Модели представления знаний

Классификация по степени использования различных видов знаний

- В зависимости от того, какие виды знаний используются, и каким образом это происходит, системы искусственного интеллекта можно классифицировать несколькими способами.

- 1. Если в программной системе используются только фактографические знания, то такую программную систему называют *базой данных* (БД) и в современных условиях обычно не считают содержащей искусственный интеллект.

- 2. Если в программной системе используются главным образом алгоритмические знания, то такую программную систему принято называть *пакетом прикладных программ*. Отнесение такого пакета программ к системе ИИ обычно явно указывается. Если пакет прикладных программ используется просто как библиотека процедур, то его не относят к искусственно-интеллектуальным системам. Если же в нем используются методы ИИ, например, если программу решения конкретной задачи строит не пользователь, а автоматический планировщик, то такой пакет прикладных программ может считаться системой с искусственным интеллектом.

- 3. Если в программной системе в той или иной форме используются *концептуальные знания*, то такую систему считают искусственно-интеллектуальной.

- **Классификация по виду ответа при решении задач**
- Решая конкретную задачу, прикладная система ИИ получает на входе знания в той или иной форме, а на выходе выдает ответ, который также имеет некоторую форму и представляет собой некоторое (новое) знание. В соответствии с введенной классификацией видов знаний искусственно-интеллектуальные системы можно классифицировать по уровню выдаваемого ответа.
- 0. **Логический ответ** (да или нет); не очень удобны в использовании, сейчас применяются редко.
- 1. **Фактографический ответ** (ответ-факт); если выдается конкретный ответ на один вопрос, то такие системы часто называют *информационными системами.* В настоящее время наиболее распространенный класс систем.

- **2.Процедурный ответ**; решая задачу, система может создать и запустить процедуру (*система синтеза программ, автоматическое программирование*). Очень интересное и перспективное направление, привлекающее внимание большого числа исследователей.
- 3. **Понятийный ответ**; ответ-закон, строится не решение, а схема решения класса задач, может быть даже на компьютере не выполнимая. В настоящее время полномасштабные реализации пока неизвестны.

- Приведем примеры ответов разных уровней, используя модельный пример с сортировкой чисел из параграфа 0 (напомним, задача состоит в следующем: мы хотим, чтобы компьютер отсортировал набор из трех чисел в порядке их возрастания).
- (0) 2 5 7 → да
- 5 2 7 → нет
- (1) 5 2 7 → 2 5 7
- (2) $\{x, y, z\}?\{x < y < z\}$ → Sort3
- (3) $\{x_i\}$ **отсортировать** □ $\forall i, j\ (i < j \rightarrow x_i < x_j)$

- На уровне 0 мы предъявляем последовательность, а система только проверяет, отсортирована она или нет. На уровне 1 мы предъявляем последовательность, и система ее послушно сортирует. На уровне 2 система строит процедуру сортировки, примерно так, как это показано в параграфе 0. Наконец, на уровне 3 гипотетическая система объясняет нам, что это значит — отсортировать массив чисел

ЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

В *логических моделях* знания представляются в виде совокупности правильно построенных формул какой-либо *формальной системы* (ФС), которая задается четверкой:

$$ФС = \{T,P,A,R\},$$

где T — множество базовых (терминальных) элементов, из которых формируются все выражения ФС;

ЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

P — множество синтаксических правил, определяющих синтаксически правильные выражения из терминальных элементов ФС;

A — множество аксиом ФС, соответствующих синтаксически правильным выражениям, которые в рамках данной ФС априорно считаются истинными;

R — конечное множество правил вывода, позволяющих получать из одних синтаксически правильных выражений другие.

ЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

Простейшей логической моделью является *исчисление высказываний.*

Высказыванием называется предложение, смысл которого можно выразить значениями: истина (T) или ложь (F). Например, предложения «лебедь белый» и «лебедь чёрный» будут вы сказываниями.

Из простых высказываний можно составить более сложные:

- «лебедь белый или лебедь черный»,
- «лебеду белый и лебедь черный»,
- «если лебедь небелый, то лебедь черный».

ЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

Логика высказываний оперирует логическими связями между высказываниями, т. е. она решает вопросы типа: «Можно ли на основе высказывания *A* получить высказывание B»; «Истинно ли *B* при истинности A?» и т.п.

Элементарные высказывания рассматриваются как переменные логического типа, над которыми разрешены следующие логические операции:

- ¬ отрицание (унарная операция);

ЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

- ^ конъюнкция (логическое умножение);
- v дизъюнкция (логическое сложение);
- → импликация (если - то);
- ↔ эквивалентность.

ЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

A	$\neg A$	B	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$A \leftrightarrow B$
T	F	T	T	T	T	T
T	F	F	F	T	F	F
F	T	T	F	T	T	F
F	T	F	F	F	T	T

ЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

Исчисление высказываний позволяет формализовать лишь малую часть множества рассуждений, поскольку этот аппарат не позволяет учитывать внутреннюю структуру высказывания, которая существует в естественных языках.

ЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

В логике высказываний для обозначения фактов используются буквы(имена или идентификаторы или фразы), не имеющие структуры (используемые как атомарные объекты), и принимающие значения "1" или "0" ("да" или "нет"). То, что фразы имеют атомарный характер, не позволяет обнаружить похожесть их смысла. Например, высказывания "расстояние от Земли до Солнца – 150 млрд. км" и "расстояние от Земли до Марса – 60 млн. км" имеют похожий смысл, но абсолютно разные в логике высказываний.

ЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

Рассмотрим ставший классическим пример рассуждения о Сократе:

- *P: «Все люди смертны»*
- *Q: «Сократ — человек»*
- *R : «Сократ - смертен»*

Используя для обозначения высказываний логические переменные *P, Q, R* можно составить формулу: *(P ^ Q) →R* , которая может быть интерпретирована как «Если все люди смертны и Сократ является человеком, то Сократ является смертным».

ЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

Однако эта формула не является общезначимой, поскольку относится только к одному объекту (Сократу). Кроме того, высказывание *R* не выводится из *P* и *Q,* т.е., если бы мы не сформулировали *R* заранее, мы не смогли бы записать приведенную выше формулу.

ЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

- Чтобы осуществить этот примитивный логический вывод, высказывание *Q* следует разделить на две части: «Сократ» (субъект) и «человек» (свойство субъекта) и представить в виде отношения «субъект — свойство», которое можно записать с помощью функции *человек* (Сократ).

ЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

- Очевидно, что свойство конкретного субъекта с именем «Сократ» быть «человеком» может быть присуще и ряду других субъектов, что позволяет заменить константу «Сократ» на некоторую переменную, например *X*. Тогда получим запись *человек* (X), которая обладает внутренней структурой, т.е. значение такого высказывания будет зависеть от его компонент. Записанная функция уже не является элементарным высказыванием, она называется предикатом.

ЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

- «Под предикатом будем понимать некоторую связь, которая задана на наборе из констант или переменных.
- Пример предиката: *«P* больше Q».

Если семантика *P и Q* не задана, то о предикате сказать особенно нечего. Но при задании семантики (т.е. областей определения переменных *P* и *Q) о* предикате можно будет сказать существенно больше.

ЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

Например, если *P* и *Q* - площади городов в России и Японии, то при задании списков городов и подстановке значений из этих списков в переменные *мы* получим отношение между двумя сущностями и сможем судить о его истинности, например:

- *«Площадь Волгограда больше площади Хиросимы» = T.*
- *«Площадь Вологды больше площади Токио»=F.*

ЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

В логике предикатов факты обозначаются *n*-арными логическими функциями – предикатами $F(x_1,x_2,...,x_m)$, где F – имя предиката (функтор) и x_i – аргументы предиката. Имена предикатов неделимы, т. е. являются так называемыми атомами.

ЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

Аргументы могут быть атомами или функциями $f(x_1,x_2,...,x_m)$, где f – имя функции, а $x_1,...,x_m$, так же как и аргументы предикатов являются переменными или константами предметной области. В результате интерпретации предиката функторы и аргументы принимают значения констант из предметной области (строк, чисел, структур и т.д.).

ЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

Выше приведенные примеры высказываний в виде предикатов будут выглядеть как "расстояние(Земля, Солнце, 150000000000)" и "расстояние (Земля, Марс, 60000000)". Так как они имеют определенную структуру, их можно сравнивать по частям, моделируя работу с содержащимся в них смыслом.

ЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

Предикат с арностью $n > 1$ может используется в инженерии знаний для представления *n*-арного отношения, связывающего между собой n сущностей (объектов) – аргументов предиката. Например, предикат отец("Иван", "Петр Иванович") может означать, что сущности "Иван" и "Петр Иванович" связаны родственным отношением, а именно, последний является отцом Ивана или наоборот.

ЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

Предикат "компьютер(память, клавиатура, процессор, монитор)" может обозначать понятие "компьютер" как отношение, связывающее между собой составные части компьютера, предикат "внутри (процессор_Pentium, компьютер)" – то, что внутри компьютера находится процессор Pentium.

ЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

- Предикат с арностью *n* = 1 может представлять свойство сущности (объекта), обозначенного аргументом или характеристику объекта, обозначенного именем предиката. Например, кирпичный(дом), оценка(5), улица("Красный проспект"), дата_рождения("1 апреля 1965 г."), быстродействие("1 Мфлопс").
- Предикат с арностью *n* = 0 (без аргументов) может обозначать событие, признак или свойство, относящееся ко всей предметной области. Например, "конец работы".

ЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

При записи формул (выражений) помимо логических связок "конъюнкция" (&), "дизъюнкция" ($\vee$), "отрицание" ($\neg$), "следование" ("импликация") ($\rightarrow$) , заимствованных из логики высказываний , в логике предикатов используются кванторы всеобщности ($\forall$) и существования ($\exists$). Например, выражение $\forall(x,y,z)$ (отец(x,y) & (мать(x,z)) -> родители(x,y,z) означает, что для всех значений x,y,z из предметной области справедливо утверждение "если y – отец и z – мать x, то y и z – родители x; выражение ($\exists x$) (студент(x) & должность(x,"инженер") означает, что существует хотя бы один студент, который работает в должности инженера.

ЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

Переменные, находящиеся в сфере действия кванторов, называют связанными, остальные переменные в логических формулах называются свободными. Для того чтобы можно было говорить об истинности какого-либо утверждения без подстановки значений в переменные, все входящие в него переменные должны быть связаны кванторами.

ЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

К недостаткам логики предикатов как метода представления знаний можно отнести следующее:

- монотонность логического вывода, т.е. невозможность пересмотра полученных промежуточных результатов (они считаются фактами, а не гипотезами);

ЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

- невозможность применения в качестве параметров предикатов других предикатов, т.е. невозможность формулирования знаний о знаниях;
- детерминированность логического вывода, т.е. отсутствие возможности оперирования с нечеткими знаниями.

Но логику предикатов можно использовать как основу для конструирования более сложных и удобных логических методов представления знаний.

Лекция 4

Фреймовая модель представления знаний

Фреймовая модель

Фреймовая модель представления знаний основана на теории фреймов М. Минского, которая представляет собой систематизированную психологическую модель памяти человека и его сознания. Эта теория имеет весьма абстрактный характер, поэтому только на ее основе невозможно создание конкретных языков представления знаний.

Фреймовая модель

В психологии и философии известно понятие абстрактного образа. Например, слово “ комната” вызывает у слушающих образ комнаты - жилое помещение с четырьмя стенами, полом, потолком, окнами и дверью. Из этого описания ничего нельзя убрать (например, убрав окна, мы получим уже чулан, а не комнату). Но в нем есть «дырки», или «слоты», - это незаполненные значения некоторых атрибутов - количество окон, цвет стен, высота потолка, покрытие пола и др. Такой образ и называется фреймом . Фреймом называется также и формализованная модель для отображения образа.

Фреймовая модель

- ПРИМЕР

Объект "мяч" представляет собой кожаный или резиновый чехол, наполненный воздухом. Во фрейме "мяч" можно обозначить слоты "радиус" (радиус оболочки мяча), "спорт" (вид спорта, для которого предназначен мяч), "накачан" (да/нет). Ситуация "лекция" может быть определена как "чтение лектором учебного материала слушателям". Фрейм "лекция" может содержать слоты "предмет" (предмет, по которому проводится лекция), "лектор" (ФИО лектора), "аудитория" (место проведения лекции), "слушатели" (количество слушателей).

Фреймовая модель

Фрейм имеет имя, служащее для идентификации описываемого им понятия, и содержит ряд описаний — *слотов,* с помощью которых определяются основные структурные элементы этого понятия. За слотами следуют *шпации,* в которые помещают данные, представляющие текущие значения слотов. Слот может содержать не только конкретное значение, но также имя процедуры, позволяющей вычислить это значение по заданному алгоритму.

Фреймовая модель

Например, слот с именем *возраст* может содержать имя процедуры, которая вычисляет возраст человека по дате рождения, записанной в другом слоте, и текущей дате. Процедуры, располагающиеся в слотах, называются связанными или присоединенными процедурами. Вызов связанной процедуры осуществляется при обращении к слоту, в котором она помещена.

Фреймовая модель

В слоте может содержаться не одно, а несколько значений, т. е. в качестве структурных составляющих фреймов могут использоваться данные сложных типов, а именно: массивы, списки, множества, фреймы и т. д. Например, в слоте с именем *брат* может содержаться список имен, если объект, описываемый данным фреймом, имеет нескольких братьев. Значение слота может представлять собой некоторый диапазон или перечень возможных значений, арифметическое выражение, фрагмент текста и т.д.

Фреймовая модель

Совокупность данных предметной области может быть представлена множеством взаимосвязанных фреймов, образующих единую фреймовую систему, в которой объединяются декларативные и процедурные знания. Такая система имеет, как правило, иерархическую структуру, в которой фреймы соединены друг с другом с помощью родо-видовых связей. На верхнем уровне иерархий находится фрейм, содержащий наиболее общую информацию, истинную для всех остальных фреймов.

Фреймовая модель

Фреймы обладают способностью наследовать значения характеристик своих родителей. Например, фрейм АФРИКАНСКИЙ_СЛОН наследует от фрейма СЛОН значение характеристики *цвет—«серый».* Значение характеристики в дочернем фрейме может отличаться от родительского, например, значением данного слота для фрейма АЗИАТСКИЙ_СЛОН является *цвет= «коричневый».*

Фреймовая модель

Существуют 2 типа фреймов – прототипы и экземпляры. Фреймы-прототипы — это готовые структуры для описания законов. В них отсутствуют конкретные значения слотов. При заполнении слотов конкретными значениями, они превращаются в конкретные фреймы. Часто в системах фреймы используются для стереотипны последовательностей действий и тогда они называются сушариями. При заполнении фреймов-прототипов, часть слотов может оставаться пустой. Фреймы-экземпляры создаются для отображения реальных ситуацийна основе поступающих данных.

Фреймовая модель

Над фреймами можно совершать некоторые теоретико-множественные операции, например объединение и пересечение. При объединении фреймов в результирующем фрейме будут присутствовать все слоты», которые встречались в исходных фреймах. В слотах, не являющихся общими, будут сохранены исходные значения. Если в объединяемых фреймах были одноименные слоты, в результирующем фрейме останется один слот с таким именем, значение его определится в результате объединения значений одноименных слотов.

Фреймовая модель

При пересечении фреймов в результирующем фрейме будут присутствовать только те слоты, которые имелись во всех исходных фреймах. Вычислить результирующие значения можно двумя способами. Первый способ состоит в том, что в результирующем фрейме присутствуют только те значения, которые совпадали в исходных фреймах. Во втором способе результирующие значения находят путем пересечения значений из исходных фреймов.

Пример фрейма “руководитель”

Имя слота	Значение	Тип значения слота
Имя слота	Иванов И. И.	Строка символов
Дата рождения	02.03.1972	дата
Возраст	age	процедура
специальность	юрист	Строка символов
Отдел	менеджмента	Строка символов
Зарплата	60000	число
Адрес	Дом_адрес	фрейм

Структура данных фрейма

Имя фрейма. Оно служит для идентификации фрейма в системе и должно быть уникальным. Фрейм представляет собой совокупность слотов, число которых может быть произвольным. Число слотов в каждом фрейме устанавливается проектировщиком системы, при этом часть слотов определяется самой системой для выполнения специфических функций (системные слоты), примерами которых являются: слот-указатель родителя данного фрейма, слот-указатель дочерних фреймов, слот для ввода имени пользователя, слот для ввода даты определения фрейма, слот для ввода даты изменения фрейма и т.д.

Структура данных фрейма

Имя слота. Оно должно быть уникальным в пределах фрейма. Обычно имя слота представляет собой идентификатор, который наделен определенной семантикой. В качестве имени слота может выступать произвольный текст.

Например, <Имя слота>= Главный герой романа Ф. М. Достоевского «Идиот», <Значение слота>= Князь Мышкин. Имена системных слотов обычно зарезервированы, в различных системах они могут иметь различные значения. Системные слоты служат для редактирования базы знаний и управления выводом во фреймовой системе.

Структура данных фрейма

Указатели наследования.

Они показывают, какую информацию об атрибутах слотов из фрейма верхнего уровня наследуют слоты с аналогичными именами в данном фрейме. Указатели наследования характерны для фреймовых систем иерархического типа, основанных на отношениях типа «абстрактное — конкретное».

В конкретных системах указатели наследования могут быть организованы различными способами и иметь разные обозначения:

Структура данных фрейма

- U (Unique) — значение слота не наследуется;
- S (Same) - значение слота наследуется;
- R (Range) - значения слота должны находиться в пределах интервала значений, указанных в одноименном слоте родительского фрейма;
- O (Override) — при отсутствии значения в текущем слоте оно наследуется из фрейма верхнего уровня, однако в случае определения значения текущего слота оно может быть уникальным. Этот тип указателя выполняет одновременно функции указателей U и S.

Структура данных фрейма

Указатель типа данных. Он показывает тип значения слота. Наиболее употребляемые *типы: frame* — указатель на фрейм; *real* — вещественное число; *integer* — целое число; *boolean* — логический тип; *text* — фрагмент текста; *list* — список; *table* — таблица; *expression* — выражение; *lisp* - связанная процедура и т.д.

Структура данных фрейма

Демоны.

Демоном называется процедура, автоматически запускаемая при выполнении некоторого условия. Демоны автоматически запускаются при обращении к соответствующему слоту. Типы демонов связаны с условием запуска процедуры.

Структура данных фрейма

Демон с условием *IF-NEEDED* запускается, если в момент обращения к слоту его значение не было установлено. Демон типа *IF-ADDED* запускается при попытке изменения значения слота. Демон *IP-REMOVED* запускается при попытке удаления значения слота. Возможны также другие типы демонов. Демон является разновидностью связанной процедуры.

Структура данных фрейма

Присоединенная процедура запускается по сообщению, переданному из другого фрейма. Демоны и присоединенные процедуры являются процедурными знаниями, объединенными вместе с декларативными в единую систему. Эти процедурные знания являются средствами управления выводом во фреймовых системах, причем с их помощью можно реализовать любой механизм вывода.

Структура фрейма «Научная конференция»

Фрейм «Научная конференция»

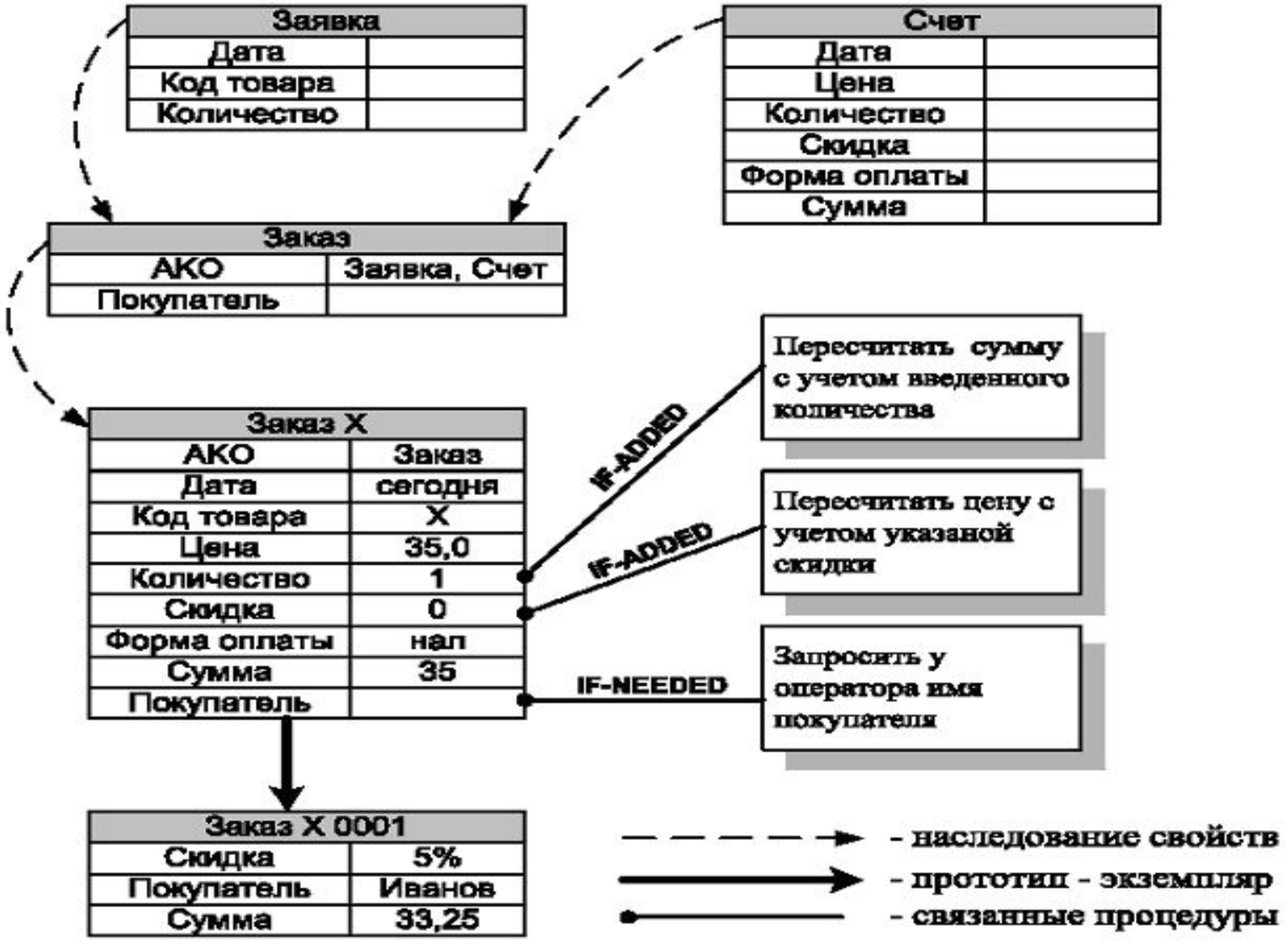
Имя слота	Значение слота	If-needed	If-added	If-removed
Дата	1.02. 10 : 10			
Место проведения	Аудитория 209		ЗАКАЗ	
Тема доклада	Прогнозирование тенденций в экономике			
Докладчик	Иванов И.И.	КТО?		

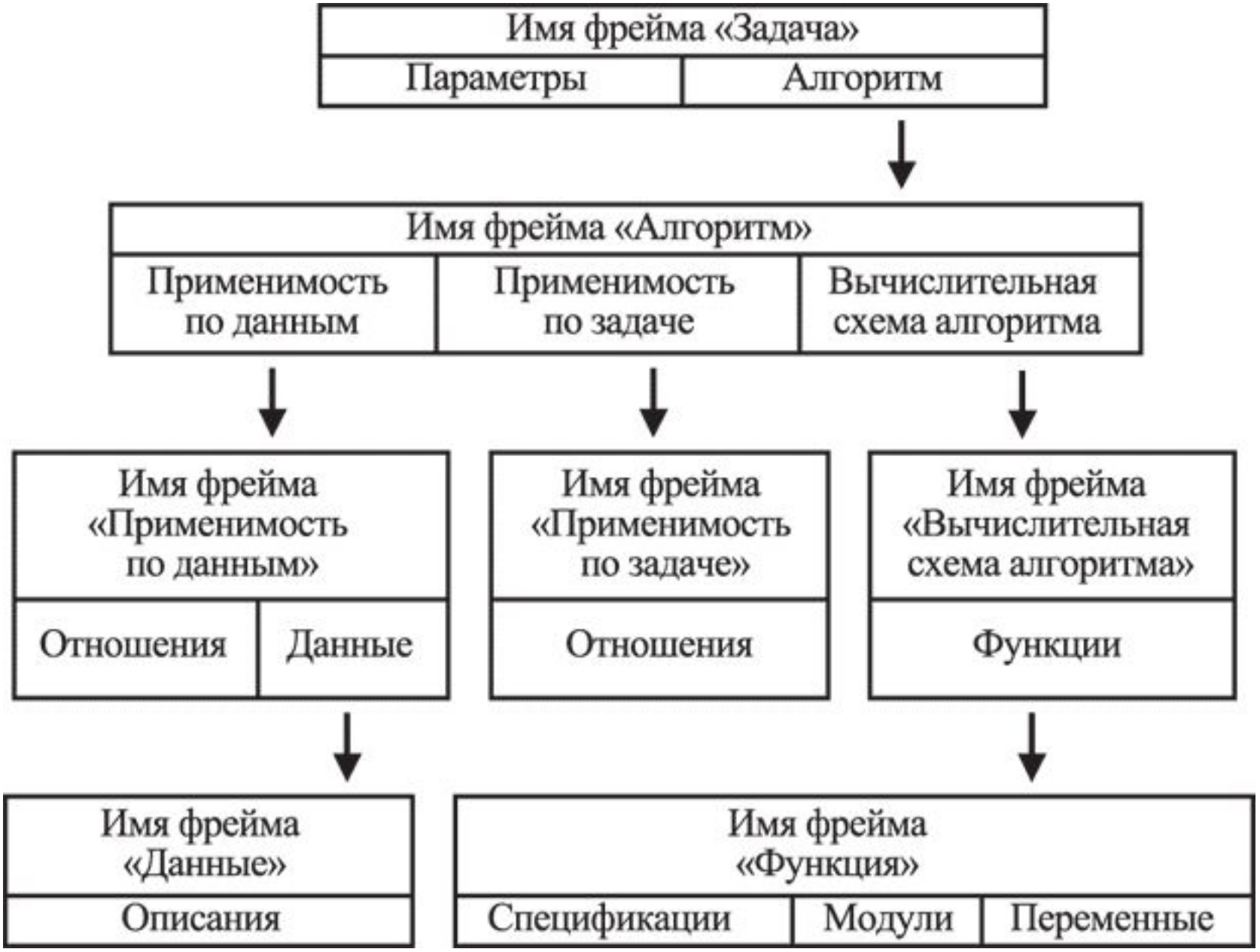
Структура фрейма «Научная конференция»

Демон *ЗАКАЗ* — это процедура, которая автоматически запускается при попытке подстановки значения в слот с именем *Место проведения.* Ее главное назначение состоит в проверке возможности заказа аудитории на нужное время.

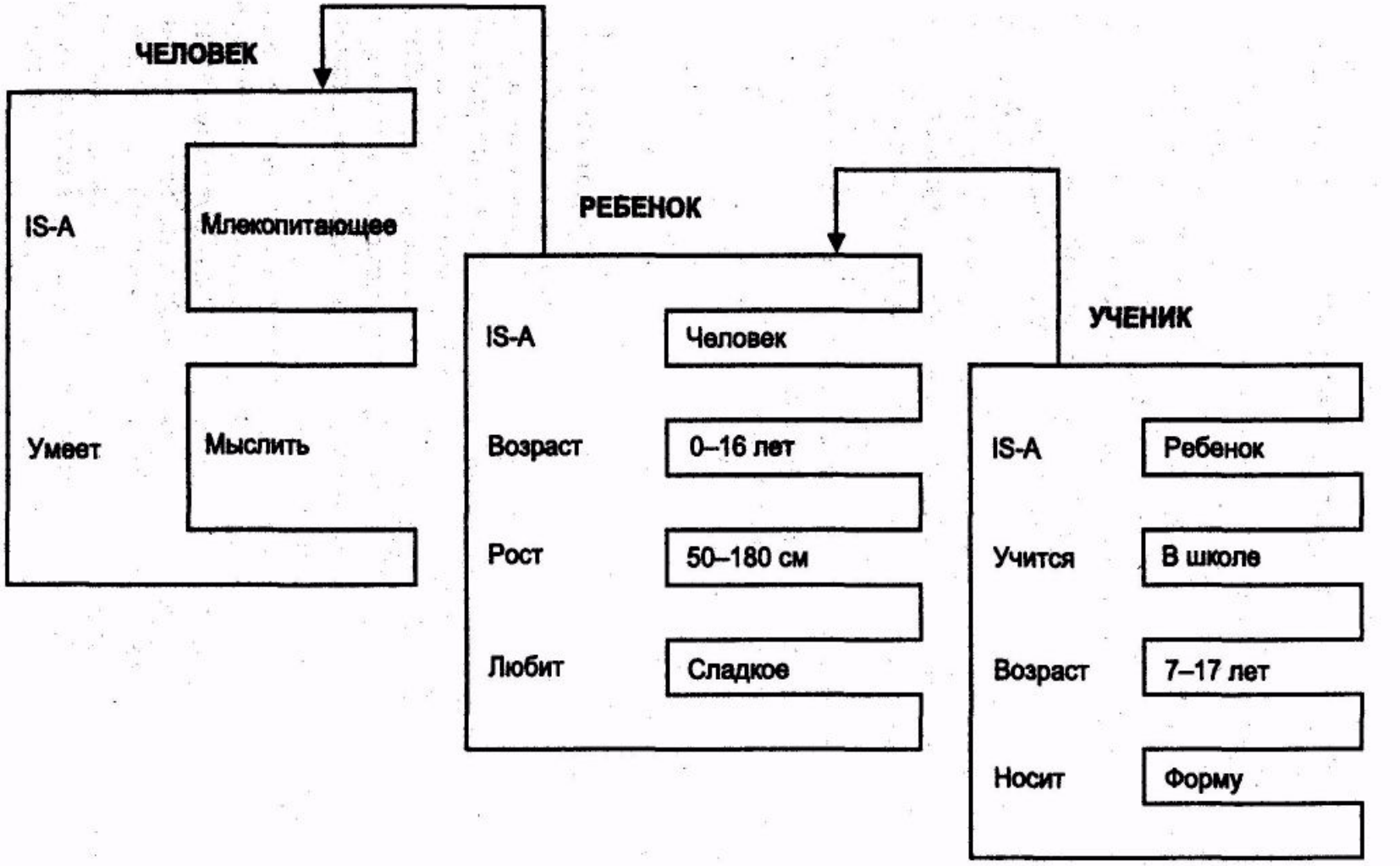
Структура фрейма «Научная конференция»

Демон *КТО?* автоматически запускается при обращении к слоту *Докладчик,* если значение этого слота не определено. Основное содержание данной процедуры — генерация запроса к пользователю типа «Кто выступает?», получение ответа и его запись в качестве значения слота.


Заявка
Дата
Код товара
Количество
Счет
Дата
Цена
Количество
Скидка
Форма оплаты
Сумма
Заказ
АКО
Заявка, Счет
Покупатель
Заказ X
АКО
Заказ
Дата
сегодня
Код товара
X
Цена
35,0
Количество
1
Скидка
0
Форма оплаты
нал
Сумма
35
Покупатель
IF-ADDED
IF-ADDED
IF-NEEDED
Пересчитать сумму с учетом введенного количества
Пересчитать цену с учетом указаной скидки
Запросить у оператора имя покупателя
Заказ X 0001
Скидка
5%
Покупатель
Иванов
Сумма
33,25
- наследование свойств
- прототип - экземпляр
- связанные процедуры


Имя фрейма «Задача»
Параметры
Алгоритм
Имя фрейма «Алгоритм»
Применимость по данным
Применимость по задаче
Вычислительная схема алгоритма
Имя фрейма «Применимость по данным»
Отношения
Данные
Имя фрейма «Применимость по задаче»
Отношения
Имя фрейма «Вычислительная схема алгоритма»
Функции
Имя фрейма «Данные»
Описания
Имя фрейма «Функция»
Спецификации
Модули
Переменные

Пример сети фреймов



Пример сети фреймов

Понятие УЧЕНИК наследует свойства фреймов РЕБЕНОК и ЧЕЛОВЕК, которые находятся на более высоких уровнях иерархии. Если будет задан вопрос «Любят ли ученики сладкое?», то следует ответ “да», так как этим свойством обладают все дети, что указано во фрейме РЕБЕНОК. Наследование свойств может быть частичным, например «возраст» для учеников не наследуется из фрейма «ребенок», так как явно указан в собственном фрейме.

Пример сети фреймов

Понятие УЧЕНИК наследует свойства фреймов РЕБЕНОК и ЧЕЛОВЕК, которые находятся на более высоких уровнях иерархии. Если будет задан вопрос «Любят ли ученики сладкое?», то следует ответ “да», так как этим свойством обладают все дети, что указано во фрейме РЕБЕНОК. Наследование свойств может быть частичным, например «возраст» для учеников не наследуется из фрейма «ребенок», так как явно указан в собственном фрейме.

Фреймовая модель

Модель фрейма является достаточно универсальной, поскольку

позволяет отобразить все многообразие знаний о мире через:

- фреймы - структуры для обозначений объектов и понятий;
- фреймы - роли;
- фреймы – сценарии;
- фреймы – ситуации.

Фреймовая модель

Пример.

Фрейм-структурами являются понятия "заем", "вексель","кредит".

Фрейм-роли - "кассир", "клиент", "сервер". Фрейм-сценарии -"страхование", "банкинг", "банкротство".

Фрейм-ситуации - "эволюция", "функционирование", "безработица".

Реализация фреймовой модели представления знаний на примере решения задачи Эйнштейна

Условия задачи следующие:

1. Есть пять домов, каждый разного цвета.
2. В каждом доме живет один человек, отличающий;

ся от соседнего по национальности: немец, англи;

чанин, швед, датчанин, норвежец.

3. Каждый пьет только один напиток, выращивает

определенное растение и держит определенное

животное.

4. Никто из 5 человек не пьет одинаковые с другими

напитки, не выращивает одинаковое растение и

Цель решения: определить, кому принадлежит

рыба?

При этом дополнительные условия включают следующее:

1. Англичанин живет в красном доме.
2. Швед держит собаку.
3. Датчанин пьет чай.
4. Зеленый дом стоит слева от белого.
5. Жилец зеленого дома пьет кофе.
6. Человек, который выращивает ячмень, держит

птицу.

7. Жилец из среднего дома пьет молоко.
8. Жилец из желтого дома выращивает томаты.

9.Норвежец живет в первом доме.
10.Тот, кто держит кошку, живет около того, кто
вы;
ращивает свеклу.
11.Человек, который содержит лошадь, живет
около
того, кто выращивает томаты.
12.Тот, кто выращивает пшеницу, пьет сок.
13.Норвежец живет около голубого дома.
14.Немец выращивает капусту.
15.Тот, кто выращивает свеклу, живет по
соседству с человеком, который пьет воду.

Если трактовать условие (4) как «зеленый дом

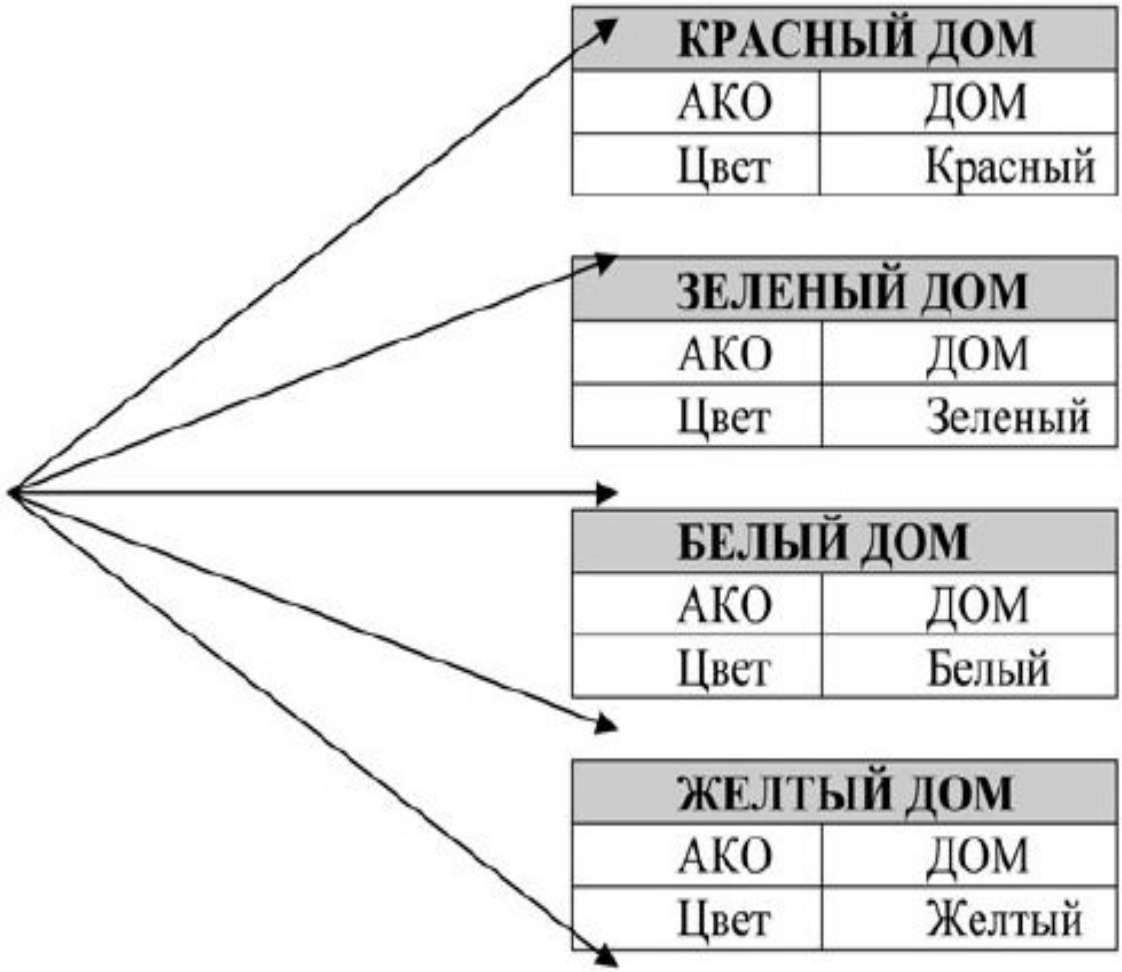
стоит **непосредственно слева от белого», то задача** имеет единственное решение. Причем под решением в данном случае понимается не только определение того, кому принадлежит рыба, но и полное распределение: дом – человек – напиток – растение – животное, охватывающее все объекты, описанные в условии.

Попробуем представить предметную область

задачи с помощью набора фреймов. В условии фигурируют объекты 5 классов: «дом», «человек», «напиток», «растение» и «животное». Каждому классу принадлежит 5 объектов. Таким образом, наиболее логичным решением в данном случае будет создание для каждого класса абстрактного фрейма, описывающего этот класс, а для каждого объекта, принадлежащего классу, создание конкретного фрейма,

- На рис. 1 представлены фреймовые диаграммы классов «Дом» и «Человек». Диаграммы классов «Напиток», «Растение» и «Животное» не показаны, поскольк представляющие их фреймы достаточно тривиальны (не содержат ни одного слота).
- Диаграмма показывает, что фреймы, представляющие конкретные объекты, наследуют слоты абстрактных фреймов, определяя значения для некоторых из них. Таким образом, часть информации, содержащейся в условии задачи, задается с помощью значений слотов. Данная информация носит декларативный характер, поскольку задается на этапе построения

ДОМ	
Цвет	
Позиция	
Жилец	Ссылка на фрейм <ЧЕЛОВЕК>



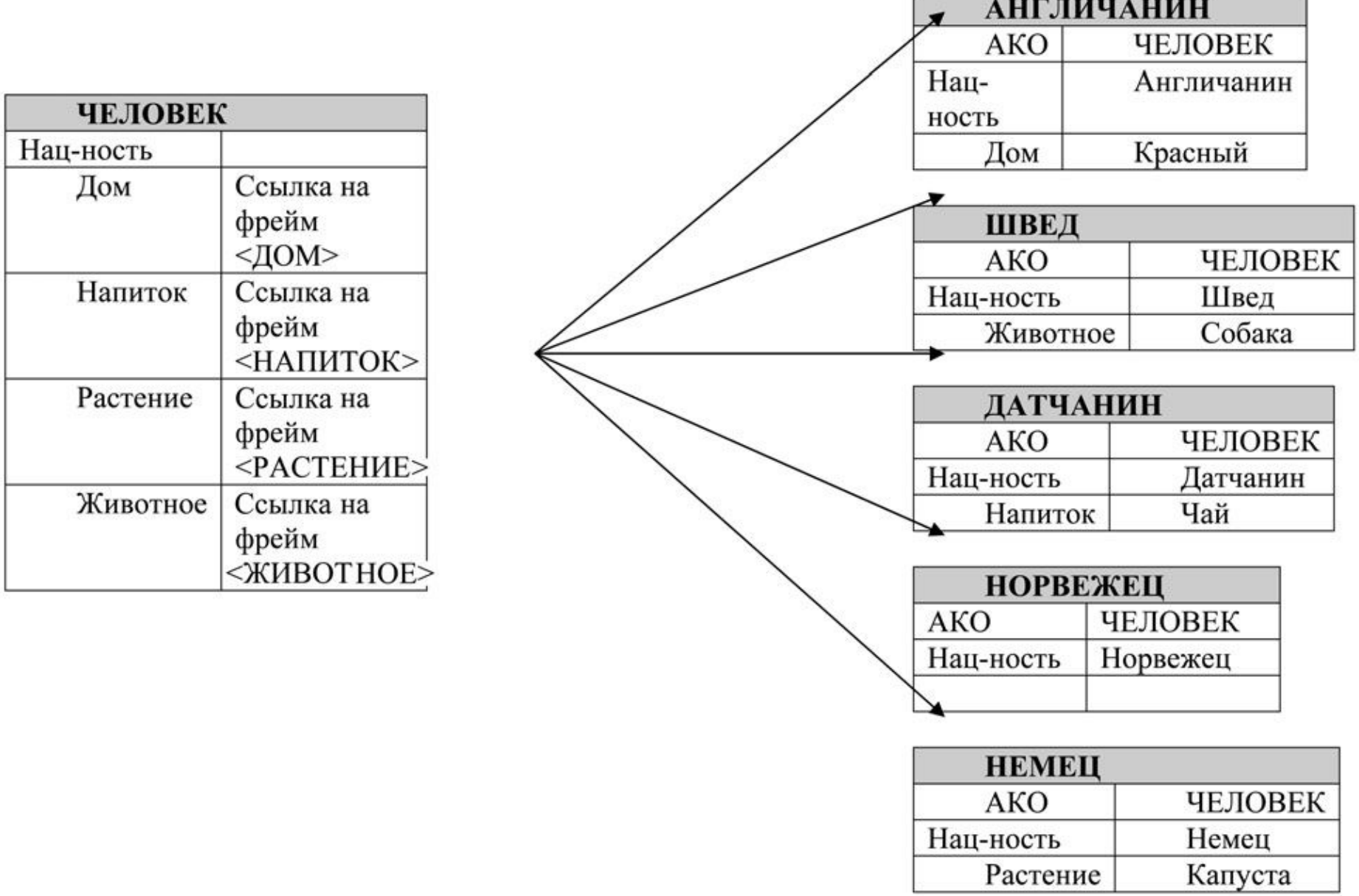
КРАСНЫЙ ДОМ	
АКО	ДОМ
Цвет	Красный

ЗЕЛЕНЫЙ ДОМ	
АКО	ДОМ
Цвет	Зеленый

БЕЛЫЙ ДОМ	
АКО	ДОМ
Цвет	Белый

ЖЕЛТЫЙ ДОМ	
АКО	ДОМ
Цвет	Желтый

ГОЛУБОЙ ДОМ	
АКО	ДОМ
Цвет	Голубой


ЧЕЛОВЕК
Нац-ность
Дом
Ссылка на фрейм <ДОМ>
Напиток
Ссылка на фрейм <НАПИТОК>
Растение
Ссылка на фрейм <РАСТЕНИЕ>
Животное
Ссылка на фрейм <ЖИВОТНОЕ>
АНГЛИЧАНИН
АКО
ЧЕЛОВЕК
Нац-ность
Англичанин
Дом
Красный
ШВЕД
АКО
ЧЕЛОВЕК
Нац-ность
Швед
Животное
Собака
ДАТЧАНИН
АКО
ЧЕЛОВЕК
Нац-ность
Датчанин
Напиток
Чай
НОРВЕЖЕЦ
АКО
ЧЕЛОВЕК
Нац-ность
Норвежец
НЕМЕЦ
АКО
ЧЕЛОВЕК
Нац-ность
Немец
Растение
Капуста

Другая часть информации может быть заложена в модель с помощью процедур, задаваемых для фрейма и вызываемых при изменении значения его слотов.

Эта информация носит характер проверяемых условий. Например, условие «жилец желтого дома выращивает томаты» может быть проверено процедурой, вызываемой при изменении значения слота «РАСТЕНИЕ» фрейма «ЖЕЛТЫЙ ДОМ».

Сложнее обстоит дело с условиями типа «тот, кто держит кошку, живет около того, кто выращивает свеклу». Для проверки таких условий каждый дом должен располагать информацией о соседних домах и их обитателях.

В программе, которая будет рассмотрена ниже, связь домов друг с другом реализована путем введения списка домов, учитывающего их расстановку, причем каждый «дом» имеет доступ к этому списку.

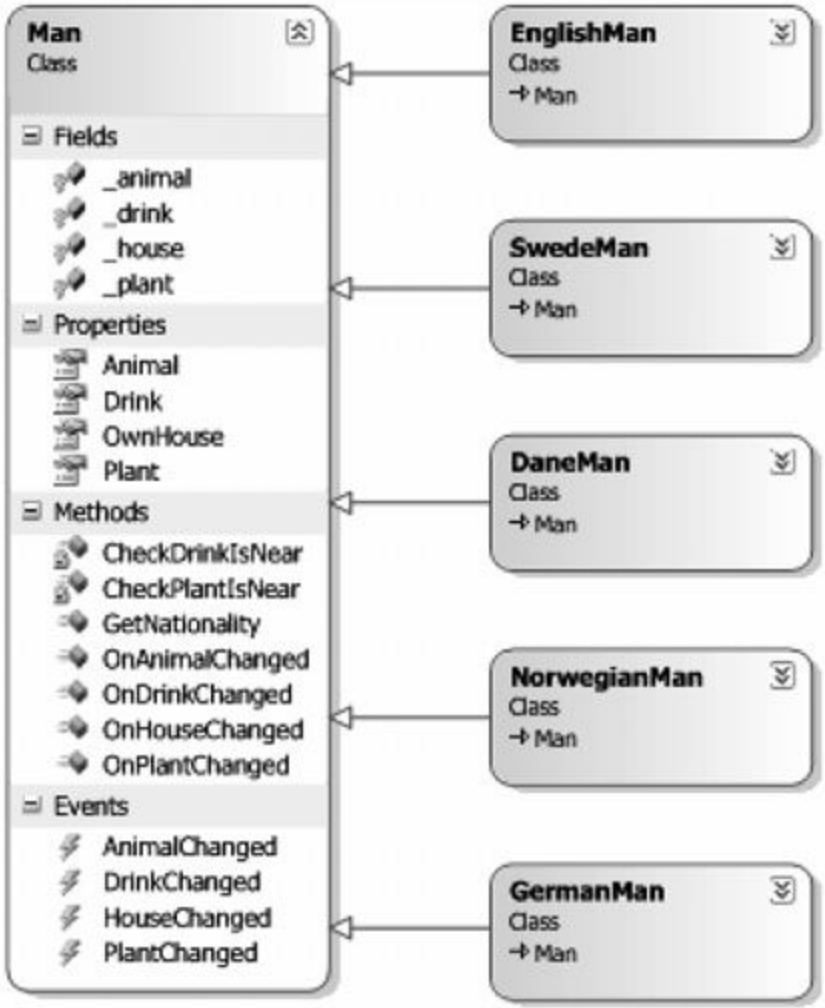
Фреймовая модель очень удобна с точки зрения программной реализации, поскольку она напрямую соответствует парадигме объектно-ориентированного программирования (ООП). С точки зрения ООП, фреймы представляют собой ничто иное, как классы.

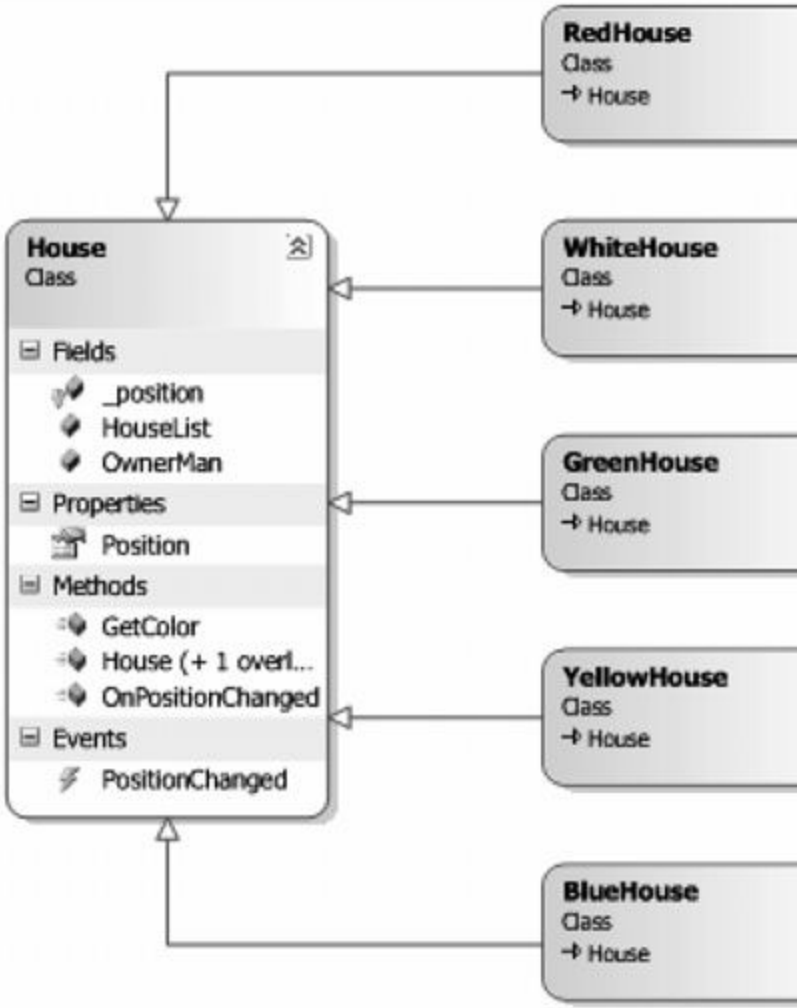
Слоты являются аналогом полей классов. Иерархия

фреймов соответствует иерархии классов, которая

строится с помощью механизма наследования. Про

На рис. 2 представлена диаграмма классов, реализующих фреймовую модель предметной области (далее просто «Модель»). Ключевыми являются классы «House» и «Man», от которых наследуются классы, описывающие уже конкретные
дома и конкретных людей. Программный модуль состоит из двух основных
частей: набора классов, реализующих фреймовую модель, и внешнего модуля, который осуществляет генерацию наборов данных, наблюдение за «реакци;
ей» модели и принятие необходимых решений.


Man
Class
Fields
_animal
_drink
_house
_plant
Properties
Animal
Drink
OwnHouse
Plant
Methods
CheckDrinkIsNear
CheckPlantIsNear
GetNationality
OnAnimalChanged
OnDrinkChanged
OnHouseChanged
OnPlantChanged
Events
AnimalChanged
DrinkChanged
HouseChanged
PlantChanged
EnglishMan
Class
Man
SwedeMan
Class
Man
DaneMan
Class
Man
NorwegianMan
Class
Man
GermanMan
Class
Man


House
Class
Fields
_position
HouseList
OwnerMan
Properties
Position
Methods
GetColor
House (+ 1 overl...
OnPositionChanged
Events
PositionChanged
RedHouse
Class
House
WhiteHouse
Class
House
GreenHouse
Class
House
YellowHouse
Class
House
BlueHouse
Class
House


drinks
Enum
unknown
tea
coffee
milk
plants
Enum
unknown
barley
tomato
beet
animals
Enum
unknown
cat
dog
poultry

- Вся предметная область разбита на пять уровней:
- 1) уровень домов → 2) уровень людей → 3) уровень напитков → 4) уровень растений → 5) уровень животных.
- Такой порядок следования уровней означает, что сначала определяется позиция дома, затем для данного дома определяется его жилец, после чего для жильца определяются его напиток, растение и животное. В общем случае порядок уровней не играет решающей роли, но для конкретной реализации он должен быть фиксирован.

Программа действует методом проб и ошибок. На самом верхнем уровне генерируется перестановка домов. Каждому дому присваивается позиция, соответствующая его номеру в перестановке. При этом программа наблюдает за «реакцией» модели. Допустим, что в перестановке зеленый дом оказался справа от белого (по условию, зеленый дом стоит слева от белого). При присвоении зеленому дому его позиции он проверит, где в дан ный момент находится белый дом.

Если позиция белого дома уже определена, и он не стоит слева от зеленого, объект, представляющий зеленый дом, выбросит исключение, сигнализирующее о том, что произошел конфликт с условием задачи. Внешняя программа перехватит исключение, произведет откат перестановки на текущем уровне (обнулит позиции всех домов) и перейдет к генерации следующей перестановки. Если же текущая перестановка не вызвала конфликт условий, программа спустится на уровень ниже.

На следующем уровне генерируется перестановка людей. После генерации каждому человеку

присваивается дом из текущей перестановки домов,

номер которого соответствует номеру данного чело;

века в перестановке людей. При этом модель автоматически присваивает значения определенным атрибутам объектов в соответствии с условиями задачи.

Допустим, внешняя программа пытается «поселить»

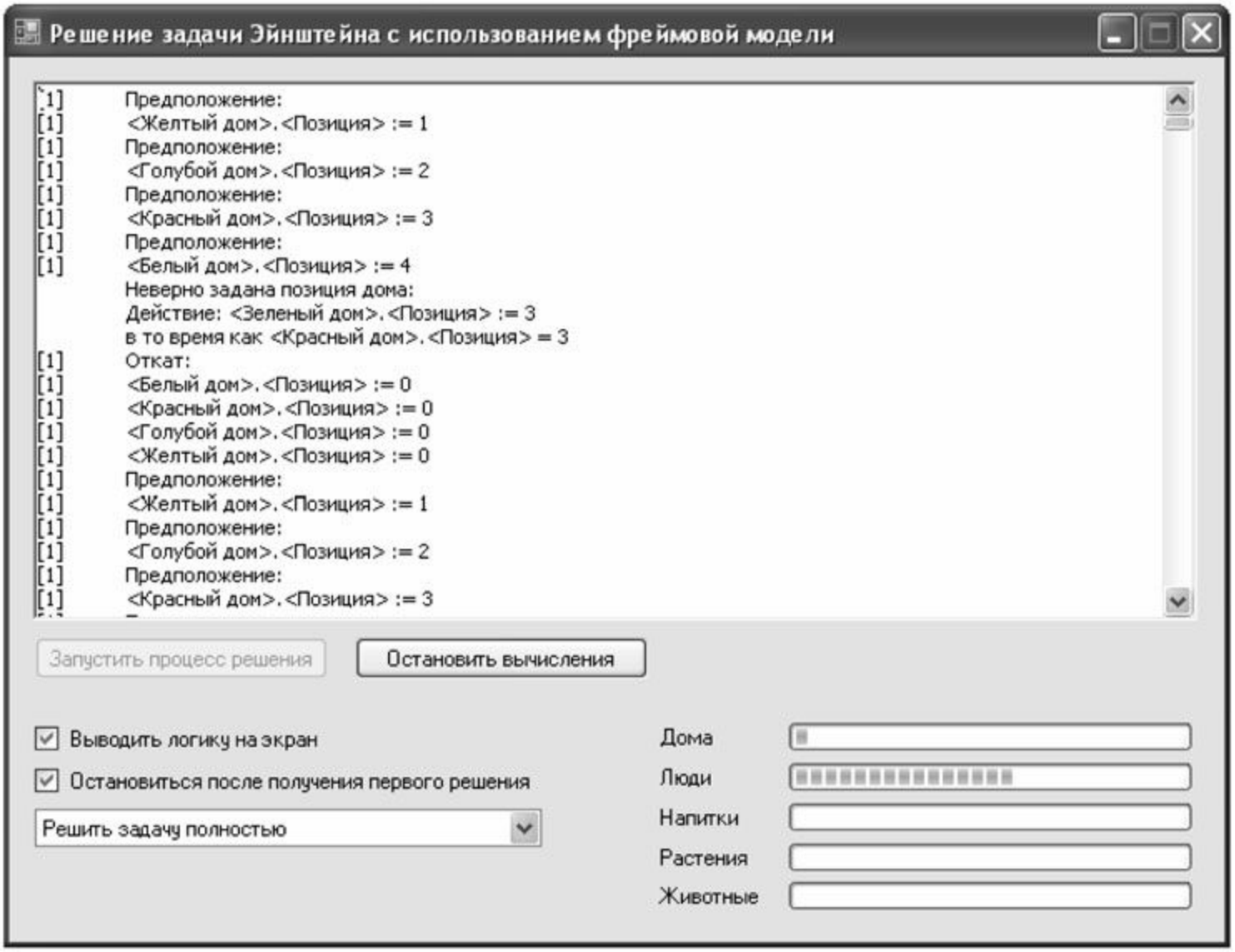
датчанина в желтый дом, стоящий на позиции «3».

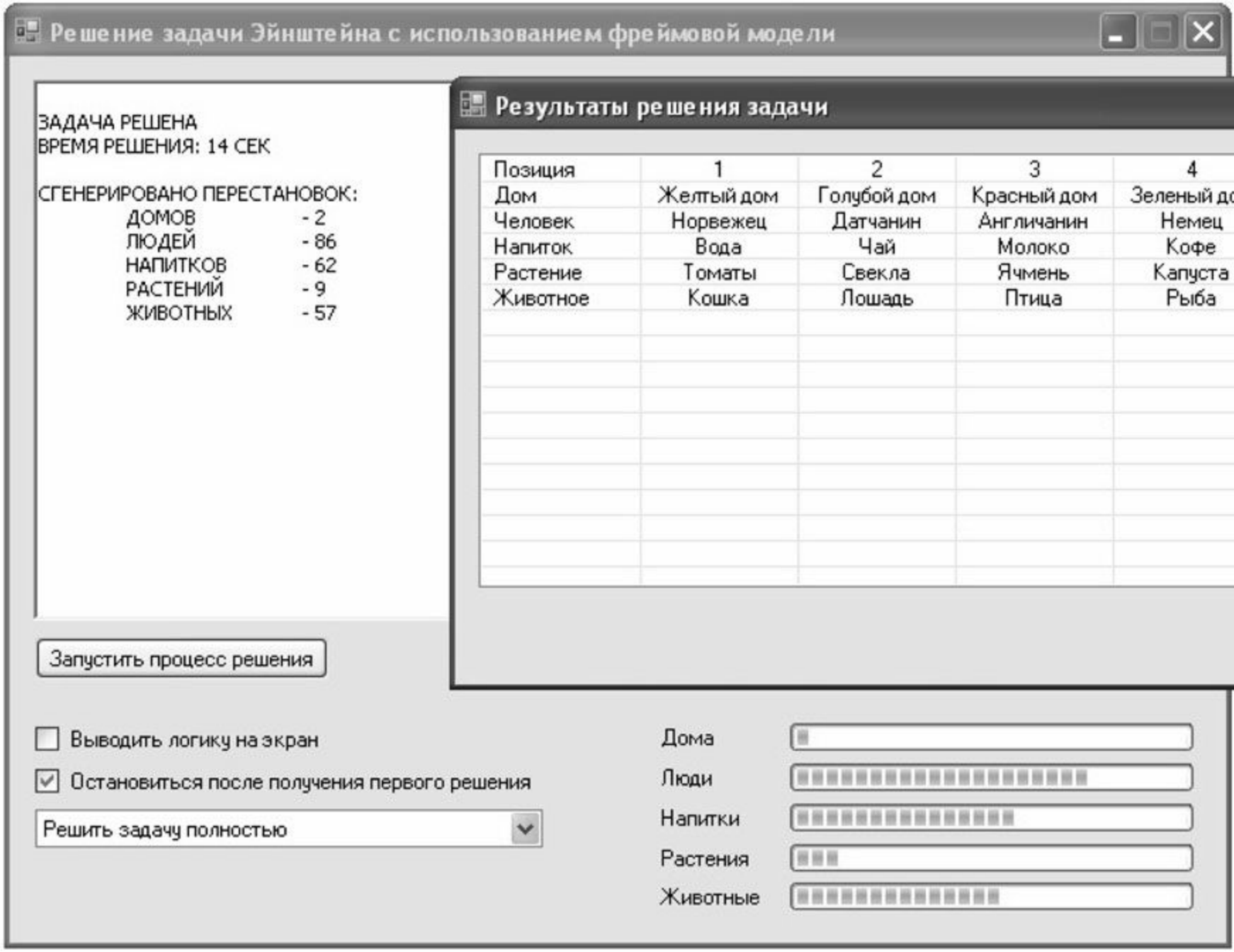
Объект, представляющий датчанина, проверяет по;

зицию желтого дома и, обнаружив, что дом стоит посередине, присваивает атрибуту «напиток» значение «молоко», т.к. по условию жилец из среднего дома пьет молоко. При этом этот же объект проверяет, не связано ли молоко с каким-либо другим человеком.

Если оказывается, что молоко уже используется для кого-то в качестве напитка, генерируется исключение. Таким образом, модель самостоятельно проверяет соблюдение всех условий задачи. Внешней программе остается лишь генерировать перестановки и следить за реакцией модели.

На рис. 3 показана работа программы в процессе решения задачи. Имеется возможность вывести на консоль ход «рассуждений» программы. При этом, как уже было сказано, предположения относительно расположения и принадлежности объектов делаются внешней программой, а проверка условий и генерация исключений (включая формирование текста ошибки) осуществляются моделью.


Решение задачи Эйнштейна с использованием фреймовой модели
[1] Предположение:
[1] <Желтый дом>.<Позиция> := 1
[1] Предположение:
[1] <Голубой дом>.<Позиция> := 2
[1] Предположение:
[1] <Красный дом>.<Позиция> := 3
[1] Предположение:
[1] <Белый дом>.<Позиция> := 4
Неверно задана позиция дома:
Действие: <Зеленый дом>.<Позиция> := 3
в то время как <Красный дом>.<Позиция> = 3
[1] Откат:
[1] <Белый дом>.<Позиция> := 0
[1] <Красный дом>.<Позиция> := 0
[1] <Голубой дом>.<Позиция> := 0
[1] <Желтый дом>.<Позиция> := 0
[1] Предположение:
[1] <Желтый дом>.<Позиция> := 1
[1] Предположение:
[1] <Голубой дом>.<Позиция> := 2
[1] Предположение:
[1] <Красный дом>.<Позиция> := 3
Запустить процесс решения
Остановить вычисления
Выводить логику на экран
Остановиться после получения первого решения
Решить задачу полностью
Дома
Люди
Напитки
Растения
Животные



Результаты решения задачи

Позиция	1	2	3	4	5
Дом	Желтый дом	Голубой дом	Красный дом	Зеленый дом	Белы
Человек	Норвежец	Датчанин	Англичанин	Немец	Ш
Напиток	Вода	Чай	Молоко	Кофе	С
Растение	Томаты	Свекла	Ячмень	Капуста	Пше
Животное	Кошка	Лошадь	Птица	Рыба	Соб

Фреймовая модель

Основные преимущества фреймов как модели представления знаний является способность отражать концептуальную основу организации памяти человека, а также естественность, наглядность представления, модульность, поддержку возможности использования правил умолчания, приписываемых слотам. Однако фрейм - представление является не конкретным языком представления знаний, а некоторой идеологической концепцией, реализуемой по-разному в различных языках.

Фреймовая модель

Теория фреймов послужила толчком к разработке нескольких языков представления знаний, которые благодаря своим широким возможностям и гибкости стали в последние годы довольно распространенными языками. К ним относятся FRL (Frame Representation Language), KRL, FMS и др., представляющие собой среду для разработок и исследований в области ИИ и широко используемые специалистами в данной области.

Фреймовая модель

Теория фреймов послужила толчком к разработке нескольких языков представления знаний, которые благодаря своим широким возможностям и гибкости стали в последние годы довольно распространенными языками. К ним относятся FRL (Frame Representation Language), KRL, FMS и др., представляющие собой среду для разработок и исследований в области ИИ и широко используемые специалистами в данной области.

Представление знаний с использованием семантических сетей

Термин *семантическая сеть* применяется для описания метода представления знаний, основанного на сетевой структуре. Семантические сети были первоначально разработаны для использования их в качестве психологических моделей человеческой памяти, но теперь это стандартный метод представления знаний в ИИ и в экспертных системах.

Представление знаний с использованием семантических сетей

Семантические сети состоят из точек, называемых *узлами,* и связывающих их *дуг,* описывающих отношения между узлами. Узлы в семантической сети соответствуют объектам, концепциям или событиям. Дуги могут быть определены разными методами, зависящими от вида представляемых знаний. Обычно дуги, используемые для представления иерархии, включают дуги типа isa (является) и has-part (имеет часть).

Представление знаний с использованием семантических сетей