УДК 681.5

РАЗРАБОТКА МЕТОДА ПОСТРОЕНИЯ ИНТЕЛЛЕКТУАЛЬНОЙ СИСТЕМЫ С ПРИМЕНЕНИЕМ ПОВЕДЕНЧЕСКОГО АНАЛИЗАТОРА

©Собчак А. П., Алёшина Е. С.

Национальный аэрокосмический университет им. Н. Е. Жуковского «ХАИ»

Рассматривается подход, позволяющий использовать интеллектуальные системы с применением поведенческого анализа результатов. Анализируются модели, которые обладают достаточной гибкостью, характеризуются наличием различных ограничений. Выделяются главные подходы к моделированию поведения. При помощи поведенческого анализа результатов интеллектуальная система является актуальной задачей, представляющей научный интерес.

Ключевые слова: информационные технологии, поведенческий анализ, компилятор, интеллектуальные системы.

Собчак А. П., Альошина О. С. «Розробка методу побудови інтелектуальної системи із застосуванням поведінкового аналізатора».

Розглядається підхід, що дозволяє використовувати інтелектуальні системи з застосуванням поведінкового аналізу результатів. Аналізуються моделі, які володіють достатньою гнучкістю, характеризуються наявністю різних обмежень. Виділяються головні підходи до моделювання поведінки. За допомогою поведінкового аналізу результатів інтелектуальна система є актуальним завданням, що представляє науковий інтерес.

Ключові слова: інформаційні технології, поведінковий аналіз, компілятор, інтелектуальні системи.

Sobchak A. P, Alyoshina E. S. “Development of a method for constructing intelligent system using behavioral analyzer”.

The approach allows to use intelligent systems using behavioral analysis results. Analyzed models which have sufficient flexibility characterized by different constraints. Stresses the main approaches to modeling behavior. From a behavioral analysis of the results an intelligent system is an important task of scientific interest.

Key words: information technology, behavioral analysis, the compiler, intelligent systems.

1. Актуальность исследования

Развитие информационных технологий предполагает наличие большого количества программно-аппаратных комплексов и платформ для эффективного управления и сопровождения производства, а также наличие промышленно функционирующих баз данных и хранилищ знаний большого объема, содержащих информацию по всем направлениям деятельности общества, наличие технологий, обеспечивающих интерактивный доступ любого пользователя к информации и ресурсам.

2. Постановка задачи

Каждая команда языка высокого уровня обычно соответствует сразу нескольким машинным командам. В связи с этим ставится задача для различных ЭВМ. Можно использовать один абстрактный, т.е. не встроенный в определенный процессор, язык высокого уровня. Для программ, написанных на языках высокого уровня, требуются более сложные транслирующие программы, называемые компиляторами.

Компилятор – это программа, которая считывает текст программы, написанной на одном языке – исходном, и транслирует (переводит) его в эквивалентный текст на другом языке – целевом. Одним из важных моментов трансляции является сообщение пользователю о наличии ошибок и исходной программе.

На первый взгляд, разнообразие компиляторов потрясает. Используются тысячи исходных языков, от традиционных, таких как Fortran и Pascal, до специализированных, возникающих во всех областях компьютерных приложений. Целевые языки не менее разнообразны – это могут быть другие языки программирования, различные машинные языки – от языков микропроцессоров до суперкомпьютеров.

3. Анализ существующих исследований

В перспективе развитие информационных систем и эксплуатационных возможностей ЭВМ. Для широкого использования появятся мониторы на редких кристаллах, с плоским экраном по диагонали, невероятно большими объемами памяти и т.п. Для введения информации в ЭВМ будут доступны голосовые методы, например, с помощью мобильного телефона; введение информации картинкой и графикой, сигналами. Будут усовершенствованы методы непосредственного введения текста, межмашинного обмена информацией. Использование компактных дисков (CD) для продолжительного хранения информации сделает такую ЭВМ по сути «бездонным» хранилищем.

С учётом прогноза информационных систем в 2015 году любую поверхность можно будет использовать в качестве дисплея. Повсеместное распространение получит видеосвязь, и на нее придется 400 экзобайт трафика. Объем скачиваемых фильмов и файлов составит 100 экзабайт. В 2025 году будут проведены эксперименты по телепортации на уровне элементарных частиц. В 2050 году, если население планеты увеличится до 9 миллиардов человек. Можно сказать, что 95 % знаний, которыми человечество будет владеть к 2060 году, станет результатом научных открытий в следующие 50 лет [1].

4. Основной материал

Основные участники экспертной системы: экспертная система, эксперты, средства построения и пользователи. Любая экспертная система структурирована в целях упрощения процесса принятия решения, включает совокупность знаний. Знания в ЭС организованы так, чтобы знания о предметной области отделить от других типов знаний системы, таких как общие знания, о том, как решать задачи или знание о том, как взаимодействовать с пользователем. Существует более высокий класс приложений, где требуется учитывать динамику изменения окружающего мира за время исполнения приложения. Такие экспертные системы получили название динамических ЭС и их обобщенная структура будет иметь вид, приведенный на рис. 1.

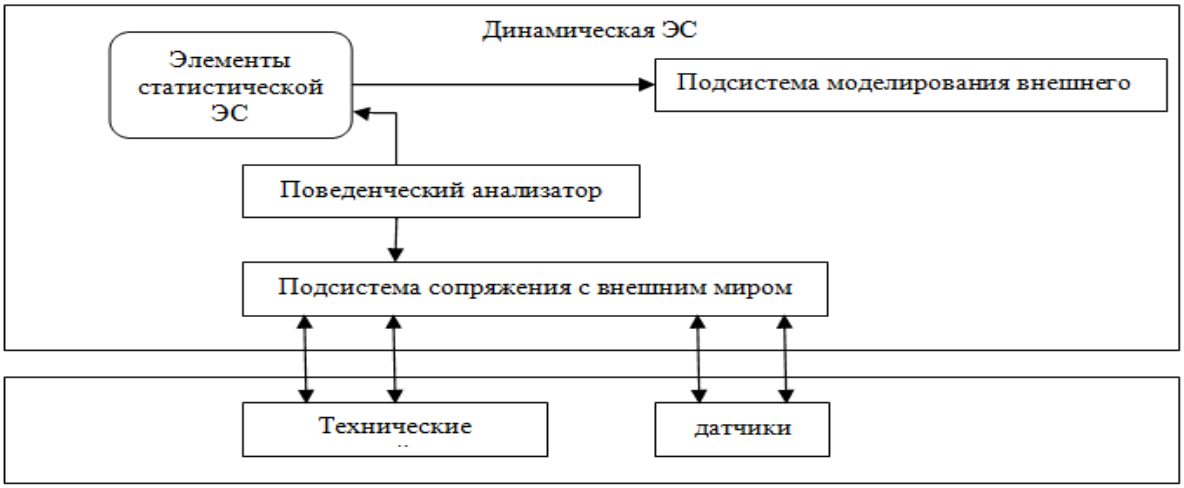

Рис. 1 –Динамические экспертные системы

Динамическая экспертная система, работа которой основана на экспертных оценках, явившихся следствием убежденности эксперта (может быть, нескольких экспертов), дает вполне однозначную рекомендацию на сложившуюся ситуацию. Динамичес-кие экспертные системы имеют изменяемую во времени базу знаний и делают выводы на ее основе. Поэтому в таких динамических экспертных системах одна и та же задача может быть решена по-разному, если вы обратитесь к экспертной системе в разное время. Знания в экспертной системе пополняются и изменяются ежедневно и даже ежечасно. [2].

4.1. Сущность и виды компиляторов

Компилятор переводит исходную программу в эквивалентную программу на языке, понятном компьютеру, то есть на машинном языке. Процесс компиляции и последующего выполнения программы можно изобразить следующим образом: сам компилятор также является программой, причем программой на машинном языке. Язык, на котором написан компилятор, называется языком реализации [3].

Одна из особенностей компилятора заключается в том, что он принимают на вход и выдают данные с очень сложной структурой, поэтому при проведении верификации на практике можно исследовать поведение компилятора лишь на небольшом подмножестве входных программ.

Языки высокого уровня являются основным средством разработки программных систем. Спецификация языка высокого уровня задаёт класс текстов, принадлежащих этому языку, и определяет семантику исполнения программ, написанных на этом языке. Задача перевода текстов с языка высокого уровня в представление, выполнимое на вычислительной системе, решается комплексами программ, которые по традиции называют компиляторами [4].

Функциональность, типичную для большинства компиляторов, можно условно декомпозировать на следующие задачи:

1. Анализ синтаксической корректности исходного теста программы.
2. Анализ выполнения контекстных условий в исходном тексте.
3. Оптимизация внутреннего представления и генерация выходных данных.

Результатом работы компилятора является объектный модуль на машинном языке или модуль на исполнимом промежуточном языке, таком как байт-код Java и Python или Intermediate Language платформы.

Процесс компиляции состоит из следующих этапов:

1. Лексический анализ. На этом этапе последовательность символов исходного файла преобразуется в последовательность лексем.

2. Синтаксический (грамматический) анализ. Последовательность лексем преобразуется в дерево разбора.

3. Семантический анализ. Дерево разбора обрабатывается с целью установления его семантики (смысла) – например, привязка идентификаторов к их декларациям, типам, проверка совместимости, определение типов выражений и т. д. Результат обычно называется «промежуточным представлением/кодом».

4. Оптимизация. Выполняется удаление излишних конструкций и упрощение кода с сохранением его смысла. Оптимизация может быть на разных уровнях и этапах – например, над промежуточным кодом или над конечным машинным кодом.

5. Генерация кода. Из промежуточного представления порождается код на целевом языке.

Существуют такие виды компиляторов:

1. Векторизующий. Транслирует исходный код в машинный код компьютеров, оснащённых векторным процессором.

2. Гибкий. Сконструирован по модульному принципу, управляется таблицами и запрограммирован на языке высокого уровня или реализован с помощью компилятора компиляторов.

3. Диалоговый. Обеспечивает использование языка программирования в режиме разделения времени.

4. Инкрементальный. Повторно транслирует фрагменты программы и дополнения к ней без перекомпиляции всей программы.

5. Интерпретирующий (пошаговый). Последовательно выполняет независимую компиляцию каждого отдельного оператора (команды) исходной программы.

6. Отладочный. Устраняет отдельные виды синтаксических ошибок.

7. Резидентный. Постоянно находится в оперативной памяти и доступен для повторного использования многими задачами.

8. Самокомпилируемый. Написан на том же языке, с которого осуществляется трансляция.

9. Универсальный. Основан на формальном описании синтаксиса и семантики входного языка. Составными частями такого компилятора являются: ядро, синтаксический и семантический загрузчики.

10. Компилятор компиляторов – программа, воспринимающая синтаксическое или семантическое описание языка программирования и генерирующая компилятор для этого языка. Таким образом, пользователю компилятора компиляторов в любом случае нужно разработать исполняющие структуры и выбрать способ преобразования каждой входной синтаксической конструкции в операции выходного языка или в машинные операции, после чего нужно написать собственно процедуры генерации кода. Следовательно, компилятор компиляторов – это полезное средство, помогающее писать компиляторы.

Clang – это компилятор для C-подобных языков, созданный специально для работы на базе LLVM. Комбинация Clang и LLVM предоставляет набор инструментов, позволяющих полностью заменить GCC. Благодаря архитектуре, основанной на библиотеках, Clang (как и LLVM) легко встраивается в другие приложения.

Одной из главных задач Clang является поддержка инкрементной компиляции, позволяющей более тесно интегрировать компилятор и графический интерфейс среды разработки, в отличие от GCC, который был создан для работы в классическом цикле «компиляция-линковка-отладка».

Программы-компиляторы бывают оценочные и профессиональные.

Оценочные или учебные компиляторы позволяют написать простейшие программы для конкретного процессора и определить, подходит ли процессор для тех задач, которые предстоит решать в процессе разработки устройства. Конечно, если программа очень проста, то можно весь программный продукт написать на оценочном компиляторе. Оценочные компиляторы позволяют транслировать одиночный файл исходного текста программы. В результате работы оценочного компилятора сразу получается исполняемый или загрузочный модуль программы, поэтому такие компиляторы называются компиляторы с единой трансляцией.

Профессиональные трансляторы позволяют производить трансляцию исходного текста программы по частям. Это позволяет значительно сократить время трансляции исходного текста программы, так как не нужно транслировать весь текст программы, а можно транслировать только ту часть программы, которая менялась после предыдущей трансляции.

В состав современных средств написания и отладки программ для микроконтроллеров обычно входят эмуляторы процессоров или отладочные платы, текстовый редактор, компиляторы языка высокого уровня (чаще всего «С») и ассемблера, редактор связей и загрузчик программы в отладочную плату.

Модельный компилятор состоит из четырех частей: сканер, блок таблиц имен, основная часть компилятора и семантические подпрограммы. Сканер читает входной файл и избавляет остальные части компилятора от необходимости следить за каждым входным символом. В таблице имен хранится информация о переменных программы. Обращение к ней осуществляется в основном через процедуры блок таблицы имен [5].

Компиляторы полностью обрабатывают весь текст программы. Они просматривают его в поиске синтаксических ошибок (иногда несколько раз), производят определенный смысловой анализ, а затем автоматически переводят (транслируют) на машинный язык – генерируют машинный код. Нередко при этом выполняется оптимизация с помощью набора методов позволяющих повысить быстродействие программы (например, с помощью инструкций, ориентированных на конкретный процессор, путём исключения ненужных команд, промежуточных вычислений и т.д.). В результате законченная программа получается законченной и эффективной, работает [6].

4.2. Назначение и структура электронного компилятора

Разработанный макет РАПЛ представляет собой комплекс инструментальных программно-аппаратных средств подготовки алгоритмов и программ в реальном времени в соответствии с настраиваемой архитектурой АС. РАПЛ можно использовать также для создания систем с ограниченными ресурсами, где затруднена быстрая адаптация программных трансляторов к изменяющимся языкам и ЭВМ. Применение специальных микропрограммных средств для построения систем значительно повышает ресурс и обеспечивает требуемую гибкость программных средств. Быстродействие аппаратных средств во многих случаях намного выше, чем необходимо для практики. Благодаря этому можно понизить тактовые частоты и повысить надежность аппаратуры по сравнению с программной реализацией [7].

Структура электронного компилятора (ЭК), разработанного в составе макета РАПЛ, показана на рис. 1, где в функциональном плане выделены следующие структурные уровни:

– программный, включающий в себя стандартное обеспечение ПЭВМ и средства реализации уникальных функций;

– аппаратный, состоящий из ЭК и устройств сопряжения с базовой ПЭВМ;

– сервисный, программно-аппаратные средства моделирования входных языков пользователей, настройки операционных и управляющих автоматов ЭК и пользовательский интерфейс.

ЭК – это ядро РАПЛ, представляющий собой систему трех специализированных процессоров: лексического анализатора, синтаксического анализатора и генератора кода. Эти процессоры выполнены в виде автономных блоков, функционально и информационно связанных между собой.

Генератор кода формирует несложный в исполнении микрокод, определяемый триадной формой представления программ.

Рис. 2 – Структура ЭК

ЭК построен по схеме синтаксически управляемого перевода в соответствии с грамматикой языка, задаваемой посредством синтаксических диаграмм и БНФ. В связи с этим значительно упрощается адаптация ЭК к изменяющимся языкам и его диагностика.

4.3. Аппаратная реализация компилятора

Для аппаратного описания компилятора применяется язык описания аппаратуры AHDL, который разработан фирмой Altera и предназначен для описания комбинационных и последовательностных логических устройств, групповых операций, цифровых автоматов (state machine) и таблиц истинности с учетом архитектурных особенностей ПЛИС фирмы Altera. Он полностью интегрируется с системой автоматизированного проектирования ПЛИС, например MAX+PLUS II. Файлы описания аппаратуры, написанные на языке AHDL. Для создания TDF-файла можно использовать как текстовый редактор системы MAX+PLUS II, так и любой другой. Проект, выполненный в виде TDF-файла, компилируется, отлаживается и используется для формирования файла программирования или загрузки ПЛИС фирмы Altera [8].

Операторы и элементы языка AHDL являются достаточно мощным и универсальным удобным в использовании средством описания алгоритмов функционирования цифровых устройств. Язык описания аппаратуры AHDL дает возможность создавать иерархические проекты в рамках одного этого языка или же в иерархическом проекте использовать как TOP-файлы, разработанные на языке AHDL, так и другие типы файлов.

4.4. Анализ поведенческих факторов

Анализ поведения человека характеризуется приверженностью к селекционистскому подходу: исключительная опора на экспериментальный анализ как источник принципов для понимания поведения, фокус на поведении индивидуумов (а не групп) существенный интерес к применению науки для улучшения условий человеческого существования.

Таблица 1 – Положительные и отрицательные черты различных типов характера

Тип характера	Положительные черты	Отрицательные черты
Шизоидный	1) Хорошая память 2) Умение замечать детали 3) Творческий подход к выполнению дел	1) Страх быть непонятым окружающими 2) Уход в себя 3) Редкое проявление эмоций 4) Отстраненность от окружающих
Нарциссический	1) Заниженное чувство агрессивности	1) Преувеличенное чувство собственной значимости 2) Резкое непринятие критики 3) Неспособность к самозащите 4) Острая потребность в заботе и поддержке
Параноидальный	1) Инстинкт самосохранении на высоком уровне	1) Подавленность 2) Недоверчивость 3) Обостренное чувство опасности 4) Острая подозрительность к людям 5) Неадекватные эмоциональные взрывы
Обсессиво-компульсивный	1) Повышенное внимание к мелочам и деталям 2) Опрятность 3) Бережливость 4) Стремление к совершенству	1) Нерешительность 2) Сомнительность 3) Отсутствие чувства юмора
Психопатический (антисоциальный)	1) Наличие качеств лидера	1) Неспособность к длительным отношениям 2) Тотальная недоверчивость 3) Низкий порог агрессивности
Истерический	1) Сердечность 2) Человечность 3) Хорошо развит инстинкт самосохранения	1) Острая чувствительность к происходящему
Депрессивный и маниакальный		1) Маниакальное состояние самоуничижения 2) Глубокая самокритика 3) Болезненная реакция на критику 4) Легко поддаются любому влиянию
Мазохический		1) Скованность 2) Неестественность 3) Неуверенность 4) Чрезмерная скромность 5) Добровольная униженность

Основными факторами поведения человека являются: количество отказов – это мгновенное закрытие сайта или же не переход на следующую страницу; количество просмотренных страниц на сайте; время, проведенное пользователем на сайте; количество переходов на сайт из поисковика – то есть, количество кликов по поисковой выдаче. Для анализа пользовательского поведения используются счетчики посещаемости. Необходимо обратить внимание, что существует возможность реализации аппаратно с помощью статистики по техническим параметрам компьютера пользователя – разрешению экрана, версией браузера, операционной системой [8].

Выводы

Исходя из всего вышесказанного можно сделать следующие выводы: компилятор языка высокого уровня – это программа, которая переводит исходный текст программы в эквивалентную ей объектную программу. Объектная программа формируется на объектном языке, который, как правило, является некоторым машинным языком. Особого внимания заслуживают алгоритмы синтеза МА и МП, поскольку на их основе могут быть разработаны аппаратные компиляторы, обладающие огромными преимуществами по сравнению с программными компиляторами. Залогом их успешной реализации служат достижения в технологии проектирования БИС с высокой степенью интеграции. Для решения проблемы синтеза аппаратных компиляторов требуется создание принципиально нового математического аппарата, в качестве которого можно использовать АРЕКС и Р-СИАА. На базе АРЕКС и Р-СИАА возможна полная формализация основных этапов проектирования аппаратных систем компиляции, функционирующих как в режиме интерпретации, так и в режиме трансляции.

Режим интерпретации означает, что генерируемые компилятором структурные команды выполняются процессором сразу, не дожидаясь окончания процесса компиляции в целом. Для осуществления этого режима МП необходимо снабдить дискретными преобразователями (ДП) на выходе и входе. Входной ДП предназначен для лексического анализа языков, выходной – для реализации отдельных команд.

При программной реализации интерпретаторы работают медленнее трансляторов, так как в первом случае процессор вынужден простаивать и дожидаться генерации очередной команды. При этом быстродействие будет на 2–3 порядка выше, чем у программных интерпретаторов. Снабдив подобные интерпретаторы средствами отладки, тестирования и распараллеливания программ, получим быстродействующую языковую машину, которая будет работать намного эффективнее, чем современные ЭВМ.

Практическая реализация проблемы синтеза аппаратных компиляторов позволит:

1) создать автоматизированную систему проектирования функционально полных компиляторов произвольного назначения с быстродействием, определяемым современной элементной базой. Исходными данными при проектировании компиляторов будут служить контекстно-свободные грамматики, представленные посредством синтаксических диаграмм, форм Бэкуса-Наура;

2) произвести при одних и тех же исходных данных генерацию программных модулей компиляторов и функционально полных тестов для моделирования и проверки работоспособности аппаратных компиляторов, реализуемых посредством модулей и кристаллов;

3) организовать разработку многофункциональных аппаратных компиляторов, использующих на входе различные ЯВУ;

4) обеспечить создание сверхбыстродействующих аппаратных компиляторов, работающих в режимах параллельной трансляции и интерпретации с высоким уровнем надежности.

В результате проанализированы подходы построения интеллектуальных систем. Рассмотрен алгоритм работы интеллектуальной системы с применением поведенческого анализатора. Предложен метод построения интеллектуальных систем с применением поведенческого анализатора. Рассмотрена физическая реализация интеллектуальной системы с помощью языково-аппаратного описания и ПЛИС технологий. Дана оценка практической реализации синтеза интеллектуальных систем с помощью аппаратных компиляторов.

Список использованных источников:

1. Интеллектуальные системы автоматического управления / под ред. И. М. Макарова, В. М. Лохина. – М. : Физматлит, 2001. – 576 с. – (Сер. Профессиональное управление).

2. Охтилев М. Ю. Интеллектуальные технологии мониторинга и управления структурной динамикой / М. Ю. Охтилев, Б. В. Соколов, Р. М. Юсупов. – М. : Наука, 2006. – 410 с.

3. Уотерман Д. Руководство по экспертным системам / Д. Уотерман. – М. : Мир, 1989. – 474 с.

4. Ахо А. Компиляторы: принципы, технологии и инструменты / А. Ахо, Р. Сети, Дж. Ульман. – М. : Вильямс, 2001. – 768 с.

5. Проблемы создания аппаратного компилятора элементарных математических функцій / А. П. Собчак, В. М. Илюшко, К. В. Ходарев, Н. Д. Смирнов // Авиационно-космическая техника и технология. – 2005. – Вып. 7(23). – С. 294–303.

6. Собчак А. П. Отношения эквивалентности на множестве диагностических моделей объектов контроля / А. П. Собчак, А. В. Чечуй, В. Н. Торчило // Системи обробки інформації : зб. наук. пр. / Харк. ун-т Повітряних Сил ім. І. Кожедуба. – Х., 2002. – Вип. 5 (21). – С. 189–193.

7. Собчак А. П. Реализация лексического анализатора на языке аппаратного описания / А. П. Собчак, А. М. Марченко, К. В. Ходарев // Открытые информационные и компьютерные интегрированные технологи : сб. науч. тр. / НАКУ «ХАИ». – Харьков, 2004. – Вып. 23. – С. 116–121.

8. Собчак А. П. Розробка цифрових електронних засобів на базі програмованих логічних інтегральних схем : навч. посіб. до лабораторного практикуму / А. П. Собчак, А. М. Марченко. – Х. : Изд-во “ХАІ”. – 2004. – 48 с.

Стаття надійшла до редакції 17 лютого 2014 р.