ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Б1.В.ДВ.05.01 Представление знаний в интеллектуальных системах

Направление подготовки 27.03.04 Управление в технических системах

Квалификация (степень) выпускника бакалавр

Форма обучения заочная

СОДЕРЖАНИЕ

1. КОНСПЕКТ ЛЕКЦИЙ

1.1 Лекция № 1 (2 часа).

Тема: «Язык программирования Пролог**»**

1.1.1 Краткое содержание вопросов:

Теоретические принципы Пролога

Пролог существенно отличается от языков, традиционно используемых в программировании. В языках Бейсик, Алгол и Паскаль основным методом программирования является разбиение задачи на дискретные шаги и их последовательное описание. Последовательность шагов отображается в машинные команды, выполняемые компьютером. Отменить ранее выполненные команды невозможно, поскольку содержимое памяти постоянно обновляется. Языки программирования такого вида называются *алгоритмическими.* Алгоритм - математическая процедура, которая позволяет решить задачу за конечное число шагов. Пролог не относится к алгоритмическим языкам.

Свое название Пролог получил от слов «Программирование на языке ЛОГики». На самом деле Пролог не считается чистым языком логического программирования, но его создание - важный этап в этом направлении. Большой интерес к Прологу проявляют специалисты в области инженерии знаний и искусственного интеллекта. Растет его популярность в академическом мире.

Пролог был принят в качестве базового языка в японской программе создания ЭВМ пятого поколения, ориентированной на исследование методов логического программирования и искусственного интеллекта, а также на разработку нового поколения компьютеров, специально предназначенных для реализации данных методов.

При программировании на Прологе значительно упрощается описание решения, и программист имеет возможность заниматься непосредственно задачей, а не поиском способа разбиения ее решения на небольшие шаги, которые можно запрограммировать, как при программировании на алгоритмических языках.

Теоретической основой Пролога является раздел символьной логики, называемый *исчислением предикатов.* Прологу присущ ряд свойств, которыми не обладают традиционные языки программирования, что делает его мощным средством в области логического программирования. К таким свойствам относятся механизм вывода с поиском и возвратом, встроенный механизм сопоставления с образцом, и простая, но выразительная структура данных с возможностью ее изменения. Пролог отличает единообразие программ и данных. Данные и программы - лишь две различные точки зрения на объекты Пролога. В единой базе данных можно свободно создавать и уничтожать отдельные элементы. Поскольку не существует различия между программами и данными, можно менять программу во время ее работы. В Прологе отсутствуют указатели, операторы присваивания и GO__TO. Естественным методом программирования является рекурсия.

Многие интересные и полезные свойства Пролога непосредственно вытекают из его декларативности. Декларативность позволяет понимать программу, не отслеживая динамику ее выполнения, и является уникальной особенностью Пролога. Однако истинная мощь Пролога базируется не на одном каком-либо свойстве, а на сочетании всех его свойств. Мы приведем их подробное описание в последующих главах, а сейчас дадим общее представление о языке Пролог.

Мы не ставим перед собой цель описать исчисление предикатов, приведем лишь основные принципы, необходимые для изучения Пролога.

ПРОЛОГ является языком исчисления предикатов. Предикат – это логическая формула от одного или нескольких аргументов. Можно сказать, что предикат – это функция, отображающая множество произвольной природы в множество {ложно, истинно}. Обозначаются предикаты F(x), F(x, y), F(x, y, z) и т. д. Одноместный предикат F(x), определенный на предметной области M, является истинным, если у объекта x есть свойство F, и ложным – если этого свойства нет.

Двухместный предикат F(x, y) является истинным, если объекты x и y находятся в отношении F. Многоместный предикат F(x1 , x2 , x3 ,..., xN) задает отношение между элементами x1, x2, ..., xN и интерпретируется как обозначение высказывания: “Элементы x1 , x2, xN находятся между собой в отношении F”. При разработке логических программ предикаты получают обычно названия, соответствующие семантике описываемой предметной области.

Мы не будем употреблять термины «истинно» и «ложно», поскольку они определяют смысл результата. Доказанное утверждение обычно является истинным, но не обязательно, так как доказательство зависит от известных фактов и сделанных на их основе выводов. Итак, мы ввели два новых термина. *Факт* есть некоторое утверждение, все факты определяются как доказанные. В частности, факт «рекс - это собака» считается доказанным по определению. Но такой запрос к факту, как «Является ли феликс собакой ?», не будет доказан с помощью известных фактов. Отрицательный ответ системы Пролог означает вовсе не то, что утверждение «феликс - собака» ложно, а лишь то, что у нас нет фактов о «феликсе» и о том, является ли он собакой. Другим термином, который мы использовали, был *вывод.* Имея множество фактов, мы можем определять новые свойства описанных в них объектов. Иными словами, допускается вывод свойств из фактов. Вернемся к факту «реке - это собака». Предположим, мы хотим на основе известных фактов выявить все объекты, обладающие свойством «быть собакой». Задав вопрос «Какие существа являются собаками?», получим ответ: «Рекс - это собака». Мы привели тривиальный пример вывода. Введем некоторую дополнительную информацию.

Если сформулировать приведенный выше запрос в алгоритмических терминах, то получится весьма сложное определение:

Объект является собакой в том случае, если он имеет свойство «быть собакой» или же имеет свойство «быть родителем» и существует объект, связанный с родителем и обладающий свойством «быть собакой».

Для того чтобы продемонстрировать алгоритмический подход к рассматриваемой задаче, приведем соответствующую программу на Паскале.

В приводимой ниже программе прежде всего описывается информация, относящаяся к «рексу», «голди» и «джоку», а затем перечисляются объекты, являющиеся собаками по определению и вследствие вывода (слайды 4 и 5).

Используемый нами синтаксис Пролога подробно будет определен позже. Пока примем, что каждый элемент, записанный строчными буквами, есть литерал. Он представляет объект и не может быть изменен. Каждое слово, начинающееся с прописной буквы (например, X, Y, Кто), является переменной. Во время выполнения программы переменным присваиваются значения. Символ ?- означает попытку согласовать, т.е. доказать следующую за ним цель (собака (Кто) является целью).

Мы определили, что рекс - это собака. Некоторый X является собакой (X называется *именованной переменной)* при условии, что X является родителем Y (другой свободной именованной переменной) и Y является собакой. Мы задали также условия, что голди - это родитель рекса и джок - это родитель рекса.

Синтаксис языка Турбо Пролог

Объекты данных Турбо Пролог

Система распознает тип объектов по его синтаксической форме в тексте программы. Это возможно благодаря тому, что синтаксис записи для различных типов имеет свои формы. Переменные начинаются с прописной буквы, атомы со строчной. Для изображения переменных и атомов используются буквы латинского алфавита: прописные (A, B, …, Z), строчные (a, b…z), цифры (0…9), специальные символы (, / (+ – * / < > = : _).

Атомы можно создать тремя способами:

1. из цепочки букв, цифр и символа подчеркивания, начиная со строчной буквы: x, sister, y10, x_y ;

2. из цепочки символов, заключенный в одинарные кавычки, это используется, если надо иметь атом начинающийся с прописной буквы: ' X ', ' Anna';

3. из специальных символов: < - - - >, = = = = . , . . . , :: =.

Объекты данных в Турбо-Прологе называются термами.

Константы представляют собой самоопределённые термы, т.е. их «значением» является их графическое представление (последовательность символов или число). Константы в Прологе используются для обозначения (именования) конкретных объектов предметной области и конкретных отношений между ними.

Переменная – последовательность букв, цифр и знака «подчеркивание», обязательно начинающаяся с прописной (большой) буквы или знака «подчеркивание». Среди переменных выделена переменная "_" (один знак подчеркивания), называемая анонимной. Она используется в ситуациях, когда нас не интересует её значение.

Область действия переменных

Областью действия переменной является утверждение. В пределах утверждения одно и то же имя принадлежит одной и той же переменной. Два утверждения могут использовать одно имя переменной совершенно различным образом. Единственным исключением из правила определения области действия переменных является анонимная переменная, например, "_" в цели любит(X,_). Каждая анонимная переменная есть отдельная сущность.

Она применяется тогда, когда конкретное значение переменной несущественно для данного утверждения. Таким образом, каждая анонимная переменная четко отличается от всех других анонимных переменных в утверждении. Переменные, отличные от анонимных, называются именованными, а неконкретизированные (переменные, которым не было присвоено значение) называются свободными.

Структура представляется на языке Пролог с помощью указания её функтора и компонент в следующем виде:

Имя структуры = функтор(компонента-1, ..., компонента-N),

где в качестве функтора должен выступать атом, а компонентой может быть любой терм (в том числе и структура).

Если объекты структуры относятся к одному и тому же типу доменов, то такая структура называется однодоменной. Если объекты структуры относятся к разным типам доменов – многодоменная.

Объекты могут быть константами, переменными или структурами. Для того, чтобы объединить объекты в структуры, надо выбрать функтор. Каждый функтор определяется двумя параметрами:

- именем, синтаксис которого совпадает с именем предиката;
- арностью – т. е. числом аргументов.

Рассмотрим в структуру date:

date (1, september, 2002),

где date – функтор, все аргументы данной многодоменной структуры являются константами.

date (Day, september,2002)

В этом примере date представляет функтор многодоменной структуры, аргументы которой являются переменные и константы.

likes ('Katja', fruits(banana, apples, oranges)),

А в этом примере likes – главный функтор, аргументы данной многодоменной структуры – константа и структура.

Рассмотренную структуру в программе можно описать следующим образом:

domains

personal_liking = fruits(type1,type2, type3)

type1, type2, type3 = symbol

predicates

likes(symbol, personal_liking)

Нормальная форма Бэкуса-Наура (БНФ),

Начнем с того, что познакомимся с так называемой **нормальной формой Бэкуса-Наура (БНФ)**, разработанной в 1960 Джоном Бэкусом и Питером Науром и используемой для формального описания синтаксиса языков программирования. Впервые БНФ была применена Питером Науром при записи синтаксиса языка Алгол-60.

При описании синтаксиса конструкций используются следующие обозначения:

Символ ::= читается как "по определению" ("это", "есть"). Слева от разделителя располагается объясняемое понятие, справа - конструкция, разъясняющая его. Например,

<Имя> ::= <Идентификатор>

В угловые скобки заключается часть выражения, которая используется для обозначения синтаксической конструкции языка, в частности объясняемое понятие. В приведенном выше примере это <Имя> и <Идентификатор>.

Символ | означает в нотации БНФ "или", он применяется для разделения различных альтернативных растолкований определяемого понятия.

Пример. Десятичную цифру можно определить следующим образом:

<цифра> ::= 0|1|2|3|4|5|6|7|8|9

Часть синтаксической конструкции, заключенная в квадратные скобки, является необязательной (может присутствовать или отсутствовать);

Пример. Запись

<Целое число> ::= [-]<Положительное целое число>

означает, что целое число можно определить через положительное целое число, перед которым может стоять знак минус.

Символ * обозначает, что часть синтаксической конструкции может повторяться произвольное число раз (ноль и более). Заметим, что иногда вместо символа * используют фигурные скобки ({,}).

Пример. Определить положительное целое число в нотации БНФ можно следующим образом:

<Положительное целое число> ::= <цифра>[<цифра>]*.

То есть положительное целое число состоит из одной или нескольких цифр.

Программа на языке Пролог, ее иногда называют базой знаний, состоит из предложений (или **утверждений**), каждое предложение заканчивается точкой.

Утверждения

Программирование на языке Пролог состоит из следующих этапов:

– Объявления некоторых фактов об объектах и отношениях между ними.

– Определения некоторых правил об объектах и отношениях между ними.

– Формулировки вопросов об объектах и отношений между ними.

Программа на Прологе есть совокупность утверждений. Утверждения состоят из целей и хранятся в базе данных Пролога. Таким образом, база данных Пролога может рассматриваться как программа на Прологе. В конце утверждения ставится точка ".". Иногда утверждение называется предложением. Основная операция Пролога - доказательство целей, входящих в утверждение.

Предложения бывают двух видов: *факты*, *правила*.

Предложение имеет вид

A:-

B1,... , Bn.

A называется заголовком или головой предложения, а B1,..., Bn - телом.

Факты

Факт констатирует, что между объектами выполнено некоторое отношение. Он состоит только из заголовка. Можно считать, что *факт* - это *предложение*, у которого тело пустое. Например, известный нам *факт*, что Наташа является мамой Даши, может быть записан в виде:

мама(Наташа, Даша).

Или предположим, что мы хотим сообщить Прологу факт: «Джону нравиться Мэри». Этот факт включает в себя два объекта Джон и Мэри и отношение между ними нравится. Этот факт на Прологе можно записать следующим образом:

нравится (джон, мэри)

Важно соблюдать следующие правила:

– Имена всех отношений и объектов должны начинаться со строчной буквы. Например, нравится, мэрии, джон.

– Сначала записывается имя отношения. Затем через запятую записываются имена объектов, а весь список имен объектов заключается в круглые скобки.

– Каждый факт должен заканчиваться точкой.

Факт представляет собой безусловно истинное утверждение.

В математической логике, отношения принято называть **предикатами.**

Если воспользоваться нормальной формой Бэкуса-Науэра, то предикат можно определить следующим образом:

<Предикат>::=<Имя> | <Имя>(<аргумент>[,<аргумент>]*),

т.е. предикат состоит либо только из имени, либо из имени и следующей за ним последовательности аргументов, заключенной в скобки. Аргументом или параметром предиката может быть константа, переменная или составной объект. Число аргументов предиката называется его **арностью** или **местностью**. В Турбо Прологе имя предиката должно состоять из последовательности латинских букв, цифр, знаков подчеркивания и начинаться с буквы или знака подчеркивания. В других версиях Пролога имя предиката может содержать символы не только из английского алфавита, но и из национального, например, из русского.

Соответственно, приведенный выше пример *факта* можно записать в Турбо Прологе, например, так:

mother("Наташа", "Даша").

Определяя с помощью фактов отношения между объектами, необходимо учитывать, в каком порядке перечисляются имена объектов внутри круглых скобок. Этот

порядок может быть произвольным, но, выбрав один раз какой-то определенный порядок, мы должны следовать ему всегда.

Приведем некоторые факты и постараемся их интерпретировать:

ценный (золото)

женщина (джейн)

владеет (джон, золото)

отец (джон, мэри)

дает (джон, книга, мэрии)

Имена объектов и отношений являются полностью произвольным. Например, вместо терма нравится (джон, мэри) мы могли с таким же успехом написать a (b,c). Но для читабельности программы, лучше всего использовать имена, которые доносят до нас какой-то смысл.

Совокупность фактов в Прологе называется базой данных.

Некоторые предикаты уже известны системе, они называются **стандартными** или **встроенными**.

Унификация

Одним из наиболее важных аспектов программирования на Прологе являются понятия унификации (отождествления) и конкретизации переменных. Пролог пытается отождествить термы при доказательстве, или согласовании, целевого утверждения.

Например, в программе для согласования запроса ?- собака(X) целевое утверждение собака (X) было отождествлено с фактом собака (рекс), в результате чего переменная X стала конкретизированной: X= рекс. Переменные, входящие в утверждения, отождествляются особым образом - сопоставляются. Факт доказывается для всех значений переменной (переменных). Правило доказывается для всех значений переменных в головном целевом утверждении при условии, что

хвостовые целевые утверждения доказаны. Предполагается, что переменные в фактах и головных целевых утверждениях связаны квантором всеобщности. Переменные принимают конкретные значения на время доказательства целевого утверждения. В том случае, когда переменные содержатся только в хвостовых целевых утверждениях, правило считается доказанным, если хвостовое целевое утверждение истинно для одного или более значений переменных. Переменные, содержащиеся только в хвостовых целевых утверждениях, связаны квантором существования. Таким образом, они принимают конкретные значения на то время, когда целевое утверждение, в котором переменные были согласованы, остается доказанным. Терм X сопоставляется с термом Y по следующим правилам. Если X и Y - константы, то они сопоставимы, только если они одинаковы. Если X является константой или структурой, а Y - неконкретизированной переменной, то X и Y сопоставимы и Y принимает значение X (и наоборот). Если X и Y - структуры, то они сопоставимы тогда и только тогда, когда у них одни и те же главный функтор и арность и каждая из их соответствующих компонент сопоставима. Если X и Y - неконкретизированные (свободные) переменные, то они сопоставимы, в этом случае говорят, что они сцеплены.

1.2 Лекция № 2 (2 часа).
Тема: «Семантические сети»
1.2.1 Краткое содержание вопросов:

Термин семантическая означает смысловая, а сама семантика — это наука, устанавливающая отношения между символами и объектами, которые они обозначают, т.е. наука, определяющая смысл знаков.

В основе семантических моделей лежит понятие сети, образованной помеченными вершинами и дугами. Вершины сети представляют некоторые сущности (абстрактные или конкретные объекты, процессы, события, явления), а дуги — отношения между сущностями, которые они связывают (связи типа «это», «имеет частью», «принадлежит» и т.п.). Наложив ограничения на описание дуг и вершин, можно получить сети различного вида. Если вершины не имеют собственной внутренней структуры, то соответствующие сети называются простыми. Если вершины обладают некоторой структурой, то такие сети называют иерархическими.

Характерной особенностью семантических сетей является обязательное наличие трех типов отношений: класс — элемент класса, свойство — значение и пример элемента класса.

В самом общем случае сетевая модель — это информационная модель предметной области. В сетевой модели представляются множество информационных единиц (объекты и их свойства, классы объектов и их свойств) и отношения между этими единицами.

В зависимости от типов отношений между информационными единицами различают сети:

классификационные, в которых используются отношения (типа часть – целое, род, вид, индивид), описывающие структуру предметной области, что позволяет отражать в базах знаний разные иерархические отношения между информационными единицами;

– функциональные (их часто называют вычислительными моделями), позволяющие описывать процедуры «вычислений» одних информационных единиц через другие;

– каузальные (называемые также сценариями), использующие причин-но-следственные отношения, а также отношения типа «средство — результат», «орудие — действие» и т.п.

– смешанные, использующие разнообразные типы отношений.

Если в сетевой модели допускаются отношения различного типа, то ее обычно называют семантической сетью.

Важной чертой семантических сетей является возможность представлять знания более естественным и структурированным образом, чем это делается с помощью других формализмов.

Виды отношений

При разработке представлений в виде семантических сетей важную роль играют следующие отношения:

– «является», отражающее принадлежность объекта некоторому классу объектов;
– «имеет», указывающее на то, что одно понятие представляет часть другого;
– «есть», указывающая на то, что одно понятие служит атрибутом другого.

Можно предложить несколько классификаций семантических сетей, связанных с типами отношений между понятиями. По количеству типов отношений сети делятся на однородные (с единственным типом отношений) инеоднородные (с различными типами отношений). По типам отношений их можно классифицировать какбинарные (в которых

отношения связывают два объекта) in-арные (в которых есть специальные отношения, связывающие более двух понятий).

Семантические отношения бывают:

– Лингвистическими (глагольными, атрибутивными и падежными) — отображают смысловую взаимосвязь между событиями, между событиями и понятиями или свойствами. Они бывают глагольными, атрибутивными и падежными.

– Логическими — операциями, используемыми в алгебре логики (дизъюнкция, конъюнкция, инверсия и импликация).

– Теоретико-множественными — отношениями подмножества, части целого, множества и элемента.

– Квантифицированными — логическими кванторами общности и существования. Они используются для представления таких знаний как «Существует работник А, обслуживающий склад В» и т.д.

Наиболее часто в семантических сетях используются следующие отношения:

– связи типа «часть — целое» («класс — подкласс», «элемент — множество» и т.п.);

– функциональные связи (определяемые обычно глаголами «производит», «влияет» и т.п.);

– качественные связи (больше, меньше, равно и т.д.);

– пространственные связи (далеко от, близко от, за, под, над и т.п.);

– временные связи (раньше, позже, в течение и т.д.);

– атрибутивные связи (иметь свойство, иметь значение и т.п.);

– логические связи (и, или, не);

– и другие.

Проблема поиска решения в базе знаний типа семантической сети сводится к задаче поиска фрагмента сети, соответствующего некоторой подсети, соответствующей поставленному вопросу.

2. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ПРОВЕДЕНИЮ ПРАКТИЧЕСКИХ ЗАНЯТИЙ

2.1 Практическое занятие №1 (2 часа).
Тема: «Знания и данные»

2.1.1 Краткое описание проводимого занятия:

Существует множество способов определять понятия. Один из широко применяемых способов основан на идее интенсионала. Интенсионал понятия - это определение через понятие более высокого уровня абстракции с указанием специфических свойств. Этот способ определяет знания. Другой способ определяет понятие через перечисление понятий более низкого уровня иерархии или фактов, относящихся к определяемому. Это есть определение через данные, или экстенсионал понятия.

Пример. Понятие "персональный компьютер". Его интенсионал: "Персональный компьютер - это дружественная ЭВМ, которую можно поставить на стол и купить менее чем за $2000". Экстенсионал этого понятия: "Персональный компьютер - это Mac, IBM PC, Sinkler...".

Существуют десятки моделей (или языков) представления знаний для различных предметных областей. Большинство из них может быть сведено к следующим классам:

- формальные логические модели;
- продукционные;
- семантические сети;
- фреймы.

2.2 Практическое занятие №2 (2 часа).
Тема: «Правила-продукции»

2.2.1 Краткое описание проводимого занятия:

Объекты группируют по классам, Петр, Джон, Фред и Анна могут мыслиться как объекты. Их можно отнести к классу “личность”. В дополнение к этому Петр, Джон и Фред могут быть классифицированы как “мужчины”, а Анна- как "женщина". Явное достоинство любой классификации заключается в том, что частично решается проблема переполнения памяти, так как достаточно помнить только характеристики класса, а не каждого объекта. Мы можем также определить отношения между классами (или отдельными объектами). Подобным образом мы можем определить отношение “руководит (A, B)”, означающее, что B находится в подчинении у A, B качестве примеров такой зависимости могут служить выражения:

руководит (Петр, Джон)

руководит (Джон, Анна)

руководит (Анна, Фред)

которые заключают в себе структуру “подотчетность” (другое отношение) между объектами Петр-Джон-Анна-Фред, Приведенный пример иллюстрирует отношения между схожими объектами. Между различающимися объектами также могут быть установлены отношения (например, “владеет (Петр, автомобиль)”). Знания об объектах и их взаимоотношениях позволяют классифицировать эти объекты и соотносить между собой.

Второй тип знания- правила. Они дают возможность определить, как вывести новые отличительные особенности класса или отношения для объектов, прежде не подразделенных на классы. Например, если мы определим отношение “отчитывается (B, A)” для того, чтобы отметить, что B подотчетен A (возможно, через других руководителей), то сможем установить правила:

“отчитывается (C, A)” есть ИСТИНА

ЕСЛИ или “руководит (A, C)” есть ИСТИНА

ИЛИ “руководит (A. B)” И “руководит (B, C)" есть ИСТИНА

Это довольно ограниченное правило. Оно применимо только для первого или второго уровня подотчетности, но в пределах ограничения оно дает нам возможность породить новый пример отношения “отчитывается”, который ранее не был известен. Например, правило позволяет сделать заключение о том, что “отчитывается (Анна, Петр)” и “отчитывается (Фред, Джон)” суть ИСТИНА. Вследствие того, что данное правило определено как двухуровневое, оно не может быть применено для получения вывода “отчитывается (Фред, Петр)” есть ИСТИНА. Для этого нам потребуется более мощное правило, включающее рекурсию:

“отчитывается (C, A)” есть ИСТИНА

ЕСЛИ или “руководит (A, C)” есть ИСТИНА

ИЛИ “руководит (A, B)” есть ИСТИНА

И “отчитывается (C, B)” есть ИСТИНА

Первая часть приведенного рекурсивного правила относится к прямой подотчетности, а вторая - к косвенной.

2.3 Практическое занятие №3 (2 часа).

Тема: «Фреймы и объекты»

2.3.1 Краткое описание проводимого занятия:

Структура фрейма может выглядеть следующим образом:

Ту же запись можно представить в виде таблицы, содержащей дополнительные столбцы для описания типа слота и возможного присоединения к тому или иному слоту специальных процедур, что допускается в теории фреймов.

Имя фрейма

имя слота	тип слота	значение слота	присоединенная процедура

Внутреннее (машинное) представление фрейма имеет более сложную организацию и содержит средства для создания иерархии фреймов, их взаимодействия, обмена информацией, порождения конкретных фреймов из общих и общих фреймов из скелетных.

При конкретизации фрейма ему и его слотам присваиваются конкретные имена, а затем слоты заполняются значениями. Переход от исходного фрейма прототипа к фрейму-экземпляру может быть многошаговым (за счет постепенного уточнения значений слотов).

Имя фрейма — это идентификатор, присваиваемый фрейму, уникальный во всей фреймовой системе. Фрейм состоит из произвольного числа слотов, часть из которых определяется системой.

Имя слота — это идентификатор слота, уникальный в пределах фрейма. Во фреймовых системах иерархического типа, основанных на отношениях «абстрактное — конкретное» создаются указатели наследования, которые

показывают, какую информацию об атрибутах слотов верхнего уровня наследуют слоты с такими же именами во фреймах нижнего уровня.

Указатель атрибутов слота — это указатель типа данных слота. К таким типам относятся FRAME (указатель); INTEGER (целое); REAL (вещественное); BOOL (булево); LISP (присоединенная процедура); TEXT (текст); LIST (список); TABLE (таблица); EXPRESSION (выражение) и др.

Каждый слот предназначен для заполнения определенной структурой данных (в скелетном фрейме все они пусты, кроме первого, который имеет значение).

Значение слота — значение, соответствующее типу данных слота и удовлетворяющее условиям наследования. Значением слота может быть практически все, что угодно: числа или математические соотношения, тексты на естественном языке или программы, правила вывода или ссылки на другие слоты данного фрейма или других фреймов (сети фреймов).

У фреймовых моделей представления знаний нет специального механизма управления выводом и пользователь должен создавать его сам. Для этого используется три способа управления выводом: с помощью механизма наследования, демонов и присоединенных процедур.

Наследование свойств, заимствованное из теории семантических сетей, является важнейшим свойством теории фреймов. Во фреймах (как и в семантических сетях) наследование происходит поAKO-связям(A-Kind-Of = это). Слот AKO указывает на фрейм более высокого уровня иерархии, откуда неявно наследуются, т.е. переносятся значения аналогичных слотов.

Существуют несколько способов получения слотом значений во фрейме экземпляре:

- по умолчанию от фрейма образца (Default-значение);
- через наследование свойств от фрейма, указанного в слоте AKO;
- по формуле, указанной в слоте;
- через присоединенную процедуру;
- явно из диалога с пользователем;
- из баз данных.