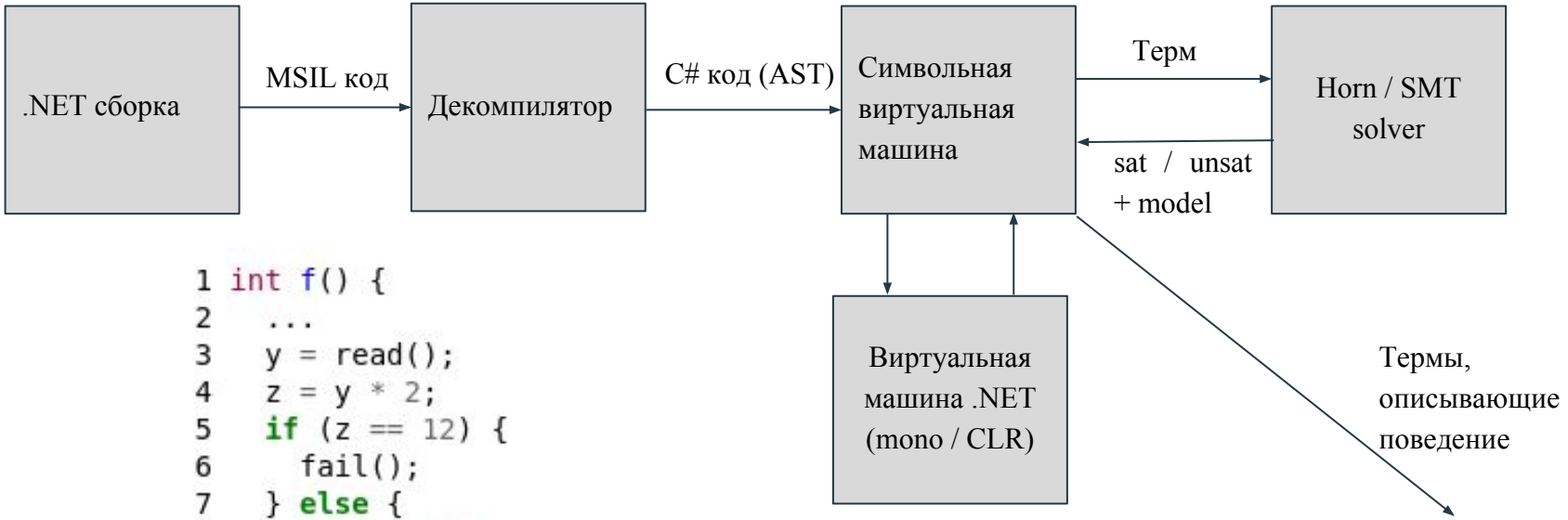
Символьное исполнение программ с указателями в .NET

Костюков Юрий, СПбГУ
Научный руководитель: Кириленко Я.А.

Символьная виртуальная машина .NET




```
1 int f() {
2   ...
3   y = read();
4   z = y * 2;
5   if (z == 12) {
6     fail();
7   } else {
8     printf("OK");
9   }
10 }
```

Область применения (проект в целом)

- Верификация
- Символьная отладка
- Символьное профилирование
- Порождение тестов
- Синтез программ
- …

Область применения unsafe

- mscorlib (.NET core runtime)
- Roslyn (.NET Compiler Platform)
- …

```
internal unsafe struct MetadataEnumResult
{
    // Keep the definition in sync with vm\ManagedMdImport.hpp
    private int[] largeResult;
    private int length;
    private fixed int smallResult[16];

    public int Length{...}

    public int this[int index]
    {
        get
        {
            if (largeResult != null)
                return largeResult[index];

            fixed (int* p = smallResult)
                return p[index];
        }
    }
}
```

Область применения unsafe

Взаимодействие с внешней средой:

- C/C++ код
- системные вызовы
- …

```c
#include <stdio.h>

void print(const char * message)
{
    printf("%s\\n", message);
}
```

```csharp
public unsafe class CallPrintfUnsafe
{
    [DllImport("libtest.so", EntryPoint="print")]
    private static extern void print(char* message);

    public static void MainFunc(string[] args)
    {
        string message = "Hello World C# => C++";
        fixed (char* p = message)
        {
            print(p);
        }
    }
}
```

Область применения unsafe

Ускорение кода (снижение накладных расходов)

Типичный код из mscorlib/system/array.cs:

```
// Make sure we're not out of range
if ( startIndex < 0 || startIndex >= array.Length) {
    throw new ArgumentOutOfRangeException("startIndex", Environment.Get
}
```

Глобальные задачи

Базовые:

- Арифметика указателей

```
public static int SimpleReinterpretation(double[] arr)
{
    fixed (double* p = arr)
    {
        int* q = (int*) p;
        return *(q + 3);
    }
}
```

Сложные:

- Реинтерпретация данных
  - Путём преобразования типов указателей
  - Путём отступа в известной среде (напр. внутри фрейма)
  - При помощи директив FieldOffset структур

Экзотические:

- Взаимодействие со средой (extern функции)
  - Отдельная область науки
- Указатели на замыкания (напр. различные фреймы рекурсивных функций)
  - Планируется в рамках работы по исследованию стека

Решённые задачи

Базовые:

- Арифметика указателей
- Корректная обработка артефактов декомпиляции

Реинтерпретация:

1. managed runtime sizeof
2. Проектирование композициональной модели памяти для структур

Арифметика указателей

1. Арифметика без потери информации
   a. на базе структуры данных “рюкзак”
2. Корректное сокращение символьной арифметики указателей

```
public static int SimplePointerDifference(int x, double y)
{
    int* p = &x;
    double* q = &y;
    long d = (double*) p - q;

    return * (int*) (q + d); // p
}

public static int PointerTriangle(int x, int y, int z)
{
    int* px = &x;
    int* py = &y;
    int* pz = &z;

    long d1 = px - py;
    long d2 = py - pz;

    int* r = pz + d1 + d2;

    return *r; // x
}
```

Артефакты декомпиляции

```
int* p = &x;
int* q = &y;
long d = p - q;

return *(q + d);
```

```
System.IntPtr num1 = (System.IntPtr) &x;
int* numPtr1 = &y;
int* numPtr2 = numPtr1;
long num2 = (long) ((num1 - (System.IntPtr) numPtr2) / 4);
return *(int*) ((System.IntPtr) numPtr1 + (System.IntPtr) (num2 * 4L));
```

```
int x = 428999;
int* p = &x;
return **&p;
```

```
return **&&428999;
```

sizeof

```
struct ArraysStruct
{
    public fixed int I_buf[20];
}

struct YetAnotherStruct
{
    public fixed double D_buf[4];
}

private static double Reinterpret(ArraysStruct[] s)
{
    fixed (ArraysStruct* p = &s[2])
    {
        return (*((YetAnotherStruct*) p + 3)).D_buf[4];
    }
}
```

sizeof

managed vs unmanaged:

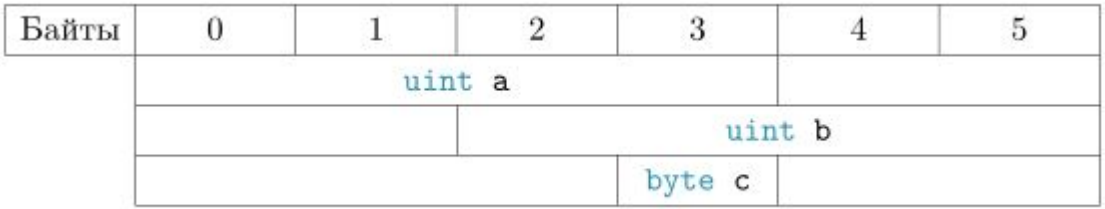
- `Marshal.SizeOf(typeof(char)) == 1`
- `sizeof(char) == 2`

	managed	unmanaged
runtime	*Наш sizeof*	Marshal.SizeOf
compile-time	sizeof<T>	-

Композициональная модель памяти структур

Базовая идея:

символьные структуры как дерево отрезков




```
[StructLayout(LayoutKind.Explicit)]
struct StrangeStruct
{
    [FieldOffset(0)]
    public uint a;
    [FieldOffset(2)]
    public uint b;
    [FieldOffset(3)]
    public byte c;
}
```

Результаты

1. Арифметика указателей без потери информации с символьным сокращением
2. Шаги к реинтерпретации:
   a. sizeof произвольного типа
   b. архитектура символьных структур
   c. (реинтерпретация — разбор случаев на базе этих решений)