REPRESENTATION OF GRAPHS AND GRAPH MODELS: BASIC TOOLS OF THE GRAPHML LANGUAGE

V. N. Kasyanov, E. V. Kasyanova

A. P. Ershov Institute of Informatics Systems SD RAS,
630090, Novosibirsk, Russia
Novosibirsk State University,
630090, Novosibirsk, Russia

Graphs are used almost everywhere in computer science. Any system that consists of discrete states (or sites) and connections between them can be modeled by a graph. For a long time among different formats being used for presentation of graphs was not a single one that would be widely accepted as the standard format. As a rule tools to work with graph models supported different graph formats, usually consisting of restricted subclasses of graphs and their representations.

Motivated by the goals of tool interoperability, access to benchmark data sets, and data exchange over the Web, the Steering Committee of the Graph Drawing Symposium started a new initiative with an informal workshop held in conjunction with the 8th Symposium on Graph Drawing. As a consequence, an informal task group was formed to propose a modern graph exchange format suitable in particular for data transfer between graph drawing tools and other applications. The main goal of the GraphML language has been formulated by the developers in the following way. The graph exchange format should be able to represent arbitrary graphs with arbitrary additional data, including layout and graphics information. The additional data should be stored in a format appropriate for the specific application, but should not complicate or interfere with the representation of data from other applications.

GraphML is designed with this and the following more pragmatic goals in mind:

1) Simplicity: The format should be easy to parse and interpret for both humans and machines. As a general principle, there should be no ambiguities and thus a single well-defined interpretation for each valid GraphML document.

2) Generality: There should be no limitation with respect to the graph model, i.e. hypergraphs, hierarchical graphs, etc. should be expressible within the same basic format.

3) Extensibility: It should be possible to extend the format in a well-defined way to represent additional data required by arbitrary applications or more sophisticated use (e.g., sending a layout algorithm together with the graph).

4) Robustness: Systems not capable of handling the full range of graph models or added information should be able to easily recognize and extract the subset they can handle.

Designed GraphML language for descriptions of graphs is based on XML. It allows describing directed, undirected and mixed graphs, hyper graphs and hierarchical graphs, as well as any specific attributes for specific applications. In particular, GraphML language fully supports attributed hierarchical graphs [17]. Thanks to its XML syntax, GraphML can be used in combination with other XML based formats. On the one hand, its own extension mechanism allows attaching <data> labels with complex content (possibly required to comply with other XML content models) to GraphML elements. Examples of such complex data labels are Scalable Vector Graphics describing the appearance of the nodes and edges in a drawing. On the other hand, GraphML can be integrated into other applications, e. g. in SOAP messages.

In this paper, only the basic tools of the GraphML language are described, which are sufficient to represent graph models in most applications. It discusses how graphs and graph data can be represented in GraphML format using a basic graph model that covers graphs containing directed and undirected edges, loops, multiple edges, and various labels (attributes) of nodes, edges, and graphs.

The work has been partially supported by the Russian Foundation for Basic Research under grant N 15-07-02029.

Key words: graph, graph data, graph model, GraphML.

References

1. Kasyanov V. N., Evstigneev V. A. Grafy v Programmirovanii: Obrabotka, Vizualizatsiia i Primenenie [Graphs in Programming: Processing, Visualization and Application]. St. Petersburg: BHV Petersburg, 2003.

2. Kasyanov V. N., Kasyanova E. V. Vizualizatziia Grafov i Grafovuh Modelej [Visualization of Graphs and Graph Models] Novosibirsk: Siberian Scientific Publishing House, 2010.

3. Di Battista G., Eades P., Tamassia R., Tollis I. G. Graph Drawing: Algorithms for the Visualization of Graphs. Prentice Hall, 1999.

4. Borgatti S. P., Everett M. G., and Freeman L. C. UCINET 6.0 , Analytic Technologies, 1999.

5. De Nooy W., Mrvar A., and Batagelj V. Exploratory social network analysis with Pajek. Cambridge University Press, 2005.

6. Winter A. Exchanging Graphs with GXL, Lecture Notes in Comput. Sci. 2002. Vol. 2265. P. 485–500. (Proc. 9th Int. Symp. Graph Drawing GD'2001).

7. Tsvetovat M., Reminga J., and Carley K. Dynetml: interchange format for rich social network data, NAACSOS Conference. Pittsburgh, PA, 2003.

8. GML. The Graph Modeling Language File Format. [Electron. Res.]: `www.infosun.fmi.uni-passau.de/Graphlet/GML/`.

9. Brandes U., Marshall M. S., North S. C. Graph data format workshop report, Lecture Notes in Comput. Sci. 2001. Vol. 1984. P. 410–418. (Proc. 8th Int. Symp. Graph Drawing GD'2000).

10. Brandes U., Eiglsperger M., Herman I., et al. GraphML progress report: structural layer proposal, Lecture Notes in Comput. Sci. 2002. Vol. 2265. P. 501–512. (Proc. 9th Int. Symp. Graph Drawing GD'2001).

11. Brandes U., Eiglsperger M., Lerner J. GraphML Primer. [Electron. Res.]: `graphml.graphdrawing.org/primer/graphml-primer.html#EXT`.

12. Brandes U., Eiglsperger M., Lerner J., Pich C. Graph markup language (GraphML). [Electron. Res.]: `www.cs.brown.edu/~rt/gdhandbook/chapters/graphml.pdf`.

13. W3C. Scalable Vector Graphics. [Electron. Res.]: `www.w3.org/TR/SVG/`.

14. W3C. SOAP. [Electron. Res.]: `www.w3.org/TR/soap12-part0/`.

15. Kasyanov V. N. Yazyk predstavleniya grafov GraphML: dopolnitel'nye vozmozhnosti, Informatika v nauke i obrazovanii [The GraphML language for graph presentation: additional tools, Informatics in science and education]. Novosibirsk: IIS SB RAN, 2012. P. 7–22.

16. Evstigneev V. N., Kasyanov V. N. Tolkovyj slovar' po teorii grafov v informatike i programmirovanii [Explanatory Dictionary of Graph Theory in Computer Science and Programming]. Novosibirsk: Nauka Publ., 1999.

17. Kasyanov V. N., Zolotuhin T. A. Visual Graph — sistema dlya vizualizacii slozhno strukturirovannoj informacii bol'shogo ob"ema na osnove grafovyh modelej, Nauchnaia vizualizatsiia [Visual Graph – a system for visualization of big size complex structural information on the base of graph models, Scientific Visualization]. 2015. Vol. 7, N 4. P. 44–59.

18. Sugiyama K., Tagawa S., Toda M. Methods for visual understanding of hierarchical system structures, IEEE Transactions on Systems, Man and Cybernetics. 1981. Vol. 11. N 2. P. 109–125.

ПРЕДСТАВЛЕНИЕ ГРАФОВ И ГРАФОВЫХ МОДЕЛЕЙ: БАЗОВЫЕ СРЕДСТВА ЯЗЫКА GRAPHML

В. Н. Касьянов, Е. В. Касьянова

Институт систем информатики им. А. П. Ершова СО РАН,
630090, Новосибирск, Россия
Новосибирский государственный университет,
630090, Новосибирск, Россия

УДК 004

Статья посвящена международному проекту GraphML, инициированному сообществом по рисованию графов в 2000 г. с целью создания стандартизованного языка описания графов на основе языка XML, и содержит описание базовых средств языка GraphML, достаточных для представления графовых моделей в большинстве приложений. В ней рассматривается, как графы и графовые данные представляются в формате GraphML с использованием базовой графовой модели, которая охватывает графы, содержащие ориентированные и неориентированные ребра, петли, кратные ребра и различные пометки (атрибуты) вершин, ребер и частей графа.

Ключевые слова: граф, графовые данные, графовая модель, GraphML.

Введение. Современное программирование нельзя представить себе без теоретико-графовых методов и алгоритмов [1]. Широкая применимость графов связана с тем, что они являются очень естественным средством объяснения сложных ситуаций на интуитивном уровне. Эти преимущества представления сложных структур и процессов графами становятся еще более ощутимыми при наличии хороших средств их визуализации [2, 3].

Ясно, что инструменты визуализации информации на основе графовых моделей, подобно всем другим инструментам, имеющим дело со структурированными данными, нуждаются в сохранении и передаче графов и ассоциированных с ними данных. Среди многочисленных форматов файлов для представлений графов применяются основанные на ASCII-кодах таблицы (матрицы) или списки, такие как табулированные файлы, к которым относятся *.dl файлы UCINET [4] и *.net файлы Паека (Pajek) [5]. Есть среди них и основанные на XML форматы для представления графов, такие как GXL [6] и DyNetML [7]. Еще один используемый формат представления графов — это язык GML (Graph Modelling Language) [8], работа над которым началась в 1995 г. на 4-м симпозиуме по рисованию графов GD-95 в г. Пассау и завершилась в 1996 г. на 5-м симпозиуме по рисованию графов GD-96 в г. Беркли. Язык GML до сих остается основным файловым форматом для системы Graphlet, а также поддерживается рядом других систем обработки графов.

Однако долгое время среди используемых форматов представления графов не находилось ни одного, который был бы достаточно широко принятым в качестве стандартного;

Работа выполнена при частичной финансовой поддержке Российского фонда фундаментальных исследований (грант РФФИ № 18-07-00024).

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 неслучайно в 2000 году наблюдательный комитет симпозиума по рисованию графов (Graph Drawing Steering Committee) организовал рабочее совещание по форматам обмена графовыми данными, состоявшееся в г. Вильямсбурге в рамках 8-го симпозиума по рисованию графов (GD-2000) [9]. Как следствие была сформирована неформальная рабочая группа по выработке основанного на языке XML формата обмена графами GraphML, который, в частности, был бы пригоден для обмена данными между инструментами рисования графов и другими приложениями и в конечном счете лег бы в основу стандарта описания графов.

Рабочая группа по созданию языка GraphML объединяет десятки специалистов из разных организаций и стран, и ее работу наравне с другими координируют Ulrik Brandes (университет г. Констанц, Германия), Markus Eiglsperger (Тюбингенский университет, Германия), Michael Kaufmann (Тюбингенский университет, Германия), Jürgen Lerner (университет г. Констанц, Германия) и Christian Pich (университет г. Констанц, Германия).

Первый отчет по языку вышел в 2001 году [10]. С тех пор язык был расширен в части поддержки основных типов атрибутов и в части включения информации для использования синтаксическими анализаторами [11, 12]. Ведется работа по включению абстрактной информации для описания топологии графа и шаблонов, с помощью которых эту информацию можно преобразовать в различные графические форматы. Программное обеспечение для поддержки работы с GraphML также находится в постоянной разработке.

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 элементов GraphML, возможно, требуемым для исполнения с другими моделями XML содержимого. Примером использования таких меток со сложным содержимым является так называемый механизм SVG (Scalable Vector Graphics) [13], описывающий появление вершин и дуг в изображении. С другой стороны, GraphML может интегрироваться в другие приложения, например, в SOAP сообщения [14].

В данной статье рассматриваются только базовые средства языка GraphML, достаточные для представления графовых моделей в большинстве приложений. В ней описывается, как графы и простые графовые данные представляются в формате GraphML с использованием базовой графовой модели языка. Базовая графовая модель охватывает графы[1], которые могут содержать ребра, дуги, петли, кратные дуги, кратные ребра и пометки (атрибуты) вершин, ребер и частей графа. Помимо рассмотренных здесь, язык GraphML содержит и другие средства, существенно обогащающие его возможности [9–12, 15]. В частности, GraphML поддерживает графовую модель, которая расширяет базовую за счет введения таких дополнительных понятий для графовой топологии, как вложенные графы, гиперграфы и порты, и может, например, использоваться для адекватного представления иерархических графовых моделей [17]. Язык позволяет также осуществлять различные расширения за счет добавления новых атрибутов к GraphML-элементам и пу-

[1]Здесь и ниже мы без определения используем стандартные понятия из теории графов (см., например, [16]).

тем расширения содержимого элементов <data>, а также использовать XSLT-механизм для преобразования GraphML-документов.

1. Цели разработки и используемая базовая графовая модель. Разработчики языка убеждены, что современный формат обмена графами не может быть монолитным, поскольку, как правило, используются сервисы рисования (визуализации) графов в качестве компонентов более больших систем, а также возникают различные сетевые сервисы. Постоянно возникает потребность обмена графовыми данными между этими сервисами или отдельными их этапами, а также между сервисами рисования графов и другими системами, специфическими для различных областей приложений.

Типичные пользовательские сценарии, которые предусматривались авторами для разработанного формата, собраны вокруг систем, спроектированных для произвольных приложений, имеющих дело с графами и различными данными, ассоциированными с ними. Такие системы могут содержать или вызывать сервисы рисования графов, которые добавляют или изменяют раскладку или графическую информацию. Такие сервисы могут вычислять только частичную информацию или промежуточные представления, поскольку они воплощают лишь часть многофазового подхода к укладке, такого как метрики топологических форм (the topology-shape-metrics) или схемы Сугиямы (Sugiyama frameworks) [5, 18].

Основную цель разработчики языка формулируют 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.

Язык GraphML проектировался с ориентацией на эту цель, а также с учетом следующих более прагматических требований.

— Простота (Simplicity). Формат должен был прост 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.

— 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.

— Расширяемость (Extensibility). Должна существовать возможность расширять формат хорошо определенным способом для представления дополнительных данных, требуемых произвольными приложениями или более сложным использованием (например, посылая алгоритм раскладки вместе с графом).

— 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.

Под графовой моделью, описываемой с помощью базовых средств языка, понимается помеченный смешанный мультиграф

$$G = (V, E, L),$$

который состоит из множества вершин V, множества ребер E (как неориентированных, так и ориентированных), соединяющих пары необязательно различных вершин, и мно-

жества разметок L, каждая из которых является частичной функцией, сопоставляющей элементам графа $\{G\} \cup V \cup E$ элементы некоторого заданного множества пометок.

Таким образом, рассматриваемая базовая графовая модель включает графы, которые могут содержать ребра, дуги, петли, кратные дуги и кратные ребра. Множества пометок могут кодировать, например, различные семантические свойства объектов, представленных в виде данной графовой модели, или различные геометрические свойства элементов графа в заданном его изображении на плоскости.

2. Вид документа. В качестве примера представления рассмотрим на рис. 1 фрагмент graphml-документа, который представляет простой непомеченный граф, изображенный на рис. 2.

Следует отметить, что указанный фрагмент GraphML-документа как XML-документ не валиден, поскольку каждый валидный XML-документ должен декларировать в своем заголовке, является ли он DTD (document type definition) или XML-схемой. По-существу, заданное множество определений DTD и XML-схем определяет подмножество всех тех XML-документов, которые и формируют некоторый конкретный язык. Но язык GraphML был определен с помощью лишь некоторой схемы. И хотя DTD-определение предназначено для поддержки парсеров, которые не могут обработать схемные определения, единственной нормативной спецификацией языка GraphML является GraphML-схема, размещенная по адресу `http://graphml.graphdrawing.org/xmlns/1.1/graphml.xsd`.

Документ, представленный на рис. 3, является минимальным GraphML-документом, который валиден по данной схеме. Понятно, что данный документ определяет пустое множество графов. Части документа, начинающиеся с символов <!– и завершающиеся символами –>, являются комментариями.

Первая строка документа — это инструкция обработки, которая определяет, что документ является подмножеством стандарта XML 1.0, и что документ выполнен в кодировке UTF-8, являющейся стандартом для XML-документов. Конечно, для GraphML-документов могут быть выбраны и другие кодировки.

Вторая строка содержит корневой элемент GraphML-документа — graphml, который, как и все остальные элементы языка GraphML, принадлежит пространству имен `http://graphml.graphdrawing.org/xmlns`.

По этой причине с помощью XML-атрибута xmlns=http://graphml.graphdrawing.org/xmlns именно это пространство имен в нашем документе и определено в качестве пространства имен данного документа, заданного по умолчанию. Следующие два XML-атрибута определяют XML-схему, которая используется для валидации данного документа. В нашем примере используется стандартная схема GraphML-документа, расположенная на сервере `graphdrawing.org`. Первый атрибут,

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance",
```

определяет xsi в качестве префикса пространства имен XML-схемы. Второй атрибут,

```
xsi: schemaLocation="http://graphml.graphdrawing.org/xmlns
http://graphml.graphdrawing.org/xmlns/1.1/graphml.xsd",
```

определяет местонахождение XML-схемы для элементов пространства имен GraphML. Он предоставляет информацию о том, что все элементы в пространстве имен GraphML являются валидными по отношению к файлу graphml.xsd, расположенному по указанному адресу. Конечно, валидация не должна обязательно выполняться с использованием данного файла. Локальные копии graphml.xsd также могут специфицироваться в качестве

```
<graphml>
 <graph edgedefault="directed">
 <node id="v1"/>
 <node id="v2"/>
 <node id="v3"/>
 <node id="v4"/>
 <edge source="v1" target="v2"/>
 <edge source="v1" target="v1" directed="false"/>
 <edge source="v1" target="v3"/>
 <edge source="v2" target="v4"/>
 <edge source="v2" target="v4" directed="false"/>
 <edge source="v3" target="v4"/>
 <edge source="v3" target="v4"/>
 <edge source="v4" target="v4"/>
 </graph>
</graphml>
```

Рис. 1. Фрагмент GraphML-документа для представления простого графа, изображенного на рис. 2

местонахождений XML-схем. Заметим, что обычно значение атрибута schemaLocation является списком пар, в которых первый элемент обозначает некоторое пространство имен, а второй указывает на файл, в котором элементы этого пространства определены.

Ссылка на XML-схему необязательна, но она обеспечивает механизм для синтаксической проверки документа и поэтому строго рекомендуется. Минимальный GraphML-документ без ссылки на схему приведен на рис. 4. Заметим, что этот файл не является валидным документом в соответствии с XML-спецификацией.

3. Топология графа. Опишем, как топология графа (его элементы) представляется на языке GraphML. Вначале еще раз рассмотрим документ, приведенный на рис. 1. Отдельный граф (часть графа) представлен на языке GraphML посредством элемента <graph>. Элемент <graphml> может содержать произвольное число элементов <graph>.

Каждый граф в GraphML может являться смешанным, другими словами, он может содержать одновременно как ориентированные, так и неориентированные ребра. Если при объявлении ребра его ориентированность не определена, то применяется ориентированность, заданная для ребер графа по умолчанию. Ориентированность ребер графа, присваиваемая по умолчанию, задается с помощью XML-атрибута edgedefault элемента <graph>. Данный XML-атрибут может принимать одно из двух значений: directed (ориентированный) и undirected (неориентированный). Значение по умолчанию должно быть задано обязательно. Дополнительно с помощью атрибута id графу может быть присвоен некоторый идентификатор. Этот идентификатор нужен тогда, когда на данный граф требуется организовать ссылку.

Вершины графа представляются в виде списка элементов <node>. Каждая вершина имеет уникальный в пределах данного документа идентификатор, который задается с помощью атрибута id.

Множество ребер представляется в виде списка элементов <edge>. Каждое ребро имеет две инцидентные вершины, задаваемые с помощью XML-атрибутов source и target.

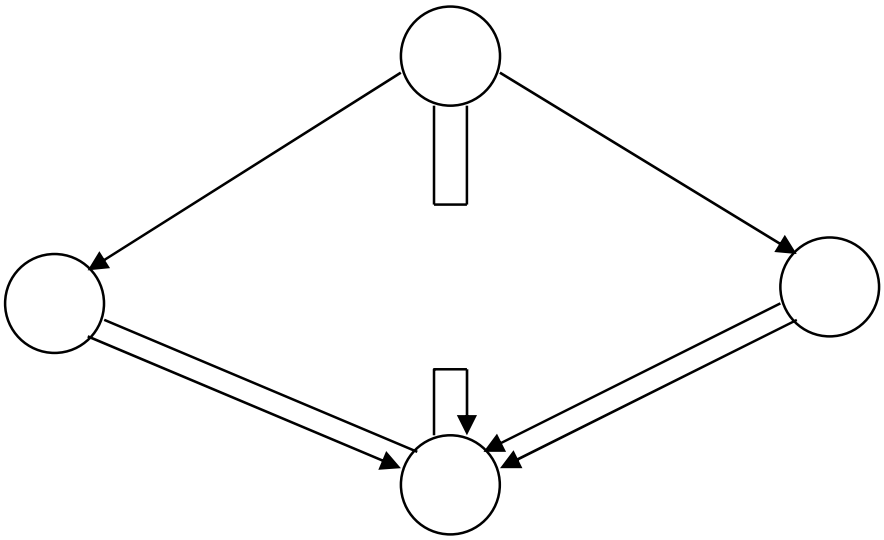

Рис. 2. Пример простого графа

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns=http://graphml.graphdrawing.org/xmlns
 xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
 xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
 http://graphml.graphdrawing.org/xmlns/1.1/graphml.xsd">
 <!--Content: List of graphs and data-->
</graphml>
```

Рис. 3. Минимальный валидный GraphML-документ

Значения атрибутов source и target должны содержать идентификаторы вершин, определенных в том же документе, что и ребро. Ребра с одной инцидентной вершиной, так называемые петли, определяются с помощью одинаковых значений, заданных в атрибутах source и target. Дополнительный XML-атрибут directed определяет ориентированность ребра, заданную в явном виде. Значение true задает ориентированное ребро (дугу), a false — неориентированное. Если ориентированность в явном виде не задана, то применяется ориентированность, заданная по умолчанию при объявлении графа. Дополнительно с помощью XML-атрибута id может быть задан идентификатор ребра. XML-атрибут id задается, когда необходимо организовать ссылку на данное ребро.

Вершины и ребра упорядочиваются произвольным образом, и язык не требует перечислять все вершины до перечисления всех ребер. Ясно, что память, требуемая для сохранения на языке GraphML графа с n вершинами и m ребрами, составляет $O(n + m)$.

4. Атрибуты. В предыдущем разделе мы обсудили порядок описания топологии графа на языке GraphML. Хотя имеется целый ряд приложений, для которых информации о топологии графов может быть достаточно, большинство приложений работает с графовыми моделями, обладающими дополнительной информацией. Поэтому в языке GraphML предусмотрены средства для включения различной информации в описание графа.

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns=http://graphml.graphdrawing.org/xmlns
 xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
 xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
 http://graphml.graphdrawing.org/xmlns/1.1/graphml.xsd">
 <!--Content: List of graphs and data-->
</graphml>
```

Рис. 4. Минимальный GraphML-документ без ссылки на схему

С помощью механизма расширения, который называется *GraphML-атрибуты,* для элементов графа может быть задана дополнительная информация простого типа. Простой тип подразумевает, что данные ограничены скалярными величинами, например, числами и строками. Расширение GraphML-атрибуты уже включено в файл

http://graphml.graphdrawing.org/xmlns/1.1/graphml.xsd,

и таким образом заголовок графа со скалярными атрибутами может иметь вид, рассмотренный в разд. 2.

GraphML-атрибуты не следует путать с XML-атрибутами, которые имеют совсем другой смысл. GraphML-атрибуты добавляют информацию к графам, множествам графов или частям графов, а XML-атрибуты добавляют информацию к XML-элементам.

В большинстве случаев дополнительная информация может и должна прикрепляться к элементам GraphML с помощью механизма GraphML-атрибутов, описанного здесь. Это гарантирует читаемость описаний графов другими GraphML-парсерами. Если же необходим более сложный формат данных в качестве атрибутов, можно воспользоваться механизмом расширения языка GraphML путем добавления новых атрибутов к GraphML-элементам или путем расширения содержимого элементов <data> (см, например, [15]).

GraphML-атрибуты рассматриваются как частичные функции, приписывающие элементам графа значения атрибутов, которые, как правило, имеют один и тот же тип. Например, веса ребер могут рассматриваться как функция из множества ребер E в множество вещественных чисел R:

$$\text{weight} : E \to R.$$

Другой пример — это формы изображения вершин, которые можно представить в виде функции из множества вершин V в множество слов над заданным алфавитом Σ:

$$\text{shape} : V \to \Sigma^*.$$

Для добавления указанных функций к элементам графа следует использовать key/data-механизм языка GraphML. Элемент <key>, размещаемый в начале документа, декларирует новую функцию разметки; более точно элемент <key> специфицирует для функции ее идентификатор, имя, области определения и значения. Сами же значения функции определяются с помощью <data> элементов.

Декларация всех функций разметки в самом начале документа позволяет парсерам построить подходящие структуры данных в начале процесса разбора. Также парсеры могут

распознавать ситуации, когда необходимые данные опущены. На рис. 5 приведен пример применения key/data-механизма. Здесь функция веса описывается строкой

```
<key id="d1" for="edge" attr.name="weight" attr.type="double"/>.
```

Элемент <key> имеет XML-атрибут с именем for, который специфицирует область определения функции. Для атрибута for в качестве значений можно указать graph, node, edge, graphml, а также имена других типов элементов графа, рассмотренные в разд. 2. Данный XML-атрибут может также получить значение all, что означает, что данные пометки могут помечать любые элементы графа. Атрибут for вместе с уникальным атрибутом id являются обязательными для элементов <key>. Расширение GraphML предоставляет еще два атрибута для <key>: это атрибут attr.name, который определяет имя функции и используется парсером для распознавания соответствующих данных, а также атрибут attr.type, который задает область значения функции и может принимать в качестве значений имена типов[2] boolean, int, long, float, double, или string.

Обычный парсер, обрабатывая веса ребер, как правило, после обработки описания функции веса инициализирует внутреннюю структуру данных, которая сохраняет двойное вещественное с каждым ребром. В отличие от него другой парсер, который не знает о функции пометок ребер весами или не нуждается в этой функции, будет просто игнорировать ассоциированные <data> элементы.

Значения функции данных на некотором элементе графа (или, что то же самое, значение GraphML-атрибута у данного элемента графа) задается с помощью элемента <data>, вложенного в описание данного элемента. Например, фрагмент документа

```
<edge id="e0" source="n0" target="n2">
<data key="d1">1.0</data>
</edge>
```

определяет значение 1.0 в качестве веса для заданной дуги <edge>. Элемент data имеет XML-атрибут <key>, который ссылается на идентификатор GraphML-атрибута. Значение GraphML-атрибута задается текстовым содержимым элемента <data>. Это значение должно иметь тип, объявленный в соответствующем элементе <key>.

Для GraphML-атрибутов можно определить значение по умолчанию. Содержимое элемента default определяет текстовое значение по умолчанию. Например, фрагмент документа

```
<key id="d0" for="node" attr.name="shape" attr.type="string">
<default>circle </default>
</key>
```

определяет значение circle в качестве значения по умолчанию для атрибута форма у вершин графа.

Могут быть такие GraphML-атрибуты, которые определены, но не объявлены с помощью элемента <data>. Если значение по умолчанию определено для данного GraphML-атрибута, то тогда это значение применяется к соответствующему (входящему в домен GraphML-атрибута) элементу графа. В вышеприведенном примере значение не определено для вершины с идентификатором n1 и GraphML-атрибута с именем shape. Однако для данного GraphML-атрибута определено значение по умолчанию circle, которое будет

[2]Эти типы определены в соответствии с аналогичными типами в языке Java.

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml>
<key id="d0" for="node" attr.name="shape" attr.type="string">
<default>circle </default>
</key>
<key id="d1" for="edge" attr.name="weight" attr.type="double"/>
<graph id="G" edgedefault="undirected">
<node id="n0">
<data key="d0">square</data>
</node>
<node id="n1"/>
<node id="n2">
<data key="d0">oval</data>
</node>
<node id="n3">
<data key="d0">square</data>
</node>
<node id="n4"/>
<node id="n5">
<data key="d0">oval</data>
</node>
<node id="n6"/>
<edge id="e0" source="n0" target="n2">
<data key="d1">1.0</data>
</edge>
<edge id="e1" source="n0" target="n1">
<data key="d1">1.0</data>
</edge>
<edge id="e2" source="n1" target="n3">
<data key="d1">2.0</data>
</edge>
<edge id="e3" source="n3" target="n2"/>
<edge id="e4" source="n2" target="n4"/>
<edge id="e5" source="n3" target="n5"/>
<edge id="e6" source="n5" target="n4">
<data key="d1">1.1</data>
</edge>
</graph>
</graphml>
```

Рис. 5. Представление атрибутированного графа, в котором ребра имеют веса, а вершины форму

присвоено данной вершине. Если же значение по умолчанию не задано, как это имеет место для GraphML-атрибута weight в вышеприведенном примере, то значение GraphML-атрибута для такого элемента графа считается неопределенным. В вышеприведенном при-

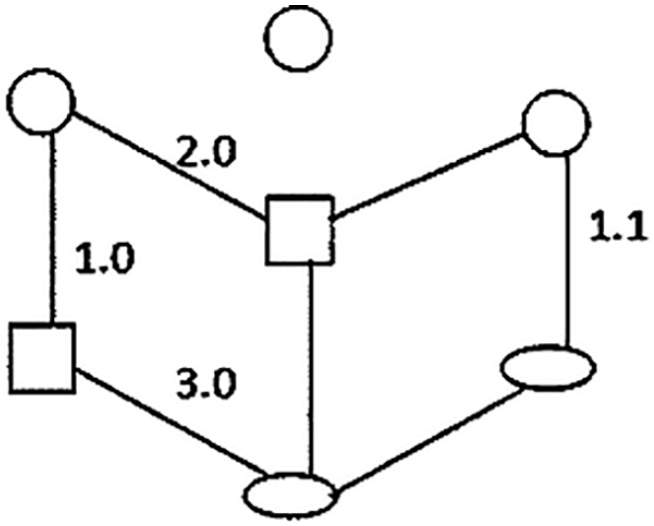

Рис. 6. Изображение атрибутированного графа, представленного в виде GraphML-документа на рис. 5

мере не определено значение GraphML-атрибута, задающего вес, у ребра с идентификатором e3.

5. Информация для парсера. Помимо возможности задания атрибутов, к базовым средствам языка относят еще одно расширение формата описания графа, называемое GraphML-Parseinfo. GraphML-Parseinfo делает возможным писать простые парсеры, основанные на дополнительной информации в GraphML-файлах. Это расширение уже включено в файл:

http://graphml.graphdrawing.org/xmlns/1.1/graphml.xsd.

Таким образом, заголовок файла с описанием графа, использующим это расширение, может иметь тот же вид, что и рассмотренный ранее (в разд. 2).

Указанное расширение направлено на оптимизацию синтаксического разбора документа с помощью парсера за счет использования специальных метаданных, которые могут быть добавлены к некоторым GraphML-элементам с помощью XML-атрибутов. Имеется два вида метаданных: информация о количестве элементов и информация о способе кодирования конкретных данных в документе. Например, для парсера, который сохраняет вершины и инцидентные ребра в форме массивов, может оказаться весьма полезной информация о количествах вершин и ребер в графе, а также степенях его вершин. Все XML-атрибуты, задающие указанные метаданные, имеют префикс parse.

Для первого вида метаданных, связанного с информацией о количестве элементов, определены следующие XML-атрибуты для элемента <graph>: XML-атрибут parse.nodes задает количество вершин в графе, XML-атрибут parse.edges определяет количество ребер в графе, XML-атрибут parse.maxindegree определяет максимальное количество ребер, входящих в одну вершину графа, а XML-атрибут parse.maxoutdegree определяет максимальное количество ребер, исходящих из одной вершины графа.

Кроме того, для элемента <node> введены новые XML-атрибуты parse.indegree и parse.outdegree, которые определяют для данной вершины количества входящих и исходящих ребер.

Для метаданных, связанных со способом кодирования, определены XML-атрибуты parse.nodeids, parse.edgeids и parse.order для элемента <graph> со следующей семанти-

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml>
<graph id="G" edgedefault="directed"
        parse.nodes="11"
        parse.edges="13"
        parse.maxindegree="2"
        parse.maxoutdegree="3"
        parse.nodeids="canonical"
        parse.edgeids="free"
        parse.order="nodesfirst">
    <node  id="n0"   parse.indegree="0"  parse.outdegree="1"/>
    <node  id="n1"   parse.indegree="1"  parse.outdegree="1"/>
    <node  id="n2"   parse.indegree="2"  parse.outdegree="1"/>
    <node  id="n3"   parse.indegree="1"  parse.outdegree="2"/>
    <node  id="n4"   parse.indegree="1"  parse.outdegree="1"/>
    <node  id="n5"   parse.indegree="2"  parse.outdegree="1"/>
    <node  id="n6"   parse.indegree="1"  parse.outdegree="2"/>
    <node  id="n7"   parse.indegree="2"  parse.outdegree="0"/>
    <node  id="n8"   parse.indegree="1"  parse.outdegree="3"/>
    <node  id="n9"   parse.indegree="1"  parse.outdegree="0"/>
    <node  id="n10"  parse.indegree="1"  parse.outdegree="1"/>
  <edge  id="edge0001"  source="n0"  target="n2"/>
  <edge  id="edge0002"  source="n1"  target="n2"/>
  <edge  id="edge0003"  source="n2"  target="n3"/>
  <edge  id="edge0004"  source="n3"  target="n5"/>
  <edge  id="edge0005"  source="n3"  target="n4"/>
  <edge  id="edge0006"  source="n4"  target="n6"/>
  <edge  id="edge0007"  source="n6"  target="n5"/>
  <edge  id="edge0008"  source="n5"  target="n7"/>
  <edge  id="edge0009"  source="n6"  target="n8"/>
  <edge  id="edge0010"  source="n8"  target="n7"/>
  <edge  id="edge0011"  source="n8"  target="n9"/>
  <edge  id="edge0012"  source="n8"  target="n10"/>
  <edge  id="edge0013"  source="n10"  target="n1"/>
</graph>
</graphml>
```

Рис. 7. Использование GraphML-Parseinfo метаданных

кой. Если XML-атрибут parse.nodeids имеет значение canonical, все вершины получают идентификатор вида nX, где X обозначает количество элементов <node>, предшествующих данному элементу. Другое возможное значение данного XML-атрибута равно free. Аналогичным образом действует XML-атрибут parse.edgeids, который задает вид идентификатора для ребер. Отличие состоит только в том, что этот идентификатор имеет вид eX. XML-атрибут parse.order определяет порядок, в котором вершины и ребра располагаются в документе. При значении, равном nodesfirst, все элементы <node> располагаются

раньше элементов <edge>. При значении, равном adjacencylist, объявление вершины предшествует объявлению смежных ей вершин. Для значения free порядок следования вершин и ребер никак не ограничивается.

Пример на рис. 7 иллюстрирует использование указанного вида информации для парсера.

Список литературы

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

2. Касьянов В. Н., Касьянова Е. В. Визуализация графов и графовых моделей. Новосибирск: Сибирское Научное Издательство, 2010.

3. Di Battista G., Eades P., Tamassia R., Tollis I. G. Graph Drawing: Algorithms for the Visualization of Graphs. Prentice Hall, 1999.

4. Borgatti S. P., Everett M. G., and Freeman L. C. UCINET 6.0 // Analytic Technologies, 1999.

5. De Nooy W., Mrvar A., and Batagelj V. Exploratory social network analysis with Pajek. Cambridge University Press, 2005.

6. Winter A. Exchanging Graphs with GXL // Lecture Notes in Comput. Sci. 2002. Vol. 2265. P. 485–500. (Proc. 9th Int. Symp. Graph Drawing GD'2001).

7. W3C. Scalable Vector Graphics. [Electron. Res.]: `www.w3.org/TR/SVG/`.

8. GML. The Graph Modeling Language File Format. [Electron. Res.]: `www.infosun.fmi.uni-passau.de/Graphlet/GML/`.

9. Brandes U., Marshall M. S., North S. C. Graph data format workshop report // Lecture Notes in Comput. Sci. 2001. Vol. 1984. P. 410–418. (Proc. 8th Int. Symp. Graph Drawing GD'2000).

10. Brandes U., Eiglsperger M., Herman I., et al. GraphML progress report: structural layer proposal // Lecture Notes in Comput. Sci. 2002. Vol. 2265. P. 501–512. (Proc. 9th Int. Symp. Graph Drawing GD'2001).

11. Brandes U., Eiglsperger M., Lerner J. GraphML Primer. `http://graphml.graphdrawing.org/primer/graphml-primer.html#EXT`.

12. Brandes U., Eiglsperger M., Lerner J., Pich C. Graph markup language (GraphML) `http://www.cs.brown.edu/~rt/gdhandbook/chapters/graphml.pdf`.

13. W3C. Scalable Vector Graphics. `http://www.w3.org/TR/SVG/`.

14. W3C. SOAP. `http://www.w3.org/TR/soap12-part0/`.

15. Касьянов В. Н. Язык представления графов GraphML: дополнительные возможности // Информатика в науке и образовании. Новосибирск: ИСИ СО РАН, 2012. С. 7–22.

16. Евстигнеев В. А., Касьянов В. Н. Толковый словарь по теории графов в информатике и программировании. Новосибирск: Наука, 1999.

17. Касьянов В. Н., Золотухин Т. А. Visual Graph — система для визуализации сложно структурированной информации большого объема на основе графовых моделей // Научная визуализация. 2015. Т. 7. № 4. С. 44–59.

18. Sugiyama K., Tagawa S., Toda M. Methods for visual understanding of hierarchical system structures // IEEE Transactions on Systems, Man and Cybernetics. 1981. Vol. 11. N 2. P. 109–125.

Касьянов Виктор Николаевич — доктор физ.-мат. наук, профессор, главный научный сотрудник и заведующий лабораторией Института систем информатики им. А. П. Ершова СО РАН, профессор Новосибирского государственного университета, e-mail: `kvn@iis.nsk.su`.

Касьянов Виктор Николаевич окончил Новосибирский госуниверситет по специальности „Математика“ в 1971 г. Защитил диссертацию кандидата физико-математических наук в 1976 г. в Совете по физико-математическим и

техническим наукам СО АН СССР. Получил степень доктора физико-математических наук в 1989 г. по решению ВАК при Совете Министров СССР и звание профессора в 1992 г. по решению Комитета по высшей школе Министерства науки, высшей школы и технической политики РФ. Член-корреспондент РАЕН (1994), член Американского математического общества (AMS, 1989), Европейской ассоциации по теоретической информатике (EATCS, 1991), Общества по индустриальной и прикладной математике (SIAM, 1996) и Сибирского математического общества (2009). Его научные интересы включают теоретические вопросы информатики, системное программирование, теорию графов в информатике, алгоритмы и сложность, высокопроизводительные вычисления, анализ и оптимизацию программ, человеко-машинный интерфейс и сетевую обработку. Автор и соавтор более 360 научных публикаций, в том числе 22 учебных пособий и 14 монографий: „Методы построения трансляторов“ (1986), „Оптимизирующие преобразования программ“ (1988), „Графы в программировании: обработка, визуализация и применение“(2003) и др.

Kasyanov Victor Nikolaevich — Dr. Sc., Full Professor, Chief Researcher and Head of Laboratory at the A.P. Ershov Institute of Informatics Systems, Professor at the Novosibirsk State University, e-mail: `kvn@iis.nsk.su`.

Kasyanov Victor Nikolaevich graduated from the Novosibirsk State University with a degree in Mathematics in 1971. He received Degree of Candidate of Physical and Mathematical Sciences (Ph. D.) in 1976 from the Council for Physical, Mathematical and Technical Sciences of the SB RAS of the USSR, Degree of Doctor of Science (Doctor Habilitate) in 1989 from the Highest Certification Committee at the Council Ministers of the USSR, and Academic Title of Full Professor in 1992 from the Ministry on Higher Education and Technical Politics of Russia. He is Corresponding Member of the Russian Academy of Natural Sciences (1994) and Member of the American Mathematical Society (AMS, 1989), European Association for Theoretical Computer Science (EATCS, 1991), Society for Industrial and Applied Mathematics (SIAM, 1996) and Siberian Mathematical Society (SMS, 2009). His research interests include formal aspects of computer science, software engineering, graph theory in computer science, algorithms and complexity, high performance computing, program analysis and optimization, man-machine interfaces and networking. He published over 360 scientific works including 22 textbooks and 14 monographs: „Methods of compiler construction“ (1986), „ Optimizing transformations of programs“ (1988), „Graphs in Programming: Processing, Visualization and Application“ (2003) and others.

Касьянова Елена Викторовна — кандидат физ.-мат. наук, доцент, старший научный сотрудник Института систем информатики им. А. П. Ершова СО РАН, доцент Новосибирского государственного университета, e-mail: `kev@iis.nsk.su`.

Касьянова Елена Викторовна окончила Новосибирский госуниверситет по специальности „Математика, Прикладная математика“ в 1993 г. Получила степень кандидата физико-математических наук в 2007 г. по решению ВАК Министерства образования и науки РФ и звание доцента в 2010 г. по решению Федеральной службы по надзору в сфере образования и науки. Ее научные интересы включают языки и системы программирования, адаптивную гипермедиа и системы учебной информатики, теорию графов в информатике, человеко-машинный интерфейс и базы данных, визуализацию информации. Является автором и соавтором 78 научных публикаций, в том числе 5 учебных пособий и двух монографий: „Адаптивные методы и средства поддержки дистанционного обучения программированию“ (2007) и „Визуализация графов и графовых моделей“(2010).

Kasyanova Elena Victorovna — Ph. D., Associate Professor, Senior Researcher at the A. P. Ershov Institute of Informatics Systems, Associate Professor at the Novosibirsk State University, e-mail: `kev@iis.nsk.su`.

Kasyanova Elena Victorovna graduated from the Novosibirsk State University with a degree in Mathematics and Applied Mathematics

in 1993. She received Degree of Candidate of Physical and Mathematical Sciences (Ph. D.) in 2007 from the Highest Certification Committee at the Ministry of Education and Science of the Russian Federation, and Academic Title of Associate Professor in 2010 from the Federal l Service for Supervision in Education and Science. Her scientific interests include programming languages and systems, adaptive hypermedia and educational informatics systems, graph theory in computer science, human-machine interface and databases, information visualization. She published 78 scientific works including 5 textbooks and two monographs: „ Adaptive Methods and Tools for Support Distance Education in Programming“ (2007) and „Visualization of Graphs and Graph Models“ (2010).

Дата поступления — 28.01.2018