BIOINFORMATICS APPLICATIONS NOTE Vol. 28 no. 12 2012, pages 1665–1667
doi:10.1093/bioinformatics/bts258

Databases and ontologies Advance Access publication May 3, 2012

graph2tab, a library to convert experimental workflow graphs into tabular formats

Marco Brandizi[1,*], Natalja Kurbatova[1], Ugis Sarkans[1] and Philippe Rocca-Serra[2]

[1]European Bioinformatics Institute, Wellcome Trust Genome Campus, Cambridge, CB10 1SD and
[2]Oxford e-Research Centre, University of Oxford, Oxford OX1 3QG, UK

ABSTRACT

Motivations: Spreadsheet-like tabular formats are ever more popular in the biomedical field as a mean for experimental reporting. The problem of converting the graph of an experimental workflow into a table-based representation occurs in many such formats and is not easy to solve.

Results: We describe graph2tab, a library that implements methods to realise such a conversion in a size-optimised way. Our solution is generic and can be adapted to specific cases of data exporters or data converters that need to be implemented.

Availability and Implementation: The library source code and documentation are available at http://github.com/ISA-tools/graph2tab.

Contact: brandizi@ebi.ac.uk.

Supplementary Information: A supplementary document describes the theoretical and technical details about the library implementation.

Received on February 24, 2012; revised on April 10, 2012; accepted on April 24, 2012

1 INTRODUCTION

Spreadsheet-like tabular formats are popular data exchange means in the biomedical field (Kuchinke *et al.*, 2009; Rayner *et al.*, 2006, 2009; Sansone *et al.*, 2008). The idea of representing complex structures, such as experimental designs, by means of matrices makes these formats easy to understand by end users, who can use familiar spreadsheet software products or specific applications (Rocca-Serra *et al.*, 2010, Shankar *et al.*, 2010) to manage experiment reporting and other biomedical data (Field *et al.*, 2010). While working with such formats is relatively easy for software developers too, it is highly desirable to build and share reusable software components for processing them. In this article we describe graph2tab, a (Java-based) programming library that allows a developer to deal with the specific (yet recurrent) problem of converting the workflow graph of a biomedical experiment into a tabular representation, defined according to a given tabular format, such as ISA-Tab (Sansone *et al.*, 2008) or MAGE-TAB (Rayner *et al.*, 2006). While the basics of such a conversion procedure are well known (http://www.mged.org/mage-tab/spec1.0.html; Rayner *et al.*, 2006) our experience shows that implementing all the details is not trivial. Considering this, we have developed the graph2tab library in a generic way, so that it can be adapted to a variety of concrete cases. A software developer can leverage on it to build tools like data exporters or data format converters. This can be done by adapting the library to a particular combination of an input object model and an output tabular format. In the following we describe the specific problems related to such conversion and the solutions that graph2tab offers to them.

1.1 Covering a DAG with a minimum path set

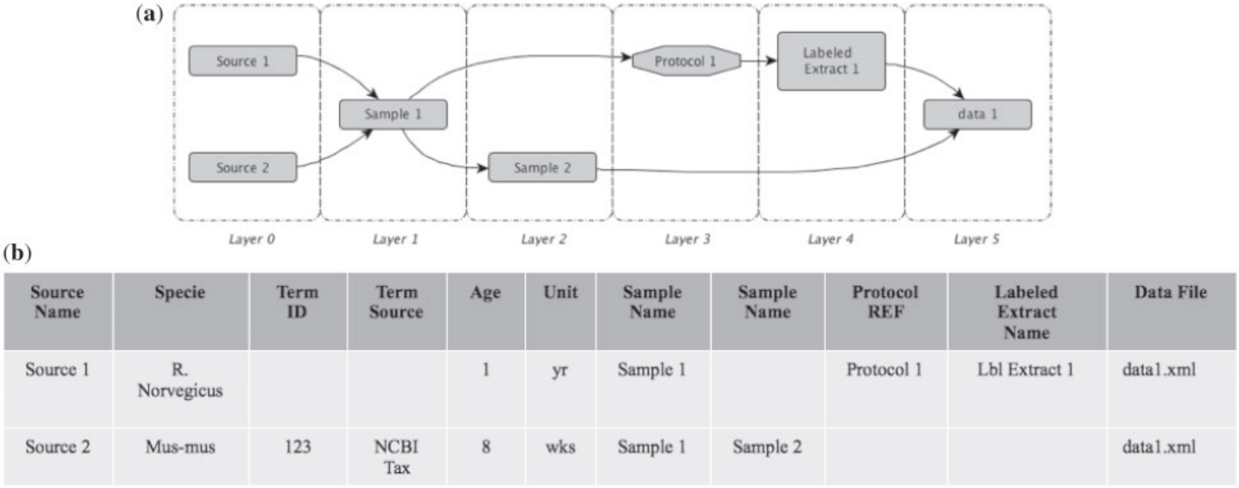
The input to the problem we deal with is essentially a directed acyclic graph, or DAG (Fig. 1a), which represents the execution workflow of a biomedical experiment, such as a microarray study or a clinical trial. The nodes of such graph can be about a biological product, such as a tissue sample, experiment's output, such as a file of gene-expression levels, or an experimental step, such as a reference to a laboratory protocol. Moreover, usually sets of typed attributes (i.e. header/value pairs), like the organism name a sample is about, are attached to the nodes. Such a graph can be converted to a headed table (Fig. 1b), such that each table row represents a particular path in the workflow, while columns are used to report the nodes/entities and their attributes. Further rules allow one to represent complex topologies. For example, in the case of Figure 1b, having 'sample 1' in two subsequent rows means the table values are about the same node (crossed by two paths). Also note that the graph in Figure 1a could be reconstructed by means of more than two paths (and hence more than two rows), for example having two paths starting from 'source 1' and a third starting from 'source 2'. However, what is usually wanted is a most compact set of paths (and hence number of rows) that allows reconstruction of the original graph. It can be shown that such solution corresponds to the minimal set of paths that covers all the edges and all the nodes of the graph. A very convenient way to solve this problem is reducing it to a minimum flow problem, for which well-known efficient algorithms are available (Ciurea and Ciupalã, 2004). We refer the reader to the Supplementary material for the details on how we do the transformation to a minimum flow case and the reasons why we have chosen to exploit the Ford–Fulkerson algorithm (Ford and Fulkerson, 1987) for solving this problem.

1.2 Layering

We associate a layer index to each node, which reflects the node arrangement in Figure 1a, where each layer contains nodes of the same type and can be used to report the nodes in a table, using a homogeneous set of columns (Fig. 1b). While the general approach to achieve such a layering is known (http://www.mged.org/mage-tab/spec1.0.html), a number of details and ambiguous situations have to be dealt with in a concrete implementation. For example, that the columns about the 'labelled extract 1' in Figure 1a

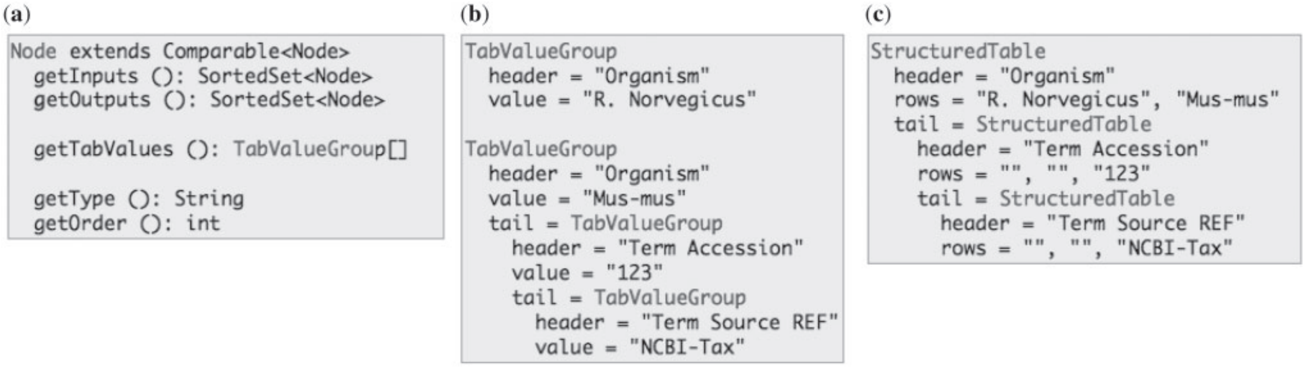
[*]To whom correspondence should be addressed.

© The Author(s) 2012. Published by Oxford University Press.
This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (http://creativecommons.org/licenses/by-nc/3.0), which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.



Source Name	Specie	Term ID	Term Source	Age	Unit	Sample Name	Sample Name	Protocol REF	Labeled Extract Name	Data File
Source 1	R. Norvegicus			1	yr	Sample 1		Protocol 1	Lbl Extract 1	data1.xml
Source 2	Mus-mus	123	NCBI Tax	8	wks	Sample 1	Sample 2			data1.xml

Fig. 1. (**a**) an experimental workflow graph and how it is layered. The node type difference between 'Protocol 1' and 'Sample 2' causes the layering gap after 'Sample 1' (**b**) the graph converted into a table (assuming nodes expose the attributes reported)



```
Node extends Comparable<Node>
  getInputs (): SortedSet<Node>
  getOutputs (): SortedSet<Node>

  getTabValues (): TabValueGroup[]

  getType (): String
  getOrder (): int
```

```
TabValueGroup
  header = "Organism"
  value = "R. Norvegicus"

TabValueGroup
  header = "Organism"
  value = "Mus-mus"
  tail = TabValueGroup
    header = "Term Accession"
    value = "123"
    tail = TabValueGroup
      header = "Term Source REF"
      value = "NCBI-Tax"
```

```
StructuredTable
  header = "Organism"
  rows = "R. Norvegicus", "Mus-mus"
  tail = StructuredTable
    header = "Term Accession"
    rows = "", "", "123"
    tail = StructuredTable
      header = "Term Source REF"
      rows = "", "", "NCBI-Tax"
```

Fig. 2. (**a**) the *Node* interface, which is the key to make graph2tab generic (**b**) the node attributes are encoded in the recursive *TabValueGroup* structure, which makes it possible to prepare a suitable result **(c)**, where values about the same headers are grouped together. Attributes are nested according to the required output format. In this example, the term accession is nested within the free-text value attribute about the organism, to reflect the fact that the identifier column has to be reported next to the 'Organism' column. In turn, the term source column is at the third nesting level, since the corresponding column must appear next to 'Term Accession'

come after the 'sample 2' has to be told to a computer somehow, in order to avoid the opposite column order.

In graph2tab, we allow one to specify this for a particular tabular format by means of the 'order' property in the *Node* interface (Fig. 2a). A concrete value for such property may be based on a pre-defined list of types, as well as on more sophisticated criteria, such as, for ambiguities involving protocols, the ontology terms about the protocol types (for instance, to establish that a sample treatment protocol comes before a labelling protocol). Further details about the layering algorithm can be found in the graph2tab user guide (https://github.com/ISA-tools/graph2tab/blob/master/ graph2tab_intro.pdf) and in the Supplementary Material.

1.3 Node headers and cell values

Once the minimum covering path set is computed and the node layers are established, the final table can be built by walking through every path and, for each node, reporting its contribution to the table headers and cell values, where such contribution consists of the node attributes mentioned above. In order to ease this last step, we require that a graph node exposes a nested structure of header/value pairs (Fig. 2a and b). This way, it is possible to take into account how the logic of the target format requires the columns to be grouped together and to progressively merge the attributes provided by each node into the final result (Fig. 2c).

2 USE CASES AND DISCUSSION

So far we have used the graph2tab library for realizing two database-to-tabular format exporters, one for the ISA software suite (Rocca-Serra *et al.*, 2010) which exports into the ISA-Tab format (Sansone *et al.*, 2008), and the other for the ArrayExpress repository (Parkinson *et al.*, 2011), which exports experimental data into the MAGE-TAB format (Rayner *et al.*, 2006). Other cases where we have used the library are: a MAGE-ML-to-MAGE-TAB converter (http://sourceforge.net/projects/mageml2tab/) and a data exporter for the Limpopo library (http://limpopo.sourceforge.net/).

Other tools similar to graph2tab exist. For instance, the caArray microarray data management software is able to export its data as MAGE-TAB files (https://cabig.nci.nih.gov/community/tools/caArray). Similarly, the BASE system (Vallon-Christersson *et al.* 2009), has a plug-in for exporting data into the Tab2MAGE format (http://baseplugins.thep.lu.se/wiki/uk.ac.ebi.Tab2MageExporter), a precursor of MAGE-TAB. Compared to these tools, graph2tab has the novel features of being a generic implementation of a size-optimized graph-to-table conversion, which, as described in this article, embeds the solution to several recurrent problems. In fact, the library can be adapted to any object model that encodes DAGs and any tabular format that represents DAGs by reporting one path per row and one attribute type per column. As we describe in detail in the user guide (https://github.com/ISA-tools/graph2tab/blob/master/graph2tab_intro.pdf) and in the on line code examples, this flexibility is achieved by letting developers to extend the library to accommodate their particular cases, for example a data exporter or data format converter. The expected input in such scenario is an object model (i.e. a set of Java objects instantiating pre-defined classes), which needs to be adapted to the *Node* interface in Figure 2a. This will include the definition of the *getTabValues()* method, i.e. of the headers and values to be generated in the output table, according to the particular tabular format that has been chosen as output.

Another benefit of the graph2tab implementation is that it makes available a graph-to-table conversion procedure that has been tested both from the theoretical point of view (see the Supplementary Material) and against real cases. Using the minimum path coverage of the input graph as a basis for the subsequent table generation corresponds to finding a maximally compact tabular representation of the original graph, a highly desirable feature, especially for the case of complex input workflows with many pooling and splitting operations. A possible limit of this approach is that, when the converted graph comes from an original submission based on a tabular format, the conversion result (e.g. a re-export needed after data review) is not necessarily identical to the original submission (e.g. if this was redundant and not minimal). In our experience, usually this is not considered as a problem by end users.

We expect that the graph2tab library will be useful in more similar cases in future, given the importance of experimental workflows in the biomedical field and the increasing popularity of tabular formats.

Moreover, since the biomedical experimental workflows are a particular case of more general provenance models (McCusker and McGuinness, 2010), the library has a potential to be useful in other application domains.

ACKNOWLEDGEMENTS

We thank Tony Burdett for the support with ArrayExpress and Limpopo code and Adam Faulconbridge, for having helped in fixing a few code flaws.

Funding: CarcinoGENOMICS (PL037712), EMBL and BBSRC (BB/I000771/1).

Conflict of Interest: none declared.

REFERENCES

Ciurea,E. and Ciupală,L. (2004) Sequential and parallel algorithms for minimum flows. *Korean J. Comput. Appl. Math.*, **15**, 53–75.

Field,D. *et al.* (2010) Meeting report: BioSharing at ISMB 2010. *Stand. Genomic Sci.*, **3**, 254–258.

Ford,L.R. and Fulkerson,D.R. (1987) Maximal flow through a network. In Gessel,I. and Rota,G.-C. (eds), *Classic Papers in Combinatorics*. Birkhäuser Boston, Boston, MA, pp. 243–248.

Kuchinke,W. *et al.* (2009) CDISC standard-based electronic archiving of clinical trials. *Methods Inf. Med.*, **48**, 408–413.

McCusker,J.P. and McGuinness,D.L. (2010) Explorations into the Provenance of High Throughput Biomedical Experiments. In, *Provenance and Annotation of Data and Processes -- Third Int Provenance and Annotation Workshop, IPAW 2010*. Troy, NY, pp. 120–128.

Parkinson,H. *et al.* (2011) ArrayExpress update—an archive of microarray and high-throughput sequencing-based functional genomics experiments. *Nucleic Acids Res.*, **39**, D1002–D1004.

Rayner,T.F. *et al.* (2006) A simple spreadsheet-based, MIAME-supportive format for microarray data: MAGE-TAB. *BMC Bioinformatics*, **7**, 489.

Rayner,T.F. *et al.* (2009) MAGETabulator, a suite of tools to support the microarray data format MAGE-TAB. *Bioinformatics*, **25**, 279–280.

Rocca-Serra,P. *et al.* (2010) ISA software suite: supporting standards-compliant experimental annotation and enabling curation at the community level. *Bioinformatics*, **26**, 2354–2356.

Sansone,S.-A. *et al.* (2008) The first RSBI (ISA-TAB) workshop: 'can a simple format work for complex studies?' *OMICS*, **12**, 143–149.

Shankar,R. *et al.* (2010) Annotare—a tool for annotating high-throughput biomedical investigations and resulting data. *Bioinformatics*, **26**, 2470–2471.

Vallon-Christersson,J. *et al.* (2009) BASE—2nd generation software for microarray data management and analysis. *BMC Bioinformatics*, **10**, 330.