Правительство Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Санкт-Петербургский государственный университет»

Кафедра Системного программирования

Озерцов Александр Сергеевич

Апробация библиотеки Generic-transformers на примере компилятора языка Oberon0

Курсовая работа

Научный руководитель:
к. ф.-м. н., дофцент кафедры
системного программирования Булычев Д. Ю.

Санкт-Петербург
2016

Оглавление

Введение

При разработке промышленных систем довольно часто приходится использоать, в сущности, одни и те же операции. В таких ситуациях говорят о компонентной организации приложений.

Компилятор – это языковой процессор, основаный на трансляции языка высокого уровня в машинный язык или язык ассемблера. При разработке компилятора компонентизация особенно важна по причине возможного расширения языка и переопределения его свойств.

Компонентная организация програмного обеспечения — повсеместный хорошо зарекомендовавший себя на практике подход, заключающийся в повторном переиспользовании готовых компонент, что позволяет значительно сократить время разработки и тестирования приложения. На сегодняшний день компонентизация в области разработки компиляторов используется слабо, к примеру сборка из готовых компонент не представляется возможной без переопределения под конкретное использование.

Oberon-0 представляет собой императивный язык программирования, разработанный Н. Виртом [1] в образовательных целях и использованный им в качестве эталонного языка, удовлетворяющий требованиям достаточной простоты как компилятора, так и самого языка. Этот язык содержит стандартный набор императивных черт, таких как переменные, выражения, простейшие конструкции управления (присваивание, условие, циклы), процедуры с передачей параметров по значению и по ссылке, вложенные процедуры, агрегатные типы (массивы и структуры) с именной эквивалентностью. Таким образом, по замыслу автора, Oberon-0 является модельным языком, содержащим всё, что надо для того, чтобы проиллюстрировать основные задачи компиляции, но в то же время достаточно простым, чтобы эта иллюстрация была обозримой.

По своей структуре язык программирования Oberon0 можно рассматривать в виде члена семейства языков, часть которых являются его языковыми расширениями (надъязыками), а часть - подъязыками,

при этом один язык семейства получается из другого расширением. В таком контексте проблема компонентной организации становится критичной.

Язык программирования OCaml[1] обладает набором уникальных для функциональных языков возможностей, например, реализацией полиморфных вариантов и поддержкой парадигмы объектно-ориентированного подхода, что в свою очередь делает возможным комбинирование различных способов обобщенного программирования, таких как обобщенное программирование с помощью полиморфных вариантов и generic-функций и возможность переопределения поведения методов в классах.

Предыдущая реализация компилятора языка Oberon0 была основана на неявном выведении полиморфных вариантов, в результате чего требовалось вводить дополнительные конструкции для функций преобразования типов данных, что порождало множество однотипного кода. ТАкже недостаток такого подхода состоит в сложности отладки и модификации поведения тех или иных алгоритмов, что так же накладывает дополнительные неудобства.

Библиотека Generic-transformers использует выразительность языка Ocaml для обобщенного программирования, что позволяет ей совмещать в себе все вышеперечисленные подходы обобщенного программирования. Так же, используя переопределение методов, становится возможным изменение поведения лишь отдельных компонент функций преобразования, что позволяет уменьшить количество кода.

[1] http://caml.inria.fr/ocaml/

1. Постановка задачи

Целью данной работы является уменьшение размера кода компилятора языка программирования Oberon-0 путем использования библиотеки Generic-transformers. Для достижения поставленной цели были сформулированы следующие задачи.

- Изучить основные подходы обобщенного программирования. Изучить библиотеку Generic-transformers.
- Реализовать компилятор Oberon-0 с использованием библиотеки Generic-transformers.
- Сравнить полученный результат с предыдущей реализацией.

2. Обзор

2.1. Представление языка Oberon0 в виде семейста языков

Несмотря на внешнюю монолитность, язык Oberon-0 представляется в виде семейства языков, где каждое последующее семейство является расширением предыдущего. Это сделано для удобства изучения компиляторов и их строения[2]. Характеристика этого семейства дана в таблице 1.

Уровни языка	
L0	Константы, переменные, атомарные типы, простые выражения и присваивания
L1	Конструкции IF и WHILE
L2	Конструкции FOR и CASE
L3	Процедуры
L4	Массивы и структуры
L5	Вложенные процедуры

Таблица 1: Языки семейства.

Самому языку Oberon-0 в этой иерархии соответствует уровень L4. L5 есть язык, синтаксически схожий с L4, но обладающий вложенными процедурами. Собственно, Oberon-0, хоть и содержит вложенные функции как синтаксическую конструкцию, такой семантикой не обладает.

2.2. Задачи компиляции

Набор решаемых задач компиляции компиятора языка Oberon-0 показан в Таблице 2. Он представляет собой типичный набор параметров, которые должен реализовывать практически любой компилятор статически-типизированного язы- ка. В качестве целевой платформы здесь выступает язык C, но с тем же успехом мог бы быть использован, например, Java.

[2]Пункты 2.1, 2.2, 2.3 более подробно разобраны в [8]

Задачи компиляции	
T1	синтаксический анализ и форматирование
T2	анализ имен (идентификация)
T3	анализ типов
T4	понижение уровня (desugaring)
T5	генерация C

Таблица 2: Подзадачи компиляции.

2.3. Комбинации языка семейства и задач компиляции

Каждая комбинация языка семейства и конкретной задачи образует некоторый законченный артефакт. Например, выбрав язык L2 и задачу T3 мы получаем артефакт, который может осуществлять типовой анализ программ, содержащих конструкции до FOR и CASE включительно, но не поддерживающий процедуры или агрегатные типы и не имеющий возможности генерировать код.

Для реализации были выбраны лишь некоторые артефакты; их характеристика дана в Таблице 3.

Артефакт	Язык	Задачи	Описание
A1	L2	T1-2	Базовый язык с форматированием и идентификацией
A2a	L3	T1-2	A1 плюс процедуры и их идентификация
A2b	L2	T1-3	A1 плюс анализ типов
A3	L3	T1-3	A2a и A2b
A4	L4	T1-5	Все задачи для L4
A5	L5	T1-5	Все задачи для L5

Таблица 3: Таблица артефактов.

2.4. Принцип работы Generic-transformers

Одним из основных принципов компонентной организации является обобщенное программирование. В этой главе приведен обзор извест-

ных методов обобщенного программирования таких как полиморфные варианты, обобщенные функции и возможность переопределения поведения методов в классах. Библиотека Generic-transformers совмещает в себе данные подходы.

2.4.1. Полиморфные варианты

Одной из главных причин появления полиморфных вариантов стало отсутствие расширяемости алгебраических типов данных, что вынуждало разработчика переписывать функцию при добавлении в тип нового конструктора. Данная проблема получила название expression problem и была описана Филиппом Вадлером с соответствующей статье [2]. Язык OCaml решил данную проблему путем добавления особого типа, содержащего в себе всевозможные конструкторы.

Такой подход позволил создавать функции обработки типов практически без учета их конструкторов, а при добавлении новых конструкторов в тип использовать функцию, которая новый конструктор обрабатывает как требуется, а для обработки старых конструкторов вызывает ранее описанную функцию. Подробнее такой подход описан в [3]

Основная проблема такого подхода заключается в том, что отсутствует механизм препятствованию вызова функции с типом, содержащим необрабатываемый аргумент. Более того, зачастую функции представляют собой, по сути, одну и ту же функцию с незначительной разницей выполнения для одного из конструкторов.

2.4.2. Scrap your Boilerplate

Обобщенные функции были введены группой исследователей Microsoft Research и описаны в цикле статей Scrap your Boilerplate [4], [5], [6]. Суть подхода заключается в том, чтобы применить функцию ко всем требуемым аргументам типа, при этом учитывая, что тип может представлять собой древовидную структуру.

Такой подход позволил упростить написание функций путем написания отдельной функции обхода, которая принимает функцию преоб-

разования дерева типа и сам тип в качестве параметров и применяет её ко всем данным в этом дереве.

Недостаток подхода заключается в том, что тип невозможно расширить.

2.4.3. Наследование классов и переопределение поведения методов

Одной из из основных компонент объектно-ориентированный парадигмы программирование является возможность обеспечения специфического поведения метода подкласса относительно аналогичного метода суперкласса путем его перегрузки без потери поведения остальных методов.

Данный подход позволяет упростить изменение обработки типов путем изменения переопределенного метода, что позволяет уменьшить количество написанного кода.

Недостаток подхода заключается в том, что тип преобразуемой страктуры данных невозможно расширить новыми конструкторами.

2.4.4. Описание библиотеки

Библиотека Generic-transformers [7] является расширением синтаксиса camlp5[3]. Данная библиотека совмещает принципы обобщенного программирования заложенные в объектно-ориентированной и функциональной парадигмах программирования.

Основными компонентами данной библиотеки являются наличие класса-преобразователя, методы которого соответствуют конструкторам типа и преобразуют каждый отдельный конструктор в заданном виде и функция обхода, которая генерируется автоматически, на основе заданных преобразований кождой типовой переменной, входящей в состав типа с учетом атрибутных преобразований.

Такой подход удобен тем, что позволяет описать в общем виде преобразование типов, а при вызове функции трансформации указать как

[3]http://camlp5.gforge.inria.fr/

именно преобразовывать отдельные компоненты типа, при этом сохраняется контроль над количеством конструкторов типа.

3. Реализация

3.1. Принцип реализации компилятора Oberon0

По причине модульности языка Oberon0 было решено выполнять работу в следующей последовательности:

1. описать базовые конструкции, используемые в уровне языка L1;
2. описать конструкции, используемые в последующих уровнях языка путем дополнения ранее объявленных конструкций;
3. реализовать обобщенные преобразователи для операций уровня языка L1;
4. реализовать обобщенные преобразователи в последующих уровнях языка путем описания новых конструкций языка и наследованием преобразователей старых;
5. для каждого уровня языка Oberon0 использовать преобразователь требуемого типа.

4. Результаты

В процессе выполнения курсовой работы были достигнуты следующие результаты.

- Изучены основные подходы обобщенного программирования. Изучена библиотека Generic-transformers.
- Реализован L1 уровень компилятора Oberon-0 с использованием библиотеки Generic-transformers.
- Найдены ошибки в некоторых функциях библиотеки Generic-transformers.

Список литературы

[1] Niklaus Wirth. Compiler construction. International computer science series. Addison-Wesley, 1996.

[2] Phillip Wadler. Expression Problem. // http://homepages.inf.ed.ac.uk/wadler/papers/expression/expression.txt

[3] Jacques Garrigue. Code Reuse Through Polymorphic Variants. FOSE-2000.

[4] Ralf L̈ammel, Simon Peyton Jones. Scrap your Boilerplate: A practical Pattern for Generic Programming. Workshop on Types in Language Design and Implementation, 2003.

[5] Ralf L̈ammel, Simon Peyton Jones. Scrap you Boilerplate: Reflection, Zips, Generalised Casts. ICFP-2004.

[6] Ralf L̈ammel, Simon Peyton Jones. Scrap Your Boilerplate with Class: Generic Functions. ICFP-2005.

[7] Boulytchev Dmitri. Code Reuse With Transformation Objects. // http://oops.math.spbu.ru/papers/transformation-objects.pdf

[8] Boulytchev Dmitri. Компонентизация языковых процессоров на основе расширяемых типов данных и управляемых ими преобразователей // http://www.sysprog.info/2012/04.pdf