Модель концептов в императивном языке программирования

Ю. В. Белякова

Научный руководитель: доцент каф. АДМ, к. ф.-м. н. С. С. Михалкович

Рецензент: доцент каф. ИВЭ, к. ф.-м. н. Д. В. Дубров

Направление 010300 — Фундаментальная информатика
и информационные технологии

Факультет математики, механики и компьютерных наук
Южный федеральный университет

11 июня 2014 г.

Содержание

Содержание

Задачи

- Сравнительный анализ концептов и средств обобщённого программирования в языке C#.
- Проектирование дизайна концептов (описание синтаксиса и семантики конструкций, относящихся к механизму обобщённого программирования на основе концептов) для подмножества императивных объектно-ориентированных .NET-языков.
- Построение некоторых алгоритмов семантического анализа кода, использующего механизм концептов.
- Разработка модели перевода кода с концептами в базовый язык программирования.
- Программная реализация алгоритма унификации ограничений.

Содержание

Обобщённое программирование на основе явных ограничений

	C++	SML	OCaml	Haskell	Eiffel	Java	C#	Cecil
Multi-type concepts	-	●	○	●*	○	○	○	◒
Multiple constraints	-	◒	◒	●	○†	●	●	●
Associated type access	●	●	◒	●⋆	◒	◒	◒	◒
Constraints on assoc. types	-	●	●	●	◒	◒	◒	●
Retroactive modeling	-	●	●	●	○	○	◒	●
Type aliases	●	●	●	●	○	○	○	○
Separate compilation	○	●	◒	●	●	●	●	◒
Implicit arg. deduction	●	○	●	●	○	●	◒	◒

*Using the multi-parameter type class extension to Haskell (Peyton Jones *et al.*, 1997).
⋆Using the functional dependencies extension to Haskell (Jones, 2000).
†Planned language additions.

Table 1: *The level of support for important properties for generic programming in the evaluated languages. A black circle indicates full support, a white circle indicates poor support, and a half-filled circle indicates partial support. The rating of "-" in the C++ column indicates that C++ does not explicitly support the feature, but one can still program as if the feature were supported due to the permissiveness of C++ templates.*

Рис. 1: Существенные характеристики обобщённого программирования [CmpStudy, 2007]

Концепты — замена ограничений-интерфейсов

Концепты были предложены для C++ [CppConcepts, 2006] как замена шаблонов. Концепты — механизм явных ограничений. Поддерживают все основные характеристики на Рис. 1.

Концепты можно адаптировать к .NET-языкам.

Зачем это нужно?

Концепты устраняют недостатки использования интерфейсов в качестве ограничений на типовые параметры.

Основное отличие .NET концептов от других дизайнов (C++ [CppConcepts, 2006], [ConceptGCC, 2006], G [Siek, 2005]) — поддержка *ограничений подтипирования*.

Сравнение с G и C# с расширением

Возможность	G	C#-ext	.NET-concepts
Ограничения на несколько типов	+	−	+
Ограничения подтипирования	−	+	+
Адаптация типа	+	−	+
Доступ к содержимому концепта «по точке»	−	+	+
Перегрузка на основе ограничений	+	−	−
Автоматическое построение моделей	−	−	+

Рис. 2: Основные различия в дизайне концептов G, расширенных интерфейсов C# [CSharp-Jarvi, 2005] и .NET концептов

Содержание

Унификация ограничений

Концепты могут содержать противоречивые ограничения на типы. Проверка непротиворечивости набора ограничений выполняется с помощью **алгоритма унификации**.

Унификация является основным этапом семантического анализа концептов, а также применяется:

- при анализе моделей;
- при инстанцировании обобщённых классов и методов конкретными типами;
- при анализе выражений, зависящих от обобщённых типов.

Алгоритм унификации

Основан на алгоритме унификации Хиндли-Милнера [tapl].
Базовый алгоритм применяется к набору ограничений *равенства*

$$\mathrm{S} = \mathrm{T}$$

Унификация .NET ограничений

Набор ограничений для .NET концептов включает ограничения *подтипирования*

$$\mathrm{S} <: \mathrm{T}$$

Требуется существенная модификация базового алгоритма и связанных понятий (типовые переменные снабжаются атрибутами-«границами» типа, накладываются ограничения на подстановку типов).

Общее описание

Назначение приложения

Унификация набора ограничений равенства и подтипирования.

Приложение — «мини-компилятор»:

- исходный текст «программы» задаёт набор ограничений (в ограничениях можно использовать стандартные классы или классы из пользовательских .dll);
- «компиляция» программы выполняет анализ и унификацию ограничений;
- итогом компиляции является информация, описывающая результат унификации.

Содержание

Преимущества трансляции .NET концептов

1. Благодаря использованию атрибутов, трансляция сохраняет всю исходную информацию. Это позволяет обеспечить **модульность**: скомпилированный модуль может быть подключён к проекту на расширенном языке и использован. В G и расширенном C# существенная часть информации теряется.
2. В отличие от G, концепт-требования переводятся в типовые параметры обобщённого кода, а не в дополнительные поля классов и параметры методов. Это снижает стоимость этапа выполнения.

Замечание. Трансляция в базовый язык — один из возможных способов реализации новых конструкций в компиляторе.

Содержание

Результаты работы

1. Разработан **дизайн концептов** для подмножества объектно-ориентированного .NET-языка C#.
2. Построен **алгоритм унификации ограничений**.
3. Доказана **завершимость** алгоритма унификации.
4. Разработана **концепция трансляции** кода с концептами в базовый язык и восстановления из него.
5. Построены некоторые алгоритмы семантического анализа и перевода в базовый язык.
6. Выполнена **программная реализация алгоритма унификации**.

Содержание

Список литературы I

Ronald Garcia, Jaakko Järvi, Andrew Lumsdaine, Jeremy Siek, Jeremiah Willcock.
An Extended Comparative Study of Language Support for Generic Programming.
Journal of Functional Programming, Volume 17, Issue 2, March 2007.

Jeremy J. Siek
A Language for Generic Programming.
Doctotal Dissertation. Indiana University, Computer Science, 2005.

Bjarne Stroustrup.
The C++0x "Remove Concepts"Decision
July 22, 2009.
http://www.drdobbs.com/cpp/the-c0x-remove-concepts-decision/218600111

Gabriel Dos Reis, Bjarne Stroustrup
Specifying C++ concepts.
Proceedings of the 2006 POPL Conference, Volume 41 Issue 1, January 2006.

Douglas Gregor, Jaakko Järvi, Jeremy Siek, Bjarne Stroustrup, Gabriel Dos Reis, Andrew Lumsdaine
Concepts: linguistic support for generic programming in C++.
Proceedings of the 2006 OOPSLA Conference, Volume 41 Issue 10, October 2006.

Список литературы II

Jaakko Järvi, Jeremiah Willcock, Andrew Lumsdaine.
Associated Types and Constraint Propagation for Mainstream Object-Oriented Generics.
OOPSLA '05 Proceedings of the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, Volume 40 Issue 10, October 2005.

Бенджамин Пирс.
Типы в языках программирования
М. : Изд-во ДМК, 2012.