УДК 004.046

РЕВЕРСИВНЫЙ ИНФОРМАЦИОННЫЙ ГРАФ ДЛЯ ФУНКЦИОНАЛЬНО-ПОТОКОВОЙ ПРОГРАММЫ

Матковский И. В.
научный руководитель доктор техн. наук Легалов А. И.
Сибирский Федеральный Университет

Разработка архитектурно-независимого ПО в последнее время все чаще и чаще используется в качестве решения одной из наиболее серьезных проблем, стоящих перед современными программистами – чрезвычайно большим количеством архитектур параллельных вычислительных систем (ПВС). Ориентация на работу с конкретной архитектурой сильно затрудняет отладку и сопровождения кода; отказ от такой ориентации ведет к появлению кода неоптимального.

Во время разработки архитектурно-независимого ПО программист может описывать максимально возможный параллелизм (на уровне базовых операций) и не заботиться об учете имеющихся ресурсов. Уже на этапе исполнения модуль интерпретации будет обрабатывать предварительно отлаженную программу в соответствии с требованиями определенных исполняющих ПВС.

В функциональных языках параллельного программирования процесс вычисления описывается в виде системы отношений между отдельными функциями – что сильно упрощает организацию параллелизма. Избавиться от необходимости контролировать процессу управления вычислениями можно, исполняя функции по готовности данных (dataflow).

Разумеется, и у функциональных языков есть свои недостатки – так, на данный момент они в среднем могут уступать более популярным императивным по чистой производительности. Данная проблема, впрочем, относится уже к деталям воплощения исполняющего (интерпретирующего) механизма.

“Пифагор” представляет собой язык для написания функционально-потоковых параллельных программ. [1,2,3] Одной из главных особенностей работы с программами на языке “Пифагор” – наряду с отсутствием привязки к архитектуре и контроля ресурсов – является разделение информационного графа (ИГ) программы и управляющего графа (УГ). Это разделение позволяет корректировать и изменять стратегии управления вычислениями, оставляя неизменными сами вычисления.

С точки зрения событийной машины программа на языке “Пифагор” представляет собой сочетание четырех частей:

1. Реверсивный информационный граф описывает существующие в программе зависимости по данным.
2. Управляющий граф показывает, каким образом в программе передаются сигналы о готовности тех или иных данных.
3. Слой автоматов определяет текущие состояние тех или иных вершин и, при необходимости, создают новые управляющие сигналы.
4. Слой данных хранит все существующие в рамках функции данные – как заданные изначально, так и сформированные по ходу программы.

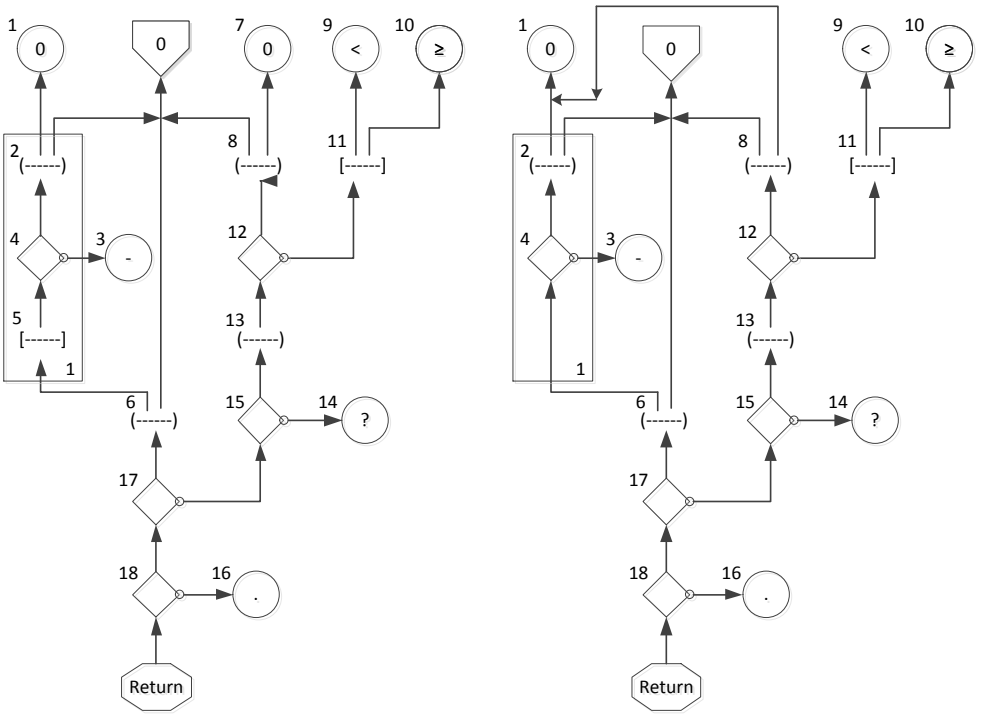
Подробно устройство и взаимодействие слоев описано в [3]

В качестве примера для демонстрации устройства графов будет использоваться программа вычисления абсолютного значения числа:

```
math.abs << funcdef X
{
```

```
        val<<({(0,X):-},X);
        kluch<<((X,0):[<,>=]):?;
        return<<val:kluch:.;
}
```

В зависимости от знака входного аргумента X данная программа возвращает либо сам X, либо 0-X. Выбор определяется результатом сравнения X с 0. С более подробным описанием синтаксиса языка можно ознакомиться в [1].



а б

Рисунок 1 – Реверсивный информационный граф программы.

На рисунке 1 изображены два варианта реверсивного информационного графа (РИГ) программы получения абсолютного значения числа. Слово “реверсивный” показывает, что связи в данном графе идут от приемника данных к источнику. Процесс нахождения значения функции, таким образом, можно воспринимать, как процесс нахождения значения, поступающего в его узел return.

Уже сформированный РИГ может быть подвергнут оптимизации; простой пример такой оптимизации показан на рисунке 1б. В данном случае из графа была удалена одна вершина-константа (являющаяся фактическим двойником другой) и один параллельный список, состоящий лишь из одного элемента.

Ранее сформированные РИГ хранились в двоичном виде; увы, такой формат легко воспринимался только утилитами, но не человеком. Сейчас система инструментальной поддержки переводится на работу с одним только текстовым представлением; пример его приведен ниже.

```
External
        0       math.abs
Local
        0       0
        1       {1}2
id      delay   operation       links           positions
```

```
0     0     arg                 pos 1 21 1 22
1     1     (---)     loc:0 0   pos 2 14 2 27
2     1     :         1 -       pos 2 21 2 22
3     0     (---)     loc:1 0   pos 2 21 2 22
4     0     (---)     0 loc:1   pos 3 21 3 22
5     0     [---]     < >=      pos 3 21 3 22
6     0     :         4 5       pos 3 22 3 23
7     0     (---)     6         pos 3 21 3 22
8     0     :         7 ?       pos 3 21 3 22
9     0     :         3 8       pos 4 21 4 22
10    0     :         9 .       pos 4 21 4 22
11    0     return  10          pos 4 21 4 22
```

В текстовом представлении РИГ можно условно выделить три части – объявление внешних (external) ссылок, внутренних (local) ссылок и вершин самого графа. К внешним ссылкам относятся все вызовы внешних пользовательских функций, к внутренним – все отсылки к константным значениям. Для каждой из вершин РИГ указывается уникальный номер, номер соответствующего задержанного списка, тип, перечень входящих связей и ссылка на позицию элемента, соответствующего данной вершине в исходном коде программы.

ЛИТЕРАТУРА.

1. Легалов А. И. Функциональный язык для создания архитектурно-независимых параллельных программ // Вычислительные технологии : журнал. — 2005. — Т. 10. — № 1. — С. 71-89.
2. Легалов А.И., Редькин А.В., Матковский И.В. Функционально-потоковое параллельное программирование при асинхронно поступающих данных // Параллельные вычислительные технологии (ПаВТ'2009) : Труды международной научной конференции (Нижний Новгород, 30 марта – 3 апреля 2009 г.). — Челябинск: Изд. ЮУрГУ, 2009. — С. 573-578.
3. Легалов А.И, Непомнящий О.В., Матковский И. В., Фарков. М.А., Особенности преобразования и выполнения функционально-потоковых параллельных программ // Труды НПО 2011: материалы Ершовской конференции по информатике (Новосибирск, 27 июня – 1 июля 2011 г.). — Новосибирск: Институт систем информатики, 2011. — С. 146-153.