Visual Graph – система визуализации сложно структурированной информации большого объема на основе графовых моделей *

В.Н. Касьянов[1,2]***, Т.А. Золотухин***[1]
kvn@iis.nsk.su|tzolotuhin@gmail.com
[1]Институт систем информатики им. А.П. Ершова СО РАН, Новосибирск, Россия;
[2]Новосибирский государственный университет, Новосибирск, Россия

Доклад посвящен создаваемой в лаборатории конструирования и оптимизации программ ИСИ СО РАН универсальной системе Visual Graph для визуализации сложно структурированной информации большого объема на основе графовых моделей.

Ключевые слова: *атрибутированные иерархические графы, визуализация графов большого размера, визуализация информации на основе графовых моделей, интерактивная визуализация, системы визуализации.*

Visual Graph – a system for visualization of big size complex structural information on the base of graph models*

V.N.Kasyanov[1,2]***, T.A.Zolotuhin***[1]
[1]Institute of Informatics Systems, Novosibirsk, Russia;
[2]Novosibirsk State University, Novosibirsk, Russia

The paper is devoted to the Visual Graph system being under development by the laboratory for program construction and optimization of Institute of Informatics Systems as an universal system for visualization of big size complex structural information on the base of graph models.

Keywords: attributed hierarchical graphs, big size graph visualization, information visualization based on graph models, interactive visualization, visualization systems.

Введение

Визуализация информации играет большую роль в жизни человека. Считается, что около 90% всей получаемой информации человек получает через зрение. Человечество за тысячи лет преодолело путь от простейших способов визуализации в виде наскальных рисунков до карт, схем и диаграмм. В настоящее время визуализация на основе теоретико-графовых моделей – неотъемлемый элемент обработки сложной информации о строении объектов, систем и процессов во многих приложениях в науке и технике [1-5]. На рынке широко представлены наукоемкие программные продукты, использующие методы визуализации информации на основе графовых моделей. В основном это многочисленные специализированные системы или встроенные компоненты систем, ориентированные на визуализацию определенных подклассов графовых моделей и/или специальные применения, но есть и универсальные системы, предназначенные для визуализации графов общего вида и широкого назначения, такие как Higres [6, 7], aiSee [8], yEd [9] и Cytoscape [10].

Поскольку информация, которую желательно визуализировать, постоянно увеличивается и усложняется, возникает все больше ситуаций, в которых классические графовые модели перестают быть адекватными. Требуются более мощные теоретико-графовые формализмы для представления информационных моделей, обладающих иерархической структурой. Иерархичность является основой многочисленных методов анализа и синтеза сложных информационных моделей в различных областях применения вычислительных систем. Одним из таких формализмов являются иерархические графы и графовые модели [11, 12].

В данном докладе будет рассмотрена универсальная система визуализации атрибутированных иерархических графов Visual Graph, которая распространяется под лицензией BSD и создается в рамках ведущегося в лаборатории конструирования и оптимизации программ ИСИ СО РАН проекта по разработке методов и средств для визуализации сложно структурированной информации большого объема на основе графовых моделей.

Область применения

При создании экспериментальной версии системы визуализации атрибутированных иерархических графов Visual Graph мы рассматривали в качестве основного ее использование разработчиками систем конструирования программ (таких как компилятор) для визуализации структур данных, возникающих при работе этих систем.

Структуры данных, возникающие в таких системах, естественным образом представляются в ви-

Работа выполнена при финансовой поддержке РФФИ, грант 15-07-02029 и опубликована при финансовой поддержке РФФИ, грант 15-07-20347.

де теоретико-графовых моделей большого размера. Так, например, синтаксические деревья используются в качестве внутреннего представления транслируемых программ почти во всех компиляторах или интерпретаторах. Оптимизирующие и реструктурирующие компиляторы требуют выявление в транслируемой программе управляющих и информационных связей и их представление в виде более общих теоретико-графовых моделей, таких как, например, управляющий граф.

Номенклатура этих графовых моделей в той или другой системе конструирования может своя, отличная от других. Более того, представления одинаковых графовых моделей могут различаться между собой в разных системах и даже в разных версиях одной и той же системы. Однако все эти теоретико-графовые модели являются частными случаями атрибутированных иерархических графов.

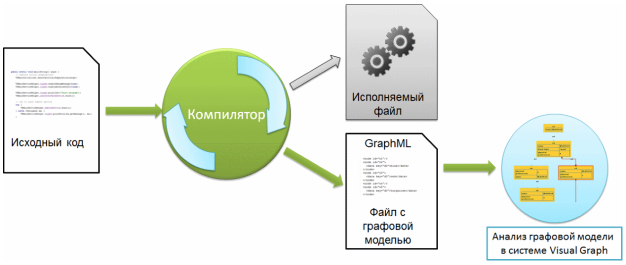

Рис. 1: Использование системы Visual Graph

Поэтому предполагается следующий сценарий использования системы Visual Graph. Сначала компилятор (сам либо с помощью вспомогательной программы) переводит графовую модель из внутреннего представления в файл одного из поддерживаемых системой Visual Graph форматов, как правило, GraphML-файл (Рис. 1). После этого система Visual Graph сможет прочитать эту графовую модель из файла, визуализировать ее и предоставить пользователю средства навигации по ней.

Иерархические графы и графовые модели

Рассмотрим ряд определений из [12].

Пусть G обозначает граф произвольного вида, элементы (вершины и ребра) которого отличаются один от другого какими-либо пометками, называемыми их *именами*, например, G может быть неориентированным или ориентированным графом, мультиграфом (с кратными ребрами) или псевдографом (с петлями).

Граф C называется *фрагментом* графа G, обозначаем $C \subseteq G$, если C – часть графа G, т. е. C образован подмножеством элементов графа G.

F – *иерархия фрагментов* графа G, если F – такое множество фрагментов графа C, что $G \in F$ и для любых двух фрагментов C_1 и C_2 из F либо фрагменты C_1 и C_2 не пересекаются, либо один из них является частью (*подфрагментом*) другого. Фрагмент G – *основной* (*главный*) фрагмент иерархии F. Фрагмент $C \in F$ – *элементарный*, если в F нет фрагментов G, являющихся подфрагментами фрагмента C.

Пусть задана некоторая иерархия фрагментов F графа G. Для любых $C_1, C_2 \in F$ фрагмент C_1 – *прямой* подфрагмент C_2 (или, что то же самое, фрагмент, *непосредственно* вложенный в C_2), если C_1 – подфрагмент C_2 и не существует такого $C_3 \in F$, отличного от C_1 и C_2,что $C_1 \subseteq C_3 \subseteq C_2$.

Иерархический граф $H = (G, T)$ состоит из графа G и корневого дерева T, вершины которого соответствуют элементам некоторой иерархии в G, а дуги отражают отношение их непосредственной вложенности. T называется *деревом вложенности*, а G – *основным графом* иерархического графа H.

Важный частный случай иерархических графов образуют так называемые *простые* иерархические графы, в которых все фрагменты являются подграфами основного графа. В частности, он содержит составные и кластерные графы.

Поскольку подграфы однозначно определяются множествами своих вершин, есть возможность определять простой иерархический граф H как пару (G, T), состоящую из основного графа G и *вершинного дерева вложенности* T, удовлетворяющего следующим условиям. Вершины дерева T соответствуют некоторым подмножествам вершин основного графа G таким образом, что подмножества вершин, соответствующие листьям T, образуют разбиение множества всех вершин графа G на одноэлементные подмножества. Дуги дерева T отражают непосредственную вложенность соответствующих подмножеств. Такое представление иерархического графа называется *вершинным* (Рис. 2).

Под графовой моделью в общем случае мы понимаем класс графовых объектов, имеющих вид атрибутированных (помеченных) графов, с заданным на нем отношением эквивалентности [12].

При этом при задании графовой модели будем различать статическую (или синтаксическую) часть описания, определяющую класс помеченных графов, образующих указанную модель, и динамическую (или семантическую) часть, задающую разбиение данного класса графов на подклассы попарно эквивалентных.

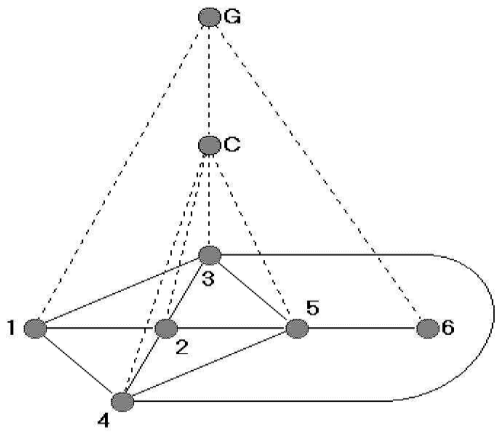

Рис. 2: Вершинное представление простого иерархического графа с двумя нетривиальными фрагментами: сплошные линии – это ребра основного графа, штриховые линии – ребра вершинного дерева вложенности

Пусть имеется множество объектов V, называемых *метками*, распадающееся на попарно непересекающиеся подмножества *классов* меток. В качестве классов меток могут использоваться определенные множества чисел, символов, строк (цепочек символов), формул, графов и объектов других видов.

Пусть задано множество объектов W, называемых *типами*, и пусть каждому элементу $w \in W$ поставлено в соответствие множество *пометок* $V(w)$, имеющее вид декартового произведения $V_{i_1} \times V_{i_2} \times \times \ldots \times V_{i_s}$, где $V_{i_j} \in V$ – некоторый класс меток для любого j.

Со статической точки зрения *иерархическая графовая модель* – это тройка (H, M, L), где H – иерархический граф, M – *функция типа*, приписывающая каждому элементу (вершине, ребру и фрагменту) h иерархического графа H его тип $M(h) \in W$, а L – *функция меток*, приписывающая каждому элементу h графа H его пометку – некоторый элемент $L(h) \in V(M(h))$.

При изображении графовой модели тип ее элементов может быть связан с определенной геометрической формой соответствующих представлений и/или их цветовой гаммой, а также местом и способом представления пометок, относящихся к элементам соответствующего типа.

Что касается динамической части иерархической графовой модели, то она привносит в визуализацию графовых моделей различные анимационные аспекты.

Обычно используется два основных подхода к заданию семантической части графовой модели: путем явного задания набора *инвариантов* (свойств, присущих всем эквивалентым между собой моделям), который различает классы эквивалентности графовых моделей, либо через так называемые *эквивалентные* преобразования графовых моделей, которые сохраняют указанный набор инвариантов. Оба подхода к заданию семантической части графовой модели опираются на преобразования графов и активно развиваются в рамках теории схем программ.

Язык GraphML

Долгое время среди используемых форматов представления графов не находилось ни одного, который был бы достаточно широко принятым в качестве стандартного. По-существу, инструменты работы с графами поддерживали (да и сейчас многие из них поддерживают) лишь некоторую часть из существующих клиентских форматов, обычно состоящую из видов графовых представлений, ограниченных по выразимости и специфике конкретной областью применения.

Поэтому не случайно в 2000 г. наблюдательный комитет симпозиума по рисованию графов (Graph Drawing Steering Committee) инициировал работы по выработке формата обмена графами GraphML, который, в конечном счете, должен лечь в основу стандарта описания графовых моделей.

Основную цель, которую ставили перед собой разработчики языка, они сформулировали следующим образом. Формат обмена графами должен быть в состоянии представлять произвольные графы с произвольными дополнительными данными, включая укладку и графическую информацию. Дополнительная информация должна сохраняться в формате, подходящем для заданного конкретного приложения, но не должна усложнять представление данных из других приложений или мешать ему.

GraphML проектировался с ориентацией на эту цель, а также с учётом следующих более прагматических требований:

1) Простота (Simplicity). Формат должен быть простым для разбора и интерпретации как людьми, так и машинами. В качестве общего принципа формулируется отсутствие неоднозначностей и таким образом существование единственной хорошо-определенной интерпретации для каждого валидного (valid) GraphML-документа.
2) Общность (Generality). Не должно существовать ограничений по отношению к графовой модели, т. е. гиперграфы , иерархические графы и т. д. должны быть выразимы с помощью одного и того же базисного формата.
3) Расширяемость (Extensibility). Должна существовать возможность расширять формат хорошо-определённым способом для представления дополнительных данных, требуемых произ-

вольными приложениями или более сложным использованием (например, посылая алгоритм раскладки вместе с графом).

4) Робастность (Robustness). Система, не способная обработать весь диапазон графовых моделей или дополнительной информации, должна быть в состоянии легко распознавать и извлекать то подмножество, которое она может обработать.

Разработанный язык описания графов GraphML базируется на XML [13]. Он позволяет описывать ориентированные, неориентированные и смешанные графы, гиперграфы и иерархические графы, а также любые специфичные для конкретных приложений атрибуты. В частности, язык GraphML полностью поддерживает атрибутированные иерархические графы. Благодаря XML синтаксису, GraphML может использоваться в комбинации с другими форматами, основанными на XML. С другой стороны, свой собственный механизм расширения позволяет прикреплять <data> метки со сложным содержимым (возможно требуемый для исполнения с другими моделями XML содержимого) элементов GraphML.

Возможности системы

Пользовательский интерфейс системы изображен на рис. 3. Он включает рабочий стол, миникарту, навигационную панель, а также атрибутную панель.

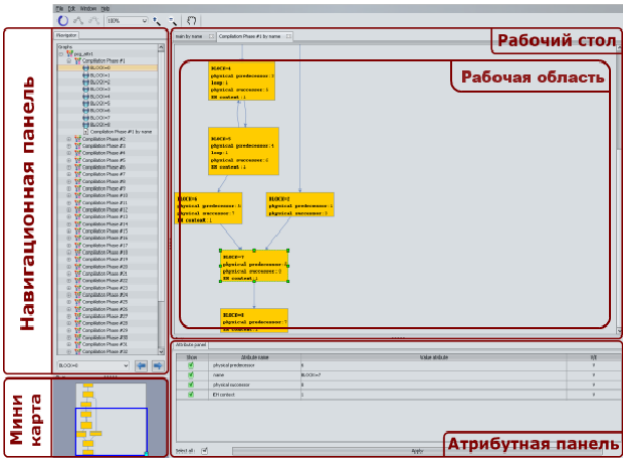

Рис. 3: Пользовательский интерфейс системы Visual Graph

Рабочий стол. Рабочий стол состоит из набора вкладок, открываемых пользователем для визуализации выбранной части графовой модели в виде ее изображения на плоскости. Для улучшения полученного автоматически изображения, пользователь может менять форму вершин и ребер, раскладку и ее параметры, отображаемые атрибуты, масштаб видимой области и многое другое.

Миникарта. Данный инструмент позволяет обозревать целиком весь граф, показанный в текущей вкладке, а также помогает перемещать и масштабировать видимую его область – ту часть графа, которая видна в текущей вкладке.

Навигационная панель. Навигационная понель предназначена для визуализации всех графов, с которыми работает пользователь, в виде изображения с помощью отступов деревьев вложенности фрагментов этих графов. Для быстрого поиска по деревьям реализована строка поиска, которая позволяет пользователю без труда найти интересующие его элементы, используя регулярные выражения. После чего пользователь может выделить интересующие его элементы и открыть их в новой вкладке.

Атрибутная панель. Атрибутная панель – инструмент, который позволяет управлять визуализацией атрибутов для выбранных вершин и ребер в текущей вкладке. Для этого пользователю необходимо выделить у графа из текущей вкладке те вершины и ребра, которые ему интересны, после чего отметить в атрибутной панели галочками те атрибуты, которые он хочет визуализировать у этих элементов.

Так же с помощью данного инструмента можно задать набор атрибутов, которые будут видны у элементов графа, открытых в новой вкладке.

Фильтр. Фильтр – инструмент, который поддерживает поиск в текущей вкладке элементов (вершин и ребер) графовой модели по их атрибутам. Условия поиска задаются пользователем с использованием имен атрибутов и их значений (Рис. 4). Заданные пользователем условия могут быть объединены в выражения с помощью логических операций и скобок. Результатом работы фильтра является набор всех тех элементов графовой модели, которые удовлетворяют заданным условиям (Рис. 5).

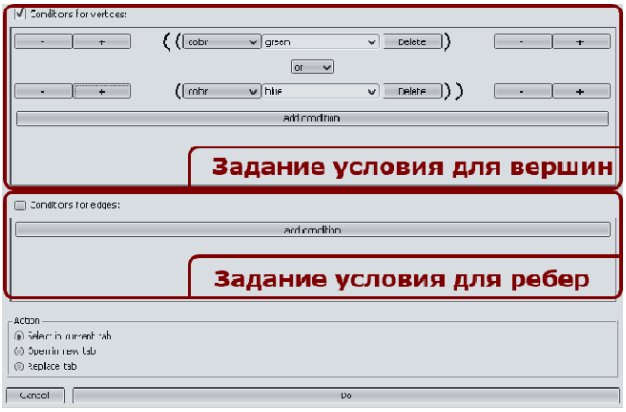

Рис. 4: Фильтр

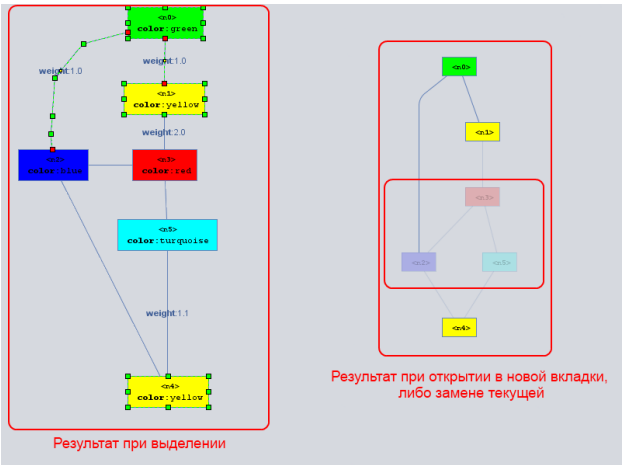

Рис. 5: Результат работы фильтра

Блокнот. Блокнот – инструмент, позволяющий загружать дополнительную информацию в виде текстовых файлов и связывать ее с графовой моделью. После этого пользователь может переходить от элементов графовой модели к связанной с ними дополнительной текстовой информации. Например, это можно использовать для связи синтаксического дерева с исходным кодом.

Средства структурного анализа. К этим средствам относятся различные алгоритмы работы с графовыми моделями, которые помогают пользователю выделять и визуализировать нужную ему информацию в их изображениях. К ним относятся, например, такие средства, как подсветка кратчайшего пути, циклов, обязательных предшественников и т. д., а также поиск максимального общего подграфа двух графов [14]. Результат работы такого поиска изображен на Рис. 6.

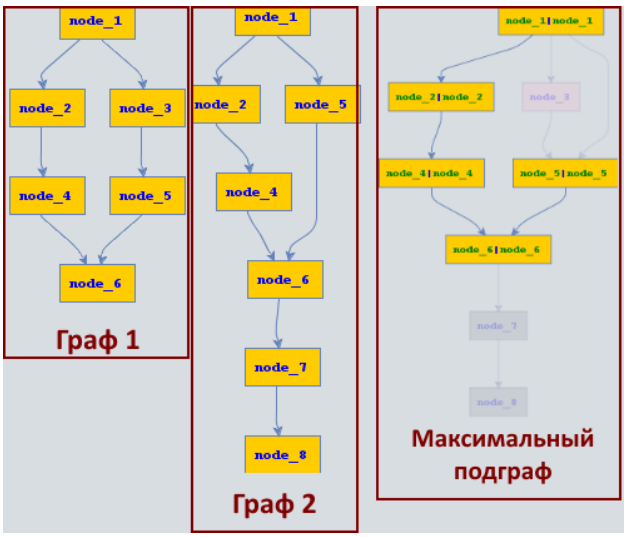

Рис. 6: Пример работы алгоритма поиска максимального общего подграфа двух графов

Заметим, что задача нахождения максимального общего подграфа двух графов является NP-трудной, но возникает перед разработчиком компилятора довольно часто. Например, когда он хочет найти разницу в поведении разных версий одного и того же компилятора. Эту разницу можно увидеть благодаря графам, возникающим внутри компилятора в момент компиляции. В частности, благодаря ним можно узнать какие оптимизации были сделаны или не сделаны в одной версии в отличии от оптимизаций, реализованные в другой версии.

При небольших графах (порядка 5 вершин), пользователь может визуально пытаться решить эту задачу, не прибегая к дополнительному инструментарию. Но стоит увеличить количество вершин до 15 – 20 и эта задача уже не будет столь тривиальной.

Раскладчики. Задача раскладки графа на плоскости не имеет и не может иметь оптимального решения в связи с тем, что наборы критериев для оценки качества раскладки различны для различных приложений [1 – 5]. Известно, что для тех или других приложений одни критерии могут вносить большую значимость в наглядность графа, чем другие. Тем не менее, для конкретных задач (и связанных с ними классов визуализируемых графов) обычно можно подобрать виды раскладок, их примерные параметры и алгоритмы.

Основные графы, с которыми работает компилятор, имеют относительно небольшое количество элементов при их представлении в виде иерархических графов, когда надо визуализировать в отдельных вкладках фрагменты, в которых вложенные фрагменты свернуты в вершины. Как правило, алгоритм поуровневой укладки (иерархический раскладчик) хорошо справляется с такими видами графов. В зависимости от размера графа в нем можно использовать различные эвристики для минимизации ширины или высоты получаемой раскладки, а также для уменьшения числа пересечений между ребрами. Система Visual Graph предоставляет пользователю возможность использования нескольких раскладчиков, основным из которых является иерархический раскладчик (Рис. 7). При включении в систему он был максимально адаптирован для работы с графами, используемыми в компиляторах. Но при желании этот (и любой другой) раскладчик системы может всегда быть изменен и/или настроен под другие типы задач.

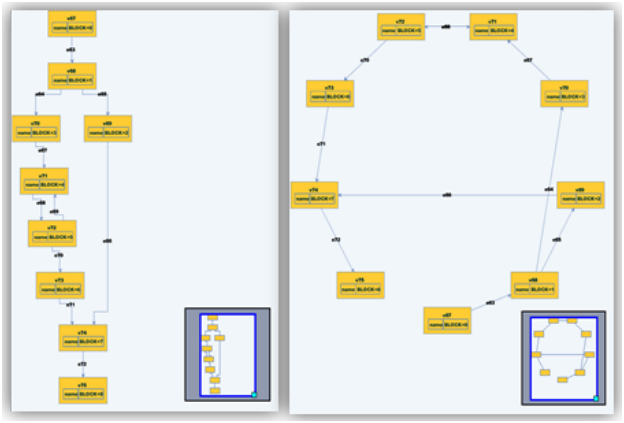

Рис. 7: Примеры раскладок управляющего графа

Особенности Реализации

Хранение графовых моделей внутри системы. Размер файла с входной графовой моделью может достигать сотни мегабайт, что не позволяет использовать оперативную память компьютера для хранения визуализируемых графов в системе. Поэтому было решено использовать кэширование данных на жесткий диск, используя для этого реляционную базу данных. В качестве реляционной базы данных была выбрана встраиваемая база данных SQLite [15]. В отличии от большинства популярных реляционных баз данных, для SQLite не требуется установка сервера, а вся клиент-серверная архитектура сводится к работе с файлами.

В системе Visual Graph часть данных расположена в базе данных, а часть в оперативной памяти, благодаря чему достигается высокая скорость при работе с графовыми моделями большого размера.

Расширяемость системы. Все возможности системы Visual Graph, в том числе по навигации, визуализации и структурному анализу, реализованы (и предоставляются пользователям системы) с помощью набора средств, который может расширяться как самими разработчиками системы Visual Graph, так и сторонними разработчиками.

Для достижения простой расширяемой системы было принято решение об использовании продукта Apache Felix [16], который в свою очередь является реализацией спецификации OSGi [17]. Данное решение является де-факто стандартным для подобного типа задач и позволяет разработчикам легко расширять систему за счет написания новых плагинов.

Основные аналоги системы

В настоящее время на рынке представлен достаточно широкий круг систем для визуализации графовых моделей. Наиболее известными из них являются: aiSee[8], yEd[9], Cytoscape[10] и Higres[6, 7].

Область применения, этих систем, частично совпадает с областью применения системы Visual Graph, но в целом несколько шире. Так многие из них рассчитаны на редактирование существующих графов и создание новых. В то время как система Visual Graph предназначена, исключительно, для просмотра уже существующих графовых моделей без возможности их редактирования.

Основными задачами, решаемыми системой Visual Graph, являются отображение графовых моделей и инструменты навигации по ним.

Поэтому в рамках этих двух задач и будем в основном рассматривать особенности каждой из перечисленных систем.

Система aiSee. Это – коммерческая система, которая автоматически строит раскладку исходного графа, описанного на специально разработанном языке GDL. Затем пользователь может интерактивно исследовать данный граф (без возможности редактирования), отпечатать его и сохранить в различных форматах. Первоначально система была разработана для визуализации структур данных, обрабатываемых компиляторами. На сегодняшний день ее используют десятки тысяч людей по всему миру в самых разных прикладных областях, в том числе:

— бизнес-менеджмент (структурные схемы предприятий, визуализация бизнес-процессов),
— генеалогия (генеалогические деревья, эвалюционные диаграммы),
— разработка программного обеспечения (блок-схемы, управляющие графы, графы вызовов функций),
— веб-дизайн и оптимизация сайтов (карты сайтов, графы движения пользователей, визуализация графов на страницах вики),
— спорт (турнирные деревья) и др.

В aiSee присутствует несколько раскладчиков, и множество настроек для них, которые в разной степени влияют на получаемый результат. Качество раскладок определенных типов графов системой весьма высокое.

В качестве инструментов навигации, aiSee предоставляет:

1. Рабочий стол – инструмент, который визуализирует графовую модель целиком, выдавая пользователю статичную картинку, которую нельзя изменять, например, передвигая элементы.
2. Поисковик – инструмент, позволяющий пользователю искать элементы графовой модели с помощью их имен. Он поддерживает задание регулярных выражений, выбор категорий элементов для поиска и сохранение предыдущих поисковых запросов.

Система yEd. Данная система предназначена для быстрого создания и редактирования высоко-

качественных диаграмм. Создание диаграмм происходит вручную или с помощью импортирования из внешних данных. После чего к полученной диаграмме можно применить богатый набор алгоритмов для проведения анализа с последующим получением необходимой информации.

В системе yEd присутствует большое количество раскладчиков и опций для них, которые предоставляют возможность тонкой настройки визуализации каждого графа пользователем.

В качестве инструментов навигации система yEd предоставляет следующие:

1. Рабочий стол, схожий с рабочим столом, который предоставляет система aiSee. Но в отличие от aiSee здесь есть возможность работать с несколькими графами одновременно в одном экземпляре программы (создается вкладка для каждого графа), а также есть возможность редактирования элементов графа, начиная от смены их положения и размеров до задания атрибутов, влияющих на их визуализацию.
2. Навигатор – инструмент, отображающий граф, находящийся в текущей вкладке.
3. Миникарта – инструмент, показывающий весь граф, находящийся в текущей вкладке, и выделяющий ту его часть, которая видна на вкладке.

Система Cytoscape. Cytoscape – свободная система с открытым исходным кодом для визуализации и анализа сетей. Основная область применения данной системы является биоинформатика. Данная система стала очень популярной благодаря тому, что легко расширяется, позволяя сторонним разработчикам писать для нее различные плагины. Cytoscape имеет как собственные алгоритмы раскладки, так и сторонние, среди которых можно отметить присутствие алгоритмов из yFiles. Стоит также отметить, что результаты, получаемые в системе yEd намного лучше, чем аналогичные результаты, получаемые в системе Cytoscape, с настройками раскладчиков по умолчанию. Выделим основные средства навигации, которые предоставляет система Cytoscape для решения данной задачи:

1. Рабочий стол, схожий с рабочим столом, который предоставляет система yEd. Отличие заключается в том, что система Cytoscape не умеет работать с иерархическими графами.
2. Миникарта, идентичная миникарте в системе yEd.
3. Атрибутная панель – инструмент, позволяющий отображать текущие атрибуты и задавать новые. Данный инструмент выглядит в виде таблицы, по горизонтали, которой расположены имена атрибутов, по вертикали список выделенных вершин, а в ячейках соответствующие значения того или иного атрибута для той или иной вершины.
4. Фильтр – инструмент, позволяющий осуществлять поиск вершин и ребер по заданным условиям на атрибутах.

Система Higres. Higres – первый отечественный универсальный визуализатор и редактор графовых моделей. Созданная в ИСИ СО РАН система Higres ориентирована на многооконную работу с простыми иерархическими графами (Рис. 8). Каждому фрагменту иерархического графа соответствует некоторый прямоугольник плоскости, внутри которого располагаются все его вершины. Кроме того, для каждого фрагмента можно открыть отдельное окно, в котором видны только вершины данного фрагмента и его подфрагменты. При этом каждый подфрагмент можно объявить закрытым – тогда изображаются только его контуры, либо открытым – тогда изображаются все его вершины и инцидентные им дуги. Для изображения контуров фрагментов в системе используется прием создания эффекта тени. Закрытые фрагменты выглядят слегка выступающими вверх – как будто они закрыты крышками, открытые же слегка утоплены вниз.

Важным отличием системы Higres от других универсальных систем визуализации является ее способность сохранять во внутреннем представлении и визуализировать не только сам граф, но и его семантику, представленную в виде системы типов атрибутированных вершин, дуг и фрагментов графа, а также библиотеки алгоритмов обработки – так называемых внешних модулей. Причем пользователь системы может легко управлять методами визуализации графовой модели, а также корректировать и доопределять ее семантику. Такой подход обеспечивает, с одной стороны, универсальность системы Higres, с другой – возможность ее специализации. Он также позволяет использовать систему как платформу для исполнения и анимации алгоритмов работы с иерархическими графами. Запустив внешний модуль, пользователь может регулировать параметры обработки графовой модели, прерывать алгоритм на любом шаге, просматривать в любую сторону последовательность изображений промежуточных результатов шагов работы алгоритма как в форме анимации, так и в покадровом режиме.

В качестве инструментов навигации Higres предоставляет:

1. Рабочий стол, который хотя и аналогичен рабочему столу системы aiSee, но в отличие от него дает возможность редактировать элементы визуализируемого иерархического графа, начиная от смены их положения, формы и размеров до задания типов и атрибутов элементов графа, влияющих на их визуализацию, а также закрывая или открывая те или другие фрагменты.

2. Миникарта, аналогичная миникарте в системе yEd.

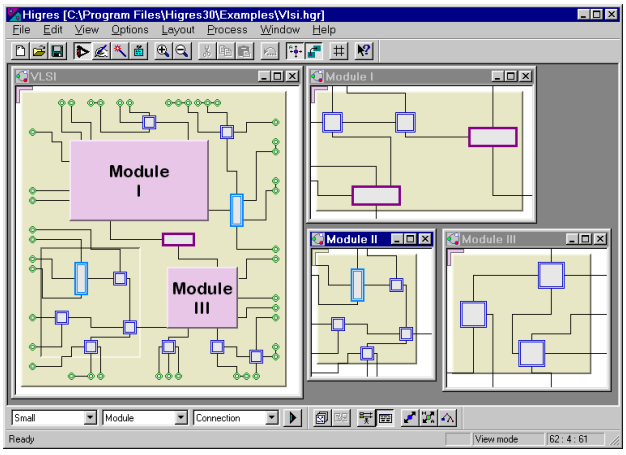

Рис. 8: Система Higres

Выводы

В докладе была рассмотрена система Visual Graph, которая создается в лаборатории конструирования и оптимизации программ ИСИ СО РАН для визуализации сложно структурированной информации большого объема на основе графовых моделей.

В отличие от зарубежных аналогов, таких как aiSee система Visual Graph поддерживает обработку произвольных атрибутированных иерархических графов (в том числе составных и кластерных графов) и использование для спецификации входного (визуализируемого) графа стандартного языка описания графов GraphML, ориентирована на пошаговое построение многооконного изображения графовой модели, состоящего из укрупненных изображений интересных пользователю фрагментов модели, и предоставляет богатые возможности для навигации по графовой модели и ее структурного анализа, работы с атрибутами ее элементов, а также простого расширения и настройки системы на нужды конкретного пользователя.

Созданная экспериментальная версия системы Visual Graph ориентирована на визуализацию структур данных, возникающих в компиляторах, позволяет одновременно работать с ними как в графовой, так и в текстовой форме, и обеспечивает плавность выполнения основных операций над графами, содержащими до 100000 элементов (вершин и дуг). Система проходит тестовое использование в компании Intel.

Ведется работа по встраиванию системы Visual Graph в визуальную среду параллельного программирования, создаваемую в рамках проекта по поддержке облачных супервычислений на базе языка Cloud Sisal [17, 18]. Цель проекта – дать возможность широкому кругу лиц, находящихся в удаленных населенных пунктах или в местах с недостаточными вычислительными средствами, разрабатывать параллельные программы на доступных персональных компьютерах, имеющих выход в Интернет, и оперативно дистанционно использовать для их исполнения вычислительные мощности, сосредоточенные в крупных вычислительных центрах.

Создаваемая среда должна любому пользователю, имеющему выход в Интернет, позволить без установки дополнительного программного обеспечения на своем рабочем месте в визуальном стиле создавать и отлаживать переносимые параллельные программы на языке Cloud Sisal, а также в облаке осуществлять эффективное решение своих задач, исполняя на некотором супервычислителе, доступном ему по сети, созданные и отлаженные переносимые Cloud-Sisal-программы, предварительно адаптировав их под используемый супервычислитель с помощью облачного оптимизирующего кросс-компилятора, предоставляемого средой.

Среда использует внутреннее представление Cloud-Sisal-программ, которое имеет вид атрибутированных иерархических графов.

Предоставляемые системой возможности по визуализации внутреннего представления программ будут позволять пользователю получать изображение структуры программы и отслеживать процессы конструирования, отладки и преобразования Cloud-Sisal-программы с помощью графических образов.

Литература

[1] *Di Battista G., Eades P., Tamassia R., Tollis I. G.* Graph Drawing: Algorithms for Visualization of Graphs. – PrenticeHall, 1999.– 397 p.

[2] *Herman I., Melancon G., Marshall M. S.* Graph visualization and navigation in information visualization: a survey// IEEE Trans. on Visualization and Computer Graphics. – 2000.– Vol. 6. – pp. 24-43.

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

[4] *Касьянов В. Н., Касьянова Е. В.* Визуализация графов и графовых моделей. – Новосибирск: Сибирское Научное Издательство, 2010. – 123 с.

[5] *Касьянов В. Н., Касьянова Е. В.* Визуализация информации на основе графовых моделей// Научная визуализация – 2014. – Т. 6, N 1. – С. 31 - 50.

[6] *Lisitsyn I. A., Kasyanov V. N.* Higres – visualization system for clustered graphs and graph algorithms // Lecture Notes in Computer Science. – 1999. – Vol.1731. – pp. 82-89.

[7] `http://pco.iis.nsk.su/higres` – Система Higres.

[8] `http://www.aisee.com` – Система aiSee.

[9] `http://www.yworks.com` – Система yEd.

[10] http://www.cytoscape.org – Система Cytoscape.

[11] *Kasyanov V. N.* Hierarchical graphs and visual processing // ICM 1998 International Congress of Mathematicians. Abstracts of Short Communications and Poster Sessions. – Berlin, 1998. – p. 292.

[12] *Касьянов В. Н.* Иерархические графы и графовые модели: вопросы визуальной обработки// Проблемы систем информатики и программирования.– Новосибирск, ИСИ СО РАН, 1999. – С. 7-32.

[13] http://graphml.graphdrawing.org/specification.html – Спецификация языка GraphML.

[14] *Золотухин Т.* Алгоритм поиска максимального подграфа двух графов и его реализация в рамках системы VisualGraph // Инновационные технологии: теория, инструменты, практика (InnoTech 2013). Материалы V Международной Интернет-конференции молодых ученых, аспирантов и студентов. – Пермь: ПНИПУ, 2014. – С. 190-196.

[15] http://www.sqlite.org – SQLite.

[16] http://felix.apache.org – Apache Felix.

[17] http://www.osgi.org/Main/HomePage – OSGi Alliance.

[18] *Касьянов В.Н., Идрисов Р.И., Касьянова Е.В., Стасенко А.П.* Методы и средства параллельного программирования на основе языка Sisal// Материалы XV Международной конференции “Информатика: проблемы, методология, технология”. – Воронеж: ВГУ, 2015. – Том 3. – С. 166-170.

[19] *Kasyanov V.N., Kasyanova E.V.* Cloud system of functional and parallel programming for computer science education// Proceedings of 2015 2nd International Conference on Creative Education (ICCE-2015), June 27-28, 2015, London, UK. – SMSSI, 2015. – pp. 270-275. –(Advances in Education Sciences, Vol. 10).