Композициональные частично определенные типы в символьном интерпретаторе для платформы .NET Framework

Александр Мисонижник
группа 444
научный руководитель: ст. преп. Я. А. Кириленко
рецензент: Принстонский университет, постдок, кандидат наук, Г. Г. Федюкович
консультант: программист JetBrains Д. А. Мордвинов

Санкт-Петербургский государственный университет
Кафедра системного программирования

2018

Введение

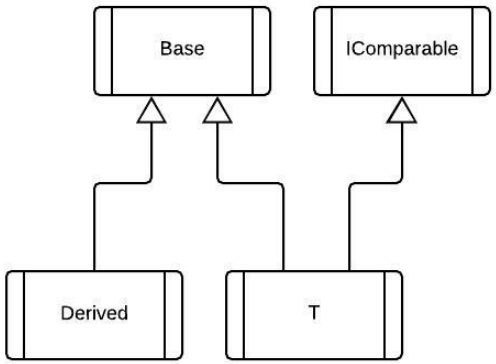
- *Символьное исполнение* — техника, которая позволяет моделировать исполнение кода, при котором часть входных данных остаётся неопределенной
- `V#` — это проект, направленный на создание символьного интерпретатора для платформы .NET

Введение

Пример

```
class Base {}
class Derived : Base {}

int GetResult<T>() where T : new ()
{
  T t = new T();
  return t is Base &&
         t is IComparable ?
         throw new Exception() : 42;
}
```

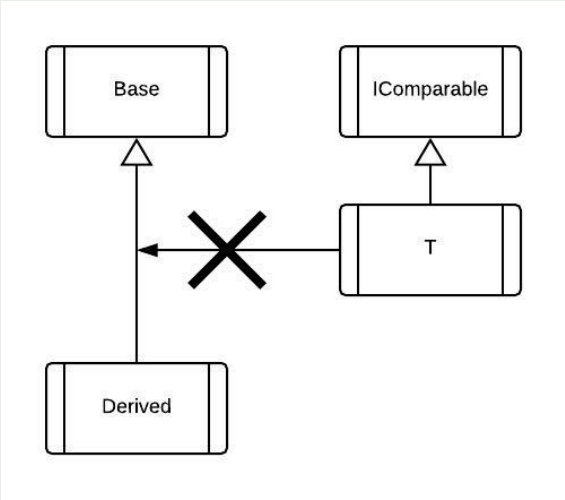


Введение

Пример

```
class Base {}
class Derived : Base {}

int GetResult<T>() where T : new ()
{
  T t = new T();
  Derived d = new Derived();
  return t is Base &&
         t is IComparable && d is T ?
         throw new Exception() : 42;
}
```



Введение

Пример

```
interface IComponent {}
interface IBuilder<out Z> where Z: IComponent {}
sealed class Symbol: IComponent, IBuilder<Symbol> {}
class Line: IComponent, IBuilder<Line> {}

int GetResult<X, Y>()
  where X: IComponent, IBuilder<Y>, new()
  where Y: IComponent, IBuilder<X>
{
  X x = new X();
  Line l = new Line();
  return x is IBuilder<Symbol> && x is Line &&
         l is Y ? throw new Exception() : 42;
}
```

Цель работы

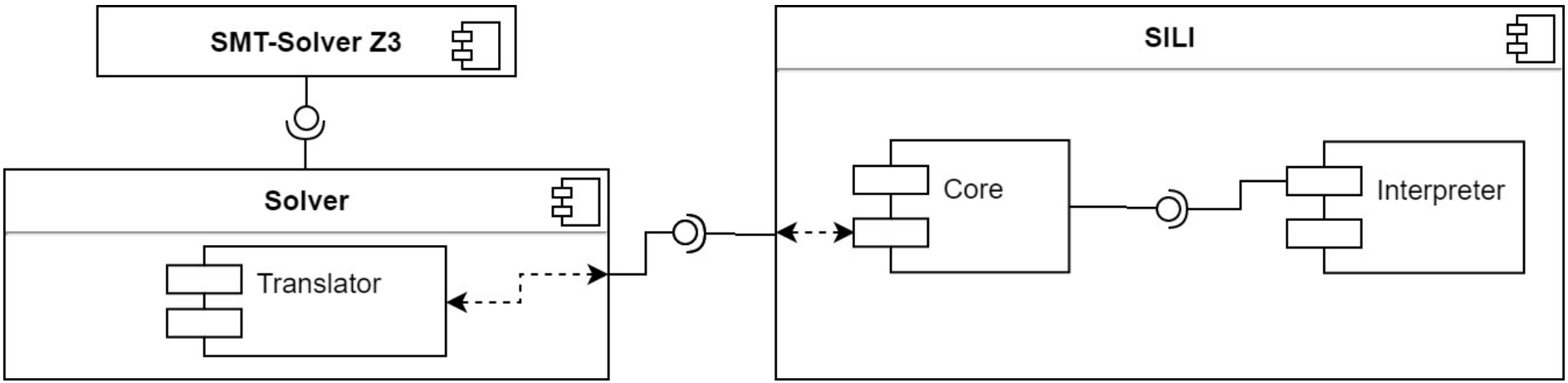
Целью работы является поддержка композиционного символьного исполнения .NET-кода в условиях неопределенности динамических типов объектов

Постановка задачи

В контексте данной работы были поставлены следующие задачи

- Спроектировать архитектуру подсистемы, отвечающей за интерпретацию типов
- Реализовать символьное исполнение функций с учетом открытых типов и полиморфизма, учитывая необходимость композициональности исполнения
- Разработать и реализовать алгоритм решения систем ограничений на типы
- Провести тестирование функциональности реализованной подсистемы

Архитектура подсистемы



Частично определенные типы

- Частично определенный тип отображает неопределенность динамического типа объекта
- Типовые ограничения на тип являются символьными булевыми термами
- Исполнение операторов, связанных с типами, выражается в терминах типовых ограничений
- Типовые ограничения могут конкретизироваться в путях исполнения

Подстановка типов

- Результат символьного исполнения — набор символьных термов и символьное состояние
- Операция взятия композиции в случае с типами является применением подстановки
- Для получения результата символьного исполнения остаётся лишь пересчитать результаты выполнимости условий пути

Ограничения на типы

Ограничения на типы

- Доказано, что задача выполнимости системы ограничений на .NET типы разрешима для моделей с некоторыми ограничениями
- Есть все шансы, что задача выполнимости системы ограничений на .NET типы разрешима в общем виде
- Полученный алгоритм является решателем бесквантворных формул в теории .NET типов
- Реализацию планируется выложить как расширение Z3 .NET API

Тестирование функциональности

- Для тестирования функциональности использовались интеграционные тесты для всей системы
- Код, реализующий описанную функциональность, был влит в основную ветку проекта `V#`

Результаты

- Спроектирована архитектура подсистемы, отвечающей за интерпретацию типов
- Реализовано символьное исполнение функций с учетом открытых типов и полиморфизма
- Разработан и реализован алгоритм, который решает ограничения на .NET типы
- Написана статья на конференцию SEIM 2018, с последующей публикацией в CEUR (индексируется в базе данных SCOPUS).
- Проведено тестирование функциональности реализованной подсистемы