Technical Report: Using Provenance for Generating Automatic Citations

Dai Hai Ton That, Andrew Youngdahl, Tanu Malik, Alexander Rasin
School of Computing, DePaul University, Chicago, IL, 60604, USA
{dtonthat, ayoungdahl, tmalik1, arasin}@depaul.edu

Abstract

When computational experiments include only datasets, they could be shared through the Uniform Resource Identifiers (URIs) or Digital Object Identifiers (DOIs) which point to these resources. However, experiments seldom include only datasets, but most often also include software, execution results, provenance, and other associated documentation. The Research Object has recently emerged as a comprehensive and systematic method for aggregation and identification of the diverse elements of computational experiments. While an entire Research Object may be citable using a URI or a DOI, it is often desirable to cite specific sub-components of a research object to help identify, authorize, date, and retrieve the published sub-components of these objects. In this paper, we present an approach to automatically generate citations for sub-components of research objects by using the object's recorded provenance traces. The generated citations can be used "as is" or taken as suggestions that can be grouped and combined to produce higher level citations.

CCS Concepts • **Software and its engineering** → **General programming languages**; • **Theory of computation** → *Program analysis*

Keywords Research Object, Provenance Graph, Automatic Citations, sub-component Citations

1. Introduction

Scientific research relies on the embedded citation of scholarly artifacts within publications. As computational experiments are becoming increasingly integral parts of research publications we must devise methods to publish and make citable the sub-components (files, computer code, outputs, etc.) which comprise these experiments. Doing so not only permits an author to give credit where credit is due, but also facilitates re-use of the whole, or parts, of the computational experiment.

However, computational experiments often have large numbers of sub-components. Moreover, many of these components could either be aggregated into general groupings and cited as such, or could be cited at a very granular and specific level. Thus, manual citation can be very laborious and error-prone. Authorship can be made much easier if an author is able to publish or download automatically generated citations at an appropriate level of specificity.

Recently, some studies about citation generation have been conducted. For instance, Buneman et al. have successfully proposed a general method to automatically generate citations when data is extracted from a database (Buneman et al. 2016). Citation is constructed from both the query and data. Another method presented in (Car et al. 2016) explores utilization of web services to provide citation information. Silvello G. introduces a system to automatically construct citations for hierarchical data using machine learning (Silvello 2017). In practice there are open online databases (Pawson AJ 2014; Car et al. 2016) which have citations generated automatically whenever the database returns a result to a user query. A user can easily cite the accessed data by copy-paste or other similar method.

While promising approaches, these methods rely on databases or web services being online to handle a users' query. This is not the case for reusable research objects which are packaged using a reproducibility tool such as `Sciunit` (Yuan et al. 2018) and published at third-party scholarly exchange websites (e.g., Hydroshare (Hydroshare.com 2017) or Figshare (Figshare.com 2017)). In this paper, we present an approach to automatically generating citations for sub-components of a research object by using the provenance traces recorded in `Sciunit`. To the best of our knowledge, our work is the first study in the context of reproducibility that generates citations for a reusable research object. Our study is built on top of our `Sciunit` reproducibility tool (Ton That et al.; Yuan et al. 2018).

The rest of this paper is presented as follows. Section 2 presents the frame work for our automatic generation of ci-

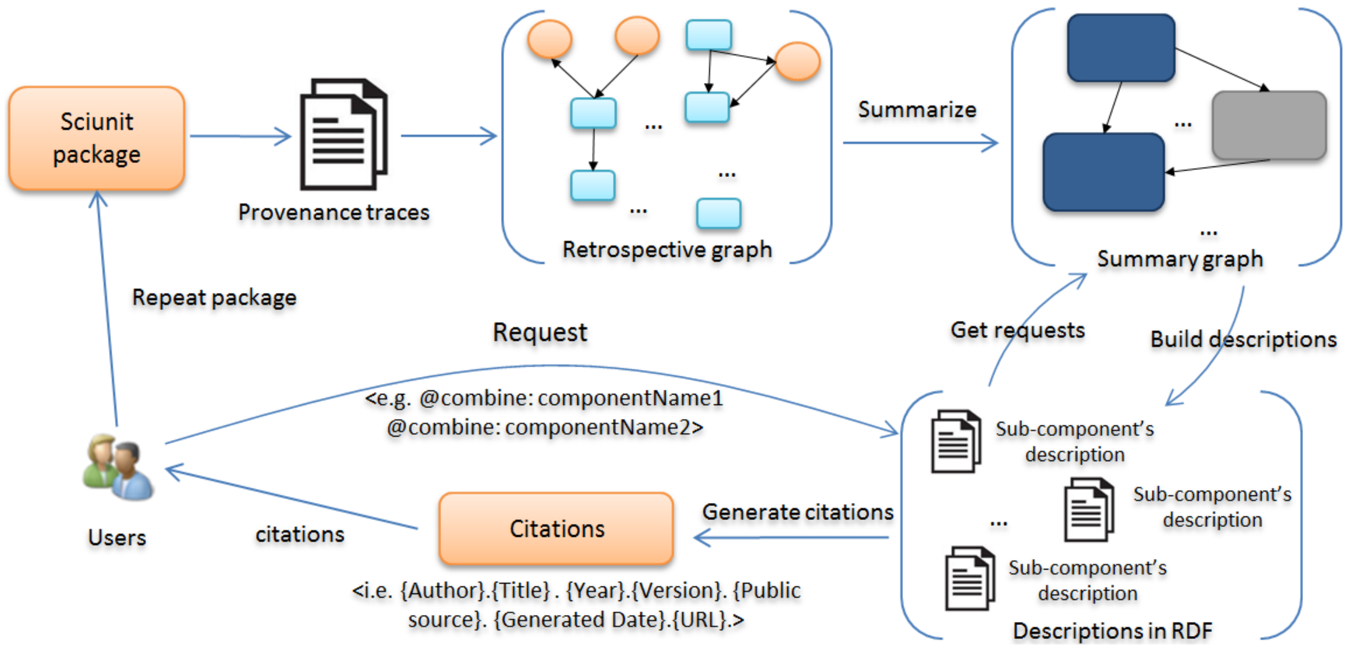

Sciunit package
Provenance traces
Retrospective graph
Summarize
Summary graph
Repeat package
Request
Get requests
Build descriptions
<e.g. @combine: componentName1 @combine: componentName2>
Sub-component's description
Sub-component's description
Sub-component's description
Descriptions in RDF
Users
citations
Citations
Generate citations
<i.e. {Author}.{Title}. {Year}.{Version}. {Public source}. {Generated Date}.{URL}.>

Figure 1. Automatic Generating Citation Framework

tations for a research object. We present preliminary evaluations with two use cases in Section 3. Finally, we conclude the paper in Section 4.

2. Citation Generation Framework

As stated in the introduction, computational research objects are playing roles similar to that of academic publications and authors and readers increasingly need to cite either an entire object or a particular component within an object. To provide this citation information in a packaged publication, citations will need to be embedded in the research object container. Figure 1 shows the automatic citation generation framework in `Sciunit`. Generating citations for a research object package consists of the following steps: (i) Record all provenance traces in a package execution and build a retrospective graph. (ii) Summarize the retrospective graph; this step should identify citable sub-component groupings. (iii) Create RDF descriptions of the package sub-component groupings as requested by the user. Each sub-component's RDF description will record all information needed to create a citation for that sub-component. (iv) Get data from a sub-component's description and construct the citations. (v) Adjust the suggested citations.

Provenance traces and retrospective graph. Provenance information consists of execution traces and the relationships between objects (e.g., resources, processes, dependencies, etc.) in a computing research object. In `Sciunit`, we capture a provenance log file for use in auditing and repeating processes. This information is crucial for understanding a research object, analyzing executions or experiments (Dey et al. 2015) or even aiding in exact, partial or modified reuse (Ton That et al.; Yuan et al. 2018). In this research, we exploit provenance trace information for generating citations.

However, raw provenance traces are typically very long and not easily read by a human. Considering this we build a retrospective graph from the provenance log file to assist in trace visualization. This retrospective graph is often structured in such a way that it provides a reasonable basis for producing citation groupings.

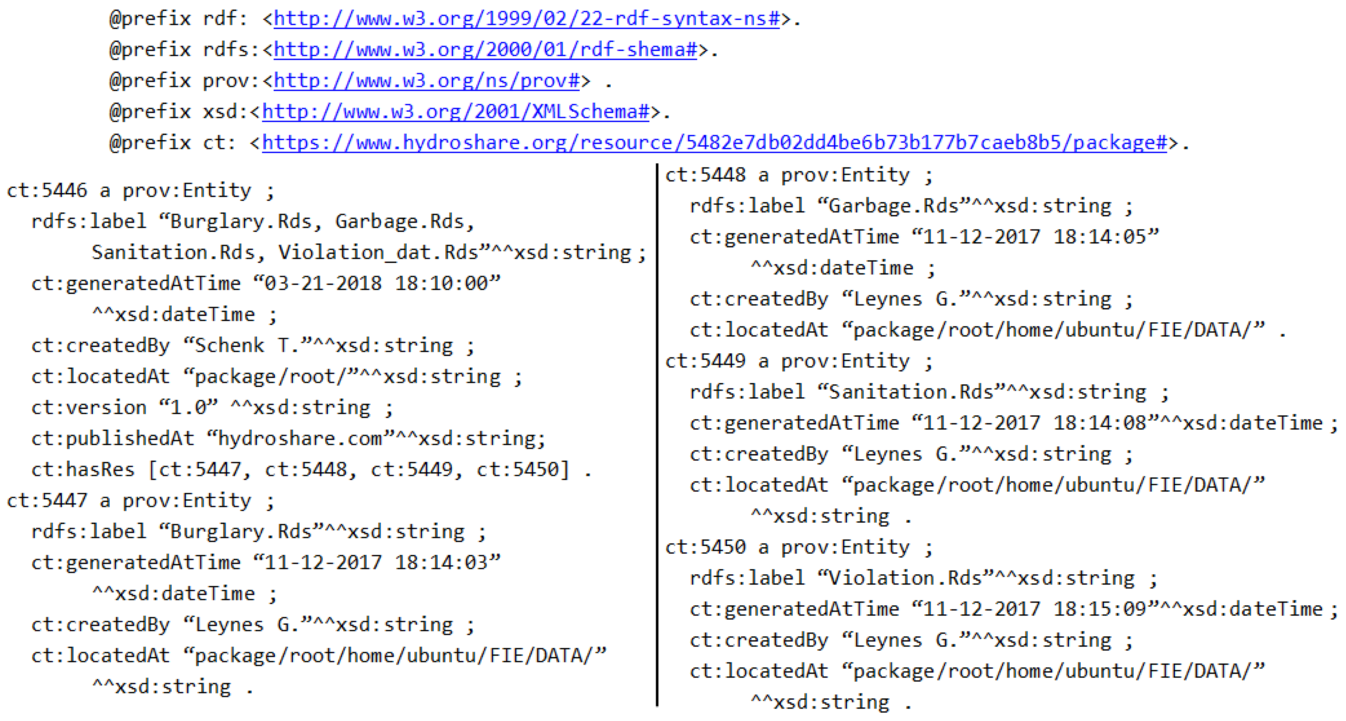
Summarize retrospective graph.

As noted previously, a good method of citation generation ought to produce citations that cite as specifically as possible the utilized resource (e.g., a table in a paper, a file in a system, a model in a container, etc.) rather than cite the resource in its entirety (e.g., a book, a database website, etc.) while avoiding redundancy and also keeping the number of citations to a human readable amount. In hopes of achieving this we summarize as follows.

Consider a graph $G = (V, E)$ where V and E are sets of nodes and edges respectively. Graph nodes are classified into two types of node: `Activity` node and `Entity` node according to processes or files in the provenance graph. Meanwhile, there are three types of edges (i.e., $used$, $wasGeneratedBy$ and $wasInformedBy$) with respect to three types used in W3C PROV standard.

Now, create summary groups according to the following rules:

Rule 1. Common type *(Every node in a group should share the same type).* Given a grouping $\Phi = \{g_1, g_2..., g_k\}$ in G, $\forall u, v \in g_i$, then $Type(u) = Type(v)$.



```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs:<http://www.w3.org/2000/01/rdf-shema#>.
@prefix prov:<http://www.w3.org/ns/prov#> .
@prefix xsd:<http://www.w3.org/2001/XMLSchema#>.
@prefix ct: <https://www.hydroshare.org/resource/5482e7db02dd4be6b73b177b7caeb8b5/package#>.
```

```
ct:5446 a prov:Entity ;
  rdfs:label “Burglary.Rds, Garbage.Rds,
       Sanitation.Rds, Violation_dat.Rds”^^xsd:string ;
  ct:generatedAtTime “03-21-2018 18:10:00”
       ^^xsd:dateTime ;
  ct:createdBy “Schenk T.”^^xsd:string ;
  ct:locatedAt “package/root/”^^xsd:string ;
  ct:version “1.0” ^^xsd:string ;
  ct:publishedAt “hydroshare.com”^^xsd:string;
  ct:hasRes [ct:5447, ct:5448, ct:5449, ct:5450] .
ct:5447 a prov:Entity ;
  rdfs:label “Burglary.Rds”^^xsd:string ;
  ct:generatedAtTime “11-12-2017 18:14:03”
       ^^xsd:dateTime ;
  ct:createdBy “Leynes G.”^^xsd:string ;
  ct:locatedAt “package/root/home/ubuntu/FIE/DATA/”
       ^^xsd:string .
```

```
ct:5448 a prov:Entity ;
  rdfs:label “Garbage.Rds”^^xsd:string ;
  ct:generatedAtTime “11-12-2017 18:14:05”
       ^^xsd:dateTime ;
  ct:createdBy “Leynes G.”^^xsd:string ;
  ct:locatedAt “package/root/home/ubuntu/FIE/DATA/” .
ct:5449 a prov:Entity ;
  rdfs:label “Sanitation.Rds”^^xsd:string ;
  ct:generatedAtTime “11-12-2017 18:14:08”^^xsd:dateTime ;
  ct:createdBy “Leynes G.”^^xsd:string ;
  ct:locatedAt “package/root/home/ubuntu/FIE/DATA/”
       ^^xsd:string .
ct:5450 a prov:Entity ;
  rdfs:label “Violation.Rds”^^xsd:string ;
  ct:generatedAtTime “11-12-2017 18:15:09”^^xsd:dateTime ;
  ct:createdBy “Leynes G.”^^xsd:string ;
  ct:locatedAt “package/root/home/ubuntu/FIE/DATA/”
       ^^xsd:string .
```

Figure 2. An example of sub-component's description in RDF format

Rule 2. Similar ancestry *(All units (i.e., groups or nodes) in a group have to share the same ancestry).* Given a group g in G, $\forall u_1, u_2 \in g$, then $Ancestor(e_1) = Ancestor(e_2)$.

Rule 3. Isomorphism grouping *(All units in a group should be isomorphic).* Given a group g in $G, \forall u_1, u_2 \in g$, if there is a edge $\exists e_1 \in E$, $e_1 = (u_1, x)$ *or* (x, u_1), then there is $e_2 \in E$, $e_2 = (u_2, x)$ *or* (x, u_2) and $Label(e_1) = Label(e_2)$.

This sub-component grouping often generates summary graphs which not only resemble human drawn application workflows, but also provide reasonable groupings for aggregating citations for display and publication. Due to the space limit, we omitted the presentation of the algorithm for this summarization technique. For more details, see our technical report (Ton That et al. 2018).

The algorithm 1 shows how the summarization technique was implemented.

Build sub-component descriptions. Each sub-component can contain many elements which may not be organized hierarchically. To adequately describe these elements we build a single RDF format [1] document for each subgroup. These descriptions are used in citation generation. An example of sub-component's description is shown in Figure 2.

Generating citations. We create human-readable citations in the following format:

```
{Author}.{Title}.{Year}.{Version}.
{Public source}.{Generated Date}.{URL}.
```

[1] https://www.w3.org/RDF

The individual fields of the citation (such as author, title, date, url) are built via operations on the RDF description for the sub-component. For example, authorship information is the union of all the authors of all elements in a sub-component. `URL` is the uniform resource locator of the sub-component description. An example of a generated citation to a sub-component for a research object (City of Chicago 2017) is shown in Table 1.

Adjusting the citations. Citations are automatically generated whenever a `Sciunit` package is published or re-published. However, the generated citations may not be to the author's liking. To address this possibility our system allows the user to either split a sub-component or to combine multiple sub-components by adding a meta description file with the following syntax:

```
To combine: @combine <componentName1> ...
<componentNameN>
To split: @split <componentName>
To rename: @rename <oldName> <newName>
```

The citations will be updated when the user re-publishes the container using `Sciunit`.

3. Preliminary evaluation in two research object use cases

We apply our method on two research object use cases: FIE (City of Chicago 2017) and Data-driven street flood severity modeling in Norfolk (DSF) (Sadler 2018). As shown in Table 2, there are some differences between cita-

Table 1. Citation examples

FIE	suggested	`Schenk T. and Leynes G.. Burglary.Rds, Garbage.Rds, Sanitation.Rds, Violation_dat.Rds. 2018. Version 1.0. Hydroshare.com. Generated 03-21-2018. https://www.hydroshare.org/resource/5482e7db02dd4be6b73b177b7caeb8b5/package/5446.`
	manual	`Schenk T. and Leynes G.. Heat map data: Burglary.Rds, Garbage.Rds, Sanitation.Rds, 2018. Version 1.0. Hydroshare.com. Generated 03-21-2018. https://www.hydroshare.org/resource/5482e7db02dd4be6b73b177b7caeb8b5/package/heat_map_data.`
DSF	suggested	`Sadler J.. Prepare flood events table. 2018. Version 1.0. Hydroshare.com. Generated 03-21-2018. https://www.hydroshare.org/resource/8db60cf6c8394a0fa24777c8b936f32d/package/3514.`
	manual	`Sadler J.. Script for processing street flood reports. Version 1.0. Hydroshare.com. Generated 03-21-2018. http://dx.doi.org/10.4211/hs.38a4ce62960942b4ad8398ee58a777cf.`

Table 2. Preliminary results in two use cases

	FIE	**DSF**
# of citations (manually)	11	10
# of suggested citations	9	5
# of matched citations	8 (~73%)	4 (~40%)

tions generated by our framework and those manually made by users. The results are quite close in FIE (i.e., 73% match), however in DSF only 4 citations out of 10 match citations built by users. Some citation examples are shown in Table 1. The automatically generated citations are sightly different from those manually generated in the `Name` and `URL` snippets. The differences in `Name` are a result of obtaining names directly from sub-component element names as opposed to human authored names. As a correction a user may supply desired names via a meta data file and re-publish the package. The differences in `URL` are due to the differences of publishing methods between our system (i.e., using `Sciunit`) and user (i.e, manually publishing).

4. Conclusion

We have presented our framework for automatic generation of citations of a container in `Sciunit`. Our construction method relies on a summarization technique that decomposes a retrospective graph into sub-components. Experiments show that the suggested citations for the evaluated projects are reasonable. However, the automatic generation can still require some author intervention to alter citation content in some cases.

A. Appendix Title

Algorithm of summarization the retrospective graph is as following:

Algorithm 1: Citing component identifying

```
1  Grouping ():
2      Group all $\{nodes\}$ with same types into $\{g_i\}$ and store all groups in $TypSet = \{g_i\}$;
3      $AncSet = \emptyset$;
4      $\Phi = \emptyset$;
5      foreach (group $g$ in $TypSet$) do
6          foreach (element $u$ in $g$) do
7              Calculate Ancestor of $u$;
8          Partition all $\{u_i\} \in g$ into $\{h_i\}$ with the same Ancestor;
9          $AncSet \longleftarrow \{h_i\}$;
10     foreach (group $g$ in $AncSet$) do
11         $NbSet$ = NeighborSet($g$);/*Get all neighbor groups of $g$*/
12         foreach (element $v$ in $NbSet$) do
13             foreach (element $u$ in $g$) do
14                 foreach (type $t$ in $EdgeType$) do
15                     Calculate the number of edges between $u$ and $v$;
16         Partition all $u_i$ in $g$ into $\{p_i\}$ with the same number of edges for each type and destination/source;
17         $\Phi \longleftarrow \{p_i\}$;
18     return $\Phi$;
```

Acknowledgments

The authors would like to thank Tom Schenk and Gene Leynes for the FIE use case and Jonathan Goodall and Jeff Sadler for the DFS use case. This work is supported by the National Science Foundation under grants NFS ICER-1639759, ICER-1661918, ICER-1440327 and ICER-1343816.

References

P. Buneman, S. Davidson, and J. Frew. Why data citation is a computational problem. *Commun. ACM*, 59(9):50–57, Aug. 2016. ISSN 0001-0782. doi: 10.1145/2893181.

N. J. Car, L. S. Stanford, and A. Sedgmen. Enabling web service request citation by provenance information. IPAW 2016, 2016.

City of Chicago. Food Inspection Evaluation. `https://chicago.github.io/food-inspections-evaluation/`, 2017. [Online; accessed 05-2017].

S. Dey, K. Belhajjame, D. Koop, M. Raul, and B. Ludäscher. Linking prospective and retrospective provenance in scripts. TaPP'15, pages 11–11, Berkeley, CA, USA, 2015.

Figshare.com. Figshare. `https://figshare.com/`, 2017. [Online; accessed 2-May-2017].

Hydroshare.com. Hydroshare. `https://www.hydroshare.org`, 2017. [Online; accessed 2-May-2017].

B. H. e. a. Pawson AJ, Sharman JL. The iuphar/bps guide to pharmacology: an expert-driven knowledgebase of drug targets and their ligands. *Nucleic Acids Research*, 42, 2014.

J. Sadler. Data-driven street flood severity modeling in Norfolk, Virginia USA 2010-2016. HydroShare. `http://www.hydroshare.org/resource/9db60cf6c8394a0fa24777c8b9363a9b`, 2018. [Online; accessed 21-Mar-2018].

G. Silvello. Learning to cite framework: How to automatically construct citations for hierarchical data. *Journal of the Association for Info. Science and Tech.*, 68(6):1505–1524, 2017.

D. H. Ton That, G. Fils, Z. Yuan, and T. Malik. Sciunits: Reusable research objects. In *eScience 2017*, Auckland, New Zealand.

D. H. Ton That, A. Youngdahl, T. Malik, and A. Rasin. Technical report. `https://sciunit.run/papers/TechReport.pdf`, 2018.

Z. Yuan, D. H. Ton That, S. Kothari, G. Fils, and T. Malik. Utilizing provenance in reusable research objects. *Informatics*, 5(1), 2018. ISSN 2227-9709. doi: 10.3390/informatics5010014.