**Л 6-2. Представление знаний в виде фреймов (дополнительная лекция 2)**

Фрейм, его структура и свойства. Вывод на фреймах. Достоинства и недостатки фреймового представления. Гибридные модели представления знаний.

Из предложенных ранее моделей представления знаний в виде системы продукций, семантических сетей, а еще ранее - в первой части - в виде формальных моделей мы видели, каким образом характер предметной области, типы задач для решения в СИИ влияют на выбор конкретной модели организации знаний и вывода результатов. Так, в продукционных моделях знания фокусируются на множестве характерных, или ключевых состояний, требующих принятия решения. Соответственно, модель предметной области или БЗ, в конце концов, представляется в виде множества правил типа:

<прототип состояния> → <решение + действия по изменению состояния ПрО, соответствующие данному решению>.

Каждое правило представляет собой элементарный кирпичик знания о целом. Очевидно, что для достаточно сложных предметных областей БЗ могут быть настолько сложными и запутанными, что их разработка и использование становятся проблематичным.

В семантических сетях - другой недостаток. Здесь фокус внимания сосредоточен на понятиях предметной области, отождествляемых с реальными объектами (сущностями, процессами). Однако представление в семантических сетях знаний недекларативного характера, например, процедурного, затруднено так же, как и информации о решениях. Поэтому естественно, что следующим началом в развитии СИИ стал поиск такого представления, который был бы, с одной стороны, достаточно крупным фрагментом знания о мире, а с другой - сохранял бы наиболее важные достоинства других представлений, такие как возможность использования процедурных знаний (как в продукционных моделях), наглядность и иерархичность (как в семантических сетях), возможность моделирования правдоподобного вывода (как в формальных системах). Поиски привели к модели представления знаний в виде фреймов.

**Фрейм, его структура и свойства**

Впервые понятие фрейма (frame - рамка, каркас, структура) было введено М. Минским в 1975 году. В основу его положены представления гештальтпсихологии, занимающейся изучением восприятия человеком внешнего мира в форме целостных фрагментов. Таким фрагментом может быть объект внешнего мира с его наиболее характерными свойствами, ситуация, процесс и т.п. **По Минскому**, фрейм - это структура данных, содержащая минимально необходимую информацию для представления класса объектов (явлений или процессов), которая однозначно определяет эти объекты.

**По определению Поспелова Д.А**., фрейм - это термин для обозначения описания какого-либо объекта или явления, обладающего тем свойством, что удаление из этого описания любой его части приводит к потере определяющих суть данного объекта или явления свойств.

**Еще одно определение**: фрейм - это структура данных для представления стереотипных ситуаций. Различают **фрейм-прототипы** (протофреймы) и **фрейм-примеры** (или фрейм-экземпляры).

**Фрейм-прототип** содержит знания, общие для всех частных случаев, т.е. примеров, а фрейм-пример содержит знания, отличающие частный случай от общего.

В наиболее простом случае фрейм представляется в следующем виде:

где - имя фрейма; - имя слота;- значение слота (); - имя присоединенной процедуры ().

В качестве **имен** фреймов могут фигурировать имена объектов, событий процессов и т.п. **Слотами** выступают характерные свойства или атрибуты описываемых объектов с именем .

В качестве **значений** слотов могут выступать:

• имена других фреймов,

• имена процедур,

• значения слотов могут задаваться массивом, матрицей, списком, любым типом данных.

Из одного фрейма можно сослаться на один или несколько других, образуя сложные сетевые структуры. Таким образом, существует возможность ссылки из одного фрейма в один или несколькодругих, что позволяет вводить структурированные связи между фреймами. При этом возникает однородная СС со сложными вершинами, допускающими дальнейшую развертку. Семантическую сеть можно рассматривать как частный случай фрейма, так как очевидно, что любая СС может быть переведена во фреймовое описание и наоборот. В случае рекурсивного вложения фреймов друг в друга порождается иерархическая система фреймов.

Рассмотрим пример, представленный на рис. 1 и описывающий в виде фреймов понятие ДЕЛОВАЯ ПОЕЗДКА.

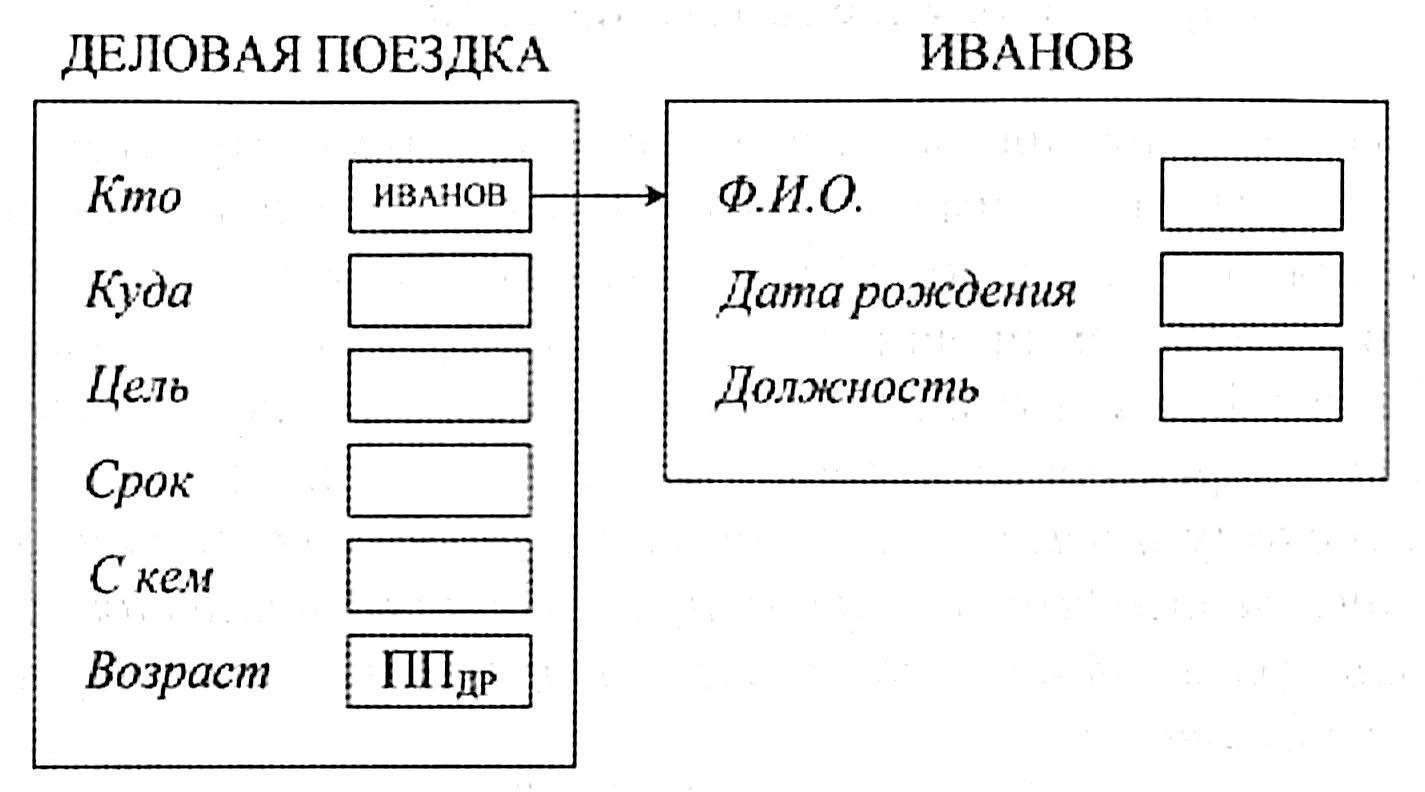


Рис.1. Структура иерархической системы фреймов, описывающих понятие ДЕЛОВАЯ ПОЕЗДКА

Очевидно, что на рис. 1 показаны не все вложения, но принцип ясен. В общем случае слотами могут быть незаполненные подструктуры фрейма, заполнение которых приводит к тому, что данному фрейму ставится в соответствие некоторая конкретная ситуация, объект или явление.

Фрейм с не полностью заполненными слотами соответствует протофрейму. Заполнение слотов формирует соответственно фрейм-пример.

Слоты могут быть **терминальными или нетерминальными**.

**Нетерминальные** слоты служат именами других фреймов, составляющих содержание данного слота. **Терминальные** слоты задают список соответствующих значений. Так, слоты КУДА, ЦЕЛЬ, СРОК и др. являются терминальными, а слот КТО - нетерминальный.

Фреймы типа ДЕЛОВАЯ ПОЕЗДКА называют еще ролевыми, т.к. в качестве слотов служат указатели ролей типа: кто, что, куда, когда, где, почему и т.п.

При этом выделяются слоты **обязательные и необязательные** для заполнения. Например, слот С КЕМ может быть необязательным. Значениями слотов могут быть также присоединенные процедуры (ПП). Необходимость в них возникает, когда активизация фрейма возможна лишь при выполнении некоторого предусловия. Проверка предусловия осуществляется при наличии ссылки на него во фрейме.

Например, слот ВОЗРАСТ заполняется в результате обращения к присоединенной процедуре (ППдр) и вычисляется в результате обращения к значению слота ДАТА РОЖДЕНИЯ соответствующего фрейма. **Примерами ПП** могут быть, например, процедуры заполнения слотов, активизации других фреймов, выполнения некоторых дополнительных действий, проверка условий, введение новых слотов или прототипов и т.д. Таким образом, слоты могут заполняться значениями, символами, множествами, а также управляющей информацией, благодаря которой фреймы и организуются в структуру.

Если сравнить фрейм с СС, то очевидно, что **протофрейм** соответствует **интенсиональному** описанию СС, а **фрейм-пример - экстенсиональному**. Отсюда следует основной механизм соотнесения фрейм-прототипов и фрейм-примеров. Извлекается фрейм-пример и сравнивается с множеством фрейм-прототипов. Если один из них «накладывается» на фрейм-прототип, то соответствующий образец активизируется. Условие перехода к другому фрейму может записываться как в декларативной, так и в процедурной формах. Можно выделить два типа процедур:

• проверяющие некоторые условия,

• разрешающие выполнять некоторые действия (например, вычисления).

Первый тип присоединенных процедур называют **процедурами-слугами**, вторые - **процедурами-демонами**. Последние активизируются при каждой попытке добавления или удаления данных из слота (по умолчанию). Очевидно, что формирование множества присоединенных процедур (ПП) связано с поддержкой основных типов процессов обработки фреймовых структур. Используются следующие основные типы процессов:

1) создание нового экземпляра фрейма;

2) активизация фреймов;

3) организация вывода путем последовательного поиска и активизации сети фреймов до нахождения наиболее соответствующего и построения на его основе экземпляра фрейма.

ПП могут также реализовывать эвристики, направленные на поиск необходимой для заполнения слотов информации.

**Вывод на фреймах**

Для организации процесса вывода используются механизмы наследования информации и присоединенных процедур. Наиболее системная структура фрейма, удобная для описания, приведена на рис. 2.

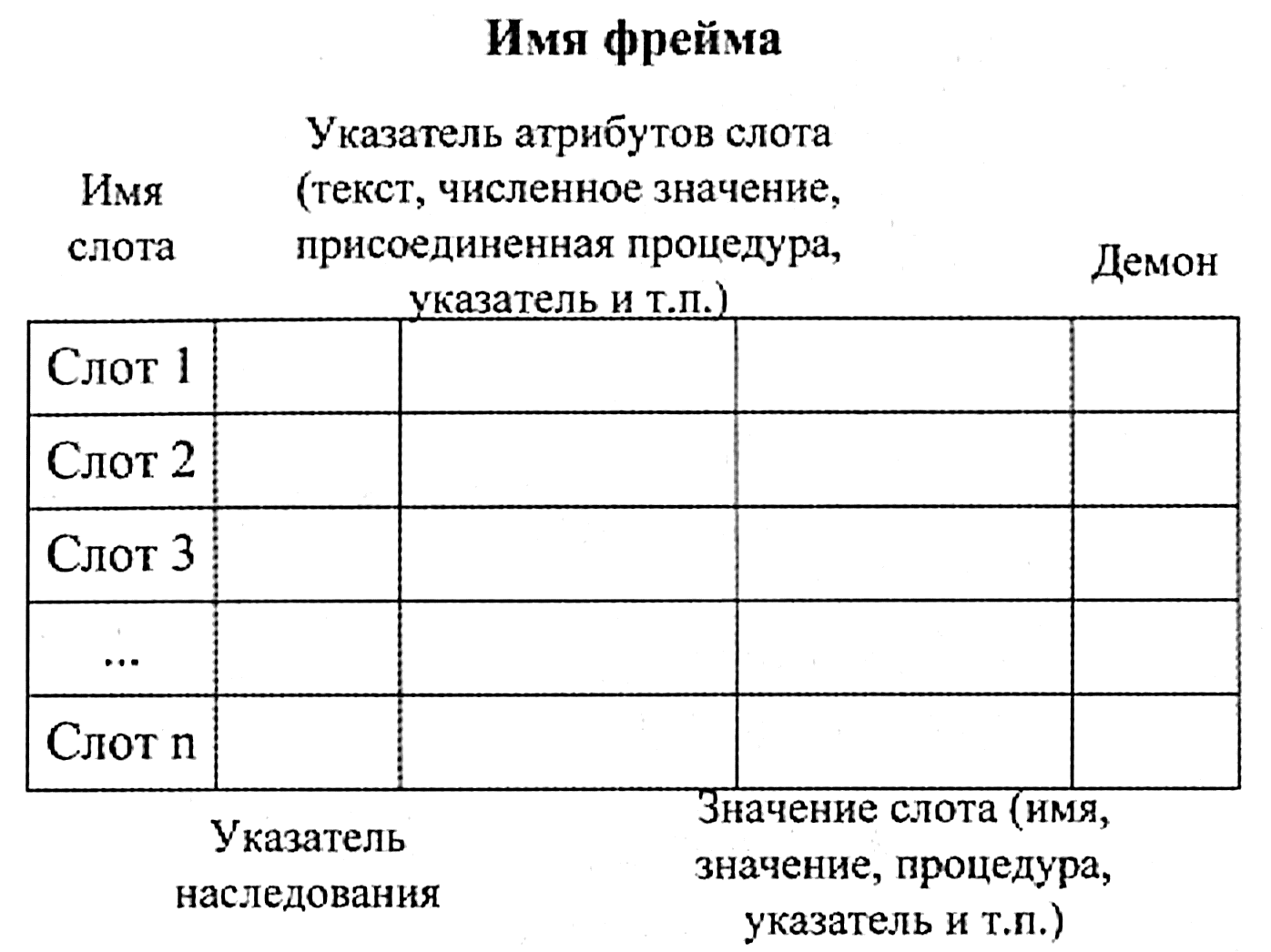


Рис. 2. Структура данных слота

Здесь:

1) *имя фрейма* - уникальный идентификатор. Для каждого фрейма определяется слот IsA - родитель данного фрейма - и слот дочерних фреймов, задаваемый списком имен (указателей). В список слотов включается ряд слотов, необходимых для организации интерфейса с пользователем, для организации внутрисистемных процессов и т.п. В их число входят, например, слоты для определения имени пользователя, даты определения, даты изменения текста, комментария и т.п.;

2) *имя слота* также служит его идентификатором, т.е. уникальным именем во фрейме, к которому он принадлежит. Могут вводиться специальные идентификаторы, отождествляемые с отношениями IsA, Part-of, причина-следствие и др. Кстати, в качестве слота может выступать слот «Условие», а его значением может быть продукционное правило. Таким образом, реализуется возможность использования продукций для организации вывода. Вводятся также слоты типа: «Дата модификации фрейма», «Комментарий», «Отношение» и др. Такие слоты называются системными и используются для редактирования БЗ и управления выводом;

3) указатели наследования необходимы только для фреймовых систем иерархического типа. Типичными указателями могут служить: «уникальный» (*unique: U*), «игнорировать» (*override: О*), «такой же» (*same: S*), «установить» (*range: R*) и т.п. Метка *U* показывает, что каждый фрейм может иметь слоты с различными значениями; *S* - все слоты должны иметь одинаковое значение; *R* - значения слотов фрейма нижнего уровня должны находиться в пределах, указанных значениями слотов фреймов верхнего уровня; *О* - при отсутствии указания значение слотов фрейма верхнего уровня становится значением слота фрейма нижнего уровня;

4) *указатель атрибутов слота* показывает тип данных - число, указатель другого фрейма, т.е. является именем (FRAME (указатель на другой фрейм), INTEGER (целый), REAL (действительный), BOOL (логический), TEXT (текстовый), LIST (список), TABLE (таблица), EXPRESSION (выражение) и др.);

5) *значение слота*, очевидно, должно совпадать с указанным типом данных этого слота, а также должно выполняться условие наследования;

6) *демон* - присоединенная процедура. Примеры таких ПП: If-needed, If-added, If-removed и т.д. Эти процедуры запускаются автоматически при выполнении некоторого условия. ПП-слуга - обычно программы процедурного типа, называемые служебными (в языке LISP), методами (в языке Smalltalk). Процедура запускается по сообщению из другого фрейма.

**Как таковой формальный механизм вывода отсутствует и реализуется с помощью механизма наследования и присоединенных процедур. Это позволяет объединять возможности сетевого и иерархического представления знаний.**

Из данного примера видно, насколько разнообразными могут быть конкретные реализации СИИ фреймового типа, а также их достоинства.

Приведем пример записи иерархии фреймов по отношению ISA с указанием фрейма более высокого уровня, из которого наследуются свойства (рис. 3).

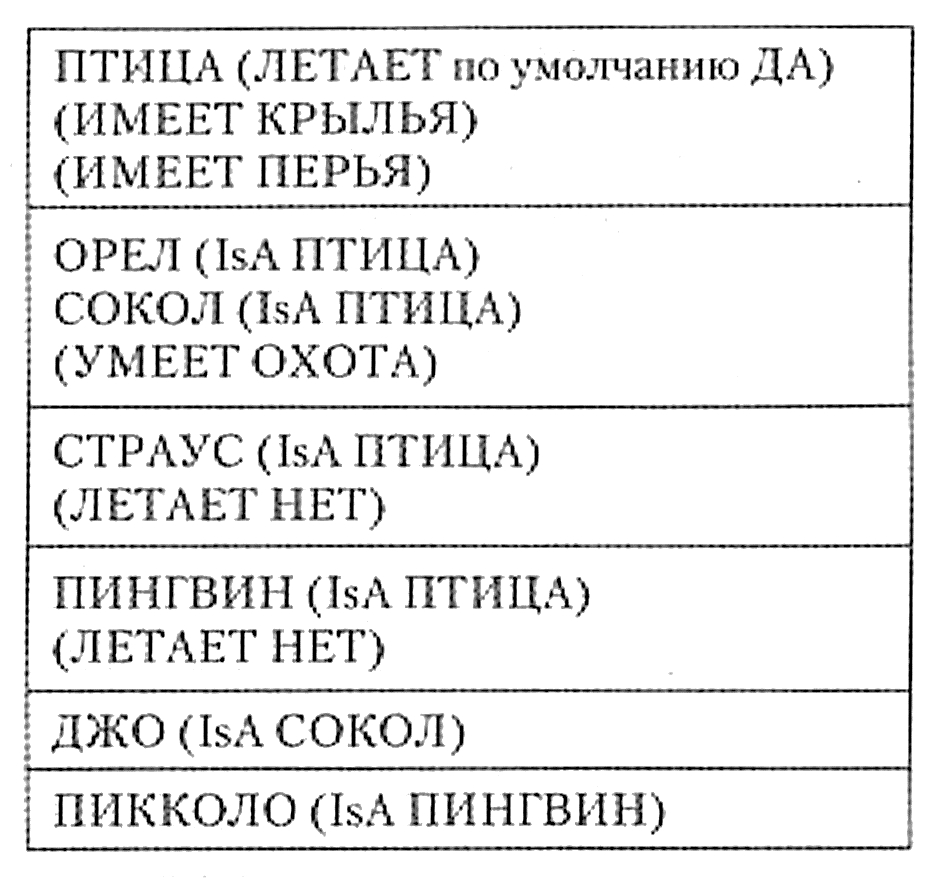


Рис. 3. Структура данных слота

Значение слота ЛЕТАЕТ по умолчанию означает, что если нет специальной оговорки, все подмножества ПТИЦ и отдельные ПТИЦЫ - летают.

Остальные свойства (ИМЕЕТ КРЫЛЬЯ), (ИМЕЕТ ПЕРЬЯ) наследуются всеми элементами, связанными отношением IsA.

Функция значений «по умолчанию» очень популярна в представлении фреймов, поскольку позволяет однозначно определять значения слотов на иерархии.

Например, если задать вопрос к базе фреймов: способен ли к охоте Джо? Система обращается к фрейму СОКОЛ и выясняет, что сокол умеет охотиться. То есть это свойство наследуется из фрейма более высокого уровня. Если мы хотим получить ответ на вопрос: летает ли ПИККОЛО, то системе приходится разрешать коллизию, возникающую при анализе фреймов ПТИЦА (ПИНГВИН есть ПТИЦА) и ПИНГВИН. Свойство ЛЕТАТЬ не определено во фрейме ПИККОЛО, поэтому система обращается к фрейму ПИНГВИН по указателю IsA. И здесь есть прямое указание, что пингвин не летает. Это свойство наследуется во фрейме ПИККОЛО. И хотя указатель нас отправляет к фрейму ПТИЦА, где по умолчанию стоит значение ДА, предпочтение отдается значению в слоте ПИНГВИН, откуда была последняя ссылка.

Часто языки фреймового типа называют объектно-ориентированными. Таким языком является Smalltalk. Другими примерами языков программирования фреймового типа являются языки FML (Frame Manipulation Language), FRL (Frame Representation Language), KRL (Knowledge Representation Language), являющиеся надстройками над LISP-системами. Очень часто такие языки также называют объектно-ориентированными, но, для предотвращения путаницы с языком Smalltalk, обычно сохраняют название «язык фреймового типа».

**Достоинства и недостатки фреймового представления**

К **достоинствам** фреймового представления знаний следует отнести то, что оно:

1) обеспечивает эффективную реализацию процедур вывода;

2) имеет возможность логических скачков, т.е. немонотонного вывода;

3) обеспечивает возможность образования СС фреймов, что дает большую экономию памяти при представлении информации за счет наследования свойств фреймов более высоких уровней во фреймах более низких уровней;

4) обеспечивает хорошее соответствие реальной действительности;

5) позволяет комбинировать различные модели представления знаний, объединяя их достоинства и компенсируя их недостатки.

К **недостаткам** отнесем:

1) каждый фрейм представляет собой достаточно сложный фрагмент знаний. Поэтому удаление или включение нового фрейма - весьма болезненная процедура, так как должна предусматривать и удаление всех составляющих элементов, которые могут быть составными частями других фреймов;

2) достаточно сложно осуществлять на фреймах представление временных процессов;

3) отсутствует формальная теория вывода на фреймах. Поэтому на инженере по знаниям целиком лежит ответственность за корректность организации иерархии фреймов и их заполнения.

**Гибридные модели представления знаний**

При проектировании баз знаний сложных предметных областей, в которых необходимо представление знаний всех известных типов - иерархического, декларативного, процедурного, разрабатываются гибридные модели, сочетающие продукционную, семантическую и фреймовую модели.

Наиболее популярными являются следующие **гибридные модели**: семантико-продукционная, семантико-фреймовая, фреймо-продукционная, семантико-фреймо-продукционная. В таких моделях разработчики стремятся компенсировать недостатки моделей одного типа достоинствами других. Например, недостатком продукционной модели является неэффективность вывода и нечитабельность (т.е. трудность понимания того, что делает система в процессе вывода) при больших БЗ. Этот недостаток хорошо компенсируется фреймовой моделью, когда декларативные знания представляются в виде фреймов иерархического типа, а процедурные знания - в виде слотов обращения к присоединенным процедурам. В итоге формируется иерархия, вершинами которой являются фреймы, моделирующие левую часть правил-продукций, а правая часть правил моделируется слотами - присоединенными процедурами. Если знания имеют слабую иерархичность или вложенность, то удобнее семантико-фреймовая модель, в которой вершины задаются фреймами, а связи между ними описываются семантической сетью.

Во фреймо-продукционной модели база знаний состоит из продукций, задающих причинно-следственные отношения между простыми и сложными объектами (любыми сущностями) моделируемой предметной области. При этом в качестве сложных объектов выступают фреймы. На основе объектов-условий определяется значение выделенного объекта-цели, играющего роль одной из гипотез решения. Вывод истинной (т.е. наилучшей с точки зрения имеющейся БЗ) гипотезы-решения может осуществляться как по прямой, так и по обратной стратегиям. При прямом выводе обход дерева условий выполняется от корня до соответствующей заданным условиям гипотезы. При обратном выводе выполняется проверка истинности условий, при которых предполагаемая гипотеза оказывается истинной. Возможно применение и смешанной стратегии вывода.