Решение задачи Эйнштейна на основе фреймовой модели представления знаний

Олег Викторович Рогозин,
доцент МГТУ им. Баумана, кандидат технических наук

В СИСТЕМАХ, ИСПОЛЬЗУЮЩИХ МЕХАНИЗМ ЛОГИЧЕСКОГО ВЫВОДА РЕШЕНИЯ, ОСНОВНУЮ РОЛЬ ИГРАЕТ СПОСОБ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ. НАИБОЛЕЕ РАСПРОСТРАНЕНЫ ЛОГИЧЕСКАЯ, ПРОДУКЦИОННАЯ, ФРЕЙМОВАЯ МОДЕЛИ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ, А ТАК ЖЕ СЕМАНТИЧЕСКАЯ СЕТЬ. РАССМОТРИМ ПРИМЕНЕНИЕ ФРЕЙМОВОЙ МОДЕЛИ НА ПРИМЕРЕ РАЗРАБОТКИ ПРОГРАММНОГО МОДУЛЯ РЕШЕНИЯ ЗАДАЧИ ЭЙНШТЕЙНА.

Определим фреймовую модель представления знаний следующим образом:

• знание организовано в виде совокупности взаимосвязанных описаний — классов данной предметной области;

• формальная структура для такого описания названа фреймом. Это даёт возможность упаковать знания в структуры, организованные подходящим образом, сохраняя их обозримость на всех уровнях;

• фреймы организуются в иерархии по уровню абстракции; конкретным объектам соответствуют экземпляры фреймов; свойства общих понятий наследуются конкретным понятием в иерархии. Наследование представляет собой особый вид дедукции, что даёт выигрыш в эффективности и, что более важно, облегчает построение системы фреймов — базы знаний. Такие описания выглядят гораздо более близкими к мыслительным структурам человека, нежели формализмы типа исчисления предикатов;

• структурное описание предполагает содержание ссылок на другие фреймы, что позволяет описывать различные отношения между понятиями данной предметной области. Указанные ссылки могут иметь вид фрейма — прототипа со значениями его свойств-слотов. Сопоставление с прототипом также является механизмом, частично заменяющим обычную дедукцию, в то же время более адекватным механизму мышления человека;

• с фреймами связываются процедуры, описывающие те или иные специализированные для данной предметной области процессы (проблемно-логический подход, поддержка целостности базы знаний, учёт взаимосвязей экземпляров различных понятий и т.п.). Таким образом, фрейм представляет собой декларативно-процедурный модуль, включающий собственно предметную область, а также дополнительную информацию о способе его интерпретации.

Между различными концептуальными объектами существуют некоторые аналогии, в результате чего и фреймы, представляющие такие образы, выстраиваются в иерархическую систему с классификационными и обобщающими свойствами. При этом сложные объекты представляются комбинацией нескольких фреймов (вложенными фреймами). Свойством сети фреймов, заимствованным из семантических сетей, является наличие AKO-связей («A-Kind-Of»), которые связывают фреймы с фреймами, находящимися на уровень выше в иерархии, откуда неявно наследуются (переносятся) значения слотов. Каждый фрейм имеет уникальное имя (идентификатор) в пределах системы фреймов.

ЧЕЛОВЕК	
АКО	Млекопитающее
УМЕЕТ	Мыслить

ЛЕКТОР	ЛЕКТОР
АКО	Человек
ОБРАЗОВАНИЕ	Высшее
ВОЗРАСТ	23–60

Рис 1. Фрагмент иерархии «человек — лектор»

В данном случае представлено одно звено иерархии (ЧЕЛОВЕК-ЛЕКТОР). Здесь фрейм «ЧЕЛОВЕК» является обобщающим для фрейма «ЛЕКТОР». Таким образом, фрейм «ЛЕКТОР» наследует от фрейма «ЧЕЛОВЕК» значение слота «УМЕЕТ» (а также других слотов, не показанных в примере). Цепочка наследования может быть продолжена вплоть до, например, фрейма «ЖИВОЕ СУЩЕСТВО».

Такая структура позволяет систематизировать большой объём информации, оставляя её при этом максимально удобной для использования. Кроме того, система фреймов способна отражать концептуальную основу организации памяти человека. Рассмотрим реализацию фреймовой модели представления знаний на примере решения задачи Эйнштейна. Условия задачи следующие:

1. Есть пять домов, каждый разного цвета.
2. В каждом доме живёт один человек, отличающийся от соседа по национальности: немец, англичанин, швед, датчанин, норвежец.
3. Каждый пьёт только один напиток, выращивает определённое растение и держит определённое животное.
4. Никто из 5 человек не пьёт одинаковые напитки, не выращивает одинаковое растение и не держит одинаковое животное.

Цель решения: определить, кому принадлежит рыба? При этом дополнительные условия следующие:

1. Англичанин живёт в красном доме.
2. Швед держит собаку.
3. Датчанин пьёт чай.
4. Зелёный дом стоит слева от белого.
5. Жилец зелёного дома пьёт кофе.
6. Человек, который выращивает ячмень, держит птицу.
7. Жилец из среднего дома пьёт молоко.
8. Жилец из жёлтого дома выращивает томаты.
9. Норвежец живёт в первом доме.
10. Тот, кто держит кошку, живёт около того, кто выращивает свёклу.
11. Человек, который содержит лошадь, живёт около того, кто выращивает томаты.
12. Тот, кто выращивает пшеницу, пьёт сок.
13. Норвежец живёт около голубого дома.
14. Немец выращивает капусту.
15. Тот, кто выращивает свёклу, живёт по соседству с человеком, который пьёт воду.

Если трактовать условие 4 как «зелёный дом стоит непосредственно слева от белого», то задача имеет единственное решение. Причём под решением в данном случае понимается не только определение того, кому принадлежит рыба, но и полное распределение дом — человек — напиток — растение — животное, охватывающее все описанные в условии объекты. 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 — конкретного фрейма, наследуемого от абстрактного и описывающего этот объект. На рис. 2 представлены фреймовые диаграммы классов «Дом» и «Человек». Диаграммы классов «Напиток», «Растение» и «Животное» не показаны, поскольку представляющие их фреймы достаточно тривиальны (не содержат ни одного слота).

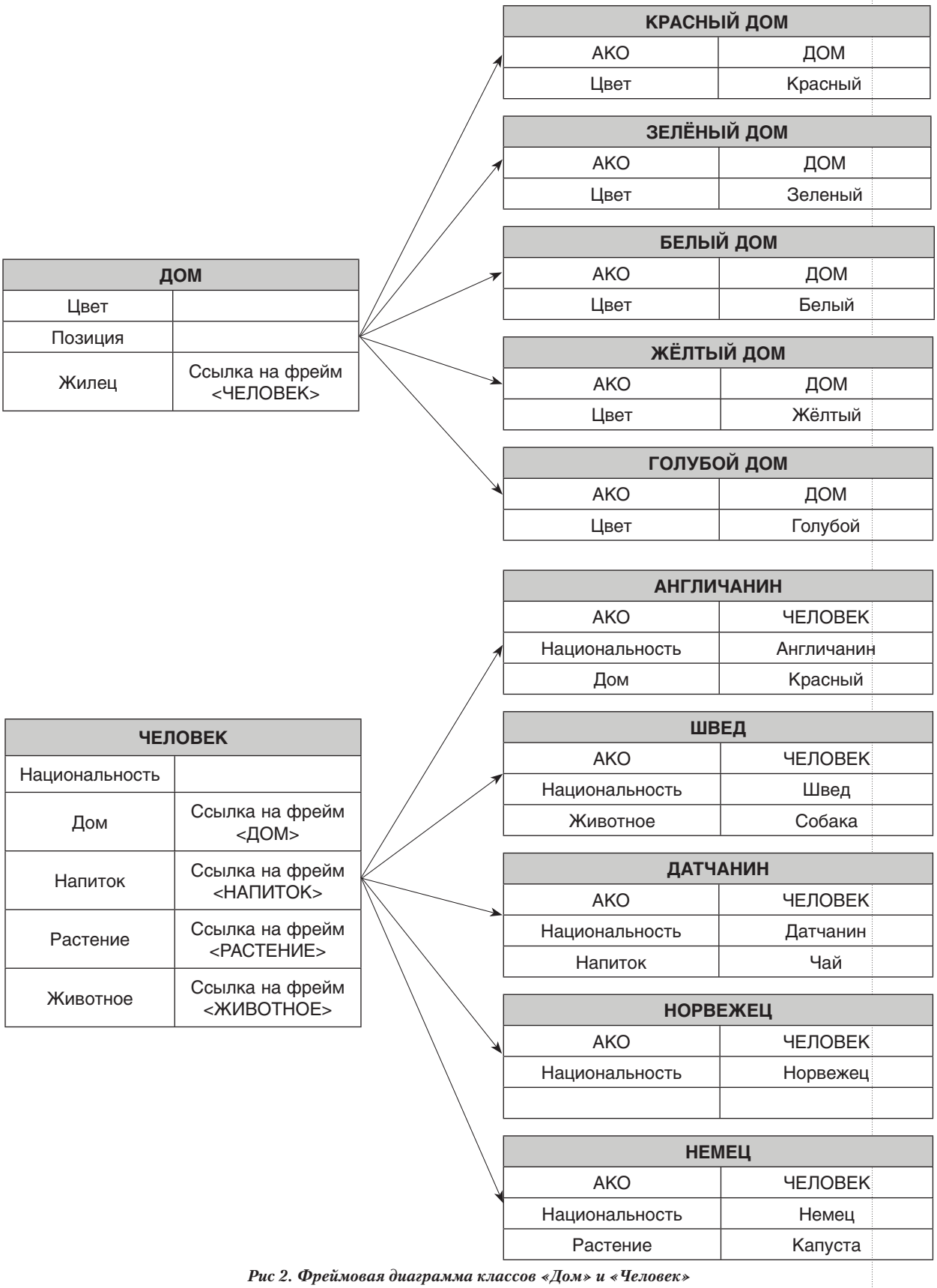

Рис 2. Фреймовая диаграмма классов «Дом» и «Человек»

Диаграмма на рис. 2 показывает, что фреймы, представляющие конкретные объекты, наследуют слоты абстрактных фреймов, определяя значения для некоторых из них. Таким образом, часть информации, содержащейся в условии задачи, задаётся с помощью значений слотов. Данная информация носит декларативный характер, поскольку задаётся на этапе построения модели и не требует дополнительных проверок.

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 жёлтого дома выращивает томаты» может быть проверено процедурой, вызываемой при изменении значения слота «РАСТЕНИЕ» фрейма «ЖЁЛТЫЙ ДОМ». Сложнее обстоит дело с условиями типа «тот, кто держит кошку, живёт около того, кто выращивает свёклу». 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 путём введения списка домов, учитывающего их расстановку, причём каждый «дом» имеет доступ к этому списку.

Фреймовая модель очень удобна с точки зрения программной реализации, поскольку она напрямую соответствует парадигме объектно-ориентированного программирования. С точки зрения ООП, фреймы представляют собой не что иное, как классы. Слоты являются аналогом полей классов. Иерархия фреймов соответствует иерархии классов, которая строится с помощью механизма наследования. Процедуры фреймов, выполняемые при определённых условиях, можно реализовать с помощью обработчиков соответствующих событий, определённых для полей классов. Горизонтальные связи реализуются путём добавления в класс ссылок на объекты других классов, с которыми планируется взаимодействие. На рис. 3 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. Классы, представляющие напитки, растения и животных, в силу своей тривиальности реализованы в виде перечислений.

Программный модуль состоит из двух основных частей: набора классов, реализующих фреймовую модель, и внешнего модуля, который осуществляет генерацию наборов данных, наблюдением за «реакцией» модели и принятием необходимых решений. Вся предметная область разбита на 5 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 — 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 присваеваются позиция, соответствующая его номеру в перестановке. При этом программа наблюдает за «реакцией» модели. Допустим, что в перестановке зелёный дом оказался справа от белого (по условию, зелёный дом стоит слева от белого). При присвоении зелёному дому его позиции он проверит, где в данный момент находится белый дом. Если позиция белого дома уже определена и он не стоит слева от зелёного, объект, представляющий зелёный дом, выбросит исключение, сигнализирующее о том, что произошёл конфликт с условием задачи. Внешняя программа перехватит исключение, произведёт откат перестановки на текущем уровне (обнулит позиции всех домов) и перейдёт к генерации следующей перестановки.

Если же текущая перестановка не вызвала конфликта 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 человека в перестановке людей. При этом модель автоматически присваивает значения определённым атрибутам объектов в соответствии с условиями задачи. Допустим, внешняя программа пытается «поселить» датчанина в жёлтый дом, стоящий на позиции «3». Объект, представляющий датчанина, проверяет позицию жёлтого дома и, обнару-

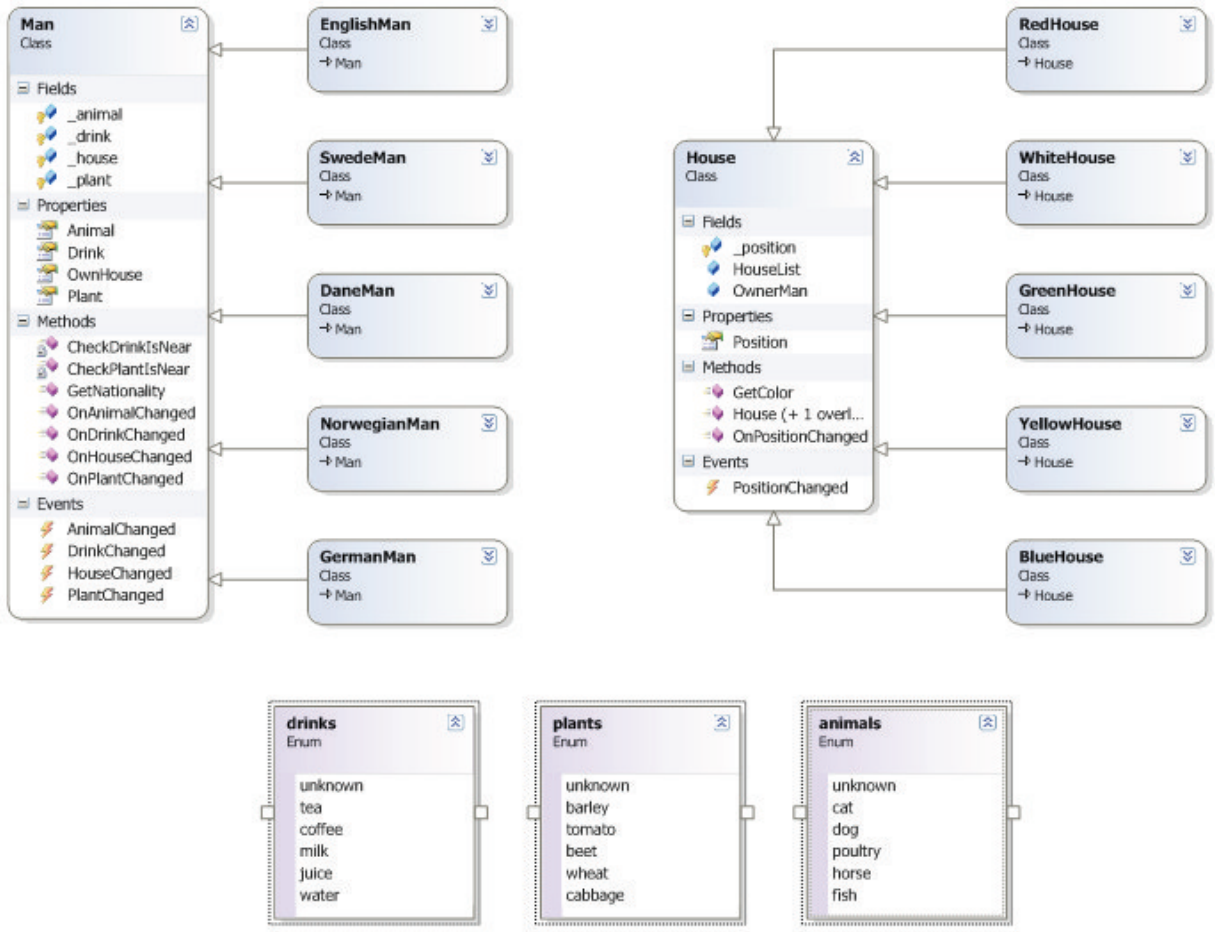

Рис. 3. Диаграмма классов предметной области задачи

жив, что дом стоит посередине, присваивает атрибуту «напиток» значение «молоко», т.к. по условию жилец из среднего дома пьёт 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.

Модель самостоятельно проверяет соблюдение всех условий задачи. Внешней программе остаётся лишь генерировать перестановки и следить за реакцией модели. Как было описано выше, в случае, если очередная перестановка не вызвала конфликта с условиями задачи, программа спускается на уровень ниже и начинает генерацию на более низком уровне. Если же ни одна из перестановок на текущем уровне не дала положительного результата, программа возвращается на предыдущий уровень и продолжает генерацию на более высоком уровне. Таким образом, задача решается методом перебора всех возможных вариантов. С одной стороны, решение полным перебором требует в общем случае наибольших временных затрат среди существующих методов решения. С другой стороны, осуществление полного перебора гарантирует единственность решения задачи (а если решений несколько — позволяет найти их все).

На рис. 4 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. Индикаторы в нижней правой части окна приложения показывают процент сгенерированных перестановок от общего числа перестановок на каждом уровне (максимальное число перестановок на одном уровне равно 5! = 120). На рис. 5 показаны результаты работы программы:

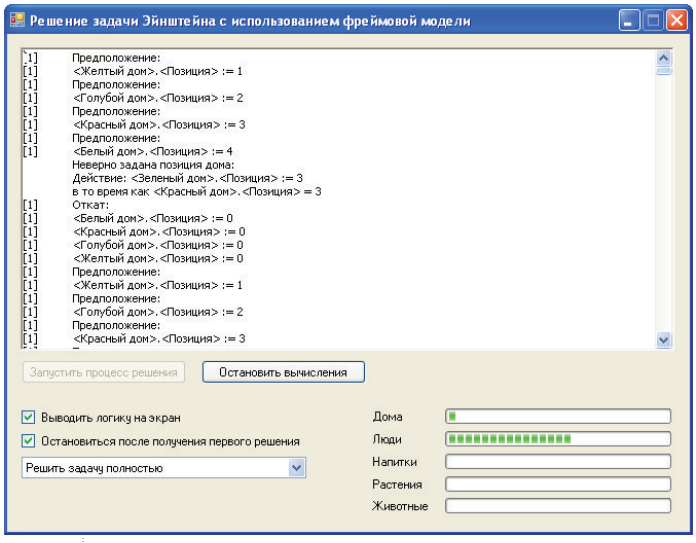

Рис. 4. Программа в процессе решения задачи

таблица, отображающая полное решение задачи, а также статистика — время, затраченное на решение, и число сделанных перестановок на каждом уровне.

Метод решения задачи полным перебором вариантов имеет некоторую особенность. Если программа останавливается после получения первого решения (т.е. происходит перебор только части вариантов), то время решения сильно зависит от того, насколько хорошо «угаданы» начальные перестановки объектов, которые задаются в коде программы. На рис. 5 видно, что программа решила задачу, сделав только две перестановки домов, т.е. начальная перестановка была очень близка к той, которая дала положительный результат. А проверка одной перестановки домов (не вызвавшей конфликта на уровне домов) требует наибольших временных затрат, поскольку под ней находятся ещё 4 уровня генерации и проверок. При этом предметная область задачи состоит из 5 уровней по 5 объектов в каждом, т.е. количество всех возможных перестановок составляет $(5!)^5 = 24\ 883\ 200\ 000$. Для каждого уровня существуют условия, которые позволяют «забраковать» перестановку ещё до спуска на уровень ниже. Для уровня домов это условие «Зелёный дом стоит слева от белого», которое сразу отбрасывает больше половины всех возможных перестановок домов. На уровне людей имеются 2 условия: «Англичанин живёт в красном доме» и «Норвежец живёт в первом доме». Вероятность того, что программа «попытается поселить» англичанина именно в красный дом составляет 1/5, вероятность того, что красный дом окажется не 1-м по порядку, равна 4/5, и, наконец, вероятность того, что норвежец будет «поселён» именно в 1-й дом, равна 1/4 (в одном доме уже «живёт» англичанин).

Таким образом, вероятность того, что сгенерированная на уровне людей перестановка будет удовлетворять условиям задачи, не превышает 1/25. Учитывая то, что одновременно с «расселением» людей модель будет расставлять и проверять атрибуты более низкого уровня (например, при «поселении» человека в зелёный дом, ему, согласно условию, будет приписано кофе в качестве напитка, а если этим человеком окажется датчанин, который по условию пьёт чай, расстановка будет забракована), вероятность заметно уменьшается. Таким образом, уже после применения условий для 1-го и 2-го уровней от исходных 25 миллиардов остаются не более 500 миллионов перестановок, проверка которых требует спуска на нижние уровни модели. ❑

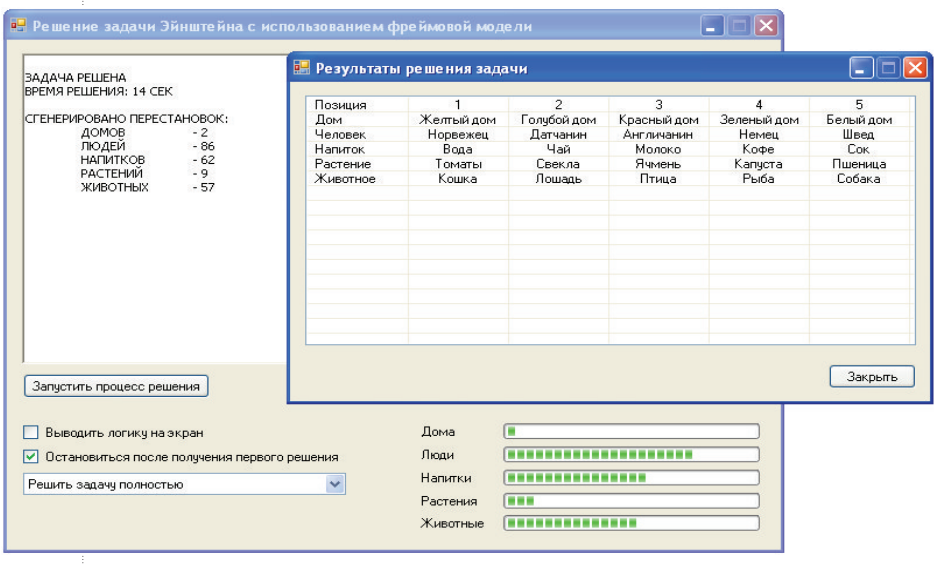

Рис. 5. Результаты работы программы