ИПИАН

СИСТЕМА ОБРАБОТКИ ДЕКЛАРАТИВНЫХ СТРУКТУР ЗНАНИЙ "ДЕКЛАР" — 2

ПРЕПРИНТ

МОСКВА 1989

АННОТАЦИЯ

В ПРЕПРИНТЕ приведены подробные сведения о назначении, условиях применения и возможностях системы обработки знаний декларативного характера. Отмечается, что основным преимуществом системы по сравнению с традиционными является возможность работы на уровне логических представлений как при описании объектов, входящих в состав сложных структур, так и при описании действий над ними. Указанные достоинства реализуются посредством органично встроенных в систему специальных языков - РСС, ДЕКЛ, МОДЛ, СД и т.п.

В разработке системы ДЕКЛАР-2 принимали участие:

И.П.Кузнецов - ИПИАН СССР,

М.М.Шарнин, В.Б.Крюков - ИО АН СССР,

В.В.Пузанов - МИРЭА,

О.В.Золотарев

Содержание

Введение

ДЕКЛАР - гибкая система обработки знаний, основанная на аппарате расширенных семантических сетей (РСС). Особенность системы состоит в том, что все основные средства, функции, компоненты, обеспечивающие ее взаимодействие с пользователем (имеется в виду язык команд, языки ввода знаний, сценарии диалога и др.), поддерживаются за счет сугубо декларативных и однородных структур знаний - РСС. За счет однородности удалось значительно упростить механизмы работы над знаниями и, соответственно, минимизировать программную часть системы ДЕКЛАР. И в то же время удалось обеспечить возможность приближения системы к пользователю, ее развитие только за счет накопления структур знаний - РСС. Такое накопление не требует традиционного программирования и может осуществляться в достаточно удобных для человека формах - с помощью уже поддержанных средств. Их частным случаем являются продукции, правила языков логического программирования. Допускается также ввод знаний на ограниченном естественном языке (ЕЯ).

Система ДЕКЛАР может быть достаточно независимо использована для решения задач так называемого концептуального типа, т.е. основанных на эвристиках, знаниях, выражаемых достаточно точным способом. Имеются в виду задачи экспертных, информационных систем, а также некоторые задачи интеллектуального проектирования и программирования. Для их решения средствами ДЕКЛАР не требуется программирования в традиционном понимании - как составление наборов команд. Система ДЕКЛАР ориентирована в основном на категорию пользователей, называемую инженерами по знаниям. Она допускает возможность ввода достаточно независимых - декларативных структур, представляющих различного рода конкретные, обобщенные сведения и зачастую не имеющих какой-либо направленности. За счет этого может быть значительно уменьшен объем вводимой информации, определяющей функции конструируемой системы, облегчается отладка.

Система ДЕКЛАР может быть использована как логическое ядро для ряда прикладных пользовательских систем: диагностики, сапр, автоматического программирования, дедуктивных банков данных.

1. ОСНОВНЫЕ ПОНЯТИЯ

(Кузнецов И.П. ,Шарнин М.М. ,Пузанов В.В.)

1.1 Общие сведения о системе ДЕКЛАР

ДЕКЛАР - система нового типа, предназначенная для обработки различных видов знаний и базирующаяся на сугубо декларативной основе - расширенных семантических сетях (РСС).

ДЕКЛАР содержит:

- программное ядро, обеспечивающее обработку структур знаний;
- базу знаний (БЗ), расположенную на диске. Знания в БЗ поделены на разделы. Разделы объединяются в группы, которые записываются в файлах. В каждом файле может находиться один или несколько разделов знаний.

В существующей версии ДЕКЛАР знания вводятся посредством одного из следующих языков:

- расширенных семантических сетей (РСС);
- продукционного программирования (ДЕКЛ);
- ограниченном естественном (ЕЯ);
- модально-логического типа (МОДЛ);
- сценариев диалога (СД).

Имеется также специальный режим меню-анкет, облегчающий ввод знаний. Последние запрашиваются у пользователя, который может и не знать особенностей системы.

Язык РСС

РСС является языком низшего уровня. Знания на РСС имеют вид множеств именованых фрагментов, представляющих свойства, отношения и содержащих как константы, так и переменные. Язык РСС допускает представления следующих видов знаний:

- фактов, декларативных структур, т.е. связанных между собой сообщений, высказываний;
- продукций (правил типа ЕСЛИ ... ТО ...).

Рассмотрим пример декларативной структуры.

Пример 1.1

ОТЕЦ(X1,ИВАН) ДРУГ(X1,ПЕТР/DD1)

Это есть выражение на языке РСС, состоящее из двух фрагментов. Оно означает: отец Ивана есть друг Петра. Фрагмент ОТЕЦ(X1,ИВАН) представляет следующее: некто, обозначенный с

помощью переменной X1, является отцом Ивана. Второй фрагмент представляет, что этот некто (X1) является другом Петра. При этом второму фрагменту присвоено имя DD1. Подобные имена необходимы для представления сложных видов информации, обеспечения сложных видов обработки на так называемом "металогическом" уровне. В наличии таких имен - существенное отличие понятия фрагмента от предиката.

Язык ДЕКЛ

ДЕКЛ служит для выражения правил обработки фрагментов, входящих в состав семантических сетей. Эти правила называются продукциями. Продукции - это выражения типа "ЕСЛИ ... ТО ....".

Каждая продукция содержит левую часть (или условие) и правую часть (или следствие). Левая и правая части составлены из множества фрагментов.

Пример 1.2

ЕСЛИ КК* ОТЕЦ(X1,X2) ОТЕЦ(X2,X3) ТО ДЕД(X1,X3);

Это есть пример записи продукций на языке ДЕКЛ. Для отделения левой части от правой используются слова ЕСЛИ...ТО. (допускается применение английских аналогов IF...THEN...).

Данная продукция означает, что если X1 (некто) является отцом для X2, а X2 - отцом для X3, то X1 - это дед X3. Компонента КК* играет служебную роль. Подобные компоненты (они будут называться индикаторами) необходимы для управления применением продукций. За счет них может быть обеспечено значительное ускорение процесса применения продукций - вычислений.

Любая продукция может быть записана на РСС, тогда для отделения левой части от правой используются фрагменты специального вида (называемые сборками). Продукция записывается как множество фрагментов, порядок которых зачастую не играет какой-либо роли. Такая запись позволяет осуществить металогическую обработку без перехода на списочный уровень.

Продукции играют основную роль при реализации всех видов обработки. Программное ядро обеспечивает применение продукций, записанных на РСС, к другим РСС, т.е. к фактам, декларативным структурам и другим продукциям.

Итак, вся обработка, которая сводится к применению продукций, осуществляется на уровне РСС.

За счет специальных средств (индикаторов, операторов вызова) одни продукции могут активизировать другие. Программным ядром делаются последовательные попытки применения только активизированных продукций. При этом применение отдельной продукции к некоторой РСС сводится к проверке условия (т.е. наличия в этой РСС фрагментов, аналогичных фрагментам левой части продукции). Если проверка удовлетворяется, то продукция считается применимой. Тогда выполняются действия, записанные в ее правой части. В простейшем случае к РСС просто добавляется такая часть, т.е. фрагменты, входящие в следствие. В сложных

случаях такое применение может приводить к активизации других продукций, к обращению к специальным программам (через "встроенные" фрагменты или предикаты) и т. д.

Система ДЕКЛАР последовательно пробует применить все продукции, кот орые находятся в оперативной памяти (ОП) и которые активизированы. Продукции применяются к фрагментам, также находящимся в ОП. Итак, для обеспечения работы системы соответствующие разделы знаний должны быть считаны в ОП.

Система ДЕКЛАР имеет необходимые средства для гибкого управления применением продукций. Для этого в левую часть каждой продукции вводится упоминавшийся ранее индикатор. Множество продукций может иметь один и тот же индикатор. Последний может быть активизирован. Применяются только продукции с активизированными индикаторами. За счет индикаторов обеспечивается деление продукций на модули с выполнением в каждый момент времени действий одного модуля. Порядок включения модулей задается с помощью продукций, активизирующих индикаторы других продукций. Таким образом реализуется идея двойной черной доски (DOUBLE BLACK-BOARD).

Порядок применения продукций определяется последовательностью их активизации, которая задается с помощью самих же продукций, и порядком расп оложения. Активизированные продукции (с одним и тем же индикатором) пробуют применяться в порядке их записи, т. е. вначале первая, затем - вторая и т. д. Работа заканчивается, когда исчерпаны все варианты применения всех имеющихся в ОП продукций, т. е. в ОП нет применимых продукций. Работа возобновляется при поступлении новой информации на РСС:

- при считывании очередного раздела знаний с диска в ОП;
- при вводе в ОП соответствующей информации с клавиатуры терминала.

ДЕКЛ и МОДЛ - это языки высокого уровня.

Язык ДЕКЛ служит для записи продукций, обеспечивающих решения пользовательских задач, а также реализацию системных функций. Продукции записываются в форме, приближенной к нотации языка ПРОЛОГ. Язык ДЕКЛ предназначен для системных программистов и инженеров знаний. Программы, написанные на языке ДЕКЛ, при считывании с диска в ОП автоматически транслируются на РСС, где осуществляется их выполнение программным ядром.

Ранее был приведен один пример продукции на языке ДЕКЛ. Рассмотрим другой пример.

Пример 1.3

```
ЕСЛИ INFIL D:FILZN(X) N:BEG(X) ТО B:A(X);
```

В данной продукции используются операторы D: (удалить) N: (нет) и B: (выполнить), которые стоят перед фрагментами.

Продукция будет применяться, если она активизирована (через индикатор INFIL). Тогда в знаниях, находящихся в ОП, ищется

фрагмент вида FILZN(...). Если он есть и нет фрагмента BEG(...) с тем же аргументом, тогда продукция делается применимой. Из этих знаний удаляется - FILZN(...) и выполняется процедура, связанная со встроенным фрагментом A(...). Встроенные фрагменты служат для обращения к программам ввода, вывода и т.д. С их помощью осуществляется ввод в ОП новой информации, вывод результатов логической обработки и др.

Язык СД

Язык СД служит для программирования сценариев диалога, т.е. задания акций взаимодействия системы ДЕКЛАР с пользователем. Он вводится как некоторое подмножество языка РСС, интерпретируемое специальным образом (процессором, основанном на знаниях). Конструкции языка СД - это сугубо декларативные структуры, составленные из фрагментов. С их помощью указывается порядок выдачи меню-анкет, а также ответные действия системы на реакцию пользо вателя. Этот язык предназначен для инженеров по знаниям, разрабатывающим прикладные пользовательские системы. На языке СД заданы сценарии, по которым осуществляется взаимодействие системы ДЕКЛАР с пользователями.

Пример 1.4

```
AFT(TBL1,"1",-,INF1)     KTL(MENU,INF1)
AFT(TBL1,"6",INDEL*,INF2)
```

Здесь проиллюстрированы простейшие конструкции языка СД. Предикаты AFT(...) интерпретируются следующим образом: если система находится в состоянии TBL1 и пользователь нажал клавишу "1", то система должна перейти в состояние INF1. Если же пользователь нажал клавишу "6", то осуществляется выполнение процесса INDEL* (применение продукций с индикатором INDEL*), после чего система переходит в состояние INF2. Предикат KTL(MENU,INF1) означает, что при переходе в состояние INF1 на экран должна быть вызвана таблица с именем INF1, находящаяся в файле MENU.Z.

Язык СД будет описан в п.5.2. Он обладает средствами задания достаточно сложных сценариев, которые могут быть разбиты на самостоятельные части - подсценарии. При этом от пользователя может требоваться нажатие отдельных клавиш, а также ввод множеств слов и фрагментов. Путем обращения к тому или иному процессу последние могут быть подвергнуты специальному анализу, определяющему последующую реакцию системы.

Язык МОДЛ

Язык МОДЛ предназначен для инженеров по знаниям. Язык МОДЛ служит для ввода декларативных сведений, обобщенной информа ции, имеющей оттенок долженствования - "должен быть" или "может быть". За счет таких сведений осуществляется проверка полноты и непротиворечивости знаний, вводимых в систему ДЕКЛАР, организуется ее активность - с целью выявления у пользователей полной и непротиворечивой информации о той или иной предметной области. Подобные действия осуществляются в специальном режиме - наводящих запросов (см.п.1.2).

Пример 1.5

ЕСЛИ ЧЕЛ(X) МУЖ(X) ТО MUST:РАБ(X,X1) ?:X1;

Это пример выражения на языке МОДЛ. Оно означает, что если X - человек (ЧЕЛ) и мужчина (МУЖ), то он должен где-то работать (РАБ). Запись ?:X1 означает, что нужно поинтересоваться местом работы. Подобные выражения интерпретируются специальными продукциями.

Ограниченный естественный язык (ЕЯ)

Ограниченный естественный язык может быть использован при вводе запросов и сообщений, формулируемых с помощью фиксированного набора слов. При этом в специальном режиме меню-анкет обеспечивается ввод новых слов - имен объектов, понятий, а также слов, выражающих отношения, действия с указанием управляемых ими форм. Допускается использование связок типа "тот,который", причастных оборотов, анафорических ссылок. При отсутствии тех или иных лингвистических знаний организуется системная активность. Задаются встречные вопросы, ответы на которые также можно давать на естественном языке.

С помощью подобных средств система ДЕКЛАР может быть настроена на язык той или иной предметной области. При формировании запросов и сообщений допускается широкое использование слов - кванторов ("каждый", "все", "есть", "некоторые").

Для обработки выражений ЕЯ требуется их предварительная трансляция на язык РСС, где уже осуществляются необходимые проверки, поиск информации и т. д.

Трансляция выражений ограниченного ЕЯ осуществляется за счет специальных знаний - лингвистических, состоящих из продукций и декларативных структур. Трансляция выражений начинается, как только такие знания вызваны в ОП и настроены на считывание информации из источника, где набрано то или иное выражение в требуемой форме. После этого продукции сами обеспечивают считывание выражений в ОП и вызывают соответствующие преобразования.

Работа транслятора или лингвистического процессора поддерживается только знаниями, т. е. не требуется каких-либо дополнительных программных средств. За счет введения таких знаний обеспечиваются достаточно широкие возможности в плане расширения входных языков (МОДЛ, ЕЯ) их дополнения новыми изобразительными средствами. Причем объем лингвистических знаний оказывается сравнительно небольшим, что предопределяет относительно невысокие трудозатраты при создании трансляторов.

Здесь использована новая методика построения лингвистических процессоров, основанная на декларативно-продукционных принципах. В рамках упомянутой методики обеспечивается создание и поддержка других внешних языков, предназначенных для той или иной категории пользователей. Такие языки могут в значительной степени быть приближены к естественным формам выражения знаний.

Деклар обладает гибкой системой команд, которые вводятся с терминала (в языке РСС) и обрабатываются продукциями из специального раздела знаний. С помощью продукций задается, что нужно делать при поступлении той или иной команды.

Как уже говорилось, знания в системе ДЕКЛАР делятся на разделы. Под разделом понимается достаточно произвольный набор выражений какого-либо языка, ограниченный специальными метками начала и конца, см. П.3.3. Это могут быть выражения языков РСС, ДЕКЛ, а также выражения МОДЛ, ограниченного ЕЯ, всевозможные меню-анкеты, таблицы и т.д. При этом различаются:

- знания, непосредственно готовые к использованию (это выражения языков РСС, МОДЛ, а также меню-анкеты, которые просто нужно вызвать на экран);

- знания, требующие предварительной трансляции (выражения языков МОДЛ и ограниченного ЕЯ).

Каждый раздел и меет свое имя. Все разделы знаний хранятся на диске вместе с программным ядром системы ДЕКЛАР. Разделы хранятся в файлах, имеющих расширитель ".Z". В каждом таком файле может содержаться один или множество разделов.

Например, PROD.Z - есть файл, в котором хранится множество разделов знаний с именами SYS, SD, SDD, ..., поддерживающих системные функции. В файле UZER.Z находятся разделы, обеспечивающие решение ряда пользовательских задач. KTL.Z - есть файл , в котором находится лишь один раздел знаний с именем KTL (каталог), MENU1.Z и HELP.Z - файлы, в которых находятся меню-анкеты. Каждое такое меню организовано как раздел со своим именем - имя является обозначением меню. Например, TBL1 - есть имя "главного меню", которое выдается в самом начале работы.

Различаются разделы знаний следующих видов:

- общесистемные (поддерживающие инструментальную часть, т.е. внешний язык, систему команд и др.);

- интерфейсные (в них содержатся тексты справочной информации, таблицы, разного рода меню);

- пользовательские (они необходимы для решения задач пользователей).

Общесистемные разделы системы (кроме KTL) хранятся в файлах PROD.Z и USER.Z. С помощью них организуется сценарий диалога, то есть находящиеся в них фрагменты и продукции определяют, в какой последовательности выдаются меню, анкеты, какие при этом возможны реакции пользователя и т.д.

В файле KTL.Z находится каталог разделов знаний. Такой каталог состоит из набора фрагментов вида KTL(<имя раздела>,<имя файла>), которые оформлены как самостоятельный раздел с именем KTL. Например, KTL(TBL1,MENU1) означает, что раздел TBL1 находится в файле MENU1.Z. Подобные фрагменты встречаются и в других общесистемных разделах.

Систем а /ДЕКЛАР реализована на разнотипных ЭВМ (IBM/PC, ЕС-1840, ДВК-2, СМ-4), имеющих различную клавиатуру и свои особенности включения в работу, вызова операционной системы и др. Будем считать, что такие особенности известны пользователю. Будем обозначать клавишу "ввода" (которая в IBM/PC называется ENTER, в СМ-4 - RETURN, а в ДВК-2 - <ВК>) через ENTER. Будем отличать символы готовности к выполнению команд, выдаваемые операционной системой (для "ОС РАФОС - это точка, для ОС_РВ СМ-4 - ">" ,а для IBM/PC - ":\>"), от приглашения, выдаваемого системой ДЕКЛАР (это символы *>, **>, и +>).

Для работы системы ДЕКЛАР необходимо иметь на диске файл DD.EXE, в котором находится программное ядро DECLAR, а также файлы DEKLAR.ZNN, KTL.Z и USER.Z. Это минимальный набор файлов, обеспечивающий вызов главного меню и обращениие к его пунктам,

но без решения связанных с ними задач. Для такого решения требуется иметь поддерживающие их разделы знаний.

1.2. Общесистемные клавиши

Для вызова программы необходимо набрать на клавиатуре - DD и нажать ENTER. Сверху экрана появится информация об "общесистемных клавишах". Последние позволяют пользователю включать специальные режимы работы, возвращаться к меню более высокого уровня, получать справочную информацию. После того, как система выдала очередное приглашение вида *> или **> (но не +>), вы можете набрать любую из указанных клавиш и нажать ENTER. Тогда произойдет следующее.

При нажатии "С" будет осуществлен переход (в рамках имеющегося сценария диалога) к меню более высокого уровня, то есть откуда был переход в текущее состояние.

Клавиша "Т" вызывает включение режима простой трассировки. На экран будут последовательно выдаваться все продукции, которые оказались применимыми. Здесь можно увидеть процесс вычисления, решение задачи. При этом на местах переменных уже будут стоять означивающие их константы. Этот режим позволяет отлаживать программы на продукциях. При этом следует помнить, что продукции, у которых имена (или индикаторы) начинаются с символа @, относятся к общесистемным - на них не следует обращать внимания.

Знакосочетание ==<ИМЯ>== означает окончание применения продукции с указанным именем (это может быть индикатор).

Клавиша "F" вызывает включение режима полной трассировки. На экране вы увидете всю работу, которая выполняется системой ДЕКЛАР - продукции, которые оказались применимыми, а также продукции, которые не смогли примениться. При этом перед фрагментами, которых не оказалось в знаниях (из-за чего продукция стала не применимой) будет стоять знак вопроса - "?". В применимой же продукции правая часть (после "ТО") будет перенесена на новую строку.

Клавиша "D" - вызывает снятие режима трассировки.

При нажатии клавищи "I" будет выдан запрос: "Укажите раздел знаний *>". Указанный вами раздел будет считан с диска и введен в ОП.

Нажатие клавиши "P" вызывает выдачу информации об имеющихся разделах знаний с указанием файлов, в которых они находятся.

Клавиша "L" необходима для получения информации. Содержимое указанного вами раздела будет выдано на экран.

Клавиша "E" - это обращение к экранному редактору. При . ее нажатии будет выдан запрос "Укажите раздел знаний *>". После такого указания нужно нажать ENTER. Система обратится к экранному редактору и вызовет в соответствующее окно файл, в котором находится указанный вами раздел.

Клавиша "K" вызовет переход в режим выполнения команд. Любая команда из имеющегося перечня (см. Прилохение 1) будет выполнена системой. В режиме команд можно вызывать в ОП необходимые разделы знаний, активизировать продукции, выдавать их на экран, - то есть управлять всей работой по решению той или иной задачи. Однако во многих случаях это легче делать и без перехода в режим команд - за счет общих ключей и встраивания задачи в сценарий диалога (см. ниже).

"H" - клавиша помощи. При ее нажатии выдается справочная информация (если таковая предусмотрена в текущем состоянии).

Примечание. Описанные выше действия, связанные с клавишами "T","F","D","E" и "H", могут быть вызваны и при отсутствии приглашения +> или **>, то есть в любом месте, в том числе и в процессе работы. Это осуществляется одновременным их нажатием с клавишей CTRL. Например, одновременное нажатие CTRL и "T" (обозначается CTRL/T) приведет к включению режима простой трассировки. Здесь имеются и дополнительные возможности:

CTRL/O - вызывает выдачу "окрестности" указанной вами константы. При одновременном нажатии этих клавиш появится приглашение "укажите вершину (константу) +>". Для набранной константы будут выданы на экран все фрагменты, в которые она входит и которые в текущий момент находятся в ОП. Это необходимо в процессе отладки программ, написанных на языке ДЕКЛ.

1.3. Как начинать работу

Вызов программы осуществляется нажатием клавиши DD и ENTER. На экране появится - "Нажмите ENTER". Такое нажатие означает переход к главному меню. (нажатие "S" - переход к диалогу на ЕЯ). На экране появится главное меню (TBL1), например, такого вида :

Основные функции

- примеры экспертных систем (1)
- режим активного выявления сведений (2)
- расчетно-логические задачи (3)
- фрагменты диалога на естественном языке (4)
- конструирование экспертных систем (5)

Специальные функции

- стандартная типовая задача (6)
- режим отладки задач (7)
- описание системы ДЕКЛАР (8)

Примечание. С пунктом (2) связан язык МОДЛ. С помощью его выражений осуществляется управление режимом активного диалога. Пункт (4) поддерживается пробным лингвистическим процессором, отличным от того, который вызывается по клавише "S". Пункт (5) пока не поддерживается знаниями.

Вы можете выбрать любой из интересующих вас пунктов. Соответственно, будет осуществлен переход в другое меню и т.д. При этом с диска в ОП будут вызываться соответствующие разделы знаний, необходимые для решения той или иной задачи (такие вызовы на цветном дисплее высвечиваются красным). Будет иметь место так называемая "динамическая подкачка" знаний. Возврат - по клавише "C".

Следует помнить, что для решения задач, связанных с тем или иным пунктом меню, нужно иметь на диске файлы с необходимыми разделами знаний. Они предоставляются при передаче системы ДЕКЛАР в полном комплекте. Например, с п.(2) связаны разделы, находящиеся в файлах AD.Z и CLASS.Z. В последнем хранятся выражения языка МОДЛ. Дополнив эти выражения или создав новый раздел, можно легко настроить систему на диалог в новой предметной области.

Как начать решение вашей задачи

Для этого необходимо иметь программу, написанную на языке ДЕКЛ, оформленную как раздел знаний, см.п.2.3. Этот раздел создается с помощью экранного редактора и помещается в соответствующий файл. Пусть, например, ваша задача решается с помощью средств (продукций и фрагментов), оформленная как раздел с именем AAA, который помещен в файл BBB.Z. Пусть само решение осуществляется путем активизации (оператором T1:) ваших продукций с индикатором CCC*.

Вначале необходимо включить вашу задачу в сценарий диалога. Пусть вы хотите включить ее в пункт, связанный с отладочным режимом. Имеется в виду таблица DEBO, переход к которой осуществляется по п.7 "главного меню". Тогда в файле MENU1.Z находится эта таблица. В ней заводится новая строка, которой сопоставляется своя цифра. Пусть это цифра 6. В строку помещается рубрикарий (имя) вашей задачи. Далее с помощью экранного редактора вызывается файл USER.Z, где в языке СД задан сценарий диалога. Вам нужно дополнить этот сценарий. Для этого вы находите в этом файле фрагменты вида:

```
AFT(DEBO,"1",....)
AFT(DEBO,"2",....)
```

Соответственно, они и указывают, что нужно делать, если в меню DEBO выбран пункт "1","2" и т.д. К ним добавляются новые фрагменты, осуществляющие вызов вашей задачи (порядок их расположения не играет роли):

```
AFT(DEBO,"6",-,EE1)
AFT(EE1,%,CCC*,DEBO)
TIE(EE1,AAA,ADD)   KTL(AAA,BBB)
```

где E1 - любая ранее не используемая константа, которая соответствует новому состоянию в сценарии диалога, связанному с вашей задачей. Первый фрагмент указывает, что когда на экране высвечено меню, то при нажатии клавиши "6" система перейдет к вашей задаче. Второй фрагмент приведет к однократной активизации ваших продукций - с индикатором CCC*, то есть до первого применения. После этого снова будет выдано меню DEBO. Нижние фрагменты - TIE(...) и KTL(...) говорят, что при переходе в состояние EE1, (т. е. к вашей задаче) нужно вызвать (ADD) из файла BBB.Z раздел AAA. Если таких разделов несколько, то должно быть и несколько подобных пар фрагментов.

Далее все это запоминается в файле UZER.Z и осуществляется вызов системы - нажатием DD. При переходе от главного меню по п.(7) и затем - по вновь созданному пункту вы увидите результат (а в трассировке - процесс решения вашей задачи).

Например, попробуйте таким образом ввести продукцию:

```
ЕСЛИ CCC* ТО B:A("****");
```

Вы увидите на экране четыре звездочки (на их месте может быть любой записанный вами текст).

Если вы хотите, чтоб ваши продукции вызывались (активизировались) многократно, то вставьте еще один фрагмент - PASS(CCC*).

Аналогичным образом ваша задача может быть включена в любые другие меню, кроме главного (TBL1). Только вместо DEBO следует выбрать другую таблицу и проделать ту же работу. Если ваша задача имеет свои разветвленные пункты меню, то ее следует оформить в виде подсценария, см. п. 5.2. Для этого требуется более детально знать, как устроен язык СД.

2. ЯЗЫК РАСШИРЕННЫХ СЕМАНТИЧЕСКИХ СЕТЕЙ - РСС
(Кузнецов И.П., Шарнин М.М.)

2.1. Основные компоненты языка

Этот язык служит для ввода фактов и декларативных сведений, связанных с решением задач. Он может быть использован и для ввода продукций. Однако, для этих целей лучше использовать язык более высокого уровня - ДЕКЛ.

Компоненты языка РСС

1) Основные символы:

- буквы (латинские A - Z, русские А - Я), большие и малые;
- цифры (0 - 9);
- знаки пунктуации (пробел, запятая, ; " : / ( ) [ ], а также код клавиши "ВК" или ENTER);
- знаки операций (+ = * < > # ?);
- спецсимволы (- . & % _).

В новейших версиях операционных систем допускается применение как больших, так и малых букв (они считаются различными символами - в стандарте ASCII они имеют свои коды).

2) Константы.

Константа - это последовательность букв, цифр, знаков, операций и спецсимволов, которые начинаются не с буквы Х (ЛАТ. или РУС.) и не с цифры.

Итак, константа не может содержать знаков пунктуации. Числа не являются константами, т.к. Они начинаются с цифры. В то же время отдельные знаки операций или их сочетания - это константы.

Примеры констант:

A12, **, **15, ИВАН1, БРАТ, JANE-1, F125, +, <, >, = .

Константы служат для обозначения объектов, понятий и др.

П р е д о с т е р е ж е н и е: Не следует использовать в качестве констант отдельные латинские буквы (V, S, P, L, N, B, T). Константа (кроме знаков операций) должна содержать более одного символа. Запрещается использование в качестве констант системных (служебнных) имен, см. Приложение 3.

3) Строковые константы.

Строковая константа -это последовательность символов, которая обрамляется кавычками (").

Примеры строковых констант:

"15","А12","ИВАН","++1","Х1","ВВЕСТИ ТЕКСТ".

В строковых константах допускаются знаки пунктуации, операций, а также спецсимволы.

Если в начале и в конце любой константы или переменной поставить кавычки, то образуется строковая константа.

Строковые константы - это ничего не обозначающие последовательности символов. Напрмер, "ИВАН1" рассматривается системой как последовательность букв. В то же время ИВАН1 - это уже обозначение об'екта.

4) Каждая строковая константа есть константа.

5) Числа.

Число - это последовательность цифр. Например, 2, 345, 824, 1286 - это числа. Допускаются только целые положительные числа от 0 до 32000. Естественно, между цифрами не может стоять каких-либо знаков пунктуации - точек, запятых и т.д.

6) Каждое число есть константа.

7) Знаки арифметических операций:

+ - сложить	<= - меньше или равно
* - умножить	>= - болъше или равно
М- - вычесть	= - равно
DIV - разделить (цело-численное деление)	# - не равно
< - меньше	> - больше

8) Каждая арифметическая операция есть константа.

9) Внутрисистемные коды (имена).

Это числа от 0 до 99, к которым добавляется знак плюс (+) (когда вводится новое имя) или знак минус (-) (когда используется уже введенное имя). Такие числа играют роль имен или обозначений, порождаемых самой системой, к примеру, когда генерируются новые знания или требуется представить не обозначенный объект.

Например, 1+ и 1- - есть обозначение одного и того же объекта (или отношения), а 2+ и 2- - уже другого, и т.д. Если еще раз встретится 1+ , то это будет уже обозначение третьего объекта (знак + означает - появился новый объект). Все

последующие 1- будут соответствовать третьему объекту. Следует отметить, что хотя первый и третий объекты обозначены одинаково 1+ , однако система всегда будет отличать один от другого. С каждым плюсом, следующим после числа, будет заготавливаться свое представление.

Количество внутрисистемных имен, сопоставляемых объектам, может быть достаточно большим, и ограничивается только объемом памяти ОП, выделяемой под знания.

10) Каждое внутрисистемное имя есть константа.

Итак, в системе ДЕКЛАР последовательности цифр используются:

- для представления чисел;
- в качестве внутрисистемных имен или кодов (тогда за последней цифрой ставится знак "+" или "-").

11) Переменные.

Переменная - это последовательность цифр, перед которой ставится заглавная буква X (лат.).

Примеры переменных:

X, X1, X21,...,X99

Переменные служат для обозначения неизвестных обьектов, отношений и др.

12) Простейшие фрагменты.

Они составляются из констант и переменных следующим образом. На первом месте ставится константа или переменая, называемая именем отношения. Это может быть и знак арифметической операции. На следующих местах (в круглых скобках) ставятся константы или переменные , называемые аргументами отношения. Они отделяются запятыми и представляют (или обозначают) объекты, участвующие в отношении.

Пример 2.1

ДЕД(ИВАН1,ПЕТР1) - фрагмент, представляющий отношение "ДЕД" между объектами (людьми), обозначенными через ИВАН1 и ПЕТР1.

Пример 2.2

МУЖ(X1,МАША1) - представляет, что некоторый неизвестный объект (человек) X1 является мужем для МАША1. Здесь МУЖ - имя отношения.

Для простеших фрагментов выбрана предикатная форма записи. Это не случайно, так как простейший фрагмент представляет ту же информацию, что и элементарные предикаты в логике. Однако в общем случае понятие фрагмента шире, чем предиката, что будет показано ниже.

Простейшие фрагменты могут иметь различную местность (в данной версии системы она не должна превышать числа 5). Это позволяет представлять отношения достаточно высокой арности. Выше были проиллюстрированы двуместные фрагменты, представляющие бинарные отношения. Рассмотрим другие примеры.

Пример 2.3

ШКОЛ(МАША1)- одноместный фрагмент, представляющий, что МАША1 является школьницей. В качестве имени отношения использована константа ШКОЛ, означающая - быть школьницей. Здесь представлено свойство, которое рассматривается как унарное отношние.

Пример 2.4

X1() - нульместный фрагмент, где именем отношения является переменная. Такая запись допускается в РСС.

Пример 2.5

ИМЯ("ИВАН",1+) - двуместный фрагмент, представляющий, что строковая константа "ИВАН" именует (имя отношения - ИМЯ) новый объект, которому сопоставлен внутрисистемный код 1+.

Пример 2.6

+(27,X1,30) - означает 27 + X1 =30

Следует отметить, что деление на объекты и отношения является весьма условным. Допускаются случаи, когда на аргументных местах в каких-либо фрагментах ставятся имена отношений, или же обозначение объектов используется в качестве имен отношений. Однако, желательно, чтобы при этом не терялась содержательная интерпретация.

Среди множества имен отношений выделяются системные, которые интерпретируются и обрабатываются программным ядром. Это константы, которые записываются в виде одиночных букв - латинских или русских. К ним относятся обозначения операторов (D, T, B, N), имя продукционного отношения (P), отношения "часть-целое" (S), а также имена встроенных фрагментов и спецфрагментов. В связи с этим использование отдельных букв в качестве несистемных имен отношений следует избегать.

13) Именованые фрагменты.

Они включают в себя те же компоненты, что и обычные фрагменты. Но в конце добавляется часть, служащая для именования всей конструкции. В качестве имени используются константы или переменные. Эта часть записывается после всех аргументов и отделяется от них косой линией.

Пример 2.7

ДЕД(ИВАН1,ПЕТР1/D1)

Это фрагмент, поименованный константой D1. Константа D1 обозначает "комплексный" объект, состоящий из предыдущих (ИВАН1,ПЕТР1) с учетом их отношения.

Пример 2.8

ШКОЛ(МАША1/X1)

Это фрагмент, поименованный переменной X1. Она ставится в том случае, когда имя фрагмента неизвестно, но существенно.

Пример 2.9

FFF(/2+)

Это нуль-местный фрагмент с именем 2+ (внутрисистемным кодом.

Следует отметить, что два фрагмента обязательно должны иметь различные имена. Имя - это уникальная характеристика фрагмента (его "код"), которая необходима по следующим соображениям.

Во-первых, именование фрагментов необходимо, когда объекты и связывающее их отношение рассматриваются как единое целое, что было проиллюстрировано в приведенных ранее примерах. Тогда это целое может быть обозначено своей константой или переменной, которая может участвовать для представления отношений этого целого с какими-либо другими объектами. Это содержательная сторона.

Во-вторых, именование фрагментов необходимо с точки зрения поиска, обработки - для правильной работы программного ядра. Поэтому при вводе простейших фрагментов они автоматически "именуются", т.е. становятся именованными. При этом в качестве имени берется внутрисистемный код. Например, при вводе фрагмента ШКОЛ(МАША1) будет автоматически сформировано ШКОЛ(МАША1/1+), где 1+ - очередное внутрисистемное имя.

14) Каждый простейший фрагмент есть фрагмент.

15) Каждый именованый фрагмент есть фрагмент.

Следует отметить, что понятие фрагмента шире, чем предиката по трем соображениям:

во-первых, во фрагментах имя отношения в некотором смысле равнозначно именам объектов - на соответствующих местах могут стоять те же самые константы или переменные;

Во-вторых, в предикатах отсутствуют средства их именования, которые есть во фрагментах;

В третьих, понятие фрагмента, в отличии от предиката, никак не связано с логическим значением "истинности" и "ложности", хотя такие значения всегда могут быть введены во фрагмент - для них может быть выделено свое аргументное место.

16) Символ "_" (нижняя черточка).

Он ставится во фрагментах, входящих влевую часть продукции,

на места, которые не существенны или не играют роли. Этот символ есть переменная специального назначения, которая в процессе применения продукции не означивается, см. п. 4.2.

Пример 2.10

ДЕД(ИВАН1, _)

Здесь представлено, что ИВАН1 является ДЕДом - не важно кому.

Символы "_" могут стоять на различных аргументных местах, в том числе, и на месте имени отношения. Такие символы могут соответствовать различным объектам или отношениям.

Пример 2.11

ДРУГ(_, _/DD)

Это именованый фрагмент, который означает : кто-то кому-то является другом. Это могут быть разные люди.

Пример 2.12

_(ИВАН1, ПЕТР1).

Этот фрагмент представляет, что ИВАН1 и ПЕТР1 как-то между собой связаны, но не важно как.

Еще раз отметим, что символы "_" могут использовать только во фрагментах, составляющих левую часть продукции. При этом символ "_" не может стоять на месте имени фрагмента. Например, запись типа R1(X1/_) недопустима. В этом случае следует использовать другую эквивалентную запись - R1(X1).

Формально, понятие фрагмента определяется следующим образом:

<имя отношения> --> <константа> \ <переменная>
<имя фрагмента> --> <константа> \ <переменная>
<аргумент> --> <константа> \ <переменная>

<простейший фрагмент> -->

<имя отношения>() \ <имя отношения>(<аргумент>) \

<имя отношения>(<аргумент>,...,<аргумент>)

<именованый фрагмент> -->

<имя отношения>(/<имя фрагмента>) \

<имя отношения>(<аргумент>/<имя фрагмента>) \

<имя отношения>(<аргумент>,..., <аргумент>/<имя фрагмента>)

<фрагмент> --> <простейший фрагмент> \ <именованый фрагмент>

В данном определении косая черта \ играет роль разделительного "ИЛИ".

Следует отметить, что для записи констант, обозначающих объекты или отношения, лучше использовать лишь заглавные буквы - латинские или русские (в данной версии системы деклар использованы заглавные латинские буквы). Это позволяет избежать многих ошибок, вызванных привычкой инвариантного использования больших и малых букв (напомним, что в системе ДЕКЛАР - это различные символы).

2.2. Понятие сети

Сеть (РСС) - есть один или множество фрагментов, которые отделяются пробелами :

<сеть> --> <фрагмент> \ <фрагмент> <фрагмент> ... <фрагмент>

Пример 2.13

ШКОЛ(МАША1) ШКОЛ(МАША2)

Это есть сеть, представляющая, что МАША1 и МАША2 являются школьницами.

Если сеть состоит из фрагментов, которые содержат только константы, то она представляет набор фактов, как это имело место в примере. Фрагменты, в которых содержатся одни и те же переменные, представляют связанную информацию. Порядок записи фрагментов в сети с точки зрения содержательной интерпретации не играет какой-либо роли.

Пример 2.14

ДЕД(X1,ПЕТР1) БРАТ(X1,ЛЕНА1)

Это сеть, которая представляет, что некто (X1) является дедом для ПЕТР1 и братом для ЛЕНА1. Здесь переменная X1 как бы связывает обе части сообщения. Это есть пример декларативной структуры.

Пример 2.16

МУЖ(ИВАН2,ЛЕНА2/3+) ДРУЖНАЯ(3-)

Это сеть, в которой первый фрагмент представляет, что ИВАН2 и ЛЕНА2 являются мужем-женой. Этой паре сопоставляется внутрисистемный код 3+. При его повторном применении он трансформируется в 3-. Второй фрагмент представляет, что упомянутая пара обладает свойством ДРУЖНАЯ. Итак, 3+ и 3- обозначают одно и то же.

Пример 2.17

*(2,X1,X2) +(X2,34,88)

Это означает, что 2*X1+34=88. При этом X2 сопоставляется результату операции умножения, т.е. 2*X1=X2, X2+34=88.

На языке РСС может вводиться любая информация, в том числе бессмысленная. Поэтому при ее вводе важно иметь в виду содержательную интерпретацию.

Если в расмотренных примерах переставить фрагменты, то интерпретация сети не изменится. Следует отметить, что порядок записи фрагментов в сети не играет роли только с точки зрения интерпретации. И в то-же время такой порядок важен с точки зрения обработки - вначале обрабатываются (ищутся, анализируются) фрагменты, которые стоят первыми. Сказанное в особой степени относится к продукциям.

2.3. Оформление разделов знаний на РСС

Напомним, что выше было отмечено отличие интерфейсных разделов знаний от пользовательских и системных. Первые - это меню, анкеты, таблицы, которые находятся на диске и которые только время от времени нужно выдавать на экран. Их нет необходимости держать в ОП.

Вторые тоже находятся на диске. Но по мере необходимости их требуется считывать с диска в ОП, изменять, снова записывать на диск. При этом в ОП одновременно может находиться множество разделов, которые приходится отделять один от другого.

В связи с различным функциональным назначением оформление интерфейсных разделов п роще, чем пользовательских и общесистемных. Остановимся на таком оформлении.

Любой раздел, находящийся на диске, начинается с так называемой "метки раздела", т.е. с одноместного фрагмента вида $(<раздел>), где <раздел> - это константа, состоящая только из латинских букв и цифр. Она является именем раздела.

Заканчивается раздел символом $. Символы $ обязательно должны стоять в начале строки - на первой позиции. У общесистемных и пользовательских разделов в начало и в конец (перед $) вводятся дополнительные метки - фрагменты вида BEG(<раздел>) и END(<раздел>).

С их помощью разделы, считанные в ОП, отделяются один от другого.

Пример 2.18

```
$(TBL1)
  .
  .
  .
$(GRFF)
 BEG(GRFF)
```

```
 END(GRFF)
$(GRFP)
 BEG(GRFP)
  .
  .
  .
 END(GRFP)
$$$
```

Здесь записаны три раздела знаний - с именами TBL1, GRFF и GRFP. Содержимое разделов обозначено в виде многоточий. В частности, содержимое раздела TBL1 - это главное меню, GRFF - это набор фрагментов, представляющих граф, а GRFP - набор продукций, обеспечивающих обработку.

Последние два раздела являются пользовательскими - у них имеются дополнительные метки BEG(...) и END(...). Считывание каждого такого раздела с диска в ОП осуществляется следующим образом. По имени раздела в файле находится метка $(<раздел>) и считывается все, что содержится до следующего символа $, кроме них самих, т.е. в ОП не будет метки раздела и символа $.

Разделы знаний могут находиться на диске в одном или множестве файлов. В приведенном примере предполагалось, что три упомянутых раздела находятся в одном файле.

При оформлении разделов, содержащих продукции, вводятся дополнительные средства блокировки. Как уже говорилось, в системе ДЕКЛАР возможно применение одних продукций к другим. Но этого, как правило, не требуется. Чтобы исключить возможность применения продукции к какому-либо разделу (где могут находиться другие продукции) вводятся средства блокировки - спецфрагменты вида @BL(<имя фрагмента>,<имя фрагмента>). Зона, находящаяся между фрагментом с указанным именем, закрыта для применений продукций.

Пример 2.19

Пусть имеется раздел FF, который состоит только из одних продукций. Тогда в он ОП оформляется следующим образом:

```
$(FF)
BEG(FF)
BEG(/FFB)
  .
  .
  .
END(/FFE) @BL(FFB,FFE)
END(FF)
$$$
```

Спецфрагмент @BL(FFB,FFE) указывает на недопустимость применения каких либо продукций к сети (продукциям), находящейся между фрагментами с именами FFB и FFE, т.е. между BEG(/FFB) и END(/FFE). Такая сеть будет закрыта от применений - в процессе применения продукции как бы не будут замечать эту сеть. В то же время сами продукции, находящиеся в разделе FF, будут применяться обычным образом (но не к самим себе).

Пример 2.20

Пусть имеется раздел "НН", который содержит множество фрагментов, представляющих факты или данные (к ним должны применяться продукции), а также содержит набор продукций (к ним не должны применяться продукции). Тогда раздел оформляется следующим образом:

```
$(НН)
BEG(НН)
  ...
<Фрагменты, представляющие данные>
  ...
BEG(/ННВ)
  ...
<Продукции>
  ...
END(/ННЕ) @BL(ННВ,ННЕ)
END(НН)
$$$
```

В данном случае спецфрагмент @BL(...) блокирует только часть раздела НН.

Каждая введенная в систему сеть (фрагменты, продукции) может быть оформлена как раздел. Однако, желательно, чтобы таким образом оформлялись лишь сети, которые содержательно и функционально имеют достаточно законченный характер. Например, как раздел могут быть оформлены продукции, решающие какую-либо задачу, а также множество фрагментов, представляющих достаточно законченную модель.

3. ПРОДУКЦИОННЫЙ ЯЗЫК ПРОГРАММИРОВАНИЯ - ДЕКЛ
(Кузнецов И.П., Шарнин М.М.)

ДЕКЛ - это новый язык программирования. Он служит для преобразования семантических сетей.

Язык ДЕКЛ предназначен для ввода в систему правил, или продукций, которые записываются в форме "ЕСЛИ ... ТО ...". С помощью таких продукций задается решение пользовательских задач, поддерживается система команд, внешний язык и т.д. Семантика языка реализуется программным ядром, работающим, как интерпретатор.

Напомним, что продукции, записанные в форме "ЕСЛИ ... ТО ...", при считывании с диска в ОП автоматически транслируются на язык РСС, где уже работает программное ядро, которое осуществляет применение продукций.

Язык ДЕКЛ по некоторым своим компонентам близок к типовому языку логического программирования - ПРОЛОГ Однако формы его несколько другие, богаче по своим возможностям. В ДЕКЛ условная часть записывается первой, т.е. слева, а следствие - справа (в языке ПРОЛОГ - наоборот). Соответственно, выбрана другая процедура применения правил. Такое применение управляется индикаторами и сводится к проверке условия с порождением следствий. С помощью продукций реализуется как прямой, так и обратный вывод. При этом удалось устранить многие ограничения, характерные для языка ПРОЛОГ.

Во-первых, формы языка ДЕКЛ не ограничиваются правилами, у которых в части, соответствующей следствию, - только один предикат. В продукциях таких предикатов может быть множество.

Во-вторых, правила или продукции (без дополнительных списковых операций) могут применяться к правилам, что весьма необходимо, например, когда нужно анализировать правила, менять их. Более того, одни правила могут строить другие и тут же включать их в работу, что необходимо при металогической обработке.

В-третьих, процедура вычисления не носит "жесткого" характера, как в языке ПРОЛОГ. Правила при применении осматривают все знания, находящиеся в ОП, а не только предикаты запроса. Можно легко управлять такой процедурой, заставляя правила осматривать только нужную часть знаний.

За счет продукций языка ДЕКЛ удается поддерживать достаточно удобные для пользователей языки и для этого не требуется программирования работы над списками. Поддержка языков сводится к обработке структур РСС, что осуществляется на металогическом уровне только с помощью продукций. Специальных средств работы со списками не требуется. Отсюда - проще программное ядро.

В связи со сказанным, язык ДЕКЛ представляется более удобным для представления знаний, и соответственно, для разработки экспертных и других систем, пустых оболочек для них. Сами знания ("программы" на продукциях) получаются более простыми и естественными по своей структуре. Хотя формы языка ДЕКЛ пока еще далеки от конструкций естественного языка. Такие формы скорее приближены к логическим.

3.1. Формы записи продукций

В языке ДЕКЛ продукции могут записываться в различных формах. Одна из них:

<имя продукции>: ЕСЛИ <левая часть> ТО <правая>;

В качестве имени продукции используется какая-либо константа, не входящая ранее в имеющиеся знания на РСС. Слова "ЕСЛИ" и "ТО" ограничивают левую и правую части, а точка с запятой указывает на конец продукции.

Допускаются модификации данной формы. Имя продукции можно не писать (тогда в качестве имени будет взят очередной внутрисистемный код), а вместо слов "ЕСЛИ...ТО" могут быть использованы "IF...THEN".

Левая часть продукции состоит из индикатора, за которым следует набор фрагментов. Индикаторы служат для управления работой продукций, определяют порядок их применения. Фрагменты записываются на языке РСС и разделяются пробелами. Перед фрагментами могут стоять специальные префиксы, которые будем называть операторами. Они определяют роль фрагментов в процессе применен ия продукции, то есть соответствующие действия. В случае, когда операторы отсутствуют, левая часть продукции есть индикатор, за которым следует сеть на РСС.

Другая форма записи продукции:

<индикатор>: ЕСЛИ <левая часть> ТО <правая часть>;

Здесь индикатор вынесен вперед. В левой части его не будет. При такой форме продукции записываются без имен.

В левой части продукции допускаются следующие операторы:

N - нет;

D - удалить;

B - выполнить;

T1 - активизировать.

Оператор ставится перед фрагментом и отделяется от него двоеточием. Например, если R(...) - некоторый фрагмент, записанный на РСС, то N:R(...), D:R(...) и B:R(...) есть фрагменты с операторами. Операторы записываются латинскими буквами.

Правая часть продукции составляется из набора фрагментов, перед которыми могут стоять лишь операторы:

B: - выполнить;

T: (T1: или T!:) - активизировать.

Фрагмент, который стоит после оператора B:, будет называться встроенным (по аналогии с понятием "встроенный предикат" языка ПРОЛОГ). Такой фрагмент определяет обращение к соответствующей программе или процедуре.

Будем называть T:, T1: и T! - операторами активизации, а фрагмент, который стоит за оператором T:(или T1:, или T!), будет называться - индикатором. Если у упомянутого фрагмента имеются какие-либо о аргументные места, то будем говорить, что индикатор имеет параметры. В противном случае индикатор будет записываться как отдельная константа. Примером оператора активизации является T:UR1, где UR1 есть некоторая константа (индикатор). С помощью операторов активизации осуществляется управление применением продукций - активизация индикаторов и, соответственно, самих продукций. Каждая такая активизация приводит к попыткам применения продукции. При этом:

T!: - это оператор последовательного применения (каждая продукция с указанным индикатором активизируется многократно - во всех вариантах - вначале первая, затем вторая, и так далее до последней продукции);

T1: - оператор однократного применения (также осуществляется последовательная активизация, но до первой применимой продукции);

T: - оператор циклического применения (продукции активизируются, пока среди них есть применимые, т.е. вначале первая, затем вторая. Если при таком проходе были применимые продукции, то снова - первая и т.д.).

Пример 3.1

```
SS1: ЕСЛИ UR1 R1(X1) N: R2(X1) D: R3(X1, A1) ТО
        R4(X1)  B: A(X1)  T: UR2(X1);
```

Продукция имеет имя SS1. В левой части она содержит индикатор U R1, фрагмент R1(X1), а также фрагменты R2(X1) и R3(X1, A1) с операторами N: и D:. В правой части имеется фрагмент R4(X1), фрагмент A(X1) с оператором B: и оператор активизации T:UR2(X1) с индикатором UR2(X1).

Продукция означает, что если X1 имеет свойство R1, не имеет свойства R2 и связано отношением R3 с A1, то нужно удалить (D:) из ОП соответствующий экземпляр после днего отношения, добавить к X1 свойтство R4, выполнить (B:) над X1 процедуру "A" и активизировать (T:) индикатор UR2 с параметром X1. Такие действия выполняются в процессе применения продукции и связаны с

поиском в ОП соответствующих фрагментов и т. д. Причем действия становятся возможными, когда активизирован индикатор UR1.

Следует отметить, что оператор B: может стоять только перед так называемыми встроенными фрагментами, связанными со своими процедурами .

Пример 3.2

```
SS2: ЕСЛИ UR2 B: I(X1) N: _( _/X1) ТО  MT(X1) T: UR3;
```

Эта продукция вызывает специальные действия, которые будут описаны далее. Сейчас нас интересует форма записи продукций.

Язык ДЕКЛ допускает достаточно произвольное количество фрагментов как в левой, так и в правой части - столько, сколько позволяет объем ОП, отведенный под знания.

В языке ДЕКЛ множество фрагментов левой части может быть сгруппировано в сеть, перед которой может стоять оператор N:. Для этого используются квадратные скобки. Подобная группировка позволяет задавать сложные виды отношений.

Пример 3.3

```
UL2: ЕСЛИ US1 TRUE(X1) DAND(X1, X2) N: [DAND(X3, X2) N: TRUE(X3)]
     ТО  TRUE(X2);
```

Здесь имеет место двойное отрицание. Первый оператор N: распространяется на два фрагмента, стоящие в квадратных скобках, а второй - на фрагмент TRUE(X3).

Продукция дословно означает: "если у X1 имеется свойство TRUE, X1 связан отношением DAND с X2 и если нет такого, что X2 связан отношением DAND с какой либо X3, не обладающей свойством TRUE, то это значит, что X2 обладает свойством TRUE" (и это свойство к X2 добавляется). Как будет показано ниже, данной продукцией реализуется переход по дугам "И" в "И-ИЛИ" графе.

В языке ДЕКЛ допускаются так называемые "вложенные" структуры, т. е. когда одни квадратные скобки находятся внутри других и т. д.

В левых и правых частях одних продукций могут находиться другие продукции, которые записываются на языке РСС. Это необходимо работы на металогическом уровне, где часто требуется производить анализ содержимого знаний, включающих в себя продукции, осуществлять формирование новых продукций и др.

Отметим, что в правой части продукции может стоять множество операторов активизации. Более того, правая часть может состоять только из таких операторов. Тогда будет иметь место так называемая управляющая продукция. Она сама ничего не изменяет в знаниях, но последовательно активизирует другие продукции, вызывая попытки их применения.

Итак, после операторов активизации могут стоять как константы, так и отдельные фрагменты. В последнем случае активизация продукций будет сопровождаться передачей параметров.

Например, оператор T:VS*(X) будет активизировать продукции с индикатором VS*(...), осуществляя передачу в них параметра X, т.е. передачу результата означивания X, см. ниже.

В правой части продукции после операторов активизации могут стоять переменные, встречающиеся в левой части, например, T:X. В случае применимости продукции активизированным индикатором будет константа, которой была означена переменная X.

Наконец, отметим, что операторы активизации вида T1: могут стоять и в левой части продукции. Тогда по указанному индикатору идет обращение к другим продукциям (у них в конце ставится - RET), которые определяют, продолжить или нет применения первой продукции, а если продолжить, то в каком варианте, т.е. осуществляется передача ей параметров. Формально, левая и правая части продукции описываются следующим образом:

```
<индикатор>     ->  <константа>\<фрагмент>.

<л-фрагмент>    ->  B:<фрагмент>\D:<фрагмент>\T1:<индикатор>
                    N:<фрагмент>\<фрагмент>\N:[<фрагмент> ...
                    .. <фрагмент>]
<левая часть>   ->  <индикатор>\ <индикатор> <л-фрагмент>\
                    <индикатор> <л-фрагмент> ... <л-фрагмент>

<п-фрагмент>    ->  T:<индикатор>\T:<переменная>\T1:<индика-
                    тор>\ T1:<переменная>\ T!:<индикатор>\
                    T!:<переменная>\B:<фрагмент>\<фрагмент>.

<правая часть>->  <п-фрагмент>\ <п-фрагмент>...<п-фраг-
                  мент>.
```

Здесь, как и ранее, символ "\" означает разделительное "ИЛИ".

3.2. Принципы применения продукций

В данном разделе будут описываться принципы применения продукций, у которых отсутствуют операторы. Пусть левая часть такой продукции - есть индикатор, за которым следует сеть, а правая часть - есть просто сеть. Обе сети записаны на РСС.

Сначала приведем несколько простейших примеров применения продукций. Затем такое применение будет описано более точно.

Пример 3.4

```
SS1:ЕСЛИ UR1 ДЕД(X1,X2) ТО ОТЕЦ(X1,X3) ОТЕЦ(X3,X2) ;
```

С помощью это продукции представлено: если X1 является дедом X2, то это значит, что X1 является отцом для X3, а X3 - отцом для X2.

Продукция начинает применяться, только когда активизирован индикатор UR1 (это осуществляется другой продукцией, содержащей в правой част и оператор T:UR1 (или T1:UR1, или T!:UR1). Будем называть подобную продукцию активизированной. При этом в сети (РСС), находящейся в ОП, ищется фрагмент вида ДЕД(AI,AJ), где, в

частности, AI и AJ могут быть некоторыми константами. Если такой фрагмент имеется, то ее переменные X1 и X2 означиваются этими константами - AI и AJ. Продукция об'является применимой. Результат означивания (AI и AJ) переносятся в правую часть и замещает переменные (X1 и X2). На место X3 ставится внутрисистемный код N+, где N - некоторое Число, а полученные таким образом фрагменты ОТЕЦ(AI,N+) ОТЕЦ(N-,AJ) добавляются в ОП.

Если в ОП не найден фрагмент вида ДЕД(AI,AJ), то продукция считается неприменимой и никакого добавления не производится.

Если продукция рассматриваемого примера была активизирована оператором T:UR1 Вли T!:UR1, то она будет применимой столько раз, сколько различных фрагментов вида ДЕД(AI,AJ) имеется в ОП. И каждый раз к ОП будет добавляться правая часть, как это было рассмотрено выше.

Если же продукция была активизирована оператором T1:UR1, то она применится лишь один раз вне зависимости от наличия тех или иных фрагментов в ОП.

Продукция будет применимой и в том случае, когда в ОП есть фрагменты вида ДЕД(XI,AJ) или ДЕД(AI,XJ) или ДЕД(XI,XJ), то есть на местах констант могут стоять переменные. Переменные могут означиваться переменными. Действия применения остану тся теми же. Например, в случае применимости продукции к сети с фрагментом ДЕД(XI,AJ) к ней будет добавлено ОТЕЦ(XI,N+) ОТЕЦ(N-,AJ). При этом X1 означивается XJ.

Пример 3.5

ЕСЛИ UR1 _(_/X1) ТО МТ(X1) ;

У данной продукции нет имени (что допустимо). Эта продукция носит чисто служебный характер. Ее активизация посредством UR1 приведет к поиску в ОП каких-либо одноместных фрагментов. При этом не важно, какие у них имена отношений и какие аргументы. Символ "_" в левой части продукции означает, что на этих местах в упомянутых одноместных фрагментах может стоять что угодно: константы, переменные или такие же символы. Они могут быть различными.

Если хотя бы один подобный фрагмент имеется в ОП и его имя не является аргументом МТ(...), то продукция сделается применимой. При этом X1 будет озна чена именем данного фрагмента. За счет правой части на основе этого имени будет сформирован новый фрагмент. Например, если в ОП имеется фрагмент RR1(A1/DD1), то X1 будет означено DD1. В результате будет сформировано МТ(DD1).

Итак, при применении продукции символы "_", находящиеся в левой части могут означиваться чем угодно, но результат означивания не сохраняется и никак не учитывается. Опишем вначале с помощью абстрактных понятий действия применения п родукций. Для этого введем понятие соответствия сетей.

Будем говорить, что одна сеть соответствует другой, если первая сеть может быть получена из второй путем следующих замен:

символа "_" на любую константу (или переменную), при этом символы "_", стоящие на различных местах, могут заменяться на различные константы (или переменные).

Описанные замены эквивалентны означиванию, а соответствие есть частный случай унификации.

При установлении соответствия учитывается тот факт, что фрагмент, записанный в ОП в виде R(...) преобразуется системой в R(.../N+), где N+ - внутрисистемный код. Аналогичный фрагмент, находящийся в левой части продукции, преобразуется в R(.../X), где X - "служебная" переменная, которая нигде не фигурирует, но которая в процессе применения продукции означивается именами других фрагментов.

Рассмотрим примеры соответствия отдельных фрагментов.

Таблица 1

N	фрагмент в левой части продукции.	соответствующий фрагмент в ОП	результат означивания
1.	R1(A1)	R1(A1)	
2.	R(X1,X2)	R(A1,A2)	X1 -> A1 X2 -> A2
3.	R(_,X)	R(A1,A2)	X -> A2
4.	R(X,_)	R(X1,A2)	X -> X1
5.	_(A1,_)	R(A1,A2)	
6.	R(A1,X1,/X2)	R(A1,A2)	X1 -> A2 X2 -> N+
7.	R(X1,X2)	R(A1,A2/D)	X1 -> A1 X2 -> A2
8.	R(X1,A2/X2)	R(A1,A2/KK)	X1 -> A1 X2 -> KK

Отметим, что поскольку фрагмент R(A1,A2) из шестой строки таблицы своего имени не имеет, то система производит означивание переменной X2 очередным внутрисистемным кодом, обозначенным через N+.

Перейдем к описанию на абстрактном уровне действий применения отдельных продукций.

Случай 1. В левой части продукции имеется только один фрагмент. Применение такой продукции сводится к поиску соответствующего фрагмента в ОП, согласно таблице 1. Если такой фрагмент найден, то продукция объявляется применимой, иначе - считается неприменимой). Результаты означивания запоминаются в специальном "стеке означиваний".

Далее, на основе правой части продукции система формирует новые фрагменты. Для этого переменные из фрагментов правой части заменяются на результат их означивания (т.е. на соответствующие константы, взятые из стека означиваний). Если какой-либо

переменной нет в стеке, то это значит, что она входитодит только в правую часть продукции. В этом случае переменная заменяется на внутрисистемный код. Указанным способом формируются новые фрагменты, которые добавляются в ОП. При этом сама продукция не меняется.

Случай 2. В левой части продукции имеется множество фрагментов. Обозначим такое множество через W. Остановимся на действиях применения такой продукции.

Пусть в ОП находится сеть, содержащая другое множество фрагментов - V. Продукция будет применимой, если в V имеется подмножество фрагментов - сеть, соответствующая W. Словом, в V должны быть фрагменты, которые могут быть получены из W путем описанных ранее замен: переменных на константы или переменные, а символа "_" - на что угодно.

Замещения переменных определяют означивания, которые заносятся в стек. Далее на основе правой части продукции формируются новые фрагменты, то есть некоторая сеть.

Таблица 2.

N	фрагменты в левой части продукции	соответствующие фрагменты в ОП	результат означивания:
1.	R1(A1,X) R2(X)	R1(A1,A3) R2(A3)	X -> A3
2.	R1(_, _) R2(_)	R1(A1,A3) R2(A5)	
3.	R1(X1,X2) _(X1)	R1(A1,A3) R2(A1/DD)	X1 -> A1 X2 -> A3
4.	R1(A1,X/X1)	R1(A1,A3)	X -> A3 X1 -> K+

Пример 3.6

Пусть имеется продукция:

SS1: ЕСЛИ VV1 ДЕД(X1,X2) МУЖ(X1) ТО ОТЕЦ(X1,X3) ОТЕЦ(X3,X2);

По сравнению с продукцией из примера 3.4 она содержит дополнительный фрагмент МУЖ(X1), представляющий, что X1 - это мужчина.

Будучи активизирована, такая продукция станет применимой, если в сети, находящейся в ОП, имеются фрагменты вида ДЕД(AI,AJ) МУЖ(AI), где AI,AJ - константы (это могут быть также переменные). Тогда X1 будет означено AI, а X2 - AJ. Применение продукции вызовет такие же добавления, как и в случае продукции примера 3.4.

3.3. Процесс применения продукций, последовательность действий

Ранее действия применения отдельных продукций были описаны путем использования абстрактных замен. На самом деле программное

ядро, осуществляющее такое применение, реализует последовательный процесс, в котором порядок расположения фрагментов в продукции оказывается существенным.

Действия применения отдельной продукции сводятся к последовательному выполнению элементарных акций, связанных с поиском соответствующих фрагментов и означиваниями переменных. Вначале берется первый фрагмент, находящийся в левой части продукции. Ищется соответствующий ему фрагмент в ОП. Если он есть, то акция считается успешной. Тогда осуществляется означивание переменных (в текущем варианте), которое запоминается в стеке. Далее, берется второй фрагмент левой части. Ищется соответствующий фрагмент в ОП, но с учетом уже выполненных означиваний. Если такого фрагмента нет, то акция считается безуспешной. Осуществляется возврат к первому фрагменту левой части продукции с поиском другого соответствующего ему фрагмента и выбором другого варианта означивания, и т. д. Уже последний вариант будет запомнен в стеке "означивания", а прежний - стирается.

Процесс продолжается до тех пор, пока в ОП не будет найдена подсеть (один или несколько фрагментов), соответствующая фрагментам левой части продукции. Если такая подсеть найдена, то продукция объявляется применимой. Запомненное в стеке означивание переносится в правую часть. Фрагменты, сформированные за счет ее правой части, добавляются к ОП. Если подсеть не найдется, то продукция считается неприменимой. Такая продукция ничего не изменяет в ОП.

Пример 3.7

Рассмотрим применение продукции из примера 3.6. Пусть в ОП имеются фрагменты:

ДЕД(A1, A2) МУЖ(A3) ДЕД(A3, A4),

представляющие, что A1 является дедом A2, A3 - дедом A4 и A3 - мужчина.

Процесс применения продукции начинается с того, что берется ее первый фрагмент - ДЕД(X1, X2). Осуществляется поиск соответствующего фрагмента в ОП. Если такой фрагмент имеется, то означиваются переменные X1 и X2. Означивания запоминаются в стеке. В данном примере X1 будет означено A1, а X2 - A2. Далее в продукции берется следующий фрагмент - МУЖ(X1) как МУЖ(A1), то есть учитываются уже выполненные означивания. Данный фрагмент ищется в ОП. Его нет. Тогда означивания (в стеке) стираются. Снова осуществляется переход к первому фрагменту продукции ДЕД(X1, X2) с поиском соответствующего фрагмента в ОП, но уже другого. Им будет ДЕД(A3, A4). Переменная X1 означивается A3, X2 - A4. В ОП ищется фрагмент МУЖ(A3). Он есть. Тогда продукция объявляется применимой. На основе правой части будут сформированы фрагменты ОТЕЦ(A3, 1+) ОТЕЦ(1-, A4) и добавлены в ОП.

Итак, поиск сети, соответствующий левой части продукции, сводится к поиску в ОП соответствующих фрагментов. Такой поиск осуществляется последовательно - вначале для первого фрагмента

левой части, затем второго, и т.д. После этого продукция объявляется применимой или неприменимой. В связи с описанным процессом очень важно, какой фрагмент в левой части продукции стоит первым. Если у этого фрагмента есть много соответствующих фрагментов в ОП, то будет делаться много попыток применения продукции с выполнением возвратов (когда для других фрагментов из левой части не окажется соответствующих фрагментов). Чтобы таких попыток было меньше, лучше первым ставить фрагмент с "уникальными" константами, редко встречающимися в ОП. Вообще, расположение и других фрагментов в левой части должно определяться степенью уникальности входящих в них констант, например, имен отношений. Таким способом можно значительно уменьшить обьем работы, выполняемой системой в процессе применения продукции.

Следует отметить, что в общем случае допускается означивание переменных левой части продукции продукции символом "_" (нижняя черточка). Но такое означивание не считается окончательным. За счет других фрагментов будут делаться попытки повторного означивания. Например, пусть в ОП имеется:

ДЕД(_, А2) МУЖ(А3)

Первый фрагмент означает: "неважно, кто является дедом для А2". Тогда при применении рассмотренной ранее продукции Х1 будет означено "_". Но за счет второго фрагмента МУЖ(Х1) это означивание будет замещено на А3. Процесс ее применения будет продолжен.

3.4. Операторы в левой и правой частях продукции

Напомним, что перед фрагментами в левой части продукции могут стоять следующие операторы: N: - нет, D: - удаление и B: - выполнить и T1: - активизировать. В процессе применения продукции последовательно берутся фрагменты ее левой части с выполнением действий поиска. Если перед фрагментом стоит оператор, то это означает необходимость выполнения других действий:

N: <фрагмент> - проверяется отсутствие в ОП фрагмента, соответствующего указанному фрагменту, то есть стоящему после N:. только при его отсутствии продукция может быть применимой.

D: <фрагмент> - фрагмент, соответствующий указанному фрагменту, удаляется из ОП. Это делается сразу же после того, как продукция оказалась применимой.

B: <фрагмент> - выполняется процедура или функция, связанная с указанным фрагментОМ. Подобные фрагменты будем называть встроенными. При этом в ОП не осуществляется поиска фрагментов, соответствующих указанному.

T1: <индикатор> - активизируются продукции с указанным индикатором (у них в конце должно стоять -

RET). Если среди них есть применимые, то продолжается применение текущей продукции, вызвавшей активизацию. В эту продукцию могут быть переданы результаты означиваний.

Итак, описанные действия выполняются, когда при применении продукции процесс дошел до одного из перечисленных операторов. При этом учитываются произведенные ранее означивания.

Акция, связанная с выполнением оператора N:, может быть успешной, когда фрагмент не найден. И безуспешной, когда фрагмент найден. В случае успешного выполнения процесс применения продукции продолжается, иначе - заканчивается. Продукция считается неприменимой.

Оператор D: срабатывает сразу же после того, как продукция оказалась применимой. Тогда выполняется акция удаления (D), которая всегда будет успешной, то есть обязательно будут вызваны необходимые изменения в ОП.

Акции, связанные с выполнением оператора B: из левой части продукции, могут быть успешными и безуспешными. Это определяется вызываемой процедурой или функцией, с которой связан стоящий за B: встроенный фрагмент. В случае успешности процесс применения продукции продолжается, то есть берется следующий фрагмент и т. д. В случае безуспешности - заканчивается, и продукция считается неприменимой.

Акция, связанная с выполнением оператора T1:<индикатор> будет успешной, если хотя бы одна из активизированных продукций оказалась применимой. Тогда применение текущей продукции продолжается, иначе - заканчивается.

При этом от применимых продукций, активизированных с помощью T1:<индикатор>, (для ранее не означенных переменных, входящих в индикатор), осуществляется передача результатов означивания в первую продукцию. На необходимость такой передачи указывает спецконстанта - RET (см. п. 3.5.).

Пример 3.8

Рассмотрим применение продукции, приведенной в примере 3.1. Эта продукция имеет вид:

```
SS1: ЕСЛИ UR1 R1(X1) N:R2(X1) D:R3(X1,A1)
     ТО   R4(X1) B:A(X1) T:UR2(X1);
```

При применении данной продукции первым берется фрагмент R1(X1), за счет которого осуществляется означивание переменной X1. Пусть X1 означена константой AI. Тогда оператором N:R2(X1) будет осуществляться проверка отсутствия фрагмента R2(AI) в ОП. Если проверка удовлетворяется, то будет взят следующий фрагмент - D:R3(X1,A1). Будет осуществляться поиск в ОП фрагмента вида R3(AI,A1). Если такой фрагмент имеется, то продукция будет применимой. Тогда вначале фрагмент R3(AI,A1) будет изъят из ОП, а затем уже к ОП будет добавлено R4(AI). Далее будет выполнена процедура с именем "A" (выдача на экран) и активизирован индикатор UR2(AI). Это вызовет активизацию других продукций с передачей им параметра - AI.

Пример 3.9

```
SS2: ЕСЛИ  VV1  D: ДЕД(X1,X2)  N: ЖЕН(X1)
     ТО            ОТЕЦ(X1,X3) ОТЕЦ(X3,X2);
```

Здесь представлено, что если X1 является дедом X2 и X1 не женщина (ЖЕН), то это значит, что X1 является отцом для X3, а X3 отцом для X2. При этом предполагается, что отношение "БЫТЬ ДЕДОМ" должно быть замещено на "БЫТЬ ОТЦОМ ОТЦА". Фрагмент, соответствующий ДЕД(X1,X2), должен быть удален из ОП.

Продукция будет применимой, когда в ОП имеются фрагменты вида ДЕД(AI,AJ). В результате применения продукции фрагмент ДЕД(AI,AJ) будет удален из ОП и будет добавлено ОТЕЦ(AI,N+) ОТЕЦ(N-,AJ). Такие действия будут выполняться последовательно. Вначале будет осуществлен поиск фрагмента, соответствующего ДЕД(X1,X2), с означиванием X1 и X2. Затем будет осуществлена проверка отсутствия в ОП фрагмента ЖЕН(...). Если проверка не удоволетворяется, то ищется новый фрагмент, соответствующий ДЕД(X1,X2), и т.д.

Рассмотрим применение продукции примера 3.2. Вначале выполняется процедура "I", осуществляющая ввод символа с клавиатуры и означивание этим символом переменной X1. Это действие всегда будет успешным. Далее проверяется отсутствие в ОП одноместных фрагментов, именем которых является считанный символ (пусть это будет символ AJ). Если проверка удовлетворяется, то продукция объявляется применимой. В оп добавляется фрагмент МТ(AJ) и активизируются продукции с индикатором UR3.

Следует отметить, что с помощью оператора "выполнить" (B:), стоящего в левой части продукции, допускается обращение только к ограниченному числу процедур или функций. Это могут быть лишь функции, осуществляющие означивание переменных и разного рода проверки - равенства, неравенства, больше, меньше. Здесь не может быть функций, осуществляющих вывод на экран, блокировки, стирания и т.д. Иначе может оказаться, что перечисленные действия будут выполнены, хотя продукция оказалась неприменимой. Такая ситуация нежелательна.

Оператор N: может стоять не только перед отдельными фрагментами, но и перед сетями, заключенными в квадратные скобки. Тогда осуществляется проверка одновременного отсутствия в ОП множества соответствующих фрагментов. Отметим, что такая проверка может быть не эквивалентной последовательной проверке отсутствия фрагментов. С помощью конструкций вида N:[...] реализуются сложные виды отрицания.

Пример 3.11

Рассмотрим применение продукции вида (из примера 3.3).

```
UL2: ЕСЛИ US1 TRUE(X1) DAND(X1,X2) N:[DAND(X3,X2) N: TRUE(X3)]
     ТО   TRUE(X2);
```

.....

С помощью данной продукции обеспечивается переход по дугам "И" в графе "И-ИЛИ". При этом ветви типа "И" представляются с помощью фрагментов DAND(...,...).

Пусть в ОП имеются следующие фрагменты:

DAND(A1,A3) DAND(A2,A3) TRUE(A1) TRUE(A2)

Данная сеть означает, во-первых, что если истинны A1 и A2, то истинно A3 (см. Первые два фрагмента), и, во-вторых, что A1 - истинно и что A2 - истинно (последние два фрагмента).

Остановимся на процессе применения продукции. Вначале в ОП будет осуществляться поиск фрагмента, соответствующего TRUE(X1). Переменная X1 будет означена A1. Далее будет осуществляться поиск фрагмента, соответствующего DA- ND(A1,X2). В результате X2 будет означено A3. После этого выполняется проверка отсутствия (N:) в ОП сети, соответствующей фрагментам, которые заключены в квадратные скобки.

Для разных вариантов означивания X3 (пусть это AJ) в ОП ищется фрагмент вида DAND(AJ,A3), такой, что отсутствует TRUE(AJ). Если такой фрагмент удается найти, то выполнение оператора N: объявляется безуспешным, а продукция - неприменимой. В приведенном примере подобного фрагмента нет в ОП. Выполнение N: будет объявлено успешным, а применение продукции продолжено. В результате будет сформировано - TRUE(A3). Интерпретация подобных действий изложена в п.7.2.

Перед фрагментами в правой части продукции могут стоять лишь операторы "выполнить" и "активизировать". Эти операторы начинают свои действия только в том случае, если продукция оказалась применимой. Соответственно, ее переменные, входящие в левую часть, уже означены. Тогда последовательно берутся фрагменты правой части и осуществляются необходимые действия.

Напомним, что если в правой части перед фрагментом нет операторов, то такие фрагменты просто добавляются к ОП с учетом имевших место означиваний.

Если встретился оператор B:R(...), где R - некая константа, то выполняется процедура или функция, связанная с R(...). При выполнении учитывается результат означивания, имевшего место при применении продукции. Переменные в R(...) заменяются на соответствующие константы.

Пример 3.12

ЕСЛИ UT1 ДЕД(X1,_) N:MN1(X1) ТО B:A(X1) ;

В данной продукции B:A(...) - есть обращение к встроенному фрагменту выдачи на экран. Такая продукция будет осуществлять поиск в ОП фрагментов вида ДЕД(AI,AJ) с выдачей на экран констант AI, не входящих во фрагменты MN1(AI). Если в ОП не было таких фрагментов, на экран будут выданы все константы и переменные, обозначающие дедов, то есть стоящие на первом месте во фрагментах ДЕД(...).

Выполнение оператора B:, стоящего в правой части продукции, может привести к следую щим результатам:

- быть успешным (когда получен приемлемый результат);

- быть ошибочным;

- вызвать прерывание работы с переходом в режим ожидания;

- привести к прекращению работы ядра ДЕКЛАР и выходу в монитор ОС.

В случае успешного и ошибочного выполнения процесс применения продукции продолжается, то есть берется следующий фрагмент или оператор из ее правой части. Соответствующий фрагмент добавляется к ОП, а оператор - выполняется, и т.д.

В случае ошибочного выполнения дополнительно выдается сигнал об ошибке. Например, такой случай имеет место, когда из буфера клавиатуры считывается выражение, выходящее за рамки допустимых конструкций входного языка (это делается встроенным фрагментом "IN").

Прерывание вызывает приостановку применения продукции, например, когда считывается очередной символ из буфера клавиатуры, а его нет. Тогда система переходит на режим ожидания. В это время никакие продукции не применяются. Как только такой символ появится (будет введен пользователем), работа возобновляется. Выполнение оператора считается успешным. Применение продукции продолжается.

В случае прекращения работы осуществляется общий сброс. Из ОП удаляется программное ядро системы ДЕКЛАР и все имеющиеся разделы знаний. На экране появляется символприглашение ОС. Начинать работу нужно сначала. Такой случай может возникнуть, например, когда с диска требуется считать файл, которого там нет. Последний случай вызван особенностью обращения к файловой системе языка ПАСКАЛЬ, используемого при реализации программного ядра ДЕКЛАР.

3.5. Управление применением продукций

В системе ДЕКЛАР имеются средства, позволяющие вызывать только "нужные" продукции, а также управлять процессом их активизации. Такими средствами являются следующие операторы активизации:

T1:<индикатор> - оператор акивизации продукций до первого применения хотя бы одной из них. Последовательно делаются попытки применения продукций. Вначале берется первая продукция с указанным индикатором и делаются попытки ее применения, затем - вторая и т.д. После того, как одна из продукций оказалась применимой, процесс заканчивается, то есть больше не делается попыток применения других продукций с таким же индикатором.

T!:<индикатор> - оператор последовательной активизации продукций за "один проход". Последовательно делаются попытки применения всех продукций с указанным индикатором (во всех

вариантах). Здесь может быть много применимых продукций. Процесс заканчивается, когда доходит до последней продукции - после всех попыток ее применения.

T:<индикатор> - оператор циклической активизации продукций. Последовательно делаются попытки применения всех продукций с указанным индикатором во всех вариантах. Но в отличие от предыдущего, процесс не заканчивается, когда доходит до последней продукции. Если какая-либо продукция оказалась применимой, то снова берется первая продукция и т.д. Процесс заканчивается, когда нет применимых продукций с указанным индикатором.

С помощью этих операторов обеспечивается высокая эффективность процесса вычисления, минимизация числа попыток применения продукций.

Операторы активизации T!: и T: могут стоять только в правой части какой-либо продукции, а T1: - в левой и правой.

Выполнение каждого такого оператора вызывает прерывание процесса применения активизировавшей ее продукции. Такой процесс приостанавливается. При этом активизация предыдущего индикатора (вызвавшего применение данной продукции) временно снимается и активизируется другой индикатор - стоящий после T:(или T1:, или T!:). Это вызывает активизацию других продукций. Делаются попытки их применения. Если среди них оказались применимые продукции, то акция, связанная с выполнением оператора "активизации", объявляется успешной, а если ни одна продукция не применилась, то - безуспешной. Прерывание первой продукции снимается. Осуществляется переактивизация индикаторов - снова активизируется индикатор первой продукции и процесс ее применения продолжается (или заканчивается).

Если оператор активизации стоит в правой части продукции, то всегда процесс ее применения продолжается - вне зависимости от успешности или безуспешности акции, а если оператор активизации стоит в левой части, то процесс продолжается только в случае успешности акции. При этом в том случае, когда в ОП не было продукций с указанным индикатором, то есть нечего было активизировать, акция считается безуспешной.

Пример 3.13

```
... :ЕСЛИ UR1 ... ТО ... T:MM1... ;
... :ЕСЛИ MM1 ... ТО ... ;
```

В данном примере схематично проиллюстрирован принцип работы оператора T:, входящего в правую часть продукции. Здесь имеется две продукции. Первая применяется при активизации UR1. Если она оказалась применимой, то выполняется T: и активизируется индикатор MM1. Делается попытка применения второй продукции (в общем случае множества продукций с индикатором MM1). После окончания попыток осуществляется переход к первой продукции. Не трудно видеть, что здесь использован принцип возврата (BACK TRACKING), реализуемый с помощью активизации индикатора MM1 при вызове продукций.

Следует отметить, что оператором вида T: следует пользоваться с большой осторожностью. Такой оператор (в отличие

от T1: и T!:) должен активизировать продукции, которые что-то изменят в ОП. За счет подобного изменения продукции в конце концов должны сделаться неприменимыми. Иначе процесс их вызова может быть безостановочным. Например, если вторая из только что рассмотренных продукций всегда применима, то она будет активизироваться оператором T:MM1 безостановочно. Система "зациклится".

Продукции, у которых правая часть состоит только из операторов T:, будем называть управляющими, а если таких операторов нет, то - обрабатывающими. Возможны и промежуточные случаи. С помощью управляющих продукций реализуется принцип "двойной черной доски" (DOUBLE BLACK BOARD). Они вызывают последовательную активизацию индикаторов. Причем в каждый момент времени может быть активизирован лишь один индикатор, и соответственно, делаются попытки применения сугубо ограниченного множества продукций - обрабатывающих или управляющих. Таким способом можно задавать сложные стратегии вычислений. За счет левых частей управляющих продукций можно придать процессу условный характер. При умелой организации процесса общее количество попыток применения продукций (и соответственно, объем работы, связанной с вычислениями) может быть уменьшено во много раз, что особенно важно при работе системы в реальном масштабе времени.

В общем случае активизация индикаторов может сопровождаться передачей параметров. Например, при выполнении оператора T1:AN(X1,X2,X3) активизируются продукции с индикатором AN(...), и в эти продукции передаются параметры - ими является результат означивания X1, X2 и X3. Применение активизированных продукций будет начинаться уже при наличии означенных переменных, см. Пример 3.13.

В системе ДЕКЛАР имеется один индикатор (он обозначается латинским K), от которого как бы начинается вычисление. Он называется "корень". Корень активизируется самой системой, когда нет других активизированных индикаторов. Такой процесс будет безостановочным.

Пример 3.14

```
ЕСЛИ  K  TO  B:F(MENU1,PR1G)  T1:INZ(SD)  T1:INZ(SDD)
   AKT(TBL1) T:WO;

 ЕСЛИ INZ(X) KTL(X,X1) TO B:IN(X1,X);
```

Первая продукция является управляющей. Она инициирует работу системы ДЕКЛАР. Ее инициатором является корень "K". Она будет активизироваться первой и вызывать выполнение встроенного фрагмента "F" - выдачу на экран таблицы PR1G из файла MENU1. после этого применение первой продукции временно прерывается, и активизируется вторая продукция, имеющая индикатор INZ(...).

Процесс активизации будет сопровождаться передачей параметра - переменная X будет означена константой SD. Вторая продукция осуществляет ввод раздела знаний с диска в ОП. По каталогу (KTL) находится имя файла, в котором хранится данный

раздел. В частности, если в разделе KTL есть предикат KTL(SD,PROD), то X1 будет означен константой "PROD". Вторая продукция сделается применимой, что вызовет выполнениие встроенного фрагмента B:IN(PROD,SD) - считывание из файла PROD.Z в ОП раздела SD. После однократного применения второй продукции (как этого требует оператор T1:) управление будет передано первой продукции, в которой берется следующий оператор - T1:INZ(SDD). Аналогично описанному ранее, в ОП вводится раздел SDD.

Далее в ОП добавляется фрагмент AKT(TBL1). Он представляет, что внимание системы акцентировано на первой вершине графа диалога - на TBL1. Далее активизируются продукции с индикатором "WO", которые, в соответствии с графом диалога обеспечивают взаимодействие с пользователем.

После применения первой продукции снова будет активизирован корень - K. Будет делаться попытка повторного применения. Такие попытки будут делаться безостановочно, т.е. пока работает система ДЕКЛАР. Естественно, первые попытки будут приводить к изменению знаний в ОП. Последующие же попытки могут оказаться холостыми, т.е. применимыми будут только управляющие продукции. Обрабатывающие же не будут применяться. Изменений в ОП не будет. На экран также ничего не будет выдаваться. Процесс обработки как бы заканчивается. Такой процесс может быть возобновлен введением в ОП новых продукций или новых фрагментов.

Пример 3.15

```
ЕСЛИ UR00 ТО T:UK1 T:UK2 T:UR1 T:UR2 T:UR3;
```

Данная продукция является управляющей. Ее применение вызывается активизацией индикатора UK00. Это приведет к последовательному включению в работу множеств продукций: вначале с индикатором UK1, з атем - UK2 и т.д. Как бы все продукции разбиваются на уровни. Вначале делаются попытки применения продукций первого уровня (с UK1), затем второго (с UK2) и, наконец, третьего (с UR3). Причем на любом из уровней может быть пустое множество продукций. Все равно процесс последовательного включения будет продолжаться.

Описанное деление продукций на уровни является весьма удобным с точки зрения обработки. Например, с уровнями UK1 и UK2 можно связать работу лингвистического процессора (соответствующих продукций), с уровнями с UR1 и UR2 - продукций, осуществляющих решение пользовательских задач, с UR3 - выдачи результатов. Тогда будет строго регламентирована последовательность действий: поступающая с экрана информация вначале будет обрабатываться лингвистическим процессором, осуществляющим ее преобразование в сеть на языке РСС. Затем (уже с помощью других продукций) будет осуществлен анализ этой сети на предмет, является ли поступившая информация условием задачи. Если да, то реализуются требуемые действия, связанные с решением. По их окончании выдается результат и из ОП удаляются все ненужные сведения. Система готова к восприятию следующей информации, причем на каждом уровне могут быть свои управляющие продукции, например, реализующие по этапам работу

лингвистического процессора, и т. д.

В заключение рассмотрим еще один пример, иллюстрирующий принципы передачи параметров между продукциями или их группами (модулями).

Пример 3.16

```
ЕСЛИ UR1 ... R1(X) ... ТО ... T1:MM1(X);

ЕСЛИ MM1(X1) T1:NN1(X1,X2) ТО ... ;

ЕСЛИ NN1(X1,X2)  R2(X1,X2) ТО RET;
```

Здесь схематично проиллюстрирован случай, когда от первой продукции передается параметр во вторую. Причем, в первой продукции этот параметр вычисляется - в качестве него берется результат означивания переменной X. Пусть это - A1. Тогда при активизации второй продукции осуществляется передача константы A1 - ей означивается переменная X1. Применение второй продукции производится уже с учетом такого означивания. В левой части этой продукции стоит оператор T1:. Продукция будет применимой (и будут выполнены действия, указанные на месте многоточия) только в том случае, если акция активизации окажется успешной, то есть продукция с индикатором NN1(X1,X2) оказалась применимой.

Обращение к третьей продукции осуществляется с передачей значения X1 - константы A1. Переменная же X2 не означена. Такое означивание осуществляется в процессе применения третьей продукции. Она будет искать в ОП фрагменты вида R2(A1,...). Пусть, например, в ОП есть фрагмент R2(A1,A2). Тогда X2 будет означена константой A2, которая будет передана (на что указывает - RET) во вторую продукцию. Она сделается применимой и т. д.

В настоящей версии системы количество передаваемых параметров может быть не более максимального количества аргументных мест во фрагментах, равного 5.

4. ВСТРОЕННЫЕ ФРАГМЕНТЫ , СПЕЦФРАГМЕНТЫ.
(Шарнин М. М.)

Как уже говорилось, в системе ДЕКЛАР многие функции поддерживаются за счет знаний. Сказанное относится к системе команд, лингвистическим процессорам, поисковым операциям и т. д. Однако, для реализации некоторых видов деятельности требуются сугубо программные средства. Это прежде всего функции обращения к внешним устройствам (диску, терминалу), разного рода операции очистки памяти, блокировки, включения-выключения программных модулей и др. Сюда же входят некоторые специальные операции над сетями, арифметические операции и др.

Для обращения к программным средствам служат встроенные фрагменты. В языке ДЕКЛ перед такими фрагментами ставится оператор B: - "выполнить".

Обращение к той или иной программе (процедуре) идет по имени отношения. При этом переменные, стоящие на аргументных местах встроенного фрагмента, служат для передачи параметров.

Аналогом встроенных фрагментов являются встроенные предикаты языка ПРОЛОГ. Ниже будут рассмотрены основные встоенные фрагменты системы ДЕКЛАР, работающие во всех ее версиях. Полный перечень таких фрагментов (в том числе специфичных для версий на машинах класса IBM-PC) приведен в приложении 2.

Будем различать два типа встроенных фрагментов:

А. Встроенные фрагменты, осуществляющие означивания переменных. К ним относятся:

+(...,...,...)	- сложить;
*(...,...,...)	- умножить;
DIV(...,...,...)	- разделить;
M-(...,...,...)	- вычесть;
>(...,...)	- больше;
<(...,...)	- меньше;
>=(...,...)	- больше или равно;
<=(...,...)	- меньше или равно;
=(...,...)	- равно;
#(...,...)	- не равно;
SELF(...,...)	- означивание именем переменной;
I(...)	- считывание символа с клавиатуры;
SET(...,...,...)	- установка требуемой константы на определенное место фрагмента;
ARG(...,...,...)	- выделение компонент фрагмента (аргументов).

Перечисленные фрагменты могут стоять только в левой части продукции, где перед ними ставится оператор B:.

Б. Встроенные фрагменты, реализующие функции связи с внешними устройствами, а также разного рода вспомагательные операции - стирания, блокировки и т. д. Они могут стоять только

в правых частях продукций. Такие фрагменты не означивают переменные, но используют результаты выполненных означиваний.

Помимо встроенных фрагментов в системе ДЕКЛАР имеются так называемые спецфрагменты. Их нахождение в ОП вызывает срабатывание тех или иных программных блоков. В отличие от встроенных фрагментов перед ними не ставится оператор В:. Достаточно поместить спецфрагмент в ОП, чтобы соответствующий программный блок включился.

4.1. Арифметические операции и операции над именами

Ниже будут рассматриваться встроенные фрагменты, реализующие следующие операции:

+, *, DIV, м-, >, >=, <, <=.

Все такие встроенные фрагменты могут стоять только в левой части продукции.

Примечание. Напомним, что в данной версии системы допускается использование только целых положительных чисел, хотя в процессе выполнения арифметических операций могут возникать и целые отрицательные числа, которыми означиваются переменные. Использование вещественных чисел не допускается.

Сложение

+(<число>, <число>, <переменная>) - вызывает сложение двух указанных целых чисел. Результатом является число, которым означивается указанная переменная.

Умножение

*(<число>, <число>, <переменная>) - вызывает умножение двух указанных целых чисел. Результатом является число, которым означивается указанная переменная.

Вычитание

М-(<число>, <число>, <переменная>) - вызывает вычитание из первого числа второго. Результатом означивается указанная переменная. Если первое число меньше второго, то результатом будет отрицательное число.

Деление

DIV(<число>, <число>, <переменная>) - вызывает деление первого числа на второе. Результат, который округляется в большую сторону и становится целым числом, означивает указанную переменную.

В рассмотренных фрагментах на первых двух аргументных местах могут стоять не числа, а переменные, которые перед выполнением оператора должны быть означены целыми числами.

Пример 4.1

```
ЕСЛИ FF* B:+(5,30,X1) B:*(X1,6,X2) ТО ;
```

Применение данной продукции вызовет следующие действия (5+30)*6=X2. Результатом будет целое число 900, которым означится переменная X2.

Пример 4.2

```
ЕСЛИ GR* D:GRAD(X1) B:*(X1,X1,X2) ТО B:A(X2) ;
```

Данная продукция реализует команду возведения в квадрат числа, содержащегося во фрагменте GRAD(X1). Пусть в ОП введен фрагмент GRAD(6). Тогда рассматриваемая продукция сделается применимой. Переменная X1 будет означена числом 6. Выполнение оператора в:*(X1,X1,X2) приведет к вычислению 6*6=36 и означиванию X2 числом 36. За счет в:A(X2) это число будет выдано на экран.

Больше

>(<число>,<число>) - вызывает сравнение двух указанных чисел. Выполнение оператора B:>(...,...) может быть:

- успешным (если первое число больше второго);

- безуспешным (если первое число меньше или равно второму).

В первом случае применение продукции, у которой в левой части есть такой оператор, продолжается, во втором - заканчивается (продукция считается неприменимой).

В рассматриваемом встроенном фрагменте на аргументных местах могут стоять переменные, которые до обращения к фрагменту должны быть означены целыми числами.

Меньше, больше или равно, меньше или равно

<(<число>,<число>) - первое число должно быть меньше второго;

=<(<число>,<число>) - первое число должно быть меньше или равно второму;

>=(<число>,<число>) - первое число должно быть больше или равно второму.

В остальном все сказанное ранее для фрагмента >(...,...) справедливо и для данных фрагментов.

Пример 4.3

```
ЕСЛИ MOR* FUN(X1,X2) B:DIV(X1,X2,X3) B:>(X3,2) ТО
B:A("Да");
```

Данная продукция проверяет следующее неравенство X1/X2>2. Пусть в ОП находится фрагмент FUN(8,3). Он как бы играет роль команды вычисления указанной функции для заданных аргументов.

Тогда в продукции переменные X1 и X2 будут означены 8 и 3, соответственно. В результате целочисленного деления (DIV) будет получено число 3, которое означит переменную X3. Так как 3>2, то выполнение B:>(X3,2) будет успешным, а продукция применимой. На экран будет выдано слово - "Да".

4.2. Операции над именами

Такие операции задаются встроенными фрагментами:

=(...,...) - проверка равенства имен (констант, чисел);

#(...,...) - проверка неравенства;

SELF(...,...) - означивание именем переменной или константой;

FR(...,...,...) - выделение по имени фрагмента констант, стоящих на аргументных местах, и наоборот;

SET(...,...,...) - установка во фрагменте указанной константы на указанное место;

NEW(...) - создание новой константы (внутреннего кода);

UN(...,...) - слияния, замены имен;

Первые четыре встроенные фрагмента (до FR включительно) могут стоять только в левой части продукции (так как они осуществляют действие проверки и означивания), а оставшиеся - только в правой.

Равенство (режим проверки)

=(<константа>,<константа>) - вызывает проверку на совпадение (графическое равенство) указанных констант. Ими могут быть и целые числа.

На местах констант могут стоять переменные, которые должны быть означены ранее. Тогда осуществляется такая же проверка. При этом выполнение оператора B:=(...,...) может быть:

- успешным (имеет место совпадение);

- безуспешным (нет совпадения).

В первом случае применение продукции с оператором B:=(...,...) в левой части продолжается, а во втором - заканчивается.

Пример 4.4

```
ЕСЛИ JJ* NUM1(X1) NUM2(X2) B:=(X1,X2) ТО...;
```

Продукция будет применимой, если в ОП имеются одноместные фрагменты NUM1(...) и NUM2(...) с одним и тем же аргументом - им может быть константа или число.

Аргументами встроенного фрагмента =(...,...) могут быть символы и "_" (нижняя черточка). Тогда будет происходить сравнение с данным символом (его кодом).

Равенство (режим означивания)

Такой режим имеет место в случае, когда у встроенного фрагмента =(...,...) одна из переменных является неозначенной. Тогда осуществляется ее означивание. Например, если до выполнения оператора B:=(A1, X) переменная X не была означена, то она означивается A1. Аналогично происходит и при последовательном выполнении операторов B:=(X, A1) B:=(X1, X), где X1 будет означена A1. В случае, когда оба аргумента есть неозначенные переменные, не происходит никакого означивания. Применение продукции будет продолжено.

Выполнение оператора B:=(...,...) в режиме означивания всегда является успешным.

Неравенство (режим проверки)

#(<константа>,<константа>) - вызывает проверку на несовпадение (графическое неравенство) указанных констант. При этом на местах констант могут стоять означенные переменные.

Встроенный фрагмент неравенства в режиме означивания не работает. Хотя его выполнение будет считаться успешным, а применение продукции будет продолжено. В остальном - как для равенства (=).

Пример 4.5

```
ЕСЛИ SS* REL(X) X(X1/X3) D:X(X1/X4) B:#(X3,X4) ТО ;
```

Данная продукция обеспечивает выявление в ОП одноместных фрагментов с одним и тем же именем отношения и аргументом. В результате ее применения останется только один из этих фрагментов - первый. При этом имя отношения задается с помощью - REL(...). Например, если в ОП есть REL(R1) REL(R2), то действие продукции будет распространяться только на одноместные фрагменты вида R1(...) и R2(...).

Означивание именем переменной (или константой)

SELF(<переменная>,<переменная>) - вызывает означивание второй переменной именем первой переменной. Например, при выполнении B:SELF(X1, X2) переменная X2 будет означена величиной X1. Такое означивание оказывается необходимым при металогической обработке, когда к именам переменных должны подсоединяться определенного вида фрагменты.

При выполнении B:SELF(A1, X2) переменная X2 будет означена константой A1. При выполнении B:SELF(...,...), у которого на втором месте стоит константа или означенная переменная, никакого означивания не осуществляется.

Выполнение B:SELF(...,...), который может стоять только в

левой части продукции, всегда является успешным. Во всех случаях применение продукции будет продолжено.

Слияние - объединение имен

UN(<константа>,<внутренний код>) - вызывает замену во всех фрагментах, находящихся в ОП, второй константы (внутреннего кода) на первую. Этот фрагмент не осуществляет каких-либо означиваний и ставится в правых частях продукций.

Например, если в ОП имеются фрагменты R1(A1) R2(A1,1+) R3(A3), то выполнение B:UN(A1,1-) приведет к - R1(A1) R2(A1,A1) R3(A1), т.е. 1+ будет замещено на A1. И так во всех фрагментах из ОП, в которые входит константа (внутренний код) 1-.

UN(X1 ,X2) - вызывает те же замены, о которых говорилось выше. Только обьекты замен - это величины, которыми означены X1 и X2.

Выполнение оператора B:UN(...,...) всегда является успешным.

Путем объединения имен во многих случаях может экономно выполняться действие замены фрагментов.

Пример 4.6

ЕСЛИ ММ* LR(X1,X2,X3) D:LR(X3,_,X5) ТО B:UN(X3,X5);

Данная продукция выявляет цепочку, состоящую из двух фрагментов. При этом второй фрагмент удаляется из ОП, а его третий аргумент "сливается" с третьим аргументом первого фрагмента.

Например, если в ОП были фрагменты LR(A1,B1,1+) LR(1-,X2,2+) R(A3), то в результате применения продукции останется - LR(A1,B1,1-) R3(1-).

Ограничения не допускается когда на втором агрументном месте стоит константа - не внутренний код. Тогда будет выдан сигнал об ошибке.

Фрагмент SET (установить определенный аргумент)

Встроенный фрагмент SET имеет следующий формат:

SET(<имя фрагмента>,<номер позиции>,<константа>)

Он помещает указанную константу на указанную позицию фрагмента с указанным именем. Считается, что в позиции 0 находится имя отношения, в позиции 1 - первый аргумент, и т.д.

Например, пусть имеется фрагмент R1(A1,A2/AAA). Тогда выполнение оператора B:SET(AAA,1,B1) приведет к замене этого фрагмента на R1(B1,A2/AAA). Выполнение оператора B:SET(AAA,0,R2) приведет к замене последнего на R2(B1,A2/AAA).

Оператор B:SET(...) может быть только в правой части продукции. При его выполнении пространственное расположение фрагментов в ОП не меняется. Происходит лишь замещение, что осуществляется достаточно быстро. При этом на аргументных местах могут стоять переменные, которые до обращения к B:SET(...) должны быть означены. Если они не означены, то операция замещения не будет выполнена.

Оператор B:SET(...) позволяет реализовать принцип регистровой памяти. В ОП могут быть фрагменты, месторасположение которых остается постоянным, а содержимое - постоянно меняется за счет выполнения обсуждаемого оператора. Например, в таких фрагментах ("регистрах") удобно хранить коды состояний, промежуточные результаты и т.д.

Фрагмент FR (выделения компонент в указанном фрагменте)

Встроенный фрагмент FR имеет следующий формат:

ARG(<имя фрагмента>,<номер позиции>,<константа>).

Он осуществляет поиск по имени фрагмента его аргументов, стоящих на указанных позициях, и наоборот. С помощью этого фрагмента осуществляется означивание переменных. Поэтому он может стоять лишь в левой части продукции.

Например, при наличии в ОП фрагмента R1(A1,A2/AAA) выполнение B:FR(AAA,1,X2) приведет к означиванию X2 константой A1; B:FR(X,1,A1) - к означиванию X константой AAA; а B:FR(AAA,0,X3) - к означиванию X3 константой R1. При многократном выполнении B:FR(...) будут иметь место различные варианты означивания. Например, двукратное выполнение B:FR(AAA,X1,X2) приведет к означиванию X1 и X2 вначале 1 и A1, а затем - 2 и A2.

Оператор B:FR(...) позволяет анализировать фрагменты, находящиеся в ОП, выделять их компоненты. Например, с помощью B:FR(X,_,A1) могут быть выделены все фрагменты, содержащие указанную константу - A1. При многократном выполнении данного оператора перемнная X будет последовательно означиваться именем этих фрагментов.

Фрагмент NEW (создание внутреннего кода)

Выполнение B:NEW(X), стоящего в правой части какой либо продукции, вызовет формирование в ОП нового внутреннего кода, которым означивается переменная X (вне зависимости от ее прежнего значения). Такой оператор необходим, когда в ОП устанавливаются различного рода "метки" - стирания, блокировки (см. ниже). Роль таких меток могут играть не только имена фрагментов, но и новые внутренние коды, создаваемые с помощью B:NEW(<имя переменной>).

Выполнение оператора B:NEW(...) всегда является успешным.

4.3. Пересылка информации из ОП на диск и выдача на экран

Для этой цели служат встроенные фрагменты:

A(...) - вывод указанных строковых констант (а также фрагментов с указанными именами) из ОП на экран или их запись на диск;

OUT(...) - выделение в ОП зоны (по указанным меткам начала и конца) и ее вывод на экран или запись на диск;

REW(...) - управляет работой первых двух встроенных фрагментов и определяет, куда выводить информацию - на экран или на диск (в какой файл);

TT("T") - отменяет действие встроенного фрагмента REW(...), устанавливая выдачу информации на экран;

TT("F") - отменяет действия предыдущего встроенного фрагмента, устанавливая выдачу в файл, указанный во фрагменте REW(...);

ВК - спецконстанта вывода на экран с новой строки.

Выполнение B:A(...) приводит к выдаче на экран или записи на диск конкретных констант или фрагментов, указанных с помощью аргументных мест.

Выполнение B:OUT(...) - приводит к выдаче на экран или записи на диск множества фрагментов. При этом на аргументных местах OUT(...) указывается, от какого места в ОП и до какого брать эти фрагменты. Роль указателей играют имена фрагментов.

Выполнение в:REW(...), B:TT("T") и B:TT("F") устанавливает источник, куда осуществляется выдача (запись) информации. Если не выполнялся ни один из этих операторов, то по умолчанию подразумевается, что фрагменты A(...) и OUT(...), если в последнем специально не указывается имя файла, осуществляют выдачу информации на экран.

Остановимся на перечисленных фрагментах более подробно.

REW("<имя файла>","<расширение>") - устанавливает запись информации в файл <имя файла>.<расширение>. Например, выполнение B:REW("PROD","Z") устанавливает вывод информации в файл PROD.Z. Такой файл как бы "открывается" для записи.

TT("T"), TT("F") - первый отменяет, а второй восстанавливает действие встроенного фрагмента REW(...). Выполнение B:TT("T") отменяет такое действие, устанавливая выдачу на экран, а B:TT("F") - восстанавливает его. Будет осуществляться запись в "открытый файл", указанный в REW(...).

Фрагмент "A" (вывод на экран слов или указанных фрагментов)

A("<строковая константа>") - осуществляет вывод на экран (или диск) цепочки символов, записанной в кавычках. Строковой константой может быть достаточно длинный текст, не содержащий внутри двойных кавычек.

Например, выполнение B: A("Вводите текст *>") вызовет вывод на экран сообщения - Вводите текст *> (без кавычек). Если ранее был выполнен B: REW("ISS","Z"), то сообщение будет записано в конец файла ISS.Z.

A(<имя фрагмента>) - осуществляет вывод из ОП на экран или запись на диск фрагмента с указанным именем. Если такой фрагмент в ОП отсутствует, то ничего не будет выдано на экран или записано на диск.

Например, если в ОП есть фрагмент RR(EE,CC/KK), то оператор B: A(KK) вызывает выдачу его на экран (или запись на диск). Если в ОП нет фрагмента с именем KK, то ничего не будет выдано.

Выполнение оператора B: A(...) всегда является успешным.

A(X) - осуществляет вывод на экран или диск константы, которой была означена X.

Встроенный фрагмент A(...) может иметь до пяти аргументных мест, на которых могут стоять строковые константы, имена фрагментов или же переменные, которые должны быть означены.

Пример 4.7

```
ЕСЛИ UR1 D: RR(X1,_) ТО BK B: A("ARG1 - ",X1);
```

Многократное применение такой продукции приведет к выдаче на экран всех первых аргументов бинарного отношения RR. Перед каждым таким аргументом будет стоять слово ARG1-, начинающееся с новой строки. При этом выдаче будет предшествовать удаление из ОП фрагментов RR(...). Так если в ОП были фрагменты RR(BB,CC), RR (EE,CC/KK), то они будут удалены из ОП и на экране появится:

```
ARG1 - BB
ARG1 - EE
```

Пример 4.8

```
ЕСЛИ UK1 PRINT(X1) X1(_/X2) ТО BK B: A(X2) ;
```

С помощью этой продукции осуществляется выдача на экран всех одноместных фрагментов с указанным именем отношения. Каждый такой фрагмент появится на экране с новой строки - на это указывает спецконстанта BK. Так, если ввести в ОП фрагмент PRINT(RR) и в ОП есть одноместный фрагмент вида RR(.../KK), то продукция сделается применимой. Данный фрагмент будет выдан на экран.

Если в ОП есть одноместные фрагменты RR(...), то продукция тоже будет применимой. На экран будет выдано RR(.../N+), где N+ - некоторое число (внутрисистемный код).

Фрагмент OUT (вывод зоны из ОП на экран или на диск)

Зоной называется находящаяся в ОП информация (множество фрагментов), в начале и в конце которой стоят фрагменты с

определенными именами. Такие фрагменты называются метками начала и конца зоны. Отметим, что роль этих меток могут также играть внетренние коды, создаваемые с помощью оператора B: NEW(...).

Фрагмент OUT может работать в двух режимах:

- в первом выдается на экран (или выводится на диск) зона вместе с метками начала и конца,

- а во втором - без этих меток.

У фрагмента OUT может быть три аргументных места (когда зона должна выдаваться вместе с метками) или четыре (без меток). Рассмотрим вначале первый из них.

OUT(<имя файла>,<имя фрагмента>,<имя фрагмента>), где:

<имя фрагмента> --> <константа> <переменная>

- осуществляет вывод из ОП информации, находящейся между фрагментами с указанными именами, и запись ее на диск в файл <имя файла>.Z. При этом константы - это имена фрагментов, отмечающих начало и конец считываемой информации, т.е. зоны. Фрагменты, ограничивающие зону, также записываются на диск. Информация, записанная на диск, сохраняется и в ОП.

Отметим, что когда во фрагменте OUT(...), указано имя файла, то действие B: REW(...) на него не распространяется. Вывод зоны осуществляется только в файл.

Пример 4.9

Оператор B: OUT(AAA, X1, X2) обеспечивает поиск в ОП зоны и запись ее на диск. Пусть X1 и x2 были означены константами W1 и W2. Тогда в ОП ищутся фрагменты вида ...(.../W1) и ...(.../W2), ограничивающие зону. Все, что находится между этими фрагментами (вместе с ними, но кроме фрагментов вида L(...), Z(...) и _(_/...)), выводится на диск в файл AAA.Z.

Если файл AAA.Z, в который должен быть записан новый раздел знаний, уже был на диске, то прежняя информация стирается и заменяется на новую.

Примечание. В данной версии ДЕКЛАР знания имеют статус файловых данных, защита от стирания которых является функцией ОС, а не ДЕКЛАР. В последующих версиях системы данная концепция будет существенно переработана.

Выполнение оператора B: OUT(...) может быть успешным или ошибочным. Если хотя бы один фрагмент с указанным именем не найден, то вывод осуществляется неправильно или вовсе не осуществляется. При этом информация, находящаяся в указанном файле, может быть стерта.

Следует помнить, что при формировании нового раздела знаний и его выводе на диск, раздел должен быть соответствующим образом оформлен, т.е. в начале должен стоять фрагмент BEG(.../W1), а в конце- END(.../W2).

OUT (X1, X2, X3) - делается то же самое, что было описано выше, только в качестве имени файла и имен фрагментов берутся константы, означивающие X1, X2 и X3.

Пример 4.10

K2: ЕСЛИ UK1 D: OUT(X1, X4) BEG(X1/X2) END(X1/X3) ТО B: OUT(X1, X2, X3) ;

С помощью этой продукции поддерживается двухместная команда OUT(...) - записать раздел с указанным именем на диск в указанный файл. При вводе с Клавиатуры фрагмента OUT(AAA, II) и наличии в ОП меток BEG(AAA/...), END(AAA/...) продукция сделается применимой. В результате все фрагменты, находящиеся между этими метками, будут записаны в файл II.Z.

OUT(_, <имя фрагмента>, <имя фрагмента>) - в отличие от того, что было сказано ранее - осуществляет выдачу зоны (множества фрагментов) вместе с метками начала и конца не на диск, а на экран, на что указывает символ "_" (нижняя черточка). При этом на OUT(_,...) начинает распространяться действие оператора B: RGW(_); Если им для записи открыт какой-либо файл, то при выполнении B: OUT(_,...) осуществляется вывод указанной зоны из ОП уже не на экран, а в конец файла (с записью на диск), открытого оператором B: REW(...).

Прейдем к рассмотрению второго режима.

Если во фрагменте OUT(...) задействовано четыре аргументных места (при этом не важно, что стоит на четвертом аргументном месте), то зона выдается без меток начала и конца, т.е. ограничивающих ее фрагментов. В остальном режим полностью аналогичен описанному ранее.

Например, выполнение B: OUT(AAA, X1, X2, _), где X1 и x2 означены именами фрагментов, вызовет запись из ОП на диск зоны без упомянутых фрагментов, т.е. без меток начала и конца.

Пример 4.11

ЕСЛИ UR1 D: OUT(X) BEG(X/X1) END(X/X2) ТО B: OUT(_, X1, X2, _);

С помощью этой продукции поддерживается одноместная команда OUT выдачи на экран зоны ОП, содержащей указанный раздел знаний. Метки начала и конца зоны не выводятся.

4.4. Ввод в ОП знаний и их компоновка

Для этой цели служат следующие встроенные фрагменты:

IN(...) - ввод информации с диска или с клавиатуры в ОП;

VST(...) - вставка зоны в указанное место;

UP(...) - перепись зоны в конец ОП.

Все перечисленные фрагменты могут стоять только в правой части продукции. Они позволяют компоновать имеющуюся в ОП информацию, обеспечивая запись новой информации и ее пересылку в нужное место.

Фрагмент IN (ввод информации в ОП)

IN(<имя файла>,<раздел>) - осуществляет ввод с диска в ОП указанного раздела знаний, записанного на РСС или ДЕКЛ. Такой раздел берется из файла <имя файла>.Z.

Например, оператор B:IN(LIN,LIN) вызовет считывание в ОП раздела LIN, находящегося на диске в файле LIN.Z.

Выполнение B:IN(...) может быть:

- успешным, когда требуемые файл и раздел найдены и во вводимых знаниях нет синтексических ошибок (т.е. записи удовлетворяют синтаксису языка РСС или ДЕКЛ);

- ошибочным, когда файл и раздел найдены, но в разделе есть синтаксические ошибки, или же когда в файле нет указанного раздела;

- привести к прекращению работы, если требуемого файла вообще нет на диске.

В первых двух случаях применение продукции продолжается, а в последнем работа ДЕКЛАР прекращается и управление получает монитор ОС. Во втором случае выдается сообщение об ошибке. В ОП правильно будут считаны лишь фрагменты, стоящие до ошибочного. Остальные могут быть считаны неправильно. В случае отсутствия требуемого раздела или знака его начала ($) ничего не будет считано в ОП. Напомним, что этот знак в сочетании с $(<имя раздела>) должен стоять в строке на первой позиции.

IN(X,X1) - вызывает ввод с диска в ОП раздела знаний. В качестве имен файла и раздела берутся константы, которыми означены X и X1.

IN() - вызывает ввод в ОП сети, набранной с помощью клавиатуры на языке РСС или ДЕКЛ. Допускается также ввод одной или множества констант (слов).

Выполнение B:IN() может быть:

- успешным, когда информация на языке РСС или ДЕКЛ правильно набраны на клавиатуре;

- ошибочным, когда во вводимых знаниях допущена синтаксическая ошибка;

- с переходом на режим ожидания, когда на клавиатуре ничего не набрано.

В первых двух случаях выполняются те же действия, что и для

IN(...). Последний случай возникает, когда на клавиатуре не было набрано никакой информации или же не была нажата клавиша ENTER. Система будет ждать, пока какая-либо информация не появится в буфере. При поступлении информации сделается возможным выполнение B:IN(). Применение продукции будет продолжено.

При выполнении оператора B:IN() к именам всех вводимых с клавиатуры фрагментов, а также к отдельным вводимым константам, привешивается метка @@(...). Например, если в ОП введен фрагмент R1(A1/DD) и константа AA, то дополнительно в ОП будут сформированы фрагменты @@(DD) и @@(AA). Если введен фрагмент R1(A1), то дополнительно будет сформирован фрагмент @@(N+), где N+ - внутрисистемное имя первого фрагмента.

Привешивание меток @@(...) необходимо для выделения в ОП только что введенных фрагментов и констант.

Пример 4.12

SS1:ЕСЛИ L1 D:MT() ТО B:A("вводите условие *>") B:IN() ;

Данная продукция будет применимой, если в ОП есть фрагмент MT(). Ее применение вызовет выдачу на экран текста, что осуществляется с помощью в:A(...), и переход в режим ожидания - за счет B:IN(). После того, как пользователь наберет на клавиатуре соответствующую информацию и нажмет ENT- ER, работа будет продолжена и информация введена в ОП. Продукция закончит свое применение.

Фрагмент VST (вставка зоны в указанное место)

VST(<имя фрагмента>,<имя фрагмента>,<имя фрагмента>) - осуществляется поиск в ОП зоны, началом и концом которой являются первые два из указанных фрагментов. Эта зона (без упомянутых фрагментов) переносится в другое место - вставляется перед третьим фрагментом. Например, пусть в ОП имеются фрагменты:

BEQ(UL/2+)... END(UL/3+)... R1(AA/4+)

Тогда выполнение B:VST(2-,3-,4-) вызовет выделение в ОП всех фрагментов, которые находятся между первыми двумя. Выделенные фрагменты будут перенесены и встанут перед R1(AA/4+).

Пример 4.13

ЕСЛИ UR1 D:UNION(X1,X2) BEG(X1/X3) END(X1/X4) END(X2/X5) ТО B:VST(X3,X4,X5);

Данная продукция реализует команду объединения двух разделов, находящихся в ОП. Первый раздел (с именем X1) помещается в конец второго раздела (с именем x2). Будет расширен второй раздел. Например, если на вход подать команду UNION(GR1,GR2), то при наличии указанных разделов в ОП продукция сделается применимой. В результате, содержимое первого раздела, т.е. все, что находится между BEG(GR1) и END(GR1), будет поставлено перед END(GR2) (в конец второго раздела). Сами метки BEG(GR1) и END(GR1) будут удалены из ОП. Останутся только метки второго раздела.

Фрагмент UP (перепись зоны в конец ОП)

UP(<имя фрагмента>,<имя фрагмента>) - осуществляются те же действия, что и для VST(...), только указанная зона перемещается в конец ОП. Она встает вслед за последним фрагментом, находящимся в отведенной под знания части ОП.

4.5. Удаление информации, очистка зон

В системе ДЕКЛАР имеется только одно хранилище текущей информации - часть ОП, отведенная под знания. Все, что делается системой, запоминается в ОП в виде множеств фрагментов. После выполнения какой-либо команды или решения задачи в ОП останутся все промежуточные результаты. Чтобы подготовить систему к следующему этапу решения или очередной задаче, требуется удаление "отработавших" фрагментов, очистка соответствующих зон. При этом фрагменты, необходимые для выполнения следующих этапов, должны быть оставлены в ОП. Для этой цели служат встроенные фрагменты:

DEL(...) - удаление зоны из ОП (очистка);

C(...) - удаление последней части знаний из ОП;

SQU() - сжатие семантической сети в ОП.

Помимо этого имеются:

H(...) - спецфрагмент ("метка"), защита от удаления;

DELH(...) - встроенный фрагмент удаления меток защиты.

Перечисленные встроенные фрагменты могут быть только в правой части продукции. Перед ними ставится оператор B: - "выполнить".

Фрагмент DEL (удаление зоны из ОП); Может быть использован для удаления отдельных фрагментов.

DEL(<имя фрагмента>,<имя фрагмента>) - вызывает удаление из ОП указанной зоны. Имена фрагментов играют роль меток начала и конца очищаемой зоны. Эти метки тоже удаляются из ОП.

Например, выполнение в: DEL(W1,W2) вызовет поиск в ОП фрагментов вида ...(.../W1) и ...(.../W2). Все, что дится между ними (вместе с ними) удаляется из ОП.

Выполнение оператора B: DEL(...) может быть только успешным.

Если в ОП не найден второй фрагмент (нет метки конца зоны), то очищается часть ОП, которая начинается с первого фрагмента и до конца.

Если в ОП не найден первый фрагмент (нет метки начала зоны), то очистка не производится. Никакие фрагменты не удаляются из ОП.

DEL (<имя фрагмента><имя фрагмента>,H) - вызывает удаление из ОП указанной зоны, кроме фрагментов с метками защиты, см. ниже.

DEL(<имя фрагмента>) - осуществляется удаление из ОП фрагмента с указанным именем. Например, при выполнении B:DEL(AA1) из ОП будет удален фрагмент с именем AA1.

Ограничение. С помощью таких операторов нельзя удалять фрагменты, имена которых являются аргументами каких-либо других фрагментов. Например, если в ОП имеется R1(KH/DD) R2(DD), то выполнение B:DEL(DD) не вызовет какого-либо удаления.

Фрагмент "C" (удаление последней части знаний)

B:C(<имя фрагмента>) - вызывает удаление из ОП зоны, которая начинается с указанного фрагмента, и кончается последним находящимся в ОП фрагментом (первый из упомянутых фрагментов играет роль метки начала, он тоже удаляется из ОП).

B:C(<имя фрагмента>H) - вызывает удаление из ОП зоны, кроме фрагментов с меткой "защиты", см. ниже.

Выполнение оператора B:C(...) всегда является успешным. Если нет метки начала, то его выполнение не осуществляется.

Пример 4.14

```
ЕСЛИ DD* MT1(_/X1) MT2(_/X2) MT3(_/X3) TO B:DEL(X1,X2)
B:C(X3);
```

При применении данной продукции в ОП будут найдены фрагменты MT1(...), MT2(...) и MT3(...). Будет стерто все, что находится между первыми двумя (вместе с ними) и после третьей. При этом роль меток будут играть первые из найденных фрагментов.

```
ЕСЛИ DDD* MM1(A1/X1)...TO...B:DEL(X1);
```

В данном случае из ОП будет удален лишь один фрагмент - MM1(A1). Операторы типа B:DEL(X1) позволяют осуществлять удаление фрагментов не сразу после того, как продукция оказалась применимой, а по мере выполнения действий ее правой части.

```
ЕСЛИ DD*...TO M1(X1/X2)...B:DEL(X2,X2);
```

Данная продукция вначале добавляет в ОП фрагмент вида M1(...), затем будут выполнены какие-то другие действия (на что указывает многоточие), а в конце этот фрагмент будет удален из ОП.

Спецфрагменты H(...)

H(<константа>) - наличие этого спецфрагмента в ОП означает, что если в очищаемой (с помощью B:C(...,H) или B:DEL(...,H))

зоне есть фрагмент, именем которого является указанная константа, то такой фрагмент доллжен быть оставлен в ОП. Имеет место защита от удаления. Такая защита не распространяется на операторы D:, B: C(<имя фрагмента>).

Например, пусть в ОП имеются фрагменты R1(A1/DD) и H(DD). Тогда при удалении из ОП зоны, куда они входят (если такое удаление осуществляется операторами с константой H), останутся оба эти фрагмента. Если же применилась продукция, в левой части которой были операторы D: R1(X1/X2) D: H(X2), то оба фрагмента будут удалены из ОП.

DELH(<имя фрагмента>) - встроенный фрагмент удаления меток защиты. Такое удаление осуществляется начиная от указанного фрагмента.

Например, выполнение B: DELH(KK) вызовет удаление в ОП всех фрагментов вида H(...), расположенных после фрагмента с именем KK. Далее, если выполнить B: C(KK, H), то из ОП будут удалены все фрагменты, начиная от KK (вместе с ним).

Фрагмент SQU (сжатие сети)

SQU() - осуществляется сжатие всей семантической сети, находящейся в ОП.

Применение продукций, у которых есть операторы D:, а также выполнение B: DEL(..), приводит к образованию в ОП свободных мест, т.е. "пустот". Для их удаления необходимо выполнить B: SQU(). Это приведет к увеличению свободной части ОП, в которой могут располагаться новые знания. Напомним, что в системе ДЕКЛАР (для IBM/PC и ЕС-1840) под знания отведено 64 кбайт ОП. При решении прикладных задач, требующих большого объема данных, память (ОП) нужно использовать достаточно экономно.

SQU(<имя фрагмента>) - осуществляется сжатие в ОП семантической сети, начиная от указанного фрагмента.

Пример 4.15

ЕСЛИ D* MT1(X/X1) ... ТО B: SQU(X1);

Применение данной продукции приведет к сжатию в ОП семантической сети, начиная от фрагмента MT1(...).

Ограничения. Переменная "X" в операторе B: SQU(X) должна означиваться в той же самой продукции. Не допускается передачи из другой продукции метки, указывающей начало сжатия.

4.6. Операции с дисковой памятью

В системе ДЕКЛАР все знания хранятся на диске и в случае необходимости пересылаются в ОП или вызываются на экран. Ранее уже были описаны средства записи на диск (OUT). Ниже будут рассмотрены следующие встроенные фрагменты:

F(...) - выдача раздела знаний с диска на экран;

UNB(...) - выдача файла с диска на экран;

REN(...) - копирование файлов на диске;

OUT1(_) - запись знаний на диск в загрузочной форме.

Фрагмент F (выдача с диска на экран)

F(<имя файла>,<раздел>) - вызывает чтение указанного во фрагменте раздела знаний из файла, находящегося на диске (напомним, что именем файла может быть лишь константа, состоящая из латинских букв и цифр). Раздел знаний может быть записан на языках - РСС, ДЕКЛ, или содержать какие-либо тексты. Знания считываются из файла <имя файла>.Z.

Например, выполнение B:F(PROD,GRFP) вызовет вывод на экран раздела знаний GRFP, находящегося на диске в файле PR- OD.Z. Данный раздел состоит из продукций, записанных на языке ДЕКЛ.

Выполнение оператора B:F(...,...) может быть:

- успешным (если требуемые файл и раздел найдены);
- ошибочным (если файл найден, но в нем нет указанного раздела);
- приводить к прекращению работы системы ДЕКЛАР (если файла нет на диске).

F(X,X1) - вызывает выдачу с диска на экран раздела знаний. В качестве имени файла и имени раздела берутся константы, которыми означены X и X1.

Встроенный фрагмент F(...,...) широко используется для выдачи анкет, меню, осуществляемой в процессе диалога. За счет продукций такая выдача может быть легко связана с выполнением определенных условий.

Пример 4.16

```
ЕСЛИ TERZN(X) KTL(X,X1) ТО B:F(X1,X) ;
```

Активизация данной продукции с помощью оператора T1:TERZN(UL) вызовет поиск по каталогу (KTL) имени файла, в котором находится раздел UL. Пусть в KTL.Z имеется фрагмент KTL(UL,PROD). Тогда продукция будет применимой. Выполнение B:F(PROD,UL) приведет к считыванию из файла PROD.Z раздела UL и вывод его на экран терминала.

Фрагмент UNB (выдача файла с диска на экран)

UNB("<имя файла>","<расширитель>") - осуществляется выдача на экран файла <имя файла>.<расширитель>.

Например, выполнение B:UNB("PROD","Z") вызовет выдачу на экран содержимого PROD.Z, где находится несколько разделов знаний.

На UNB(...) распространяется действие B:REW(...) - открыть файл для записи. Если такой файл открыт, то вместо выдачи информации на экран осуществляется ее запись в этот файл. Здесь также с помощью B:TT("T") и B:TT("F ") может блокироваться или возобновляться действие B:REW(...).

Фрагмент REN (копирование файлов на диске)

REN("<имя файла>","<расширитель>") - вызывает копирование указанного файла <имя файла>.<расширитель> в файл <имя файла>.BAK.

Например, выполнение B:REW("GRFF","Z") вызовет копирование файла GRFF.Z в GRFF.BAK. Информация, находящаяся в GR- FF.Z, сохраняется. Подобные действия необходимы в случае, когда использование каких-либо разделов знаний, находящихся в одном файле, связано с их изменением. Тогда в ряде случаев нужно сохранять "первоисточник".

Фрагмент OUT1 (запись знаний на диск в загрузочной форме)

OUT1() - осуществляется запись всех знаний, которые находятся в ОП, на диск в так называемой "загрузочной" форме, т.е. в виде цепных списков. Знания записываются в файл DECLAR.ZNN.

Дело в том, что при считывании в ОП знаний, записанных на РСС и ДЕКЛ, они преобразуются в чисто машинное представление, где имеют место уже не имена констант и переменных, а цепные списки, всевозможные переадресации и т.п.

Для обеспечения быстрого ввода можно воспользоваться "загрузочной" формой. С помощью B:OUT1() машинное представление знаний может быть записано в файл DECLAR.ZNN. Тогда при считывании этих знаний в ОП уже не будет требоваться трудоемких операций перекодировки имен с построением цепных списков. Загрузка знаний будет осуществляться гораздо быстрее.

Оператор B:OUT1() может быть использован при первоначальной загрузке в ОП разделов знаний. Таким образом в существующей версии системы осуществляется загрузка общесистемных разделов - SYS, SSD, SDD.

4.7. Посимвольная обработка литеральных конструкций

Для организации посимвольной обработки литеральных цепочек служат встроенные фрагменты:

R() - определяет источник, откуда должна считываться информация;

G() - сдвигает указатель позиции, из которой может быть считан символ;

I(X) - считывает символ из указанной позиции;

LN() - перемещает указатель на новую строку (в первую позицию); EOLN() -проверяет, имеет ли место конец записи (строки).

Перечисленные действия необходимы для побуквенного считывания информации (с экрана или из файла) с построением сети, представляющей так называемое пространственное расположение между буквами. Работа над списками и текстами сводится к преобразованию подобных сетей.

Фрагмент I(X), вызывающий означивание переменной X, может стоять только в левой части продукции, а остальные фрагменты - в правой.

Фрагмент "R" (указание адреса литеральной цепочки)

R() - указывает, что информация должна считываться с клавиатуры. Подобное указание распространяется лишь на операторы B:G(), B:I(X), B:LN() и B:EOLN(). Как бы устанавливается режим их работы. Источником считываемых символов будет буфер клавиатуры.

Напомним, что информация в буфере клавиатуры становится доступной только после нажатия клавиши ENTER.

Выполнение B:R() всегда будет успешным.

R(<имя файла>) - указывает, что информация должна считываться из файла <имя файла>.Z.

В этом случае действия, вызываемые B:G(), B:I(X), B:LN() и B:EOLN будут направлены на работу с указанным файлом. Например,

B:R(LL1) - есть указание, что символы должны считываться из файла LL1.Z.

Выполнение B:R(<имя файла>) может быть:

- успешным;
- приводить к прекращению работы ДЕКЛАР (если на диске нет требуемого файла).

Фрагмент "I" (чтение символа из указаной позиции)

I(<переменная>) - вызывает считывание символа, на который направлен указатель. Этот символ означивает переменную.

Например, если указатель направлен на букву "Д", то выполнение B:I(X) приведет к означиванию X этой буквой.

Выполнение B:I(<переменная>) всегда успешно.

I("<символ>") - вызывает проверку того, что указатель направлен на отмеченный символ.

Выполнение B:I("<символ>") может быть:

- успешным (если указатель направлен на отмеченный символ);

- безуспешным (если указатель направлен на другой символ).

Например, выполнение B:I("S") будет успешным, если указатель направлен на символ "S", и безуспешным - во всех других случаях.

Фрагмент "G" (сдвиг указателя на следующую позицию)

G(I) - вызывает перемещение указателя на следующую позицию. Перемещение осуществляется только в случае, если на следующей позиции есть какой-нибудь символ. Если его нет, то осуществляется переход в режим ожидания. Процесс применения продукции с оператором B:G() будет продолжен с появлением новых символов.

Итак, выполнение в:G() может быть:

- успешным;

- приводить к переходу в режим ожидания (заполнения буфера экрана или акции считывания из файла).

Источник, на который направлены действия B:G() и B:I(...), определяется оператором B:R(...). По умолчанию предполагается работа с клавиатурой.

Пример 4.17

ЕСЛИ T01 D:MD(X1) B:I(X2) TO LR(X1,X2,X3) MD(X3) B:G() ;

С помощью этой продукции, вызываемой посредством T:T01, осуществляется посимвольное считывание информации с построением пространственной структуры. При этом, если ранее не выполнялся оператор B:R(<имя файла>), то считывание будет осуществляться с клавиатуры.

Пусть в ОП имелся фрагмент MD(1+), а на клавиатуре были набраны символы "ABC". Тогда нажатие ENTER приведет к их чтению в буфере и трехкратному применению продукции. В результате будет сформирована сеть:

LR (1-,"A",2+) LR(2-,"B",3+) LR(3-,"C",4+) MD(4-)

Это и есть пример пространственной структуры (ПС). Метка MD(...) указывает на последний элемент строящейся ПС. Вначале будет MD(1+). В результате первого применения продукции будет считан символ "A", построен фрагмент LR(1+,"A",2+), а метка MD(...) будет сдвинута, т.е. фрагмент MD(1-) будет удален из ОП, а MD(2+) добавлен. За счет B:G() указатель будет сдвинут на следующую позицию. Не трудно видеть, что пространственная структура цепочки, состоящей из трех символов, будет сформирована трехкратным применением указанной продукции. Система перейдет в режим ожидания, когда все символы будут считаны.

Фрагмент LN (перемещение указателя на следующую строку)

LN() - вызывает перемещение указателя на первую позицию

новой строки. Все символы, оставшиеся в предыдущей строке, как бы игнорируются (можно считать - удаляются). Система готова к восприятию новой информации.

Выполнение B:LN() может быть только успешным. Источник, на который направлены действия, определяется оператором B:R(...).

Пример 4.18

SS2:ЕСЛИ UK1 B:I(X2) ТО MTT(X2) B:LN() ;

Эта продукция прочитает с буфера клавиатуры первый символ, поставит на него метку MTT(...) и перейдет к следующей строке. Остальные символы текущей строки будут игнорированы.

Фрагмент EOLN (проверка конца записи)

Выполнение B:EOLN() будет:

- успешным, если указатель находится в позиции, за которой в данной строке нет больше каких-либо символов. В самой позиции должен быть символ пробела (он всегда автоматически добавляется к считанной строке - при нажатии ENTER);

- безуспешным во всех остальных случаях.

Данный фрагмент необходим для установления момента, когда закончено считывание символов и требуется вызов соответствующих отработавших продукций.

4.8. Управление режимом работы

Ниже будут рассмотрены следующие спецфрагменты:

@BL(...) - спецфрагмент блокировки зон оп;

PAR(...) - встроенный фрагмент, вызывающий включение или выключение дополнительных программных блоков системы ДЕКЛАР.

Во многих случаях заранее известно, к каким разделам знаний или зонам в ОП должны применяться (или не применяться) те или иные продукции. Чтобы сделать такое применение более направленным необходимы специальные средства. В системе ДЕКЛАР их роль играют спецфрагменты @B L (...). Они блокируют зоны от применения продукций, т.е. фрагменты, входящие в эти зоны, не рассматриваются в процессе такого применения. Подобные спецфрагменты позволяют исключить побочные эффекты, связанные с применением одних продукций к другим. С их помощью можно значительно уменьшить объем работы, связанной с попытками применения продукций. Для этого достаточно заблокировать зоны, к которым те или иные продукции заведомо не должны применяться, а потом уж активизировать данные продукции.

Фрагмент @BL (блокировка зоны)

@BL(<имя фрагмента>,<имя фрагмента>) - наличие этого спецфрагмента в ОП вызывает блокировку зоны от применениия продукций. Метками начала и конца зоны являются фрагменты с указанными именами.

Например, наличие в ОП фрагмента @BL(W1,W2) сделает невозможным применение всех продукций к фрагментам, находящимся в ОП между ...(.../W1) и ...(.../W2), в том числе и к этим меткам. Если в ОП отсутствует одна из указанных меток, то блокировка не срабатывает.

@BL(<имя фрагмента>) - наличие этого спецфрагмента в ОП вызывает блокировку от применения продукций зоны, которая начинается с указанного фрагмента. Блокировка распространяется до конца ОП, отведенной под знания.

Например, наличие в ОП спецфрагмента @BL(W1) сделает невозможным применение продукций к фрагментам, находящимся в ОП после ...(.../W1).

Если в ОП отсутствует указанный фрагмент, то при вводе @BL(...) формируется фрагмент _(/W1) и ставится позади всех фрагментов.

Снятие блокировки осуществляется:

- удалением фрагмента @BL(...) из ОП;

- командой PAR("4" "5"), см. ниже.

Пример 4.19

Пусть в ОП уже был считан раздел знаний с именем AAA. Пусть известно, что продукции, например с индикатором UU1, заведомо не должны применяться к указанному разделу. Тогда их активизация может осуществляться следующим образом:

```
ЕСЛИ UR1 BEG(AAA/X1) END(AAA/X2) ТО @BL(X1,X2) T:UU1 ;
```

За счет левой части продукции в ОП выделяются метки начала и конца раздела AAA. За счет правой части осуществляется блокировка зоны ОП, в которой находится этот раздел, и активизация других продукций.

Программное ядро системы ДЕКЛАР состоит из блоков или процедур. Некоторые из них выполняют специальные функции - трассировки с выдачей на экран тех или иных сообщений и т.д. Их включение в работу осуществляется с помощью оператора B:PAR(...).

Фрагмент "PAR" (изменение состояния системы, ее "параметров")

Встроенный фрагмент PAR(...) имеет два аргументных места - на первое ставится "параметр" (число от 1 до 4), а на втором - "значение параметра" (число от 0 до 5). Параметры с их значениями определяют, какие дополнительные блоки должны быть включены в работу. При этом параметр - "1" определяет вид трассировки;

"2" - выдачу на экран справочной информации о заблокированных зонах (для продукций);

"3" - привешивание меток к введенным с экрана фрагментам;

"4" - снимает или устанавливает режим блокировки зон.

По умолчанию предполагается, что параметр имеет значение "0", т.е. такое значение ставится автоматически, если не было операторов B:PAR(...).
Итак:

PAR("<число 1>","<число 2>"),

где < число 1> -->1, 2, 3, 4;

<число 2> -->0, 1, 2, 3, 4, 4.

PAR("1","2") - вызывает включение блока простой трассировки. Этот блок следит за применением продукций. На экран выдаются продукции, которые оказались применимыми.

B:PAR("1","5") - вызывает включение блока полной трассировки. На экран выдается вся информация, связанная с попытками применения продукций.

B:PAR("1","0") - вызывает выключение блоков трассировки. По умолчанию предполагается, что трассировка выключена, т.е. значение параметра равно "0".

B:PAR("3","5") - вызывает привешивание ко всем фрагментам, вводимым с экрана, меток @@(...). Напомним, что на месте многоточия во фрагменте метки стоит имя вводимого фрагмента.

B:PAR("3","0") - метки @@(...) к вводимым фрагментам не привешиваются. Этот режим предполагается по умолчанию.

Отметим, что к отдельным вводимым константам метки @@(...) привешиваются всегда.

B:PAR("4","5") - вызывает снятие режима блокировки зон, т.е. разблокировку всех заблокированных зон. При выполнении B:PAR("4","5") продукции будут применяться ко всем фрагментам, находящимся в ОП. Спецфрагмент BL(...) игнорируется.

B:PAR("4","0") - вызывает включение режима блокировки зон.

Спецфрагменты BL(...) начинают выполнять свои функции.

Перечисленные режимы работы могут устанавливаться с помощью операторов B:PAR(...), а также в командном режиме с помощью одноименной команды PAR(...) - установить параметр с его значением. Подобная команда поддерживается следующей продукцией:

ЕСЛИ UK1 D:PAR(X1,X2) ТО B:PAR(X1,X2) ;

При вводе команды с экрана будут выполнены действия, которые были описаны ранее.

В версиях системы ДЕКЛАР, работавших на ЭВМ класса IBM-PC, имеется дополнительный набор встроенных фрагментов, обеспечивающих создание окон, установки цвета, вызов редактора текста и др.

5. ОСОБЕННОСТИ ПРОГРАММИРОВАНИЯ И СИСТЕМЕ ДЕКЛАР
(Кузнецов Й. П.)

В данной главе будут рассмотрены вопросы, связанные с составлением правильных и эффективных программ на языках ДЕКЛ и РСС. Будут описаны дополнительные возможности, которые могут быть получены в рамках языка РСС - в плане металогического программирования, управления применением продукций др.

Будет рассмотрен еще один язык - СД, служащий для программирования сценариев диалога.

5.1. Правила составления программ на языке ДЕКЛ

Как уже говорилось, язык ДЕКЛ заметно отличается от языка ПРОЛОГ. В ДЕКЛ продукции (правила) имеют более общий вид, допускается применение одних продукций к другим. Есть возможность управления применением продукций, см. п. 3. В ДЕКЛ выбрана другая процедура вычисления. Каждая активизированная продукция применяется в различных вариантах. После этого становится возможным применение других продукций, которые, в частности, могут активизироваться текущей. Такое отличие порождает свои особенности в программировании. Остановимся на основных правилах, которые позволят правильно писать программы на ДЕКЛ.

Правило 1.

Каждая продукция, активизированная оператором T:<индикатор>, должна что-то изменять в знаниях, находящихся в ОП, иначе она будет применяться безостановочно.

Примечание. Сказанное не относится к продукциям, активизированным операторомы T!:<индикатор> и T1:<индикатор>.

Напомним, что изменение знаний в ОП может осуществляться:

- во-первых, за счет оператора удалить (D:), стоящего в левой части продукции,

- во-вторых, за счет фрагментов, находящихся в правой части продукции и не являющихся встроенными (они добавляются к ОП),

- в-третьих, за счет встроенных фрагментов "слияния" (UN), "ввода знаний" (IN) и "удаления" (DEL,C) и

- в-четвертых, за счет активизации других продукций, которые вызывают изменения.

То, или другое, или третье, или четвертое обязательно должно быть в любой продукции, активизированной с помощью T!.

Пример 5.1

```
ЕСЛИ UR1 R1(X1) ТО B:A(X1) ;
```

Недопустима активизация данной продукции оператором T:UR1. Для этого следует использовать оператор T!:UR1, не вызывающий безостановочных попыток применения. В последнем случае на экран будут выданы все аргументы отношения R1. Следует заметить, что и при активизации T:UR1 можно исключить безостановочное применение. Для этого в правую часть продукции следует ввести фрагмент,

играющий роль метки, или же можно перед фрагментом R1(...) поставить оператор "D:". Тогда вывод на экран будет сопровождаться удалением фрагментов.

Правило 2.

При вызове продукции оператором T!: происходит перебор вариантов, которые остаются при удалении фрагментов. Например, продукция

```
ЕСЛИ UR2 D:R1(X) R2(X) ТО...;
```

При ее вызове посредством T!:UR2 будет применяться столько раз, сколько фрагментов вида R2(...) имеется в ОП.

Правило 3.

При составлении продукций помните о возможности применения одних продукций к другим. За счет этого возможны побочные эффекты. В связи с этим требуется блокировка зон, в которых находятся продукции.

Напомним, что одна продукция будет применима к другой, если в последней есть фрагменты, соответствующие левой части первой продукции - ее (!) фрагментам.

Пример 5.2

```
TR1:ЕСЛИ UR1 R1(X1) ТО ... ;

TR2:ЕСЛИ UR1 R1(X1) R2(X1,X2) ТО ... ;

TR3:ЕСЛИ UR2 R3(X1) ТО R1(X1) R2(X1,X2) ;
```

Первая из этих продукций будет применяться ко второй (ее левой части) и третьей (ее правой части). Вторая продукция будет применяться к третьей (ее правой части). Такое применение, если оно не предусмотрено в рамках алгоритма, может вызвать порождение ненужной информации (фрагментов в ОП).

Чтобы одни продукции не применялись к другим, можно воспользоваться следующими приемами:

- заблокировать зону или разделы в ОП, в которых находятся продукции. Напомним, что это осуществляется с помощью спецфрагмента @BL(...,...). Продукции не будут применяться к фрагментам, находящимся в заблокированных зонах. Чтобы избежать случая применения продукций, показанного в предыдущем примере, следует оформить их как раздел знаний: в начале поставить метку

BEG(.../W1), а в конце - метку END(.../W2). Тогда введением в ОП спецфрагмента @BL(W1,W2) будет исключена возможность описанного применения продукций к продукциям.

- сделать в левой части каждой продукции один из фрагментов именованым и ввести в эту часть дополнительный фрагмент N:_(_,XJ), где XJ - имя фрагмента. Это будет исключать возможность применения одних продукции к другим.

Пример 5.3

ЕСЛИ UR1 R1(X1/X2) N:_(_,X2) ТО ... ;

Эта продукция не будет применима ни к одной из трех продукций примера 5.2. Попытки такого применения будут делаться, но все они будут безуспешными из-за фрагмента N:_(_,X2). Дело в том, что любая продукция, записанная на языке ДЕКЛ, вначале преобразуется на РСС (что осуществляется при ее считывании в ОП). Только после этого преобразования становится возможным ее применение. В то же время, при записи продукции на РСС имена всех ее фрагментов включаются в так называемые "сборки". Между тем, оператор отрицания N:_(_,X2) будет проверять, чтобы таких сборок не было и чтобы фрагмент R1(X1/...) не входил в какие-либо продукции. Пользуясь указанным приемом, можно избежать применения одних продукций к другим, расположенным в незаблокированных зонах ОП.

Правило 4.

Путем умелого расположения фрагментов в левой (и правой) части продукции можно значительно уменьшить объем работы, связанной с попытками применения продукций. Желательно, чтобы первыми стояли фрагменты с константами, редко встречающимися в ОП, то есть "уникальными". Вообще, порядок расположения фрагментов в продукции должен определяться степенью уникальности констант, частотой встречаемости их комбинаций, и т.п.

Правило 5.

Чем меньше объем разделов знаний, находящихся в ОП, тем быстрее будет осуществляться обработка, хотя зависимость эта не будет пропорциональной. Дело в том, что сети (РСС) представляются в ЭВМ в виде цепных списков - для каждой константы будет свой список. Чем больше раз встречается та или иная константа в ОП, тем длинней будет ее список и, следовательно, больше времени будет тратиться на работу (программное ядро постоянно пробегает такие списки). Итак, время обработки зависит от частоты встречаемости констант в ОП. Если два раздела знаний содержат непересекающиеся множества констант, то наличие в ОП одного раздела никак не будет влиять на время работы, осуществляемой в рамках другого раздела. Если разделы содержат одинаковые константы, то влияние будет иметь место.

Правило 5.

При очистке памяти встроенными фрагментами "DEL" и "C" следует предусмотреть, чтобы не была уничтожена нужная информация.

Особенность системы ДЕКЛАР состоит в том, что при применении продукций все вновь получаемые фрагменты помещаются

вслед за имеющимися. Такие фрагменты могут быть достаточно разнородными. Некоторые из них могут играть лишь вспомогательную, второстепенную роль. Такие фрагменты будут находиться "вперемешку" с другими. Поэтому возникает специальная задача - выделения "нужных" из них, которые, к примреру, должны быть оставлены в ОП и в дальнейшем запомнены на диске, или же использоваться при последующей обработке.

Правило 7.

Не допускается формирование новых фрагментов, имена которых совпадают с именами уже имеющихся в ОП констант.

Дело в том, что имя фрагмента есть его уникальная характеристика. Наличие двух фрагментов с одним именем не допустимо с точки зрения работы программного ядра. В связи со сказанным недопустимы продукции следующего вида:

пример 5.4

```
ЕСЛИ ... ТО ... R1(X1/DD) ... ;
```

На месте "DD" должна стоять переменная, не входящая в левую часть продукции.

Правило 8.

Не допускается наличие двух или нескольких продукций с одинаковыми именами. Напомним, что имен у продукций может и не быть. Тогда их роль будут играть внутрисистемные коды. Хотя, это может оказаться неудобным при отладке продукций.

Правило 9.

Следует осторожно пользоваться циклами и рекурсивными обращениями в процессе активизации продукций. Например, когда одна продукция активизировала другую, та - третью, а третья - первую. Например, нежелательно множество продукций вида:

```
ЕСЛИ W1 ... ТО ... T:W2 ... ;

ЕСЛИ W2 ... ТО ... T:W3 ... ;

ЕСЛИ W3 ... ТО ... T:W1 ... ;
```

При организации рекурсивного обращения следует знать максимальное количество вызовов.

```
ЕСЛИ W1(X) ... ТО ... T1:W1(X1) ... ;
```

Такого сорта обращения в общем случае допустимы. Однако, следует помнить, что в случае большого количества шагов рекурсии (если их более, чем 256) может переполняться стек, в котором запоминается "история" активизации продукций (стек необходим для выполнения возвратов).

Чтобы этого не было, процесс применения продукций следует организовывать таким же образом, как вызов модулей в структурном программировании: верхний модуль вызывает нижний, а тот,

отработав, передает управление вызвавшему его модулю. Подобный вызов реализуется операторами активизации. Процесс активизации должен иметь вид дерева.

В заключение следует отметить, что если при работе системы ДЕКЛАР произошло прекращение ее работы с выходом в монитор ОС, то основными причинами могут быть:

- обращение к файлу, которого нет на диске (имя этого файла выводится на экран);
- переполнение части ОП, отведенной под знания;
- переполнение стека вызовов, в котором хранится история активизации продукций.

Если система "повисла", то основной причиной может быть неправильная активизация продукций, см. Правила 1 и 9.

5.2. Язык сценариев диалога - СД

Этот язык предназначен для инженеров по знаниям. Он служит для программирования диалога - задания акций взаимодействия прикладных систем с пользователями. Такое взаимодействие можно изобразить в виде графа диалога, состоящего из вершин и меченых дуг.

Вершины соответствуют состоянию диалога. С каждой из них может быть связано сообщение, которое должно быть выдано пользователю при переходе в данное состояние. Дуги, исходящие из той или иной вершины, определяют переход. Они соответствуют возможным реакциям пользователя на выданное сообщение. Метки на дугах представляют характер таких реакций.

Например, это может быть выбор пункта меню, ввод слова, отдельного фрагмента и т.п., но что-нибудь одно. В зависимости от реакции пользователя осуществляется переход по той или иной дуге к следующей вершине или состоянию после чего оно становится "текущим состоянием". С дугами могут быть связаны определенные акции (процессы), выполняемые при упомянутом переходе, или же дефиниции, определяющие обращение к подзадачам.

Например, упомянутые акции могут сводиться к анализу введенного слова, вызова определенных разделов знаний в ОП и т.д. С помощью дефиниций передается управление самостоятельным задачам или подзадачам, решение которых осуществляется в рамках собственного подграфа диалога.

Описанный граф может быть представлен в виде сети (РСС), состоящей из фрагментов. Последние сопоставляются дугам графа, представляют связь вершин с выдаваемыми сообщениями, указывают на дефиниции и т.д. Подобные сети оформляются в виде разделов знаний, служащих для управления диалогом.

Примером такого раздела является "SDD", который определяет взаимодействие системы ДЕКЛАР с пользователями. В этом разделе есть свои продукции, обеспечивающие обработку пользовательских сообщений. Сами действия перехода из одного состояния в другое реализуются с помощью специального набора продукций, которые как бы постоянно осматривают граф диалога. Они играют роль интерпретатора. Эти продукции являются общими для всех сценариев

диалога. Они находятся в разделе SD.

Итак, синтаксис языка СД совпадает с РСС, а его конструкции - есть подмножество языка РСС, интерпретируемое специальным образом.

Перейдем к описанию основных средств языка СД.

Вершины графа диалога или состояния представлены на СД в виде констант. Отметим вначале, что текущему состоянию (вершине) сопоставляется фрагмент вида: АКТ(<константа>), который играет роль метки. Переход от одного состояния в другое, реализуемое с помощью продукций, сводится к перевешиванию таких меток.

Например, АКТ(TBL1) - означает, что текущее состояние есть TBL1. Переход в состояние TBL2 - осуществляется путем замены в ОП этого фрагмента на АКТ(TBL2). Эти метки "закрыты" для пользователя.

Состояние (константа), в которое осуществляется переход, определяет вызываемое сообщение. Последнее может задаваться (пользователем) двояким способом:

- оно может находиться в интерфейсном разделе, именем которого является упомянутое состояние (константа>. На это указывает фрагмент KTL(<константа>,<имя файла>);

- оно может иметь вид строковой константы. На это указывает фрагмент вида:

INF(<константа>,<строчечная константа>).

Например, INF(INF1,"укажите раздел знаний*>") означает, что при переходе в состояние INF1 на экран должно быть выдано приглашение:

укажите раздел знаний *>.

Наличие интерфейсного раздела TBL1 (например, в файле MENU1.Z) и фрагмента KTL(TBL1,MENU1) говорит о том, что при переходе в состояние TBL1 на экран должен быть выдан текст этого раздела.

Если для кокого-либо состояния, например INF2, отсутствует как фрагмент KTL (INF2,...), так и INF(INF2,...), то при переходе в данное состояние ничего не выдается на экран.

Дуги графа представляются на СД в виде фрагментов AFT(...). Приведем вначале несколько простых примеров:

AFT(TBL1,"1",-,TBL2) - означает, что если текущим состоянием было TBL1 (и уже выдано связанное с ним сообщение), то, когда пользователь нажмет клавишу 1, система должна перейти к состоянию TBL2. Дефис (знак "минус") на третьем аргументном месте указывает, что при этом не нужно выполнять каких либо дополнительных действий или акций.

AFT(INFK,%,INFK*,INF1) PASS(INFK*) - означает, что если текущим состоянием было INFK, то вне зависимости от реакции пользователя (на это указывает знак процента) нужно прейти в состояние INF1, но предварительно требуется применить продукции с индикатором INFK*. Фрагмент PASS(INF K*) означает, что активизация продукций должна осуществляться оператором T!:.

AFT(TBL4,%W,-,INF5) - означает, что после выдачи на экран сообщения, связанного с TBL4, пользователь должен ввести с клавиатуры слово или набор слов или набор фрагментов (на это указывает символ %W), после чего осуществляется переход в состояние INF5. При этом дополнительных действий не требуется. Если потребность в таких действиях есть, то на третьем аргументном месте вместо дефиса следует поставить индикатор каких-либо продукций. Тогда, перед тем как перейти в состояние INF5, будут делаться попытки применения этих продукций.

AFT(INF2,NEW,-,INF3) - означает, что после выдачи на экран сообщения, связанного с INF2, может быть осуществлен переход в состояние INF3. Такой переход осуществляется только в том случае, если в ОП имеется фрагмент GOTO(INF2,NEW). На месте NEW может стоять любая константа (кроме, см. приложение 3). Подобные фрагменты формируются своими продукциями, которые как бы подготавливают "программный" переход от INF2 к INF3. Таким способом реализуются так называемые переключатели. При этом от пользователя не требуется нажатия каких-либо клавиш.

AFT(TASK,"2",TASK2,INFT) DEF(TASK2)

Фрагмент DEF(TASK2) означает, что TASK2 является начальным состоянием самостоятельного подграфа диалога. Конечными состояниями такого подграфа является специальная константа FIN.

Предыдущий фрагмент AFT(...) означает, что после выдачи на экран сообщения, связаного с TASK, и нажатия пользователем клавиши "2" осуществляется переход к состоянию TA- SK2. После того, как встретилось состояние FIN, осуществляется обратный переход к состоянию, указанному на четвертом аргументном месте фрагмента AFT(...), в данном случае к INFT. При этом в ОП автоматически стираются все незащищенные фрагменты, порожденные продукциями, активизированными после перехода к TASK2 (осуществляется как бы "чистка" части оп). Если бы вместо DEF(TASK2) стояло бы DEF1(TASK2), то такого стирания не было бы. За счет такого стирания (оно осуществляется до специально установленной метки) реализуется принцип "магазинной памяти" или стековый механизм.

Дадим формальное описание фрагментов введенного вида:

AFT(<состояние1>,<реакция>,<указатель>,<состояние2>), где:

```
<состояние1>   -->  <константа>
<состояние2>   -->  <константа>
<реакция>      -->  "<цифра>" \ "<буква>" \
                    <константа> \ % \ %W
<указатель>    -->  - \ <индикатор> \ <константа>
```

Примечание. Символ "\" означает разделительное "ИЛИ".

Последняя константа должна быть отмечена DEF(<константа>) или DEF1(<константа>).

Итак, на первом и четвертом аргументных местах фрагмента AFT(...) стоят константы, определяющие, соответственно, текущее состояние и целевое состояние. Второе место определяет возможное поведение пользователя, его реакции. Это же место может служить для задания программных переходов - безусловного, условного и переключающего.

На втором месте может стоять:

- цифра или буква в кавычках. Она означает, что переход по данной дуге к следующему состоянию осуществляется лишь при нажатии пользователем соответствующей клавиши;

- процент (%). Он означает безусловный переход к следующему состоянию (указанному на четвертом месте) вне зависимости от реакции пользователя;

- символ %W (или %F). Они означают, что пользователь должен ввести с клавиатуры набор слов или фрагментов (они отделяются пробелами), после чего автоматически осуществляется переход в следующее состояние, указанное на четвертом месте. Вводимые слова и фрагменты отмечаются меткой WORD(...). Например, если введены слова AA DD, то в ОП будут сформированы фрагменты WORD(AA) и WORD(DD);

- константы, отличающиеся от дефиса, символов %W и символов (цифр) в кавычках. Такие константы означают программный переход, который осуществляется при наличии фрагментов вида GOTO(<состояние>,<константа>). Последние формируются с помощью продукций.

Третье аргументное место фрагмента AFT(...) указывает, что нужно сделать, прежде чем осуществить переход к следующему состоянию. На этом месте может находиться:

- дефис ("-"). Он означает, что переход в следующее состояние не связан с выполнением каких-либо дополнительных действий;

- индикатор продукций. Он означает, что прежде чем перейти в следующее состояние, нужно активизировать (попробовать) применить продукции с данным индикатором. При этом наличие PASS (<индикатор>) означает активизацию продукций с помощью оператора T!:<индикатор>, CIRC(<индикатор>) - с помощью T:<индикатор>, а отсутствие подобных фрагментов означает T1:<индикатор>;

- константа, отмеченная DEF(<константа>). Тогда управление из графа диалога передается подграфу с начальной вершиной - указанной константой. После того, как текущим состоянием оказалась константа FIN (что означает "конец подграфа"), осуществляется обратный переход - к состоянию, указанному на четвертом аргументном месте.

Пример 5.5

```
BEG(SDD)
      AFT(TBL1,"1",-,INF1) кTL(TBL1,MENU1)
      AFT(INF1,%W,INF1*,INF3)
      AFT(INF3,OLD,-,TBL3)
      AFT(INF3,NEW,-,TBL1)
      INF(INF1,"Укажите раздел знаний *>")
      INF(INF2,"Открыть новую секцию - Z или N >")
    BEG(/SDDB)
      SDD1:ЕСЛИ INF1* D:WORD(X) KTL(X,_)
           TO SECT(X) GOTO(INF3,OLD);
      SDD2:ЕСЛИ INF1* D:WORD(X) N:KTL(X,_)

 TO SECT(X) GOTO(INF3,NEW);

    END(/SDDE) @BL(SDDB,SDDE)
END(SDD)
```

Здесь поиллюстрирована часть графа диалога, представленного в разделе SDD. С помощью фрагмента @BL(...) осуществляется блокировка от самоприменения продукций SDD1 и SDD2. Граф задает следующие действия . Вначале из файла MENU1.Z на экран выдается интерфейсный раздел TBL1, то есть "главное меню". Если пользователь нажимает клавишу "1", то осуществляется переход в состояние INF1. На экран будет выдано сообщение:

Укажите раздел знаний *> .

Требуется, чтобы пользователь ввел с клавиатуры какое либо слово - на это указывает %W. К введенному слову будет привешена метка @@(...). После этого делается попытка применения продукций с индикатором INF1*. Отсутствие фрагментов PASS(INF1*) и CIRC(INF1*) означает активизацию продукций с помощью оператора T1:INF1*, т.е. до первого применения. Продукция SDD1 при ее активизации проводит анализ введенного слова в каталог е имеющихся разделов знаний - KTL(...,...). Если слово находится в каталоге, т.е. оно является именем какого-либо раздела знаний, то формируется

SECT(<слово>) и GOTO(INF3,OLD).

Таким способом подготавливается переход от INF3 в состояние TBL3, где начинается ввод новых знаний. Вторая продукция, т.е. SDD2, будет применимой в том случае, если слова нет в каталоге. Тогда формируются SECT(<слово>) и GO- TO(INF3,NEW). Слово будет воспринято как имя " нового" раздела знаний. Будет подготовлен переход от INF3 обратно к "главному меню" - TBL1.

В рамках языка СД реализована методика так называемой "динамической подкачки" знаниями. Ее суть заключается в следующем. При решении тех или иных задач требуются лишь определенные разделы знаний, которые должны быть вызваны в ОП. Другие разделы могут быть удалены из ОП. Соответственно, в графе диалога выделяются состояния, при переходе к которым выполняются указанные действия. В языке СД для этого используются фрагменты:

- TIE(<состояние>,<раздел>,ADD) - служит для задания разделов, которые должны быть вызваны (ADD) при переходе в указанное состояние;

- TIE(<состояние>,<раздел>,DEL) - задаются разделы, которые должны быть удалены (DEL) при переходе в указанное состояние.

Принцип динамической подкачки необходим при работе со знаниями большого объема, которые целиком не умещаются в выделенную для них зону ОП. Введенные вами средства позволяют осуществлять их вызов по мере необходимости.

В разделе SD имеются специальные продукции, которые осматривают фрагменты TIE(...) и выполняют задаваемые ими действия.

Пример 5.6

```
AFT(TBL4,"1",TASK1,TBL4) TBL(TBL4)
DEF(TASK1)
INF(TASK1,"Введите условие задачи *>")

AFT(TASK1,%F,UR1,FIN)
TIE(TASK1,GRFF,ADD)
TIE(TASK2,GRFP,ADD).
```

После выдачи на экран интерфейсного раздела TBL4 и нажатия пользователем клавиши "1" осуществляется переход к подзадаче TASK1. В ОП вызываются разделы знаний GRFF и GR- FP, осуществляющие решение задачи транзитивного замыкания на графе. На экран выдается приглашение:

Введите условия задачи *>.

После ввода условия активизируются продукции с индикатором UR1. Это вызовет применение продукций из раздела GR- FP с решением задач. По окончании осуществляется переход в состояние FIN, что вызовет удаление из ОП всех фрагментов, которые были порождены в процессе решения задачи, и "возврат" к состоянию TBL4. При этом сами разделы GRFF и GR- FP останутся в ОП.

5.3. Запись продукций на РСС

Как уже говорилось, продукции, записанные на языке ДЕКЛ, вначале преобразуются на язык низшего уровня - РСС, где уже делается возможным их применение. Продукции могут записываться прямо на языке РСС, хотя это менее удобно. Дело в том, что на языке РСС для выделения левой и правой частей используются два множества вспомогательных фрагментов, называемых сборками. В результате запись делается более однородной, но и более громоздкой. В РСС есть только фрагменты (нет никаких дополнительных синтаксических средств). Это позволило упростить программное ядро. В то же время громоздкость затрудняет запись продукций прямо на РСС - для этого требуются опредеделнные навыки.

Материал данного раздела необходим для "металогического" программирования, когда с помощью одних продукций формируются другие, которые тут же включаются в работу. Чтобы добиться этого, необходимо формировать запись продукций на языке РСС.

Рассмотрим особенности записи отдельных продукций на языке РСС.

Во-первых, в РСС левой и правой части продукции сопоставляются собственные константы. Для левой части такой константой является имя продукции, а для правой - это же имя, но с окончанием R. Пусть имя продукции есть FF. Тогда левой и правой частям продукции будет сопоставлено FF и FFR. Отметим, что если у продукции нет имени, то в качестве него берется внутрисистемный код. Он сопоставляется левой части продукции. Правой части будет сопоставлен другой код.

Во-вторых, вводится дополнительный фрагмент - P(FF,FFR), который связывает упомянутые константы. P - это имя так называемого "продукционного отношения", с помощью которого левая часть продукции отделяется от правой.

В-третьих, для каждой переменной, входящей в левую или правую части продукции, вводится дополнительный фрагмент, например если в продукции имеется переменная X1, то вводится - V(_,X1), где V - имя отношения "быть переменной".

В-четвертых, каждому простейшему фрагменту, входящему в левую или правую части продукции, присваивается свое имя. Им является внутрисистемный код. Для именованных фрагментов этого не делается. В результате каждый фрагмент будет иметь определенное имя. Сказанное относится и к индикаторам.

В-пятых, индикатор продукции (пусть это II*) связывается с константой, соответствующей левой части, с помощью фрагмента S(FF,II*), где S - имя отношения "часть-целое". Этот фрагмент вводится дополнительно.

Если индикатор содержит параметры, т.е. это фрагмент, то с упомянутой константой связывается имя фрагмента. Например, если индикатором является II*(X), то последнему присваивается имя II*(X/N1+) и формируется S(FF,N-). Последний ставится впереди всех других фрагментов, являющихся записью продукции.

В-шестых, для каждого фрагмента (пусть его имя есть QQ), входящего в левую часть продукции, вводится дополнительный фрагмент следующего вида:

- если перед фрагментом нет операторов, то вводится S(FF,QQ);
- если перед фрагментом стоит оператор D, то вводится D(FF,QQ);
- если перед фрагментом стоит оператор B, то вводится B(FF,QQ);
- если перед фрагментом стоит оператор N, то вводится N(FF,QQ).

Введенные выше дополнительные фрагменты (в том числе для индикаторов), сформированные на основе левой части, образуют множество, называемое сборкой левой части продукции. Такая сборка необходима для работы программного ядра, которое с ее помощью осуществляет выделение всех фрагментов левой части продукции.

В-седьмых, для каждого фрагмента (пусть его имя UU), входящего в правую часть продукции, вводится:

- если перед фрагментом нет операторов, то вводится фрагмент S(FF,UU);
- если перед фрагментом стоит оператор B, то вводится B(FF,UU).

В-восьмых, для каждого оператора T: активизации индикатора (пусть это T:JJ) вводится фрагмент T(FF,JJ).

Если индикатор - это фрагмент, то формируется имя фрагмента (пусть это N+) и вводится T(FF,N-). Например, если в правой части продукции FF есть оператор T:JJ*(X), то формируется JJ*(X/N+), а также TT(F,N-).

Все сказанное справедливо для операторов T1: и T!:. Только вместо т(...,...) будут формироваться соответственно T1(...,...) и T!(...,:..).

Введенные дополнительные фрагменты, сформированные на основе компонент правой части продукции, образуют множество, называемое сборкой правой части продукции. Такая сборка служит для выделения компонент и выполнения связанных с ними действий.

Итак, при записи продукции на РСС добавляются фрагменты сборки левой и правой части, а также фрагмент, соответствующий продукционному отношению. Порядок их записи следующий. Первым ставится фрагмент S(...), введенный на основе индикатора продукции. Далее следуют фрагменты левой части и их фрагменты "сборки" - S(...,...) или D(...,...), или B(...,...), или N(...,...). Далее следует фрагмент P(...). Он как-бы отделяет левую часть от правой. И наконец, следуют фрагменты правой части и сформированные на их основе фрагменты сборки. Если какой либо фрагмент левой или правой части содержит переменную, то формируется "указатель" этой переменной - фрагмент U(...,...).

Пример 5.7.

```
KZA: ЕСЛИ UR2 R1( _/X2) N: _( _, X2) ТО LL( _/X3) B: A( X2, X3);
```

Эта продукция записывается на РСС следующим образом:

```
S(KZA, UR2)
V( _, X2)
R1( _/1+)
S(KZA, 1-)
_( _, X2/2+)
N(KZA, 2-)
P(KZA, 3+)
V( _, X3)
LL(( _/4+)
S(3-, 4-)
A( X2, X3/5+)
B(3-, 5-)
```

Здесь 1+ и 1-, 2+ и 2- являются внутрисистемными именами,

которые присваиваются простейшим фрагментам N: _(_, X2) и C(X2, X3). Код 3+ сопоставляется правой части продукции.

Применение продукции осуществляется программным ядром, которое использует фрагменты сборки для выделения значи мых компонент. Вначале по индикатору (при его активизации) с помощью S(...,...) выделяется константа, соответствующая левой части продукции. Применительно к примеру 5.7 по UR2 с помощью S(KZA, UR2) будет выделено KZA. По ней с помощью V(...,...) выделяются переменные, входящие в левую часть продукции. Например, по V(_, X2) будет выделена переменная X2. Такие переменные образуют "стек означиваний".

Далее начинается последовательное выделение фрагментов левой части. Для этого используются фрагменты сборки. Они берутся в порядке их расположения. Если фрагмент выделен по S(...,...) или D(...,...), то для него ищется соответствующий фрагмент в ОП. Если он есть и содержит переменные, то заполняется стек означиваний, как это было описано в п. 3. 3.

Применение продукции продолжается. Если фрагмент выделен по B(...,...), то выполняется связанная с ним процедура, которая сама решает, закончить процесс применения продукции или нет. Если выделение фрагмента осуществлялось по N(...,...), то в ОП ищется соответствующий фрагмент. Но применение продукции продолжается только в том случае, когда такой фрагмент отсутствует в ОП.

В примере 5.7. Вначале по S(KZA, 1-) будет выделено имя фрагмента - 1-, а по имени - сам фрагмент R1(_/1+). Для него будет осуществляться поиск соответствующего фрагмента в ОП. Если он есть, то по N(KZA, 2-) будет выделено другое имя - 2- и фрагмент с таким именем - _(_, X2/2+). Для него будет проверяться отсутствие соответствующего фрагмента в ОП.

После того, как была отработана левая часть продукции, по фрагменту P(...,. ..) выделяется константа, сопоставленная правой части. И так же по S(...,... _), или B(...,...), или T(...,...) будут последовательно выделяться фрагменты правой части с выполнением соответствующих действий.

В примере 5.7. По P(KZA, 3+) будет выделена константа 3+. Далее, по S(3-, 4-) и B(3-, 5-) будут выделены фрагменты LL(_/4+) и A(X2, X3/5+). Для первого будет осуществляться поиск соответствующих фрагментов в ОП, а для второго - выполняться связанная с ним процедура.

6. РЕШЕНИЕ ПРИКЛАДНЫХ ЗАДАЧ

(Пузанов В. В., Золотарев О. В., Крюков В. Б.)

Система обработки знаний ДЕКЛАР может иметь широкое применение при решении задач логико-лингвистического характера в различных предметных областях. В этом разделе рассмотрено несколько прикладных задач разной степени сложности, иллюстрирующих работу системы ДЕКЛАР.

6.1. Поиск транзитивного замыкания на графе

Введем вначале обозначения. Символ ">" будет обозначть связку "принадлежать". Например, запись A > B означает, что "A принадлежит B". Перейдем к постановке задачи.

Пусть дан граф Q(VG,DG), где VG - множество вершин V(I) графа, V(I) > VG, I=1,2,...,N, N - число вершин графа, а DG - множество ориентированных дуг D(I,J), соединяющих вершины графа, D(I,J) > DG, I=1,...,N, J=1,...,N, I не равно J.

На этом графе выделяется пара V(I) и V(J) - начальная и конечная вершины. Требуется определить все пути, ведущие из начальной вершины в конечную, и выдать на печать информацию обо всех дугах, входящих в эти пути (хотя бы в какой либо один из них).

Если обозначить через L(K) к-ый путь, ведущий из вершины V(I) в вершину V(J), то множество путей LG, L(K) > LG образуют транзитивное замыкание между вершинами V(I) и V(J).

Такого типа задачи часто возникают при проектировании многослойных печатных плат, оптимальном управлении транспортными потоками и в ряде других областей.

Предположим, для конкретности, что исходный граф Q(VG,DG) состоит из 9 вершин, что изображено на рис.1.

На этом графе выделим начальную V1 и конечную V8 вершины. Топологию этого графа можно представить в виде следуюего набора фрагментов. Они помещаются в раздел знаний с именем GRFF:

```
DUG(V2,V5/D25)   DUG(V8,V8/D84)   DUG(V1,V3/D13)   DUG(V3,V4/D34)
DUG(V1,V5/D15)   DUG(V2,V3/D23)   DUG(V3,V6/D36)   DUG(V5,V7/D57)
DUG(V6,V8/D68)   DUG(V5,V4/D54)   DUG(V7,V9/D79)   DUG(V5,V6/D56)
DUG(V8,V9/D89)   DUG(V4,V9/D49)
```

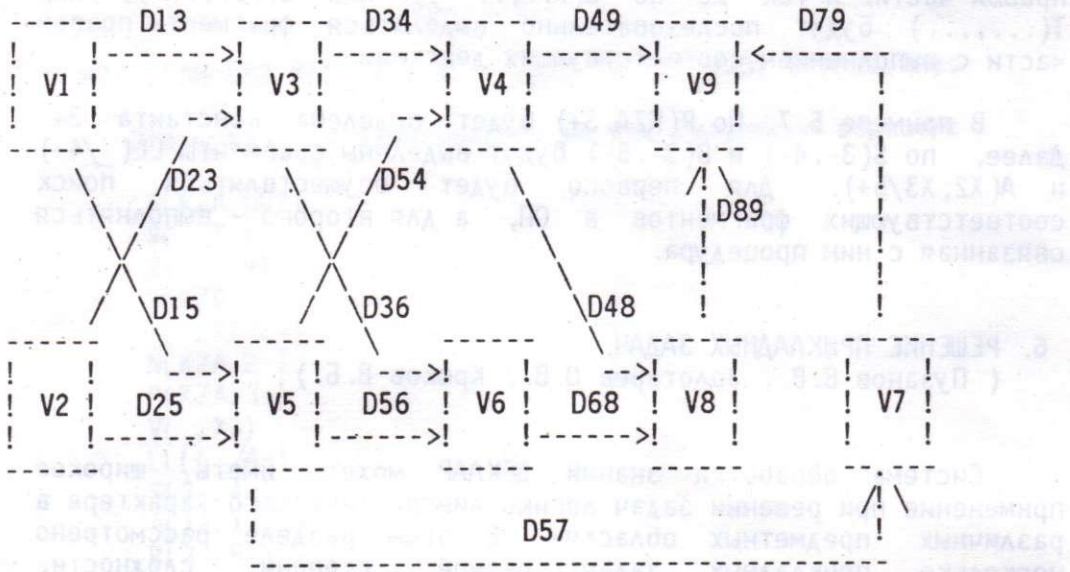

рис. 6.1

.....

Здесь каждый фрагмент имеет имя отношения DUG (связь по дуге). На первом аргументном месте стоит имя вершины, из которой исходит дуга, на втором - имя вершины, в которую входит дуга. На третьем аргументном месте после символа "/" (в соответствии с правилами записи на РСС) стоит имя фрагмента. Оно соответствует дуге. Такое представление графа Q(VG,DG) компактно, что существенно в тех случаях, когда размерность графа Q велика, а матрица инцидентности вершин редкая.

Взаимное расположение фрагментов типа DUG(...) в разделе GRFF не играет никакой роли. При изменении состава вершин и дуг граф легко модифицировать путем добавления или исключения фрагментов, что может осуществляться в произвольном порядке.

Будем решать эту задачу "волновым способом". Сущность этого способа состоит в следующем. Начальная и конечная вершины метятся специальными метками. Пусть это MET1 и MET2. Затем выделяется "окрестность" начальной вершины, т.е. выделяются все вершины, в которые входят дуги, исходящие из начальной вершины. Все выделенные вершины также помечаются меткой MET1, а связывающие дуги - меткой MD2. На следующем шаге только что выделенные вершины объявляются начальными. Теперь вешины, входящие в их окрестности, помечаются меткой MET1.

Распространяя такую "волну" слева направо по графу, помечаем весь граф. Аналогичную "волну", но уже справа налево, организуем из конечной вершины, помечая при этом соответствующие вершины и дуги меткой MET2 и MD2, соответственно. Очевидно, что те дуги, которые оказались помеченными сразу двумя метками MD1 и MD2 будут принадлежать множеству путей LG транзитивного замыкания.

Продукция, поддерживающая распространение "волны" слева направо (начиная от вершины, помеченной меткой MET1), имеет следующий вид:

```
MD1P: ЕСЛИ  UR1  MET1(X1)  DUG(X1,X2/X3)

      ТО  MET1(X2)  MD1(X3) ;
```

Пусть начальной и конечной являются вершины V1 и V8, т.е. созданы фрагменты MET(V1) и MET(V8). Так как в ОП имеется фрагмент MET1(V1), то применение данной продукции приведет к означиванию X1 именем вершины V1. Затем результат этого означивания будет использован при попытках означивания X2 во втором фрагменте, т.е. в DUG(V1,X2/X3). В результате многократных попыток поиска в ОП в разделе GRFF будет найдено два фрагмента, содержащих на первом месте вершину V1 - это DUG(V1,V3/D13), DUG(V1,V5/D15). Выбрав первый из них - DUG(V1,V3/D13), система присвоит X2 значение V3, а X3 - значение D13.

Результаты означивания переменных левой части продукции поступают в ее правую часть, на основе которой в ОП будет создано два новых фрагмента: MET1(V3) и MD1(D13). Применение этой же продукции к фрагменту DUG(V1,V5/D15), приведет к образованию еще двух новых фрагментов - MET1(V5) и MD1(D15). При появлении фрагментов - MET1(V3) или MET(V5), продукция MD1P как бы получит новую "пищу" и применится вновь и т.д. Пока за счет ее правой части будут создаваться все новые фрагменты.

Продукция, поддерживающая построение "волны" справа налево (начиная от вершины, помеченной меткой MET2) имеет следующий вид:

```
MD2P: ЕСЛИ UR1 MET2(X1) DUG(X2,X1/X3) ТО MET2(X2) MD2(X3) ;
```

Применяется она точно так же, как и продукция MD1P, только при ее применении будут создаватся фрагменты MET2(...) и MD2(...). Отметим также, что переменная X1 во фрагменте DUG(X2,X1/X3) в продукции MD2P стоит на втором месте, а в соответствующем фрагменте продукции MD1P - на первом. Это связано с ориентацией дуг графа Q. Продукция MD2P закончит применяться после того, как перестанут создаваться новые фрагменты.

Продукция, обеспечивающая выдачу на экран результатов решения, имеет следующий вид:

```
MD3P: ЕСЛИ UR2 MD1(X1) MD2(X1) ТО **(X1,X2)
BK B: A("DUG - ",X2);
```

Продукция ищет в ОП фрагменты MD1(...), созданные первой продукцией. В процессе применения второй продукции переменной X1 присваиваются определенные значения, например, V5. Тогда в ОП уже ищется фрагмент MD2(V5).

Если такой фрагмент найден, то продукция станет применимой, что приведет к выдаче имени дуги на экран.

Ранее было показано, что для успешной работы продукций MD1P и MD2P необходимо, чтобы были указаны начальная и конечная вершины, то есть в ОП были фрагменты MET1(...) и MET2(...). Это осуществляется следующей продукцией:

```
PAT: ЕСЛИ UR1 PATH(X1,X2) ТО MET(X1) MET(X2) ;
```

Фактически с помощью нее поддерживаются команды, например PATH(V1,V8), требующая найти транзитивное замыкание между вершинами V1 и V8. Для того, чтобы иметь возможность решать задачу автоматически в режиме меню, необходимо в таблицу TBL4 (из файла MENU1.Z) ввести номер пункта, соответствующго данной задаче, см. П.5.3. Кроме того, в раздел системных знаний TASK (из файла USER.Z) необходимо ввести фрагменты, обеспечивающие загрузку разделов знаний данной задачи. Пусть для задачи транзитивного замыкания в меню TBL4 выбран пункт "1":

```
.............
    $TABL4
    ------------------------------------------------------
                        задачи:

    - Поиск путей на графе                          (1)
    - Логический вывод нА графе "и-или"             (2)
    - Выбор структуры КТС (*)                       (3)
    - Диалоговая классификация деталей              (4)
    - Информация о содержащихся знаниях             (5)
    - Получение ответов на вопросы                  (6)
    ------------------------------------------------------
       *) КТС - комплекс технических средств
```

Для обращения к задаче в разделе TASK необходимо ввести следующий фрагмент:

```
    AFT(TBL4,"1",TASK1,TBL4) DEF(TASK1)
```

Эти фрагменты интерпретируются общесистемными продукциями, обеспечивающими переход от пункта TBL4 (если пользователь нажимает клавишу "1") к подзадаче или подпроцессу TASK1, а затем - возврат к таблице TBL4. Подзадача TASK1 задается пятью следующими фрагментами:

```
    DEF(TASK1)
    TIE(TASK1,GRFF,ADD) KTL(TASK1, MENU1)
    TIE(TASK1,GRFP,ADD)
    AFT(TASK1,%F,UR1,TASK1Q)   CIRC(UR1)
    AFT(TASK1Q,%,UR2,FIN)      CIRC(UR2)
```

Процесс решения начинается с выдачи из файла MENU1.Z на экран таблицы TASK1 на что указывают фрагменты TIE(...). Таблица имеет следующий вид:

```
------------------------------------------------------
 TASK1
      условие задачи излагается в форме:
      PATH(<вершина1>,<вершина2>),
 где: <вершина1> - начало пути,
      <вершина2> - конец пути,
 например:  **>PATH(V1,V8)
 GRFF - граф,
 GRFP - правила поиска путей.
      Вводите условие задачи **>
------------------------------------------------------
```

Затем в ОП подгружаются (на это указывает ADD) разделы знаний GRFF и GRFP. Фрагменты AFT(...) вызовут считывание из таблицы TASK1 строки символов (%F) и передачу управления продукциям с индикаторами UR1 и UR2. Служебные фрагменты CIRC(UR1) и CIRC(UR2) вызовут циклическое применение продукций с этими индикаторами, т.е. до тех пор, пока среди них есть применимые.

Общесистемные продукции, интерпретирующие фрагменты TIE(...), обеспечивают подкачку необходимых разделов знаний только в том случае, если имена разделов каталогизированы.

.....

Поэтому для каждого раздела в систему должны быть введены фрагменты вида KTL(...,...). В них на первом аргументном месте стоит имя раздела, а на втором - имя файла, в котором этот фрагмент хранится. В данном случае это будут:

KTL(GRFF,PROD)
KTL(GRFP,PROD)

Таким образом, для решения задачи выбора путей на графе требуется:

1. Создать раздел базы знаний (GRFF) и ввести в него исходные данные в виде набора фрагментов (топологию графа Q).

2. Создать раздел базы знаний (GRFP) и ввести в него продукции, поддерживающие решение задачи.

3. Дополнить сеть, представляющую сценарий диалога фрагментами, реализующими диалог и вызывающими своевременный вызов с диска необходимых разделов знаний, см. п. 5.2.

4. Ввести в каталог (KTL.Z) фрагменты, указывающие на месторасположение используемых в задаче разделов знаний.

6.2. Вывод на "И-ИЛИ" графе

Для широкого круга применений (сапр, диагностика заболеваний, неисправностей оборудования и т.п.) перспективным представляется использование принципов логического вывода, а также иерархического пространства состояний, заданного в виде графа "и-или". Пример такого графа показан на рис. 2.

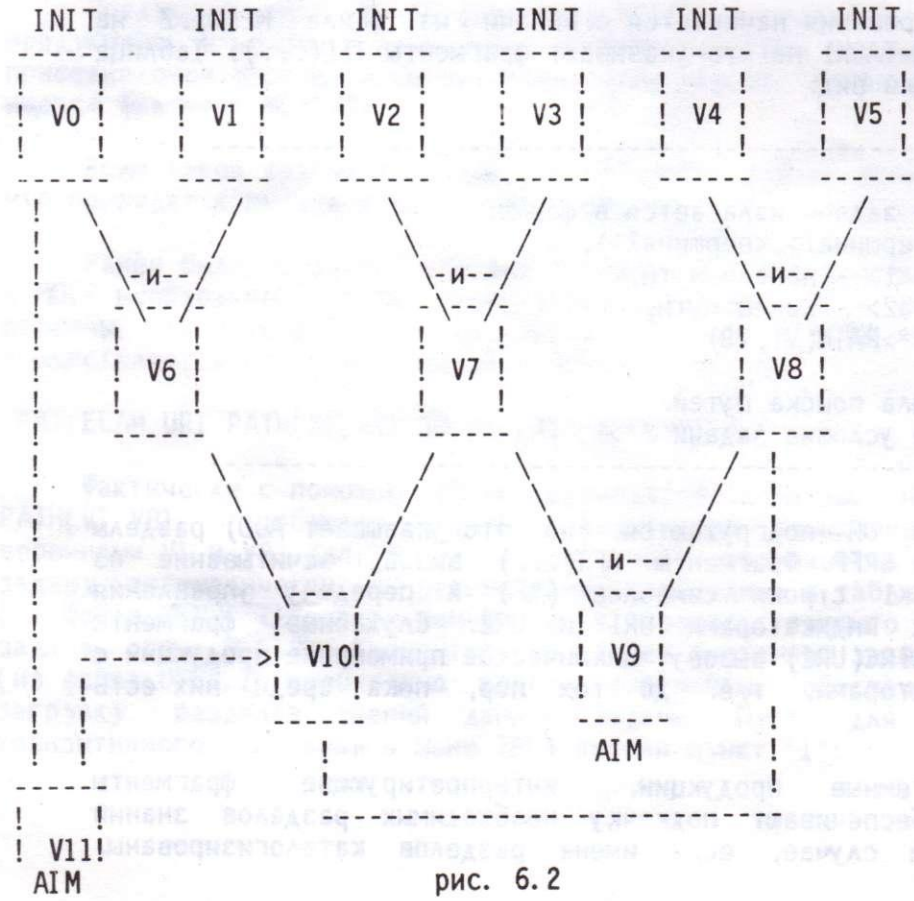

рис. 6.2

В этом графе верхние ("исходные") вершины соответствуют исходным фактам, сообщаемым пользователем. На рис. 2 они помечены термином "INIT". Вершины других уровней графа соответствуют посылкам, которые вытекают из этих фактов. Самые нижние вершины графа, помеченные термином "AIM", соответствуют целевым состояниям или конечным посылкам. Между вершинами графа могут существовать связки как типа "И" (на рис. 2 они отмечены как "--и--"), так и типа "ИЛИ". Логический вывод сводится к передвижению меток "истинности" от одних вершин графа к другим.

В общем случае задача вывода может носить вероятностный характер. С каждой входной вершиной связывается вероятность появления соответствующего факта. В процессе вывода (перехода из состояния в состояние) осуществляется подсчет вероятностей, например, на основе аппарата нечетких множеств или по формуле Байеса, см. П. 6. 5.

Средства решения задачи вводятся в систему ДЕКЛАР так же, как для задачи п. 6.1. Рассмотрим принципы ее представления. Пусть дан граф Q(VG, DG), где VG - множество вершин V(I) графа, V(I) > VG, I=1, 2, ..., N, N - число вершин графа, а DG - множество ориентированных дуг D(I, J), соединяющих вершины графа, D(I, J) > DG, I=1, 2, ..., N, J=1, 2, ..., N, I не равно J. Граф, изображенный на рис. 2, записывается на языке РСС в виде следующего набора фрагментов:

```
$(GR1)
BEG(GR1)
DAND(V0, V6)  DAND(V1, V6)  DOR(V2, V7)  DOR(V3, V7)
DAND(V4, V8)  DAND(V5, V8)  DAND(V8, V9)  DAND(V6, V9)
DAND(V7, V10)  DAND(V8, V10)  DOR(V10, V11)  DOR(V0, V11)
DOR(V10, V11)  DOR(V0, V11)
INIT(V0)  INIT(V1)  INIT(V2)  INIT(V3)  INIT(V4)  INIT(V5)
AIM(V9)  AIM(V11)
END(GR1)
```

Из рассмотрения рис. 6. 2 видно, что в процессе логического вывода состояние V7 может быть объявлено истинным, если истинны состояния V2 или V3, а состояние V6 - если истинны состояния V0 и V1. Фрагменты DOR и DAND, связывающие вершины графа Q, представляют логические связки типа "ИЛИ" и "И". Верхние вершины помечены Метками "INIT", а целевые вершины - метками "AIM".

Вначале в режиме меню-анкет (который реализуется своими средствами) осуществляется ввод исходных фактов. В результате верхние вершины отмечаются специальной меткой истинности - TRUE(...). Логический вывод сводится к продвижению меток истинности по ветвям дерева (графа Q) до тех пор, пока не будет достигнута одна из целевых вершин. Это осуществляется следующими продукциями:

```
UL1: ЕСЛИ US1 TRUE(X1) DOR(X1, X2) ТО TRUE(X2) ;
```

.....

```
UL2: ЕСЛИ US1 TRUE(X1) DAND(X1,X2) N:[DAND(X3,X2) N: TRUE(X3)]
  ТО TRUE(X2) ;

UL3: ЕСЛИ US2 AIM(X1) TRUE(X1) N: MT(X1) ТО
     MT(X1) T1: TERZ(X1) ;
```

Продукция UL1 обеспечивает продвижение метки TRUE по дуге типа "ИЛИ". Если, например, в ОП после ввода исходных фактов оказался фрагмент TRUE(V2), то при применении продукции UL1 в ОП будет осуществляться поиск фрагментов типа DOR, у которых на первом аргументном месте стоит вершина V2. При нахождении такого фрагмента происходит означивание переменной X2. В частности, таким фрагментом является DOR(V2,V7). Поэтому X2 означится вершиной V7. Результат этого означивания передается в правую часть. Будет сформирован фрагмент TRUE(V7).

Продукция UL2 обеспечивает продвижение метки TRUE(...) по дуге "И". В левую часть входит конструкция N:[DAND (X3,X2) N: TRUE (X3)], Которая читается так: "Нет такой дуги, ведущей из вершины X3 в вершину X2, у которой не было бы метки TRUE". Иначе говоря, в вершину X2 должны входить только дуги, исходящие из вершин, отмеченных меткой истинности.

Продукция UL3 проверяет для целевой вершины (помеченной меткой AIM), есть ли на ней метка TRUE. В результате на эту вершину ставится метка MT(...). Это делается только в том случае, если такой метки ранее не было. Для этого в левую часть введено отрицание N: MT(X1). На каждую вершину может быть поставлено не более одной метки MT(...). Далее активируется системная продукция с индикатором TERZ(X1), обеспечивающая вывод на экран файла с именем X1. В этот файл можно поместить текст или графический образ целевого состояния.

В процессе вывода несколько целевых вершин могут получить метки TRUE. Если в этом случае необходимо выдать сообщение "обнаружено множество решений", то используется следующая продукция:

```
UL16: ЕСЛИ US2 AIM(X) AIM(X1) B: #(X,X1) D: TRUE(X) D: TRUE(X1)
ТО ВК B: A("Обнаружено множество решений");
```

Эта продукция применима только в случае, когда имеются две целевые вершины, помеченные меткой TRUE. Эти метки TRUE удаляются, а за счет правой части продукции осуществляется вывод на экран с новой строки (ВК) сообщения, текст которого записан в кавычках.

Пусть в процессе вывода ни одна целевая вершина не получит метку TRUE. Тогда необходимо вывести сообщение "Решение не найдено". Это делается следующей продукцией:

```
UL17: ЕСЛИ US2 N:[AIM(X1) TRUE(X)] ТО ВК B: A("Решения не
найдено")
```

Продукции, осуществляющие ввод в систему исходных фактов от пользователя и объяснение полученных результатов будут рассмотрены в п. 6. 3.

6.3. Выбор комплекса технических средств АСУ (КТС)

Эта задача предметно ориентпрована. Однако, методы ее решения могут быть использованы и в других предметных областях. Основу решения составляет граф "И-ИЛИ", представляющий связь исходных данных, полученных от пользователя, с мендуемым ему комплексом средств АСУ. Используется граф, описанный в п. 6.2.

Поэтому для логического вывода здесь служат те же самые продукции (UL1 - UL3). В связи с этим, далее будем рассматривать только средства ввода исходных данных.

Для решения задачи в режиме меню, необходимо, так же как и в задаче п. 6.1, дополнить соответствующие разделы знаний и таблицы, что осуществляется с помощью экранного редактора.

Закрепим за задачей пункт 3 меню TABL4, тогда в раздел системных знаний должен быть введен следующий фрагмент:

```
AFT(TABL4,"3",-,TABL5)
```

Этот фрагмент обеспечивает переход к состоянию TABL5 (без применения каких-либо продукций) с выдачей на экран слудующей таблицы:

```
Выбор КТС (прямой)      - (1)
Выбор КТС (обратный)    - (2)
Выбор КТС (взвешенный)  - (3)
```

Эти пункты связаны с задачами прямого, обратного или взвешенного (вероятностного) выбора КТС.

Чтобы при нажатии клавиши была вызвана своя задача в соответствующий раздел (им является EXSR из файла USER.Z) вводятся фрагменты:

```
AFT(TABL5,"1",TASK3,TABL4)
AFT(TABL5,"2",TASK8,TABL4)
AFT(TABL5,"3",TASK9,TABL4)
```

Рассмотрим решение задачи прямого выбора КТС.

Первый из фрагментов AFT(...) означает, что при нажатии клавиши "1" необходимо от таблицы TABL5 перейти к подзадаче TASK3, после чего вернуться к TABL4 (в исходное меню).

С этой подзадачей связаны следующие фрагменты:

```
DEF(TASK3)
TIE(TASK3,EXPL,ADD)
TIE(TASK3,GR2,ADD)
TIE(TASK3,UL,ADD)
TIE(TASK3,KTC,ADD)
```

Они указывают на необходимость ввода разделов знаний GR2 (граф "И-ИЛИ"), EXPL (объясняющие знания), UL (средства логического вывода, см. П.6.2), КТС (анкеты и средства ввода исходных данных).

.....

Для выдачи меню-анкет и ввода исходных даных в ОП слу жат фрагменты AFT(...). На экран вызываются альтернативы, из которых пользователь должен сделать выбор. Рассмотрим пример:

```
AFT(TASK3,%,-,KTC1)
AFT(KTC1,"1",TRUE1*,KTC2)
AFT(KTC1,"2",TRUE1*,KTC2)
AFT(KTC2,"1",TRUE1*,KTC3)
AFT(KTC2,"2",TRUE1*,KTC3)
AFT(KTC3,"1",TRUE1*,KTC4)
AFT(KTC3,"2",TRUE1*,KTC4)

AFT(KTC4,"1",TRUE1*,KTCE)
AFT(KTC4,"2",TRUE1*,KTCE)
AFT(KTCE,%,US1,0+) CIRC(US1)
AFT(0-,%,USZ,EXP3)
```

Второй и третий фрагменты AFT(...), в частности, приведут к выдаче на экран таблицы (KTC1) вида:

```
Сколько абонентов в системе?
    меньше 40     -   (1)
    больше 40     -   (2)
```

Пользователь выбирает подходящий ему пункт ("1" или "2"). Далее активизируется продукция с индикатором TRUE1*, которая устанавливает метку истинности (TRUE) на соответствующей вершине графа "И-ИЛИ" (см. Ниже). Связь выбора, осуществляемого пользователем с вершиной, на которую вешается метка TRUE(...), представляется с помощью фрагментов:

```
TLINK(KTC1,"1",V1)
TLINK(KTC1,"2",V3)
TLINK(KTC2,"1",V0)
TLINK(KTC2,"2",V2)
TLINK(KTC3,"1",VV)
TLINK(KTC3,"2",V5)
TLINK(KTC4,"1",VV)
TLINK(KTC4,"2",V4)
```

Эти и вышеупомянутые фрагменты анализируются уже упоминавшейся продукцией:

```
KTCP1: ЕСЛИ TRUE1* @PR(X) D:I(X1) TLINK(X,X1,X2)
    ТО TRUE(X2);
```

В частности, если выбран пункт "2", то с помощью B:I(X1) переменная X1 будет означена "2", а X3 - V3. Продукция станет применимой, метка истинности TRUE будет установлена на вершину V3.

Аналогично, в зависимости от ответа пользователя, осуществляется установка меток и на другие вершины графа "И-ИЛИ".

Далее осуществляется само решение задачи (выбора), которое сводится к логическому выводу на графе "И-ИЛИ".

.....

6.4. Реализация компоненты объяснений

Для объяснения результатов, полученных после логического вывода, используется идея "обратного" прохода от вершин к "исходным", т.е. соответствующим исходным посылкам (данным) и последующего объясняющего прохода. Вначале метка MT(...), которой был отмечен результат, передвигается снизу вверх (по вершинам, отмеченным TRUE). Таким образом выделяются пути , по которым должно идти объяснение. Для этого служат продукции:

```
UL4: ЕСЛИ VV1 MT(X2) DAND(X1,X2) TO MT(X1) ;

UL5: ЕСЛИ VV1 MT(X2) DOR(X1,X2) TRUE(X1) TO MT(X1) ;
```

В результате их применения каждая вершина "И-ИЛИ" графа, принявшая участие в достижении цели, получает "объясняющую" метку - MT(...). Сами объяснения хранятся в специальных фрагментах (связанных с вершинами "И-ИЛИ" графа) и имеют вид строковых констант. Объяснения выдаются в процессе еще одного прохода ("объясняющего") - сверху вниз, но уже по меткам MT(...). По каждой вершине, выделяемой в процессе такого прохода, выдаются тексты "посылок" и объяснений. Например, при переходе к вершине VI по дуге типа DAND выдаются:

- тексты (посылок), связанные с вершинами (VJ), от которых идут дуги DAND к VI. Связь задается фрагментом FACT(VJ,"<текст>");

- текст объяснения, который связан с VI с помощью фрагмента EXPL(VI,"<текст>");

- текст результата. Для его выделения используется фрагмент FACT(VI,"<текст>").

Такая выдача управляется следующей продукцией:

```
UL7: ЕСЛИ VV2 TRUE(X1) INIT(X1) MT(X1) DAND(X1,X2) MT(X2)
  TO BK B:A("Я знаю ответ") MAND(X2) T:VVV2 INIT(X2)
  B:A("Нажмите "ВК"") B:G(_) ;
```

Продукция устанавливает метки MAND на те вершины, на основе которых необходимо выдать объяснения, и передает управление следующей продукции (с индикатором VVV2):

```
UL8: ЕСЛИ VVV2 MAND(X2) DAND(X1,X2) D:MT(X1) FACT(X1,X3)
  TO BK B:A(X3) ;
```

Эта продукция "осматривает" вершины, от которых идут дуги DAND к исходной вершине (X2) и выдает на экран связанные с ними тексты посылок. Текст объяснения выдается продукцией:

```
    UL9: ЕСЛИ VVV2 D:MAND(X1) EXPL(X1,X3) FACT(X1,X4)
  TO BK B:A(" Мной применено правило:",X3)
  B:A(" В результате получено",X4);
```

.....

Аналогичные тексты объяснений, но для вершин типа "ИЛИ", выдаются другой продукцией:

```
UL6: ЕСЛИ VV2 TRUE(X1) D: MT(X1) INIT(X1)  DOR(X1,X2)  MT(X2)
  FACT(X1,X3) EXPL(X2,X4)  FACT(X2,X5) TO BK
  B: A("я знаю, что:",X3) B: A(" мной применено правило:",X4)
  B: A("в  результате получено:",X5) INIT(X2)
  B: A("нажмите ВК") B: G();
```

Эта продукция (как и UL7) применяется только к вершинам, отмеченным MT. Рассмотрим пример ее применения. Пусть при означивании переменная X1 приняла значение V1, а X2 - значение V6. Пусть в ОП имеются три фрагмента:

```
FACT(V1,"число абонентов более 40,")
EXPL(V4,"Если число абонентов в системе более 40 и они уда-
лены на расстояния более 200 м...")
FACT(V6,"требуется ЭВМ телеобработки абонентов").
```

Тогда в результате применения продукции UL6 на экране появится следующий объясняющий текст:

"Я знаю, что:

число абонентов в системе более 40, мной применено правило:

Если число абонентов в системе более 40 и они удалены на расстояния более 200 м...

В результате получено:

требуется ЭВМ телеобработки абонентов"

Последовательность объяснений определяется метками INIT, которые "перевешиваются" продукциями UL7 и UL6 в процессе "объясняющего" прохода - сверху-вниз (движение осуществляется только через вершины, отмеченные меткой MT). При этом метки MT для вершин, уже получивших объяснения, уничтожаются.

6.5. Вывод с подсчетом вероятностей

Ниже будет рассматриваться задача "взвешенного" вывода на И-ИЛИ графе. Она сводится к перевешиванию меток ("истинного" знания), с которыми связывается вес. Соответственно, при перевешивании веса пересчитываются. Такая задача возникает в случае когда исходные данные заданы в неопределенном виде. Например, пользователь сомневается в той или иной посылке (факте). Такое сомнение выражается в виде весовых коэффициентов. Ответом является не ДА-НЕТ, а некоторое целое число от 0 до 100. Продукция, поддерживающая ввод данных и логический вывод, имеет следующий вид:

```
KTVP1: ЕСЛИ VES1*@PR(X) D: WORD(X1) TUNK(X,X2) TO VES(X2,X1);
```

В отличие от аналогичной продукции KTCP1, данная продукция за счет общесистемных продукций считывает слово (X1) вероятности исходного события и для соответствующей вершины создает вместо

одноместного фрагмента TRUE двухместный фрагмент "VES", у которого на первом аргументном месте стоит имя вершины, а на втором - ее "вес", полученный из фрагмента "WORD". Соответственно, продукции логического вывода имеют вид:

```
KUL1: ЕСЛИ US1 VES(X1,X3) DOR(X1,X2) N:[DOR(X4,X2)
      VES(X4,X5) B:>(X5,X3)] TO VES(X2,X3);

KUL2: ЕСЛИ  US1  VES(X1,X3)  DAND(X1,X2)  N:[DAND(X6,X2)
N:VES(X6,_)]  N:[DAND(X4,X2)  VES(X4,X5)  B:<(X5,X3)]  TO
VES(X2,X3);
```

Рассмотрение этих продукций показывает, что продвижение "весов", полученных при вводе, осуществляется по правилам теории нечетной логики. Если исходные факты образуют результирующую вершину типа "ИЛИ", то этой результирующей вершине присваивается вес, равный максимальному значению вероятности исходной вершины. Если исходные факты образуют результирующую вершину типа "И", то этой вершине присваивается вес, равный минимальному значению вероятности исходной вершины. Для этого в продукциях используются встроенные фрагменты реализации арифметических операций ">" и "<".

Необходимо заметить, что в данном случае в процессе вывода принципиально возможно получение множества решений, имеющих разные вероятностные веса. В связи с этим, продукция, обеспечивающая вывод на экран результатов достижения цели, также меняется и будет иметь следующий вид:

```
KUL3: ЕСЛИ  US2  AIM(X1)  D:VES(X1,X2) N:[AIM(X3) VES(X3,X4)
B:>(X4,X2)] TO BK B:A("вероятность=",X2) T1:TERZ(X1) ;
```

Следует заметить, что если система не устанавливает значения веса целевой вершины, то, аналогично тому, как это рассматривалось в задаче 2, выдается сообщение "Нет решений". Соответствующая продукция KUL17 практически не отличается от UL17.

Модификацией задачи 3 на основе логического вывода является "обратный вывод" по графу "И-ИЛИ". В этом случае пользователь указывает известную ему цель, а система продукций ищет набор тех верхних входных вершин графа, которые обеспечивают достижение этой цели (см. рис. 2). В этом случае, как и ранее, пользователь с помощью меню-анкеты выбирает цель. С каждой целью связан определенный пункт меню.

Работа меню поддерживается фрагментами типа "AFT", содержащимися в соответствующем разделе базы знаний:

```
AFT(TASK9,%,-,CEL1)
AFT(CEL1,"1",CL*,CELE)
AFT(CEL1,"2",CL*,CELE)
AFT(CELE,%,CL1*,O+)  CIRC(CL1*)
AFT(O-,%,CL2*,O+)  CIRC(CL2*)
AFT(O-,%,CL2*,EXP3)
```

В связи с тем, что целевые вершины могут являться как сборками типа "И", так и сборками типа "ИЛИ", то для

формирования исходных весов целевых вершин используются две продукции:

```
CELP1: ЕСЛИ CL* @PR(X) B:I(X1) TLINC(X,X1,X2) DAND(X3,X2)
AIM(X2) TO VES(X2,1) ;

CELP2: ЕСЛИ CL* @PR(X) B:I(X1) TLINC(X,X1,X2) DOR(X3,X2)
AIM(X2) TO VES(X2,2) ;
```

Из рассмотрения работы этих продукций видно, что в ОП вводится фрагмент VES(X2,1), если целевая вершина является сборкой типа "И", и VES(X2,2), если целевая вершина является сборкой типа "ИЛИ". В обоих случаях данные о номере вершины извлекаются из фрагментов TLINK вида:

```
TLINK(CEL1,"1",V9)
TLINK(CEL1,"2",V11)
```

Например, при выборе пользователем пункта "1" меню встроенный фрагмент B:I(X1) осуществляет считывание "1" с клавиатуры, а фрагмент TLINK осуществляет выбор целевой вершины V9.

Продвижение метки VES по дугам графа "И-ИЛИ" осуществляется с помощью продукций:

```
CELP3: ЕСЛИ CL1* DAND(X2,X1) VES(X1,X3) TO VES(X2,X3) ;
CELP4: ЕСЛИ CL1* DOR(X2,X1) VES(X1,X3) TO VES(X2,2) ;
```

В данных продукциях всем вершинам, являющимся сборками типа "ИЛИ", принудительно присваивается весовое значение "2". Следовательно, все вершины, которые будут пройдены после текущей, также получают вес "2". Это делается для того, чтобы в ходе логического вывода иметь возможность при достижении верхних терминальных вершин выделить вершины, обязательно участвующие в формировании цели, и вершины, об участии которых в формировании цели можно говорить только с той или иной степенью вероятности. Следует иметь в виду, что в данном случае мы не вводили вероятностную метрику графа И-ИЛИ. Выдачу на печать результатов решения задачи можно осуществить с помощью следующих продукций:

```
CELP5: ЕСЛИ CL2* INIT(X1) D:VES(X1,X3) B:=(X3,1) TO
B:A("Вершина И",X1) ;

CELP6: ЕСЛИ CL2* INIT(X1) D:VES(X1,X3) B:=(X3,2) TO
B:A("NODE OR",X1) ;
```

Представленные примеры решения конкретных задач могут быть использованы как основа реализации более сложных задач логической обработки информации. Для этого в разделы базы знаний на диске необходимо ввести нужные изменения и дополнения, обеспечивающие глобальную логику решения поставленной задачи. При этом большинство описанных выше продукционных средств изменений не потребуют. Более того, указанные средства при их умелом применении позволят значительно сократить объем необходимой собственной разработки.

Прилохение 1.

Встроенные фрагменты

Перед встроенными фрагментами в продукциях ставится оператор B:, например: IF B:I(X) THEN B:A(X) B:G();. при обращении к встроенному фрагменту выполняется соответствующая "встроенная функция". Результат может быть успешным ("истина") или безуспешным ("лохь"). При этом могут осуществяться означивания переменных. При безуспешном выполнении встроенного фрагмента, стоящего в левой части продукции, продукция обьявляется неприменимой. Безуспешное выполнение встроенного фрагмента, стоящего в правой части продукции, вызывает окончание процесса ее применения.

Для описания встроенных фрагментов будут использоваться следующие обозначения:

Ki - константы или переменные, означенные константами;
Ці - целые числа или переменные, означенные этими числами;
_ - символ "_" (подчеркивание) или переменные, означенные этим символом;
Xi - неозначенные переменные;
Пi - переменные, означенные или неозначенные;
Ci - символы или переменные, означенные этими символами;
Ai - переменные (означенные или неозначенные),константы,символ "_";

FNAME - строковые константы - имена файлов;
RAS - строковые константы - расширения файлов;
RNAME - строковые константы - имена разделов.

Номер, имя встроенного фрагмента	Допустимые варианты означиваний аргументов встроенного фрагмента	Результат: успешный(истина), безуспешный(лохь)	Выполняемые функции: означивание пер., изменение сем.сети и т.д.	Название и краткое описание функционирования встроенного фрагмента
1	2	3	4	5
1 = равенство	=(K1,K1)	истина		
	=(_,_)	истина		
	=(K1,K2)	лохь		
	=(_,K1)	лохь		
	=(K1,_)	лохь		
	=(X1,X2)	лохь		
	=(K1,X1)	истина	означ.X1	Переменная X1 будет означена
	=(X1,K1)	--//--	---//---	константой K1.
	=(_,X1)	истина	означ.X1	Переменная X1 будет означена
	=(X1,_)	--//--	---//---	символом "_" (подчеркиван.)

1	2	3	4	5
2				
#	#(K1,K1)	ложь		
	#(_,_)	ложь		
неравен-	#(X1,X2)	ложь		
ство	#(K1,K2)	истина		
	#(_,K1)	истина		
	#(K1,_)	истина		
	#(K1,X1)	истина		
	#(X1,K1)	истина		
	#(_,X1)	истина		
	#(X1,_)	истина		
3				
>	>(Ц1,Ц2)	истина,		Результат будет "истина",
		ложь(и/л)		если Ц1 больше Ц2,
больше				где Ц1 и Ц2 целые числа.
	в остальн.	ложь		
	случаях			
4				
>=	>=(Ц1,Ц2)	и/л		Результат будет "истина",
				если Ц1 больше или равно Ц2,
больше				где Ц1 и Ц2 целые числа.
или	в остальн.	ложь		
равно	случаях			
5				
<	<(Ц1,Ц2)	и/л		Результат будет "истина",
				если Ц1 меньше Ц2,
меньше				где Ц1 и Ц2 целые числа.
	в остальн.	ложь		
	случаях			
6				
<=	<=(Ц1,Ц2)	и/л		Результат будет "истина",
				если Ц1 меньше или равно Ц2,
меньше				где Ц1 и Ц2 целые числа.
или	в остальн.	ложь		
равно	случаях			
7				
+	+(Ц1,Ц2,П3)	истина	означива-	Переменная П3 получает зна-
			ется	чение Ц1+Ц2 (независимо от
слохить			(означ.)П3	своего прехнего значения).
	в остальн.	ложь		
	случаях			
8				
*	*(Ц1,Ц2,П3)	истина	означ.П3	Переменная П3 получает зна-
				чение Ц1*Ц2.
умно-				
жить	в остальн.	ложь		
	случаях			

1	2	3	4	5
9				
M-	M-(Ц1,Ц2,П3)	истина	означ.П3	Переменная П3 получает значение Ц1-Ц2.
вычесть	в остальн. случаях	ложь		
10				
DIV	DIV(Ц1,Ц2,П3)	истина	означ.П3	Переменная П3 получает значение Ц1/Ц2.
целочисленное деление	в остальн. случаях	ложь		
11				
+%	+%(Ц1,Ц2,П3)	истина	означ.П3	Переменная П3 получает значение Ц1+Ц2-((Ц1*Ц2)/100).
сложение процентов	в остальн. случаях	ложь		
12		всегда		
SELF	SELF(П1,П2)	истина	означ.П2	Переменная П2 означивается именем переменной П1 из продукции.
означивание именем переменной				
13		всегда	управляет	
PAR		истина	трассир. и т.п.	
управление режимом работы	PAR("1","0")			Отключает трассировку.
	PAR("1","2")			Включает простую трассиров.
	PAR("1","5")			Включает полную трассировку.
	PAR("3","0")			Отключает разметку фрагментов при работе IN().
	PAR("3","5")			Включает разметку фрагментов при работе IN().
	PAR("4","0")			Включает блокировку.
	PAR("4","5")			Отменяет блокировку.
14		всегда	означ.П2	Переменная П2 получает значение равное количеству аргументных мест во фрагменте К1 с именем К1.
KARG	KARG(К1,П2)	истина		

1	2	3	4	5
15 FR выделение компонент	FR(X1,X2,X3)	ложь и/л		Выделение фрагментов, содержащих К3. Х1 означается именем фрагмента, а Х2 - номером позиции, где расположена К3.
	FR(X1,Ц2,X3)	ложь и/л		Выделение фрагментов, у которых в позиции Ц2 находится К3.
	FR(K1,X2,X3)			Выделение компонент фрагмента с именем К1. Х2 означается номером позиции, а Х3 - что на нем стоит.
	FR(K1,X2,K3)			Поиск во фрагменте К1 позиции, на которой стоит К3.
	FR(K1,Ц2,X3)			Выделение в К1 аргумента, стоящего на позиции Ц2.
	FR(K1,Ц2,K3)	и/л		Проверка того, что во фрагменте К1 в позиции Ц2 стоит К3.
16 SET	SET(K1,Ц2,K3)	всегда истина	изменяет в ОП сем.сеть	Во фрагменте с именем К1 изменяет аргумент в позиции с номером Ц2 и делает его равным К3.
17 UN слияние окрестностей	UN(K1,K2)	всегда истина	изменяет в ОП сем.сеть	Заменяет во всех фрагментах, находящихся в ОП, К2 на К1.
18 NP	NP()	всегда истина	изменяет в ОП сем.сеть	Добавляет в сем.сеть фрагмент @@(FFF,@@), где FFF - имя последней примененной продукции.
19 FRI	FRI(П1)	всегда истина	означ.П1	Присваивает переменной П1 значение,равное размеру свободной памяти для сем.сети.
20 NEW	NEW(П1)	всегда истина	изменяет в ОП сем.сеть, означ.П1	Порождает новую вершину в сем.сети и переменной П1 присваивает ее код.
21 A вывод на экран (или диск)	A(A1) A(A1,A2) A(A1,A2,A3) A(A1,A2,A3,A4) A(A1,A2,A3,A4,A5)	всегда истина	вывод на диск или на экран	На экран (или диск) выводятся A1,A2,A3,A4,A5. Например, это могут быть строковые константы (они распечатаются без двойных кавычек).

1	2	3	4	5
22 F выдача меню	F(FNAME,RNAME)	всегда истина	вывод на экран	Выводит на экран содержимое раздела RNAME из текстового файла FNAME.Z.
23 OUT вывод из ОП указанной зоны	OUT(FNAME,K2,K3)	всегда истина	вывод на диск и на экран	Вывод из ОП зоны, т.е. всего что находится между фрагментами с именами K2 и K3, в файл FNAME.Z.
	OUT(FNAME,K2 K3,1)			Вывод из ОП зоны (от фрагмента с именем K2 до K3, не включая их) в файл FNAME.Z.
	OUT(_.K2,K3)			Вывод из ОП зоны (от K2 до K3) на экран или в текущий файл, открытый с помощью REW(...).
	OUT(_,K2,K3,1)			Вывод из ОП зоны (от фрагмента с именем K2 до K3, не включая их) на экран или в текущий фэйл, открытый с помощью REW(...).
24 OUT1	OUT1(FNAME)	всегда истина	вывод на диск и	Выводит из ОП на диск в фаил FNAME.ZNN всю сем. сеть в закодированной форме.
25 REW управление выводом	REW()	всегда истина		Настраивает фрагменты вывода (OUT, A) на экран.
	REW(FNAME,RAS)			Настраивает фрагменты вывода (OUT, A) на запись в файл FNAME.RAS. старый файл с тем же именем - уничтожается.
	REW(FNAME,RAS,STR)			Настраивает фрагменты вывода (OUT, A) на запись в файл FNAME.RAS. выведенная информация записывается после контекстной строкой STR. если такой строки нет, то - запись в конец файла.
26 TT управление выводом	TT(TRUE)	всегда истина		Настраивает фрагменты вывода (oUT, A) на экран.
	TT(FALSE)			Если вывод был переключен на экран с диска с помощью TT(TRUE), то переключает его обратно на диск.

1	2	3	4	5
27 EOLN проверка конца строки	EOLN()	истина, ложь (и/л)		Результат будет "истина", когда головка считывания находится на конце строки.
28 G сдвиг головки считывания	G()	всегда истина	изменяет состояние буфера вх.текста	Устанавливает головку считывания на следующий символ в буфере входного текста (при вводе с клавиатуры или из файла).
29 LN чистка буфера входного текста до конца строки	LN()	всегда истина	изменяет состояние буфера вх.текста	Удаляет до конца строки все символы из буфера входного текста, расположенные правее головки считывания. Оставляет головку считывания на прежней позиции.
30 ввод одного символа (с клавиатуры или из файла)	I(X1)	истина	означивается (означ) X1	X1 означивается символом, на который указывает головка считывания.
	I(C1)	истина, ложь (и/л)		Результат будет "истина", когда головка указывает на символ C1, и "ложь" в противном случае.
31 IN ввод в ОП выражений, записанных на декл	IN()	всегда истина	изменяет в ОП сем.сеть	Ввод с клавиатуры или из файла фрагментов, продукций и отдельных констант Ki. Они "метятся" с помощью фрагментов вида: @@(Ki).
	IN(FNAME)			Ввод из файла FNAME.Z всех фрагментов и продукций, не оформленных в виде раздела знаний.
	IN(FNAME,RAZD)			Ввод раздела RAZD из файла FNAME.Z .
32 R управление вводом текста	R(FNAME)	всегда истина		Настраивает фрагменты ввода IN, G на файл FNAME.Z

1	2	3	4	5
33 C удале- ние фраг- ментов	C(K1)	всегда истина	изменяет в ОП сем.сеть	Удаление из ОП всех фрагментов, записанных после фрагмента с именем K1 (вместе с ним).
	C(K1,H)			Удаление из ОП всех фрагментов, записанных после K1, кроме фрагментов вида: ...(.../KK) H(KK), где KK - имя фрагмента.
34 DEL удале- ние фраг- ментов	DEL(K1,K2)	всегда истина	изменяет в ОП сем.сеть	Удаляет в ОП все те фрагменты от K1 до K2 (зону), у которых имена не входят во внешние для этой зоны фрагменты.
	DEL(K1,K2,H)			Удаляет в ОП все фрагменты от K1 до K2, кроме защищенных фрагментов (имеющих вид: ...(.../KK) H(KK), где KK - имя фрагмента), и тех фрагментов, у которых имена не входят во внешние для этой зоны фрагменты.
	DEL(K1)			Удаляет фрагмент K1.
35 DELFR	DELFR(K1)	всегда истина	изменяет в ОП сем.сеть	Удаляет из ОП все незаблокированные фрагменты сем.сети, имеющие имя отношения K1.
36 DELH	DELH(K1)	всегда истина	изменяет в ОП сем.сеть	Удаление из ОП всех фрагментов, имеющих имя отношения H и записанных после фрагмента с именем K1.
37 SQU	SQU(K1)	всегда истина	изменяет в ОП сем.сеть	Выполняет сжатие сем.сети (удаляет в ней пустоты) от фрагмента с именем K1 и до конца.
38 CALL	CALL(FNAME,RAS)	истина		Вызывает редактор текста LEXICON. (IBM PC)
39 EDIT	EDIT(FNAME,RAS)	истина		Вызывает редактор текста LEXICON. (IBM PC)
40 EXIT	EXIT()	всегда истина		Останов программы. (IBM PC)
41 COLD	COLD(Ц1)	всегда истина		Устанавливает цвет фона. (IBM PC)

1	2	3	4	5
42 COLT	COLT(Ц1)	всегда истина		Устанавливает цвет текста. (IBM PC)
43 WIND	WIND(Ц1,Ц2,Ц3,Ц4)	истина		Устанавливает размеры окна. (IBM PC)
44 OLDW	OLDW()	истина		Устанавливает размеры предыдущего окна. (IBM PC)
45 OKR	OKR(K1,H)	всегда истина	вывод на экран	Выводит на экран окрестность K1 (т.е. все фрагменты, содержащие константу K1).
	OKR(K1)			Выводит на экран все незаблокированные фрагменты, содержащие константу K1.
46 LIPS	LIPS()	всегда истина	вывод на экран	Выдает на экран информацию о количестве логических выводов. выполненных от момента пуска программы.

Прилоxение 2.

Стандартные процедуры (продукции)

Стандартные процедуры записываются в виде продукций. такие продукции будут называться стандартными. Обращение к ним происходит так xе как к обычным продукциям - с помощью оператора T1:, за которым ставится индикатор стандартной продукции, например, T1:INZ(GRFF). Для описания стандартных процедур будут использоваться следующие обозначения:

RNAME - имена разделов: строковые константы или переменные, означенные такими константами; каxдый такой раздел оформляется соответствующим образом и описывается в каталоге в виде - KTL(RNAME,FNAME), где FNAME - имя файла (без расширения), содержащего раздел RNAME;

SCEEN - имена подсценариев диалога: строковые константы или переменные, означенные такими константами;

N - целые числа или переменные, означенные этими числами.

Имя стандартной процедуры	Индикатор соответствующей стандартной продукции	В каком разделе знаний описана	Краткое описание стандартной процедуры
1	2	3	4
ввод раздела в ОП	INZ(RNAME)	SYS	Если раздела RNAME нет в ОП и он описан в каталоге, то он вводится в ОП. Если же его нет ни в ОП ни в каталоге, то выдается сообщение: "Раздел не может быть введен - отсутствует в каталоге /KTL/".
ввод указанных разделов в ОП	INSEC*	SD	Запрашиваются имена разделов и затем эти разделы вводятся с помощью стандартной процедуры ввода раздела.
удаление раздела из ОП	DELZ(RNAME)	SYS	Если раздела RNAME имеется в ОП, то он удаляется оттуда. В противном случае ничего не делается.
вывод раздела в файл	OUTZN(RNAME)	SYS	Если раздела RNAME есть в ОП, то он выводится в файл RNAME.Z. В противном случае выдается сообщение: "Раздел не может быть выведен на диск".
вывод раздела на экран	TERZ(RNAME)	SYS	Если раздела RNAME описан в каталоге, то из соответствующего файла на экран выдается содержимое этого раздела. В противном случае содержимое этого раздела выдается из ОП. Если же раздела RNAME нет ни в каталоге ни в ОП, то выдается сообщение: "Раздел отсутствует в KTL".
вывод (из ОП) раздела на экран	TEROP(RNAME)	SYS	Из ОП на экран выдается раздел RNAME.
распечатка каталога	LISEC*	SD	На экран выдается список разделов, описанных в каталоге, с указанием, в каких файлах они находятся.
вставка в раздел	VSTZ(RNAME1,RNAME2)	SYS	Раздел RNAME2 - удаляется, а его содержимое вставляется в конец раздела RNAME1. Это происходит одновременно и в ОП и на диске.
вызов подсценария	CALL1(SCEEN)	SD	Осуществляет переход на подсценарий диалога SCEEN, который был описан на языке сд через DEF(SCEEN) или DEF1(SCEEN).

1	2	3	4
установка цвета	COL(N1,N2)	SYS	Устанавливается цвет фона N1 и текста N2. (IBM PC)
возврат цвета	CLR*	SYS	Устанавливается цвет фона N1 и текста N2, которые задаются для текущего подсценария диалога SCEEN - с помощью фрагмента CLR(SCEEN,N1,N2). Если такой фрагмент отсутствует, то устанавливается цвет: черные буквы на белом фоне. (IBM PC)
центральное окно	WIND1(RNAME)	SYS	Выдает содержимое раздела RNAME в окно, расположенное в центре экрана. Размер окна - 50(букв по горизонтали) * 6(строк). (IBM PC)
правое окно	WIND2(RNAME)	SYS	Выдает содержимое раздела RNAME в окно, расположенное в правом верхнем углу экрана. Размер окна - 48 * 11. (IBM PC)

Прилохение 3.

Системные имена

1. Раздел PROD

Продукционные сборки:

P - продукционное отношение,
S - обычная сборка,
D - удаление,
N - отрицание,
B - выполнить встроенную функцию,
T - циклический вызов продукций,
T1 - однократный вызов продукций,
T! - однопроходный вызов продукций.

Спец.имена, по которым нельзя вести поиск фрагментов (используемые программным ядром):

V - метка переменных,
Z - имя-значение,
L - соединение букв,
H - защита,
_ - подчеркивание,
_D - удаленный аргумент.

Спец.имена, используемые программным ядром:

```
K              - корень,
@BL            - блокировка,
@@             - метка ввода с клавиатуры,
если ESLI IF   - выделение условной части продукций,
то TO THEN     - выделение следствия,
SYS            - имя системного раздела,
PROD           - файл, в котором находится системный раздел.
```

Имена, используемые в откомпилированных продукциях:

```
SLOV MD LR OGR OGR1 OGR2 DEBO ZAPR ZAPR1 ZAPR2 LSB
COP FLX ок сл % MOGR
```

Имена встроенных фрагментов:

```
= # > >= < <= + * M- DIV +% SELF PAR KARG FR SET UN NP FRI NEW
A  F  OUT OUT1 REW TT EOLN G  LN I  IN R  C  DEL DELFR DELH SQU
CALL EDIT  EXIT  COLD  COLT  WIND OLDW
OKR LIPS RET MORFOL*
```

2. Раздел KTL (содержит имена файлов и разделов)

```
MENU1 : TBLO TBL1  PRIG1 KOMH SYPR RSS EXPP DEBO TASS TASK2
        TBL5 ISS1 SEL
PROD  : SYS SD SDD QA
HELP  : HELIS TASK1H TASK2H EXP3H ANS-QH
USER  : DEB TASK TASK2 EXPS GRFP GRFF DIAP DIAF GR11 UL KOMAN
KTC   : KTLKTC EXP1H EXP1HH EXP1AH EXP1BH V9 V11 EXPL
CL    : CLPR CLGR QAA
ISS   :
MON   : MREG MLIB
LIN   : LINTH SLIN LIN
LIB   :
AR    :
LL    : ASK NATL SYNT NEWS ASKW PRAG
LIF   :
OVERL :
COMP  :
NTO   :
NT    : HELNT
AD    : &ID DES &HEL &HEL3 &D &DD &A &C &HEL &HELC
CLASS : PER&
PER   :
```

3. Раздел SYS

Использует имена, начинающиеся символом "@".

Управление трассировкой и др.:

```
TR0  - отключение трассировки,
TR1  - простая трассировка,
TR2  - полная трассировка,
BK   - переход на новую строку при печати,
RET  - возврат параметров.
```

Имена, используемые в простейшем обратном лингвистическом процессоре: / вызывается с помощью T1:OLI*(X) /

OLIT - метка приоритетных имен отношений,
OLIN - метка запрещенных к выдаче имен отношений,
NAME - связь с полным именем.

4. Раздел SD

Использует имена, начинающиеся символом "@", а также следующие:

AFT - сценарий диалога,
%W - запрос имени,
%F - запрос фрагмента,
GOTO - безусловный переход,
HELP - выдача подсказки,
WORD - метка ввода,
CIRC - циклическая активизация,
PASS - однопроходная активизация,
TIE - динамическая подкачка,
CLR - управление цветом,
WINDOW - выдача окна,
FIN - конец подсценария,
INF - выдача сообщения,
DEF - определение подсценария,
DEF1 - подсценарий (без стирания).