Композициональное символьное исполнение CIL-кода

Константин Батоев
группа 444
научный руководитель: ст. преп. Д. А. Мордвинов

Санкт-Петербургский государственный университет
Кафедра системного программирования

13 июня 2019 г.

Символьное исполнение

- Исполнение кода над символьными значениями
- Для каждой точки программы хранится состояние – условие пути и отображение переменных в символьные значения
- Результат символьного исполнения – отображение входных значений в выходные в зависимости от условий

Пример

```
int f(int a, int b) {
    if (b < a || b > 2*a ) {
        return b;
    } else if(a <= b) {
        return a;
    } else {
        return a - b;
    }
}
```

Результат исполнения

$$res_f = \begin{cases} b & \text{if } b < a \vee b > 2a \\ a & \text{otherwise} \end{cases}$$

Композициональность

Идея переиспользования результатов символьного исполнения

Пример

```
int f(int a, int b) {
    if (b < a || b > 2*a ) {
        return b;
    } else if(a <= b) {
        return a;
    } else {
        return a - b;
    }
}
int g() {
    return f(10, 20);
}
```

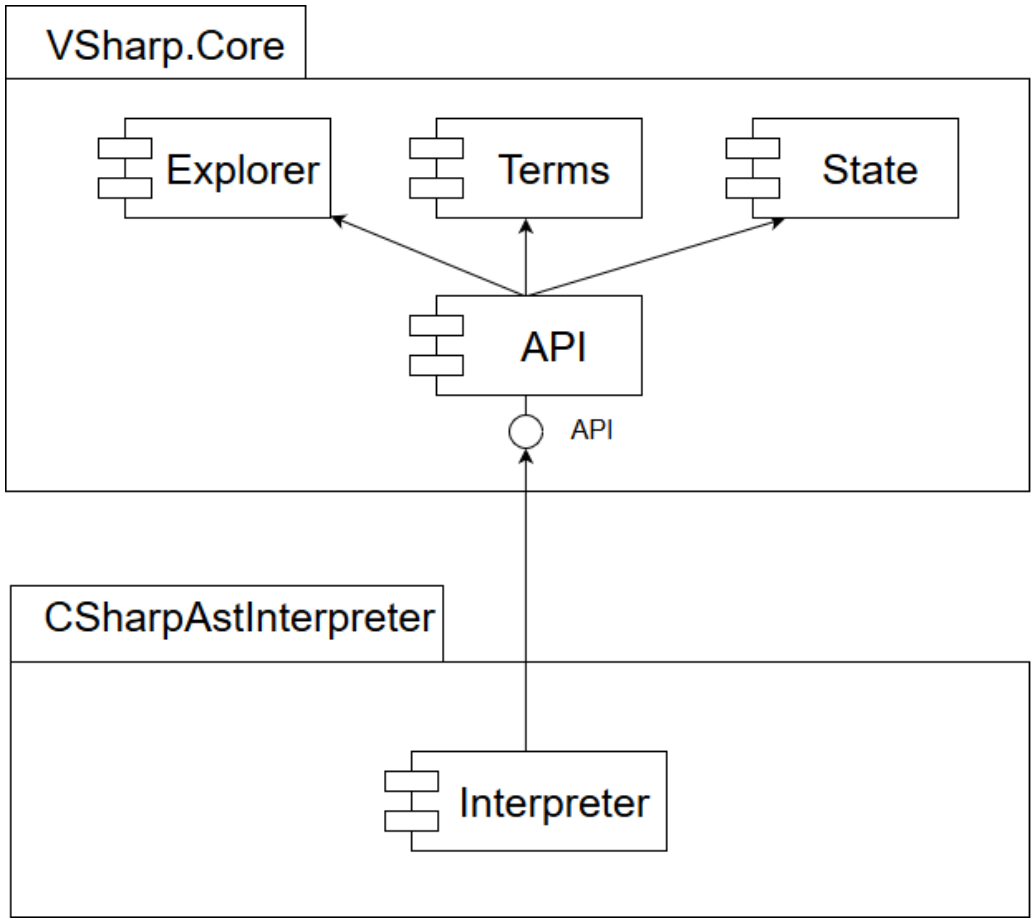
$$res_f = \begin{cases} b & \text{if } b < a \vee b > 2a \\ a & \text{otherwise} \end{cases}$$

$$\sigma = \{a \mapsto 10, b \mapsto 20\}$$

$$\sigma \circ res_f = 10$$

V#

- Символьная виртуальная машина для анализа .NET
- Инструмент статического композиционального исполнения без раскрутки отношения перехода



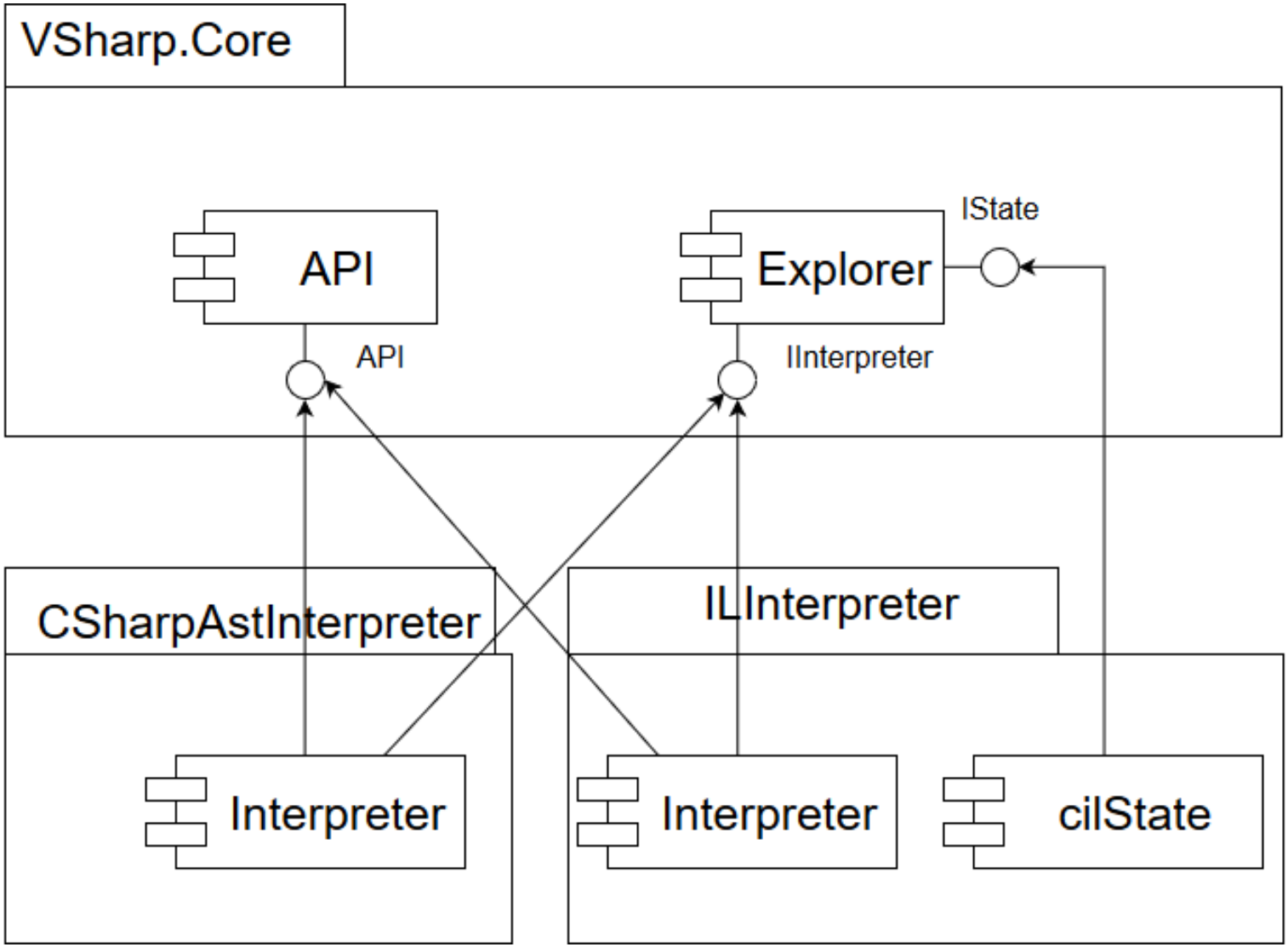
Цель

Реализовать символьный интерпретатор промежуточного языка CIL платформы .NET для символьной виртуальной машины V#

Постановка задачи

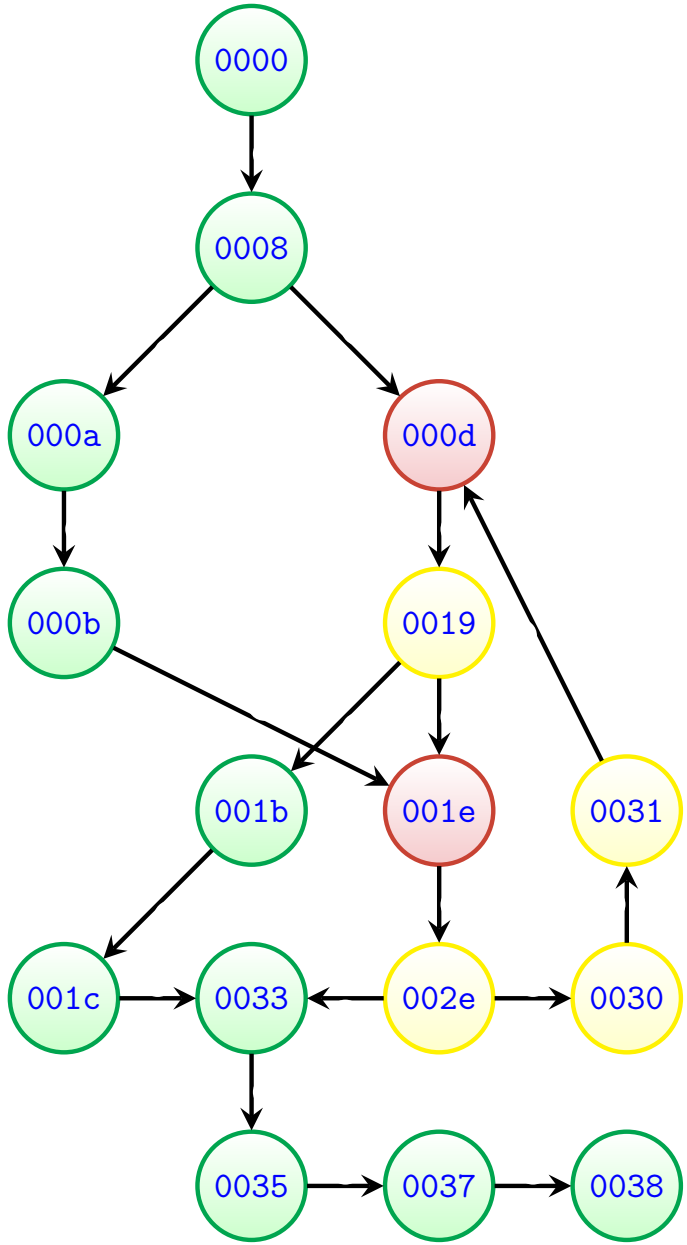
- Провести обзор алгоритмов интерпретации исходного кода для символьного исполнения
- Придумать алгоритм композиционального символьного исполнения
- Разработать архитектуру общей подсистемы интерпретации и реализовать интерпретатор CIL
- Провести апробацию интерпретатора

Архитектура



Произвольные графы потока управления

```
public static int Gotos(int x)
{
    if (x <= 10) {
        goto labelB;
    }
    labelA:
    x += 5;
    if (x >= 100) {
        goto exit;
    }
    labelB:
    x -= 2;
    if (x % 5 == 0) {
        goto labelA;
    }
    exit:
    return x;
}
```



Пути в графе

Введенные определения:

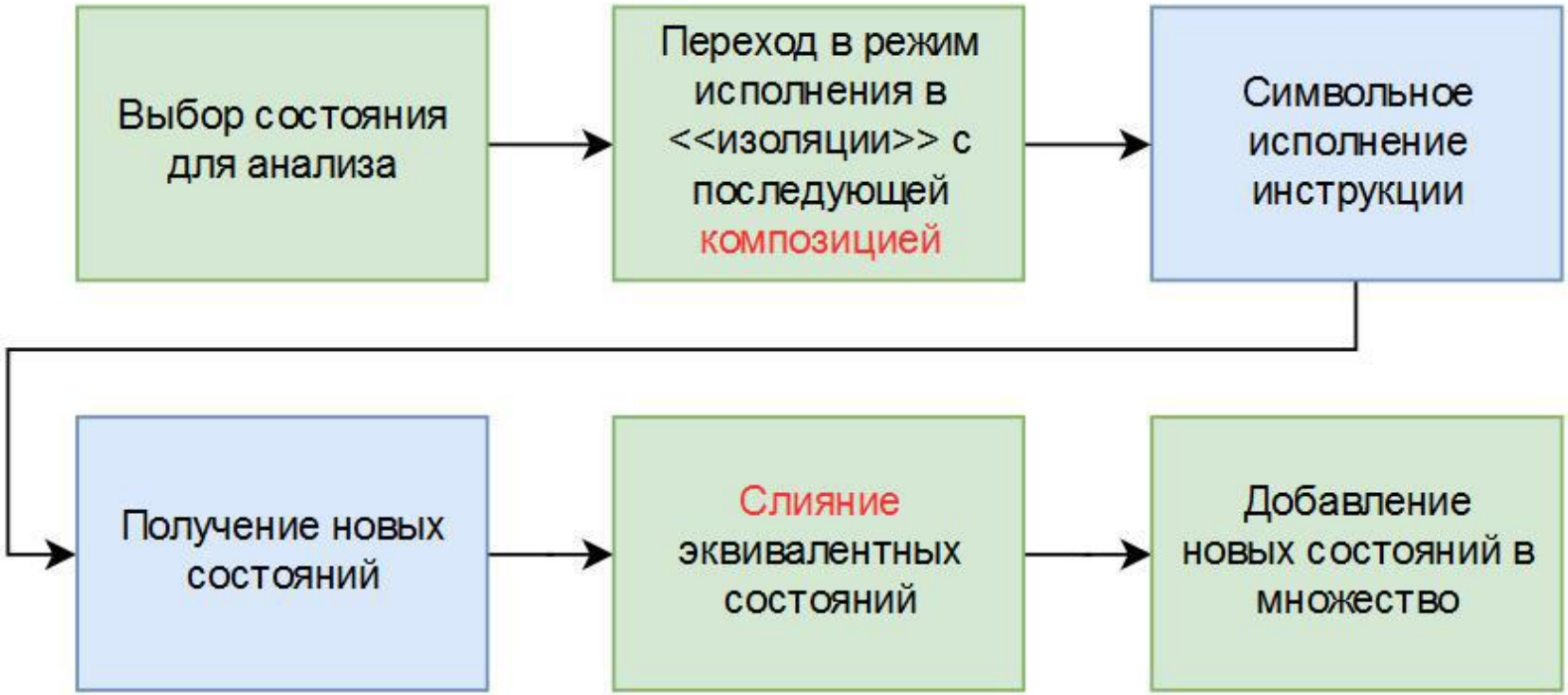
$$\Pi(u, v, D) = \bigcup_{(u,v) \in E_G} \{(u, v)\} \quad \cup$$
$$\bigcup_{\substack{t \notin RV \\ t \neq v \\ (u,t) \in E_G}} (u, t) \circ \Pi(t, v, D) \quad \cup$$
$$\bigcup_{\substack{t \in RV \\ t \neq v \\ (u,t) \in E_G \\ t \notin D}} (u, t) \circ Rec(t, D \cup \{t\}) \circ \Pi(t, v, D \cup \{t\})$$

$$Rec(u, D) = \Pi(u, u, D) \circ Rec(u, D) \cup \{\varepsilon\}$$

Теорема

$\Pi(Start, Exit, \varnothing)$ – все пути исполнения программы.

Схема алгоритма интерпретации



Детали реализации

Написаны модули:

- парсера бинарных программ
- построения графа потока управления
- единой схемы интерпретации в ядре проекта
- для интерпретатора языка CIL
  - символьного исполнения инструкций языка CIL
  - реализованы методы интерфейса IInterpreter для осуществления схемы обхода графа

Общие сведения:

- код написан на языке F#
- 4 KLOC

Апробация

Количественные характеристики тестов			
Название тестового набора	Количество тестов в наборе	Количество успешно пройденных тестов	Количество инструкций CIL
Arithmetics	80	77	1968
Logics	75	75	1458
Conditional	10	6	943
Recursive	5	5	751
Lambdas	2	2	404
Generic	14	14	194
Strings	15	15	219
Unsafe	16	16	312
Typecast	19	19	969
Methods	10	10	238
Lists	9	9	1420
Всего	255	248	8876

Таблица: Результаты тестирования

Результаты

- Проведен обзор алгоритмов интерпретации программ для символьного исполнения
- Разработан алгоритм композиционального символьного исполнения программы без раскрутки отношения перехода
- Реализованы интерпретатор CIL и общая архитектура интерпретаторов
- Поддержаны все тесты проекта V#, кроме тестов с исключениями