ЕВРАЗИЙСКОЕ НАУЧНОЕ ОБЪЕДИНЕНИЕ
EURASIAN SCIENTIFIC ASSOCIATION
www.esa-conference.ru

УДК 004.822

Вопросы реализации и применения концептуальных графов в сочетании с программированием потоков данных

Назаренко Петр Александрович, кандидат технических наук, доцент
Бикмурзин Роберт Маратович, студент магистратуры
Поволжский государственный университет телекоммуникаций и информатики (г. Самара)

Сочетание принципов программирования потоков данных с концептуальными графами и семантическими сетями позволяют создавать интеллектуальные информационные и управляющие системы. Рассматривается архитектура и реализация исполняемых узлов графа вычислений в программировании потоков данных, выполненного в виде концептуального графа, а также состав базового набора узлов такого графа.

Ключевые слова: *Семантическая сеть, концептуальный граф, информационная система, программирование потоков данных, исполняемый узел, ПЛИС.*

Семантические сети и являющиеся их разновидностью концептуальные графы применяются для представления знаний, организации баз знаний и данных. Их интеграция в процессы обработки информации позволяет создавать интеллектуальные информационно-управляющие системы. Такие сети и графы классифицируются по нескольким различным критериям [4]. Значительный интерес представляют обучаемые, исполняемые и гибридные сети и графы. В статье рассматривается реализация и применение исполняемых и гибридных концептуальных графов для решения задач обработки данных с использованием принципов программирования потоков данных в интеллектуальной информационной системе. Областями применения подобной системы может быть обработка сигналов, в том числе от датчиков, управление технологическим или телекоммуникационным оборудованием и т.п.

Предпосылкой к объединению программирования потоков данных и исполняемых (а также, гибридных) концептуальных графов служит то обстоятельство, что последние содержат узлы, в которых размещаются вычислительные операции. Кроме того, при практической реализации концептуальных графов могут проявляться дополнительные сходные черты с узлами графа вычислений программирования потоков данных [2, 3]. Наконец, сам граф вычислений может, в зависимости от характера решаемой задачи, совпадать по своей архитектуре с соответствующим концептуальным графом или являться его подграфом.

В наиболее простом виде предполагается, что обрабатываемые данные передаются между узлами вычислений, в том числе в явном виде. Граф вычислений используется только для собственно вычислений (в том числе достаточно сложных), что может быть недостаточно для организации полноценной информационной и управляющей системы. Указанные обстоятельства могут считаться недостатками технологии программирования потов данных в её исходном понимании. Одним из методов устранения таких недостатков может считаться объединение этой технологии с концептуальными графами и семантическими сетями.

Пусть концептуальный граф, по крайней мере, часть которого используется для осуществления какой-либо обработки данных путём выполнения вычислений, кроме исполняемых узлов содержит несколько (не менее двух) узлов ввода данных. Минимальное количество таких узлов определяется числом операндов в имеющихся элементарных операция, максимальное их количество – характером решаемой задачи. Данные могут поступать в узлы ввода по командам от внешних источников, например, программы обслуживания концептуального графа, или обновляться непрерывно и асинхронно. Кроме того, должны иметься узлы результата (один и более), а также, возможно, потребуются узлы промежуточных вычислений. Собственно, выполняемые операции должны храниться в исполняемых узлах и могут быть реализованы различными способами.

Рассмотрим структуру узла исследуемого концептуального графа или семантической сети. Каждый узел должен иметь уникальный идентификатор, позволяющий однозначно выделить конкретный узел из всего набора узлов, полезные данные, отображающие хранящиеся знания, идентификатор «старшего» или «родительского» узла в бинарном отношении, а также идентификатор узла, определяющего тип специфицированного отношения между «старшим» и «младшим» узлами в бинарном отношении. На языке высокого уровня, например, Си++, такой узел может быть описан следующим образом.

```
struct node1 {
 float Id, Id2, Id3;
 unsigned long Size;
 unsigned char* Memory;
};
```

Кроме того, как показывают результаты исследований и практического применения концептуальных графов, при решении определённых задач целесообразно хранить в узле также время его создания. Структура узла показана на рис. 1.

Узел графа
Идентификатор 1
Идентификатор 2
Идентификатор 3
Размер данных
Данные
Метка времени

Рис. 1. Структура узла концептуального графа

Таким образом, операция, размещаемая в исполняемом узле, относится к категории полезных данных.

Для автономной информационной системы представляют интерес следующие способы реализации операций. Как простые операции, так и вычисления по алгоритмам произвольной сложности могут храниться в исполняемых узлах в виде законченных наборов машинных команд. Формироваться эти наборы могут путём трансляции фрагментов программ на одном из языков программирования, и загружаться в узлы концептуального графа при каждом старте информационной системы.

С точки зрения простоты разработки и построения информационной системы предпочтительно формировать операции и алгоритмы в виде функций на языке высокого уровня. Затем с помощью программы обслуживания концептуального графа в соответствующие исполняемые узлы помещаются либо машинные команды этих функций, либо адреса функций [1]. С целью унификации и простоты использования все такие функции должны иметь одинаковые сигнатуры, т.е. возвращаемые значения и аргументы, причём набор аргументов должен обеспечивать достаточно широкие возможности по обработке и передаче данных, например, переменное число операндов для различных операций. Для реализации такого метода язык программирования должен иметь развитую адресную арифметику, либо возможность работы с делегатами функций. Подобным способом можно реализовывать, в том числе, рекурсивные алгоритмы, причём наиболее просто создаются арифметические операции, включая операции сравнения. При создании информационной системы, предметной областью которой является обработка или передача данных, узлы графа, содержащие эти операции, могут не только выполнять свою непосредственную нагрузку, но и формировать ядро информационной системы, т.е. набор узлов, позволяющий как описать саму систему, так и реализовать произвольные алгоритмы.

В состав такого ядра, кроме указанных операций, целесообразно включить узлы, описывающие элементы собственно концептуального графа и его компонентов:

– адрес участка памяти;

– некоторый абстрактный узел графа – узел, адрес блока памяти в котором не равен нулю;

– ноль – узел, содержащий значение 0;

– пустой узел – адрес блока памяти в узле равен нулю;

– содержимое узла – адрес блока памяти непустого узла;

– идентификатор – результат операции возврата номера узла;

– вход данных – фиксированный узел ввода, имеющий фиксированный идентификатор;

– новый узел – результат операции создания нового узла, возвращающей его адрес;

– единица – узел, содержащий значение 1;

– размер – результат возврата размера блока данных в узле графа;

– блок данных – узел, размер блока в котором больше 1.

Отдельные узлы могут использоваться для представления служебных операций:

– получение адреса узла по идентификатору;

– ввод данных в узел ввода;

– копирование содержимого одного узла в другой;

– перемещение содержимого узла-источника в узел-приёмник;

– выполнение машинных команд, хранящихся в узле.

Кроме того, выше уже были указаны узлы результатов операций и промежуточных значений. Все перечисленные узлы можно разделить на две группы – факты и действия. С учётом указанных выше требований к аргументам, например, функция получения адреса размещённых в узле данных может выглядеть на языке Си++ следующим образом:

```
void* Adres(void* r, void* =NULL, void* =NULL, void* =NULL)
{
 return ((node1*)r)->Memory;
}
```

Собственно, программа обработки данных в этом случае может выглядеть как список идентификаторов узлов концептуального графа с аргументами, которые также являются узлами, например:

8, 11 – ввод данных в узел 11;

9, 13, 11 – копирование данных из узла 11 в узел 13;

8, 12 – ввод данных в узел 12;

9, 14, 12 – копирование данных из узла 12 в узел 14;

6, 13, 14, 19, 1 – суммирование в узле 19 содержимого узлов с 13 по 14;

7, 21, 19 – преобразование данных из узла 19, результат – в узле 21.

Программа может храниться либо в одном из узлов в виде сплошного набора чисел, интерпретируемых специальной процедурой, либо по одной строке в отдельном узле, а в выделенном узле программы содержатся номера узлов строк. Пример такого варианта:

A: 8, 11 – ввод данных в узел 11;

B: 9, 13, 11 – копирование данных из узла 11 в узел 13;

C: 8, 12 – ввод данных в узел 12;

D: 9, 14, 12 – копирование данных из узла 12 в узел 14;

E: 6, 13, 14, 19, 1 – суммирование в узле 19 содержимого узлов с 13 по 14;

F: 7, 21, 19 – преобразование данных из узла 19, результат – в узле 21.

Здесь *A* ... *F* – условные идентификаторы узлов строк программы, список которых, в данном случае <*A*, *B*, *C*, *D*, *E*, *F*>, и хранится в выделенном узле программы.

Следует указать также такое применение программирования потоков данных, как реализация графа вычислений на микросхемах ПЛИС (программируемая логическая интегральная схема) или ППВМ (программируемая пользователем вентильная матрица, FPGA – Field-Programmable Gate Array). В этом случае узлы обработки могут представлять собой как достаточно простые или даже элементарные операции, логические и арифметические, так и полноценные микро-ЭВМ, количество которых на одном кристалле FPGA может состав-

ЕВРАЗИЙСКОЕ НАУЧНОЕ ОБЪЕДИНЕНИЕ
EURASIAN SCIENTIFIC ASSOCIATION
www.esa-conference.ru

лять от 8 и более, в зависимости от конкретного изделия. В этом случае достигается аппаратное распараллеливание и достаточно высокая скорость обработки данных. При этом концептуальный граф как структура более высокого уровня может формироваться с помощью запоминающих устройств, входящих в состав микросхем FPGA. Областью применения подобной реализации графа вычислений и программирования потоков данных может быть аппаратная обработка сигналов датчиков в системах управления различного назначения – как в крупных промышленных АСУТП, так и в системах управления, например, телекоммуникационным оборудованием, системами видеонаблюдения и т.п.

Разработано тестовое программное обеспечение на языках Си++ и Python. Выполнено программное моделирование концептуальных графов и графов вычислений, подтверждающее работоспособность рассмотренных механизмов концептуальных графов, включая построение исполняемых узлов, практическое применение базового набора понятий концептуального графа.

Литература:

1. Назаренко П.А. Реализация исполняемых узлов гибридной семантической сети / П.А. Назаренко // XVI Российская научная конференция профессорско-преподавательского состава, научных сотрудников и аспирантов (26 января – 30 января 2009, г. Самара). Материалы конференции. – Самара.: ПГУТИ, 2009. – С. 180.

2. Carkci M. Dataflow and Reactive Programming Systems: A Practical Guide. – CreateSpace Independent Publishing Platform, 2014. – 570 p.

3. Sharp J.A. Data Flow Computing: Theory and Practice. – Intellect, Limited, 1992. – 566 p.

4. Sowa John F. Semantic Networks [Электронный ресурс] / John F. Sowa, 2015. – Режим доступа: http://www.jfsowa.com/pubs/semnet.htm, свободный.