Модель визуализации изменений в графе внутреннего представления программ языка Cloud Sisal

Д.С. Гордеев
gds@iis.nsk.su
Институт систем информатики СО РАН, Новосибирск, Россия

В работе описывается решение задачи визуализации изменений в графовой модели внутреннего представления программ с целью отражения процессов, происходящих с графовой моделью в процессы вычислений или обработки с помощью графовых алгоритмов. Для решения задачи применяется модель, описывающая изменения в графовой модели как графические анимации, отображающие изменения в визуальных стилях, которые соответствуют изменениям в атрибутах графовой модели. Каждое изменение атрибута, добавление вершины или удаление ребра отражается на визуальном отображении анимацией графических стилей. Ключевым моментом модели является, выделение понятия контекста графической анимации, как визуального отражение внешнего окружения по отношению к происходящему изменению.

Ключевые слова: *визуализация алгоритмов, анимация алгоритмов, динамический граф.*

Visualization of changes in Cloud Sisal program internal representation graph

D.S. Gordeev
gds@iis.nsk.su
Institute of Informatics Systems SB RAS, Novosibirsk, Russia

This paper describes the solution of problem of visualization of changes in graph model of internal representation of programs in order to reflect processes which happens during calculation of programs or processing with graph algorithms. We use model which interprets every change of graph model as corresponding change of visual graphical styles, which corresponds to changes of attributes of graph model elements or to structure of graph model. Each single changing of an attribute value, adding of node to graph model or removing of edge from graph model is reflected over visual representation as animation of graphical styles of pieces of image. The key point of described model is an idea of context of visual animation as surrounding visual information in relation to happening change in graph model.

Keywords: *algorithm visualization, algorithm animation, dynamic graph.*

1. Введение

В рамках развития проекта облачной системы параллельного программирования CPPS [8], целью которой является предоставление возможности решения вычислительно-ёмких задач с помощью доступных по сети супервычислителей, поддерживаемых системой CPPS, широкому кругу пользователей без прямого доступа к вычислителям большой мощности, но имеющих выход в Интернет, где пользователи создают и отлаживают программы на языке Cloud Sisal [4] с помощью вебприложения в одном из стандартных браузеров, а решение вычислительных задач осуществляется на поддерживаемом системой супервычислителе после предварительной адаптации с помощью настраиваемого оптимизирующего компилятора из комплекта системы CPPS. При этом возникает задача поддержки процесса программирования и отладки исполнения Cloud Sisal программы на конкретном вычислительном оборудовании, при условии, что код программы инструментирован для получения соответствующей информации о ходе вычислений.

При трансляции программы на языке Cloud Sisal генерируется граф внутреннего представления IR входной программы, который представим в виде иерархического ориентированного графа [3]. При отладке программы полезно иметь визуальное отображение исполнения для понимания движения данных по графу, и на каких узлах вычислителя осуществляются вычисления тех или иных функций. При этом возникает задача отображения графа, и свойств вершин и дуг этого графа с помощью графических примитивов, таких как многоугольники, ломаные линии, различных цветовых стилей, и визуального отображения изменений атрибутов вершин или дуг. Данная задача является нетривиальной, так как основным пользователем системы является человек, и его ожидания должны совпадать с наблюдением за отображением изменений, чтобы возникло корректное представление о том, как происходило вычисление в заданной программе.

Методы визуализации процесса вычислений зависят от решаемой задачи и её предметной области, в данном случае это предметная область графов потока данных, где результатом визуализации в том или ином виде является последовательность изображений состояния графовой модели со всеми графически отображаемыми атрибутами и структурой. Другими словами результатом являет динамический граф, и для визуализации динамических графов в основном известно два семейства техник визуализации: автономная (offline) и интерактивная (online) [9]. В первом случае строится последовательность изображений графа, при визуализации которой между двумя соседними изображениями используются техники интерполяции для получения промежуточного изображения, и набор таких промежуточных изображений описывает трансформацию между двумя соседними автономными изображениями графа. Во втором случае используется динамическое вычисление очередного изображения исходя из текущего состояния графовой модели и следующего события, которое следует графически отобразить.

В данной работе обсуждается реализация модели визуализации динамических графов для решение задачи отображения процесса вычислений программ на языке

Работа опубликована при финансовой поддержке РФФИ, грант 18-07-20045\18

Cloud Sisal в рамках проекта развития облачной системы параллельного программирования CPPS.

2. Внутреннее представление IR

Текст Cloud Sisal программы транслируется внутреннее представление IR, которое описывается как иерархический ориентированный граф с портами. Формально такие графы можно описывать следующим образом. Пусть задан граф

$$\mathrm{G} = (\mathrm{V},\ \mathrm{P},\ F^v_{p,i},\ F^v_{p,o},\ \mathrm{E}), \quad (1)$$

где V это непустое конечное множество вершин, P это непустое конечное упорядоченное множество портов, представимое в виде объединения двух непересекающихся множеств P_i и P_o. $F^v_{p,i}$ это функция $F^v_{p,i} : P_i \to V$, $F^v_{p,o}$ это функция $F^v_{p,o} : P_o \to V$,. E это подмножество декартового произведения $< P_o \times P_i >$.

Множество P_i называется множеством входных портов, и с помощью функции $F^v_{p,i}$ позволяет говорить об упорядоченном множестве входных портов для каждой вершины из графа G. Аналогично множество P_o называется множеством выходных портов, и с помощью функции $F^v_{p,o}$ позволяет говорить об упорядоченном множестве выходных портов для каждой вершины из графа G.

В случае если фунции $F^v_{p,i}$ и $F^v_{p,o}$ являются тождественными, то каждая вершина имеет ровно один входной порт и ровно один выходной порт, и граф G эквивалентен классическому определению графа G′ = (V', V'), где E' есть подмножество декартового произведения $< V' \ x \ V' >$.

Для вершины v из V и ребра (p_o, p_i) из E будем говорить о инцидентности, если и только если $F^v_{p,i}(p_i) = v$ или $F^v_{p,o}(p_o) = v$. Для вершин v_1 и v_2 из V будем говорить о смежности вершин, если и только если существует ребро (p_o, p_i) из E такое, что ($F^v_{p,o}(p_o) = v_1$ и $F^v_{p,i}(p_i) = v_2$ или ($F^v_{p,o}(p_o) = v_2$ и $F^v_{p,i}(p_i) = v_1$).

Каждая вершина графа G внутреннего представления IR соответствует функции Cloud Sisal программы, по которой построен граф внутреннего представления. Входные порты P_i вершины v отвечают аргументам соответствующей функции, а выходные порты P_o вершины v отвечают результатам вычислений соответствующей функции. Дуга e в графе G, соединяющая выходной порт p^1_o и p^2_i, обозначает передачу данных из выходного порта p^1_o во входной порт p^2_i.

3. Графовая модель визуального представления внутреннего представления IR

При визуализации вершины графов обычно представляются геометрическими фигурами, такими как прямоугольники, окружности или более сложные многоугольники. Дуги часто представлены в виде ломаных или гладких кривых. Порты, в зависимости от применения изображения графа, могут отображаться с помощью кругов или более мелких квадратов, относительно вершины. Причём обычно порты располагаются на границах многоугольников, представляющих вершины. При записи решения задачи с помощью языка программирования часто функции вызывают другие функции. На графе внутреннего представления это отражается наличием вложенных графов, являющихся метками вершин [3]. Таким образом задаётся иерархия в графе внутреннего представления.

Также для определения стилей визуального отображения геометрических фигур требуется хранить информацию об атрибутах вершин и их значениях. Далее будем рассматривать графовую модель G', расширяющую определение графа G (1) атрибутами. Пусть

$$\mathrm{G}' = (\mathrm{V},\ \mathrm{P},\ F^v_{p,i},\ F^v_{p,o},\ \mathrm{E}, \mathrm{A}_v, \mathrm{A}_p, \mathrm{A}_e), \quad (2)$$

где $A_v : V \to \{(n_v,\ u_{n_v})\}$, где n_v это уникальное имя атрибута, а u_{n_v} это значение атрибута с именем n_v. Аналогично $A_p : P \to \{(n_p,\ u_{n_p})\}$, и $A_e : E \to \{(n_e,\ u_{n_e})\}$, . Для вершин, например, именем атрибута может быть “форма” со значениями “прямоугольник” или “окружность”, “цвет границы” со значениями цвета соответственно, “цвет фона” также со значениями цвета или “подграф” со значением графа, удовлетворяющего определению (2).

Визуальное изображение графа G' строится в зависимости от структуры графа, а также атрибутов и их значений. При наличии такой зависимости отражать ход вычислений на графе IR можно с помощью изменения структуры графа или изменения значений атрибутов. В этом случае можно говорить о динамическом графе [5] DG', где DG' это последовательность графов G'_i, где каждый G'_{i+1} граф получается из G'_i через преобразование $G'_{i+1} = \varphi(G'_i)$, где φ осуществляет теорерико-графовую операцию, например, удаление или добавление ребра или дуги, добавление или удаление вершины, а также изменение значения атрибута вершины, дуги или порта.

4. Поток изменений в графовой модели

Каждое изменение в графе DG', соответствующее преобразованию φ, можно описать как событие изменения [2] в графовой модели G'. Определим каждое такое событие ev как вектор $(el, n, v_1, v_2, t, ctx)$, где el это вершина, порт или дуга графа G', n это имя атрибута элемента el, v_1 это предыдущее значение атрибута с именем n, v_2 это новое значение атрибута с именем n, t это момент времени, когда произошло изменение значения атрибута с именем n, а ctx это вектор контекстной информации, представляющий собой пару (a, b), где a обозначает признак либо входа либо выхода из контекста, b это контекстная информация, например, множество вершин, множество дуг, или целый подграф входного графа.

Рассматриваются следующие события изменений:

1. Добавление вершины в граф. Например, это может соответствовать добавлению функции или выражения в текст программы или добавлению вершины во вспомогательные структуры данных, например стек или очередь.
2. Добавление порта к вершине. Например, это может соответствовать добавлению аргумента в список аргументов функции или добавлению дуги во вспомогательные структура данных.
3. Добавление дуги. Например, это может соответствовать использованию результатов функции в выражении.
4. Удаление вершины.
5. Удаление порта.
6. Удаление дуги.
7. Изменение значения атрибута вершины, порта или дуги.

Кроме того рассматриваются следующие события:

1. Вычисление предиката на вершине.
2. Вычисление предиката на порте.

The publication of the conference proceedings was supported by the RFBR, grant No. 18-07-20045\18

3. Вычисление предиката на дуге.
4. Перебор смежных вершин с предикатом или без.
5. Перебор смежных дуг с предикатом или без.
6. Получение предка по ориентированной дуге.
7. Получение смежные дуг с предикатом или без.
8. Получение смежные вершин с предикатом или без.
9. Получение элемента из множества по предикату.
10. Чтение значения атрибута вершины, дуги или порта.
11. Вход или выход из контекста.

Поток событий можно получить разными способами, например, это может быть информация о том, какая вершины вычислялась в какой момент времени, и по каким дугам передавались данные между портами. Также можно получать методом выполнения какого-то графового алгоритма, не теряя общности, когда алгоритм запускается на входном графе, в таком случае граф должен быть иерархическим ориентированным графов с портами, возможно несвязным, если в изображении используются вспомогательные структуры данных из алгоритма. Например, стек из алгоритма обхода в глубину.

Модель обрабатывает полученный поток событий изменений, сверяясь со своим внутренним состоянием, и преобразовывая каждое изменение в серию изменений графического состояния подсистемы вывода, о которой речь пойдёт в следующем разделе.

5. Визуальное представление графовой модели

В каждом программном проекте существует подсистема вывода информации. В рамках данной работы подсистема вывода является графическим интерфейсом, основанным на технологии SVG, которая поддерживается большинством современных браузеров. Обычно для изображения элементов графов используются различные геометрические фигуры. Для описания двумерной векторной графики существует язык SVG, который также является расширением языка XML [6]. Язык SVG поддерживает такие фигуры как эллипсы, прямоугольники и более сложные фигуры, задаваемые с помощью последовательностей точек. Данный язык поддерживается основными браузерами, что делает его подходящим для представления графов при визуализации в контексте реализации облачного сервиса. Будем строить изображение графов по следующим правилам. Вершины будем представлять с помощью прямоугольников с границей. Порты будем представлять с помощью точек с координатами, заданными относительно содержащих их вершин. Порты вершин будем располагать только на верхних или нижних гранях прямоугольников, изображающих вершины. Координаты вершин также задаются относительно содержащих их вершин-графов. Дуги будем представлять с помощью ломаных линий или кривых Безье. Последние поддерживаются форматом SVG естественным образом [6]. Кроме того, следует отметить, что при использовании SVG изображений в браузерах, изображения естественным образом включаются в DOM модель документа [7]. Данное обстоятельство позволяет использовать стандартные методы для обеспечения интерактивного взаимодействия с построенным изображением. Например, это позволяет с помощью стандартных браузерных событий реализовать сворачивание и разворачивание вершин графа, что облегчает навигацию по изображению. Также естественное включение в DOM позволяет использовать существующие средства для поиска вершин в графе по заданным значениям атрибутов, например, по имени функции.

На рис. 1 представлен пример изображения внутреннего представление IR программы языка Cloud Sisal с помощью технологии SVG [1].

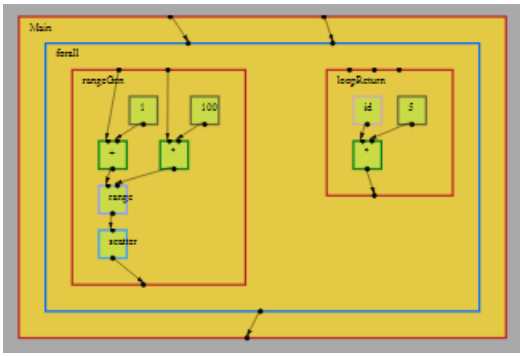

Рис. 1. Граф внутреннего представления программы, содержащей цикл.

6. Визуальные эффекты и отображение контекстной информации

Визуальные эффекты предназначены для моделирования графического отображения изменений в графовой модели. Простой визуальный эффект можно описать как непрерывную функцию F: $[0,1] \to S$, где S это множество векторов (l, w, c), описывающих визуальное состояние геометрических линий, где l это вектор точек, задающий форму линии, w это толщина линии, а c это цвет линии.

Составные визуальные эффекты представляют собой последовательности простых эффектов, где каждый следующий эффект запускается после предыдущего. Таким образом составной эффект можно описать как функцию:

$$F^c(t) = \begin{cases} F_1^S(t * 2), & 0 \le t \le \frac{1}{2} \\ F_2^S(t * 2), & \frac{1}{2} < t \le 1 \end{cases}, \quad (1)$$

в случае, если составной эффект состоит из двух эффектов, и первый из них действует половину времени, а второй оставшуюся половину времени.

В более общем случае эффект может быть задан следующим образом:

$$F^c(t) = \begin{cases} F_1(t * |d_1|), & t \in d_1 \\ F_2(t * |d_2|), & t \in d_2 \\ \dots \\ F_n(t * |d_n|), & t \in d_n \end{cases}, \quad (2)$$

где $d_1, d_2, \dots, d_n$ это интервалы, в объединении покрывающие отрезок $[0.1]$.

Таким образом, если требуется отображать изменение значения атрибута N вершин от значения A до значения B в течение, например, двух секунд, то для этого можно использовать функцию $F_N(t)$, где $F_N(0) = A$, а $F_N(1) = B$, причём для получения промежуточных значений использовать выражение $F_N(x/2000)$, где x меняется от 0 до 2000 миллисекунд.

При отображении изменений значений атрибутов элементов графовой модели, для которого используются функции визуальных эффектов, также полезно отображать контекстную информацию, соответствующую контексту, где происходит изменение значения атрибута. Так, например, при перечислении смежных вершин для заданной вершин происходит последовательный перебор элементов из множества смежных вершин, причём для

Работа опубликована при финансовой поддержке РФФИ, грант 18-07-20045\18

каждой отдельно итерации перебора, множество смежных вершин является контекстной информацией.

Для отображения контекстной информации также применимы визуальные эффекты с тем замечанием, что контекстная информация отображается графически в течение всего времени жизни контекста. Например, множество смежных вершин графически выделено всё время, пока производится перебор смежных вершин.

Другими примерами контекстной информации могут служить отображение смежной вершины из множества в рамках итерации цикла, который осуществляет перебор, или отображение подграфа, на котором производится вычисление внутри подпрограммы, если программа использует подпрограммы для осуществления вычислений.

7. Реализация модели

Реализация предложенной модели визуализации предназначена для использования в качестве компонента подсистемы отладки в системе CPPS, где, при интеграции с компонентом, интерпретирующем отлаживаемую Cloud Sisal программу, компонент визуализации графовой модели использует интерпретатор в качестве источника событий, отражающих изменения в графовой модели внутреннего представления Cloud Sisal программы. Предложенная модель позволяет отображать поток событий, отражающих изменения графовой модели, в множество взаимозависимых анимаций над визуальными образами элементов графа, а также осуществлять приостановку анимаций, и запуск анимаций в обратном направлении. На момент написания реализация модели находится на стадии технического демо, представляющего собой автономное вебприложение, реализованного на языке JavaScript, где вместо потока событий от интерпретатора Cloud Sisal программы используется симуляция ожидаемых событий.

8. Заключение

Целью данной работы является описать решение задачи визуализации графовых алгоритмов с помощью модели, описывающей каждое изменение в графовой модели под воздействием алгоритма, как графический эффект, отображающий изменения в визуальных стилях, которые соответствуют изменениям в атрибутах графовой модели. Каждое изменение атрибута, добавление вершины или удаление ребра отражается на визуальном отображении анимацией графических стилей. При графическом отображении графов обычно требуется тем или иным способом решить задачу определения размеров вершин графа и укладки графа на плоскость, представляя вершин геометрическими фигурами, а дуги ломаными или гладкими кривыми. В случае, если при графическом отображении изменений в графовой модели изменения соответствуют добавлению вершин, портов или дуг, то задачу укладки приходится решать на лету. Существенной трудностью здесь является сохранение относительного расположения вершин после удаления, так как алгоритм укладки может существенно изменить финальное графическое изображение, передвинув оставшиеся вершины в неожиданные места. Для практического применения существует метод, который существенно упрощает реализацию модели.

Графовая модель внутреннего представления модифицируется добавлением служебного атрибута ко всем элементам графа, обозначающим «видимость». Если значение атрибута задано, то вершина должна быть отображена графически, если не задано, то соответственно вершина должна быть невидима. Также меняется логика вычислений над множествами вершин, портов и дуг графа так, что, если значение атрибута видимости задано, то вершина при операциях перечисления элементов в множестве должна быть доступна для перечисления, а если значение не задано, соответственно, вершин должна быть недоступна для перечисления. Также меняется алгоритм укладки графой модели на плоскость, и вычисление будет проходить по следующей схеме:

1. Потребуется взять всю последовательность графов из динамического графа DG', объединить множества вершин всех элементов последовательности, а также множества портов и дуг.
2. Применить алгоритм укладки.
3. Пометить все элементы первого графа из последовательности DG' как видимые с помощью атрибута «видимости».

Соответственно, при отображении событий добавления или удаления вершин, вместо удаления проставляется или очищается значение атрибута видимости.

Данный метод позволяет решить задачу укладки более простым способом, чем перевычислять укладку при очередном удалении или добавлении вершин в графовую модель в случае, если операций удаления или добавления значительное количество, относительно общего числа изменений графовой модели.

9. Благодарности

Исследование выполнено за счет гранта Российского научного фонда (проект 18-11-00118).

10. Литература

[1] Гордеев Д.С. визуализация внутреннего представления программ на языке Cloud SISAL // Научная визуализация. - 2016.- Том. 8, N 2.- С.98 - 106.

[2] Demetrescu C., Finocchi I., Stasko J. T., Specifying Algorithm Visualizations: Interesting Events or State Mapping? // In Proc. of Dagstuhl Seminar on Software Visualization – Lect. Notes in Comput. Sci. – 2001. – P. 16–30.

[3] Касьянов, В. Н., Евстигнеев, В. А. Графы в программировании: обработка, визуализация и применение. – СПб.: БХВ-Петербург, 2003. – 1104 с.– ISBN 5-94157-184-4.

[4] Касьянов В. Н., Касьянова Е. В. Язык программирования Cloud Sisal. – Новосибирск, 2018. – 42 с. – (Препринт/ РАН, Сиб. отд-ние, ИСИ; N181).

[5] Zaki A. Comprehensive survey on dynamic graph models // IJACSA – 7(2). – 2016.

[6] http://www.w3.org/TR/SVG/

[7] http://www.w3.org/DOM/

[8] Касьянов В. Н., Касьянова Е. В. Методы и система облачного параллельного программирования // Проблемы оптимизации сложных систем: Материалы XIV Международной Азиатской школы-семинара - Алматы, 2018. - Часть 1. - С. 298-307.

[9] Federico P., Aigner W., Miksch S., Windhager F., Zenk L.: A visual analytics approach to dynamic social networks. In Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies - 2011. - p. 47.

Об авторе

Гордеев Дмитрий Станиславович, м.н.с лаборатории конструирования и оптимизации программ института систем информатики СО РАН.

The publication of the conference proceedings was supported by the RFBR, grant No. 18-07-20045\18