УДК 004.415.52:004.432.42

ИСПОЛЬЗОВАНИЕ СТАТИЧЕСКОЙ ТИПИЗАЦИИ ДЛЯ СТАТИЧЕСКОГО АНАЛИЗА ДИНАМИЧЕСКИХ ЯЗЫКОВ

Доморацкий Э. А. (ИТМО)
Научный руководитель – к. ф.-м. н., доцент Булычев Д. Ю. (СПбГУ)

Введение. Бестиповые и динамические языки программирования (такие как Python, LISP, Tcl и др.) обладают рядом преимуществ по сравнению со статически типизированными языками. Среди таких преимуществ можно выделить: простоту реализации, за счёт отсутствия лишних синтаксических конструкций и этапов компиляции; большую выразительную мощность, благодаря отсутствию ограничений системы типов; простоту использования и скорость разработки.

Тем не менее, при написании программ на таких языках, программист вынужден всегда быть уверенным в поведении программы, потому что компилятор или интерпретатор не в состоянии выявить их заранее, а в некоторых случаях, и во время возникновения ошибки. Из-за этого ошибка может проявляться спонтанно, неожиданно для программиста и пользователя программы, а также поиск ошибки может требовать значительного количества времени на отладку программы.

Одним из способов предупреждения таких ошибок является статический анализ программ. В данной работе продемонстрирован подход к статическому анализу, основанный на статической типизации с автоматическим выводом типов. Такой подход позволяет не расширять синтаксис языка для явного указания типовых аннотаций, как это сделано в популярных языках программирования (Python, TypeScript, Scheme [1] и др.). Вместо этого программа остаётся написана на исходном бестиповом языке, но статический анализатор самостоятельно выводит типы всех выражений. В работе подход применяется к активно развивающемуся учебному бестиповому компилируемому языку программирования LaMa, описанному в документе по ссылке: https://raw.githubusercontent.com/PLTools/Lama/1.20/lama-spec.pdf.

Основная часть.

Система типов. Первый этап работы – это разработка системы типов для языка, достаточно выразительной для практической применимости в реальных программах и достаточно простой для возможности автоматического вывода типов. Язык программирования LaMa разработан для того, чтобы быть достаточно выразительным для удобной разработки компилятора языка LaMa (так называемого «бутстрапинга» или «раскрутки компилятора»), поэтому выразительность системы типов должна быть как минимум достаточной для выполнения данной задачи. С другой стороны, для возможности автоматического вывода типов система должна обладать некоторыми свойствами, среди которых можно выделить «синтаксическую направленность» (от англ. «syntax-directed») правил типизации, благодаря которой выбор возможных правил для применения к определённому узла синтаксического дерева становится ограничен не более, чем одним конкретным правилом, зависящим от текущего узла. Для этого в разработанной системе типов применяются ограничения, устанавливающие требуемые соотношения между произвольными типами.

Вывод типов. Благодаря использованию ограничений, задача вывода типов для разработанной системы типов сводится к двум шагам, работающих поочерёдно: выводу ограничений и их решению. Вывод ограничений делается аналогично, описанному в статье [2], обходом синтаксического дерева и вызовом подпрограммы решения ограничений, которая отвечает за упрощение ограничений и определение несовместности ограничений.

Решение ограничений. Для решения ограничений используется реляционное программирование. Данный подход позволяет упростить реализацию алгоритма поиска решения и также интересен как один из способов автоматически выводить типы в языках программирования.

Выводы. Предложенный подход позволяет статически анализировать программы на бестиповом языке программирования для предупреждения ошибок без необходимости расширения языка специальными конструкциями для явной типизации за счёт автоматического вывода типов.

Список использованных источников:

1. Siek J., Taha W. Gradual typing for objects //European Conference on Object-Oriented Programming. – Berlin, Heidelberg : Springer Berlin Heidelberg, 2007. – С. 2-27.
2. Odersky M., Sulzmann M., Wehr M. Type inference with constrained types //Theory and practice of object systems. – 1999. – Т. 5. – №. 1. – С. 35-55.