Труды XII Международного симпозиума
«Методы дискретных особенностей в задачах математической физики»
(МДОЗМФ-2005)
Харьков-Херсон, 2005, с.71-74

УДК 517.968.519.6

Программная реализация языковых средств поддержки вычислительных экспериментов

А. Е. Гавва, В. О. Мищенко
Национальный банк Украины, Украина
Харьковский национальный университет им. В.Н. Каразина, Украина

We discuss about software tools that must be supported computer-based experiments in DSM area. The Ada program of compilers compilation is presented. We proved and demonstrated that this COCO/R-Ada is adequate to demands for the discussed tools. COCO/R-Ada is based on the known standard that was established by Hanspeter Mossenbock in form of COCO and COCO/R systems at 1983-88.

1. Введение в предмет и актуальность задачи

Транслятор - это обобщающее название для обширного класса программ, которые эквивалентным (в каком-то смысле) образом преобразуют одни программы в другие [1]. Например, текст на одном формальном языке преобразуется в "такой же" на другом языке. Другие примеры: броузер, который транслирует язык HTML в команды "рисования" в экранной области памяти, SQL-сервер СУБД - транслятор языка SQL в последовательность низкоуровневых операций с базой данных и т. п.

В проектах автоматизации и информационной поддержки применений методов дискретных особенностей (МДО) [2-4] оригинальная системная часть их программной реализации относится, в основном, к трансляторам. Это интерпретаторы команд, компиляторы размеченных текстов, визуализаторы. То, что указанные проекты пока не реализованы как программные продукты, ясно показывает высокую трудоёмкость "ручной" разработки трансляторов.

Ряд универсальных и специализированных сред для разработки программного обеспечения имеет стандартизованные инструменты построения необходимых пользователям трансляторов. Например, неотъемлемые утилиты ОС UNIX - `lex` и `yacc` или, обращаясь к теме нашей работы, те компиляторы компиляторов (**Co**mpiler-**Co**mpiler или, короче, *COCO*), которые на разных платформах продолжают оригинальную систему [5]. Высокопроизводительные нисходящие рекурсивные синтаксические анализаторы-трансляторы *COCO/R* [6] созданы под системы на языках `Oberon`, `Modula-2`, `Turbo Pascal`, `C`, `Java` и даже `C#` (см. документ в Интернете http://www.scifac.ru.ac.za/coco/).

В этом перечне до настоящего времени отсутствовали как раз те системы, которые адекватны вычислительным применениям МДО: `Fortran`, `Delphi`, `C++` и `Ada`. Важное обстоятельство! - ведь пользователь должен подать на вход *COCO* наряду с синтаксисом исходной транслируемой программы описание действий, составляющих цель трансляции. Это делается на языке программирования используемой системы. Более того, сложный вычислительный эксперимент обычно сопряжен с характерными только для

него систематическими (по ходу эксперимента) модификациями определённых участков кода используемых программ. Необходимых для автоматизации этого инструментальных средств нет ни в одном из вышеупомянутых языков, за исключением языка программирования Ada.

Из сказанного понятна высокая актуальность воспроизводства генератора компиляторов *COCO/R* в совершенно новой для его реализации среде языка Ада.

2. Характеристика интерфейса разработанной системы

Первый из авторов реализовал (на языке Ада) алгоритм сборки исходных текстов компиляторов на языке Ада по описаниям на входном языке *COCO/R* .

COCO/R-Ada интегрирует описание сканера и описание синтаксического анализатора, что позволяет избежать проблемы построения интерфейса между генерируемыми частями. Входной язык *COCO/R* основан на *атрибутивной грамматике*, которая представляет собой формализм для спецификации его семантики; в данном - случае средствами алгоритмической нотации.

Поставка реализации *COCO/R*-Ada распространяется под лицензией GPL и доступна по адресу http://www.ada-ru.org/src_acr.html (http://www.ada-ru.org/files/cocor_ada-1.53.1.tar.gz - архив с исходными текстами; http://www.ada-ru.org/files/cocor_doc_ru.zip - документация на русском языке) Предполагается, что реализация *COCO/R*-Ada не зависит от платформы, однако установка и использование данного инструмента пока тестировались только под ОС Linux.

Для генерации с помощью *COCO/R*-Ada исходных текстов необходимой пользователю программы-компилятора используются файлы "compiler.frm", "parser.frm" и "scanner.frm". Затем для сборки исполняемого модуля компилятора используются файлы "fileio.ads", "fileio.adb", "sets.ads" и "sets.adb" из вышеуказанной поставки.

Выбор транслируемого языка ограничивается классом LL(1), сочетающим вполне достаточную широту с высокой эффективностью грамматического разбора [7]. Описание компилятора состоит из нескольких частей и имеет вид:

```
 COMPILER Имя_компилятора
--Текст на языке Ада, содержащий спецификаторы контекста и описания
--глобальных объектов, типов, подпрограмм которые
-- потом потребуются в семантических действиях, например,
 with Operations; use Operations;
        --Формальная грамматика для сканера:
CHARACTERS
--Символы текста транслируемой программы
TOKENS
--Лексемы транслируемой программы
        --Формальная грамматики для синтаксического анализатора:
PRODUCTIONS
--Порождающие правила атрибутивной грамматики
END Имя_компилятора .                                          (1)
```

При этом базовым средством описания всех грамматик служит РБНФ. Для определения символов текста (которые попадают на вход сканера) и лексем, передаваемых синтаксическому анализатору, порождающие правила имеют обычный вид. Например,

```
digit    = "0123456789".   letter    = "ABCDIKLMNPRSTUFXYZ".
```

```
TOKENS
Number = digit { digit }.      Identifier = letter | { letter | digit }
Symbol = "+" | "-" | "*" | "/"                                          (2)
```

Особенностью определения атрибутивной грамматики для синтаксического анализатора является возможность описывать (в угловых скобках) средствами языка Ада атрибуты грамматических понятий и (в квази-скобках) - алгоритмические действия, которые должен выполнять транслятор при распознании данного понятия в данном контексте. Например, для компиляции операций со списком целых аргументов (в духе Лиспа) формальная грамматика и семантическая атрибутика могут (схематически) выглядеть так

```
PRODUCTIONS
Sign    <S: out String1..1);>
        =  Symbol          (.  S:= Symbol); .).
Arg     <Val: out Integer;>
        =       Number                          (* Val вычисляем*)
            |
                Identifier                      (*  Val  находим  в
            таблице*).
Operation <Res: out Integer;> (.    P: Oper_Access .)
        =       "("
                Sign<C: caracter;> (.   P:= Oper_Table(C); .)
                Arg <V: Integer;>  (.   Res:= Val); .)
            {   Arg <V: Integer;>  (.   Res:= P.all (Res, Val); .)
            }       ")".                                                (3)
```

Контроль корректности описания синтаксиса транслируемого языка поддерживается автоматически. В отношении семантики поддержка осуществляется в смысле формальной корректности собираемой генератором Ada программы (компилятора), но не в отношении смысла трансляции. За него, конечно же, отвечает сам пользователь *COCO/R*-Ada.

3. Пояснения на демонстрационном примере

Обратимся к условному и весьма упрощенному примеру для того, чтобы пояснить характер проблем проектирования и техники реализации средств поддержки вычислительных экспериментов с помощью *COCO/R*-Ada. Предположим, создана библиотека программ, принимающих исходные данные из файлов и сохраняющих в файлы. Программы могут формировать матрицы СЛАУ МДО (Matr), аппроксимировать эти матрицы с целью снижения размерности (Appr), искать для них предобуславливатели (Pred), решать системы и анализировать их обусловленность (Solv и Cond), рисовать диаграммы и графики (Diag и Grph) и т. п. Хотелось бы проводить численные эксперименты в различных режимах: проверочном с выдачей максимально подробной информации в процессе счёта ("Trace"), в рабочем режиме ("Compute") и др. Подходящая для этого формальная грамматика

```
Instruction
        =  "Check"      |   "Trace"|   "Compute" .
Arg
        =       Identifier  |  String  |  Float  |  Number   .
Program
```

```
=      Identifier  "("  Arg    {   Arg    }   ")".
Experiment
=      "("  Instruction     Program   {   Program   }          ")".
```

имеет ту же структуру, что и (3), - её сканер и кодогенерация могли бы быть аналогичными. Но при этом отсутствовал бы, например, контроль за интерфейсом между вызываемыми программами по файлам данных.

Как такой интерфейс обеспечивается "вручную"? В тексте очередной по списку вызовов программы, скажем, Solv, отыскиваются операции ввода и проверяется совпадение атрибутов входного файла с атрибутами выходного у предыдущей. Для того, чтобы делать это просто и надёжно программным путем для языка Ада существует стандартизованый инструментальный пакет ASIS.

4. Выводы и направления дальнейших исследований

Представленный в докладе компилятор компиляторов стандарта *COCO/R*, разработанный для среды программирования языка Ада, является адекватным средством разработки систем автоматизации подготовки и осуществления вычислительных экспериментов. Доступность *COCO/R*-Ada в Интернете должна обеспечить продвижение соответствующих проектов к их завершению.

Было бы полезно оценить надежность результатов и повышение производительности труда за счёт применения *COCO/R*-Ada в решении реальных задач построения сред вычислительных экспериментов МДО.

ЛИТЕРАТУРА

1. Вирт Н. Алгоритмы + структуры данных = программы.- Мир, М.:1985,- 406 с.
2. Мищенко В.О., Высочин Л.А. Язык организации вычислений в компьютерном моделировании методом дискретных зарядов // Методы дискретных особенностей в задачах математической физики. Труды VIII межд.симпозиума /Мин.Укрины по делам науки и технологий, ХГУ, ХГТУ, Ин-т математики НАН Украины, ГосНИЦ ЦАГИ (Россия), ВАТУ (Россия).- Харьков: ХГУ, 1999. - С.85-87.
3. Гандель Ю.В., Жолткевич Г.Н. Применение параметрических представлений сингулярных интегральных преобразований и систем компьютерной математики в плоских задачах дифракции // Труды X международного симпозиума "Методы дискретных особенностей в задачах математической физики" (МДОЗМФ-2001).- ISBN 966-630-05-2.- Херсон, 2001.- С. 96-99.
4. Мищенко В.О. Семантическая структуризация текстовых данных в базе данных, поддерживающей распределенную разработку программного обеспечения // Вестник ХГТУ. - 2002, N 1 (14). - я. 304-307.
5. Terry P.D., Rhodes University, 1996 "Compilers and Compiler Generators" (http://www.scifac.ru.ac.za/compilers/)
6. Hanspeter Mössenböck "A Generator for Production Quality Compilers" (http://www.ssw.uni-linz.ac.at/Research/Papers/Moe90.html)
7. Hanspeter Mössenböck "Coco/R - A Generator for Fast Compiler Front-Ends." (http://www.ssw.uni-linz.ac.at/Research/Reports/CocoReport.html)