МЕТОД ТОКЕНИЗАЦИИ В ТЕКСТАХ КОМПЬЮТЕРНЫХ ПРОГРАММ

Лось Н.С.

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Черкас Н.Л. – канд. физ.-мат. наук

Рассмотрен метод токенизации для оптимизации обработки исходного кода программ.

Часто в компьютерном коде программ можно найти структуры и строки кода, что у которых одинаковая функциональность, и из одной можно получить другую без особых усилий. Чтобы бороться с такого рода средствами сокрытия плагиата было придумано представление кода, называемое токенизированным. Основная идея этого представления – сохранение существенных и игнорирование всех поверхностных (то есть легко модифицируемых и не меняющих поведения) деталей кода программы. Процедура токенизации выглядит примерно так:

1 Каждому оператору языка (кроме пустого – его игнорируем), который не является операндом, приписываем код/символ, назначенный заранее для каждого класса операторов. Также коды можно приписывать блочным операторам (например, begin/end, {}), подключениям библиотек и заголовочных файлов. Таким образом создается шифр алфавита языка программирования.

2 Строим строку из полученных кодов, сохраняя порядок следования их в исходном коде программы. Один символ строки (токен) – код одного оператора. Таким образом, мы автоматически игнорируем названия функций и переменных (классов, объектов и так далее), разделительные символы, минимизируем и, чаще всего, предотвращаем влияние мелких изменений кода программы.

К одному классу операторов обычно относят те, которым соответствует один идентификатор языка программирования, все вызовы функций, вызовы методов классов, объявления переменных элементарных типов, объявление экземпляров классов. Пример токенизации схожих программ отображен на рисунке 1.1.

Файл с исходным кодом:	Последовательность токенов:
fact.c:	
`int fact = 1;`	`KION`
`int n = 5;`	`KION`
`for (int i = n; i > 1; i--)`	`KKIOIIONIO`
`fact = fact * i;`	`IOIOI`
prod.c:	
`int prod = 1;`	`KION`
`int m = 5;`	`KION`
`for (int j = 1;`	`KKION`
`j <= m;`	`IOI`
`j++)`	`IO`
`prod = j * prod;`	`IOIOI`

Рисунок 1.1 – Пример токенизации

Нужно отметить, что процесс токенизации и разбиение операторов на классы зависим от используемого языка программирования. С другой стороны, после токенизации зашифрованный код не привязан к какому-либо языку, а, следовательно, при тщательном выборе токенов, классов операторов и конструкций в различных языках программирования, можно построить систему, способную после токенизации проверять на плагиат файлы, изначально написанные с помощью различных средств. Например, такие языки как C, C++, C#, Java имеют очень похожие синтаксические блоки. [1]

Абстрактное синтаксическое дерево (АСД) – в информатике это конечное, помеченное, ориентированное дерево, в котором внутренние вершины сопоставлены с операторами языка программирования, а листья – с соответствующими операндами. Пример абстрактного синтаксического дерева представлен на рисунке 1.2.

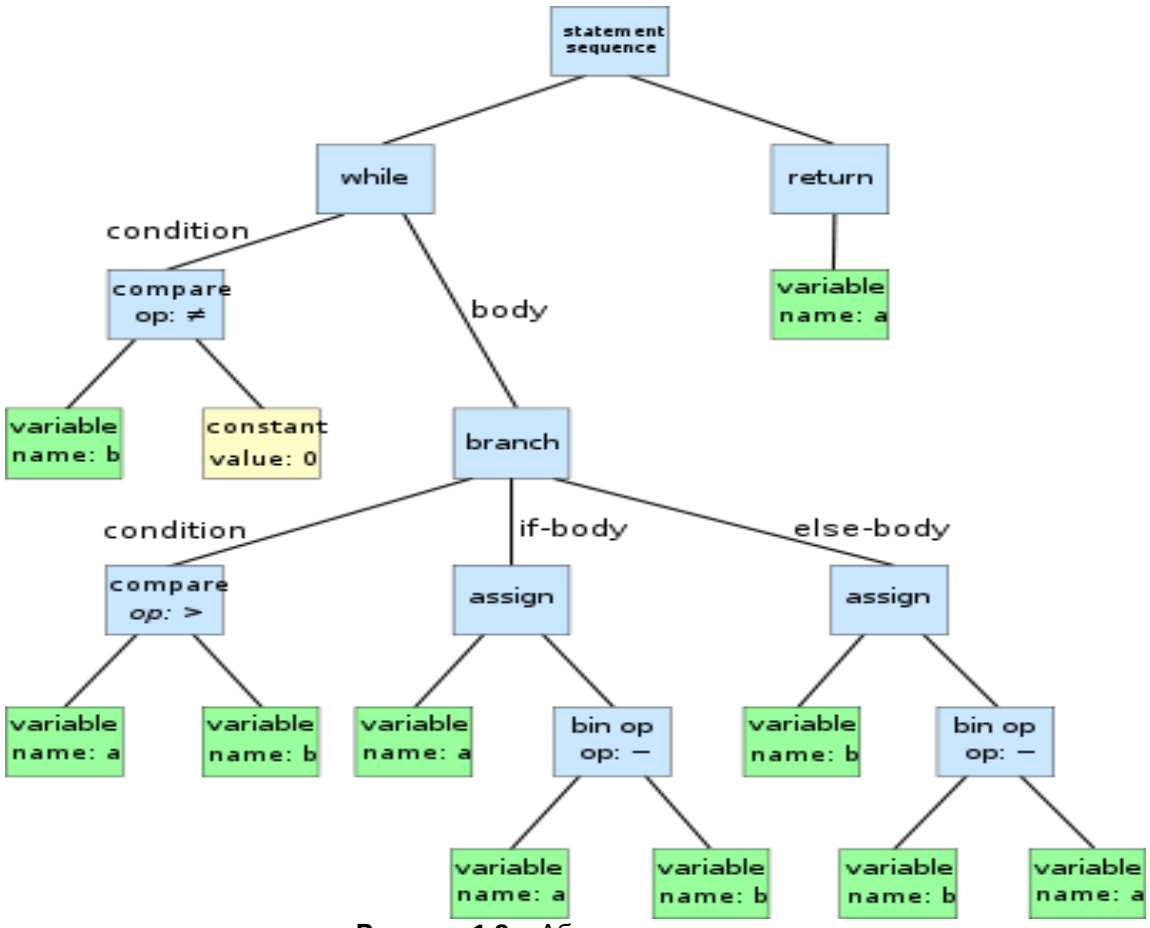

Рисунок 1.2 – Абстрактное синтаксическое дерево

Для языка, который описывается контекстно-свободной грамматикой, какими являются большинство языков программирования, создание абстрактного дерева является тривиальной задачей. Большинство правил в грамматике создают новую вершину, а символы правила становятся рёбрами.

Преобразование программы в АСД зависит от конкретного языка программирования, но позволяет выявить зависимости, которые бы пропустила обычная токенизация. Процесс является очень трудоёмким [1].

Список использованных источников:

1. Макаров, В.В. Идентификация дублирования и плагиата в исходном тексте прикладных программ / В.В. Макаров// Лаборатория компьютерной графики [Электронный ресурс]. - 2016г. - Режим доступа: http://lab18.ipu.rssi.ru/projects/conf2006/1/%D0%92.%D0%92.%D0%9C%D0%B0%D0%BA%D0%B0%D1%80%D0%BE%D0%B2.htm. - Дата доступа: 13.05.2020.