Министерство образования и науки Российской Федерации
Московский физико-технический институт (государственный университет)

Физтех-школа радиотехники и компьютерных технологий
Кафедра микропроцессорных технологий в интеллектуальных системах управления
Лаборатория (OS LAB)

Выпускная квалификационная работа магистра

Поддержка union типов в статическом языке программирования

Автор:
Студент M01-206 группы
Акмаев Алексей Михайлович

Научный руководитель:
Добров Андрей Дмитриевич

Научный консультант:
Бронников Георгий Кириллович

Москва 2024

Аннотация

Поддержка union типов в статическом языке программирования
Акмаев Алексей Михайлович

В данной работе исследуются способы поддержки union типов в статическом языке программирования, а также варианты генерации байткода для union типов и его оптимизации.

Работа включает: todo

Ключевые слова: union типы; компилятор; MyTS; байткод; нормализация;

Цель работы: реализация union типов в статическом TS-подобном языке программирования с дальнейшей оптимизацией байткода.

Задачи:

- Реализация базовых union типов
- Нормализация union типов
- Поддержка доступа к полям
- Внедрение литералов в union типы
- Написание lowering фаз для корректной кодогенерации
- Оптимизация байткода

Abstract

Support for union types in a static programming language

Содержание

1 Введение **4**

2 Постановка задачи **6**

3 Обзор существующих решений **7**

4 Исследование и построение решения задачи **8**

5 Описание практической части **9**

6 Заключение **10**

Приложение **12**

1 Введение

Известно, что разработка хороших, многоразово используемых библиотек является очень сложной задачей. В популярных статических объектно-ориентированных языках, таких как Java и C++, наследование и подтипизация (а в последнее время и обобщения в том числе) используются в качестве основных механизмов, способствующих многоразовому использованию кода. В то время как наследование позволяет одному классу повторно использовать реализацию другого класса, например, объявления переменных и сигнатур методов, подтипирование предназначено для взаимозаменяемости. Под взаимозаменяемостью подразумевается такое свойство сущности, что если объект одного типа может быть использован в определенном месте, то и другой объект, являющийся его подтипом, также может быть использован в том же месте. Взаимозаменяемость может быть также перефразирована как возможность повторного использования контекстов в том смысле, что если некоторый контекст применим к объекту одного типа, то тот же контекст также применим к любому объекту его подтипа. Для ясности дадим определения подтипа и супертипа. Подтип - это тип, являющийся производным от другого типа, который называют супертипом. Подтип наследует свойства и поведение своего супертипа, но также может добавлять дополнительные свойства или переопределять существующие. Подтипирование - это способ выразить, что один тип является специализированной версией другого. Супертип предоставляет общее определение, которое может быть расширено или специализировано с помощью его подтипов. Таким образом, проблемы проектирования, связанные с отношениями наследования и подтипирования, несколько различаются: для наследования необходимо учитывать, как новые классы могут повторно использовать существующую реализацию, а для подтипирования - как объекты могут использоваться в клиентском коде.

В популярных языках связь между подтипами по большей части основана на отношениях наследования. Исключением являются только wildcards в Java 5.0. Может случиться так, что два класса, используемые в схожих контекстах, но с довольно разными реализациями, будут разделены в иерархии классов наследования, что приведет к отсутствию полезного супертипа этих классов. Интерфейсы, как программная конструкция, в Java являются решением этой проблемы: можно определить суперинтерфейс классов схожего назначения, независимо от заданной иерархии наследования, и пользоваться преимуществами подтипирования. Однако интерфейсы не могут быть добавлены после определения класса, поэтому разработчикам библиотек по-прежнему приходится много работать над планированием иерархий интерфейсов перед выпуском библиотеки в релиз. Эта проблема считается существенным ограничением систем типов, основанными на взаимозаменяемости, как в Java.

В этой работе предлагается решение — объединение или union типы. Union типы или объединения - это тип данных в некоторых языках программирования, позволяющий переменной хранить значение, которое мо-

Рис. 1: Общая схема исполнения программы

жет быть одним из нескольких различных, в том числе несвязанных между собой наследованием, но фиксированных типов. Только один из типов, входящих в объединение, может быть ассоциирован с переменной в рантайме в конкретный момент времени. Они позволят решить проблему невозможности добавления супертипов к существующим типам, таким как классы и интерфейсы.

На практике различают два вида объединения - тегированный и нетегированный union. Нетегированное объединение можно представить как фрагмент памяти, который используется для хранения переменных разных типов данных. Как только ему присваивается новое значение, существующие данные перезаписываются новыми данными. Область памяти, в которой хранится значение, не имеет внутреннего типа (кроме просто байтов или слов памяти). Однако это значение можно рассматривать как один из нескольких абстрактных типов данных, имеющий тип значения, которое было последним записано в область памяти. Нетегированные объединения обычно довольно ограничены в использовании и представлены только в не типобезопасных языках программировании, таких как C. Тегированное объединение можно рассматривать как тип с несколькими компонентами, каждая из которых должна быть корректно обработана при манипулировании этим типом. Говоря об объединении, по умолчанию будет подразумеваться тегированный union.

Поскольку объединения состоят из существующих типов, они дают возможность определять супертип даже после того, как иерархия классов установлена. Как следует из названия, тип объединения обозначает объединение множества заданных типов, рассматриваемых как наборы экземпляров, которые принадлежат к этим типам, и на уровне байткода и рантайма ведут себя как наименьший супертип. Типы объединений могут использоваться не только при анализе вариантов, как в ML-типах данных, но и при прямом доступе к элементам, как обычные типы. Фактически, для некоторых типов их объединяющий тип можно рассматривать как интерфейс, который "выделяет"их общие черты, то есть поля с одинаковыми именами и методы с похожими сигнатурами. Мы ожидаем, что объединяющие типы могут быть полезны для группировки независимо разработанные классы со схожими интерфейсами, с указанием их супертипа, а также для реализации гетерогенных коллекций, таких как списки, в которых, скажем, строки и целые числа смешаны в качестве элементов.

todo: описать боксинг анбоксинг

2 Постановка задачи

Необходимо формально изложить суть задачи в данной секции, предоставив такие ясные и точные описания, которые позволят в последующем оценить, насколько разработанное решение соответствует поставленной задаче. Текст главы должен следовать структуре технического задания, включая как описание самой задачи, так и набор требований к ее решению.

3 Обзор существующих решений

Здесь надо рассмотреть все существующие решения поставленной задачи, но не просто пересказать, в чем там дело, а оценить степень их соответствия тем ограничениям, которые были сформулированы в постановке задачи.

4 Исследование и построение решения задачи

Требуется разбить большую задачу, описанную в постановке, на более мелкие подзадачи. Процесс декомпозиции следует продолжать до тех пор, пока подзадачи не станут достаточно простыми для решения непосредственно. Это может быть достигнуто, например, путем проведения эксперимента, доказательства теоремы или поиска готового решения.

5 Описание практической части

Если в рамках работы писался какой-то код, здесь должно быть его описание: выбранный язык и библиотеки и мотивы выбора, архитектура, схема функционирования, теоретическая сложность алгоритма, характеристики функционирования (скорость/память).

6 Заключение

Здесь надо перечислить все результаты, полученные в ходе работы. Из текста должно быть понятно, в какой мере решена поставленная задача.

Список литературы

[1] *Mott-Smith, H.* The theory of collectors in gaseous discharges / H. Mott-Smith, I. Langmuir // *Phys. Rev.* — 1926. — Vol. 28.

[2] *Морз, Р.* Бесстолкновительный PIC-метод / Р. Морз // Вычислительные методы в физике плазмы / Ed. by Б. Олдера, С. Фернбаха, М. Ротенберга. — М.: Мир, 1974.

[3] *Киселёв, А. А.* Численное моделирование захвата ионов бесстолкновительной плазмы электрическим полем поглощающей сферы / А. А. Киселёв, Долгоносов М. С., Красовский В. Л. // Девятая ежегодная конференция «Физика плазмы в Солнечной системе». — 2014.

Приложение

Здесь необходимо написать приложение, которое вы должны придумать самостоятельно.