А.В. Макаров, С.Ю. Скоробогатов, А.М. Чеповский

Учебный курс “CIL и системное программирование в Microsoft .NET”

Лекция 13.

Граф потока управления

Введение

- Код метода в сборке .NET представляет собой линейную последовательность CIL-инструкций и массив описателей блоков обработки исключений
- Так как представленный в теле метода алгоритм в общем случае нелинейный, то есть содержит ветвления и циклы, то кодирование его в виде линейной последовательности требует определения семантики передачи управления от одной инструкции CIL к другой

Механизмы передачи управления между инструкциями

1. Явная передача управления с помощью инструкции перехода
2. Неявная (или естественная) передача управления на следующую инструкцию в последовательности
3. Передача управления на обработчик исключения при выходе (нормальном или аварийном) из защищенного блока
4. Передача управления между методами при вызове методов и при выходе из метода

Пример

```
.method private static int32  find(int32[] X, int32 k) {
            .locals init (int32 i, int32 result)
            .try {
            ldc.i4.0                [2]
            stloc.0                 [2]
            br.s        loop_cond   [1]
loop_body:  ldloc.0                 [2]
            ldc.i4.1                [2]
            add                     [2]
            stloc.0                 [2]
loop_cond:  ldarg.0                 [2]
            ldloc.0                 [2]
            ldelem.i4               [2]
            ldarg.1                 [2]
            bne.un.s    loop_body   [1]
            ldloc.0                 [2]
            stloc.1                 [2]
            leave.s     exit        [3]
            }
            catch System.IndexOutOfRangeException {
            pop                     [2]
            ldc.i4.m1               [2]
            stloc.1                 [2]
            leave.s     exit        [2]
            }
exit:       ldloc.1                 [2]
            ret                     [4]
}
```

Линейная последовательность инструкций – компактный способ представления кода

- Схема представления кода в виде линейной последовательности инструкций типична для ассемблерных языков и является наиболее компактной. Действительно, бóльшая часть кода метода состоит из линейных последовательностей инструкций с неявной передачей управления, а так как неявная передача определяется порядком следования инструкций и не требует дополнительного кодирования, то код занимает меньше места

Неудобство использования линейной последовательности инструкций

- Некоторые метаинструменты, выполняющие только анализ CIL-кода, могут непосредственно работать с линейной последовательностью инструкций
  - JIT-компиляторы, интерпретаторы, верификаторы и отладчики.
- Для метаинструментов, которые выполняют преобразование CIL-кода, такое представление неудобно:
  - Вставка и удаление инструкций из последовательности требует корректировки адресов в инструкциях перехода и в описателях блоков обработки исключений
  - Невозможно проводить генерацию сразу нескольких ветвей кода

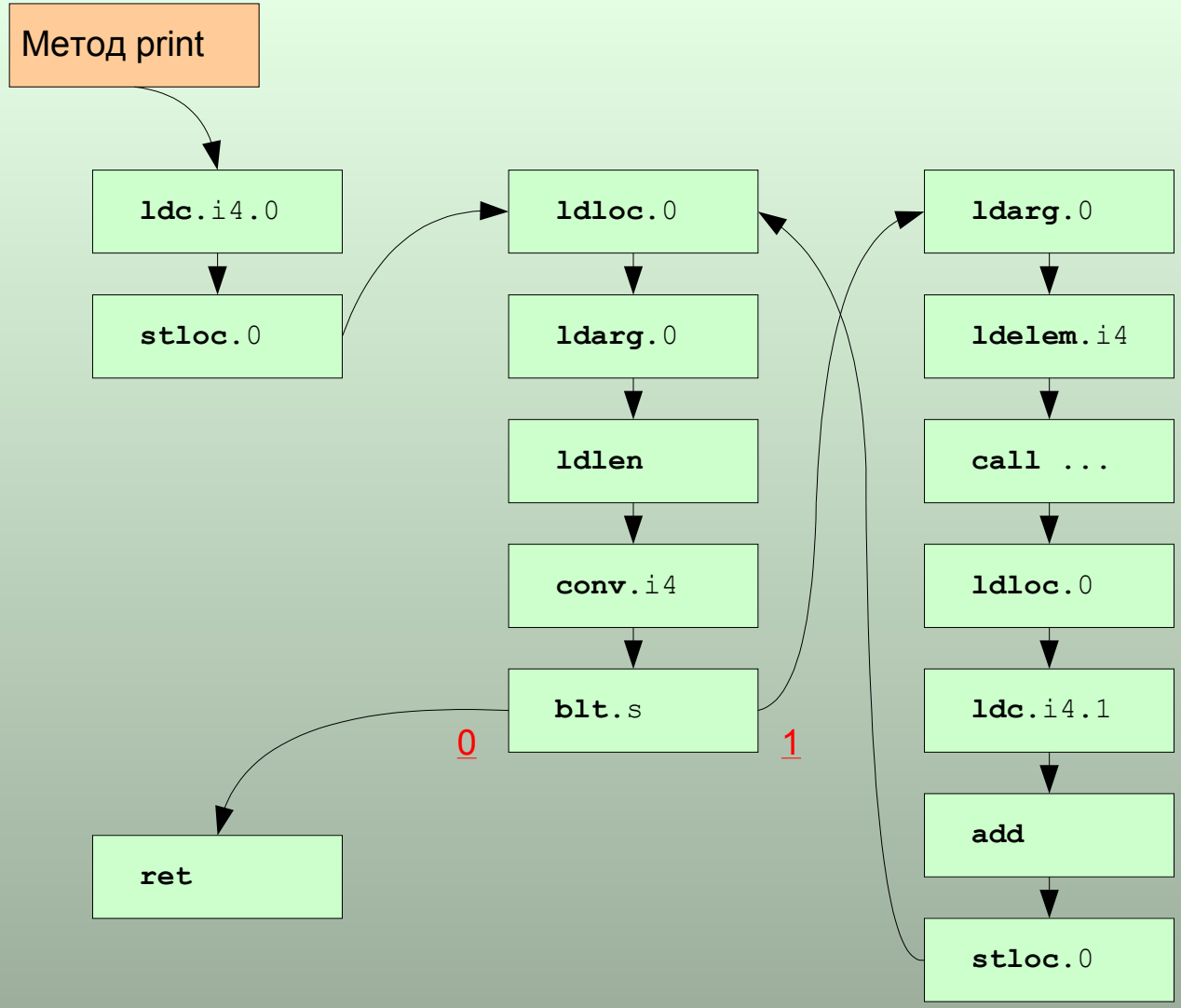
13.1. Структура графа потока управления

- Граф потока управления – это ориентированный граф, узлы которого соответствуют инструкциям CIL, а дуги изображают передачу управления между инструкциями

Пример (метод print)

```
.method private static void print(int32[] X) {
            .maxstack 2
            .locals init (int32 i)
            ldc.i4.0
            stloc.0
            br.s        loop_cond
loop_body:  ldarg.0
            ldloc.0
            ldelem.i4
            call        void System.Console::WriteLine(int32)
            ldloc.0
            ldc.i4.1
            add
            stloc.0
loop_cond:  ldloc.0
            ldarg.0
            ldlen
            conv.i4
            blt.s       loop_body
            ret
}
```

Пример (граф потока управления для метода print)



Варианты нумерации дуг графа

Вариант нумерации	Количество дуг	Семантика
Нумерация для инструкции switch	произвольное	Дуга с номером 0 обозначает передачу управления, которая происходит при “неудаче” (когда ни один из случаев, перечисленных в инструкции **`switch`**, не получил управления). Если в инструкции **`switch`** записано N переходов, то передачи управления для этих переходов обозначают дугами с номерами от 1 до N.
Нумерация для инструкции условного перехода	2	Дуги с номерами 0 и 1 обозначают соответственно false-ветку и true-ветку условного перехода.
Нумерация для последовательных инструкций	1	Дуга с номером 0 обозначает передачу управления на следующую инструкцию.
Нумерация для “тупиковых” инструкций	0	Из узлов, соответствующих некоторым инструкциям, связанным с выходом из блока (**`throw`**, **`rethrow`**, **`endfinally`**, **`endfilter`**, **`ret`**), вообще не исходит дуг.

13.2. Блоки обработки исключений в графе потока управления

- Как известно, блоки обработки исключений в CIL реализованы в виде массива описателей, который хранится отдельно от CIL-кода
- Для адекватного представления блоков обработки исключений нам придется добавить в граф потока управления специальные узлы, обозначающие входы в блоки, а также специальные дуги, отражающие взаимосвязи между ними
- Кроме того, мы обобщим понятие блока, введя так называемый блок тела метода

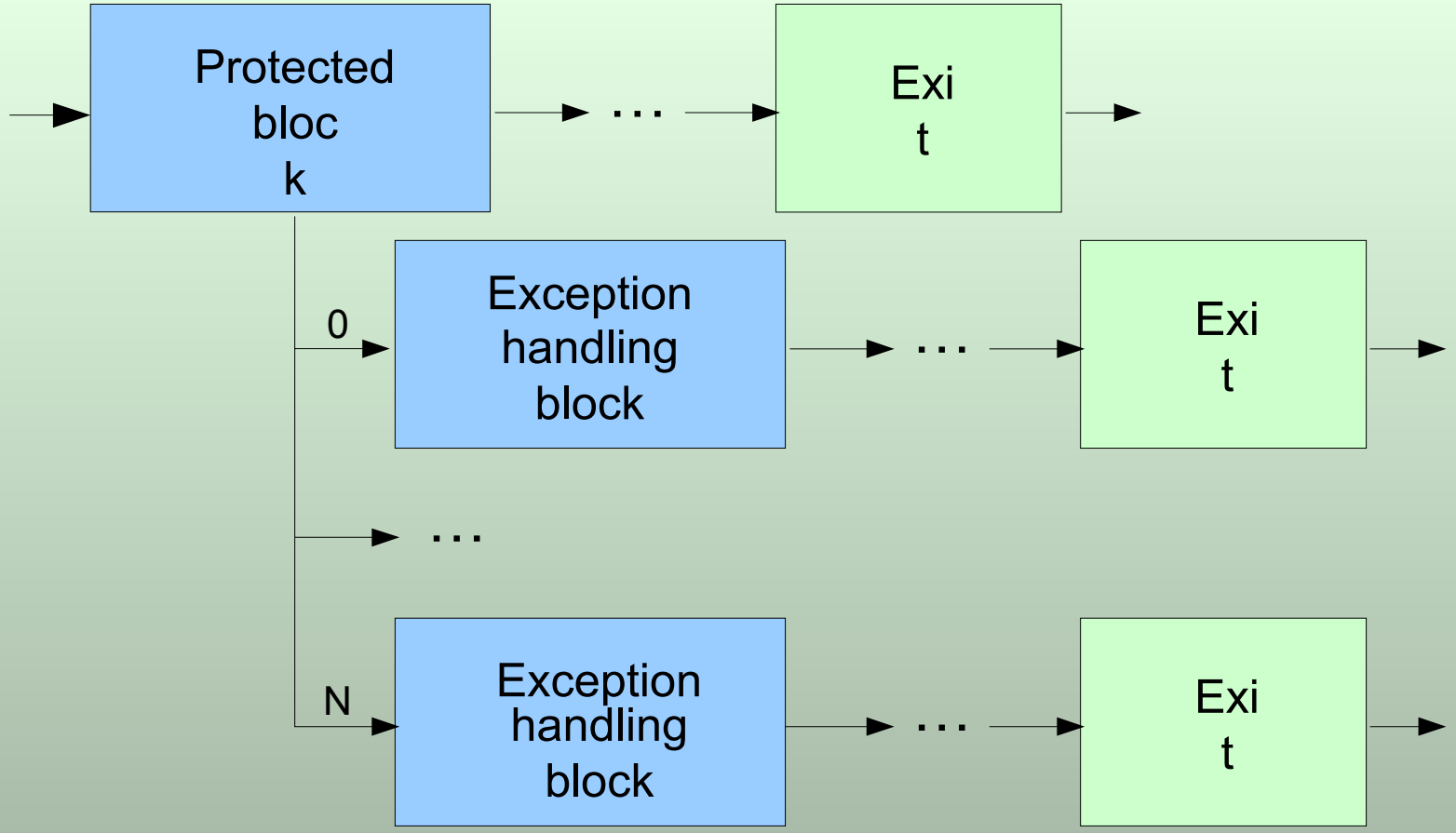
Категории блоков

- Блок тела метода. Этот блок содержится в графе в единственном экземпляре и задает точку входа в граф
- Защищенный блок – соответствует try-блоку в программе
- Блок обработки исключений – прикреплен к защищенному блоку и может получить управление при выходе из этого защищенного блока
- Блок фильтрации – прикреплен к блоку обработки исключений с пользовательской фильтрацией и осуществляет принятие решения о передачи управления на обработчик

Дополнительные дуги для присоединения блоков обработки исключений

- Схема обработки исключений отражена в структуре графа с помощью введения дополнительных связей между защищенным блоком и блоками обработки исключений
- Так, каждый защищенный блок имеет ссылки на соответствующие блоки обработки исключений. При этом ссылки пронумерованы в том порядке, в каком происходит выполнение обработчиков исключений
- Другими словами, обработка исключений требует введения в граф особых дуг, которые отражают не передачу управления между инструкциями, а последовательность обработчиков исключений, присоединенных к защищенному блоку

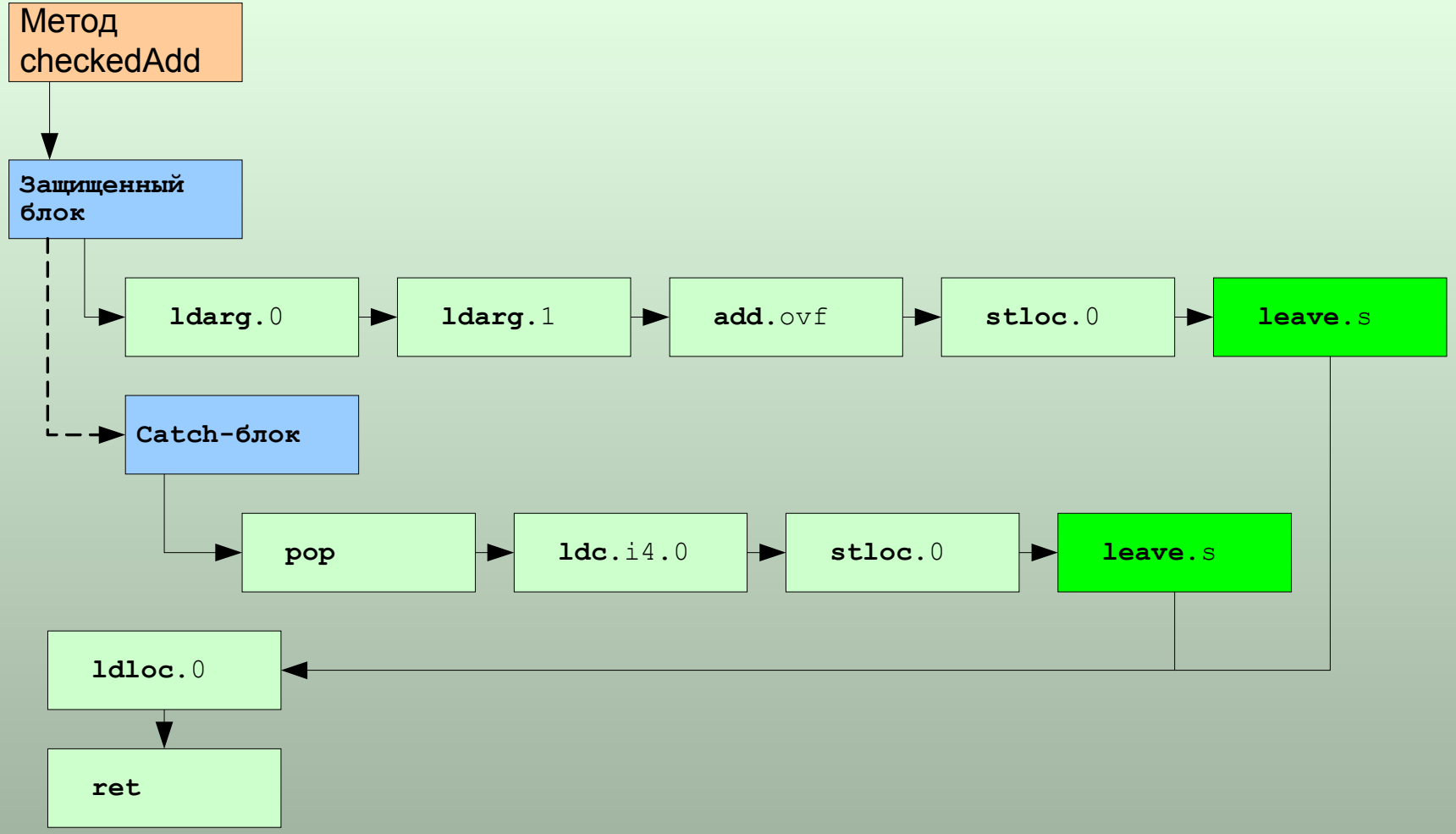
Защищенный блок и прикрепленные к нему блоки обработки исключений



Пример (метод checkedAdd)

```
.method private static int32 checkedAdd(int32 x, int32 y) {
            .maxstack  2
            .locals init (int32 result)
            .try {
            ldarg.0
            ldarg.1
            add.ovf
            stloc.0
            leave.s    exit
            }
            catch System.OverflowException {
            pop
            ldc.i4.0
            stloc.0
            leave.s    exit
            }
exit:       ldloc.0
            ret
}
```

Пример (граф потока управления для метода checkedAdd)



13.3. Дерево блоков в графе потока управления

- В случае представления кода в виде линейной последовательности инструкций для каждой инструкции всегда существует возможность определения блока, в который эта инструкция входит
- Для обеспечения этой возможности в графе потока управления можно построить дерево:
  - в нелистовых вершинах – блоки
  - в листовых вершинах – обычные инструкции
  - ребра обозначают вхождение в блок

Схема дерева блоков