ГОСУДАРСТВЕННЫЙ КОМИТЕТ СВЯЗИ, ИНФОРМАТИЗАЦИИ И ТЕЛЕКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

НУКУССКИЙ ФИЛИАЛ ТАШКЕНТСКОГО УНИВЕРСИТЕТА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра информационных технологий

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Жалгасбаева Алпамыса студента 4-го курса факультета компьютерный инжиниринг по направлению информационный сервис

ТЕМА: Средства для создания экспертных систем на языке Prolog

Научный руководитель:_____доц.Бурханов Ш.А.

Зав. кафедрой:_____доц. Арзымбетов Т.З.

НУКУС - 2014 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Сегодня всем, кто работает в области информатики или интересуется этой областью науки, известен термин "экспертные системы". Экспертная система (ЭС) (expert system, knowledge based system) - это программная система, знания и умения которой сравнимы с умением и знаниями специалистов в какой-нибудь специальной области знаний. ЭС вместе с системами обработки естественных языков являются наиболее важными в коммерческом плане областями использования искусственного интеллекта.

В рамках исследования искусственного интеллекта созданы многочисленные ЭС для разных областей знания, таких, например, как медицинская диагностика и обследование пациентов, генные и молекулярные исследования, составление конфигурации вычислительных машин, образование, поиск неисправностей в устройствах и системах и многие другие практические приложения.

Экспертные системы должны решать задачи, требующие для своего решения экспертных знаний в некоторой конкретной области. В той или иной форме экспертные системы должны обладать этими знаниями. Поэтому их также называют *системами, основанными на знаниях*. Однако не всякую систему, основанную на знаниях, можно рассматривать как экспертную. Экспертная система должна также уметь каким-то образом *объяснять* свое поведение и свои решения пользователю, так же, как это делает эксперт-человек. Это особенно необходимо в областях, для которых характерна неопределенность, неточность информации (например, в медицинской диагностике). В этих случаях способность к объяснению нужна для того, чтобы повысить степень доверия пользователя к советам системы, а также для того, чтобы дать возможность пользователю обнаружить возможный дефект в рассуждениях системы. В связи с этим в экспертных системах следует предусматривать дружественное взаимодействие с пользователем, которое делает для пользователя процесс рассуждения системы "прозрачным".

ЭС не говорит что лучше, она предоставляет те варианты, критерии которых устраивают пользователя в большей мере. Главным достоинством ЭС является возможность накопления знаний и сохранение их длительное время. В отличие от человека к любой информации ЭС подходят объективно, что улучшает качество проводимой экспертизы.

В настоящее время проблема разработки ЭС отражены в работах Джозефа Джарратано и Гари Райли «Экспертные системы: принципы разработки и программирование», Таунсенда К., Фохта Д. «Проектирование и программная реализация экспертных систем на персональных ЭВМ». Изучением различных аспектов данной проблемы занимались Питер Джексон «Введение в экспертные системы», Уотермен Д. «Руководство по экспертным системам».

В данной выпускной квалификационной работе рассматриваются программные средства создания экспертных систем на языке Пролог.

Задачи исследования:

1. Изучить и проанализировать научную литературу по проблеме исследования;
2. Изучить понятие ЭС и ее структуру;
3. Изучить технологию проектирования ЭС;
4. Спроектировать и реализовать ЭС на языке программирования PROLOG.

Работа состоит из введения, 4-х параграфов, заключения и литературы.

Во введении излагается актуальность и важность выбранной тематики исследовании.

В первом параграфе приводятся сведения по основным понятиям и классификациям экспертных систем.

Второй параграф посвящен этапам разработки экспертных систем.

Третий параграф посвящен основные элементам языка Турбо-Пролог и приемам работы в данной среде программирования.

В четвертом параграфе рассматриваются вопросы проектирования и реализации экспертной системы «Определение неисправности при запуске компьютера» в среде Турбо-Пролог.

В заключении кратком виде приведены полученные результаты и их значения.

В приложении приведена текст программы экспертной системы.

§1. Основные понятия и классификация экспертных систем

Понятие и свойства экспертных систем

В информатике ЭС рассматриваются совместно с базами знаний как модели поведения экспертов в определенной области знаний с использованием процедур логического вывода и принятия решений, а базы знаний – как совокупность фактов и правил логического вывода в выбранной предметной области деятельности.

Знания являются явными и доступными, что отличает ЭС от традиционных программ, и определяет их основные свойства такие как:

1. Решения проблем высококачественного опыта, который представляет уровень мышления наиболее квалифицированных экспертов в данной области, что ведет к решениям творческим, точным и эффективным.

2. Наличие прогностических возможностей, при которых ЭС выдает ответы не только для конкретной ситуации, но и показывает, как изменяются эти ответы в новых ситуациях, с возможностью подробного объяснения каким образом новая ситуация привела к изменениям.

3. Обеспечение такого нового качества, как институциональная память, за счет входящей в состав ЭС базы знаний, которая разработана в ходе взаимодействий со специалистами организации, и представляет собой текущую политику этой группы людей. Этот набор знаний становится сводом квалифицированных мнений и постоянно обновляемым справочником наилучших стратегий и методов, используемых персоналом.

4. Возможность использования ЭС для обучения и тренировки руководящих работников, обеспечивая новых служащих.

Для формирования полноценной ЭС необходимо, как правило, реализовать в ней следующие функции.

- функции решения задач, позволяющие использовать специальные знания в проблемной области (при этом может потребоваться обеспечить работу в условиях неопределенности).

- функции взаимодействия с пользователем, которые, в частности, позволяют объяснить намерения и выводы системы в процессе решения задачи и по завершении этого процесса.

Каждая из этих функций может оказаться очень сложной, а способ их реализации может зависеть от проблемной области и практических требований. К тому же разработка и реализация проекта такой системы часто требует решения разнообразных и сложных проблем. К ним относится выбор способа представления знаний и соответствующих средств проведения рассуждений.

Классификация, структура и задачи экспертных систем

Реальные ЭС в настоящий момент представляют собой один из наиболее сложных вариантов программных комплексов. Поэтому они имеют множество характеристик и их классификация достаточно сложна. К тому же ЭС – достаточно молодая отрасль науки, поэтому в ней нет общепринятых критериев классификации. Существуют классификации по областям применения и назначению, по используемым методам и глубине анализа предметной области, по классу систем и инструментальной реализации и т.д. [5]. Для создания ЭС определяющую роль играет такая характеристика как форма представления знаний в ЭС. Согласно этому фактору ЭС можно разделить на следующие типы:

- системы, базирующиеся на правилах;
- системы, базирующиеся на логике;
- системы, базирующиеся на фреймах;
- системы, базирующиеся на моделях.

Последние два типа ЭС имеют достаточно сложную организацию, анализ которой выходит за рамки работы. Системы на фреймах использует представление знаний, основанное на логических группах атрибутов объектов, которые описываются в виде фреймов. В системах, базирующихся на моделях, необходимо знать структуру и принципы функционирования

объекта исследования. Такие системы нашли широкое применение в ЭС проектирования сложных технических объектов.

Системы, базирующиеся на правилах, наиболее популярны из-за своей простоты и наглядности. В этих системах вся информация хранится в виде правил Пролога. Второй тип ЭС – системы, базирующиеся на логике, - наиболее точно отражают структуру Пролога. В подобных системах основу базы знаний составляют факты (предложения-утверждения).

Для классификации ЭС можно использовать различные критерии.

1. По *назначению* ЭС можно условно разделить на консультационные (информационные), исследовательские и управляющие. Консультационные ЭС предназначены для получения квалифицированных ответов; исследовательские – для помощи пользователю квалифицированно решать научные задачи; управляющие – для автоматизации управления процессами в реальном масштабе времени.

2. По *сложности и объему базы знаний* –неглубокие и глубокие. Неглубокие (простые) ЭС имеют относительно малые БЗ. Доказательства их заключений обычно коротки, большинство выводов являются прямыми следствиями информации, хранимой в базе знаний. Такие ЭС в основном предназначены для решения относительно простых задач типа ответов на запросы по требуемой информации.

Глубокие ЭС делают свои выводы обязательно из моделей происходящих процессов, хранящихся в базах знаний. Сама модель процесса представляет собой набор правил, предназначенных для объяснения большого количества эмпирических данных. В глубоких ЭС доказательства выводов значительно длиннее, основываются на знаниях, выведенных из моделей.

3. По *области применения* ЭС делятся следующие классы.

1) *Диагностика.*Например, медицинская диагностика, когда системы используются для установления заболеваний; техническая диагностика,

когда определяют неисправности в механических и электрических устройствах.

2) *Прогнозирование.* Прогнозирующие системы предсказывают возможные результаты или события на основе данных о текущем состоянии объекта (погода, урожайность, поток пассажиров).

3) *Планирование и проектирование.* Такие системы предназначены для достижения конкретных целей при решении задач с большим числом переменных (консультации по приобретению товаров, проектирование космических станций, и так далее).

4) *Интерпретация.* Интерпретирующие системы обладают способностью получать определенные заключения на основе результатов наблюдения (например, местоположение и тип судов в океане по данным акустических систем слежения).

5) *Контроль и управление* (например, регулирование финансовой деятельности предприятия и оказание помощи при выработке решений в критических ситуациях, управление воздушным движением, атомными электростанциями).

6) *Обучение.* Экспертно-обучающие системы реализуют следующие педагогические функции: учение, обучение, контроль и диагностику знаний, тренировку.

4. По *связям с реальным миром.*

1) Статические ЭС разрабатываются в предметных областях, в которых БЗ и интерпретируемые данные не меняются во времени, они стабильны. Например, диагностика неисправностей в автомобиле.

2) Квазидинамические ЭС интерпретируют ситуацию, которая меняется с некоторым фиксированным интервалом времени. Например, микробиологические ЭС, в которых снимаются лабораторные изменения с технологического процесса один раз в 4 -5 часов и анализируется динамика полученных показателей по отношению к предыдущему измерению.

3) Динамические ЭС работают в сопряжении с датчиками объектов в режиме реального времени с непрерывной интерпретацией поступающих в систему данных. Например, управление гибкими производственными комплексами, мониторинг в реанимационных палатах.

Можно выделить четыре основных класса ЭС: классифицирующие, доопределяющие, трансформирующие и мультиагентные.

1) *Классифицирующие* ЭС решают задачи распознавания ситуаций. Основным методом формирования решений в таких системах является дедуктивный логический вывод.

2) *Доопределяющие* ЭС используются для решения задач с не полностью определенными данными и знаниями. В таких ЭС возникают задачи интерпретации нечетких знаний и выбора альтернативных направлений поиска в пространстве возможных решений. В качестве методов обработки неопределенных знаний могут использоваться байесовский вероятностный подход, коэффициенты уверенности, нечеткая логика.

3) *Трансформирующие* относятся к синтезирующим динамическим ЭС, в которых предполагается повторяющееся преобразование знаний в процессе решения задач. В ЭС данного класса используются различные способы обработки знаний:

- генерация и проверка гипотез;
- логика предположений и умолчаний (когда по неполным данным формируются представления об объектах определенного класса, которые впоследствии адаптируются к конкретным условиям изменяющихся ситуаций);
- использование метазнаний (более общих закономерностей) для устранения неопределенностей в ситуациях.

1) *Мулътиагентные* системы – это динамические ЭС, основанные на интеграции нескольких разнородных источников знаний. Эти источники обмениваются между собой получаемыми результатами в ходе решения задач. Системы данного класса имеют следующие возможности:

реализация альтернативных рассуждений на основе использования различных источников знаний и механизма устранения противоречий;

распределенное решение проблем, декомпозируемых на параллельно решаемые подзадачи с самостоятельными источниками знаний;

применение различных стратегий вывода заключений в зависимости от типа решаемой проблемы;

обработка больших массивов информации из баз данных;

использование математических моделей и внешних процедур для имитации развития ситуаций.

2) *Синтезирующие*. В системах решение синтезируется из отдельных фрагментов знаний.

5. По *видам используемых данных* и знаний различают ЭС с *детерминированными* и *неопределенными* знаниями. Под неопределенностью знаний и данных понимаются их неполнота, ненадежность, нечеткость.

Большинство ЭС включают знания, по содержанию которых их можно отнести одновременно к нескольким типам. Например, обучающая система может также обладать знаниями, позволяющими выполнять диагностику и планирование. Она определяет способности обучаемого по основным направлениям курса, а затем с учетом полученных данных составляет учебный план. Управляющая система может применяться для целей контроля, диагностики, прогнозирования и планирования. Система, обеспечивающая сохранность жилища, может следить за окружающей обстановкой, распознавать происходящие события (например, открылось окно), выдавать прогноз (вор-взломщик намеревается проникнуть в дом) и составлять план действий (вызвать полицию).

Структура экспертной системы

ЭС должна решать задачи логического вывода с достаточной гарантией получения верного результата. Программа ЭС должна взаимодействовать со значительными объемами информации, называемой базой знаний (БЗ). Во время проведения консультаций с БЗ система должна выводить логические

заключения (принимать решения). Кроме того, ЭС должна иметь средства взаимодействия с пользователем-непрофессионалом. Эти средства, включающие оконный интерфейс, программы диалога и общения на одном из естественных языков, объединяются в систему пользовательского интерфейса.

База знаний является центральной частью ЭС. Она содержит факты и правила, описывающие взаимосвязи между фактами. БЗ располагается на диске и содержит статическую (не изменяющуюся во время решения текущей задачи) и динамическую (изменяющуюся при решении задачи) информацию.

Механизм логического вывода включает в себя средства, реализующие формальные методы доказательства (вывода заключений). Вывод осуществляется при помощи поиска и сопоставления по образцу (в Турбо Прологе для этого используется поиск по дереву решений с применением операций унификации, конкретизации и автоматического возврата на предшествующий уровень дерева). Следует помнить, что пользователь будет получать ответы на свои вопросы в соответствии с логикой, заложенной в системе. Для Турбо Пролога любая цель будет доказываться одним и тем же способом: система будет двигаться по логической цепочке в обратном направлении от цели к фактам, пытаясь установить истинность или ложность исходного положения. Механизм логического вывода, как правило, поддерживается стандартными средствами систем искусственного интеллекта. Например, в случае Турбо Пролога весь механизм вывода в процессе компиляции автоматически присоединяется к программе пользователя. В том случае, если механизм вывода Пролога не устраивает по каким-либо причинам пользователя, необходима разработка собственных программных средств механизма логического вывода.

Система пользовательского интерфейса (СПИ) обеспечивает взаимодействие, пользователя с ЭС. В современных ЭС это взаимодействие обычно включает:

- прием и отображение информации с использованием устройств ввода и вывода инструментальной ЭВМ;
- поддержка удобной для пользователя формы диалога (с использованием естественного языка);
- обработка исключительных ситуаций непонимания между пользователем и ЭС (включая элементы обучения системы).

Ситуации непонимания могут возникать по двум причинам: либо из-за ошибки (при вводе данных), либо на принципиальной основе (недостаток знаний в БЗ). Для подобных случаев должны быть предусмотрены методы разрешения конфликтов, начиная с простого изложения фактов, присутствующих в базе, до процессов обучения, модифицирующих факты и правила в БЗ.

Первый элемент ЭС – база знаний – зависит от конкретной области приложения. Механизм логического вывода и системы пользовательского интерфейса, в общем случае, не зависят от области приложения. Поэтому их принято объединять в понятие универсальной оболочки ЭС.

Существует ряд прикладных ***задач***, которые решаются с помощью систем, основанных на знаниях, более успешно, чем любыми другими средствами. При определении целесообразности применения таких систем нужно руководствоваться следующими критериями.

1. Данные и знания надежны и не меняются со временем;

2. Пространство возможных решений относительно невелико;

3. В процессе решения задачи должны использоваться формальные рассуждения;

Существуют системы, основанные на знаниях, пока еще не пригодные для решения задач методами проведения аналогий или абстрагирования (человеческий мозг справляется с этим лучше). В свою очередь традиционные компьютерные программы оказываются эффективнее систем, основанных на знаниях, в тех случаях, когда решение задачи связано с

применением процедурного анализа. Системы, основанные на знаниях, более подходят для решения задач, где требуются формальные рассуждения.

4. Должен быть по крайней мере один эксперт, который способен явно сформулировать свои знания и объяснить свои методы применения этих знаний для решения задач.

В целом ЭС не рекомендуется применять для решения следующих типов задач:

- математических, решаемых обычным путем формальных преобразований и процедурного анализа;
- задач распознавания, поскольку в общем случае они решаются численными методами;
- задач, знания о методах решения которых отсутствуют (невозможно построить базу знаний).

Перечень типовых задач, для решения которых предназначены ЭС, включает:

- формирование информации из первичных данных (например, из сигналов, поступающих от гидролокатора);
- диагностика неисправностей (как в технических системах, так и в человеческом организме);
- структурный анализ сложных объектов (например, химических соединений);
- выбор конфигурации сложных многокомпонентных систем (например, распределенных компьютерных систем);
- планирование последовательности выполнения операций, приводящих к заданной цели (в финансировании проектов различного уровня рискованности).

Следует отметить, что для решения перечисленных задач могут применяться программные системы, которые необязательно относятся к классу ЭС. Это могут быть, как традиционные прикладные системы, так и системы искусственного интеллекта. Для того чтобы выделить ЭС в

отдельный, четко определенный класс программных систем, необходимо определить набор признаков, которые им присущи в той или иной степени. Эти признаки определяются в результате анализа различных характеристик ЭС.

§2. Этапы разработки экспертных систем

Этапы разработки экспертных систем (ЭС) имеет существенные отличия от разработки обычного программного продукта. Опыт создания ЭС показал, что использование при их разработке методологии, принятой в традиционном программировании, либо чрезмерно затягивает процесс создания ЭС, либо вообще приводит к отрицательному результату.

Использовать экспертные системы следует только тогда, когда разработка ЭС *возможна, оправдана и* методы инженерии знаний *соответствуют* решаемой задаче. Чтобы разработка экспертных систем была *возможной* для данного приложения, необходимо одновременное выполнение по крайней мере следующих требований:

1) существуют эксперты в данной области, которые решают задачу значительно лучше, чем начинающие специалисты;

2) эксперты сходятся в оценке предлагаемого решения, иначе нельзя будет оценить качество разработанной ЭС;

3) эксперты способны вербализовать (выразить на естественном языке) и объяснить используемые ими методы, в противном случае трудно рассчитывать на то, что знания экспертов будут "извлечены" и вложены в экспертных системах;

4) решение задачи требует только рассуждений, а не действий;

5) задача не должна быть слишком трудной (т.е. ее решение должно занимать у эксперта несколько часов или дней, а не недель);

6) задача хотя и не должна быть выражена в формальном виде, но все же должна относиться к достаточно "понятной" и структурированной области, т.е. должны быть выделены основные понятия, отношения и известные (хотя бы эксперту) способы получения решения задачи;

7) решение задачи не должно в значительной степени использовать "здравый смысл" (т.е. широкий спектр общих сведений о мире и о способе его функционирования, которые знает и умеет использовать любой нормальный человек), так как подобные знания пока не удается (в достаточном количестве) вложить в системы искусственного интеллекта.

Использование экспертных систем в данном приложении может быть возможно, но не оправдано. Применение ЭС может быть *оправдано* одним из следующих факторов:

1. решение задачи принесет значительный эффект, например экономический;

2. использование человека-эксперта невозможно либо из-за недостаточного количества экспертов, либо из-за необходимости выполнять экспертизу одновременно в различных местах;

3. использование ЭС целесообразно в тех случаях, когда при передаче информации эксперту происходит недопустимая потеря времени или информации;

Приложение *соответствует* методам экспертных систем, если решаемая задача обладает совокупностью следующих характеристик:

1) задача может быть естественным образом решена посредством манипуляции с символами (т.е. с помощью символических рассуждений), а не манипуляций с числами, как принято в математических методах и в традиционном программировании;2) задача должна иметь эвристическую, а не алгоритмическую природу, т.е. ее решение должно требовать применения эвристических правил. Задачи, которые могут быть гарантированно решены (с соблюдением заданных ограничений) с помощью некоторых формальных процедур, не подходят для применения экспертных систем;3) задача должна быть достаточно сложна, чтобы оправдать затраты на разработку экспертных систем. Однако она не должна быть чрезмерно сложной (решение занимает у эксперта часы, а не недели), чтобы ЭС могла ее решать;4) задача должна быть достаточно узкой, чтобы решаться методами ЭС, и практически значимой.

При разработке ЭС, как правило, используется концепция "быстрого прототипа". Суть этой концепции состоит в том, что разработчики не пытаются сразу построить конечный продукт. На начальном этапе они создают прототип (прототипы) ЭС. Прототипы должны удовлетворять двум

противоречивым требованиям: с одной стороны, они должны решать типичные задачи конкретного приложения, а с другой - время и трудоемкость их разработки должны быть весьма незначительны, чтобы можно было максимально параллеливать процесс накопления и отладки знаний (осуществляемый экспертом) с процессом выбора программных средств. Для удовлетворения указанным требованиям, как правило, при создании прототипа используются разнообразные средства, ускоряющие процесс проектирования.

Прототип должен продемонстрировать пригодность методов инженерии знаний для данного приложения. В случае успеха эксперт с помощью инженера по знаниям расширяет знания прототипа о проблемной области. При неудаче может потребоваться разработка нового прототипа или разработчики могут прийти к выводу о непригодности методов ЭС для данного приложения. По мере увеличения знаний прототип может достигнуть такого состояния, когда он успешно решает все задачи данного приложения. Преобразование прототипа ЭС в конечный продукт обычно приводит к перепрограммированию ЭС на языках низкого уровня, обеспечивающих как увеличение быстродействия экспертных систем, так и уменьшение требуемой памяти. Трудоемкость и время создания ЭС в значительной степени зависят от типа используемого инструментария.

В ходе работ по созданию экспертных систем сложилась определенная технология их разработки, включающая шесть следующих этапов (Рисунок 1.4): идентификацию, концептуализацию, формализацию, выполнение, тестирование, опытную эксплуатацию. На этапе *идентификации* определяются задачи, которые подлежат решению, выявляются цели разработки, определяются эксперты и типы пользователей. На этапе *концептуализации* проводится содержательный анализ проблемной области, выявляются используемые понятия и их взаимосвязи, определяются методы решения задач.

На этапе *формализации* выбираются инструментальные средства и определяются способы представления всех видов знаний, формализуются основные понятия, определяются способы интерпретации знаний, моделируется работа системы, оценивается адекватность целям системы зафиксированных понятий, методов решений, средств представления и манипулирования знаниями.

На этапе *выполнения* осуществляется наполнение базы знаний, создание прототипа ЭС. Главное в создании прототипа заключается в том, чтобы этот прототип обеспечил проверку адекватности идей, методов и способов представления знаний решаемым задачам.

Создание первого прототипа должно подтвердить, что выбранные методы решений и способы представления пригодны для успешного решения, по крайней мере, ряда задач из актуальной предметной области, а также продемонстрировать тенденцию к получению высококачественных и эффективных решений для всех задач предметной области по мере увеличения объема знаний.

В ходе этапа *тестирования* производится оценка выбранного способа представления знаний в ЭС в целом. Для этого инженер по знаниям подбирает примеры, обеспечивающие проверку всех возможностей новой ЭС.

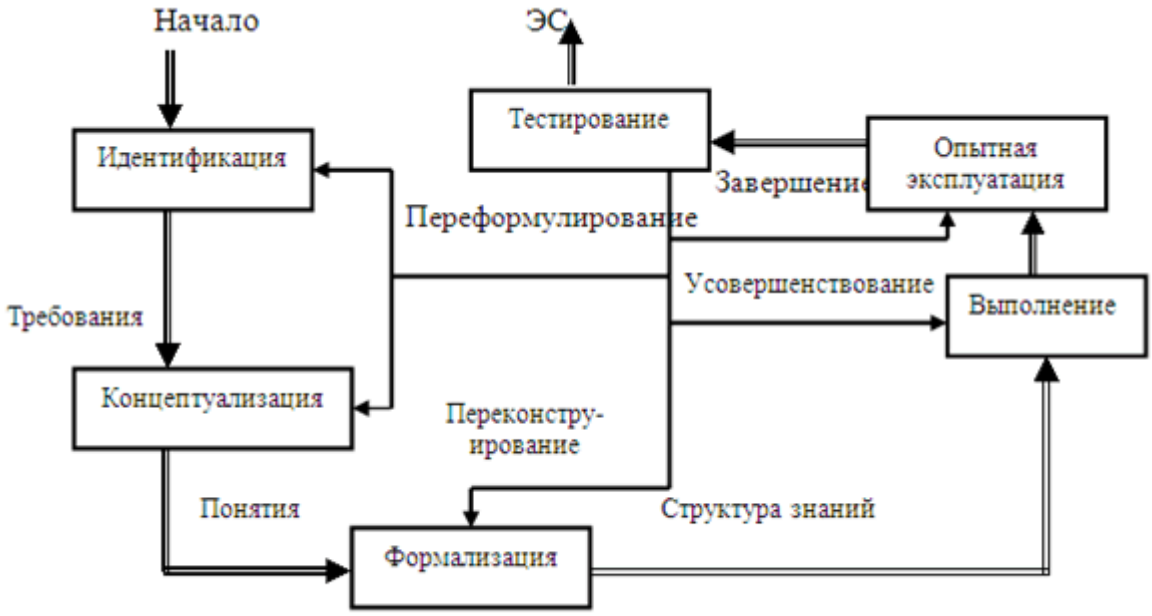

Рисунок 1. Технология разработки ЭС

Различают следующие источники неудач в работе системы: тестовые примеры, ввод-вывод, правила вывода, управляющие стратегии.

Показательные тестовые примеры являются наиболее очевидной причиной неудачной работы экспертных систем. Поэтому при подготовке тестовых примеров следует классифицировать их по под проблемам предметной области, выделяя стандартные случаи, определяя границы трудных ситуаций и т.п.

Критерии оценки ЭС зависят от точки зрения. При тестировании промышленной системы превалирует точка зрения инженера по знаниям, которого в первую очередь интересует вопрос оптимизации представления и манипулирования знаниями. И, наконец, при тестировании ЭС после опытной эксплуатации оценка производится с точки зрения пользователя, заинтересованного в удобстве работы и получения практической пользы.

На этапе *опытной эксплуатации* проверяется пригодность экспертных систем для конечного пользователя. Пригодность экспертных систем для пользователя определяется в основном удобством работы с ней и ее полезностью. Под полезностью ЭС понимается ее способность в ходе диалога определять потребности пользователя, выявлять и устранять

причины неудач в работе, а также удовлетворять указанные потребности пользователя (решать поставленные задачи). В свою очередь, удобство работы с ЭС подразумевает естественность взаимодействия с ней (общение в привычном, не утомляющем пользователя виде), гибкость ЭС (способность системы настраиваться на различных пользователей, а также учитывать изменения в квалификации одного и того же пользователя) и устойчивость системы к ошибкам (способность не выходить из строя при ошибочных действиях неопытного пользователях).

В ходе разработки экспертных систем почти всегда осуществляется ее модификация. Выделяют следующие виды модификации системы: переформулирование понятий и требований, переконструирование представления знаний в системе и усовершенствование прототипа.

В коллектив разработчиков ЭС входят как минимум четыре человека: эксперт, инженер по знаниям, программист и пользователь.

Возглавляет коллектив инженер по знаниям, это ключевая фигура при разработке систем, основанных на знаниях. ЭС может полностью взять на себя функции, выполнение которых обычно требует опыта человека эксперта или играть роль ассистента для человека принимающего решение.

Процесс разработки ЭС можно разделить на следующие этапы:

1. Выбор проблемы.
2. Разработка прототипа ЭС.
3. Доработка до промышленной ЭС.
4. Оценка ЭС.
5. Стыковка ЭС.
6. Поддержка ЭС.

Выбор подходящей проблемы.

На этом этапе:

- определяется проблемная область;
- подбираются специалисты-эксперты;
- подбирается коллектив разработчиков;

- определяется предварительный подход к решению проблемы;
- готовится подробный план разработки.

Разработка прототипа ЭС.

Прототипная система является сокращенной версией ЭС, спроектированной для проверки правильности представления фактов, связей и стратегий рассуждения эксперта. Объем прототипа – несколько десятков правил, фреймов или примеров. Разработка прототипа ЭС делится на шесть стадий: идентификация проблемы, извлечение знаний, концептуализация (структурирование) знаний, формализация, реализация прототипа, тестирование.

a) *Идентификация проблемы* – знакомство и обучение членов коллектива разработчиков, а также создание неформальной формулировки проблемы. На этом этапе уточняется задача, планируется ход разработки прототипа ЭС, определяются:

ресурсы (время, люди и т.д.);¬

источники знаний (книги, дополнительные эксперты);¬

имеющиеся аналогичные ЭС;¬

классы решаемых задач и т.д.¬

b) *Извлечение знаний* – получение инженером по знаниям наиболее полного из возможных представлений о предметной области и способах принятия решения в ней. Для извлечения знаний инженер использует различные методы: анализ текстов, диалоги, лекции, дискуссии, интервью, наблюдение и др.

c) *Концептуализация (или структурирование) знаний* – разработка неформального описания знаний о предметной области в виде графа, таблицы, диаграммы или текста, которое отражает основные концепции и взаимосвязи между понятиями предметной области. На этом этапе определяются: терминология, список основных понятий и их атрибутов, отношения между понятиями, структура входной и выходной информации, стратегия принятия решений и т.д.

d) *Формализация знаний* – это разработка базы знаний на языке представления знаний. На этом этапе используются: логические методы, продукционные модели, семантические модели, фреймы, объектно-ориентированные языки.

e) *Реализация прототипа* –разработка программного комплекса, демонстрирующего жизнеспособность подхода в целом. На этом этапе создается прототип ЭС (включающий базу знаний, остальные программные модули) при помощи: языков программирования (традиционных, специализированных), инструментальных средств разработки ЭС, «пустых» оболочек ЭС.

f) *Тестирование* –процесс выявления ошибок в подходе и реализации прототипа. Прототип проверяется на: удобство и адекватность интерфейса ввода/вывода, качество проверочных примеров, полнота и непротиворечивость правил в базе знаний.

Развитие прототипа до промышленной ЭС.

Основная работа на этом этапе заключается в расширении базы знаний (добавление правил, фреймов, узлов семантической сети или других элементов знаний). После установления основной структуры знаний ЭС инженер по знаниям приступает к разработке и адаптации интерфейсов, с помощью которых система будет общаться с пользователем и экспертом. Система должна предоставлять пользователю возможность уточнять непонятные моменты, приостанавливать работу и т.д.

Оценка системы.

Необходима для того, чтобы проверить точность работы программы и ее полезность. Оценка проводится по следующим критериям:

- критерии пользователя (понятность работы системы, удобство интерфейсов и т.д.);
- критерии приглашенных экспертов (оценка советов-решений, предлагаемые системой, оценка подсистемы объяснений и т.д.);

- критерии коллектива разработчиков (эффективность реализации, производительность, непротиворечивость базы знаний, количество тупиковых ситуаций и т.д.).

Этап стыковки системы.

Осуществляется соединение ЭС с другими программными средствами в среде, в которой она будет работать, и обучение людей, которых она будет обслуживать. Для подтверждения полезности системы важно предоставить каждому из пользователей возможность поставить перед ЭС реальные задачи и проследить, как она их выполняет. Стыковка включает обеспечение связи ЭС с существующими базами данных и другими системами на предприятии.

Поддержка системы.

Готовые системы для повышения ее быстродействия и увеличения переносимости можно перекодировать на другой язык (например, С), но при этом уменьшится ее гибкость. Это можно производить с системами, которые разработаны для проблемных областей, где знания не изменяются. Если же проблемная область, для которой создана система, изменяется, то ее необходимо поддерживать в той инструментальной среде, где она создавалась.

3. Практические аспекты разработки и внедрения ЭС.

В соответствии с результатами тестирования ЭС может находиться на одной из следующих стадий:

1) демонстрационный прототип (решает только часть задач в рамках ПО, демонстрируя правильность выбранного аппарата логического вывода). Срок доведения системы до этой стадии - около 3-х мес., кол-во правил в базе знаний - 50-100.

2) исследовательский прототип (решает все задачи в рамках ПО, на недостаточно устойчива в работе, не полностью проверена). Срок доведения - 1-2 года, кол-во правил - 200-500.

3) действующий прототип (решает стабильно все задачи в рамках ПО, но недостаточно эффективна в работе, в том числе нерациональное использование памяти, невысокое быстродействие). 2-3 г., 500-1000.

4) промышленная система (обеспчивает эффективную работу и высокое качество решений, содержит по сравнению со стадией 3 намного больше правил в базе знаний, программное обеспечение по сравнению с 3) переписано на язык низкого уровня). 2-4 г., 1000-1500.

5) коммерческая система (предполагает обобщение задач, уход от специфики ПО, предназначается для продажи другим потребителям. Развита по сравнинию с 4) система редактирования знаний, интерфейса с конкретным пользователем, обучения). 3-6 лет, 1000-3000.

В процессе проектирования ЭС следует учитывать тот отрицательный опыт, который накоплен разработчиками на каждой стадии проектирования.

На этапе 1 может оказаться, что система или задача настолько трудна, что её нельзя реализовать в рамках выдеенной проектировщиком системы ресурсов (время, деньги и т. д.). Проектировщики должны очень внимательно подойти к вопросу, сможет ли решение задачи принести существенную пользу, для пресонала, который её эксплуатирует. Следует учесть, что для сокращения времени проектирования нельзя расширять состав проектировщиков, так как процесс проектирования итеративный и категории проектировщиков не могут в сжатые сроки осмыслить все этапы проектирования.

Традиционной ошибкой этапа 3 является подгонка инструментального средства под понятия и взаимосвязи конкретной ПО.

Нельзя соглашаться при разработке прототипа на программирования сразу же на языках низкого уровня, т. е. без использования инструментальной системы.

На всех стадиях проектирования инженер по знаниям работает с экспертом и при этом возникает целый ряд трудностей, которые заранее надо учитывать:

эксперты всегда должны быть высококвалифицированными, их нужно заинтересовывать материально;

эксперт никогда не может найти время на работу с инженером по знаниям;

в системе обязательно должна использоваться та же терминология, что и у экспертов ПО; только в этом случае эксперт может успешно понимать структуру базы знаний и вносить в неё соответствующие изменения;

с течением времени эксперт теряет интерес к проекту системы и постоянно сокращает время работы с проектировщиком. Для устранения этого проектировщик вносит изменения в систему только вместе с экспертом, тестирует также вместе с ним;

эксперт незнаком, как правило, с компьютером, поэтому PC устанавливается на его рабочем месте и доработка системы производится там же;

при извлечении знаний эксперта проектировщику очень трудно отделить знания ПО от метазнаний, поэтому работа с экспертом должна происходить не по всем вопросам в целом, а по строго спланированным инженером по знаниям порциям.

На этапе 4 при разработке прототипа следует стремиться не разрабатывать самостоятельно средства объяснения, а использовать только те, что есть в инструментальной системе.

При разработке первого прототипа необходимо стремиться к тому, чтобы в базу знаний попали простые универсальные правила и сокращать количество специфических правил.

На этапе 6 следует учитывать, что если размер правил превышает 300, то их исправление и добавление может привести к появлению новых ошибок, поэтому должен существовать специальный тест, который проверяет систему на непротиворечивость правил.

§3. Основные элементы языка Турбо-Пролог

В последние годы внимание специалистов в области программирования, информатики и вычислительной техники все больше привлекает так называемое логическое программирование. Идейные корни логического программирования лежат в математической логике, где усилиями нескольких поколений математиков были созданы методы формального описания задач с помощью логических формул и методы формальных доказательств, позволяющие достаточно удобно описывать функции и отношения и автоматически получать по этим описаниям решения.

Название Пролог есть сокращение для "ПРОграммирование в терминах ЛОГики". Оно относится только к программированию на хорновских дизъюнктах. Логическая программа на основе хорновских дизъюнктов состоит из правил и утверждений. Системой логического вывода является резолюция.

Пролог может быть использован в различных приложениях, относящихся к искусственному интеллекту:

- общение с ЭВМ на естественном языке;
- формальные вычисления;
- написание компиляторов;
- базы данных;
- экспертные системы;
- различные области автоматизированного проектирования и т.д.

Диалект Пролога язык Турбо-Пролог является компиляторно-ориентированным языком программирования высокого уровня, разработан фирмой Borland International и предназначен для программирования задач из области искусственного интеллекта. Он также может быть использован и для других задач общего характера. Турбо-Пролог имеет окна, цветную графику

и интерактивные средства ввода-вывода, что свидетельствует о его максимальном удобстве для пользователей прикладных программ.

Вместе с тем следует отметить некоторую трудность усвоения языка начинающими пользователями. Пролог относится к так называемым декларативным языкам, требующим от автора умения составить формальное описание ситуации, пользуясь понятиями объектов различных типов и отношений между ними. Программа, составленная на этом языке, не содержит детального описания последовательности шагов, ведущих к результату. Как следствие, невелика предсказуемость поведения программы. Однако методические трудности, связанные с обучением использованию Пролога начинающих программистов, можно успешно преодолеть на основе принципа - "делай как мы". Множество последовательно усложняющихся программ гарантирует привитие читателю навыков программирования на этом своеобразном и красивом языке.

1. Основные элементы языка.

1.1 Имена.

Имена используются для обозначения переменных, символьных констант, объявлений и предикатов.

Имя может начинаться с любой латинской буквы или символа подчеркивания "_" , затем следует любая комбинация букв, цифр и символа "_".

При образовании имен необходимо учитывать следующие правила:

- имена символьных констант должны начинаться с маленькой буквы;

- имена переменных должны начинаться с большой буквы или символа подчеркивания "_".

1.2. Типы данных.

Имеется шесть следующих типов данных:

symbol	Последовательность букв, цифр и знаков подчеркивания, которая начинается со строчной буквы или заключена в кавычки
string	Любая последовательность символов, которая заключена в кавычки
char	Отдельный символ, заключенный в апострофы
integer	Целое число в диапазоне от -32768 до 32767
real	Любое число, может быть представлено в экспоненциальном формате
file	Имя файла

1.3. Константы и переменные.

Константы Турбо-Пролога должны быть записаны:

а) либо с маленькой буквы (исключая кириллицу):

fact1, summa, person ;

б) либо стоять в одинарных кавычках (отдельный символ) или бинарных кавычках (строковая константа):

'c' , "summa=", "сумма";

в) либо они являются числами, целыми или вещественными:

25, 0.5, 3.2e-4 .

Таким образом, константы могут быть любого из стандартных типов Турбо-Пролога. В программе тип констант явно не указывается.

Переменные - это цепочки, состоящие из букв, цифр и символа подчеркивания. Они начинаются с прописной буквы или с символа подчеркивания:

X, Summa, List_of_members, _x23.

Переменная может иметь один из стандартных типов, или тип ее определяется в секции domaims. Можно также использовать так называемую анонимную пременную, которая записывается в виде одного символа подчеркивания.

1.4. Программные секции Турбо-Пролога.

Программа на Турбо-Прологе состоит из нескольких программных секций, каждой из которых предшествует ключевое слово, представленное в следующей таблице:

Domains	Определение типов данных
Database	Объявление предикатов базы данных
Predicates	Объявление предикатов
Clauses	Определение фактов или правил
Goal	Цель

В программе необязательно наличие всех секций. Обычно в программе должны быть по крайней мере разделы Predicates и Clauses.

Программа может содержать несколько разделов Domains, Predicates и Clauses, при этом необходимо соблюдать следующие ограничения:

- программная секция должна начинаться с соответствующего ключевого слова (Domains, Database, Predicates, Clauses или Goal);

- может использоваться только одна цель;

- все Clauses, описывающие один и тот же предикат, должны записываться друг за другом;

- разделы, содержащие предикаты базы данных (Database), должны предшествовать объявлениям Predicates.

Ключевые слова разделов можно записывать прописными и строчными буквами.

Рассмотрим описания разделов детальнее.

1.4.1. Секция DOMAINS.

Для обьявления секции domains используются четыре формата:

1. Первый формат:

name = t ,

где name - имя Турбо-Пролога, t - один из стандартных типов. Это объявление используется для объектов, которые синтаксически схожи, а семантически различны. Например, предложение

age, number = integer

объявляет два domains целого типа.

2. Второй формат:

mylist = elevent* ,

где mylist - объявление списка элементов, element - элемент, ранее описанный пользователем в Domains либо один из стандартных типов Турбо-Пролога, “*” обозначает список. Например,

namlist = integer*

объявляет список целых чисел.

3. Третий формат. Сложный Domains, задает описание структур.

region = functor1(d1,d2,...); functor2(d3,d4,...);...

где region объявляет составную область, functor1,functor2 - имена альтернатив составной области, d1,d1,...,d3,d4 - один из типов Турбо-Пролога, стандартный или определенный ранее в программе (типы можно не указывать).

Пример:

object = int(integer); str(string)

mesto = sprava; sleva .

Данный способ указания Domains позволяет рекурсивно описывать объекты сложных типов (деревьев).

4. Четвертый формат

file = name1; name2; ... ; nameN

где name1, name2,... - символические имена файлов.

Domains файлов используется в том случае, когда пользователю необходимо обратиться к файлу по символическому имени. В программе может быть только один domains этого типа, который называется file. Символические имена файлов задаются в domains в качестве альтернатив.

1.4.2. Секция PREDICATES.

Здесь указываются все имена предикатов с соответствующими областями определения аргументов. Аргументы дизъюнктов Пролога

называются термами. Существует три типа термов: константа, переменная, составной терм (структура).

Общий вид описания предиката:

name(d1,d2,...) ,

где name - имя предиката, d1,d2,... - соответственно области определения аргумента1, аргумента2 и т.д. Например:

syn(string)

math(preson,person) .

1.4.3. Секция DATABASE.

Объявление оперативной базы данных. Синтаксис объявления такой же, как и в секции Predicates. Объявленные здесь предикаты не должны объявляться в секции Predicates, но дизъюнкты этих предикатов могут присутствовать в секции Clauses.

1.4.4. Секция CLAUSES.

Здесь описываются все дизъюнкты всех предикатов. Дизъюнкты также могут иметь название статьи или клозы. Это факты и правила, соответствующие каждому из объявленных предикатов. Такой дизъюнкт имеет следующий синтаксис:

functor(d1,d2,...):- cond1,cond2,... .

Здесь functor - имя предиката;

functor(d1,d2,...) - голова дизъюнкта;

cond1,cond2,... - условия истинности дизъюнкта, где запятые означают связку "И"; условия, в свою очередь, могут быть предикатами;

":-" -обозначение "ЕСЛИ".

Если опущены условия, то functor(d1,d2,...) описывает факт. Каждый из дизъюнктов одного предиката секции Clauses соответствует альтернативам этого предиката.

1.4.5. Секция GOAL.

Здесь указывается вопрос (цель), на который должен ответить Турбо-Пролог. Записывается он как дизъюнкт без головы и знака ":-" .

2. Программы с фактами и простыми правилами.

1. Рассмотрим следующую вводную Пролог-программу. На этом примере мы покажем, как создать, отредактировать и выполнить Пролог-программу в интегрированной среде Турбо-Пролога.

```
predicates
    hello
clauses
    hello:-makewindow(1,7,7,"My first program",4,56,10,22),
    nl,write("  Please type your name "),
    cursor(4,5),
    readnl(Name),nl,
    write("  Wellcome  ",Name).
goal
    hello.
```

После исправления ошибок в программе выберите опцию RUN в меню Турбо-Пролога. Введите свое имя (например, Аня) в окне ввода и нажмите ENTER. Программа напечатает

Welcome Аня

и будет ждать нажатия клавиши пробела (SPACE BAR).

Работа программы начинается с выполнения раздела Goal. Цель пытается удовлетворить предикат hello. Описание

likes(mark,tennis).

соответствуют утверждениям на английском языке:

Ellen likes tennis.

John likes football.

Tom likes baseball.

Eric likes awimming.

Mark likes tennis.

Подобным образом вопрос: "Что любит Джон?" сводится к поиску объекта, который отношение likes (нравится) связывает с Джоном; этот

объект и будет служить ответом на запрос. Отметим, что при определе-нии отношения между объектами существенен порядок, в котором эти объекты входят в отношение:

likes(john,football) отличается от likes(football,john).

Использование бэктрекинга для получения всех решений. В данном примере предикат-факт country содержит сведения о названии страны и населении. Программа выдает список стран, население которых превышает 1e7.

```
Predicates
  country(symbol,real)
  print_countries
Clauses
  country(england,3e7).
  country(france,2.3e7).
  country(germany,1.6e7).
  country(denmark,2.4e6).
  country(canada,7.3e6).
  country(chile,2.5e).
  print_countries :-
    country(X,P),
    P > 1e7,
    write(X),
    nl,
    fail.
  print_countries.
Goal print_countries.
```

После того, как выведен на экран очередной результат, стандартный предикат fail завершает работу предиката print_countries ложно и побуждает Турбо-Пролог к поиску других решений. Когда будут найдены все решения,

второй клоз предиката print_countries позволит программе завершиться истинно.

Построение базы данных. На Прологе чрезвычайно просто организовать реляционную базу данных в виде набора некоторых фактов. Пусть база данных содержит следующие сведения об автомобилях: марка машина, год выпуска, цвет, цена:

```
car(volvo,      1990, red,    1800).
car(mersedes, 1988, black, 2000).
car(ford,         1994, white, 3000).
```

Наберите эту программу согласно правилам оформления Пролог-программы и исполните команду RUN. Когда система в диалоговом окне запросит цель, введите запрос:

```
Goal:  car(mersedes,_,_,_).
```

Турбо-Пролог ответит

```
1 solution
Yes
```

в диалоговом окне и будет ждать, когда вы введете другую цель.

В этом запросе использовались анонимные переменные, обозначенные символом подчеркивания. Анонимная переменная используется для обозначения аргументов, значения которых в данный момент могут быть произвольны. Если анонимная переменная встречается в запросе, то ее значение не выводится при ответе системы на этот запрос. Анонимная переменная может также использоваться в записи предикатов, если в предложении она встречается один раз. Примеры записи таких предикатов встретим дальше.

Если зададите в качестве цели

```
Goal:  car(mersedes,1990,_,_)   ,
```

система ответит

```
No solution ,
```

поскольку наша база данных не содержит такого утверждения.

Теперь предположим, что мы хотим узнать все марки машин, имеющиеся в базе данных, т.е. задать вопрос: "Каковы те объекты X, которые являются марками машины?". Тогда наш вопрос запишется так:

Goal: car(X,_,_,_).

Турбо-Пролог ответит:

X=“volvo”

X="mersedes"

X="ford"

3 solutions.

Можно также задавать вопросы более общего характера. К примеру, если требуется найти все машины, выпущенные до 1992 года, то следут задать запрос, который представляет конъюнкцию целей:

Goal car(X,Y,_,_), Y<1992.

Подобная база данных имеет вид целостных информационных объектов. Более гибким является представление базы данных в виде бинарных отношений, в которых один из атрибутов должен выступать в роли ключа, объединяющего все остальные свойства. Пусть в нашем примере это будет марка машины:

```
year(volvo,       1990).
year(mersedes, 1988).
year(ford,         1994).
color(volvo,      red).
color(mersedes, black).
color(ford,       .white).
cost(volvo,       2000).
cost(mersedes, 2500).
cosr(ford,        3000).
```

В случае необходимости атрибуты можно собрать в единое целое при помощи правила

car(M,Y,C,P) :- year(M,Y), color(M,C), cost(M,P).

Динамическая база данных.

Турбо-Пролог поддерживает реляционную базу данных. Предикаты базы данных нужно объявлять в отдельном разделе Database. Программная секция Database должна предшествовать объявлению предикатов Predicates. Объявление предикатов базы данных аналогично их описанию в секции Predicates.

Динамическая база данных - раздел программы, в котором факты могут добавляться или загружаться из файла на диске во время выполнения программы или удаляться из нее. Факты, описывающие предикат базы данных, могут также обрабатываться как обыкновенные предикаты Пролога.

Для работы с базой данных используют стандартные предикаты asseta, assertz, retractall, consult, save и др. (см. раздел "Стандартные предикаты"). Также нужно иметь представление о структуре памяти системы Турбо-Пролога. Исходная программа на Турбо-Прологе хранится в области SOURSE, в то время как для размещения фактов динамической базы данных отведена область HEAP. Поэтому на экране нельзя видеть информацию, записанную в базу данных. Однако ее можно просмотреть как любой файл, если сохранить его в каталоге под заданным именем.

Встроенные предикаты asserta и assertz позволяют программисту добавлять новые утверждения в базу данных Пролога. Поскольку Пролог декларирует идентичность программ и данных, то риск изменения программы является главной причиной того, что надо проявлять известную осторожность при использовании предикатов assert и retract Этот риск минимизируется четким логическим разделением программ и данных . Изменениям подвергаются только хранимые данные.

С другой стороны, можно трактовать базу данных Пролога как некую глобальную переменную. Такая трактовка очень удобна: любая процедура

может записывать в нее свой результат, который будет доступен любой другой процедуре, какая бы дистанция между ними ни была при их последовательных динамических вызовах.

1. Создание базы данных "Таблица умножения".

```
Domains
  list=integer*
Database
  mult(integer,integer,integer)
Predicates
  tabl
  numlist(list)
  member(integer,list)
Clauses
  numlist([1,2,3,4,5,6,7,8,9]).
  tabl:-numlist(L),
    member(X,L), member(Y,L),    %  X  и Y - целые числа из списка L
     Z=X*Y,
ssertz(mult(X,Y,Z)),fail.          % Факт mult добавляется в базу данных
  tabl.
  member(X,[X|_]).
  member(X,[_|T]):-member(X,T).
```

Зададим в диалоговом окне следующую цель:

```
Goal   tabl,                    %  Создание базы данных
       mult(2,5,X).             %  Обращение к базе данных
```

Система ответит:

```
   X=10 Yes.
Goal mult(10,10,X).
Система ответит
       No ,
```

поскольку в базе данных нет такого факта. Допишем в секцию Clauses пару фактов:

```
Clauses
  mult(10,10,100).
  mult(9,10,90).
```

Тогда система будет находить эти факты, но не в области динамической базы данных, а фактов программной секции Clauses.

§4. Проектирование и реализация экспертной системы «Определение неисправности при запуске компьютера»

Реализация этапов разработки экспертной системы

Разработка ЭС разбита на три этапа: выбор метода реализации ЭС, непосредственно кодирование программы и тестирование системы.

Выбор метода реализации экспертной системы

На сегодняшний день существует две основные возможности для реализации ЭС. Это связано с наличием двух весьма различных подходов к решению задачи.

1) основывается на использовании для построения ЭС некоторого процедурного языка, со всеми его недостатками и достоинствами для решения данной задачи;

2) использование языков программирования математической логики, языков в которых имеются имеющие огромное значение для построения ЭС возможности.

Для ЭС лучшим решением является использование некоторого логического языка. Если сравнить код, реализующий одну и ту же ЭС (механизм вывода), то у процедурного языка он будет гораздо более объемным и более сложным. А наличие таких механизмов в языке как сопоставление образцов (унификации), древовидное представление структур, автоматический возврат делают его просто незаменимым языком для программирования ЭС.

Общепринятое представление ЭС в виде базы знаний и механизма вывода не полностью пригодно для ЭС, написанных на Прологе. Многие функции механизма вывода обеспечиваются самим Прологом. Базы знаний, образованные средствами Пролога, являются выполняемыми. Однако Пролог не обеспечивает некоторых важных свойств ЭС, обычно встроенных в механизм вывода. Примеры таких свойств порождение объяснений и рассуждения в условиях неопределенности. Исходя из этого, средой для

реализации основной части ЭС был выбран язык Пролог, в качестве одного из лучших представителей языков логического программирования.

2. *Построение экспертной системы: описание логической части программы*

Программа представляет собой интерактивную оболочку, которая может задавать пользователю вопросы. Когда программа задает вопрос, пользователь может ответить «Да» или «Нет».

Возможность задания вопроса «Почему» реализуется в программе с помощью списка, в котором записывается весь путь, по которому прошла программа для вывода цели к текущему моменту времени. По запросу пользователя из списка извлекается его голова и выдается на экран, это и есть то нужное правило.

Кроме ответа на вопрос «Почему» программа имеет возможность отвечать на вопрос «Как», после вывода сделанного ЭС пользователь может проследить весь процесс вывода.

База знаний представлена в виде логических правил, поясняющих ход размышлений ЭС при ответах пользователя.

- если отсутствует реакция на нажатие кнопки включения,
- если кабель питания подключен правильно то проверить надежность соединения коннекторов на материнской плате и питающие ее провода ,
- если отсутствует реакция на нажатие кнопки включения ,
- если все кабели питания подключены правильно ,
- если провода и коннекторы внутри корпуса ПК исправны,
- если при отсоединении кнопки перезагрузки от материнской платы компьютер начинает работать, то неисправна кнопка перезагрузки.
- если отсутствует реакция на нажатие кнопки включения,
- если все кабели питания подключены правильно,
- если провода и коннекторы внутри корпуса ПК исправны,

- если при замыкании двух контактов с надписью «Power Switch» происходит запуск компьютера , то неисправна кнопка запуска.
- если отсутствует реакция на нажатие кнопки включения,
- если все кабели питания подключены правильно,
- если включенный в сеть блок питания не начинает работать при замыкании черного и зеленого контактов, то неисправен блок питания.
- если при нажатии кнопки запуска компьютер начинает работать,
- если компьютер сразу прекращает работу, то неисправно какое-либо устройство и срабатывает защита блока питания.
- если при нажатии кнопки запуска компьютер начинает работу,
- если в процессе загрузки компьютер подает характерный сигнал,

то неисправно оборудование согласно соответствующему значению сигнала {для различных МП звуковой сигнал может иметь разные значения, расшифровка сигналов приведена в конце описания}.

- если при нажатии кнопки запуска компьютер начинает работу,
- если на экране монитора не отображаются никакие данные,
- если не звучит ни один из сигналов, то проблема в BIOS{можно попробовать решить ее сбросив параметры BIOS.

Clear CMOS - три контакта, два из которых соединены. Запомнить исходное положение джемпера, затем вытащить его и соединить с его помощью другую пару контактов, подождать десять секунд. После этого установить его снова в исходное положение, или, если на системной плате есть кнопка перезагрузки, нажать ее} или неисправность в МП.

- если при нажатии кнопки запуска компьютер начинает работу,
- если на экране монитора не отображаются никакие данные,
- если не звучит ни один из сигналов,
- если после перезагрузки BIOS и проверки батарейки не происходит загрузка, то неисправна материнская плата.
- если при нажатии кнопки запуска компьютер начинает работу,
- если не звучит ни один из сигналов,

- если на экране монитора появляется «Please enter Setup to recover BIOS setting | CMOS Date/Time Not Set », то проблема в настройках BIOS.
- если при нажатии кнопки запуска компьютер начинает работу,
- если не звучит ни один из предупреждающих сигналов,
- если при загрузке на мониторе появляется сообщение о том, что BIOS не может найти загрузочный носитель,
- если в BIOS не отображается HDD, то неисправен жесткий диск.
- если при нажатии кнопки запуска компьютер начинает работу,
- если не звучит ни один из сигналов,
- если при загрузке на мониторе появляется сообщение о том, что BIOS не может найти загрузочный носитель,
- если в BIOS отображается HDD
- если жесткий диск работает при подключении к другому компьютеру {или к этому же при наличии master} как slave, то поврежден загрузочный сектор.
- если при нажатии кнопки запуска компьютер начинает работу,
- если не звучит ни один из предупреждающих сигналов,
- если при загрузке на мониторе появляется логотип ОС, но загрузка не происходит, то проблема в программном обеспечении.

1. Дерево принятия решений

- Построим дерево принятия решений о неисправности компьютера. Дерево представлено на рис. 1.1.

<table>
<tr><td colspan="6">Включается ли ПК?</td></tr>
<tr><td colspan="2">НЕТ</td><td colspan="4">ДА</td></tr>
<tr><td colspan="2">Есть ли ток в розетке?</td><td colspan="4">Включается ли монитор?</td></tr>
<tr><td>НЕТ</td><td>ДА</td><td>НЕТ</td><td colspan="4">ДА</td></tr>
<tr><td rowspan="4">Восстановите подачу питания</td><td rowspan="4">Почините блок питания</td><td rowspan="4">Почините монитор</td><td colspan="4">Есть ли изображение на мониторе?</td></tr>
<tr><td colspan="2">НЕТ</td><td colspan="2">ДА</td></tr>
<tr><td colspan="2">Подключен ли монитор к ПК?</td><td colspan="2">Загружается ли ОС?</td></tr>
<tr><td>НЕТ</td><td>ДА</td><td>НЕТ</td><td>ДА</td></tr>
</table>

			Подключите монитор к ПК	**Почините монитор**	**Переустановите операционную систему**	**Ремонт не требуется**
1	**2**	**3**	**4**	**5**	**6**	**7**

Рис.1. Дерево принятия решений

На основании построенного дерева составим список правил продукции.

- Правило 1. Если не удается включить ПК, и нет тока в розетке, то требуется восстановить подачу питания.
- Правило 2. Если не удается включить ПК, и есть ток в розетке, то требуется починить блок питания.
- Правило 3. Если удается включить ПК, и не включается монитор, то требуется починить монитор.
- Правило 4. Если удается включить ПК, и включается монитор, и нет изображения на экране, и монитор не подключен к ПК, то требуется подключить монитор к ПК.
- Правило 5. Если удается включить ПК, и включается монитор, и нет изображения на экране, и монитор подключен к ПК, то требуется починить монитор.
- Правило 6. Если удается включить ПК, включается монитор, есть изображение на экране, не загружается операционная система, то требуется переустановить операционную систему.
- Правило 7. Если удается включить ПК, включается монитор, есть изображение на экране, загружается операционная система, то ремонт не требуется.

2. Проектирование интерфейса пользователя для экспертной системы

Экспертная система «Определение неисправности при запуске компьютера»

database

xpositive(symbol,symbol)

xnegative(symbol, symbol)

predicates

do_expert

do consulting

ask(symbol, symbol)

pk_is(symbol)

positive (symbol,symbol)

negative(symbol,symbol)

remember(symbol,symbol,symbol)

clear_facts

goal

do_expert.

clauses

do_expert:-

makewindow(l,7,7, "Экспертная система", 1,3,22,71),

nl,write (" ———————————————————————————"),

nl,write ("Определение неисправности при запуске компьютера"),

nl,write (" "), nl,write("Пожалуйста, отвечайте на вопросы 'yes' или 'no'."),

nl,nl, do_consulting,

write (" Для продолжения нажмите любую клавишу"),nl,

readchar (_),removewindow, exit.

do_consulting:- pk_is(X),!, nl, write(X,“.”),nl,

clear_facts.

do_consulting:- nl, write(" Извините мы не можем определить неисправность Вашего ПК!"),

clear_facts.

ask(X,Y):- write(" expert ",X," ",Y," ?"),

readln (Reply), remember(X,Y,Reply).

positive (X,Y):- xpositive(X,Y),!.

positive (X,Y):- not(negative(X,Y)),!, ask(X,Y).

negative(X,Y):- xnegative(X,Y),!.

remember(X,Y,yes):- asserta(xpositive(X,Y)).

remember(X,Y,no):- asserta(xnegative(X,Y)), fail.

clear_facts:- retract(xpositive(_,_)), fail.

clear_facts:- retract(xnegative (_,_)), fail.

pk_is("Неисправна кнопка перезагрузки "):-

positive (" если отсутствует реакция на нажатие кнопки включения "),

positive (" если все кабели питания подключены правильно"),

positive (" если провода и коннекторы внутри корпуса ПК исправны"),

positive (" если при отсоединении кнопки перезагрузки от материнской платы компьютер начинает работать"),!.

pk _is("Неисправен жесткий диск "):-

positive (" если при нажатии кнопки запуска компьютер начинает работу "),

positive ("если не звучит ни один из предупреждающих сигналов "),

positive ("если при загрузке на мониторе появляется сообщение о том, что BIOS не может найти загрузочный носитель "),

positive ("если в BIOS не отображается HDD "),!.

И т.д.

Работу экспертной системы иллюстрируют рис.1, рис.2 и рис.3.

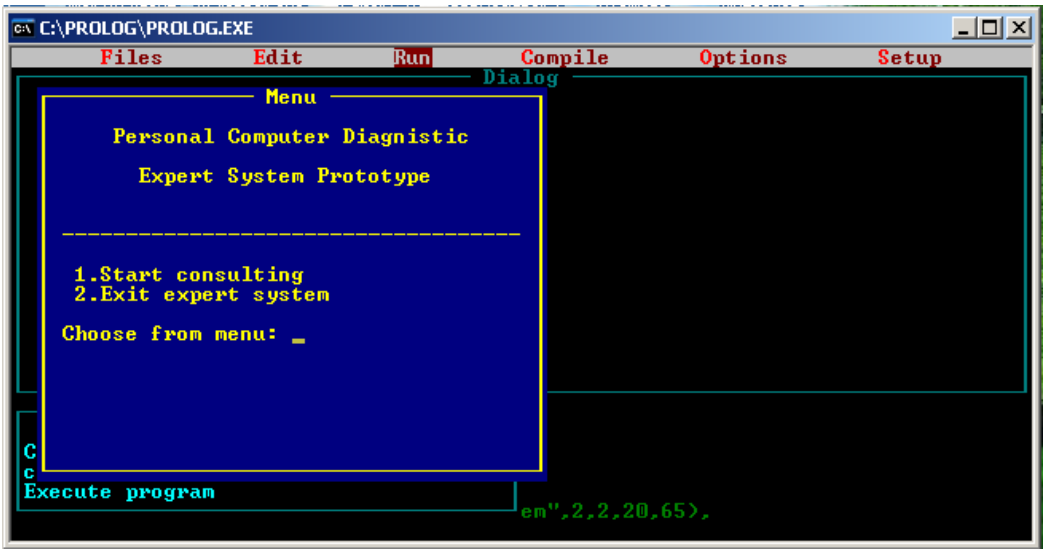

рис.1

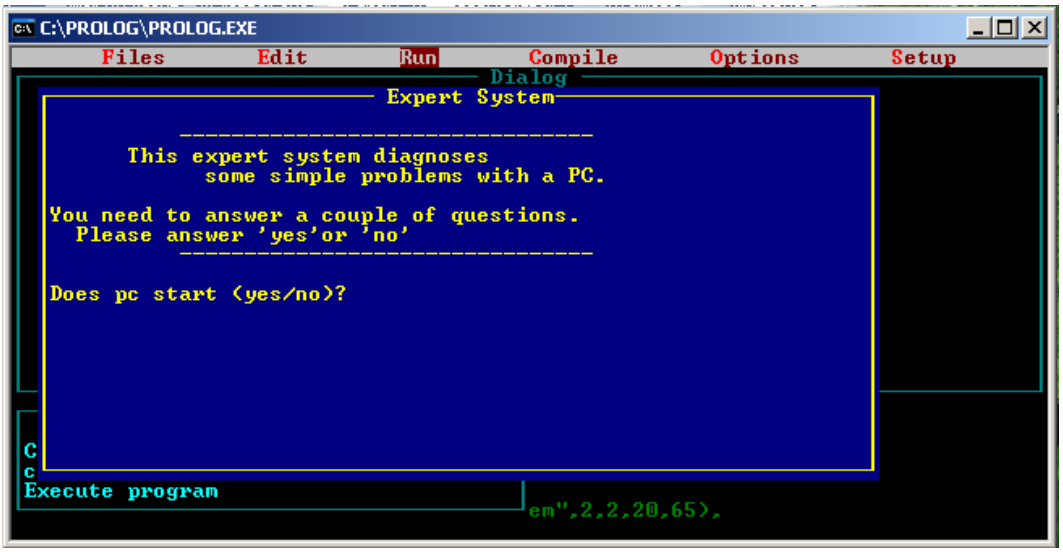

рис.2.

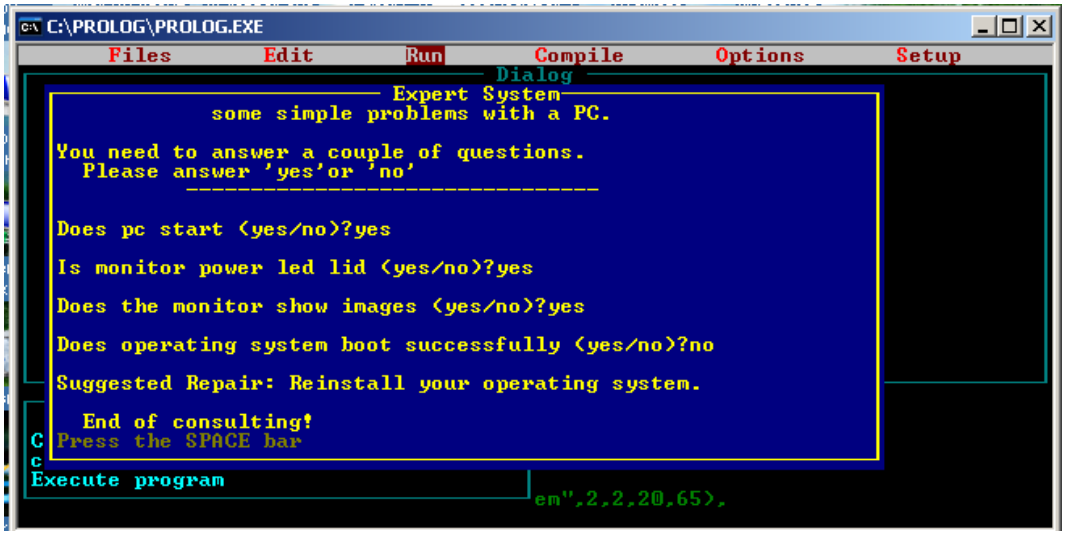

рис.3

ЗАКЛЮЧЕНИЕ

В данной работе были описаны основные характеристики, рассмотрена структура и функции ЭС, а так же преимущества перед традиционными информационными системами. ЭС разрабатываются с целью облегчить и автоматизировать деятельность экспертов в той или иной предметной области. В отличие от человека, система не знает усталости, обладает меньшей вероятностью ошибок, и в то же время способна к самообучению, накапливанию опыта подобно эксперту.

Разработка ЭС является наиболее плодотворной быстро развивающейся областью применения Пролога. Очень важно понимать, как работают ЭС, так как они могут использоваться, фактически, в любой области знаний.

В ходе выполнения данного исследования были достигнуты поставленные цель и задачи.

Результатом выполнения данной работы стало создание экспертной системы «Определение неисправности при запуске компьютера», позволяющей по наличию ряда признаков определять неисправность персонального компьютера.

В качестве основных результатов работы можно указать следующее:

- Изучение основных возможностей, структур и функций экспертных систем.
- Изучение приемов работы по созданию программ средствами языка Пролог.
- Создание конкретного примера экспертной системы на языке Пролог.

Предполагается, что данная экспертная система будет актуальна для начинающих пользователей ПК, а также студентов, изучающих архитектуру ПК.

Список литературы

1. Андрейчиков А.В., Андрейчикова О.Н. «Интеллектуальные информационные системы»: М. Наука, 2004 г.
2. Братко И. «Программирование на языке Пролог для искусственного интеллекта». / Пер. с англ. – М.: Мир, 1990. – с. 560
3. Егоров Н. В., Карпов А. Г. «Диагностические информационно-экспертные системы». – М.: Вильямс, 2002 .
4. Зубов В. В., Макушкин В. А. «Экспертная система диагностирования цифровых устройств ДИЭКС на персональной ЭВМ. Экспертные системы на персональных компьютерах». М.: МДНТП, 2005, с. 115-120.
5. Круглов В. В. «Интеллектуальные информационные системы». – СПб: Питер. - 2002. – 234 с.
6. Макушкин В. А., Щербицкий К. А. Экспертная система для контроля и диагностирования цифроаналоговых устройств. Новые информационные технологии в планировании, управлении и в производстве. М.: МДНТП. – 2001. - С. 121-125.
7. Могилев А. В. и др. «Информатика»: Учеб. пособие для студ. пед. вузов/ Могилев А. В., Пак Н. И., Хеннер Е. К.; Под. ред. Хеннера Е. К. – 2-е изд., стер. – М.: Изд. центр «Академия», 2001. – С. 816
8. Муромцев Д.И. «Введение в технологию экспертных систем». СПб: СПб ГУ ИТМО, 2005.
9. Павлов И.О., Кулакова С.В. Основы программирования на языке Турбо-пролог: Методические указания к практическим занятиям по курсу «Представление знаний в информационных системах» / Воронеж. гос. технол. акад.; – Воронеж, 2001. – с. 450, с.32
10. Попов Э. В., Фоминых И. Б., Кисель Е. Б., Шапт М. Д. «Статические и динамические экспертные системы». М.: Финансы и статистика, 2003 г.

11. Соломин Д. «Использование Турбо-Пролога». / Ц. Ин.; Пер. с англ. – М.: Мир, 1993. – с. 608

12. Джарратано Джозеф, Райли Гари «Экспертные системы: принципы разработки и программирование»: Пер. с англ. – М. : Издательский дом «Вильямс», 2006. – с. 1152

13. Марселлус Д. «Программирование экспертных систем на Турбо Прологе». Пер. с англ. - М.: Финансы и статистика, 1994 г.

14. Нейлор К. «Как построить свою экспертную систему».- М.: Энегроатомиздат, 2007.

15. *Питер Джексон* «Введение в экспертные системы» = Introduction to Expert Systems. – 3-е изд. – М.: Вильямс, 2001. - с. 624.

16. Стерлинг Л., Шапиро Э. «Искусство программирования на языке Пролог». / Пер. с англ. – М.: Мир, 1990. – с. 225

17. Таунсенд К., Фохт Д. «Проектирование и программная реализация экспертных систем на персональных ЭВМ»: Пер. с англ. В. А. Кондратенко, С. В. Трубицына. – М.: Финансы и статистика, 1990. - с. 320

18. Уотермен Д. «Руководство по экспертным системам»: Пер. с англ. под ред. В. Л. Стефанюка. — М.: «Мир», 1989: - с. 388

19. http://www.swi-prolog.org/ - официальный сайт SWI-Prolog.

20. http://www.expertsys.ru - материалы сайта «Все об экспертных системах»

21. http://www.intuit.ru

22. http://www.ai.tsi.lv

23. http://www.libray.narod.ru

24. http://expro.kzn.ru

25. http://tver.mesi.ru

ПРИЛОЖЕНИЕ

Листинг программы представлен на рис.2.1.

```
/* Программа позволяет диагностировать простые проблемы */
/* при включении компьютера */

domains
database
     xpositive(symbol)
     xnegative(symbol)
predicates
     do_expert_job
     do_consulting
     ask(symbol)
     process(integer)
     answer()
     show_menu
     cons
     positive(symbol)
     negative(symbol)
     remember(symbol,symbol)
      valid_answer(symbol)
     clear_facts
goal
     do_expert_job.
clauses

     /* Система пользовательского интерфейса */
     do_expert_job:-
        show_menu,
        readchar(_),
        removewindow,
        exit.
     show_menu:-
        makewindow(1,30,30," Menu ",2,2,20,40), nl,
          write("     Personal Computer Diagnistic     "),nl,
        write("       Expert System Prototype      "),nl,nl,nl,
         write(" ------------------------------------"),nl,nl,
        write("  1.Start consulting              "),nl,
        write("  2.Exit expert system"),nl,
        write("                                  "),nl,
        write(" Choose from menu: "),
        readint(Choice),
        process(Choice).
process(1):-
        cons.
     process(2):-
        removewindow,
        exit.

     cons:-
        makewindow(1,30,30," Expert System",2,2,20,65),
        nl,write("          --------------------------------"),
        nl,write("      This expert system diagnoses"),
```

```
        nl,write("            some simple problems with a PC."),
       nl,nl,write("You need to answer a couple of questions."),
        nl,write("  Please answer 'yes'or 'no'  "),
        nl,write("          --------------------------------"),
        nl,nl,do_consulting.
     do_consulting:-
        answer(),!,nl,nl,
        write("  End of consulting!  "),
        clear_facts.

     do_consulting :-
         clear_facts.

     ask(X):-
        write(X,"?"),
        readln(Reply),
          nl,valid_answer(Reply),
        remember(X,Reply).
valid_answer(Reply):-
            Reply="yes";Reply="no".
    valid_answer(Reply):-
            K="yes",K1="no",
            Reply<>K,Reply<>K1,nl,
            write("ERROR! Answer 'yes' or 'no'"),nl.

     positive(X):-
        xpositive(X),!.
     positive(X):-
        not(negative(X)),!,ask(X).
     negative(X):-
        xnegative(X),!.
     remember(X,"yes"):-
        asserta(xpositive(X)).
remember(X,"no"):-
        asserta(xnegative(X)),fail.
     clear_facts:-
        retract(xpositive(_)),fail.
     clear_facts:-
        retract(xnegative(_)),fail.
```