РАЗРАБОТКА АЛГОРИТМА УКЛАДКИ ДИАГРАММ СОСТОЯНИЙ

М.А.Коротков
Санкт-Петербургский государственный университет информационных технологий, механики и оптики
Тел.: (812) 324-32-11, e-mail: mkorotkov@evelopers.com

Программный пакет с открытым исходным кодом *UniMod* [1] обеспечивает разработку и выполнение автоматно-ориентированных программ. Он позволяет создавать и редактировать диаграммы классов и состояний *UML* [2], которые соответствуют графу переходов и схеме связей конечного автомата [3]. После создания диаграмм существует возможность выполнить их, при этом содержимое диаграмм преобразуется в *XML*-описание, которое передается интерпретатору, также входящему в пакет *UniMod*. Такой подход является реализацией парадигмы автоматного программирования [4].

Во многих современных редакторах для текстовых языков программирования существует возможность автоматического форматирования программного кода. В случае если программным кодом является не текст, а набор диаграмм, задача форматирования текста преобразуется в задачу укладки диаграмм. Более подробно задача будет описана ниже. В рамках проекта *UniMod* основной интерес представляет укладка диаграмм состояний языка *UML*.

При укладке диаграммы необходимо учитывать ряд критериев, которые могут противоречить друг другу, например, критерии минимизации площади, занимаемой диаграммой, и наличия достаточного количества свободного места («воздуха»). Каждый критерий оценивает «качество» диаграммы для того или иного применения (отображения на мониторе, печати т.д.). Такие критерии называются **эстетиками**. Задавшись некоторым набором эстетик, можно построить штрафную функцию (которая тем больше, чем больше расхождения между изображением диаграммы и критериями), и пытаться минимизировать её, перемещая элементы, или разработать алгоритм, последовательно модифицирующий исходную диаграмму так, чтобы результат соответствовал выбранным критериям.

Рассмотрим диаграмму состояний конечного автомата. Она включает в себя состояния и переходы (и у тех, и у других есть дополнительные, связанные с ними, сущности – метки, но мы сейчас не будем останавливаться на этом вопросе). Диаграмме состояний (без вложенных состояний) можно сопоставить граф [5] (при укладке диаграммы состояний нам не важна ориентация дуг). Каждому состоянию соответствует вершина, а переходу – ребро. Нам необходимо несколько сузить поле деятельности, зафиксировав представление элементов и тип носителя, на котором мы планируем изображать диаграмму.

Диаграмма может быть изображена на плоскости или, например, может быть построена объемная модель [6]. Для представления на мониторе современного персонального компьютера наиболее естественным представляется выбор плоского носителя.

Нам необходимо формально оценивать качество укладки (точнее качество изображения, полученного по некоторой укладке). Наиболее точный ответ способен дать человек. Попробуем выделить набор эстетик (похожий набор эстетик предлагают авторы [7]), по которым мы будем оценивать качество укладки. Основная задача – обеспечить «читаемость» графа (однозначность представления информации):

- **минимизация пересечений**: количества пересечений ребер, прохождений ребер по вершинам и наложения вершин должны быть минимальны;
- **минимизация площади**: площадь, занимаемая выпуклой оболочкой уложенного графа (или площадь минимального прямоугольника, включающего в себя уложенный граф), должна быть минимальной;
- **ограничение «свободного места»**: доля свободного места на диаграмме не должны быть меньше некоторого предела;
- **минимизация изломов**: количество изломов должно быть минимально;
- **минимизация общей длины ребер**: суммарная длина ребер должна быть минимальна;
- **минимизация коэффициента сторон**: отношение длины большей стороны к длине меньшей стороны объемлющего граф прямоугольника должно быть минимально;
- **унификация длин ребер**: минимизация различий между длинами ребер на графе.

К примеру, укладка слева на рис.1 значительно лучше читается, чем укладка справа.

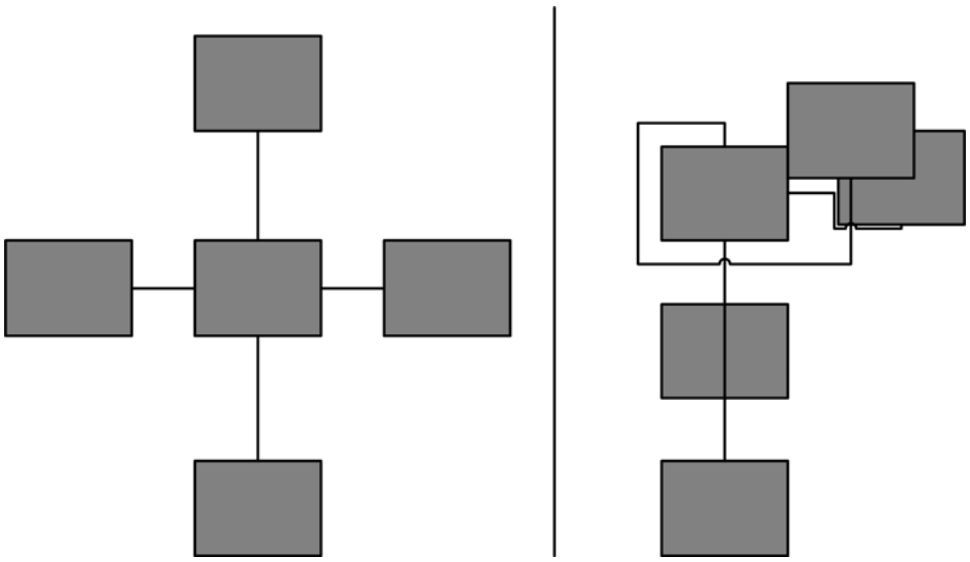

Рис. 1 Укладки графа

Граф, соответствующий диаграмме состояний, отличается от обычного графа возможностью вложения состояний друг в друга, наличием псевдосостояний, а также выделенных начальных и конечных состояний. Для оценки качества его укладки необходимо учесть дополнительные критерии:

- **унификация размеров состояний**: размер всех простых состояний (состояний, не имеющих вложенных состояний) необходимо максимально приблизить к заданному оптимальному размеру;
- **разнесение начального и конечного состояний**: расстояние между начальным и конечным состоянием должно быть достаточно велико.

Заметим, что приведенные нами эстетики не являются строгими и лишь дают понять, какими соображениями мы будем руководствоваться, оценивая результаты работы программ.

Обратим внимание на то, что в принятые в диаграмме состояний UML начальное и конечное состояние не имеют прямоугольной формы. Для простоты заменим объемлющими прямоугольниками.

Наша задача состоит в построении качественной ортогональной укладки диаграммы состояний. В разрабатываемых алгоритмах мы можем пренебрегать некоторыми из перечисленных выше критериев.

На базе данных критериев разработано два алгоритма укладки диаграмм состояний *UML,* принадлежащие к различным семействам алгоритмов укладки. Первый алгоритм – метод отжига с применением ортогонализации (рис. 2), второй – модифицированный алгоритм ортогональной укладки GIOTTO (рис. 3).

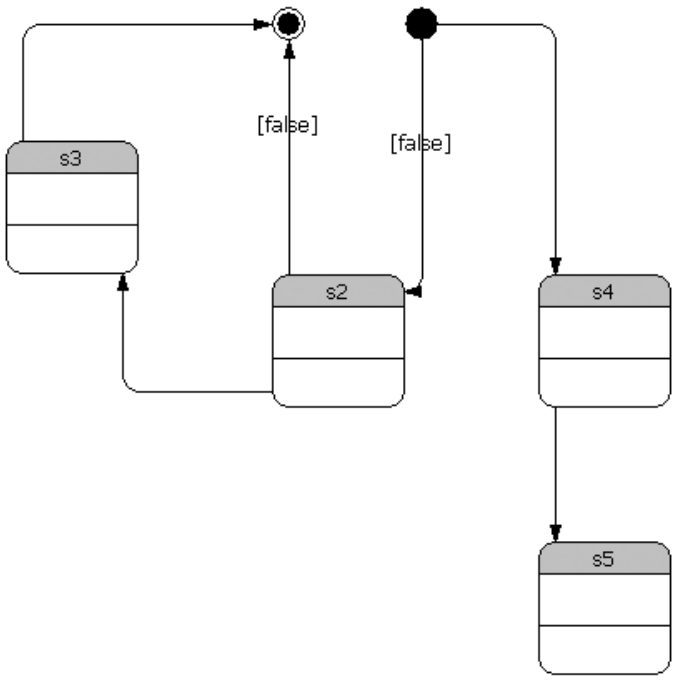

Рис. 2 Укладка методом отжига

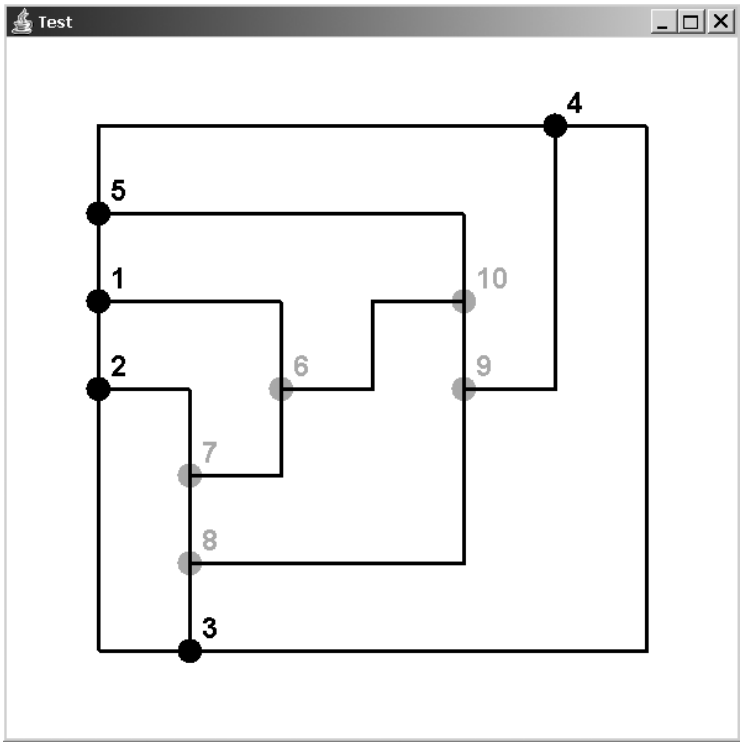

Рис. 3. Укладка GIOTTO

Алгоритм ортогонализации, примененный к методу отжига, а также модификации алгоритма GIOTTO [8] для применения его к диаграмме состояний разработаны автором. Библиотека, реализующая перечисленные выше алгоритмы, входит в состав программного пакета с открытым исходным кодом UniMod.

Применение качественных алгоритмов укладки делает работу с диаграммами значительно более комфортной, что увеличивает производительной разработчика. Кроме того, наличие алгоритма укладки принципиально важно для организации импорта файлов, созданных в сторонних редакторах, ведь для UML-редакторов не существует унифицированного механизма обмена данными, сохраняющего информацию о местоположении элементов [9].

Литература

1. Гуров В.С., Мазин М.А. Веб-сайт проекта UniMod. **http://unimod.sourceforge.net/**.

2. Буч Г., Рамбо Д., Джекобсон А. Язык UML. Руководство пользователя. М.: ДМК. 2000.

3. Шалыто А.А. Switch-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука. 1998.

4. Шалыто А.А., Туккель Н.И. Танки и автоматы //BYTE/Россия. 2003. №2, с. 69-73. **http://is.ifmo.ru/** (раздел «Статьи»).

5. Новиков Ф.А. Дискретная математика для программистов. 2-е издание. СПб.: Питер. 2004.

6. Tamassia R. Advances in the Theory and Practice of Graph Drawing. **http://www.cs.brown.edu/people/rt/papers/ordal96/ordal96.html**.

7. Касьянов В.Н., Евстигнеев В.А. Графы в программировании: обработка, визуализация, применение. – СПб.: БХВ-Петербург, 2003.

8. Battista G., Eades P., Tamassia R., Tollis I. Graph Drawing. Algorithms for the Visualization of Graphs. – New Jersey: Prentice Hall. 1999.

9. Frankel D., Model Driven Architecture. Applying MDA to Enterprise Computing. – Indianapolis: Web Publishing Inc., 2003.