Лекция 3.

Тема: Модели системы представлений знаний для ИИ.

Традиционно, *системы представления знаний (СПЗ)* для ИС используют следующие основные виды *моделей*: *фреймы, исчисления предикатов, системы продукций, семантические сети*. Рассмотрим эти *модели* подробно.

Фреймы предложены в 1975 году Марвином Минским. *Фрейм* (рамка в переводе с англ.) - это единица *представления знаний*, запомненная в прошлом, детали которой могут быть изменены согласно текущей ситуации. *Фрейм* представляет собой структуру данных, с помощью которых можно, например, описать обстановку в комнате или место встречи для проведения совещания. М.Минский предлагал эту *модель* для описания пространственных сцен. Однако с помощью *фреймов* можно описать ситуацию, сценарий, роль, структуру и т.д.

Фрейм отражает основные свойства объекта или явления. Структура *фрейма* записывается в виде списка свойств, называемых во *фрейме* слотами. Рассмотрим *запись фрейма* на языке FRL (Frame *Representation Language*) - языке, похожем на *LISP*, но только внешне из-за наличия скобок.

Например, *фрейм* СТОЛ может быть записан в виде 3 слотов: *слот* НАЗНАЧЕНИЕ (purpose), *слот* ТИП (type) и *слот* ЦВЕТ (*colour*) следующим образом:

```
(frame СТОЛ
    (purpose (value(размещение предметов для
      деятельности рук)))
    (type (value(письменный)))
    (colour (value (коричневый))))
```

Во *фрейме* СТОЛ представлены только ДЕКЛАРАТИВНЫЕ средства для описания объекта, и такой *фрейм* носит название *фрейм* -образец. Однако существуют также *фреймы* - экземпляры, которые создаются для отображения фактических ситуаций на основе поступающих данных и ПРОЦЕДУРАЛЬНЫХ средств (демонов), например, следующих:

IF-DEFAULT - по умолчанию
IF-NEEDED - если необходимо
IF-ADDED - если добавлено
IF-REMOVED - если удалено

Слот IS-A или *AKO* (A Kind Of) определяет иерархию *фреймов* в сети *фреймов*. Такая *связь* обеспечивает *наследование свойств. Слот isa* указывает на *фрейм* более высокого уровня, откуда неявно наследуются свойства аналогичных слотов.

Рассмотрим фрагмент описания из "мира блоков" (рис. 1) в виде *фреймов*.

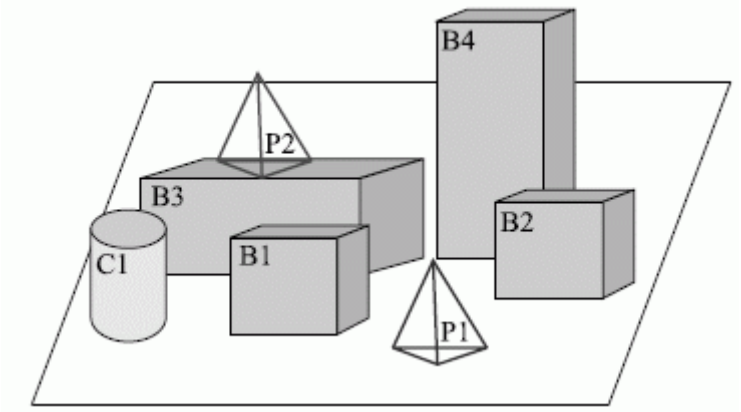

Рис. 1. "Мир блоков"

```
(frame (name (Cube))
   (isa (Block World))
   (length (NULL))
   (width (IF-DEFAULT (use length)))
```

```
   (height (IF-DEFAULT (use length))))
 (frame (name (B1))
  (isa (Cube))
  (color (red))
  (length (80)))
 (frame (name (B2))
  (isa (Cube))
  (color (green))
  (length (65))
  (who_put (value (NULL))
    (IF_NEEDED (askuser))))
```

Слот isa указывает на то, что объекты B1 и B2 являются *подтипом* объекта *Cube* и наследуют его свойства, а именно, length = width = height. *Демон* IF_NEEDED запускается автоматически, если понадобится узнать, кто поставил B2 на стол. Полученный ответ (Робби) будет подставлен в *значение* слотаwho_put. Аналогично работают демоны IF-ADDED и IF-REMOVED.

Допустим, однорукому роботу Робби дается приказ "Возьми желтый предмет, который поддерживает *пирамиду*". На языке *представления знаний* (ЯПЗ) вопрос записывается так:

```
(object ? X
   (color (yellow))
   (hold ? Y
     (type (pyramid))))
```

Программа сопоставления с образцом находит в базе знаний описание объектов:

```
(frame (name (B3))
   (type (block))
   (color (yellow))
   (size (20 20 20))
   (coordinate (20 50 0))
   (hold (P2)))
```

и

```
(frame (name (P2))
  (type (pyramid))
  ...)
```

Ответ получен X = B3, Y = P2 и Робби выдается *команда* take(object=B3).

Таков общий механизм *представления знаний* в виде *фреймов*. Реализация этого механизма потребует решения других, более сложных проблем, например, автоматического ввода знаний для трехмерных объектов, работы с трехмерными быстродвижущимися объектами (своеобразный тест на реакцию) и т.д. Эти проблемы ждут своего эффективного решения.

Исчисления предикатов

Традиционная булева алгебра и исчисление высказываний не всегда подходят для выражения логических рассуждений, проводимых людьми, более удобен для этого язык логики предикатов. Под ***исчислением предикатов*** понимается *формальный язык* для представления отношений в некоторой предметной области. *Исчисление предикатов* подробно обсуждается в ряде книг по теории ИИ. Основное преимущество *исчисления предикатов* - хорошо понятный мощный механизм математического вывода, который может быть непосредственно запрограммирован. Дальнейшее изложение ведется с учетом того, что читатель знаком с основами булевой алгебры.

Предикатом называют предложение, принимающее только два значения: "*истина*" или "*ложь*". Для обозначения предикатов применяются логические связки между высказываниями: $\neg$ - не, $\vee$ - или, $\wedge$ - и, $\supset$ - если, а также квантор $\exists$ существования и *квантор всеобщности* $\forall$

$\exists x(\dots)$ - существует такой x, что ...

$\forall x(\dots)$ - для любого x

Таким образом, *логика предикатов* оперирует логическими связками между высказываниями, например, она решает вопросы: можно ли на основе высказывания A получить *высказывание* B и т.д.

Рассмотрим некоторые примеры. *Высказывание* "у каждого человека есть отец" можно записать:

$$\forall x \exists y(\text{человек}(x) \supset \text{отец}(y, x))$$

Выражение "Джон владеет красной машиной" записывается, например, так:

$$\exists x(\text{владеет}(\text{Джон}, x) \supset \text{машина}(x) \wedge \text{красный}(x))$$

Рассмотрим *вывод*, дающий заключение на основе двух предпосылок:

Предпосылка 1: Все люди смертны

$$\forall x(\text{человек}(x) \supset \text{смертен}(x))\forall x(p(x) \supset q(x))$$

Предпосылка 2: Сократ - человек

p(a)

Заключение: Сократ - смертен

Смертен(Сократ)

q(a)

Если обозначить через f функцию одного аргумента, то логическая формула для этого высказывания будет иметь вид:

$$\forall x(f(x) \supset q(x))$$

Алфавит логики предикатов состоит из элементов (символов):

x, y, z, u, v, w - переменные;

a, b, c, d, e - *константы*;

f, g, h - функциональные символы;

p, q, r, s, t - *предикатные символы*;

$\neg, \vee, \wedge, \supset, \forall, \exists$ - логические символы.

Запишем на языке *исчисления предикатов* некоторое *выражение*:

$$\exists y \forall x(\text{человек}(x) \supset \text{отец}(y, x))$$

Что означает записанное *выражение*? Ответ очевиден: "у всех людей общий отец".

Приведем пример простого доказательства на языке *исчисления предикатов*.

Даны следующие факты:

1. "Иван является отцом Михаила" - отец(a,b)
2. "Петр является отцом Василия" - отец(c,d)
3. "Иван и Петр являются братьями" -

$$\exists w(\text{брат}(a, c) \supset \text{отец}(w, a) \wedge \text{отец}(w, c))$$

Даны следующие определения:

4. "Брат отца является дядей" -

$$\exists y(\text{дядя}(x, u) \supset \text{отец}(y, u) \wedge \text{брат}(y, x))$$

5. "Сын дяди является двоюродным братом" -

$$\exists x(\text{дв.брат}(z, u) \supset \text{дядя}(x, u) \wedge \text{отец}(x, z))$$

Требуется доказать, что "Михаил и Василий являются двоюродными братьями":

6. $$\exists x \exists y(\text{дв.брат}(b, d) \supset \text{отец}(y, b) \wedge \text{брат}(y, x) \wedge \text{отец}(x, d))$$

Делаем подстановки y = Иван, b = Михаил и x = Петр, d = Василий, видим, что предикаты 1, 2, 3 дают правильное предложение 6.

Рассмотренный нами язык называется *исчислением предикатов* первого порядка и позволяет связывать знаком квантора переменные, соответствующие объектам из *предметной области*, но не предикаты или функции.

Исчисление предикатов второго порядка позволяет связывать знаком квантора не только переменные, соответствующие объектам из *предметной области*, но и предикаты или функции. Примером *исчисления предикатов* второго порядка может служить *выражение* "Единственное качество Джона - это честность", которое записывается так:

$$\exists P(P(\text{Джон}) \wedge \text{качество}(P) \supset P = \text{честность})$$

На этом мы закончим знакомство с этой *моделью* и вернемся к ней в следующей лекции при рассмотрении правил вывода, *принципа резолюции* и методов поиска на основе *исчисления предикатов*.

Системы продукций

Под продукцией будем понимать *выражение*:

Если <X_1, X_2 ... X_n> то

<{Y_1, D_1}, ... {Y_m,D_m}>,

где: X_i,Y_i - *логические выражения*, D_i - фактор достоверности (0,1) или фактор уверенности (0,100).

Системы продукций - это набор правил, используемый как база знаний, поэтому его еще называют базой правил. В Стэндфордской теории фактор уверенности CF (certainty *factor*) принимает значения от +1 (максимум доверия к гипотезе) до -1 (минимум доверия).

А.Ньюэлл и Г.Саймон отмечали в *GPS*, что продукции соответствуют навыкам решения задач человеком в долгосрочной памяти человека. Подобно навыкам в долгосрочной памяти эти продукции не изменяются при работе системы. Они вызываются по "образцу" для решения данной специфической проблемы. Рабочая *память продукционной системы* соответствует краткосрочной памяти, или текущей области внимания человека. Содержание рабочей области после решения задачи не сохраняется.

Работа *продукционной системы* инициируется начальным описанием (состоянием) задачи. Из продукционного *множества* правил выбираются правила, пригодные для применения на очередном шаге. Эти правила создают так называемое *конфликтное множество*. Для выбора правил из *конфликтного множества* существуют стратегии *разрешения конфликтов*, которые могут быть и достаточно простыми, например, выбор первого правила, а могут быть и сложными эвристическими правилами. Продукционная *модель* в чистом виде не имеет механизма выхода из тупиковых состояний в процессе поиска. Она продолжает работать, пока не будут исчерпаны все допустимые продукции. Практические реализации *продукционных систем* содержат *механизмы возврата* в предыдущее состояние для управления алгоритмом поиска.

Рассмотрим пример использования *продукционных систем* для решения шахматной задачи хода конем в упрощенном варианте на доске размером 3 x 3. Требуется найти такую последовательность ходов конем, при которой он ставится на каждую клетку только один раз (рис. 2).

Записанные на рисунке предикаты move(x,y) составляют базу знаний (*базу фактов*) для задачи хода конем. Продукционные правила - это факты перемещений move, первый *параметр* которых определяет условие, а второй *параметр* определяет действие (сделать ход в *поле*, в которое конь может перейти). Продукционное множество правил для такой задачи приведено ниже.

P1: If (конь в поле 1) then (ход конем в поле 8)
P2: If (конь в поле 1) then (ход конем в поле 6)
P3: If (конь в поле 2) then (ход конем в поле 9)
P4: If (конь в поле 2) then (ход конем в поле 7)
P5: If (конь в поле 3) then (ход конем в поле 4)
P6: If (конь в поле 3) then (ход конем в поле 8)
P7: If (конь в поле 4) then (ход конем в поле 9)

P8: If (конь в поле 4) then (ход конем в поле 3)
P9: If (конь в поле 6) then (ход конем в поле 1)
P10: If (конь в поле 6) then (ход конем в поле 7)
P11: If (конь в поле 7) then (ход конем в поле 2)
P12: If (конь в поле 7) then (ход конем в поле 6)
P13: If (конь в поле 8) then (ход конем в поле 3)
P14: If (конь в поле 8) then (ход конем в поле 1)
P15: If (конь в поле 9) then (ход конем в поле 2)
P16: If (конь в поле 9) then (ход конем в поле 4)

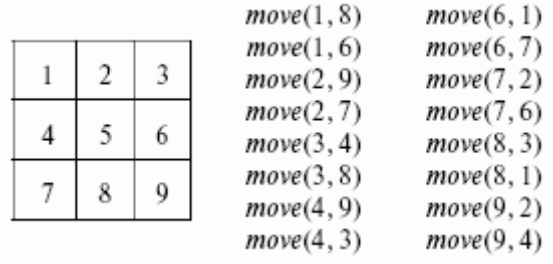

Рис. 2. Шахматная доска 3x3 для задачи хода конем с допустимыми ходами

Допустим, необходимо из исходного состояния (*поле* 1) перейти в целевое состояние (*поле* 2). Итерации *продукционной системы* для этого случая игры показаны в таблице 1.

Таблица 2.1. Итерации для задачи хода конем

№ итерации	**Текущее поле**	**Целевое поле**	***Конфликтное множество***	**Активация правила**
1	1	2	1, 2	1
2	8	2	13, 14	13
3	3	2	5, 6	5
4	4	2	7, 8	7
5	9	2	15, 16	15
6	2	2		Выход

Продукционные системы могут порождать бесконечные циклы при поиске решения. В *продукционной системе* эти циклы особенно трудно определить, потому что правила могут активизироваться в любом порядке. Например, если в 4-й итерации выбирается правило 8, мы попадаем в *поле* 3 и зацикливаемся. Самая простая стратегия *разрешения конфликтов* сводится к тому, чтобы выбирать первое соответствующее перемещение, которое ведет в еще не посещаемое состояние. Следует также отметить, что конфликтное множество это простейшая *база целей*. В следующей лекции мы рассмотрим различные стратегии поиска в *продукционных системах* и пути разрешения конфликтов. В заключение данного раздела лекции перечислим основные преимущества *продукционных систем*:

- простота и гибкость выделения знаний;
- отделение знаний от программы поиска;
- модульность продукционных правил (правила не могут "вызывать" другие правила);
- возможность эвристического управления поиском;
- возможность трассировки "цепочки рассуждений";
- независимость от выбора языка программирования;
- продукционные правила являются правдоподобной моделью решения задачи человеком.

Семантические сети

Семантика в бытовом понимании означает смысл слова, художественного произведения, действия и т.д. ***Семантическая сеть (СС)*** - это граф, дуги которого есть отношения между вершинами (значениями). *Семантические сети* появились как *модель СПЗ* при решении задач

разбора и понимания смысла естественного языка. *Модели* в виде *СС* активно развиваются в работах зарубежных и отечественных ученых, вбирая в себя важнейшие свойства других типов *моделей*.

Пример *семантической сети* для предложения типа "Поставщик осуществил поставку изделий по заказу клиента до 1 июня 2004 года в количестве 1000 штук" приведен на рис. 3.

Рис. 3. Пример семантической сети

На этом примере видно, что между объектами Поставщик и Поставка определено *отношение* "*агент*", между объектами Изделие и Поставка определено *отношение* "*объект*" и т.д.

Число отношений, используемых в конкретных *семантических сетях,* может быть самое разное. К.Филмор, один из первых поборников идеи семантических падежей при разборе предложений, проводил свои рассуждения, пользуясь дюжиной отношений. Неполный *список* возможных отношений, используемых в *семантических сетях* для разбора предложений, выглядит следующим образом.

Агент - это то, что (тот, кто) вызывает действие. *Агент* часто является подлежащим в предложении, например, "**Робби** ударил мяч".

Объект - это то, на что (на кого) направлено действие. В предложении *объект* часто выполняет роль прямого дополнения, например, "Робби взял желтую **пирамиду**".

Инструмент - то средство, которое используется агентом для выполнения действия, например, "Робби открыл дверь **с помощью ключа** ".

Соагент служит как подчиненный партнер главному агенту, например, "Робби собрал кубики **с помощью Суззи**".

Пункт отправления и *пункт* назначения - это отправная и конечная позиции при перемещении агента или объекта: "Робби перешел **из комнаты в библиотеку**".

Траектория - перемещение от пункта отправления к пункту назначения: "Они прошли **через дверь по ступенькам на лестницу**".

Средство доставки - то в чем или на чем происходит перемещение: "Он всегда едет домой на **метро**".

Местоположение - то *место,* где произошло (происходит, будет происходить) действие, например, "Он работал **за столом**".

Потребитель - то лицо, для которого выполняется действие: "Робби собрал кубики **для Суззи**".

Сырье - это, как правило, материал, из которого что-то сделано или состоит. Обычно сырье вводится предлогом из, например, "Робби собрал Суззи **из интегральных схем**".

Время - указывает на момент совершения действия: "Он закончил свою работу **поздно вечером**".

Наиболее типичный способ вывода в *семантических сетях (СС)* - это способ сопоставления частей сетевой структуры. Это видно на следующем простом примере, представленном на рис. 4.

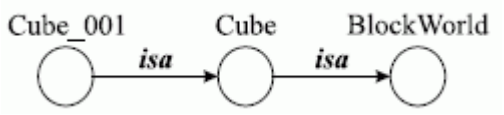

Рис. 4. Процедура сопоставления в СС

Куб *Cube* принадлежит миру BlockWorld.

Куб Cube_001 есть разновидность куба *Cube*.

Легко сделать *вывод*:

Куб Cube_001 есть часть мира BlockWorld.

Еще один пример поиска в *СС*. Представим вопрос "какой *объект* находится на желтом блоке?" в виде подсети, изображенной на рис. 5. Произведем сопоставление вопроса с сетью, представленной на рис. 6. В результате сопоставления получается ответ - "*Пирамида*".

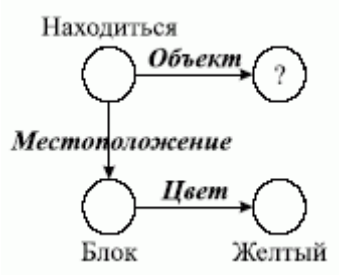

Рис. 5. Вопрос в виде СС

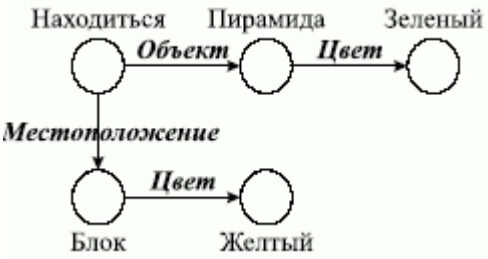

Рис. 6. Процедура сопоставления в СС