ИССЛЕДОВАНИЕ ВОЗМОЖНОСТЕЙ ФУНКЦИОНАЛЬНЫХ ЯЗЫКОВ ДЛЯ СОЗДАНИЯ ГЕНЕРАТОРОВ ПРОГРАММ

Н. Н. Дацун, А. Ю. Другобицкий

ДонНТУ, Донецк, Украина

Особенностью функциональных языков (ФЯ) является единообразие представления данных и программ, что позволяет использовать ФЯ для разработки трансляторов с этих языков. Рассмотрены: свойства ФЯ, важные при разработке трансляторов, особенности средств трансляции функциональных языков, схема создания транслятора на основе функциональных языков и вариант ее оптимизации на платформе .NET.

1. Свойства функциональных языков, важные при разработке трансляторов:

— *типизация:* современные языки функционального программирования являются строго типизированными. Строгая типизация обеспечивает безопасность и некоторую степень оптимизации. Параметрический полиморфизм поддерживается большинством ФЯ и позволяет легко и естественно создавать и организовывать динамические структуры данных (стеки, деревья и пр.), используемые алгоритмами трансляции (во многих ФЯ такие структуры являются частью языка);

— *функции высших порядков;*

— *отсутствие побочных эффектов.*

Преимуществами строго функциональных языков являются упрощение анализа программ и параллелизм (возможно вычисление независимых функций в произвольном порядке или параллельно, причем этот параллелизм может быть организован не только на уровне компилятора с языка, но и на уровне архитектуры).

2. Особенности средств трансляции функциональных языков.

Особенности средств трансляции ФЯ в отличие от императивных языков (ИЯ):

— наличие «сборщика мусора» (параллельного основной программе процесса);

— функциональная программа в процессе работы может порождать новые программные конструкции, из-за этого в процессе выполнения программы могут выполняться не только предкомпилированные инструкции, но также активизируются модули лексического, синтаксического и семантического анализа, а также модуль кодогенерации.

3. Сравнение компонент трансляторов ИЯ и ФЯ.

Работа лексического и синтаксического анализаторов не отличается для транслятора ИЯ и ФЯ. Важную роль играет полиморфизм типов данных, хранимых в структурах лексического анализатора (таблицы, стеки, очереди). В некоторых ФЯ как часть языка определены объекты инструментов создания трансляторов — Lex и Yacc.

Семантический анализатор (проверка типов). В современных ФЯ из соображений надежности и быстроты работы трансляторы проектируются в модели статической типизации (семантический анализатор выполняется перед выполнением программы по аналогии с ИЯ) в отличие от ранних ФЯ с динамической типизацией.

Модуль кодогенерации. Отличие существует только для транслятора компилирующего типа: в зависимости от исходного текста программы в выходной исполнимый файл помещается транслятор данного языка для реализации функций динамической кодогенерации, свойственных функциональным языкам; выходным языком трансляции может быть не машинный язык целевой платформы, а промежуточный язык, заведомо безопасный для выполнения и оптимальный для трансляции в машинный код на множестве платформ.

Модуль исполнения программы присутствует только в трансляторах интерпретирующего типа и включает в себя виртуальную машину исполнения (с функцией «сборщика мусора») и транслятор функционального языка (активизация функций транслятора во время исполнения является характерной особенностью функциональных языков).

4. Схема генерации программ на основе спецификации (для ФЯ).

Необходимыми элементами последовательности трансляции являются программа на ФЯ и транслятор для него. Однако оказалось возможным заменить данные звенья цепочки классической трансляции на единую систему трансляции спецификаций в транслятор целевого языка. Такими возможностями обладает платформа .NET.

В результате исследования возможностей этой платформы оказалось возможным оптимизировать:

— схему трансляции, исключив промежуточное представление на функциональном языке и транслятор с него (непосредственное преобразования спецификации языка в транслятор) и упростив процесс трансляции спецификации;

— процесс трансляции за счет переноса этих функций на системный уровень (работа с памятью, межмодульное взаимодействие, безопасность выполнимого кода и данных, с которыми он работает, встроенная реализация основных структур данных объектов трансляции; кодогенерация и рефлексия; выполнение программы на виртуальной машине.