РЕФЕРАТ

Темой ВКР является "Экспертные системы управления антропоморфным персонажем. Анализ состояний и проектирование правил вывода.". При выполнении данной дипломной работы использовались такие программные средства как, CLIPS.

Ключевые слова: экспертная система, база знаний, CLIPS, управление персонажем.

В пояснительной записке первый раздел содержит теоретическую информацию об экспертных системах. Во втором разделе проведен анализ предметной области, существующих экспертных систем. В третьем разделе описаны технические средства проектирования, представлена программная реализация.

Результатами выпускной квалификационной работы являются спроектированные правила вывода и база знаний.

Система является частью проекта по созданию технических средств обучения искусственного интеллекта необходимым движениям. На данном этапе в базе знаний представлены правила логического вывода упрощенных движений смешанных единоборств.

Пояснительная записка содержит 60 страницы, рисунков - 6, таблиц - 1, список литературы - 29 наименований. Приложения занимают 5 страниц.

ВВЕДЕНИЕ

Проблемы принятия решений существуют во всех областях деятельности человека. Люди принимают решения абсолютно во всех жизненных ситуациях. При выборе в какой магазин пойти, при покупке мебели, какую книгу почитать перед сном и во многих других ситуациях. Решение, принимаемое человеком, должно быть самым лучшим из возможных, тем не менее рассмотреть все детали, влияющие на определенный выбор и принять очевидное и в то же время разумное и правильное решение бывает проблематично без стороннего вмешательства.

Процедура принятия решений очень сложна и связана с обработкой огромных объемов информации, а также её структурированием, в этом, компьютер прекрасно дополняет человека. Например, при вычислении большого объема вероятных исходов лучше подойдет компьютер, а человек больше разбирается в целях и конечной идее.

Рост информационных технологий привел к созданию систем принятия правильных решений, а в частности экспертных систем. В настоящее время, ИИ успешно развивается, в связи с этим появляется возможность получения новых результатов в данной области, а появление новых технических средств позволяет использовать эти возможности.

Цель работы – проектирование правил логического вывода и создание базы знаний.

Объектом работы является – экспертная система.

Предметом работы является – база знаний.

Задачи работы:

* Рассмотреть теоретические основы экспертных систем.
* Рассмотреть теоретические основы баз знаний.
* Изучить технологии разработки экспертных систем.
* Изучить принципы проектирования и разработки модели экспертной системы.
* Рассмотреть классификацию экспертных систем.
* Проанализировать существующие экспертные системы.
* Выбор и изучение средств разработки.
* Проектирование правил логического вывода.
* Создание базы знаний.

ОПРЕДЕЛЕНИЯ, ОБОЗНОЧЕНИЯ И СОКРАЩЕНИЯ

*БЗ* - База знаний.

*ИИ* - Искусственный интеллект

*ИТ* - Информационные технологии.

*ЭС* - Экспертная система.

СОДЕРЖАНИЕ

[ОПРЕДЕЛЕНИЯ, ОБОЗНОЧЕНИЯ И СОКРАЩЕНИЯ 7](#_Toc484772279)

[СОДЕРЖАНИЕ 8](#_Toc484772280)

[1 ОСНОВНЫЕ ПОНЯТИЯ ЭКСПЕРТНЫХ СИСТЕМ 9](#_Toc484772281)

[1.1 Определение понятий экспертных систем 9](#_Toc484772282)

[1.2 Базы знаний 13](#_Toc484772283)

[1.3 Развитие технологий разработки экспертных систем 15](#_Toc484772284)

[2 Современные экспертные системы 20](#_Toc484772285)

[2.1 Принципы проектирования и разработки модели экспертной системы 20](#_Toc484772286)

[2.2 Классификация экспертных систем 26](#_Toc484772287)

[2.3 Анализ существующих экспертных систем 30](#_Toc484772288)

[3 ВЫБОР ТЕХНИЧЕСКИХ СРЕДСТВ И СОЗДАНИЕ БАЗЫ ЗНАНИЙ 36](#_Toc484772289)

[3.1 Общие сведения о CLIPS 36](#_Toc484772290)

[3.2 Описание возможных состояний антропоморфного персонажа 40](#_Toc484772291)

[3.3 Общее описание базы знаний 45](#_Toc484772292)

[ЗАКЛЮЧЕНИЕ 47](#_Toc484772293)

[СПИСОК ЛИТЕРАТУРЫ 49](#_Toc484772294)

[ПРИЛОЖЕНИЕ А. Структура правил вывода в интерфейс. 52](#_Toc484772295)

[ПРИЛОЖЕНИЕ Б. Блок правил. 55](#_Toc484772296)

# ОСНОВНЫЕ ПОНЯТИЯ ЭКСПЕРТНЫХ СИСТЕМ

Определение понятий экспертных систем

Экспертная система - система искусственного интеллекта, предназначенная помочь эксперту-человеку в решении узконаправленных задач, которые не имеют четкой логики. Эти системы начали разрабатываться еще в 60-х годах и понимались как искусственный интеллект специального типа [3].

Первые разработки экспертных систем содержали в себе исключительно знания эксперта-человека, т.е. знания недоступные для большинства. С течением времени эта концепция постепенно сглаживалась и сейчас под понятием ЭС подразумеваются системы, не только наполненные только экспертными знаниями, но и могут быть простые знания доступные всем.

Главная их особенность в том, что они должны уметь “объяснить” свое решение пользователю, также, как и эксперт может рассказать почему он поступил именно так в данной ситуации. Это требуется, когда есть вероятность ошибки в системе при тестировании новых знаний или, когда система находиться в области, для которой характерна некая неточность, неполнота начальных данных, требующих решение задачи. Поэтому наличие “дружеского” и “прозрачного” пользовательского интерфейса заметно облегчит работу с такой ЭС [10].

Преимущества ЭС перед человеком-экспертом:

* Они не делают поспешных выводов, которые могли бы привести к нежелательным результатам,
* ЭС работают систематизировано, рассматривая вопрос во всех мельчайших деталях, выбирая наилучшую альтернативу из возможных,
* База знаний может быть достаточно большой и стабильной. После введения в машину один раз, знания сохраняются навсегда.
* У ЭС нет предубеждений.

Несмотря на свою привлекательность, у ЭС имеются и свои недостатки:

* ЭС могут оказаться доступными, только тем экспертам, которые создавали их базы знаний,
* После работы с экспертом, навыки ЭС могут не возрасти,
* Приведение знаний, полученных от эксперта к виду, обеспечивающему эффективную. реализацию,
* При отсутствии эксперта в определенной предметной области, применение ЭС невозможно.

Структура ЭС.

Принцип работы экспертной системы (Рисунок 1.1) состоит в том, что пользователь сообщает системе факты или другие знания, а ответ получает экспертный ответ. Ее структура содержит в себе множество компонентов такие как:

* база знаний;
* машина логического вывода;
* модуль обучения;
* модуль объяснения ответа;
* интеллектуальный интерфейс.

 Рисунок 1.1 Структура экспертной системы

Пользователь

Эксперт

Инженер по знаниям

Интеллектуальный интерфейс

Машина логического вывода

База знаний

Система получения знаний

Система объяснений

Экспертная система

Однако самыми важными компонентами отличающие любую ЭС от других систем являются: база знаний и машина логического вывода [16].

Машина логического вывода - программа выполняющая логический вывод из базы фактов и правил, называемой базой знаний. Является центральной частью экспертных систем. Факт - постоянно истинное утверждение. Правило - утверждение, в котором есть условие и результат.

 Основные элементы машины вывода:

* Интерпретатор - выполняет последовательно шаги для решения проблемы, основываясь на правилах БЗ.
* Планировщик - контролирует правильность выполнения шагов, анализируя эффективность с точки зрения приоритетов и остальных критериев, существующих в БЗ.

 Машина вывода может выполнять несколько операций:

* Проверка факта на его истинность
* Нахождение множества связей и знаний, при которых это правило будет выполняться.

Механизм вывода - процедура поиска решений задачи, использующая БЗ, которая выстраивает цепочку рассуждений и приводит к нужному результату. Он в свою очередь может делиться на два способа: прямой и обратный логический вывод.

Экспертная система с прямым логическим выводом использует начальные правила и по ним ищет подходящие правила для получения результата.

С другой стороны, система с обратным логическим выводом начинает отталкиваться от гипотезы, которую выдвигает пользователь и экспертная система пытается найти правила с помощью которых сможет объяснить ее.

Система объяснений - разъясняет пользователю как экспертная

система пришла к данному выводу. В процессе работы пользователь может спрашивать у системы о промежуточных результатах, если система не может объяснить человеку свои рассуждения, то её метод рассуждения является неудовлетворительным. Также ЭС может объяснить почему произошло именно так, а не иначе, какими правилами БЗ были заблокированы другие пути рассуждений.

Система приобретения знаний - делает процесс заполнения БЗ более автоматизированным. Именно она предоставляет базе знаний информацию в конкретной предметной области от эксперта.

Рабочая память - функционирует точно так же, как и ОЗУ в компьютере, получает и хранит исходные данные, а также промежуточные результаты по решаемой в данный момент задаче.

В рабочей памяти есть область, называемая доской объявлений, в ней описывается текущая проблема с применением исходных данных. На доске объявлений записываются гипотезы, оптимальный план решения проблемы, а также действия ожидающие выполнения, либо альтернативы решения.

 Система диалога - обеспечивает дружественный интерфейс и поддерживает общение пользователя и компьютера, может использоваться в процессе решения задачи, приобретения знаний и выдачи результатов. Общение может быть реализовано как с помощью естественного языка, так и с помощью графики [29].

Базы знаний

 База знаний - база данных, в которой содержится информация о знаниях и опыте человека-эксперта в определенной предметной области, а также правила логического вывода. БЗ может содержать в себе результаты решения прошлых задач, если находится в самообучающейся системе[8].

 База знаний неотъемлемый компонент ЭС, которые используют записи БЗ для нахождения способов решения проблем определенной предметной области.

 Самыми важными требованиями к записям, хранящимся в БЗ являются:

* Достоверность сведений.
* Релевантность информации при выводе.

Записи в БЗ - они же знания, делятся на два типа, алгоритмические и неалгоритмические.

Алгоритмические, еще называемые процедурными - это знания решающие конкретные задачи алгоритмы, функции вычислений.

Неалгоритмические знания - состоят из понятий. Понятия имеют имена, структуру, они связаны между собой и входят в некую систему понятий. Еще один вид не алгоритмических знаний - это связь понятий между собой, иначе называемая утверждением о их свойствах.

В базе знаний ЭС, могут быть некоторые необязательные особенности:

* Вывод новых знаний, основанных на уже записанных в БЗ.
* Доказательство ответа и объяснение её рассуждений по запросу.
* Контроль за правильной организацией знаний.
* Самообучение в определенной предметной области.

 Для получения знаний в БЗ используется два метода: приобретение знаний извне и их систематизация. При различных возможностях системы к обучению логического вывода, возможны разные виды приобретения информации. Представление знаний определяется внутри системы, поэтому форма информации, которую система может принять, зависит от способностей системы формализовывать информацию до знаний. Чем лучше в системе реализован логический вывод, тем меньше работы отводится человеку для подготовки информации понятной системе. На рисунке 1.2 представлена базовая система обработки знаний, она включает в себя БЗ и механизмы логических выводов.

Механизмы выводов обучающей системы

Механизмы выводов пользовательской системы

База знаний

Рисунок 1.2 Базовая структура обработки знаний

Методы получения знаний разделяются на несколько видов:

* Обучение без выводов - полученная информация используется в неизменном виде.
* Обучение на примерах - производится сбор фактов, с последующим их преобразованием и обобщением, затем они становятся знаниями.
* Параметрическое обучение - самая простая форма обучения по наблюдениям, определяет общий вид правила, которое в результате будет логическим выводом и корректирует параметры, входящие в него, в зависимости от данных.
* Обучение по индукции – обучение, использующее выводы высокого уровня, происходит путем обобщения всех имеющихся данных, получая общие правила.
* Обучение по аналогии - получение знаний, преобразуя уже имеющиеся похожие на те, которые должны получить.

Основная цель создания любой базы знаний - сократить затраты времени и труда на решение типовых задач.

Развитие технологий разработки экспертных систем

Продукционные правила.

Разработка экспертной системы является одним из самых сложных пунктов в ее создании и начинается далеко до того, как программист начнет писать код. Так как база знаний основана на правилах и для их разработки нужен специалист в области инженерии знаний для сбора информации у эксперта, который в будущем заполнит этими фактами БЗ.

Инженерия знаний - сбор информации и знаний от эксперта-человека или из других источников и последующее представление знаний в ЭС.

Сбором знаний для экспертной системы занимается инженер по знаниям. Он представляет их в явном виде для внесения в базу знаний. Далее эксперт проводит оценку ЭС и передает критические замечания инженеру по знаниям. Повторение процесса идет до тех пор, пока эксперт не одобрит качество знаний, внесенных в БЗ.

Сами правила в системе могут представляться разными способами, но самый широко распространенный из них являются правила в формате IF - THEN ещё по другому их называют продукционными правилами.

IF горит красный свет THEN стоять

Правило соответствует небольшой, модульной коллекции знаний, называемой фрагментом. Фрагменты организованы в свободной форме и снабжены связями, которые ведут к относящимся к ним фрагментам знаний.

Впервые продукционные системы, которые используют продукционные правила были использованы в символической логике Постом(Post). Он доказал, что любая система, будь она математической или логической, может быть оформлена в виде системы продукционных правил. Основная идея Поста была в том, что любая математическая или логическая система представляется как набор неких правил, с помощью которых одна строка (начальные данные) может быть преобразована различными алгоритмами в другую строку (выходные данные). Что можно отнести и к ЭС, где входные данные проходя длинный путь изменяются внутри машины логического вывода. Именно эта идея послужила “фундаментом” для отдельной части экспертных систем, но ей не хватало стратегии управления, которая бы позволила установить условия и рамки для обработки начальной информации. Система Поста позволяет применять правила к строкам в любой форме, поскольку отсутствует какая-либо спецификация, определяющая то, как должны применяться те или иные правила, но еще требовалось настроить порядок выполнения этих преобразований.

Марковский алгоритм.

Таким шагом было открытие, сделанное Марковым, определившее структуру управления для продукционных систем. Отсюда появился Марковский алгоритм - набор правил, выполняющийся в заданной последовательности.

Примером марковского алгоритма (Таблица 1), который переводит первую букву входной строки в конец выходной строки. Правила упорядочены с учетом того, что правило (1) имеет высший приоритет, правило (2) — приоритет, который следует за высшим приоритетом, и т.д. Приоритет правил задан в соответствии с порядком ввода правил, как показано ниже.

 (1) αxy → yαx

 (2) α →∧

 (3) ∧→α

Таблица 1.1 Пример марковского алгоритма

|  |  |  |
| --- | --- | --- |
| Правило | Успех(S), неудача(F) | Результирующая строка |
| 1 | F | ABC |
| 2 | F | ABC |
| 3 | S | αABC  |
| 1 | S | BαAC |
| 1 | S | BCαA |
| 1 | S | BCAα |
| 2 | S | BCA |

На данный момент эти алгоритмы больше используются в приложениях распознавания образов и называются - скрытые марковские модели (Hidden Markov Model — HMM)

Rete-алгоритм.

Марковский алгоритм дал хорошее развитие основ экспертных систем, однако, он все же плохо работал с очень большим количеством правил, что делает его неэффективным для реальных рабочих ЭС, которые могут иметь несколько сотен правил или даже тысячи.

Эта проблема была решена в 1979 году Чарльзом Л. Форги (Charles L. Forgy) из университета Карнеги–Меллона. Термин rete-алгоритм происходит от латинского слова rete (по-английски читается “рити”, а по-русски — “рете”), которое означает сеть.

Rete-алгоритм работает как сеть с динамической структурой данных, которая может меняться для ускоренной обработки запросов в целях оптимизации поиска.

Скорость работы алгоритма достигается объединения правил, тем самым уменьшая или совсем исключая избыточность. Сохранение частичных соответствий между фактами при их слиянии позволяет избежать вычисления(re-evaluation), что дает возможность Rete-алгоритму работать только с самими изменениями. Также в алгоритм уже было заложено высвобождение памяти при удалении фактов. Единственное что сделало его не таким популярным в те года это - требование больших вычислительных мощностей, которыми тогда компьютеры не обладали, однако сейчас данных факт не может являться значительной проблемой, так как развитие шагнуло далеко вперед.

Позже, в 1980-х годах была разработана новая версия алгоритма Rete-II. Улучшена общая производительность сети в том числе и хешированная память для больших массивов данных, добавлен алгоритм обратного вывода и его скорость значительно повышена в сравнении с предшественником. Автором новой версии является сам создатель, Чарльз Форги, но деталей алгоритма он разглашать не стал, известно только, что официально реализован в CLIPS/R2.

И последнее улучшение сделано в 2010 году снова Ч.Форги. Он разработал новое поколение алгоритма Rete, его характеристики в 500 раз превышают быстроту работы по сравнению с первой версией и еще в 10 раз быстрее Rete-II [1].

В результате создания быстрых алгоритмов согласования с шаблонами, таких как rete-алгоритм, был полностью подготовлен фундамент для развертывания практических приложений экспертных систем. Ниже приведены общие сведения о технологиях, которые образуют фундамент современных экспертных систем, основанных на правилах. (Рисунок 1.3)



Рисунок 1.3 Технологии, образующие фундамент современных экспертных систем, основанных на правилах

Некоторые технологии экспертных систем позволяют даже создавать системы, изучающие правила на примерах с помощью вывода правил методом индукции, в котором система создает правила на основе данных [21].

Современные экспертные системы

Принципы проектирования и разработки модели экспертной системы

 Не смотря на маленькую популярность экспертных систем и их небольшую известность в мире, они используются во многих сферах нашей жизни. Они могут отличаться между собой по многочисленным и разным параметрам, о чем будет рассказано далее, но при этом всех их объединяет общий алгоритм по их проектированию и разработке.

 Разработка экспертных систем включает в себя шесть этапов (Рисунок 2.1) это - идентификация, концептуализация, формализация, выполнение, тестирование и опытная эксплуатация.

Идентификация

Концептуализация

Формализация

Выполнение

Опытная эксплуатация

Тестирование

Начало

Требования

Понятия

Структуры знаний

Переформулирование

Реконструирование

Улучшение

ЭС

Рисунок 2.1 Этапы разработки ЭС

 Все этапы в свою очередь можно разбить на более мелкие задачи.

Идентификация.

Первый этап - идентификация, можно поделить на:

* определение задачи, которую будет решать экспертная система, а также целей, для которых она разрабатывается
* выбор какие нужны эксперты для нее, и кто будет пользоваться ЭС.

Ошибка на этом этапе может стоить больших потерь ресурсов, как денежных, так и временных, а всю работу что успели провести придется начинать с ранних этапов. Для реализации любых проектов следует учесть множество различных факторов таких как: трудозатраты, требуемое время для написания экспертной системы, подготовки базы знания для нее, стоимость разработки, и многое другое. Поэтому только при правильном руководстве возможно эффективное решение задачи.

На рисунке 2.2 краткая схема по руководству проектом. Как показано на рисунке руководство разделяется на три основных этапа: управление деятельностью, управление технической составляющей программного продукта и управление ресурсами.

Этот этап очень важен от этого зависит будет ли иметь система выигрыш. Данный этап показывает конкретную заинтересованность руководителя во вложенных ресурсах, времени и затраченного труда. Выигрыш может быть не только в виде денег, нередко экспертные системы создают для повышения эффективности и производительности людей, работающих на предприятии или ускоренном обучении новых сотрудников.



Рисунок 2.2 Задачи руководства проектированием

Концептуализация.

Следующий этап - этап концептуализации. Тут нужно сформировать требования к будущей экспертной системе для этого выполняются необходимые действия:

* глубокий анализ предметной области,
* выделение основных понятий и взаимосвязей между ними,
* определение методов выполнения задачи.

Если рассмотреть более подробно, то на этом этапе конкретизируют следующие особенности проекта по разработке: начальные данные, разбиение на подзадачи, способы решения задачи (стратегии и гипотезы), виды взаимосвязей между объектами предметной области и их тип, процессы, образующиеся в ходе поиска решений, качество и состав знаний, ограничения на процессы решения, состав данных для модуля объяснения.

Формализация

После этапа концептуализации следует формализация решаемой задачи. Это означает что на данном этапе выбираются инструментальные средства, с помощью которых будет описываться экспертная система - описание на формальном языке.

На данный момент существует достаточное количество программ и приложений, позволяющих как создавать экспертную систему с нуля, так и брать готовые отдельные компоненты необходимые для конечного продукта. Также перед тем как начинать разрабатывать ЭС нужно собрать знания у экспертов, которыми руководство решило сотрудничать. Тут не обойтись без инженеров по знаниям и им в свою очередь будут требоваться, а различные инструментальные средства для выполнения своей задачи, это могут быть как самые простые блокнот с ручкой, так и сложные программные решения по сбору информации и дальнейшей ее обработки.

Прототипирование.

Когда выбраны средства разработки и задачи четко сформулированы наступает этап выполнения - создание прототипа ЭС. Суть разработки прототипа состоит в том, что проверяются основные концепции в способах решения поставленной задачи, проверка работоспособности основных методов и адекватности функционала экспертной системы в целом.

Прототип тестируется несколько раз и перед каждым следующем происходит корректировка и доработка системы под необходимые задачи. Проверяются много различных компонентов системы:

* проверка адекватности идей, гипотез, представления знаний
* дорабатывается дружественный интерфейс
* проверяется работа прототипа при расширении базы знаний и списка решаемых задач
* добавление второстепенных модулей экспертной системы

Также собирают мнение пользователей прототипом о том, что нужно скорректировать или полностью поменять. Прототип, который прошел этап тестирования может рассматриваться как полноценной экспертной системой.

Опытная эксплуатация.

 Этап опытной эксплуатации тестирует на удобность экспертной системы для конечного пользователя. также оценивается полезность ЭС - способность в ходе диалога определять потребности пользователя, выявлять и устранять причины неправильной работы, а также удовлетворять указанные потребности пользователя. Удобство же предполагает легкое использование и взаимодействие ЭС (возможность долгое время работать с ее интерфейсом не уставая), гибкость (широкий спектр настроек под каждого пользователя), устойчивость системы к ошибкам (надежность системы) [19].

Тестирование.

 Последний этап - это этап тестирования. Во времени проведения данного этапа проверяется эффективность работы экспертной системы в целом. Инженером по знаниям создаются примеры для оценки ЭС и ее возможностей.

 Всего различают несколько причин неправильной работы системы:

* некорректные тестовые примеры
* ошибки при вводе-выводе данных
* неправильно составленные правила вывода
* выбраны не соответствующие управляющие стратегии

Первый пункт чаще всего является самой простой и очевидной причиной неправильной работы ЭС. Примеры могут вообще не относиться к предметной области решаемой задачи или же оказаться однотипными из-за чего будет невозможна всесторонняя проверка работоспособности экспертной системы при таком наборе тестовых примеров. Чтобы избежать данных ошибок, нужно создавать проверочные задачи для каждой отдельной небольшой области, которую должна решать ЭС.

Проблемы с вводом-выводом характеризуются тем, что результат работы ЭС отличается от ожидаемых результатов, так как в ходе диалога с системой были заданы неправильные вопросы или их количество было недостаточно. Или же наоборот, вопросы, задаваемые системой, были недостаточно однозначны и понятны вследствие чего пользователь внес лишние данные. Также проблемы возникают, когда для человека, использующего разработанную систему, не подходит или не удобен входной язык. Например, вместо того чтобы набирать какую-то информацию в систему проще отправить заранее подготовленный документ со всеми вопросами или требуется ввод графического изображения и т.д [13].

В выходных данных ошибки случаются из-за того, что информация, представленная системой может быть непонятна пользователю. Модуль объяснения решения выдает слишком много или наоборот слишком мало данных. Также причина может кроется в неудачной организации или упорядоченности заключений.

Наиболее часто встречаются ошибки в рассуждениях системы, касающиеся правил вывода. Чаще всего причина в не структурированности и отсутствия взаимосвязей сформулированных правил. Другая причина в самом составе правил, когда они противоречат друг другу, неполно составлены, то в таком случае предсказать конечный результат.

Также бывает к ошибкам функционирования экспертной системы приводят неверно выбранные управляющие стратегии. Изменение использующейся стратегии требуется, когда ЭС проводит анализ в порядке отличном от “естественного” для эксперта. Это в первую очередь влияет на скорость работы экспертной системы.

Данный этап может проводиться на всех предыдущих уровнях разработки, не только там, где используется непосредственно программная разработка экспертной системы, но и на более ранних этапах. Это переформулирование понятий и требований, переконструирование представления знаний в системе [7].

Классификация экспертных систем

В экспертных системах, выделяются четыре основных класса:

* Классифицирующие ЭС - создаются для решения задач на распознавание ситуаций. Метод решений в ЭС данного типа, является дедуктивный логический вывод.
* Доопределяющие ЭС - решают задачи с данными и знаниями, которые определены не полностью. Методами решения, являются вероятностный подход и нечеткая логика.
* Трансформирующие ЭС - являются динамическими, в которых знания преобразуются несколько раз, во время решения задачи. Используют различные способы обработки знаний.
* Мультиагентные ЭС - используют в качестве источника знаний несколько различных объектов, при этом интегрируя их.

 Далее приведена более подробная схема классификации ЭС (Рисунок 2.3).

 

Рисунок 2.3 Классификация экспертных систем

Классификация ЭС по назначению:

* Проектирование объектов по заданным характеристикам. Основная проблема в проектировании состоит в том, чтобы получить структурное описание объекта, которое соответствует всем правилам, заявленным для данного объекта.
* Планирование действий для объектов, выполняющих какие-либо функции, в ЭС данного типа используются модели поведения реальных объектов, для правильного вывода.
* Прогнозирование развития системы, основываясь на прошлых и текущих ее состояниях. Обычно используют динамическую модель, в которой данные подстраиваются под определенную ситуацию, на выходе получается вероятностная модель.
* Диагностика системы на наличие неисправностей, а также непрерывное отслеживание ее состояний. Важным компонентом является понимание структуры функционирования системы, для более точных параметров диагностирования.
* Обучение пользователя, основываясь на ошибках прошлых операций. Система находит правильные решения и говорил о них пользователю, а также находит слабые места в знаниях, которым она обучается.
* Интерпретация данных с определением их смысла. Система анализирует данные на основе, уже существующих, корректных.

Классификация ЭС по методам представления знаний:

* Традиционные (автономные) ЭС используют модели знаний, полученные опытным путем, которые могут быть научно не обоснованы. Работают в режиме консультации с пользователем.
* Гибридные (интегрированные) ЭС используют всевозможные методы.

Классификация ЭС по степени сложности.

* Поверхностные ЭС выводят знания, основываясь на правиле “если-то”. Вывод происходит незамедлительно и цепочка правил непрерывна.
* Глубинные ЭС при обрыве цепочки правил могут определить, какие действия должны быть выполнены, чтобы продолжить решение определенной задачи. Также к ним относят сложные предметные области, в которых запись правил на естественном языке, занимает более одной трети страницы.

Классификация ЭС по предметной области.

Существуют экспертные системы по военному делу, геологии, инженерному делу, информатике, космической технике, математике, медицине, метеорологии, промышленности, сельскому хозяйству, управлению, физике, химии, электронике, юриспруденции и т.д. Однако сложность таких систем, их большая узконаправленность и требуемая большая вычислительная мощность сделало ЭС менее распространенными чем нейронные сети.

 Классификация ЭС по динамичности:

* Статические - исходные данные, описывающие предметную область, не меняются во времени.
* Динамические - исходные данные меняются в процессе решения задачи. Архитектура динамической ЭС построена на работе с датчиками и контроллерами объектов, в реальном времени и постоянной переработкой поступающих данных.

Анализ существующих экспертных систем

Пример психолингвистической ЭС.

Система ВААЛ, работа над которой велась с 1992 по 2002 гг., позволяет прогнозировать эффект неосознаваемого воздействия текстов на массовую аудиторию, анализировать тексты с точки зрения такого воздействия, составлять тексты с заданным вектором воздействия, выявлять личностно-психологические качества авторов текста, проводить углубленный контент-анализ текстов и делать многое другое.

*Области возможного применения:*

* Составление текстов выступлений с заранее заданными характеристиками воздействия на потенциальную аудиторию.
* Активное формирование эмоционального отношения к политическому деятелю со стороны различных социальных групп.
* Составление эмоционально окрашенных рекламных статей.
* Поиск наиболее удачных названий и торговых марок.
* Психо-игипнотерапия.
* Неявное психологическое тестирование и экспресс-диагностика.
* Создание легких в усвоении учебных материалов.
* Научные исследования в области психолингвистики и смежных с нею дисциплинах.
* Журналистика и другие сферы деятельности, использующие в качестве инструмента СЛОВО.
* Социологические и социолингвистические исследования.
* Информационные войны.
* Контент-анализ текстов.
* Мониторинг СМИ.

Система позволяет:

* Оценивать неосознаваемое эмоциональное воздействие фонетической структуры текстов и отдельных слов на подсознание человека.
* Генерировать слова с заданными фоносемантическими характеристиками.
* Оценивать неосознаваемое эмоциональное воздействие фонетической структуры текстов на подсознание человека.
* Задавать характеристики желаемого воздействия и целенаправленно корректировать тексты по выбранным параметрам в целях достижения необходимого эффекта воздействия.
* Оценивать звуко-цветовые характеристики слов и текстов.
* Производить словарный анализ текстов.
* Осуществлять полноценный контент-анализ текста по большому числу специально составленных встроенных категорий и категорий, задаваемых самим пользователем.
* Производить выделение тем, затрагиваемых в текстах, и осуществлять на основе этого автоматическую категоризацию.
* Производить эмоционально-лексический анализ текстов.
* Настраиваться на различные социальные и профессиональные группы людей, которые могут быть выделены по используемой ими лексике.
* Производить вторичный анализ данных путем их визуализации, факторного и корреляционного анализа.

Реализация*.*

Система реализована в виде набора DLL-библиотек, которые подключаются к наиболее популярному текстовому процессору Word for Windows. Просто в главном меню появляется новый пункт. Такой способ реализации позволяет сохранить для пользователя привычную удобную среду создания документов и максимально облегчает освоение системы ВААЛ.

Пример инструментально-экспертной системы.

Инструментально-экспертная система APM WinMachine представляет собой энциклопедию по машиностроению, включающую инструменты и программы для автоматизированного расчета и проектирования деталей машин, механизмов, элементов конструкций и узлов. Кроме этого, она имеет современные визуальные средства, встроенные базы данных, необходимую информационную базу знаний, разветвленную систему подсказок и учебник электронного типа по азам проектирования машин.

APM WinMachine содержит современные, эффективные и надежные программы для расчета:

* Энергетических и кинематических параметров;
* Прочности, жесткости и устойчивости;
* Выносливости при переменных режимах нагружения;
* Вероятности, надежности и износостойкости;
* Динамических характеристик.

Кроме того, в APM WinMachine имеется набор инструментальных средств расчета и анализа. Эти средства, а также проектируемые детали, в зависимости от назначения разделены на подсистемы (модули), которые могут функционировать как в составе системы, так и самостоятельно:

* APM Graph - плоский чертежно-графический редактор с инструментом расчета размерных цепей.
* APM Studio - модуль твердотельного и поверхностного моделирования.
* APM Mechanical Data - модуль хранения стандартных деталей и узлов, расчетных данных по машиностроению.
* APM Material Data - модуль хранения и редактирования параметров материалов.
* APM Construction Data - модуль хранения графической информации по стандартным деталям и элементам строительных конструкций.
* APM Drive - модуль автоматизированного проектирования привода вращательного движения различной структуры.
* APM Trans - модуль проектирования механических передач вращения.
* APM Shaft - модуль проектирования валов и осей.
* APM Bear - модуль проектирования подшипниковых узлов качения с учетом их класса точности.
* APM Joint - модуль проектирования соединения элементов машин.
* APM Spring - модуль проектирования упругих элементов машин.
* APM Cam - модуль проектирования кулачковых механизмов.
* APM Slider - модуль проектирования плоских механизмов различной структуры.
* APM Plain - модуль расчета подшипников скольжения.
* APM Screw - модуль счета идеальных винтовых передач.
* APM Structure3D - модуль счета напряженно-деформированного состояния, устойчивости, вынужденных колебаний деталей и конструкций с использованием метода конечных элементов.
* APM Beam - модуль счета балочных элементов конструкций.
* APM FEM2D - модуль конечно-элементного анализа плоских деталей.
* APM Truss - модуль счета ферменных конструкций.
* MDM (Основы проектирования машин) - электронный учебник.

Каждый модуль предоставляет пользователю встроенную среду, которая в общем случае включает в себя:

* специализированный графический редактор;
* встроенные базы данных;
* полный цикл вычислений;
* разнообразные средства представления результатов расчета.

Пример экспертной системы в рекламной деятельности и PR.

"EXPO: 1001 Рекламоноситель" - В 2000-м году консалтинговой фирмой "ТРИЗ-ШАНС" (Санкт-Петербург) была начата работа над компьютерной программой-консультантом, которая позволила бы специалисту по рекламе:

* за ограниченное время получить гамму решений задач,
* отобрать лучшие решения для дальнейшего доработки,
* выяснить адреса поставщиков рекламоносителей

Данная экспертная система помогает при:

* разработке PR- и промо-акций, рекламных кампаний;
* поиске новых идей для продвижения продукции;
* поиске новых, необычных рекламоносителей и их поставщиков;
* поиске идей для выставочных стендов;
* анализе полученной системы решений.

Преимущества экспертной системы “EXPO: 1001 Рекламоноситель”:

* развернутая система Help, содержащая, в том числе, словарь терминов;
* компактный Учебник выставочного дела, содержащий разделы: "Как выбрать выставку", "Цели участия в выставке", "Как уменьшить плату за выставку" и многие другие;
* адреса интернет-ресурсов выставочной тематики;
* более 350 способов "Отстройки от Конкурента";
* более 2200 эффективных решений в области маркетинга, рекламы, PR и выставочной деятельности;
* база данных поставщиков рекламоносителей.

База данных программы построена на основе анализа нескольких сотен книг по PR, рекламе и выставочной деятельности. А также нескольких сотен публикаций на эту же тему, собранных за последние 5 лет.

Если программой пользуется новичок, то без чтения массы книг, посещения платных многомесячных курсов и т.п., он за несколько минут получит возможные варианты решения своей задачи, из которых - на основании здравого смысла и отпущенного бюджета - уже сможет отобрать наиболее подходящие. Начальное освоение программы занимает 1-2 часа.

Если же пользователем является специалист, у которого с годами выработался "свой стиль" (а порой и "штампы"), то программа позволит ему взглянуть на свою работу с новой точки зрения.

Естественно, некоторые решения, выдаваемые программой, специалисту известны. Однако программа выдает целые подборки решений и позволяет ему сосредоточиться не на воспоминаниях, а на отборе лучших вариантов.

##

ВЫБОР ТЕХНИЧЕСКИХ СРЕДСТВ И СОЗДАНИЕ БАЗЫ ЗНАНИЙ

Общие сведения о CLIPS

CLIPS - программная среда для разработки экспертных систем. Синтаксис и название предложены Чарльзом Форги. Первые версии CLIPS разрабатывались с [1984 года](https://ru.wikipedia.org/wiki/1984_%D0%B3%D0%BE%D0%B4) в Космическом центре Джонсона, NASA (как альтернатива существовавшей тогда системе [ART\*Inference](https://ru.wikipedia.org/w/index.php?title=ART*Inference&action=edit&redlink=1)), пока в начале [1990-х](https://ru.wikipedia.org/wiki/1990-%D0%B5) не было приостановлено финансирование, и NASA вынудили купить коммерческие продукты.

CLIPS является продукционной системой. Реализация вывода использует алгоритм Rete.

Основная идея состоит в представлении знаний в виде такой формы:

Правило1:
 ЕСЛИ
 (выполняются условия1)
 ТОГДА
 (выполнить действия1)
 Правило2:
 ЕСЛИ
 (выполняются условия2)
 ТОГДА
 (выполнить действия2)
 ...

Такое представление близко к человеческому мышлению и отличается от программ, написанных на традиционных алгоритмических языках, где действия упорядочены и выполняются строго, придерживаясь алгоритма.

CLIPS является одной из наиболее широко используемых инструментальных сред для разработки экспертных систем благодаря своей скорости, эффективности и бесплатности. Являясь общедоступной, она до сих пор обновляется и поддерживается своим изначальным автором, Гэри Райли.

CLIPS включает полноценный объектно-ориентированный язык *COOL* для написания экспертных систем. Хотя она написана на языке Си, её интерфейс намного ближе к языку программирования LISP. Расширения можно создавать на языке Си, кроме того, можно интегрировать CLIPS в программы на языке Си.

CLIPS создан для применения в качестве языка прямого логического вывода (forward chaining) и в своей оригинальной версии не поддерживает обратного вывода (backward chaining).

Как и другие экспертные системы, CLIPS имеет дело с правилами и фактами.

Факты.

Информация, с помощью которой экспертная система делает [логический вывод](https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B2%D1%8B%D0%B2%D0%BE%D0%B4) называется фактами. В CLIPS есть 2 вида фактов: упорядоченные и шаблонные. Шаблонные факты имеют шаблон, из конструкции deftemplate. Упорядоченные не имеют явной конструкции deftemplate, однако она может быть. Шаблонный факт напоминает структуру в языке C, поля называются слотами и объявляются с помощью конструкции slot.

Факты размещаются в рабочей памяти. Размещаются в рабочую память командой assert. Например, следующая команда

(assert (cars))
добавит в рабочую память упорядоченный факт *cars*.

CLIPS не допускает помещения в рабочую память фактов с одинаковыми значениями слотов, хотя, это можно разрешить соответствующей настройкой.

Правила.

Знания выбранной области представляются в CLIPS в виде правил, которые имеют следующую структуру:

(условия) {синонимы: антецеденты в логике,
 левая часть - LHS в терминах CLIPS}
 =>
 (действия) {синонимы: консеквенты в логике,
 правая часть - RHS в терминах CLIPS}
 Левая часть правила - это условие, при котором оно срабатывает, а правая часть - это те действия, которые должны выполниться если выполняется условие. Знак => специальный символ, разделяющий LHS и RHS.

Правила объявляются с помощью команды defrule. Пример правила:

(defrule search-black-toyota
 (cars (model "toyota") (color Black))
 =>
 (printout t "Есть черный toyota!" crlf)
 )
 Данное правило активируется тогда, когда в рабочей памяти появится факт с атрибутами (model «toyota») и (color Black).

Активация правила не означает его выполнение. Активация правила — это помещение правила в рабочий список правил или agenda в CLIPS.

Чтобы активированные правила выполнялись нужно выполнить команду (run).

Стратегии разрешения конфликтов

Человек иногда не может задать полные условия, которые бы удовлетворяли действительности. Существует легенда, согласно которой Диоген Синопский на определение Платона «Человек есть животное о двух ногах, лишённое перьев», ощипал курицу и принес к нему в школу, объявив: «Вот платоновский человек!» На что Платон к своему определению вынужден был добавить «…и с широкими ногтями». Когда в базе знаний появляются правила, которые удовлетворяют фактам, но выполняют противоположные действия, то возникает конфликт правил.

Например, есть два правила:

(Если человек толкнул другого человека — наказать человека за хулиганство)

(Если человек толкнул другого человека, на которого ехал грузовик - наградить человека за спасение жизни)

Это два правила будут между собой конфликтовать. Первое правило общее и оно всегда активируется, при активации второго. Но первым должно выполниться второе правило. В CLIPS есть несколько стратегий для разрешения данных конфликтов. Но даже если нет возможности выбрать подходящую стратегию для всех случаев, то можно указать приоритеты для правил. Правила с наибольшим приоритетом будут выполняться первыми.

Различные факты могут сделать правило применимым. Применимое правило затем заносится в рабочую память (assert). Факты и правила создаются объявлением [23].

Описание возможных состояний антропоморфного персонажа

 В нашем случае рассматривается первый этап исследований и разработок, когда решаются задачи по первичному освоению инструментария логического программирования и изучению подходов к разработке экспертной системы. Поэтому для примера рассматриваются движения человека типа ударов и иных телодвижений борьбы. Причем берется приближенное описание физики таких операций, так как на данном этапе физика и механика движений не имеет значения, а основное внимание уделяется описанию знаний и формированию правил логического вывода.

Для описания человека в программе были выбраны следующие состояния:

* Движение (двигается и в каких направлениях или стоит).
* Постановка блока.
* Нанесение удара (прямой или тяжелый).

Также выделены основные части тела:

* Голова.
* Туловище.
* Правая рука.
* Левая рука.
* Правая нога.
* Левая нога.

Как уже говорилось ранее, такое упрощенное описание возможно потому что управление системой идет в созданном игровом мире, где не требуется точное описание модели человека на данном этапе. В дальнейшем же такая структура позволяет усложнять и приводить новые параметры, приближенные к настоящему человеку, при этом, не требуя создавать все с нуля, а улучшать текущую модель и расширяя базу знаний, которая используется.

Описание прямого удара руками (Джеб)

Прямой удар – самый простой удар и в ту же очередь самый важный в изучении многих видов единоборств. Он является базовым ударом, с которого и начинается мир тренировок боевыми искусствами. Возможно, используя боксёрские перчатки или другие подобные, получится скрасить некоторые неточности в исполнении приёма и не получить травму. Но, выходя на более новый уровень, обязательно придётся соблюдать правильную технику. Хороший прямой удар – это долгое время практики и тренировок.

Попробуем разобраться в некоторых его нюансах. Как и любой другой удар, он должен проходить в определённой последовательности. Разобьём его на несколько этапов.

* Выполнение техники прямого удара начинается с ноги. Нога двигается немного вперёд и подворачивается в сторону удара. Причем при ударе с правой движение идёт правой ногой, и при ударе левой рукой – двигаете левую ногу. Это движение позволяет усилить энергию удара, путём её передачи от ног через бёдра в плечо.
* Обязательно обеспечьте свободный выброс руки. Не напрягайте руку при ударе. В полёте кулак сжат не полностью, т.е. расслаблен. Сжимать кулак следует перед непосредственным ударом. В конце движения возможен доворот руки или другими словами можете ударить сверху вниз. Плечо таким образом должно прижаться к голове, защищая от дальнейших действий соперника.
* Ударил – сразу уходи в защиту. После выполнения удара нужно сразу возвращать руку, не задерживая ее не ни на мгновение, так же быстро, как наносил удар. Возвращение руки должно происходить по кратчайшей траектории в положение, способное обеспечить выполнение как защитных, так и атакующих действий.

Описание тяжелого удара руками (Свинг).

 Основные составляющие сильного удара:

* Быстрота движения для всех типов ударов (сюда входит и удар свинг) в первую очередь зависит от способностей человеческих мышц быстро сокращаться и расслабляться. Для этого мозг должен отдавать молниеносные приказы своим «солдатам» мышцам. В определенной степени все эти функции являются врожденными для каждого человека, однако, благодаря долгим, а главное правильным тренировкам, можно существенно увеличит быстроту движений. Как известно, изнурительные силовые упражнения делают человеческие мышцы малоподвижными, именно по этой причине тяжелоатлеты и борцы обладают достаточно слабым ударом;
* Волевая концентрация (сосредоточение энергии и воли на ударе). Данное правило не относиться к так называемым финтам (обманным ударам с целью сбить с толку противника), или ударам, которые противник, скорее всего, отобьёт;
* Техника удара (использование веса тела и самых сильных его мышц).

Все удары опытных боксеров (в том числе и удар свинг) сопровождаются разнообразными наклонами или выпрямлениями корпуса, переносом тяжести тела на разные ноги, толчками ли выпрямлением ног и т.д. Следуя одно за другим, эти движения сливаются в единый рывок, остриём которого служит кулак спортсмена. Именно по этой причине боксеры годами трудятся, вырабатывая хорошую технику удара, позволяющую выбрать правильное положение предплечья и кулака, чтобы максимально точно и эффективно попадать по противнику.

Описание постановки блока.

 В боевых искусствах, кроме нападения на соперника и нанесения ему ударов большую роль играет защита. Главная защита — это постановка блоков. Как ставить блоки в боксе знает любой боксёр, но новичкам, и людям присматривающимся к этому виду спорта будет интересно узнать, что есть блоки:

* Жёсткие.
* Плавающие.
* Подвижные.

 Разновидности блоков:

* Жёсткий блок, это когда боксёр выставляет руку, согнутую в локте на удар противника, тем самым отражая, блокируя его. Такой блок своей целью имеет недопущение нанесения удара. Но и не предусматривает контрудара. Он служит для предотвращения удара соперники и принятия более выигрышного положения на ринге. Жёсткий блок позволяет перемещаться боксёру.
* Плавающий блок подразумевает блокировку удара рукой и продолжение удара противника по вектору силы. Продолжить удар противника, перевести его в другую плоскость или просто отвести от себя. Такие блоки более эффективны чем жёсткие, но их применение требует определённых навыков.
* Подвижный. Здесь блок ставится одновременно с нанесением контрудара. Этот приём хорош в том случае, если боксёр одинаково хорошо владеет обеими руками.

Есть самая простая техника как ставить блоки в боксе, она появилась ещё в Англии, когда бокс только появлялся. Эта техника называется "перчатка на перчатку", а суть её в том, чтобы выбросить прямую руку на удар противника. Рука должна быть жёсткой и выбрасывать её нужно с максимальной инерцией. Движение должно закончиться при соприкосновении с перчаткой другого боксёра. Этот приём способен остановить атаку. Но после его проведения необходимо уходить либо в контратаку, либо принимать полную защиту, потому что после неудавшейся атаки противник снова будет нападать.

Блоки следует ставить и тогда, когда противник атакует корпус, но здесь блок будет короче - согнутой рукой стоит блокировать удар и сразу же менять положение своего тела в пространстве.

Описание ударов ногами.

Удары ногами были взяты уже из другого единоборства, а именно - кунг-фу. Далее будут представлены два удара: прямой - ким Ким тиеу кыок (прямой удар ноги) и тяжелый - Нгик дао кыок (круговой удар).

Описание удара Ким тиеу кыок:

* удар проводится из стойки, когда ноги на ширине плеч, колени чуть согнуты, таз вверх, носки внутрь;
* удар идет опорной ногой из стойки;
* нанесение удара обязательно начинается с поднятия колена так, чтобы прикрыть пах и уже затем производится выброс ноги вперед;
* завершение удара характеризуется тем, что носок при ударе вытянут вперед;
* после нанесения удара возврат ноги в стойку идет по той же схеме - сгибается нога в колене, прикрывая пах, затем возвращается назад, перенося вес обратно на ногу, которой нанесен удар.

 Описание удара Нгик дао кыок. Особенности.

* начинается из той же стойки что и прямой;
* разворот впереди стоящей ноги на 90 градусов во внутрь;
* круговое движение с согнутым коленом на уровни паха как в прямом ударе ким тиеу кыок;
* по завершению разворота на 360 градусов, выкидывания ноги вперед с вытянутым носком.

Особенности состоят в том, что удар имеет дополнительную силу за счет инерции.

Общее описание базы знаний

База знаний представляет собой набор правил логического вывода вида:

(defrule determine-move ""

(logical (start))

=>

(assert (UI-state (display StartQuestion)

(relation-asserted move)

(response Stay)

(valid-answers Stay Forward Back Left Right))))

Где defrule, добавляет в систему новое правило, в данном случае это move. Logical - используется для логической связи с некоторым объектом, так-как приведенное правило начальное, то logical (start) означает, что с данного правила начинается выполнение программы. Assert – добавляет в рабочую память новый факт, так-как база знаний готовится специально под соединение с Java, то используется UI-state, которая прикрепляет факт к интерфейсу. Relation-asserted – отвечает за связь конструкции с правилом, заданным defrule. Response – это изначальная позиция в интерфейсе, на которой стоит флаг выбора, в данном случае это может быть любой из верных ответов. В строке после valid-answers, задаются верные ответы для этого правила, в данном случае это:

* Stay (оставаться на месте).
* Forward (идти вперед).
* Back (идти назад).
* Left (идти влево).
* Right (идти вправо).

Далее представлена конструкция вывода ответа на определенное правило:

(defrule action-choice ""

(logical (action-choice Block) (move-choiсe Stay)

=>

(assert (UI-state (display Stay-and-Block)

(state final))))

В данном случае defrule отвечает за связь фактов из первого блока правил и правил с ответом для вывода из второго. Чтобы выбрать, на основании чего будет происходить выбор верного конечного правила, используется logical.

В данном случае logical (action-choice block) (move-choice stay), показывает, что для выбора именно этого правила, должны произойти два действия, такие как:

* Block (блокирование удара), который описан в первом блоке программы, в правиле под названием action-choice.
* Stay (оставаться на месте), также описано в первом блоке программы, в правиле move-choice.

Assert (UI-state (display Stay-and-Block) – используется для того, чтобы в интерфейс выводить верный ответ для пользователя. State – отвечает за стадию правила, в данном случае это завершающее действие, описанное final.

ЗАКЛЮЧЕНИЕ

Существующие на данный момент проблемы принятия решений будут актуальны. Тренд нашего времени это - автоматизация различных сфер деятельности человека. Сейчас есть множество различных систем, помогающих в принятии решении или же полностью выполняющих эту функцию самостоятельно, например, различные диагностические программы, нейронные сети и т.д.

В ходе данной дипломной работы решается вопрос не только о создании базы знаний, но и создания экспертной системы, для управления виртуальным антропоморфным персонажем.

Было проанализировано множество различной литературы, как технической, так и научной; рассмотрено чем являются экспертные системы и в каких сферах их можно использовать. Также рассмотрены их различные классификации и области применения. По завершению данного этапа была выбрана область смешанных единоборств, потому что в этом направлении работ очень мало и есть куда развиваться.

Работа велась совместно с Шайковым Валерием, который в свою очередь, на основе разработанной мною базе знаний, создал прототип экспертной системы, позволяющий смоделировать игровой пошаговый бой с упрощенными правилами. За одного бойца предоставляется управление пользователю, который, отвечая на вопросы, задаваемые программой выбирает действия для бойца под его контролем. Управление вторым персонажем отдано системе, против которой игрок и ведет данный бой.

Таким образом цель и задачи дипломной работы достигнуты – на основе проведенного анализа движений человека в смешанных единоборствах, спроектированы правила вывода, а также создана база знаний.

В дальнейшем развитие данного проекта может идти в нескольких направлениях:

* адаптация всей системы под управление андроидными роботами, ориентированных на боевую тематику;
* внедрение системы в управление ИИ в играх и улучшению разнообразия игровых механик.

Представленная сфера использования экспертных систем еще далеко не полностью изучена и представляет собой большой интерес для дальнейшего изучения, а нарастающая снова популярность к данным программным продуктам только подтверждает это.

СПИСОК ЛИТЕРАТУРЫ

1. Forgy C. L. Rete: A fast algorithm for the many pattern/many object pattern match problem //Artificial intelligence. – 1982. – Т. 19. – №. 1. – С. 17-37.
2. Friedman-Hill E. JESS in Action. – Greenwich, CT: Manning, 2003. – Т. 46.
3. Giarratano J. C., Riley G. Expert systems. – PWS Publishing Co., 1998.
4. IntelliJ I. The Most Intelligent Java IDE. – 2007.
5. Jet Brains I. Intellij idea //On-line at www. intellij. com. – 2011.
6. Zhou R. et al. Application of CLIPS expert system to malware detection system //Computational Intelligence and Security, 2008. CIS'08. International Conference on. – IEEE, 2008. – Т. 1. – С. 309-314.
7. Банушкина Н. А. Особенности разработки экспертных систем в зависимости от класса объектов //Известия Алтайского государственного университета. – 2011. – №. 1-2.
8. Гаврилова Т. А., Хорошевский В. Ф. Базы знаний интеллектуальных систем. – 2000.
9. Гиззатуллина А. Р. Языки программирования С# И Java. СРАВНЕНИЕ //Международная научно-практическая конференция «Интеграция науки и образования». Международный центр инновационных исследований «Омега сайнс. – 2014. – С. 137-139.
10. Джарратано Д., Райли Г. Экспертные системы: принципы разработки и программирование, 4-е издание. – Издательский дом Вильямс, 2007.ч
11. Джексон П. Введение в экспертные системы //Вильямс, 2001. – Вильямс, 2001, 2015.
12. Дзюба С. М., Дякин В. Н. Разработка интеллектуальных информационных систем с использованием CLIPS.
13. Микулич Л. И. Проблемы создания экспертных систем //Теория и модели-знаний (Теория и практика создания систем искусственного интеллекта).–Тарту: ТГУ. – 1985. – С. 87-114.
14. Морозова О. В., Григорьев А. В., Grigor'yev A. V. Анализ существующих инструментальных оболочек, основанных на продукционных моделях. – 2012.
15. Ноутон П., Шилдт Г. Java 2. Наиболее полное руководство //С-Пб.: BHV.–2001.–1072 с. – 2001.
16. Попов Э. В. Экспертные системы. – 1987.
17. Размахнина А. Н., Баженов Р. И. О применении экспертных систем в различных областях //Постулат. – 2017. – №. 1.
18. Рязанов М. А. Java интерфейс для экспертных систем на основе технологии clips //Известия Алтайского государственного университета. – 2009. – №. 1.
19. Таунсенд К., Фохт Д. Проектирование и программная реализация экспертных систем на персональных ЭВМ. – Финансы и статистика, 1990.
20. Трофимов В. База данных+ CLIPS= база знаний //Компьютеры+ программы. – 2003. – №. 10. – С. 56.
21. Уотермен Д. Руководство по экспертным системам: Пер. с англ. – Мир, 1989.
22. Хабибуллин И. Ш. Java 7 (4-е изд.). – БХВ-Петербург, 2012.
23. Частиков А. П., Гаврилова Т. А., Белов Д. Л. Разработка экспертных систем. Среда CLIPS. – СПб. : БХВ-Петербург, 2003.
24. Шилдт Г. Java: руководство для начинающих. 4-е изд. – 2009.
25. Package javax.swing [Электронный ресурс] //Режим доступа: http://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html
26. Всё о Java и SQL [Электронный ресурс] //Режим доступа: http://java-online.ru
27. Руководство по языку программирования Java. [Электронный ресурс] //Режим доступа: https://metanit.com/java/tutorial/
28. Среда CLIPS Мадорский Константин ВМК МГУ 2013. [Электронный ресурс] //Режим доступа: http://al.cs.msu.ru/
29. Хабаров С. П. Экспертные системы [Электронный ресурс] //Режим доступа: http://www. habarov. spb. ru/new\_es/index. htm.

ПРИЛОЖЕНИЕ А. Структура правил вывода в интерфейс.

(defrule ask-question

(declare (salience 5))

(UI-state (id ?id))

?f <- (state-list (sequence $?s&:(not (member$ ?id ?s))))

=>

(modify ?f (current ?id)

(sequence ?id ?s))

(halt))

(defrule handle-next-no-change-none-middle-of-chain

(declare (salience 10))

?f1 <- (next ?id)

?f2 <- (state-list (current ?id) (sequence $? ?nid ?id $?))

=>

(retract ?f1)

(modify ?f2 (current ?nid))

(halt))

(defrule handle-next-response-none-end-of-chain

(declare (salience 10))

?f <- (next ?id)

(state-list (sequence ?id $?))

(UI-state (id ?id)

(relation-asserted ?relation))

=>

(retract ?f)

(assert (add-response ?id)))

(defrule handle-next-no-change-middle-of-chain

(declare (salience 10))

?f1 <- (next ?id ?response)

?f2 <- (state-list (current ?id) (sequence $? ?nid ?id $?))

(UI-state (id ?id) (response ?response))

=>

(retract ?f1)

(modify ?f2 (current ?nid))

(halt))

(defrule handle-next-change-middle-of-chain

(declare (salience 10))

(next ?id ?response)

?f1 <- (state-list (current ?id) (sequence ?nid $?b ?id $?e))

(UI-state (id ?id) (response ~?response))

?f2 <- (UI-state (id ?nid))

=>

(modify ?f1 (sequence ?b ?id ?e))

(retract ?f2))

(defrule handle-next-response-end-of-chain

(declare (salience 10))

?f1 <- (next ?id ?response)

(state-list (sequence ?id $?))

?f2 <- (UI-state (id ?id)

(response ?expected)

(relation-asserted ?relation))

=>

(retract ?f1)

(if (neq ?response ?expected)

then

(modify ?f2 (response ?response)))

(assert (add-response ?id ?response)))

(defrule handle-add-response

(declare (salience 10))

(logical (UI-state (id ?id)

(relation-asserted ?relation)))

?f1 <- (add-response ?id ?response)

=>

(str-assert (str-cat "(" ?relation " " ?response ")"))

(retract ?f1))

(defrule handle-add-response-none

(declare (salience 10))

(logical (UI-state (id ?id)

(relation-asserted ?relation)))

?f1 <- (add-response ?id)

=>

(str-assert (str-cat "(" ?relation ")"))

(retract ?f1))

(defrule handle-prev

(declare (salience 10))

?f1 <- (prev ?id)

?f2 <- (state-list (sequence $?b ?id ?p $?e))

=>

(retract ?f1)

(modify ?f2 (current ?p))

(halt))

ПРИЛОЖЕНИЕ Б. Блок правил.

(defrule determine-move ""

(logical (start))

=>

(assert (UI-state (display StartQuestion)

(relation-asserted move)

(response Stay)

(valid-answers Stay Forward Back Left Right))))

(defrule determine-action-choice ""

(logical (move Stay))

=>

(assert (UI-state (display A-CQuestion)

(relation-asserted action-choice)

(response Hit)

(valid-answers Block Hit))))

(defrule determine-action-choice ""

(logical (move Forward))

=>

(assert (UI-state (display A-CQuestion)

(relation-asserted action-choice)

(response Hit)

(valid-answers Block Hit))))

(defrule determine-action-choice ""

(logical (move Back))

=>

(assert (UI-state (display A-CQuestion)

(relation-asserted action-choice)

(response Hit)

(valid-answers Block Hit))))

(defrule determine-action-choice ""

(logical (move Left))

=>

(assert (UI-state (display A-CQuestion)

(relation-asserted action-choice)

(response Hit)

(valid-answers Block Hit))))

(defrule determine-action-choice ""

(logical (move Right))

=>

(assert (UI-state (display A-CQuestion)

(relation-asserted action-choice)

(response Hit)

(valid-answers Block Hit))))

defrule determine-hit-choice ""

(logical (action-choice Hit))

=>

assert (UI-state (display H-CQuestion)

(relation-asserted hit-choice)

(response Lhit)

(valid-answers Lhit Hhit))))

defrule determine-Lhit-than ""

(logical (hit-choice Lhit))

=>

assert (UI-state (display LH-TQuestion)

(relation-asserted Lhit-than)

(response LH)

(valid-answers LH RH LL RL))))

(defrule determine-Lhit-where ""

logical (Lhit-than LH) (Lhit-than RH) (Lhit-than LL) (Lhit-than RL))

=>

(assert (UI-state (display LH-WQuestion)

(relation-asserted Lhit-where)

(response LH)

(valid-answers LH RH LL RL Head Body))))

defrule determine-Hhit-than ""

(logical (hit-choice Hhit))

=>

assert (UI-state (display HH-TQuestion)

(relation-asserted Hhit-than)

(response LH)

(valid-answers LH RH LL RL))))

(defrule determine-Hhit-where ""

logical (Lhit-than LH) (Lhit-than RH) (Lhit-than LL) (Lhit-than RL))

=>

(assert (UI-state (display HH-WQuestion)

(relation-asserted Hhit-where)

(response LH)

(valid-answers LH RH LL RL Head Body))))