Санкт-Петербургский государственный университет
Кафедра системного программирования

Обобщенные кандидаты при решении ограничений системы типов .NET

Вячеслав Андреевич Бучин, 21.Б15-мм

Научный руководитель: Д.А. Мордвинов, кандидат физико-математических наук
Консультант: М.П. Костицын, инженер-исследователь кафедры системного программирования

Санкт-Петербург
2022

Символьное исполнение

Символьное исполнение — техника, которая позволяет моделировать исполнение программного кода в условиях неопределённости входных данных

```
public static void Foo(int i, IDictionary<int, int> d)
{
    if (i >= 106)
        d.Add(i, 42);
    else
        d.Add(i, 0);
}
```

Ограничения

```
public static void Foo(
    int i,
    IDictionary<int, int> d)
{
    if (i >= 106)
        d.Add(i, 42);
    else
        d.Add(i, 0);
}
```

Ограничения

```
public static void Foo(
    int i,
    IDictionary<int, int> d)
{
    if (i >= 106)
        d.Add(i, 42);
    else
        d.Add(i, 0);
}
```

Ограничения:

1. $\mathtt{i} <: \mathtt{int}$

Ограничения

```
public static void Foo(
    int i,
    IDictionary<int, int> d)
{
    if (i >= 106)
        d.Add(i, 42);
    else
        d.Add(i, 0);
}
```

Ограничения:

1. i <: int
2. d <: IDictionary<int, int>

Ограничения

```
public static void Foo(
    int i,
    IDictionary<int, int> d)
{
    if (i >= 106)
        d.Add(i, 42);
    else
        d.Add(i, 0);
}
```

Ограничения:

1. `i` $<:$ `int`
2. `d` $<:$ `IDictionary<int, int>`
3. Ветвление:
   1. $i \geq 106$ — для ветки «`then`»
   2. $i < 106$ — для ветки «`else`»

Ограничения

```
public static void Foo(
    int i,
    IDictionary<int, int> d)
{
    if (i >= 106)
        d.Add(i, 42);
    else
        d.Add(i, 0);
}
```

Ограничения:

1. `i` $<:$ `int`
2. `d` $<:$ `IDictionary<int, int>`
3. Ветвление:
   1. $i \geq 106$ — для ветки «`then`»
   2. $i < 106$ — для ветки «`else`»

Кандидаты — типы, подходящие под ограничения, возникающие во время работы символьного исполнения

Цель и задачи

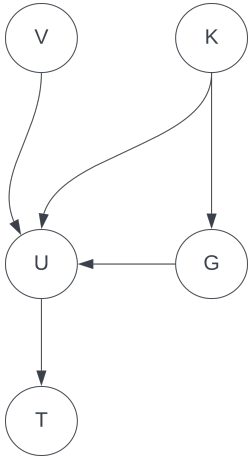
Целью учебной практики является увеличение покрытия символьного исполнения

Задачи

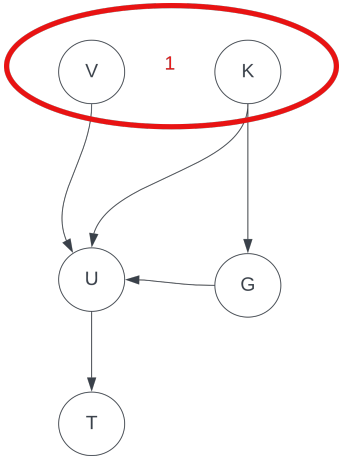
- выполнить обзор системы типов .NET
- разработать систему, позволяющую решателю типов использовать обобщенные типы
- реализовать эту систему в рамках проекта V#
- провести тестирование реализованной системы

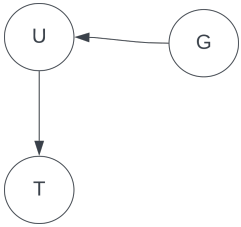
Граф зависимостей

```
public class GenericClass<T, U, V, K, G>
    where T : U
    where U : IEnumerable<V>, IDictionary<K, G>
    where K : G
{
}
```

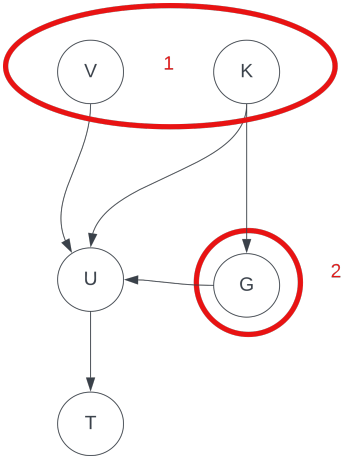


Деление на слои

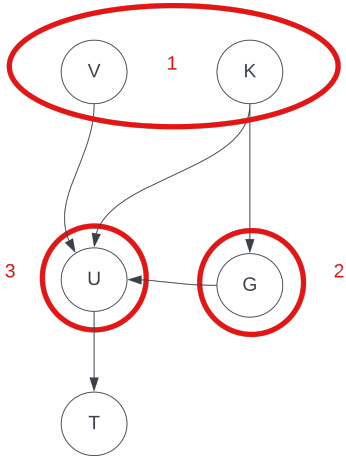




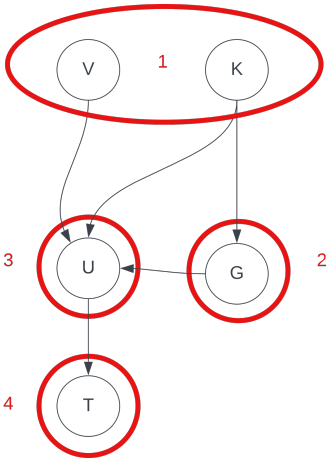
Деление на слои



Деление на слои



Деление на слои



Проталкивание ограничений

Кандидат C = `List<T>`

Приходят ограничения

1. `C` <: `IEnumerable<object>`

Получаемые ограничения на `T`

```
public class List<T> : IEnumerable<T> ... { }
public interface IEnumerable<out T> { }
```

Проталкивание ограничений

Кандидат $C =$ `List<T>`

Приходят ограничения

1. `C` $<:$ `IEnumerable<object>`

Получаемые ограничения на `T`

1. `T` $<:$ `object`

```
public class List<T> : IEnumerable<T> ... { }
public interface IEnumerable<out T> { }
```

Проталкивание ограничений

Кандидат $C =$ `List<T>`

Приходят ограничения

1. `C` $<:$ `IEnumerable<object>`
2. `C` $<:$ `List<string>`

Получаемые ограничения на `T`

1. `T` $<:$ `object`

```
public class List<T> : IEnumerable<T> ... { }
public interface IEnumerable<out T> { }
```

Проталкивание ограничений

Кандидат $C =$ `List<T>`

Приходят ограничения

1. `C` $<:$ `IEnumerable<object>`
2. `C` $<:$ `List<string>`

Получаемые ограничения на `T`

1. `T` $<:$ `object`
2. `T` $<:$ `string`
3. `T` $:>$ `string`

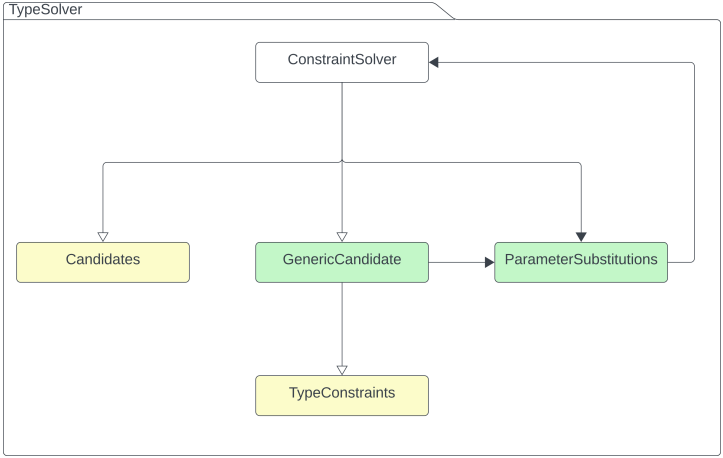
```
public class List<T> : IEnumerable<T> ... { }
public interface IEnumerable<out T> { }
```

Формализация алгоритмов

function trackIndices(t, s, indices)
    $td\langle\langle p_1, \ldots, p_n \rangle\rangle \leftarrow t$
    $sd\langle\langle sparams \rangle\rangle \leftarrow s$
    if $td = sd$ **then**
        return *indices*
    else
        $baseType \leftarrow t.baseType$
        $btd\langle\langle btparams \rangle\rangle \leftarrow baseType$
        $mapping \leftarrow$ `Dict()`
        for $p_i \in btparams$ **do**
            $mapping[p_i].append(i)$
        end for
        $track \leftarrow trackIndices(baseType, s, \{mapping[p_i] \mid p_i \in btparams\})$
        $traceback(x) \leftarrow \{track[ind] \mid i \in x, inds \in mapping[p_i], ind \in inds\}$
        return $\{\bigcup_{is \in traceback(inds)} is \mid inds \in indices\}$
    end if
end function

$layers \leftarrow$ `List<List<P>>()`
while $depsCount.size > 0$ **do**
    $zeroes \leftarrow \{p \mid depsCount[p] = 0\}$
    for $zero \in zeroes$ **do**
        $depsCount.remove(zero)$
        for $dep \in G[zero]$ **do**
            $depsCount[dep] \leftarrow depsCount[dep] - 1$
        end for
    end for
    $layers.append(zeroes)$
end while
return *layers*

Архитектура



Тестирование

Название группы тестов	Заглушки «Old», ед.	Заглушки «New», ед.	Покрытие «Old», %	Покрытие «New», %	Тестов пройдено «Old», %	Тестов пройдено «New», %	Тестов всего, ед.
MethodParameters	3	0	100	100	4	4	4
MockRelocation	1	0	50	100	1	2	2
DeepPropagating	4	0	66	100	2	3	3
Nested	1	0	35	100	1	2	2
NoSolution	4	3	100	100	2	2	2
GenericCandidates	8	0	66	100	2	3	3
NestedGenerics	0	0	100	100	2	2	2

Результаты

- выполнен обзор системы типов $.\mathrm{NET}$
- разработана система, позволяющая решателю типов использовать обобщенные типы
- эта система реализована в рамках проекта $\mathrm{V}\#$
- проведено тестирование реализованной системы

Ссылка на репозиторий: https://github.com/VSharp-team/VSharp