Улучшенная оптимизация множественного присваивания в PascalABC.NET

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
по направлению подготовки
01.03.02— Прикладная математика и информатика

Филонов А. К.

Научный руководитель: зав. каф., доц., к. ф.-м. н., Михалкович С. С.

Постановка задачи

- Разработать алгоритм оптимизации множественного присваивания для различных типов символов
- Создать архитектуру внедрения указанного алгоритма в компилятор PascalABC.NET
- Реализовать собственно алгоритм оптимизации множественного присваивания в компиляторе PascalABC.NET
- Провести сравнительный анализ по количеству присваиваний и быстродействию для языков PascalABC.NET и C#

Введение

Множественное (кортежное) присваивание – сахарная конструкция, позволяющая присвоить значения сразу нескольким переменным. Присутствует, например, в C#, Python, PascalABC.NET

```
(a, b) = (b, a)
(a,b,c) = (c, 3, p.q)
```

Множественное присваивание

Кортежное присваивание, как и все сахарные конструкции, преобразуется компилятором в более простые конструкции – в подряд идущие отдельные присваивания.

В работе [1] был разработан алгоритм для модельного языка, оптимизирующий количество генерируемых присваиваний.

Цель этой работы – внедрить алгоритм в компилятор PascalABC.NET и улучшить обработку различных типов символов.

Типы символов

- Элементы массивов
- Составные имена
- Локальные имена
- Имена пришедшие из внешнего контекста
- Var-параметры
- Указатели
- Выражения

В контексте этой работы, тип символа определяет при каких условиях он может оказаться синонимом с другим символом, то есть быть одной ячейкой памяти.

Синонимами между собой могут быть

- элементы массивов
- составные имена и имена из внешнего контекста, у которых совпадает последнее имя

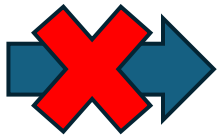
Var-параметр может быть синонимом с любым нелокальным символом.

BindCollectLightSymInfo

Визитор BindCollectLightSymInfo позволяет связывать использование символа с его определением. Для поиска используется функция bind, которой передается имя(ident). В процессе поиска строится древовидная структура пространств имен, которая заполняется и дополнятся при новых вызовах функции bind.

Если не обрабатывать возможные синонимы, то возможны **некорректные присваивания** – присваивания, в результате которых переменной из левой части кортежного присваивания присваивается не соответствующее ей значение из правого кортежа.

```
begin
  var a := ArrRandomInteger(n,0,10);
  var b := a;
  var i := 1;
  var j := 1;
  var c := 4;
  var d := 5;
  (a[i], d) = (c, b[j])
end.
```



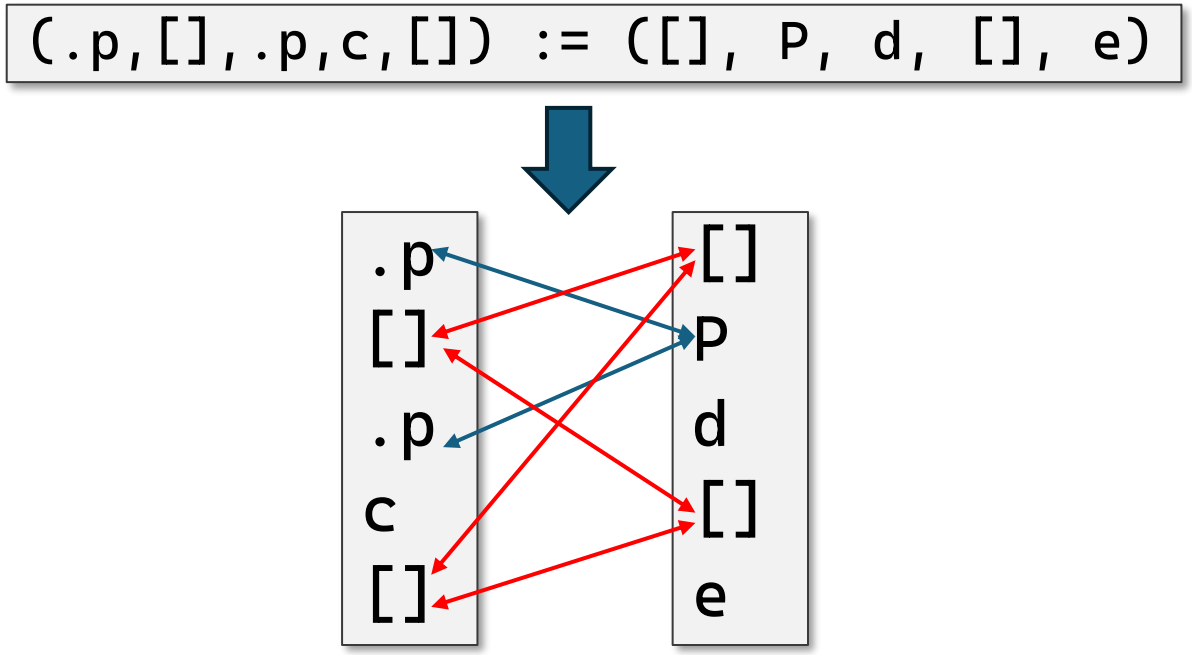
```
begin
  var a := ArrRandomInteger(n,0,10);
  var b := a;
  var i := 1;
  var j := 1;
  var c := 4;
  var d := 5;
  a[i] := c;
  d := b[j];
end.
```

Алгоритм обработки различных типов символов

1. Если есть операция разыменования или получения адреса – сделать 2*n присваиваний и закончить алгоритм
2. Ввести временные переменные для выражений и var-параметров
3. Разрешить все графы возможных синонимов

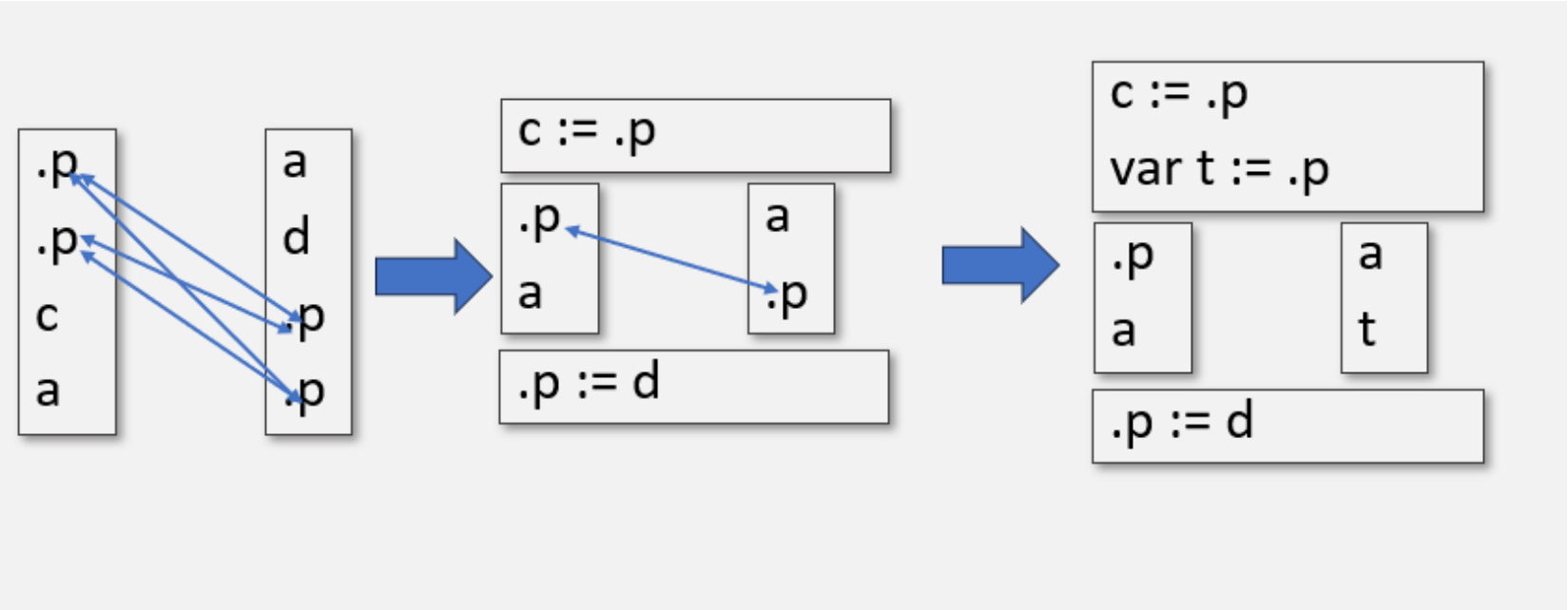
Графы возможных синонимов

Для краткости далее будут использоваться следующие сокращения: .”имя” – составное имя, оканчивающиеся на “имя”; имя, написанное заглавными буквами – имя, пришедшее из внешнего контекста; [] – элемент массива; (var) – var-параметр; var t – временная переменная



Разрешение графа возможных синонимов

1. Вынести все присваивания, которые можно вынести на этот момент
2. Выбрать сторону, на которой осталось меньше возможных синонимов
3. Выносить присваивания или вводить новую переменную для возможных синонимов из выбранной стороны



Изменения алгоритма оптимизации кортежного присваивания

Алгоритм и изменения:

1. Удаление избыточных присваиваний –> вынесение или генерация 2*n присваиваний

2. Обработка различных типов символов и выражений –> новый алгоритм

3. Создание графа присваиваний

4. Поиск простых циклов графа –> алгоритм Джонсона[2] заменен на алгоритм проверки каждой компоненты связности на наличие цикла. Это делается потому, что граф присваиваний – псевдолес[3], а каждая компонента связности псевдолеса содержит не более одного цикла. Для поиска компонент связности используется алгоритм Косарайю[4].

5. Устранение циклов графа

6. Обход леса и генерация отдельных присваиваний

Внедрение алгоритма в компилятор PascalABC.NET

Для внедрения алгоритма в компилятор PascalABC.NET был написан визитор NewAssignTuplesDesugarVisitor, который наследуется от визитора AssignTuplesDesugarVisitor, который использовался для раскрытия кортежных присваиваний для этого.

Также NewAssignTuplesDesugarVisitor оптимизирует кортежные объявления, то есть конструкции вида переводя их в n одиночных присваиваний.

```
var (a, b) = (c, d)
```

Кортежное объявление

Сравнения количества генерируемых присваиваний с компилятором C#

Раскрываемое множественное присваивание	Количество отдельных присваиваний	
	C#	PascalABC.NET
(a, b) = (b, a)	3	3
(a, b, c, d, e, f) = (b, c, d, e, f, a)	11	7
([], []) = ([], [])	6	3
(.p, .p, .p) = (.p,.p,.p)	8	5
(.p, .p, .q, c, [], a) = (.q, .p, .q, [], [], b(var))	16	7

Замеры прироста производительности

Алгоритм	Без оптимизации	С оптимизацией	Прирост производительности
1	320 мс	280 мс	12.5%
2	3.1 мс	2.1 мс	32%

```
for var i := 1 to n-1 do
  for var j := 1 to n-1-i do
    if a[j] > a[j+1] then begin
      (a[j], a[j+1]) := (a[j+1], a[j]);
    end;
```

Алгоритм 1

```
loop n do
  (a, b, c, d, e) := (b, c, d, e, a);
```

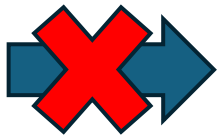
Алгоритм 2

Ситуация, когда алгоритм неправильно раскрывает множественное присваивание

Когда в множественном присваивании присутствуют имена, являющиеся частью других, то возможно неправильное раскрытие кортежного присваивания.

Для правильного раскрытия нужны ref-конструкция и функционал, позволяющий находить имена, являющиеся частью других.

```
(a, a[i]) := (b, c)
```



```
a := b;
a[i] := c;
```

Результаты

- Написан визитор для поиска информации о символах
- Разработан алгоритм оптимизации множественного присваивания для различных типов символов
- Разработанный алгоритм внедрен в компилятор PascalABC.NET
- Проведено сравнение количества генерируемых присваиваний с компилятором C#
- Показан прирост производительности при использовании оптимизации

Литература

1. Филонов А. К. Оптимизация множественного присваивания в PascalABC.NET – URL: https://hub.sfedu.ru/repository/material/801312059/ (дата обращения 1.06.2024)
2. Donald B. Johnson 1975. Finding All the Elementary Circuits of a Directed Graph. SIAM J. Comput., 4(1), p.77–84.– Алгоритм Джонсо-на поиска всех простых циклов на графе.
3. Kruskal, C., Rudolph, L., and Snir, M. 1990. Efficient Parallel Algorithms for Graph Problems. Algorithmica, 5, p.43-64. – Псевдолес
4. https://ru.wikipedia.org/wiki/Алгоритм_Косарайю - Алгоритм Косарайю