1. Продукционные модели.

Продукционные модели в последнее время широко используются в системах представления знаний. Первоначально предложенные Постом в 1943 г. [118], они были впервые применены в системах ИИ в 1972 г. [116].

Продукционные модели могут быть реализованы как процедурно, так и декларативно. Их простота и строгая форма послужили основой ряда интересных свойств, что сделало их удобным средством представления знаний. Рядом исследователей отмечалось, что использование продукционных моделей имеет уже само по себе особую психологическую важность, хотя они могут быть с успехом использованы и вне рамок психологического моделирования.

Продукционные модели — это набор, правил вида «условия — действие», где условиями являются утверждения о содержимом некой базы данных, а действия представляют собой процедуры, которые могут изменять содержимое БД.

В продукционных системах можно выделить три основные компоненты:

1. Неструктурированная или структурированная БД.

2. Некоторое число продукционных правил или просто продукций. Каждая продукция состоит из двух частей:

- условий (антецендент); в этой части определяются некоторые условия, которые должны выполняться в БД для того, чтобы были выполнены соответствующие действия;
- действий (консеквент); эта часть содержит описание действий, которые должны быть совершены над БД в случае выполнения соответствующих условий. В простейших продукционных системах они только определяют, какие элементы следует добавить (или иногда удалить) в БД.

3. Интерпретатор, который последовательно определяет, какие продукции могут быть активированы в зависимости от условий, в них содержащихся; выбирает одно из применимых в данной ситуации правил продукций; выполняет действие из выбранной процедуры.

Продукционные модели в основном находят применение в качестве решателей или механизмов выводов.

В БД системы хранятся известные факты о некоторой предметной области. Продукции содержат специфические для данной области знания о том, какие

дополнительные факты могут быть допущены, если специфические данные найдены в БД.

Действия продукций могут состоять из активных процедур, которые автоматически производят необходимые операции над содержимым БД (либо подобно «демонам» проверять самих себя на предмет того, выполняются ли их условия активации). В этом случае форма представления знаний является процедурной, хотя и в весьма ограниченном виде. В последующих итерациях факты, добавленные в БД, могут подключать (активировать) другие продукции и т. д.

В классических продукционных системах БД представляют собой переменную часть системы, в то время как правила и интерпретатор чаще всего не меняются. Будучи реализованы процедурно, классические продукционные модели обладают весьма привлекательным свойством модульности. Поэтому правила могут быть добавлены .или удалены без возникновения неожиданных побочных эффектов. Причина этого заключается также в том, что в классических системах вызов процедур осуществляется только в зависимости от состояния данных; процедуры, как правило, не активируются другими процедурами. Поэтому продукционные системы могут быть с большим успехом использованы для областей знаний, о которых располагаем только некоторым набором независимых правил (эвристик), а не четкой теорией, вполне завершенной и последовательной, и где поэтому нет алгоритмов, прямо приводящих к цели.

Продукционные системы все более широко используются для реализации продукционных ЭС.

Как правило, классические продукционные системы не содержат сведений о применении, т. е. знаний о том, например, какие продукции использовать для достижения цели. Это ведет к значительному снижению эффективности их работы: хотя в каждой итерации только одна продукция может быть активирована, должны быть проверены условия всех продукций. Для большого числа правил это может потребовать значительного расхода ресурсов. Более того, последовательность выполнения продукций зависит на каждой итерации от состояния всех переменных системы. В этом случае появляется проблема комбинаторного «взрыва».

Для решения этих проблем предлагались подходы, связанные с методами структурного совершенствования БД и условий в продукциях, что позволило бы повысить эффективность функционирования. Предпринимаются также попытки повлиять на ход управления.

В каждом цикле все правила, условия которых удовлетворены содержимым БД, считаются определенными. Если существует несколько таких правил, то вопрос о том, какое из правил выбрать, решается с помощью какой-либо приемлемой стратегии «разрешения конфликтов» (например, выбирается правило с наивысшим приоритетом из заранее определенного перечня приоритетов). Все действия, связанные с выбранным правилом, выполняются и вызывают соответствующие изменения в БД.

Другая возможность заключается в осуществлении точного контроля последовательности выполнении продукций. В простейшем случае продукции могли бы формировать специальные сигналы в БД, которые подключали бы соответствующие продукции в других циклах. В некоторых системах сами продукции могут активировать или дезактивировать другие продукции и даже влиять на работу интерпретатора.

Рассмотрим более подробно некоторые основные формы представления знаний.

В настоящее время разработано множество моделей представления знаний, используемых для реализации систем, основанных на знаниях, в которых знания представлены с помощью правил вида ЕСЛИ-ТОГДА (явление – реакция, условие – действие). Систему продукций можно считать наиболее распространенной моделью представления знаний. Примерами реальных систем, основанных на знаниях, в которых в качестве основной модели представления знаний использовалась система продукций, являются EMYCIN, OPS-5, AGE.

Если систему продукций рассматривать как модель представления знаний, то правилам, рассматриваемым с точки зрения человека как средства прямого описания способа логического вывода для решения задач в предметной области, можно придать ясный смысл. При этом отличительной чертой представления знаний с высокой модульностью является простота дополнения, модификации и аннулирования.

Кроме того, со стороны компьютера имеется возможность определения простого и точного механизма использования знаний с высокой однородностью, описанных по одному синтаксису. Эти две отличительные черты, по видимому являются причинами столь широкого распространения метода представлений знаний правилами.

5.3.2. Логические модели представления знаний

Исчисление предикатов

Классическим механизмом представления знаний в системах является исчисление предикатов (использовалось уже в 50-х годах в исследованиях по ИИ). В системах, основанных на исчислении предикатов, знания представляются с помощью перевода

утверждений об объектах некоторой предметной области в формулы логики предикатов и добавления их как аксиом в систему. Рассмотрим основные положения логики предикатов.

Пусть имеется некоторое множество объектов, *называемых предметной областью M*. Знаки, обозначающие элементы этого множества, называют *предметными константами*, а знак, обозначающий произвольный элемент этого множества, – *предметной переменной*. Терм – это всякая предметная область или предметная константа если f – функциональная n-местная буква, и $t_1, t_2,\ldots,t_n$ – термы, то $f(t_1,\ldots,t_n)$ есть терм.

Выражение $P(x_1,x_2,\ldots,x_n)$, где x_i, $i=\overline{1,n}$ - предметные переменные, а P принимает значения 0 и 1, называется *логической функцией* или предикатом. Переменные принимают значения из произвольного конечного и бесконечного множества *M*.

Предикатом или логической функцией называется функция от любого числа аргументов, принимающая истинные значения 1 и 0. Если в данном выражении заменить x_i на y_i, где y_i – предметные константы, то получим *элементарную формулу*, т.е. предикатные буквы применимы также и к предметным константам. Элементарные формулы иногда называют атомными. Из элементарных формул с помощью логических связок $\vee$ (или), $\wedge$ (и), $\neg$ (отрицание), $\rightarrow$ (импликация) строят предметные формулы (иногда их называют *правильно построенными формулами* - ППФ). ППФ – один из важных классов выражений в исчислении предикатов. Кроме логических связок в рассмотрение вводят кванторы общности $\forall$ или существования $\exists$.

Если P – предметная формула, а x – предметная переменная, то выражения

$\forall x\ P(x)$ и $\exists x\ P(x)$ также считаются предметными формулами. В логике предикатов для компактной записи высказываний типа: “для любого x истинно $P(x)$” и “существует такое x, для которого истинно $P(x)$” вводится две новые дополнительные операции: кантор общности $\forall$ и квантор существования $\exists$. Посредством этих операций приведенные выше высказывания записываются в виде $\forall x\ P(x)$ и $\exists x\ P(x)$. Выражение $\forall x\ P(x)$ обозначает высказывание истинное, когда $P(x)$ истинно при всех $x \in M$ и ложно в противном случае.

Если $P(x)$ в действительности не зависит от x, то выражения $\forall x\ P(x)$ и $\exists x\ P(x)$ обозначают то же, что и $P(x)$.

Конкретное вхождение переменной x в формулу P называется связанным, если оно либо непосредственно следует за каким-либо квантором, либо содержится в области действия некоторого квантора $\forall$ или $\exists$. Вхождение переменной является свободным, если оно не является связанным. В выражении $\forall x\ P(x,y)$ x- связанная, y- связанная.

Связанной переменной называется переменная, если в *P* имеется вхождение этой переменной.

Использование обоих кванторов не является обязательным. Рассмотрим выражение $\overline{\forall xP(x)}$, оно обозначает высказывание «$\forall xP(x)$ложно», равносильное высказыванию «существует элемент *x*, для которого *P(x)*» ложно или, что тоже «существует элемент *x*, для которого $\overline{P(x)}$истинно». Следовательно, $\overline{\forall xP(x)}$ равносильно выражению $\exists x\overline{P}(x)$: $\overline{\forall xP(x)} = \exists x\overline{P}(x)$.

Под *интерпретацией* предикатных формул понимают конкретизацию предметной области, соответствующей данной предметной формуле, и установке соответствия между символами, входящими в предмет, и элементами (а также функциями и отношениями), определяемыми в данной предметной области.

Вывод на предикатах.

Выводом системы представления знаний на предикатах являются формулы, выводимые из аксиом с помощью правил вывода. Для организации логического вывода могут использоваться правила.

Для решения конкретной задачи начальное состояние и доступные операторы действий переводятся в формулы исчисления предикатов и добавляются к множеству аксиом. Целевое состояние также выражается формулой и рассматривается как теорема, которая должна быть выведена из аксиом с помощью активного *механизма вывода*.

Определим основные формы логического вывода.

Индукция (лат. наведение) – это форма мышления, посредством которой мысль наводится на какое-либо общее правило, общее положение, присущее всем единичным предметам какого-либо класса. Индуктивный логический вывод является перспективным направлением инженерии знаний, здесь не рассматривается.

Дедукция (лат. выведение) – такая форма мышления, когда новая мысль выводится чисто логическим путем (т.е. по законам логики) из предшествующих мыслей. Такая последовательность мыслей называется *выводом*, а каждый компонент этого вывода является либо ранее *доказанной мыслью*, либо *аксиомой*, либо *гипотезой*. Последняя мысль данного вывода называется *заключением*.

Последовательность дедукции определяет “план” того, как достигнуть цели из начального состояния.

Дедукция обычно выполняется с помощью попытки вывести противоречие из получаемого в результате преобразований множества аксиом. Либо: для того, чтобы

показать, что некоторое множество ППФ неудовлетворимо, надо доказать, что нет такой интерпретации, при которой каждая из ППФ в этом множестве имеет значение 1(истинно). Хотя эта задача и кажется трудоемкой, существуют довольно эффективные процедуры ее решения. Для выполнения этих процедур требуется представить ППФ данного множества в специальном удобном виде – в виде *предложений*.

Процесс стандартизации.

Любую ППФ исчисления предикатов можно представить в виде предложения, применяя к ней последовательность простых операций. Задача состоит в том, чтобы показать, как придать произвольной ППФ форму предложения. Этот процесс (преобразования ППФ в форму предложения) состоит из следующих этапов:

1. *Исключение знаков импликации.* В форме предложения в исчислении предикатов явно используются лишь связки $\vee$ и $\neg$. Знак импликации можно исключить в исходном утверждении вместо $A \rightarrow B \quad \neg A \vee B$.

2. *Уменьшение области действия знаков отрицания.* Необходимо, чтобы знак отрицания $\neg$ применялся не более чем к одной предикатной букве.

3. *Стандартизация переменных*, при которой осуществляется переименование переменных с тем, чтобы каждый квантор имел свою переменную. Так, вместо $(\forall x)\{P(x) \rightarrow (\exists x)Q(x)\}$ следует написать $(\forall x)\{P(x) \rightarrow (\exists y)Q(y)\}$.

В области действия любого квантора переменная, связываемая им, является “немой” переменной. По этому везде в области действия квантора ее можно заменить другой переменной, а это не приведет к изменению значения истинности ППФ. Стандартизация переменных в ППФ означает переименование “немых ” переменных, с тем чтобы каждый квантор имел собственную немую переменную.

1. *Исключение кванторов существования.* Рассмотрим ППФ $(\forall y \exists x)P(x,y)$, которую можно интерпретировать, например, так: для всех y существует такой x (возможно, зависящий от y), что x больше y. Заметим, что так как квантор существования $(\exists x)$ находится внутри области действия квантора общности $(\forall y)$, допускается, что значение x может зависеть от значения y в x. Пусть эта зависимость определяется явно с помощью некоторой функции $g(y)$, отражающей каждое значение y в x, который «существует». Такая функция называется *функцией Сколема*. Если вместо x, который “существует”, взять функцию Сколема, то можно исключить квантор существования: $(\forall y)\ P(g(y),y)$. Общее правило исключения из ППФ квантора существования состоит в замене в ней всюду переменной, относящейся к квантору существования, функцией Сколема, аргументами

которой служат переменные, относящиеся к тем кванторам общности, области действия которых охватывают области действия исключамого квантора существования.

Функциональные буквы для функций Сколема должны быть “новыми”, в том смысле, что они не должны совпадать с теми буквами, которые уже имеются в ППФ. Если исключаемый квантор существования не принадлежит области действия ни одного из кванторов всеобщности, то функция Сколема не содержит аргументов, т.е. является постой константой: так ППФ $(\exists x)P(x)$ превращается в $P(a)$, где a - константа, про которую известно, что она существует. Чтобы исключить все переменные, относящиеся к кванторам существования, надо применить описанную ранее процедуру по очереди к каждой переменной.

2. *Приведение к предваренной нормальной форме* (ПНФ). На этом этапе уже не осталось кванторов существования, а каждый квантор общности имеет свою переменную. Теперь можно перенести все кванторы общности ($\forall$) в начало ППФ и считать областью действия каждого квантора часть ППФ. Про полученную таким образом ППФ говорят, что она имеет *предваренную нормальную форму* (ПНФ). ППФ в ПНФ состоит из цепочки кванторов, называемой *префиксом*, и расположенной за ней формулы, не содержащей кванторов, называемой *матрицей***.**

3. *Приведение матрицы к конъюнктивной нормальной форме* (КНФ). Любую матрицу можно представить в виде конъюнкций конечного множества дизъюнкций предикатов и (или) их отрицаний. Говорят, что такая матрица имеет КНФ. Заменить $A \vee \{B \wedge C\}$ на $\{A \vee B\} \wedge \{A \vee C\}$.

4. *Исключение кванторов общности.* Так как все переменные ППФ должны быть связанными, то все оставшиеся на этом этапе переменные относятся к $\forall$ (общности). Так как порядок расположения кванторов общности несущественен, то эти кванторы можно явным образом не указывать, условившись, что все переменные в матрице относятся к кванторам общности. Таким образом остается лишь матрица в КНФ.

5. *Исключение связок* $\wedge$, заменой $A \wedge B$ двумя ППФ A и B. Результатом многократной замены будет конечное множество ППФ, каждая из которых представляет дизъюнкцию атомных формул и (или) их отрицаний. Атомную формулу или ее отрицание называют *литерой* (литералом), а ППФ, состоящую лишь из дизъюнкций литер, *предложением***.** Этот процесс называется стандартизацией (т.е. преобразование формул в предложения).

Процесс, демонстрирующий, что некоторое множество F ППФ неудовлетворимо (невыполнимо), начинается с превращения каждой ППФ из множества F в предложение. В

результате возникает некоторое множество *S* предложений. Можно показать, что если множество *S* неудовлетворимо, то неудовлетворимо и множество *F*.

Далее используется *метод Эрбрана*, т.е. к стандартной форме формулы применяют процедуры поиска опровержения, это делается для удобства реализации поиска, т.е. вместо доказательства общезначимости формулы доказывается, что отрицание формулы противоречиво. Возможно использование *метода резолюций Робинсона* [122].

Система представления знаний на основе исчисления предикатов имеет ряд достоинств:

1. Они достаточно хорошо исследованы как формальная система.

2. Существуют ясные правила, т.е. результаты операций над БЗ также достаточно ясно определены.

Недостатками являются ограниченная выразимость, так как существует большое число факторов, которые тяжело или даже невозможно выразить средствами исчисления предикатов. В настоящее время аппаратное исчисление предикатов в чистом виде используется редко, однако основой большинства языков логического представления знаний, в частности, языка Пролог.

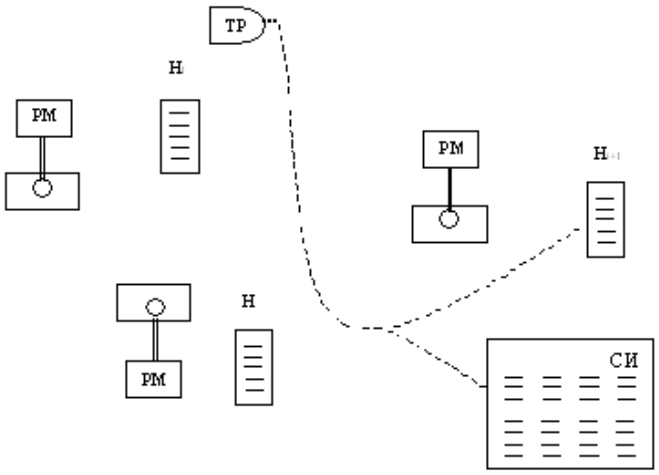

Рис.5.4. Схема роботизированного участка.

Пример:

Рассмотрим производственный роботизированный участок, оборудованный роботами манипуляторами (РМ), промежуточными накопителями готовых изделий (H_i), транспортным роботом (ТР) и складом изделий (СИ). См. рис. 5.4.

Задача ТР заключается в последовательном объезде промежуточных накопителей, сборе деталей (если они есть в накопителе) и транспортировке их на склад изделий. Если

тележка ТР заполнена, то необходимо раньше заехать на склад изделий, разгрузиться там, а затем продолжить объезд накопителей.

Введем следующие предикаты:

A = пуст (накопитель)

B = пуста (тележка ТР)

C = освободить (накопитель)

D = перейти ((i+1)- й накопитель)

E = перевезти (на склад)

F = вернуться (i-й накопитель)

Тогда сформулированные правила можно записать так:

1. $A \to D$
2. $\neg A \wedge B \to C \wedge D$
3. $\neg B \to E \wedge F$

Переведем предикаты в форму предложений:

1. $A \to D \equiv \neg A \vee D$
2. $\neg A \wedge B \to C \wedge D \equiv \neg(\neg A \wedge B) \vee (C \wedge D) =$
$= (A \wedge \neg B) \vee (C \wedge D) = (A \vee \neg B \vee C) \wedge (A \vee \neg B \vee D)$

Последнюю запись можно представить в виде двух ППФ:

$A \vee \neg B \vee C$, $A \vee \neg B \vee D$

2. $\neg B \to E \wedge F \equiv B \vee (E \wedge F) = (B \vee E) \wedge (B \vee F)$,

что также разбивается на две ППФ

$B \vee E$, $B \vee F$

Теперь система правил будет иметь вид:

1. $\neg A \vee D$
2. $A \vee \neg B \vee C$, $A \vee \neg B \vee D$
3. $B \vee E$, $B \vee F$