МОДЕЛИ ЗНАНИЙ

Имеется несколько хорошо разработанных моделей знаний, обладающих своими достоинствами и недостатками. Это

- Логические модели;
- Продукционные модели;
- Фреймы;
- Спецификации;
- Семантические сети;
- Таблицы данных.

Каждая из этих моделей допускает также деление. Например, в классе логических моделей имеются модели на основе логики высказываний и логики предикатов. Различают также модели на основе четкой и нечеткой логики, двузначной и многозначной, противоречивой и непротиворечивой логики и т.д.

Логическая модель знаний

Логические модели, на которых в настоящем материале сделан главный акцент, наиболее универсальны и формализованы. Для них имеется мощный математический аппарат. Логическая модель представляет конечное или бесконечное множество логических формул. Для записи формул используется язык, содержащий алфавит (набор символов) и множество правил образования правильно построенных формул. Символами логического языка являются:

- Константы (например, 0,1,2,… или ‘a’,’b’,’c’,…);
- Переменные: x,y,…,z;
- Функциональные символы: f,g,h, … (функциональные символы называются также функторами);
- Символы предикатов (отношений) P,Q,R,…
- Кванторы всеобщности $\forall$ и существования $\exists$.
- Логические операции (связки): & (“и”), $\vee$ (“или”), $\neg$ (“не”), $\rightarrow$ (“следует”), $\leftrightarrow$ (“эквивалентно”). & называют операцией конъюнкции, v – дизъюнкции, $\neg$ - отрицания, $\rightarrow$ - импликацией и $\leftrightarrow$ - эквиваленцией.

Кроме символов языка, рассматриваются синтаксические правила записи предложений (формул) логического языка. Отправным элементом является простейшая формула – **предикат** (в логике предикатов) и булевская переменная (в логике высказываний).

Предикат – это простейшая (атомарная) формула (с отрицанием или без него) вида P(…), где в скобках указываются аргументы формулы – переменные, константы или функторы, либо вовсе не указываются аргументы. Предикат принимает одно из двух возможных значений: истина или ложь. Аргументы предиката имеют произвольную природу. Каждый предикат есть формула логического языка. Пусть P, Q – любые две формулы. Тогда формулами также являются следующие выражения:

1. $\neg P, \neg Q$;
2. P v Q;
3. P & Q;

4. $P \rightarrow Q$;
   (**3.1**)

5. $P \leftrightarrow Q$;
6. $\forall x\, P(x)$;
7. $\exists\, x\, P(x)$

(в (6,7) переменная x входит свободно в P(…), т.е. x не связана в P каким-либо квантором).

В дальнейшем знак &, как правило, будем опускать, а знак $\leftrightarrow$заменять на = или $\equiv$.

Определение. 1. Дизъюнктом (в логике высказываний) называется дизъюнкция литер (булевских переменных) с отрицанием или без, например, $x \vee y \vee \neg z$. 2. Дизъюнктом в логике предикатов называется дизъюнкция литералов (атомарных формул), взятых с отрицанием или без него, например, $P(a,z) \vee \neg Q(z,f(x))$.

Определение. Конъюнктивной нормальной формой (КНФ) называется конъюнкция дизъюнктов.

Определение. Выполняющей интерпретацией I для заданной КНФ называется множество значений переменных этой КНФ, при которых она истинна. Например, КНФ

$(x_1 \vee \neg x_2)(\neg x_1 \vee x_2)(\neg x_2 \vee \neg x_3)(x_1 \vee x_3)$ истинная в интерпретации $I = \langle x_1 = 1, x_2 = 1, x_3 = 0\rangle$ либо в интерпретации $\langle x_1 = 0, x_2 = 0, x_3 = 1\rangle$.

Число различных интерпретаций для дизъюнкта с n>0 переменными равно 2^n . Однако не все они являются в общем случае выполняющими интерпретациями.

Определение. Задача ВЫПОЛНИМОСТЬ (коротко – ВЫП) формулируется так: для данной КНФ установить, имеется ли для нее хотя бы одна выполняющая интерпретация.

Для задачи ВЫП известно, что она является полиномиально универсальной в классе задач, разрешимых за полиномиальное время на недетерминированной машине Тьюринга, однако найти для нее полиномиальный алгоритм для детерминированной машины Тьюринга до сих пор не удалось. Следовательно, ВЫП можно отнести к категории интеллектуальных задач.

Определение. Формула B следует из формулы A, если в любой интерпретации, где A истинна, B также истинна. Пишем $A \to B$ в этом случае.

Определение. Формула A тождественно истинна, если она истинна в любой интерпретации. Тождественно истинная формула называется также тавтологией.

Определение. Правилом вывода R называется такое соотношение между формулами A_1, A_2, ..., A_n и B, которое устанавливает истинность формулы B всякий раз, когда выполняется заданное соотношение.

Определение. Формула B выводима из формул A_1, A_2, ..., A_n, если имеется конечная последовательность формул П, начинающаяся с любой из формул A_i , такая, что каждая очередная формула этой последовательности либо выводима по некоторому правилу вывода из предшествующих членов (или их части), либо совпадает с какой-то из формул A_i .

Теорема 3.1 (о полноте в узком смысле логики первого порядка и логики высказываний). Всякая тождественно истинная формула выводима.

Задача логического вывода в логике высказываний может быть эффективно сведена к ВЫП. В самом деле, пусть даны дизъюнкты D_1, D_2, ..., D_z. Спрашивается, выводим ли из них дизъюнкт R? (т.е. требуется установить тождественную истинность формулы $D_1 \& D_2 \& \ldots \& D_z \to R$. Умножим эту последнюю формулу справа и слева на $\neg R$. Получим:

$$D_1 \& D_2 \& \ldots \& D_z \& \neg R \to \text{False}$$

Следовательно, если удастся показать выполнимость системы дизъюнктов $D_1 \& D_2 \& \ldots \& D_z \& \neg R$, то получим опровержение исходной формулы $D_1 \& D_2 \& \ldots \& D_z \to R$. Если докажем, что система не выполнима, то получим доказательство исходной формулы. Таким образом, задача логического вывода сводится к задаче ВЫП.

Пример 1. В хищении подозреваются A,B и C. Известно следующее.

1. Никто, кроме A,B,C в хищении не замешан.
2. A никогда не ходит на дело без соучастников.
3. C не виновен, если виновен B.

Спрашивается, виновен ли A?

Нужно составить систему логических уравнений, введя необходимые булевские переменные. Система переменных должна быть адекватна условиям задачи. Обозначим: x_a – виновен A, x_b – виновен B, x_c – виновен C. Составим следующую базу знаний:

1. $x_a \vee x_b \vee x_c$.
2. $x_a \rightarrow x_b \vee x_c$.
3. $x_b \rightarrow \neg x_c$.

Эти логические формулы полностью соответствуют условиям задачи. Можно воспользоваться арифметическим представлением логических формул. Этот прием даст связь с рассмотренными нами выше условиями формализации . Арифметические представления таковы:

$$f \vee g \vee \ldots \vee h \equiv f+g+ \ldots +h \geq 1$$
$$\neg f \equiv (1- f)$$

(3.2)

$$f \rightarrow g \equiv \neg f \vee g \equiv (1- f) + g \geq 1$$
$$f \,\&\, g \equiv f + g = 2 \equiv \neg f + \neg g = 0 \equiv f * g$$

Теперь условия нашей задачи получат следующее представление:

1. $x_a + x_b + x_c \geq 1$.
2. $(1-x_a) + x_b + x_c \geq 1$.
3. $(1-x_a) + (1- x_c) \geq 1$.

Относительно этой модели знаний поставлен вопрос: верно ли что $x_a = 1$? Ответ на этот вопрос должна дать **машина вывода**.

Пример 2.

Следует назначить по одной работе **w1, w2, w3, w4** четырем исполнителям **a, b, c, d**, если известно, что **a** может выполнить либо работу **w2**, либо работу **w3**; **b** может выполнить работу **w1** или **w2**; **c** может выполнить **w1** или **w3** или **w4**; **d** может выполнить работу **w2** или **w4**.

Базу знаний для этой задачи можно записать с помощью формул логики предикатов таким образом:

//группа условий, показывающих, какие работы могут быть назначены исполнителям

P(**a**, X) $\rightarrow$ (X = **w2**) $\vee$ (X = **w3**)

P(**b**, Y) $\rightarrow$ (Y=**w1**) $\vee$ (Y = **w2**)

P(**c**, Z) $\rightarrow$ (Z=**w1**) $\vee$ (Z = **w3**) v (Z = **w4**)

P(**d**, R) $\rightarrow$ (R = **w2**) $\vee$ (R = **w4**)

//группа условий, показывающих, какие исполнители могут быть назначены на работы

P(X1, **w1**) $\rightarrow$ (X1 = **b**) $\vee$ (X1 = **c**)

P(Y1, **w2**) $\rightarrow$ (Y1 = **a**) $\vee$ (Y1 = **b**) $\vee$ (Y1 = **d**)

P(Z1,**w3**) $\rightarrow$ (Z1 = **a**) $\vee$ (Z1 = **c**)

P(R1, **w4**) $\rightarrow$ (R1 = **c**) $\vee$ (R1 = **d**)

//группа условий, показывающих, что исполнителям не могут назначаться одинаковые работы

P(**a**, X) & P(**b**, Y) $\rightarrow$ (X<>Y)

P(**a**, X) & P(**c**, Y) $\rightarrow$ (X<>Y)

P(**a**, X) & P(**d**, Y) $\rightarrow$ (X<>Y)

P(**b**, X) & P(**c**, Y) $\rightarrow$ (X<>Y)

P(**b**, X) & P(**d**, Y) $\rightarrow$ (X<>Y)

P(**c**, X) & P(**d**, Y) $\rightarrow$ (X<>Y)

Для этой системы знаний нужно доказать следующую формулу

$\exists$S $\exists$T $\exists$Q $\exists$V (P(**a**, S) & P(**b**, T) & P(**c**, Q) & P(**d**, V))

Заметим, что машина вывода не только должна построить доказательство, но и получить из построенного доказательства значения переменных S, T, Q, V. Как это сделать обсуждается в разделе, посвященном машине вывода.

Продукционная модель знаний

Продукционная модель представляет собой вариант логической модели, записанной с помощью продукционных формул (правил) вида

$F \rightarrow G$, где F,G – произвольные, вообще говоря логические формулы. Отмечается, что продукции “более близки” по форме “естественным” умозаключениям. На практике это чрезмерно общее условие можно ограничить, рассматривая формулы так называемого секвенционального вида

$$f_1 \& f_2 \& \ldots \& f_z \to g_1 \vee g_2 \vee \ldots \vee g_t \ ,$$

(3.3)

где все используемые в представлении **(3.3)** формулы суть атомарные. Формулу **(3.3)** нетрудно привести к форме дизъюнкта:

$$\neg f_1 \vee \neg f_2 \ \& \ \ldots \ \vee \neg f_z \ \vee g_1 \vee g_2 \vee \ldots \vee g_t \ ,$$

(3.4)

так что никакой особенной выгоды представление **(3.3)** с этой точки зрения не дает. Однако для систем формул вида **(3.3)** разработано (Генценом) секвенциальное исчисление, позволяющее строить логические выводы в форме так называемого секвенциального дерева. Система логического программирования Пролог используется еще более упрощенные секвенциальные формулы (так называемую нотацию Хорна):

$$f_1 \& f_2 \& \ldots \& f_z \to g_1$$

(3.5)

В **(3.5)** часть $f_1 \& f_2 \& \ldots \& f_z$ называется **телом** продукции, а g_1 **- головой**. Продукция без головы называется **целью** (*goal*). Каждая предикатная формула f_i называется в этом случае подцелью. Перейти от общего представления **(3.3)** к представлению **(3.4)** просто, если принять во внимание следующую эквивалентность

$$\neg f_1 \vee \neg f_2 \ \& \ \ldots \vee \neg f_z \vee g_1 \vee g_2 \vee \ldots \vee g_t \ \equiv \ \neg g_2 \ \& \ \neg g_3 \ldots \& \ \neg g_t \& f_1 \ \& \ \ldots \ \& f_z \to g_1$$

Рассмотрим представление знаний в системе программирования Пролог.

Пример 1. Владимиру нравится все, что нравится Ире.

```
нравится(владимир, X):-
нравится(ира, X).
```

Эта продукция соответствует формуле

нравится(ира,X) → нравится(владимир,X).

В Прологе тело продукции записывается после символов ":-".

Пример 2. Алексей мечтает о деньгах и карьере.

Это предложение определяет два факта:

мечтает(алексей, карьера).
мечтает(алексей, деньги).

Факты – это особый вид правил, которые не нужно доказывать. Такие правила, как можно видеть, представляют продукции без “головы”.
При программировании на языке Пролог особую трудность вызывает использование рекурсивных продукций, в которых тело продукции содержит предикат, одновременно стоящий в голове продукции. Рассмотрим пример вычисления суммы чисел от 1 до заданного числа n.

```
sum(0, X, X).

sum(Z, S, X):-
Z1=Z-1,
S1=S+Z,
sum(Z1,S1,X).
```

Во-первых, заметим, что переменные в языке Пролог пишут с больших букв. Во- вторых, в данном примере назначение аргументов предиката sum таково: первый аргумент есть текущее число, второй аргумент есть промежуточная накапливаемая сумма, третий аргумент есть результирующая сумма. Продукция sum(0, X, X) является фактом, утверждающим, что если текущее число равно 0, то промежуточная накапливаемая сумма и результирующая сумма совпадают. Вторая продукция как раз и дает пример рекурсии в Прологе. Эту продукцию следует читать таким образом:

Если (Z1=Z-1) и (S1=S+Z) и (1+2+3+…+Z1=S1) то
(1+2+3+…(Z-1)+Z=S).

Запись рекурсивных определений подчиняется некоторым общим правилам. Рекурсия обязана в конце концов сводится к простому факту. Тело рекурсии должно связывать значения аргументов доказываемой формулы с изменяемыми значениями аргументов той же формулы. В рассмотренном примере все эти характеристики рекурсии налицо.

Продукционные модули используются в определении формальных языков (грамматик). Например, определение целого числа без знака, допускающего ведущие нули, может иметь такой вид

Цифра :- 0.
Цифра :- 1.
Цифра :- 2.
Цифра :- 3.
Цифра :- 4.
Цифра :- 5.
Цифра :- 6.
Цифра :- 7.
Цифра :- 8.
Цифра :- 9.
Число :- Цифра.
Число :- Цифра Число.

Нетрудно видеть, что 00123 есть объект, соответствующий данной грамматике, причем грамматика допускает бесконечные числа, например, состоящие из одних нулей 0000..000…. .

Модель знаний на основе фреймов

Фреймы предложены Марвином Минским. Минский также ввел терминологию и язык фреймов, включающий понятия “фрейма”, “слота”, “терминала”, “значения по умолчанию”. Фрейм имеет следующую структуру:

{<имя-фрейма><имя слота$_1$ ><значение слота$_1$>
……
<имя слота$_n$ ><значение слота$_n$>}

В качестве примера рассмотрим фрейм <выбор скорости>:

{<выбор скорости>
<состояние дороги>:**0.6**
<состояние машины>:**0.8**
<состояние водителя>:**0.5**
}

Наша гипотетическая экспертная система должна определять оптимальную скорость автомобиля (в пределах дозволенной) с учетом состояния дороги, машины и водителя. В данном примере уже указаны конкретные значения, так что необходима процедура оценки скорости по конкретным значениям слотов. Такая процедура называется **демоном**. Процедура-демон запускается автоматически, когда фрейм удовлетворяет некоторому образцу. Наряду с процедурами-демонами имеются процедуры-слуги, которые используются для установления значений слотам. Так, для определения состояния дороги можно было использовать следующий фрейм:

{<состояние дороги>
  <состояние покрытия>:**0.5**
  <видимость>:**1.0**
}

Такие же фреймы могли быть установлены для состояния машины и состояния водителя. Далее, например, для состояния покрытия можно использоваться фрейм:

{<состояние покрытия>
  <материал покрытия>:**0.4**
  <наличие влаги>:**0.0**
 <изношенность дороги>:**0.1**
}

Ясно, что организация экспертной оценки скорости требует создать все необходимые фреймы и разработать процедуру комплексной оценки. Хорошей основой для такой процедуры является метод Т. Саати, который мы изложим при описании машины вывода. Модель знаний на основе фреймов является сравнительно простой и легко реализуемой. В ее основу можно положить обычную базу данных с визуальным интерфейсом.

Спецификации и семантические сети

В реальном мире любую ситуацию можно охарактеризовать следующим образом. Во-первых, указать, какие объекты участвуют в ситуации. Во-вторых, определить, в какие отношения вступают объекты. Наконец, указать свойства объектов и свойства отношений. Таким образом можно передать знания об очень широком классе ситуаций. Рассмотрим

пример. Дано предложение: “Студент Максимов сдал экзамен по химии”. В этой ситуации выделяем объекты: **Максимов** и **экзамен**. Отношения между объектами передаются с помощью глаголов. В нашем случае отношением является глагол **сдать**. Свойством Максимова является <**студент**>. Свойством экзамена является <**по химии**>. Свойством отношения <сдать> является время и характер действия (в нашем случае – это прошедшее время). Таким образом, описываемую ситуацию можно было бы представить с помощью следующей спецификации (на некотором полуформальном языке):

Situation : Сессия

Objects: Максимов(студент), экзамен (по химии)

Relation: Сдать[Максимов, Экзамен] (past time)

Спецификацию можно охарактеризовать как описание предметной ситуации (проблемы, задачи) на формализованном языке. Спецификацию можно усложнить, если включить в нее *возможные* и *необходимые* действия, тем самым задав некую динамику развертывания ситуации. Для подобных спецификаций становится актуальной задача: определить последовательность действий, которая могла бы привести к некоторой желаемой (или не желаемой) ситуации. Подобные спецификации строятся в языках моделирования сигналов и схем. Построенную выше спецификацию можно отобразить в форме сети (Рис.3.1).

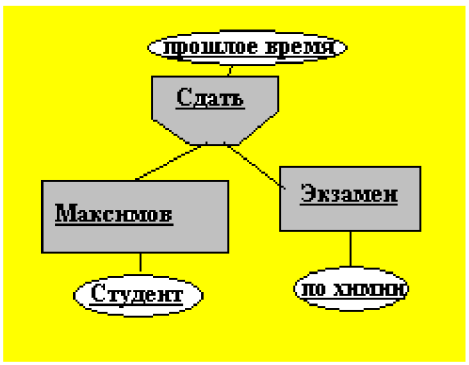

Рис.3.1

Объекты, отношения и свойства отображаются на семантической сети различными типами вершин. Ситуации могут образовывать целые сценарии. Например, рассмотрим второй фрагмент: “Экзамен по химии принимал профессор ZZZ”. Объединенная семантическая сеть представлена на рис.3.2.

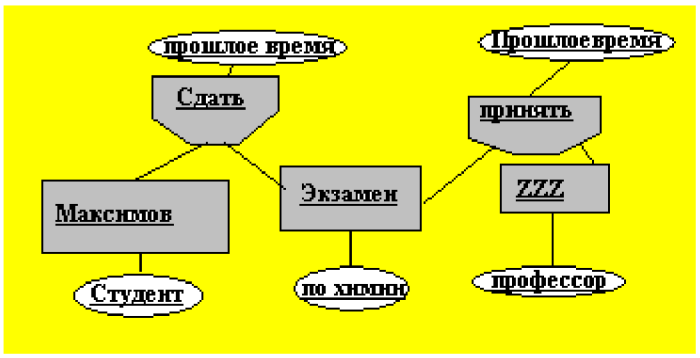

Рис.3.2

В современном программировании получили достаточное распространение текстовые документы XML. Язык XML есть удобное средство для описания сценариев. Так, “содержимое” рис.3.2 “ложится” в структуру документа XML следующим образом (русские слова заменены на английские) – рис.3.3. Документы XML обрабатываются во многих современных системах программирования, особенно в контексте Интернет-программирования.

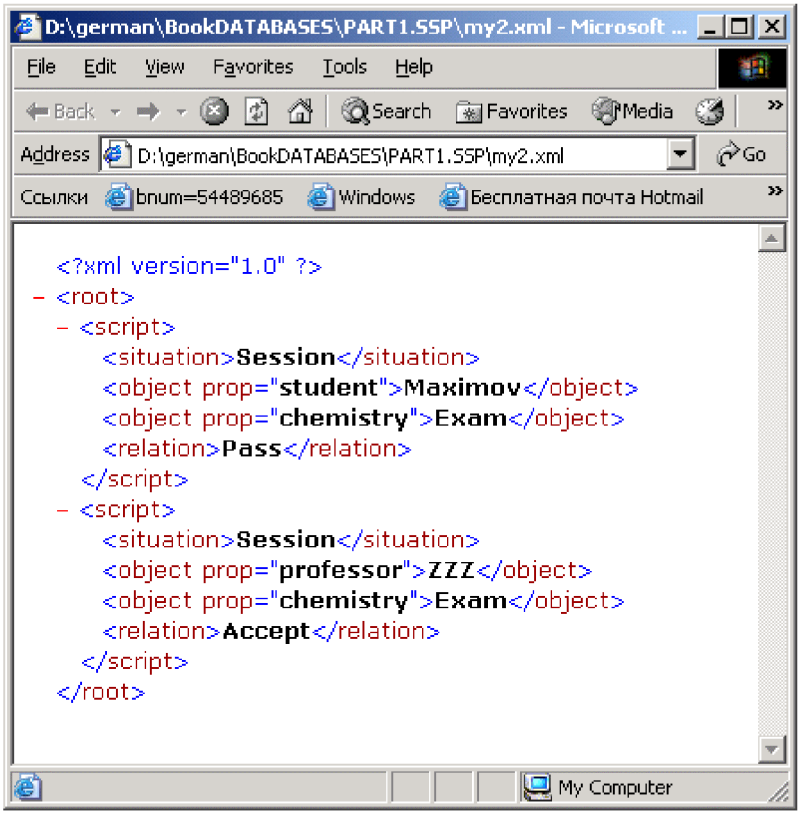

Рис.3.3

Строки XML-документа представляют собой теги и их значения. Имена тегов мы связали с объектами (object), ситуациями (situation) и отношениями (relation). Свойства передаются через атрибуты тегов. Идея использования XML-документа может состоять в

обработке запросов типа “Кто принимал экзамен по химии у Максимова?”. Для реализации этой идеи сам запрос можно перевести в XML-структуру:

<object> Maximov </object>
<object prop=”chemistry”> Exam</object>
<object> ??? </object>
<relation> Accept </relation>

Вопрос представляет фрагмент знаний. Значение ??? подлежит определению. Из контекста вопроса ясно, что это значение участвует в отношении Accept (Принять). Вторым объектом данного отношения является ZZZ. Именно это значение и должно быть выдано, поскольку других объектов у отношения Accept нет. Ясно, что ссылка на объект Maximov является наводящей. При обработке запроса система может выйти на связанную ситуацию, определяемую глаголом “Pass” (Сдать), в которой Maximov – действующее лицо. Таким образом, задача системы – выяснить, в какой мере обе ситуации связаны, чтобы считать Maximov существенной характеристикой запроса. Кроме того, при выполнении семантических запросов важно то обстоятельство, что отношения могут включать более двух объектов и не ясно, о каком объекте в запросе идет речь. Эту ситуацию пытаются развязать, используя падежные отношения объектов, передаваемые словами **кто**, **что**, **как**, **какой**, **чем** и пр. Но, опять же, более естественно искать нужный объект не с помощью падежных отношений, а через наводящие характеристики запроса. Вопросы, связанные с обработкой семантических сетей, выходят за рамки этой работы.

Сами по себе семантические сети - лишь графическое представление спецификаций ситуаций. Выгода от графического представления не столь существенна для формальной обработки (возможно, она даже препятствует эффективности обработки). Формализованные языки спецификаций дают большую выгоду для практического применения. Языков спецификаций динамических систем много, начиная от сетей Петри и автоматов и кончая языками типа HVDL и Z. Их анализ не входит в наши задачи.

Таблицы данных

Обычная таблица базы данных является моделью знаний. Например, если такая таблица содержит сведения о больных медицинского учреждения с поставленными и подтвержденными диагнозами, то на основе такой таблицы, собственно, и строят диагностическую модель. Это же касается прогнозов погоды, оценки курса акций, анализа личности по почерку и т.п. классических экспертных задач. Данные можно собирать опытным путем, а также проведением эксперимента и с помощью моделирования. Теоретическим основанием для построения моделей являются методы дискриминантного,

спектрального, регрессионного, факторного, компонентного анализов и др. В частности, многие математические пакеты, такие как MathCad, MathLab, Marple, табличный решатель MS Excel и др. позволяют строить математические модели систем, выполнять их анализ, работать с временными рядами и многомерными данными. Например, пусть даны сведения о росте цен на жилье за последние 6 месяцев и нужно дать прогноз о цене на жилье в 7 месяце. На рис.5.4 показано решение этой задачи в MS Excel (оценка качества модели в такого рода задачах играет важнейшую роль). Построены две линии тренда:

$$y = 859.63x^{0.3514}$$

$$y = 164.29x + 720$$

На основании первой из этих моделей получим прогнозную цену S1=1703.25242. Вторая модель дает прогнозную цену S2=1870.03.

Эти модели дают существенную разбежку в оценке стоимости 1 м. кв. жилья на 7-й месяц. Однако нетрудно найти лучшую из этих моделей. Ею будет та, которая дает наименьшую дисперсию адекватности оценок за предыдущие месяцы. Дисперсию адекватности можно определить по формуле

$$D_{ad} = 1/(N-1)*\Sigma^{N}_{i=1} \ (x_i - x_{imod})^2 \ ,$$

где N- число наблюденных месяцев (у нас N=6), x_i - стоимость 1-го м.кв. жилья в i-ом месяце, x_{imod} - стоимость 1-го м. кв. жилья в i-ом месяце, найденная на основе простроенной модели.

Так, модель на основе полиномиальной функции имеет дисперсию адекватности D_{ad} = 7248.73 , модель на основе линейной функции имеет дисперсию адекватности D_{ad} = 2738.105 . Из этих соображений лучше ориентироваться на линейную модель.

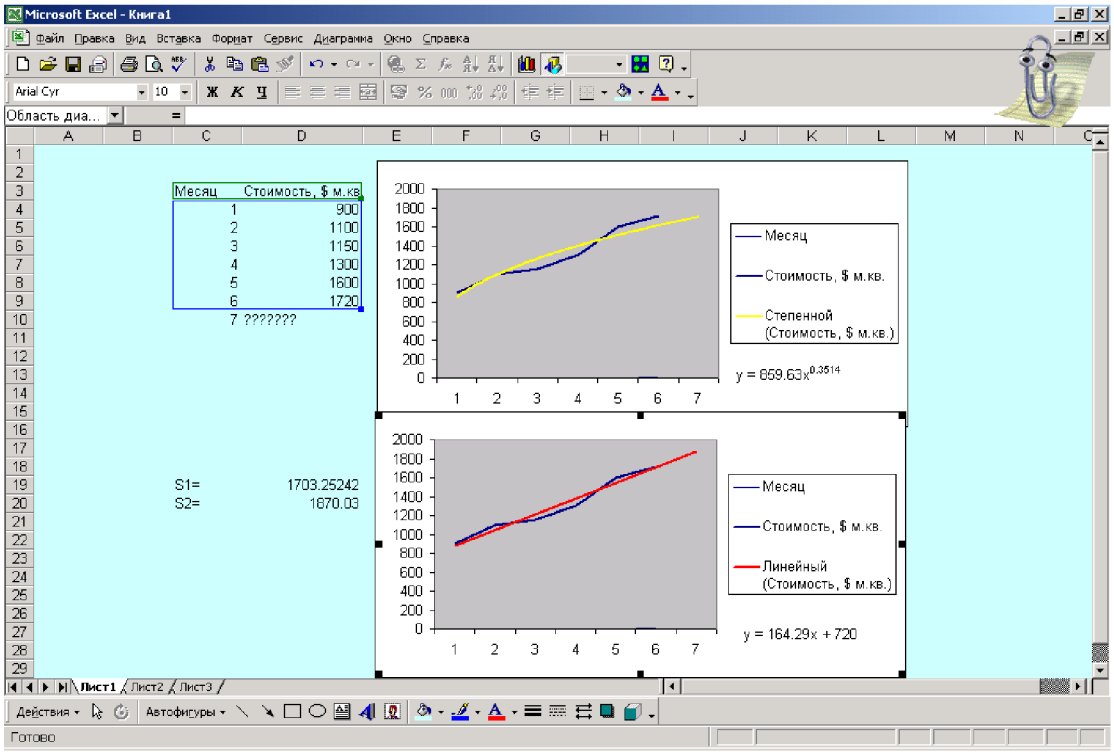
Microsoft Excel - Книга1
Файл Правка Вид Вставка Формат Сервис Диаграмма Окно Справка
Область диа...
Месяц Стоимость, $ м.кв
1 900
2 1100
3 1150
4 1300
5 1600
6 1720
7 ???????
S1= 1703.25242
S2= 1670.03
Месяц
Стоимость, $ м.кв.
Степенной (Стоимость, $ м.кв.)
y = 659.63x^0.3514
Месяц
Стоимость, $ м.кв.
Линейный (Стоимость, $ м.кв.)
y = 164.29x + 720
Лист1 Лист2 Лист3
Действия Автофигуры
Готово

Рис.3.4