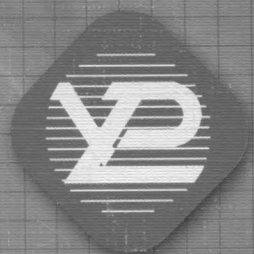
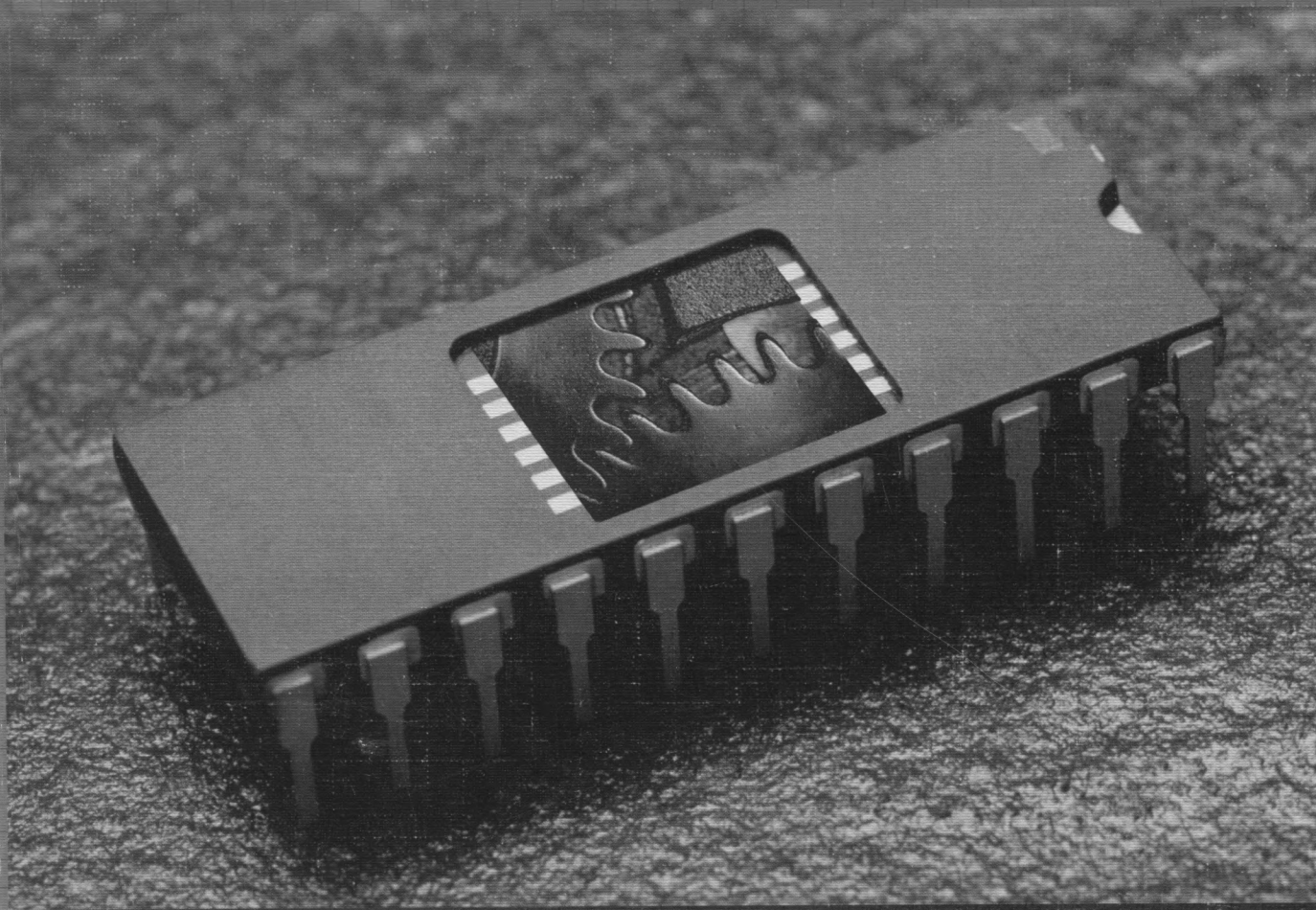


АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ УПРАВЛЕНИЯ И ПРИБОРЫ АВТОМАТИКИ



'2001
124

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ

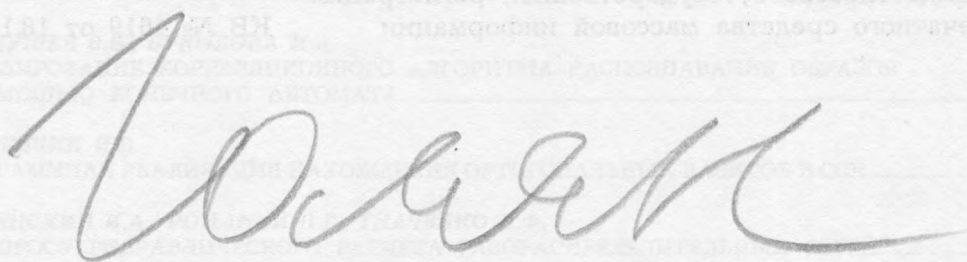
ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ
УНИВЕРСИТЕТ РАДИОЭЛЕКТРОНИКИ

АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ УПРАВЛЕНИЯ И ПРИБОРЫ АВТОМАТИКИ

Всеукраинский межведомственный
научно-технический сборник

Основен в 1965 г.

Выпуск 124

A large, stylized handwritten signature in dark ink, likely belonging to a member of the editorial board or a reviewer, is written across the middle of the page.

Харьков

2003

СОДЕРЖАНИЕ

НАТАЛУХА Ю.В., АФАНАСЬЕВ В.А., КЛИМОВА Н.П. ОБ ОДНОМ КРИТЕРИИ ПРОВЕРКИ ЛИНЕЙНОЙ НЕЗАВИСИМОСТИ ФУНКЦИЙ.....	4
ЛЕЩИНСКАЯ Е.Л. ИДЕНТИФИКАЦИЯ СЕМАНТИЧЕСКИХ ОШИБОК ПРИ СИНТАКСИЧЕСКОМ АНАЛИЗЕ.....	8
ТИХОНОВ В.А., ЗУБКОВ О.В., СИДОРОВ Г.И. ГЕНЕРИРОВАНИЕ КВАДРАТУРНЫХ СОСТАВЛЯЮЩИХ СЛУЧАЙНОГО УЗКОПОЛОСНОГО АНАЛИТИЧЕСКОГО СИГНАЛА НА НЕСУЩЕЙ ЧАСТОТЕ ПО ЗАДАННЫМ ПАРАМЕТРАМ СПЕКТРАЛЬНОЙ ПЛОТНОСТИ МОЩНОСТИ.....	18
ДОРОВСКОЙ В.А. ЭНТРОПИЙНАЯ МОДЕЛЬ ДИАЛОГОВОГО ПОВЕДЕНИЯ ПАРТНЕРОВ.....	23
ГОЛЯН В.В. VHDL-МОДЕЛЬ ПРОЕКТИРОВАНИЯ ПРЕОБРАЗОВАТЕЛЕЙ КОДОВ.....	27
КАНЦЕДАЛ С.А., КОСТИКОВА М.В. О ДВУХ АЛГОРИТМАХ ДЛЯ РЕШЕНИЯ ОБЩЕЙ ЗАДАЧИ ТЕОРИИ РАСПИСАНИЙ.....	31
СОРОЧАН М.В. МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ ПРИ ПОСТРОЕНИИ ТРЕБОВАНИЙ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ.....	37
ГВОЗДИНСКИЙ А.Н., ЯКИМОВА Н.А. ЛОГИКО-АЛГЕБРАИЧЕСКИЕ МОДЕЛИ СИСТЕМ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ.....	43
АЛИПОВ Н.В., АЛИПОВ И.Н., ХИЛЬ М.И., РЕВЕЗЮК Л.Н. ПРИМЕРЫ ПОСТРОЕНИЯ ПОСЛЕДОВАТЕЛЬНЫХ АЛГОРИТМОВ ПОИСКА ТОЧКИ С ХАРАКТЕРНЫМ ПРИЗНАКОМ, ПОМЕХОУСТОЙЧИВЫХ К СИММЕТРИЧНЫМ НЕРЕГУЛЯРНЫМ ВИРТУАЛЬНЫМ ПОСЛЕДОВАТЕЛЬНОСТЯМ.....	47
ДЮВКО Г.Ф., ХАРЧЕНКО С.Л. КОМПОНЕНТНАЯ ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ ДЛЯ ОПИСАНИЯ МОДЕЛЕЙ ЦИФРОВЫХ УСТРОЙСТВ.....	55
СИНОТИН А.М. ИССЛЕДОВАНИЕ ВЛИЯНИЯ ЭЛЕМЕНТОВ РАДИОЭЛЕКТРОННОГО АППАРАТА НА ЭФФЕКТИВНУЮ ТЕПЛОПРОВОДНОСТЬ В ОСНОВНЫХ НАПРАВЛЕНИЯХ.....	61
СИТНИКОВ Д.Э., ВИЛЬЧИНСКАЯ О.С., КРАВЕЦ Н.С. АЛГЕБРАИЧЕСКИЙ МЕТОД НАХОЖДЕНИЯ АППРОКСИМАЦИЙ ПРИБЛИЖЕННЫХ МНОЖЕСТВ И ФОРМИРОВАНИЕ НА ИХ ОСНОВЕ ЛОГИЧЕСКИХ ПРАВИЛ.....	66
ДЮВКО Г.Ф., ПОЛУНИН Р.А. РАЗРАБОТКА БАЗ ЗНАНИЙ ДЛЯ ПРОВЕДЕНИЯ ЛОГИЧЕСКОГО КОНТРОЛЯ В КОМПЬЮТЕРНЫХ ОБУЧАЮЩИХ СИСТЕМАХ.....	72
КЛИМУШЕВ В.В., БУКОЛОВА М.А. МОДЕЛИРОВАНИЕ КОРРЕЛЯЦИОННОГО АЛГОРИТМА РАСПОЗНАВАНИЯ ОБРАЗОВ С ПОМОЩЬЮ КОНЕЧНОГО АВТОМАТА.....	80
КУПРЕЙЧИК И.В. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ НАХОЖДЕНИЯ ОРТОГОНАЛЬНЫХ БАЗИСОВ В СОК.....	87
БОЖИНСКИЙ И.А., РОМАНИЙ П.Г., ТКАЧЕНКО В.Ф. К ВОПРОСУ ГИДРАВЛИЧЕСКОГО РАСЧЕТА ГАЗОРАСПРЕДЕЛИТЕЛЬНЫХ СЕТЕЙ.....	91
ПОЛОНСКИЙ А.Д. ИНВАРИАНТНАЯ АЛГЕБРА РАНГОВЫХ ПРЕДИКАТОВ ДЛЯ СИНТЕЗА КЛАССИФИКАТОРОВ АНАЛОГОВЫХ СИГНАЛОВ.....	99
ГОЛОВАШИЧ С.А., ПОНОМАРЕВ Д.В. ПОТОЧНЫЙ ШИФР VMGL И ОСОБЕННОСТИ ЕГО ПРОГРАММНОЙ РЕАЛИЗАЦИИ.....	107
АНПИЛГОВ Е.М., ДЗЮНДЗЮК В.В., СТЫЦЕНКО Т.Е. ИНТЕРПРЕТАЦИЯ И АНАЛИЗ НАСЛЕДУЕМЫХ РАЗЛИЧНЫХ ФАКТОРОВ НА ОПЕРАЦИЯХ ИЗГОТОВЛЕНИЯ СИСТЕМЫ.....	113
РЕФЕРАТИ.....	116

РАЗРАБОТКА БАЗ ЗНАНИЙ ДЛЯ ПРОВЕДЕНИЯ ЛОГИЧЕСКОГО КОНТРОЛЯ В КОМПЬЮТЕРНЫХ ОБУЧАЮЩИХ СИСТЕМАХ

Рассматриваются вопросы построения обобщенной системы логического контроля, которая автоматически отслеживает и контролирует путь решения студента. Используется подход на основе автоматического решения выданных студенту заданий. Система контроля полностью или частично содержит в себе знания, которые студент использует для решения.

1. Введение

Рассмотрим вопросы применения трехуровневой модели знаний, грамматик и логического вывода к построению подсистем логического контроля в интеллектуальных системах компьютерного обучения (СКО). Многие существующие СКО используют базу знаний (БЗ) об ошибках. Это означает, что система контроля, помимо БЗ для самостоятельного решения контролируемых задач, имеет базу знаний для поиска ошибок, которые не могут быть выведены из БЗ решения задач. Поскольку такая общая постановка затрагивает более широкое поле проблем, таких как моделирование обучаемого, здесь рассматривается лишь часть ее, касающаяся контроля правильности решения относительно эталона(-ов). Если в эталонной БЗ системы будут знания касательно принятия решений, выбора субалгоритмов, возвратов и т.п., то с ее помощью возможно обнаруживать ошибки обучаемого с любой интересующей разработчиков БЗ точностью. Таким образом, принят подход к организации процесса контроля путем: а) сравнения введенных результатов с автоматически полученными; б) сравнения решений, принимаемых при автоматическом выводе, с распознанными решениями обучаемого. Понятие «точности», или «детализации», с которой проводится контроль знаний, в нашем случае обусловлено необходимостью использовать для наполнения базы знаний предметно-ориентированные формальные примитивы, имеющие четко определенную формальную структуру и позволяющие к процессу автоматического решения задачи подключить функции ввода и контроля пути решения обучаемого. Детализация БЗ определяет размер и содержимое множества отклонений, которые может обнаружить система контроля (СК) [1]. Система контроля будет моделировать либо процедурный, либо декларативный метод «мышления», а также комбинировать их, что и подразумевается понятием «декларативно-процедурный подход» [2].

2. Актуальность, постановка задачи и цели исследования

Система логического контроля сильно зависит от знаний, на проверку которых она рассчитана, и строится именно на основе структуры этих знаний (эта структура зачастую используется в качестве модели обучаемого). Поэтому для автоматизации построения блоков логического контроля знаний очень важен вопрос применения достаточно обобщенных и в то же время эффективных подсистем построения БЗ и управления ими. Поставленные в исследовании задачи приведены ниже.

1) Исследовать предметную область задач контроля знаний в компьютерных обучающих системах, подходы к организации логического контроля знаний в СКО, в том числе связь с задачей моделирования обучаемого.

2) Исследовать существующие подходы к постановке задачи логического контроля и к реализации его блоков, а также возможности использования для этих целей грамматик и синтаксических анализаторов, предложить обобщенный подход к построению блоков логического контроля. В рамках разработки

подхода к построению блоков логического контроля предложить подход к построению и формат представления БЗ системы контроля.

3. Функционирование СК и наполнение базы знаний

3.1. Общие замечания

Будем говорить о некоторой обобщенной модели системы контроля, предоставляющей средства для построения СК под конкретные задачи. В описываемой модели СК на входе контроля должна получить:

- 1) входные данные конкретной задачи в заранее определенных терминах (т.е. в терминах БЗ 1-го уровня);
- 2) описание эталонного алгоритма для решения задачи (БЗ 2-го уровня);
- 3) входной поток, состоящий из элементов-описаний шагов решения.

При подготовке к проведению контроля СК строит рабочую БЗ, т.е. 3-й уровень БЗ в принятой 3-уровневой модели [2]. Из получаемой от пользователя информации, используя 3-й уровень БЗ, СК строит пользовательский путь решения, а из эталонного описания алгоритма по входным данным задачи строит собственный, эталонный путь решения, которое сравнивается (синхронно или же после ввода пользователем всего решения) с пользовательским. Этап генерирования пользовательского пути решения потенциально может служить местом некорректного толкования реальных намерений обучаемого, поэтому пользовательский интерфейс, с помощью которого генерируется поток данных о решении, должен быть хорошо продуман (см. матрицу анализа возможных ошибок в [1]). Входной поток полностью формируется конкретной системой контроля, использующей платформу. Ядру могут также понадобиться вспомогательные средства для генерирования пути решения по описанию эталонного алгоритма.

Можно различать алгоритмы, порождающие один и тот же путь решения независимо от входных данных (различия только в результатах применения умений), и алгоритмы, порождающие различные пути решения. Для алгоритмов, порождающих одинаковый путь решения, процесс отслеживания хода решения значительно упрощается. Однако фиксированный путь решения дают только простейшие алгоритмы, не содержащие циклов и условных переходов (в процедурном случае) и декларативные программы, в которых один предикат может присутствовать в голове лишь одной клаузы (в прологическом, декларативном случае). Поэтому такое деление не имеет практической ценности для систем логического контроля знаний, и здесь не рассматривается.

3.2. Понятие пути решения

Задача контроля в используемой нами модели состоит из двух частей: контроль конкретных результатов и контроль правильности принятия решений (в том числе выбора стратегий). Результаты (например, численных расчетов) могут быть получены системой контроля с помощью внешних по отношению к ее БЗ элементов. Эти внешние элементы выполняют роль «умений» обучаемого, в то время как содержимое БЗ – роль «знаний». Разница между умениями и знаниями полагается здесь лишь в том, что умения как таковые не содержатся в БЗ платформы – только ссылки на них, и потому машина автоматического вывода может контролировать их лишь по выходным результатам. А знания (навыки и стратегии) содержатся в БЗ платформы. Поэтому выбор места, где провести черту между умениями и знаниями, зависит лишь от человека-составителя БЗ. В ходе отслеживания решения система контроля ведет протокол-«путь решения»:

1. Если контролируемые знания описаны в процедурном стиле (блок-схема, записанная на служебном языке), то элементы пути соответствуют блокам «Если-То-Иначе» и блокам выполнения действий, т.е. путь решения выглядит как нить, обходящая граф блок-схемы. В пути решения (если наложить его на

ориентированный граф блок-схемы) могут быть циклы, если в процессе решения некоторая ветвь блок-схемы выполняется несколько раз. Итак, элементы пути решения отражают протокол принятия решений – применения знаний. Кроме того, с каждым элементом пути решения могут быть связаны один или более результатов применения умений, т.е. результаты расчетов, преобразований и т.п., выполненные внешними элементами.

2. Если контролируемые знания описаны в декларативном стиле (дизъюнкты), то понимать термины «стратегия», «выбор стратегии», «принятие решения» для целей последовательного их отслеживания во входном потоке необходимо немного иначе. Однако если посмотреть на декларативную часть БЗ как на ориентированный граф, в узлах которого находятся элементы «программы» для некоторого интерпретатора, моделирующего логический вывод, то становится очевидной возможность представления пути решения в виде нити, проходящей через эти узлы. Таким образом, для хранения транслированных декларативных описаний в БЗ возможно использовать точно такую же структуру данных, как и в случае с процедурным описанием знаний.

Теперь видно, что путем представления базы знаний в виде ориентированного графа есть возможность дать разработчику БЗ средства для ввода декларативных и процедурных описаний алгоритмов, причем хранить эти знания в БЗ одинаковым образом. Рассмотрим понятие пути решения применительно к этому графу.

Фактически путей решения необходимо вести два: один для эталонного сравнения, выполняемого по базе знаний и умений, и другой – как результат отслеживания действий пользователя. Первый путь решения генерируется без каких-либо проблем, однако процесс создания второго реализовать гораздо сложнее. Дело в том, что пользователь чаще всего вводит лишь *результаты* своих действий. Обучающая система в большинстве случаев не предоставляет обучаемому набор вариантов для *принятия решения* (т.е. варианты результатов применения знаний), а только варианты конечных результатов применения умений (например, при тестировании). Так что для каждого элемента пользовательского пути решения система обычно получает лишь одну его часть – результат применения умений, промежуточный результат. По этой информации необходимо определить, *какое знание* применил обучаемый, т.е. найти в графе БЗ нужный элемент, присоединить его к пользовательскому пути решения, а затем *проверить*, не произошло ли отклонение от правильного (одного из правильных) пути решения. Чаще всего, чем больше таких промежуточных результатов, тем с большей точностью возможно определить, какие знания обучаемый применил для принятия решения и перехода в текущую точку графа. Если система не смогла идентифицировать эти знания из-за неопределенности между несколькими вариантами, то она может задать дополнительные вопросы. Однако если знания не идентифицированы по причине их отсутствия в БЗ, то система может лишь выдать ошибку. Согласно [1], реализация поиска нужного элемента БЗ может быть источником ошибок класса $\Delta(Ribk, RsRibk)$. Упомянутая проверка результатов расчетов (применения умений) является проверкой на наличие ошибок класса $\Delta(Rsblp, bsp)$ и сама может быть источником ошибок класса $\Delta(blpr, Rsblpr)$. Наконец, проверка на отклонение от хода решения (после попытки поиска в БЗ, соответствующих введенному промежуточному результату) является проверкой на наличие ошибок класса $\Delta(RsRibk, Rsbk)$ или $\Delta(RsRibk, Rsbk)$.

3.3. Способ представления онтологической сетки и элементов предметной области, формальная структура онтоэлемента

База знаний 1-го уровня состоит из нескольких частей [2]:

1. Сетка понятий онтологических элементов (онтологическая составляющая). Это простой список строковых имен.

2. Грамматики, связываемые с каждым элементом, одни – для разбора структуры входных данных, другие – для семантической проверки целостности структуры полученных деревьев и наличия во входном дереве для каждого онтологического элемента всех его формальных составляющих, а также механизм для их извлечения из древовидного представления онтологического элемента в памяти (синтаксическая составляющая).

3. Внешняя составляющая – пользовательские средства загрузки и разбора входных данных, подключаемые к механизму разбора платформы.

Предполагаем, что каждый онтологический элемент имеет некую формальную структуру, которая позволяет из конкретного объекта (соответствующего этому элементу) извлекать и интерпретировать его части, значимые в текущем контексте. Например, в СК решения квадратных уравнений онтоэлемент «квадратное уравнение» может иметь три формальных составляющих: коэффициенты А, В, С. Далее, при описании знаний 2-го уровня, в операциях с объектами-деревьями можно непосредственно использовать уже формальную структуру соответствующих им онтоэлементов (см. пункт 3.2.2 в [2]).

Структура онтоэлемента задается с помощью грамматик формальной структуры (ГФС). Механизм извлечения данных из объекта использует эти грамматики для интерпретации узлов дерева и целых поддеревьев как элементов формальной структуры. Таким образом, ГФС не является грамматикой в обычном смысле этого слова, это лишь способ описания действий, необходимых для извлечения значений формальной структуры онтоэлементов из деревьев. С одним онтоэлементом может быть связано множество ГФС, предназначенных для обработки синтаксических деревьев различного рода. Одна и та же грамматика может быть привязана ко многим онтоэлементам. Поэтому имя этого онтоэлемента не может однозначно идентифицировать грамматику в БЗ платформы. Ясно, что при таком подходе один и тот же объект-дерево (или даже текстовая строка) может быть интерпретирован совершенно по-разному, в зависимости от применяемой к нему ГФС. Поэтому в БЗ 2-го уровня можно использовать «преобразования» объектов, например обращение к одному и тому же дереву сначала как к «уравнению», затем как к «квадратному уравнению», и т.д. Эти преобразования сводятся к созданию Структур. «Структура» – это результат применения ГФС к исходному объекту-дереву, в простом случае это контейнер для именованных (согласно формальной структуре онтоэлемента) значений. Структуры будут использованы как в декларативных, так и в процедурных описаниях. Вероятно, ГФС удобнее всего рассчитывать не просто на обработку неформализованных данных (таких как символьный текст решения), но на обработку деревьев. Поэтому видится такой подход к наполнению БЗ 1-го уровня:

1. Разработчиком БЗ выделяется поле задач, решение которых нужно контролировать.

2. Составляется единый язык (не обязательно текстовый) для описания конструкций в задачах из нашего поля. Конструкции на этом языке обучаемый будет использовать для ввода своего решения.

3. Формируется БЗ онтоэлементов, грамматик и ГФС для первичного синтаксического разбора всех конструкций.

4. Для конкретных задач контроля в БЗ вводятся дополнительные онтоэлементы и ГФС, с использованием уже существующих.

Вид связи между ГФС и онтоэлементами (один-ко-многим, один-к-одному и т.п.) определяется требованиями к реализации платформы.

3.4. Средства для декларативного описания знаний

Есть задачи, для контроля решения которых система должна иметь средства для декларативного описания знаний и логического вывода. В платформе составлен (на основе «чистого» Пролога [4]) и используется механизм, состоящий из таких компонент:

1. БЗ предикатов – аналогия прологовских. Предикаты могут быть встроенными или подключаться из внешних компонент. Кроме того, возможно с предикатами связывать (по именам) подпрограммы из процедурной части БЗ.

2. БЗ клауз и механизм логического вывода, согласно Прологу. С клаузами также могут быть связаны внешние элементы, вызываемые при срабатывании клаузы.

3. Создание «Структур» – фиктивных объектов-деревьев, которые содержат лишь значения элементов формальной структуры соответствующих онтоэлементов. Отметим, что это не-прологовское понимание термина «структура». Разница между обычным объектом и Структурой состоит в том, что обычный объект-дерево может содержать целое синтаксическое дерево, из которого грамматика формальной структуры извлекает значения, в то время как Структура содержит только значения формальной структуры онтоэлемента, и к ней нельзя применять грамматику формальной структуры. Структуры могут быть созданы при выполнении как выражений, входящих в БЗ, так и внешних элементов.

В качестве значений в предикатах и для переменных в нашей платформе используются исключительно объекты-деревья. Соответственно, возникают специфические сложности при реализации пролог-машины и ее оптимизации. Есть различные варианты применения этого механизма. Рассмотрим пошаговую проверку правильности логического вывода. Заметим, что это не обязательно логические задачи. Это, например, может быть проверка правильности выбора алгоритма решения некоторой задачи.

1. В БЗ 2-го уровня заносится программа (именованная группа клауз вместе с описаниями используемых предикатов), которая реализует «эталонную» логику решения, причем все ее варианты нужны для целей контроля. Следует строить клаузы так, чтобы при их интерпретации машина имитировала требуемое от обучаемого поведение, а не только корректно вычисляла результаты задачи. Сам по себе контроль реализуется с помощью отслеживания вызовов примитивов.

2. К декларативным примитивам прикрепляются внешние элементы-собственные средства СК.

3. На вход логического вывода подается поток сведений о процессе решения. Ядро, проводя вывод по эталонным знаниям в БЗ, одновременно отслеживает решение студента и на каждом шаге вызывает СК для принятия решения о дальнейших действиях. Заметим, что такой механизм может быть использован как при синхронном контроле, так и при контроле после ввода всех этапов решения.

3.5. Средства для процедурного описания знаний

Платформа должна предоставить для описания алгоритмов в процедурном стиле такие средства, как служебный язык (обозначим его К1) и транслятор с него в графовую структуру. Необходимы возможности вызова подпрограмм из декларативной части БЗ и подключения внешних элементов. К1-процедура – это блок-схема алгоритма решения некоторой задачи, описанная в текстовом виде. Система использует К1-процедуры для самостоятельного решения задачи и для контроля пути решения обучаемого. Как и декларативные программы, набор К1-процедур – это описание эталонного решения, по которому будет отслеживаться путь решения обучаемого. К1 должен дать возможность описать выбор из нескольких возможных стратегий решения конкретной

задачи, т.е. сказать системе о необходимости контролировать сам процесс принятия обучаемым решения о выборе стратегии. Это может быть реализовано с помощью одного из формальных примитивов.

4. Формат представления знаний системы контроля

4.1. Общие замечания

Для студента ход контроля знаний обычно заключается в последовательном вводе ответов на задаваемые системой вопросы, или просто в последовательном вводе промежуточных результатов решения. Эти «вопросы» могут быть именно текстовыми вопросами, могут быть полями для ввода результатов некоторых расчетов и т.п. Система должна реконструировать пользовательский путь решения от некоторого текущего места в алгоритме решения до предполагаемого текущего места по значениям текущих результатов. Типичным подходом к выполнению этого реконструирования в существующих разработках является использование экспертной системы или собственной пролог-машины в совокупности со специализированной базой знаний, рассчитанной на конкретную задачу. Фактически это и есть «база ошибок» [1, 3], или «неправильных» путей решения, по которым производится поиск. Но поскольку предлагается дать системе знания для выполнения решения самостоятельно, необходимо определить, какие «подходы» к мышлению она будет моделировать при выполнении автоматических вычислений. Для описания знаний, которые разработчик БЗ хочет заложить в платформу, необходимо использовать наборы формальных примитивов, позволяющие выразить либо декларативную, либо процедурную семантику знаний. Платформа должна транслировать знания из описаний на двух языках (для процедурного и декларативного подходов) в унифицированную графовую структуру, распознаваемую встроенной машиной-интерпретатором. Еще раз отметим, что детализация, точность процесса контроля, которой можно добиться с помощью такого подхода, полностью зависит от состава используемых наборов формальных примитивов.

4.2. Формальные примитивы для конструирования графа БЗ

Для «выполнения» знаний, заложенных в полученном графе, необходима машина-интерпретатор узлов этого графа. Каждый примитив, используемый для трансляции знаний, имеет формальную структуру и может предоставлять специфические возможности для организации контроля. Машина-интерпретатор при «выполнении» примитивов уведомляет систему контроля, которая может потребовать ввести очередную порцию информации, самостоятельно сделать выводы о правильности решения или выполнить другие действия. Список примитивов для описания декларативных знаний должен быть достаточным, чтобы описать Прологовские клаузы, вызовы целей, унификацию, отсечения и т.п. Аналогично, процедурные примитивы – это средства для представления понятия «цикл», условных и безусловных переходов и т.п. Помимо примитивов, платформа должна предоставить вспомогательные средства для наполнения БЗ и для подключения внешних элементов, к которым производятся обращения из примитивов БЗ. Например, если в графе БЗ содержатся обращения к предикату $A(X, Y)$, то необходимо где-то описать объявление заголовка этого предиката (аналогично тому, как это делается в Прологе). Домен переменной будет использован машиной-интерпретатором при выполнении, например, такого графа:

«BEGIN

GENERATOR VAR=X» -> «PROVE PREDICATE=A, VAR(1)=X, VAR(2)=5»

-> «END GENERATOR»

для выполнения перебора всех возможных значений X. Обозначения «BEGIN GENERATOR», «END GENERATOR» использованы для описания наименования, начала и конца области, объединяющей подмножество узлов графа. Если

в примитив PROVE включить возможность уведомления системы контроля в момент присвоения значений переменных, то она при получении такого уведомления сможет запросить из входного потока значение X, введенное обучаемым, и сравнить его с автоматически сгенерированным. Далее, если в PROVE включить возможность уведомления СК в момент *выхода* из доказательства субцели, то она сможет проконтролировать (сделать какие-либо выводы) результаты выполнения этого доказательства, например, новые значения переменных или успешность выполнения доказательства. В табл. 1 приведен список примитивов, которые можно использовать для трансляции декларативных описаний с комментариями к их структуре.

Таблица 1. Список декларативных примитивов

№п/п	Наименование	Элементы формальной структуры примитива и комментарии
1	CLAUSE (area)	Содержит примитивы CLAUSE_HEAD и CLAUSE_TAIL. Ограничивает область видимости для переменных клаузы
2	CLAUSE_HEAD	Синоним PREDICATE
3	CLAUSE_TAIL (area)	Синоним PROVE_LIST. Если пуст, то клауза является фактом
4	GENERATOR (area)	VAR_NAME, VAR_DOMAIN – имя переменной и ее домена, для автоматического перебора значений. Может включать в себя либо PROVE_LIST, либо GENERATOR Это искусственная конструкция, ограничивающая область клаузы, в которой происходит перебор значений переменных
5	PREDICATE	HEAD_NAME – имя предиката, стоящего в голове клаузы. VAR_LIST – набор элементов VAR, представляющий собой перечень формальных аргументов головы клаузы. Если HEAD_NAME пустое, то это клауза-цель. HEAD_NAME может быть пустым только если это CLAUSE_HEAD. Может обращаться к внешнему элементу
6	PROVE_LIST (area)	Обозначение для набора примитивов GENERATOR и PROVE
7	PROVE	TYPE – «FAIL» - спец.цель FAIL. «CUT» - отсечение, «STOP» - останов машины, «P» - предикат. PREDICATE – предикат-субцель. Используется, если TYPE равно «P». Указание интерпретатору перейти к доказательству субцели (в том числе специальной цели FAIL) или выполнить отсечение
8	VAR_LIST	Обозначение для набора примитивов VAR и STRUCT
9	STRUCT	STRUCT_NAME – имя Структуры. VAR_LIST – аргументы. STRUCT_VALUE – значение, связанное со Структурой. Структура и, вероятно, функция. Будет определено более точно в реализации
10	VAR	VAR_NAME – имя переменной. VAR_DOMAIN – имя домена переменной. VAR_VALUE – значение переменной (объект-дерево). Переменная либо в списке аргументов предиката, либо в списке аргументов Структуры

В табл. 1 и в приведенной ниже табл. 2 пометка «(area)», сопутствующая названию примитива, означает, что этот примитив по смыслу является некой областью, в рамках которой находятся другие примитивы. Например «GENERATOR» здесь понимается как область (подмножество субцелей) хвоста клаузы, поскольку согласно процедурной семантике пролог-машины, при выполнении перебора значений переменной многократно выполняется доказательство только тех субцелей клаузы, которые стоят правее первой субцели, содержащей эту переменную. Соответственно «область» генератора будет включать в себя все эти субцели.

В табл. 2 приведен список примитивов, которые можно использовать для трансляции процедурных описаний, с комментариями к их структуре.

Таблица 2. Список процедурных примитивов

№п/п	Наименование	Элементы формальной структуры примитива и комментарии
1	SUB (area)	SUB_NAME – текстовое имя процедурной подпрограммы. VAR_LIST – формальные аргументы. BLOCK – блок операторов. Это процедурная подпрограмма. Служит для структурирования процедурных описаний. Ограничивает область видимости аргументов
2	BLOCK (area)	Ограничивает область, содержащую примитивы FOR, CALL, EXIT
3	FOR	INIT_BLOCK – BLOCK инициализации цикла. CONDITION – условие, проверяемое перед каждым шагом цикла. ITERATION – BLOCK, выполняемый после каждого шага цикла. Цикл FOR, семантика которого позаимствована из языка С
4	CALL	SUB_NAME – имя подпрограммы. VAR_LIST – переменные. Вызов другой подпрограммы, в том числе внешнего элемента
5	EXIT	RESULT – (VAR) результат. Немедленный возврат управления
6	VAR_LIST	Обозначение для набора примитивов VAR
7	VAR	VAR_NAME – имя переменной. VAR_DOMAIN – имя домена переменной. VAR_VALUE – значение переменной (объект-дерево). Переменная. Понятие «объявление переменных» не используем, переменная объявляется неявно по имени и домену присваиваемого значения, при первом ее использовании

4.3. Ввод информации и распознавание пути решения обучаемого

Ввод данных должен быть организован средствами каждой конкретной СК, использующей данный подход. Можно выделить такие режимы работы системы контроля:

1. Контроль под управлением механизма автоматического решения. В моменты срабатывания формальных примитивов СК запрашивает сведения о решении обучаемого и проверяет, не отклонился ли он от «эталонного» пути и нет ли ошибок в его результатах. Если найдено отклонение, СК может попытаться либо скорректировать путь автоматического решения (если есть альтернативные пути решения), либо потребовать повторить попытку ввода.

2. Контроль под управлением пользовательского интерфейса СК. СК изначально предполагает ввод решения в виде неких типизированных блоков данных. При вводе очередных данных СК распознает тип введенной информации и, отталкиваясь от некоторого «текущего» узла (примитива) в графе БЗ, пытается найти в нем другой узел, в котором, судя по введенной информации, находится обучаемый.

5. Заключение

На основе имеющихся литературных источников и программных реализаций систем контроля знаний предложен подход к построению блока логического контроля, состоящий в:

- 1) использовании 3-уровневой базы знаний;
- 2) достаточности БЗ для вывода из нее всех возможных вариантов решения, подлежащих контролю;
- 3) использовании в качестве промежуточного средства для наполнения БЗ предметно-ориентированных языков.

Также рассмотрены вопросы реконструирования пути решения по имеющемуся графу формальных примитивов. Отличия рассмотренного подхода от базовых [1, 3] состоят в следующем:

- 1) не используется база знаний об ошибках или о правильных ответах;
- 2) понятие пути решения непосредственно связано с графовой структурой БЗ;
- 3) способность системы контроля реконструировать путь решения с помощью автоматически задаваемых вопросов.

Список литературы: 1. *Pierre Dillenbourg, John Self. A Framework for Learner Modelling. // AAI/AI-ED Technical Report No.74, Department of Computing, University of Lancaster, 1992.* 2. *Полунин Р.А. Разработка декларативно-процедурного подхода к построению подсистемы контроля в компьютерной обучающей системе на основе трехкомпонентной модели знаний // Тезисы доклада на 7-й международный молодежный форум «Радиоэлектроника и молодежь в XXI веке». Харьков, 2003. с. 365.* 3. *Michael Bernard Twidale M.A. The Use of Explicit Intermediate Representations in Intelligent Tutoring Systems // Department of Computing, University of Lancaster, July 1989.* 4. *Братко И. Программирование на языке Пролог для искусственного интеллекта: Пер. с англ. М.: Мир, 1990. 560 с.*

Поступила в редколлегию 29.05.2003

Дюбко Геннадий Федорович, канд. техн. наук, проф. каф. ПО ЭВМ ХНУРЭ. Научные интересы: теория формальных систем, экспертные системы, автоматизированный вывод на знаниях. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 433053.

Полунин Роман Алексеевич, студент группы ПОАС-98-1 ХНУРЭ. Научные интересы: управление и использование баз данных и знаний. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, e-mail: roman@csltd.com.ua. ICQ#: 76666965

УДК 519.713

В.Б. КЛИМУШЕВ, М.А. БУКОЛОВА

МОДЕЛИРОВАНИЕ КОРРЕЛЯЦИОННОГО АЛГОРИТМА РАСПОЗНАВАНИЯ ОБРАЗОВ С ПОМОЩЬЮ КОНЕЧНОГО АВТОМАТА

Рассматриваются вопросы синтеза быстродействующих вычислительных структур для случая ограничений, связанных с физической рекуррентностью входных сигналов. Описывается корреляционный алгоритм распознавания образов конечным автоматом (КА) на конкретном примере. Проводится сравнительный анализ формальной грамматики и КА, который позволяет предложить алгоритм процесса сопоставления полного вывода представления моделируемого алгоритма в формальной грамматике и КА. Устанавливается, что синтезированный конечный автомат моделирует все виды возможной параллельности.

1. Актуальность исследования

Определяется необходимость разработки актуальных вопросов синтеза математических моделей, описывающих алгоритмы, реализуемые в конторах управления телевизионных следящих систем, для последующего синтеза быстродействующих структур специализированных вычислительных систем.

Далее в статье используется терминология и развиваются результаты, изложенные в [1].

2. Постановка задачи

В [1] были получены результаты, позволяющие синтезировать быстродействующие вычислительные структуры в условиях параллельного поступления данных, т.е. в условиях отсутствия ограничений, накладываемых физической рекуррентностью входных сигналов.

Задача синтеза быстродействующих вычислительных структур для случая ограничений, связанных с физической рекуррентностью входных сигналов, требует описания процессов такими математическими моделями, которые моделировали бы изменение реакции системы на входные сигналы в зависи-

Для преподавателей университетов,
научных работников, аспирантов

Тематика:

- **Компьютерная инженерия**
- **Математическое моделирование**
- **Оптимизация и процессы управления**
- **Автоматизация проектирования и диагностика**
- **Информационные интеллектуальные системы**
- **Проектирование интегральных схем и микросистем**
- **Компьютерные технологии в образовании**

Адрес редакции: 61166, Харьков, просп. Ленина, 14,
ХНУРЭ, тел. 40-93-26, e-mail: ri@kture.kharkov.ua