XHTML Documents with Inline, Policy-Aware Provenance

by

Harvey C. Jones

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of Master of Engineering in Computer Science and Engineering

at the MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2007

© Massachusetts Institute of Technology 2007. All rights reserved.
Abstract

Constructing code that aids users in making information policy decisions in a decentralized Web-based environment will require seamless integration between document content and metadata. A system of embedding and reasoning over relevant policy information will be an important step towards this integration. In this thesis I present a method for annotating XHTML documents with provenance metadata using RDFa. I present a method of performing copy and paste operations on XHTML documents that preserves this metadata. Finally, I apply these tools to the practical use case of an end user composing documents from multiple sources with Creative Commons licenses. By using provenance metadata, the user is able to combine arbitrary Creative Commons works and use automated tools to calculate the applicable license of the composite work.

Thesis Supervisor: Daniel J. Weitzner
Title: Principal Research Scientist
Acknowledgments

Many people assisted me with the work described in this thesis. I would especially like to extend acknowledgments to:

Danny Weitzner for everything he did as my advisor, but especially for introducing me to the world of Internet policy.

Hal Abelson and Gerry Sussman for their excellent technical advice and extensive experience in ensuring that Masters students do, in fact, finish their theses on time.

Ben Adida, for valuable advice regarding RDFa.

Tim Berners-Lee, for his advice, particularly when I was struggling with how to address the problem of copying content.

Chris Hanson, for his patience and excellent teaching abilities, especially when I was just getting off the ground as an RA at DIG.

Sandro Hawke, for helping me with key design decisions regarding the provenance ontology.

The TAMI team as a whole, for providing weekly encouragements and sanity checks.

Jason Juang, because I learned a lot of \LaTeX{} from the final project we did together.

José Soltren, for partnering with me on the 6.805 project that was my first step down this path.

The residents of East Campus’s Second West and Second East, for giving me a home away from home.

My family, for their support throughout the years.
# Contents

1 Introduction ................................................. 9  
  1.1 Inline Provenance .................................. 11  
  1.2 Policy-Aware Applications ...................... 12  

2 Motivation .................................................. 13  
  2.1 DRM and Up-Front Enforcement ................. 14  
  2.2 Web-based Copyright Enforcement ............. 15  

3 Copyright Usage Scenarios ............................... 17  
  3.1 Motivating Scenario ............................... 17  
  3.2 Simple Scenario .................................... 18  
  3.3 Desired User Outcome ........................... 20  
  3.4 Necessary Tools ................................... 20  

4 Ontology ..................................................... 21  
  4.1 RDF Ontology Specification ................... 21  
  4.2 Original Metadata .................................. 22  
  4.3 Semantics of Nested Document Fragments ...... 23  
  4.4 RDF Ontology Tradeoffs ......................... 24  
  4.5 Implementation .................................... 25
Chapter 1

Introduction

In a world where information is the most valuable commodity, policy decisions regarding the use of information are becoming more and more important. Information policies will rely on the provenance of data: information about the data's creation, derivation, and transfers. Policies will also need domain-specific information about the data in question, such as licensing information in the copyright domain. This thesis aims to produce a set of tools that will inform users as they make policy decisions, with a special focus on copyright policy.

The copyright status of information on the World Wide Web is currently very difficult to ascertain. The ability of users to rapidly copy and recombine data from disparate sources has rapidly surpassed the ability of any technical policy or social norm to control copying.

In an environment where one piece of content may be reused in hundreds of different contexts, and one work may be an aggregation of dozens of sources, regulating the re-use of web content will require users to make policy decisions on a Web scale. Automated aids will be required to allow human decision-makers to make reasonable decisions. Without them, even conscientious document creators may end up infring-
ing on copyrights by inadvertently perpetuating someone else’s infringement; I may take information from Wikipedia that I believe to be GFDL-licensed, and only later find out that the information should never have been copied to Wikipedia in the first place. Additionally, the fact that documents may be licensed under a virtually infinite number of terms makes it difficult for end users to understand what their rights are, especially when they are constrained by the terms of multiple licenses.

The Creative Commons project is an essential first step towards addressing the problem of copyright on the Web. Creative Commons has strived to produce a simple, uniform, understandable set of licenses that content creators may issue their content under. By reducing the size of the license space from a virtually infinite number to roughly a dozen licenses, Creative Commons has given end users a much clearer picture of what their rights are. Even more importantly, by creating a small set of licenses which are internally consistent, composite document creators are much more likely to be able to reconcile licensing restrictions from several data sources.

Creative Commons is rapidly becoming the de facto set of copyright licenses on the Web. Support for searching for Creative Commons data is integrated into the leading search engines, including Yahoo and Google. Flickr, the largest Web-based photo sharing service, allows users to tag photos with Creative Commons licenses. Currently, there are over 30,000,000 photos tagged with CC licenses on Flickr. [1]

Placing information about data provenance inline with document content will enable powerful software tools to take advantage of the Creative Commons framework. Inline metadata will enable end users to create composite documents from multiple sources and perform rules-based policy calculations on these documents. The fact that policy data will be in a canonical place will guarantee that automated tools will find all relevant information. Users will be able to perform calculations such as determining the path data took to its final location without having to explicitly track information about document sources. By combining embedded metadata with a clear
combinable set of licenses such as Creative Commons, computer programs will be able
to automate many tedious parts of the licensing process. This will allow document
producers to make informed policy decisions with a minimum of effort.

1.1 Inline Provenance

An artifact’s provenance is the history of its ownership. In the context of the World
Wide Web, the artifacts are pieces of information, and provenance data is data that
describes the authorship of information, its original source, and any intermediate
locations the data may have resided before being copied to its current location.

A key question in the design of a system which uses provenance information is
where the information should be stored. One could keep track of all network trans-
actions and generate an audit trail from the logs after the fact. [2] One could create
editing tools that store information about the sources of the document in a separate
file locally. These designs do not require any changes to the document in question,
but they require serious changes to underlying infrastructure used to produce the
document. The logging approach assumes serious changes to the underlying network
infrastructure. The rewritten editing tool approach requires changes to any tool used
for creating, copying, or transmitting documents. In this project I will examine an
approach which uses annotations within the original document to create a method
of tracking provenance that does not require changes to network infrastructure or to
tools which transmit these documents.

RDFa is a method of embedding structured RDF in XHTML documents with mini-
mal additional markup. Keeping provenance inline with XHTML follows one of the
guiding principles of the Internet: the End to End Principle[4]. By embedding prove-
nance information inside the format, provenance information will only need to be
dealt with at two points; the point of document creation, and the point of document consumption. As a result, provenance-embedded documents may be transmitted via email or the Web. These documents may be encrypted or used with any filesystem, with no modifications to underlying technology.

1.2 Policy-Aware Applications

In the following chapters I present a method for annotating XHTML documents with provenance metadata using RDFa. I present a method of performing copy and paste operations on XHTML documents that preserves this metadata. I constructed a Creative Commons reasoning engine that reads document metadata and makes licensing decisions for annotated documents. Automated reasoning leverages one of Creative Commons's key strengths: the ability to take the intersection of licenses and determine appropriate licenses for a composite work.

By using these tools, an end user is able to combine arbitrary Creative Commons licensed works and automatically calculate the applicable license of the composite work.
Chapter 2

Motivation

The technical work in this thesis is set against the background of copyright law and policy. It is important to understand the terminology and enforcement regimes of copyright before explaining the proposed technical work.

First, we should review terminology that will be used throughout this paper. A “work” is any material that may be copyrighted. The US Copyright Office states that copyright protects “original works of authorship including literary, dramatic, musical, and artistic works, such as poetry, novels, movies, songs, computer software, and architecture.” [5] “Infringement” is the act of making use of a work in a way protected by copyright law (publishing, copying, performance, etc.) without permission from the owner of these rights. “Composite work” is a term coined for the purposes of this paper. It refers to the combination of literal excerpts from multiple copyrighted sources to create a new work.

The traditional method of enforcing copyright is a model based on after-the-fact auditing. Works that reference or copy previously created material embed an implicit audit trail by including recognizable portions of their antecedents. This audit trail is then analyzed by the legal system, and if works are found to be in violation of
copyright law, the infringers are subject to legal remedies.

2.1 DRM and Up-Front Enforcement

The traditional copyright enforcement approach is the antithesis of the use of Digital Rights Management (DRM) systems to control access to copyrighted material. In "The Technical and Legal Dangers of Code-Based Fair-Use Enforcement", Erickson and Mulligan [6] describe the difficulties that would be involved in trying to move copyright law towards a DRM model. They state that even those rights clearly articulated by copyright “are subject to myriad exceptions, the applicability of which depends upon a variety of factors including role, intent, and purpose of use, just to name a few.” Making real-time decisions about whether to allow or deny access to material requires modelling an enormously complex system with constantly evolving parameters.

DRM schemes ultimately require a central point of control, which can present problems from a policy perspective. Policy decisions have to be made with some frequency, but there are a number of feasible locations to embed decision-making logic. The designer of a DRM system could opt to embed control in the player, embed control in the media, or locate the control of the system in a centralized location. [cite] Embedding control in the media or end-user devices is a brittle system, as producing many copies of a movie, or many copies of a DVD player, will tend to lock the content industry into a rigid rights scheme early on. DRM manufacturers may seek to modularize control and embed it off site, but this creates its own set of problems. Embedding control of information off site runs counter to the established principles of the World Wide Web. Off-site control establishes a single point of failure for any system of control. It also makes operation of devices impossible without communication to the central site. Requiring authentication for every use of a copyrighted work
raises serious questions about end user privacy [7], and threatens to bring the (previously completely unregulated) realm of private use under a new regulation regime. If private uses of a copyrighted work must be approved by a central body, this also means that they may be monitored by a central body. As a result, corporations could track every single time a person viewed a DRM movie or listened to a DRM song.

Another shortcoming of DRM is that it lacks the flexibility of a human decision-maker. In certain situations it is possible to desire that almost all access controls on information should be disregarded. In "The Technical and Legal Dangers of Code-Based Fair-Use Enforcement", Erickson and Mulligan liken the current copyright regime to our current system of traffic control.[6] It is possible to run a red light, but anyone seeking to violate the red light policy must circumvent strong social and legal barriers to do so. In certain cases (ambulances, avoiding accidents), other needs become paramount and an actor violates the policy. In these cases, a judgment is made after the fact as to whether the violation of the red light policy was appropriate. Although copyright decisions are rarely life-and-death, the same sort of decision-making applies to fair use. Just as it would be impractical to have a central authority selectively allow ambulances to run red lights, it is impractical to have a central authority dictate when users may violate the letter of copyright law for purposes deemed "fair use".

2.2 Web-based Copyright Enforcement

Current DRM systems such as DVD encryption or those systems protecting online music stores represent a single point of failure. These systems, by focusing solely on preventing unauthorized access to decryption keys, are rendered completely useless once they are first compromised. [8] Because there is no way to restrict the copy and performance of unlicensed content, a single unauthorized access to decrypted content
can be used to create unlimited copies of the work in question. Because of this, no uncontrolled accesses can ever be allowed.

On the World Wide Web, a more flexible approach to controlling copying will be required. Often, Web authors post their content with the understanding that it will be quoted, copied, and reused. Further, they may wish that their work only be used with attribution, or only for educational use, etc. Because of the decentralized nature of the Web, it is impossible for content authors to explicitly grant permission for every individual instance of potential reuse. If authors want their content to be reused, they will not be able to use DRM to regulate the reuse. A more flexible method of constraining use is required. Luckily, copyright law has constrained use through legal and societal norms for hundreds of years by proscribing legal penalties for unauthorized use when it is discovered. By creating negative incentives for illegal uses of copyrighted data, copyright law has restricted copying in a method that is robust against technological attack. This method of enforcing copyright is ideal for a Web-based environment, as it provides negative incentives for infringement, while not preventing any authorized use.

The elimination of pre-emptive enforcement shifts the burden of making policy decisions onto the end user. Because no action is explicitly forbidden, the end user does not have obvious cues as to what behaviors may be acts of copyright infringement. This is an area in which automated policy tools will be able to help. The concept of embedding provenance as metadata in compound documents dovetails with the idea that after-the-fact auditing is an effective means of enforcing copyright. Embedding provenance data within documents will allow a hybrid system with the automated policy calculations of DRM systems and the enforcement model of the current copyright regime.
Chapter 3

Copyright Usage Scenarios

In order to apply the proposed technology to a real-world use case, we should first examine a scenario in which the tools described later in the thesis would be used. First, I will describe a complex scenario involving many sources of content which will illuminate the need for automated tools. Next, I will describe a scenario simple enough to trace by hand. Each scenario concerns a professor producing course material (the “composite work”) from several sources located on the Internet (the “original works”). These sources have had Creative Commons licenses associated with them, and the professor wants to make a good-faith effort to follow these licensing terms when redistributing the composite work. The goal of this scenario is to develop tools that will allow the professor to construct composite documents, and view the license terms that can be applied to them without having to explicitly track any licensing data.

3.1 Motivating Scenario

A professor constructs a set of lecture slides. These slides contain data from many different sources: original content created by the professor, photos with Creative
Commons licenses obtained from Flickr, and several slides copied verbatim from a colleague’s lecture slides. These copied slides also contain original content and Creative Commons-licensed photographs. The professor does not pay much attention to copyright, as he intends to use the slides only for educational purposes, generally covered under fair use.

Two years later, the professor is contacted by the editor of “Great Slideshow Presentations of the 21st Century”, a publication which will highlight series of exceptional lecture slides. The professor is now in the position of having to relicense his work for commercial use. A key component of this is obtaining clearance for all copyrighted material included in the presentation. Without any copyright information, the professor may be reduced to attempting to replicate the same searches that he performed the first time. After two years’ time on the World Wide Web, it will probably be impossible to find the original creators of content and obtain clearance.

3.2 Simple Scenario

A professor wishes to construct a website containing quotations from two sources, which are also available on the web. The first is a paper written on the privacy implications of Facebook (www.facebook.com). The second is a blog posting on the same topic.

The quotation from the paper is the text “Facebook is evil”. The quotation from the blog entry is “Facebook is a necessary evil.”

In order to provide a fuller picture of the possible outcomes of this scenario, I will examine two variations: one in which the net effect of the licensing is to allow redistribution, and one in which the net effect of the licensing is prevent any further licensed use of the document.

In the first instance, the first original work is licensed under a CC-Attribution-
Facebook analysis, by A. Professor

Facebook is evil. (CC) BY
Facebook is a necessary evil. (CC) BY-NC

Figure 3-1: A combination of BY and BY-NC content.

Facebook analysis, by A. Professor

Facebook is evil. (CC) BY-SA
Facebook is a necessary evil. (CC) BY-NC

Figure 3-2: A combination of BY-SA and BY-NC content.

NonCommercial license, and the second original work is licensed under a CC-Attribution [9] license. This produces a CC-Attribution-NonCommercial license for the composite work.

In the second instance, the first original work is licensed under a CC-Attribution-ShareAlike license, and the second original work is licensed under a CC-Attribution-NonCommercial license. Thus, the first original work's NonCommercial attribute requires that the composite work be licensed under a NonCommercial license, and the second original work's ShareAlike attribute requires that the composite work be relicensed with only the ShareAlike attribute. [10] As a result, the composite work may not be redistributed under any Creative Commons license.

19
3.3 Desired User Outcome

The professor creates a new, blank XHTML document. The professor navigates to the web site that contains the first source, and copies data from it using the “Copy” menu item. The professor returns to his created document and pastes the excerpt into it. The professor then navigates to the second site, and copies a different excerpt. The professor then returns to his created document and pastes in this second excerpt. Next, the professor saves this document, and navigates to it on his personal webspace. He clicks a “Verify” button on his bookmarks toolbar, which presents him with a list of acceptable Creative Commons licenses for his document.

3.4 Necessary Tools

This scenario presupposes the existence of a number of technical tools that were previously unimplemented. For the scenario to be completed, there must exist a document format that contains metadata about its creator, creation time, and license. This will allow the professor (or his computer) to easily determine the author and rights associated with the work he quotes. This document format must also be able to represent the fact that subsections of the professor’s document are quotations from other locations. The “copy” clipboard operation must be able to excerpt portions of this document format while preserving the formatting of any existing metadata. Additionally, it must be able to create new metadata representing the fact that the clipboard data is an excerpt from another document. Likewise, the “paste” operation must preserve this metadata. Finally, the professor needs a tool that is able to read and analyze this document format, combine the Creative Commons licenses that are associated with the various quotations, and present appropriate licenses to him.
Chapter 4

Ontology

The primary unit of the provenance ontology is the Document Fragment. A Document Fragment is a DOM node annotated with the DocumentFragment RDF type. The Document Fragment as a whole represents a quotation; the annotated DOM node is the quoted content, and the Document Fragment RDF type is a class with properties that represent all relevant information regarding the quotation. It encapsulates information related to the original creator of the content, how and when the content ended up in the current document, and what license the content creator released the content under. In developing this ontology, I read and considered the ontologies developed by the Stanford Knowledge Systems Laboratory [11], as well as Jennifer Golbeck of the University of Maryland [12].

4.1 RDF Ontology Specification

The DocumentFragment type that can have the following properties:

- creator: <person> The person that created the annotated content.
• creationTime: `<time>` The time at which the creator created the document.

• sourceURI: `<uri>` The URI of the original document.

• allowedPurposes: `<uri>` A representation of the allowed purposes associated with the data.

### 4.2 Original Metadata

In order to generate excerpts of a document annotated with data such as the creator, creation date, and license of the original work, we must have a method of obtaining these properties from the original document. The Document Fragment ontology describes quotations of works; automated tools for generating these quotations must be able to locate who the author of the original document is, and what rights are associated with the document. The de facto standard for document metadata terminology is the Dublin Core Metadata Initiative.[13] For the purposes of this thesis, source documents will be annotated with Dublin Core metadata, thus allowing the “copy” operation to transparently obtain these properties. Below is an example of an XHTML document annotated with Dublin Core and Creative Commons metadata.

```html
<html xmlns:cc='http://web.resource.org/cc'
xmlns:dc='http://purl.org/dc/elements/1.1'>
...  
<meta property='dc:author' content='John Doe'/>
<meta property='cc:license' content='.../by/2.5'/>
...  
</html>
```
4.3 Semantics of Nested Document Fragments

In a Web-based environment, documents may be derived from dozens of ancestors, each of which may have a further set of ancestors. Many applications of policy-aware provenance require a reconstruction of the path that copied content took to its location in the current document. Thus, the Document Fragment ontology must be able to represent a chain of quotations, representing an chain of "Copy" operations.

As mentioned above, a Document Fragment is an annotated DOM node. A DOM node is part of a DOM tree, and as such can contain many sub-nodes. When a DOM node in a Document Fragment's content is annotated, we then have nested Document Fragments. The provenance data associated with this RDF node is interpreted as being associated with this DOM node and all of its children, including any child nodes that have associated provenance data. Thus, nested Document Fragments can be thought of as representing a data flow. The innermost fragment contains provenance data associated with the document's creation, and each successive transfer is represented by enclosing the entire flow up to that point in another layer of provenance metadata. An annotated document can thus be turned into an audit trail by peeling off successive layers of metadata and reconstructing the flow of the data.

These semantics allow Document Fragments to nest, with \((C(B(A)))\) implying that the excerpt had provenance information \(A\) associated with it at the time of original creation. It was then copied, an event described by provenance information \(B\). Finally, it was copied a third time into its final location, a process described by provenance information \(C\). The resulting structure is thus \((C(B(A)))\).
4.4 RDF Ontology Tradeoffs

This model does not store enough information to view previous revisions of documents. Thus, it would not be possible to encapsulate all of the information associated with a full Wikipedia edit history, or a file in an Subversion repository. RDFa puts RDF data inline in an XHTML document, and as a result, is well-matched to annotating one revision of a document, rather than the full edit history. There is nothing that prevents this ontology from being used to annotate a single-document edit history, however.

The model contains minimal information on how the data inside a document fragment arrived at its final location. This allows us to sidestep the problem of developing a potentially massive and very domain-dependent ontology for general-purpose data flow, and allows us to focus on the problem of document composition. This model is well-matched to situations such as copyright, where the most important
question is the original source of embedded content and the ways that its creator has allowed it to be used. This model is not well-suited to situations where the specific class of the transfer may determine whether a use was permissible or not. It is substantially simpler than some other provenance ontologies[11], and is tailored specifically to annotate XML documents.

Modelling provenance in the method proposed here involves adding a significant amount of data per fragment to a document’s DOM tree. Because of this, the model is best suited to annotate the quotation of significant amounts of data; while the system can handle quotations of any size, the amount of markup needed to describe the fields described is on the order of a paragraph of text. As a result, this level of metadata is too heavyweight to use to annotate single-character or single-word corrections.

A final tradeoff is between semantic simplicity and power. The semantic model described here is very simple; all provenance data attached to ancestor nodes represent a chain of copies starting at the current node and ending up at the topmost Document Fragment node. This method works well for certain purposes, but breaks down once we allow users to paste content into arbitrary portions of the document.

4.5 Implementation

One can use the functionality in the ped.js API to manipulate Provenance-Embedded Documents. This API is implemented in JavaScript on top of RDFa[14]. It includes functions for extracting the provenance structure of documents, functions for presenting provenance information to a user viewing a document, and functions for performing operations over all of the fragments in a composite document.
isDocumentFragment(node)  Returns True if the given DOM node is a Document Fragment.

getContainingFragment(node)  Takes the node parameter and determines the first relevant Document Fragment. This Document Fragment is the one associated with the original content’s creation.

getAllContainingFragments(node)  Takes the given node and recurses up through its DOM parents. Returns a list of all parents that are Document Fragments. According to the specified semantics, this is a list of all of the documents this content has resided in, in chronological order.

generalGetTriples(subject, predicate, object)  This call accesses the RDF triple store, looking for triples of the form \(< subject, predicate, object >\). In addition, this call treats unspecified parameters as wildcards, enabling us to, for example, search the triple store for all triples of the form \(< *,prov:rights,* >\).
Chapter 5

Copy and Paste Operations

When creating documents from fragments of other documents, care must be taken to ensure that the documents created have proper provenance information. When data is copied out of source documents, all of the relevant provenance assertions from about the copied data must be preserved. Also, the copy operation itself must be documented in the markup of the copied content. Finally, the insertion of the copied content into the new document must be consistent with the semantics of Document Fragments.

5.1 Selection Difficulties

A copy operation that preserves embedded provenance information is substantially complicated by the nature of selecting sections of XHTML documents. Contiguous ranges of text in a web browser selection often do not correspond to a neatly contained section of the DOM tree. Thus, the node in which the selection starts may be buried many nodes deep in the DOM tree, and may have many ancestors which may contain
relevant information. All of these ancestors will need to be copied as well. To solve these problems, the Copy algorithm walks the DOM tree in a way that guarantees all relevant nodes are copied.

A related problem is ensuring that all relevant RDF assertions are copied along with the selected content. RDFa does not provide any guarantees about the locations of assertions within a document, so a selection of a range of text may not contain relevant assertions which need to be preserved. As a result, any copy operation needs to query the source document’s triple store and manually copy all relevant assertions into the selected segment.

5.2 Copy Algorithm

In order to mitigate these problems, we must walk the DOM tree, ignoring all nodes that occur before the selection start, and all nodes that occur after the selection’s end. When we encounter a node which is annotated with provenance information, we must export all information regarding this node from the triple store into the copied DOM fragment.

Before describing the algorithm in full, it will be useful to define some terms. The “start node” is the node that contains the start point of the selection. The “end node” is the node that contains the end point of the selection. The “common node” is the deepest node in the DOM tree that contains both the start node and the end node. A 'DocumentFragment node’ is a DOM node annotated with provenance RDFa. The 'copied node’ is the toplevel DOM node representing all copied content. The 'target node’ is the node that the top level loop (or subroutine) writes to.
There are also several helper procedures which need to be defined before the algorithm is described. \textsc{Shallow-Copy}(node) produces a copy of the parameter DOM node, but does not copy any of the parameter’s child nodes. \textsc{Deep-Copy}(node) copies the toplevel node and then recursively copies all child nodes. \textsc{Common-Node}(startNode, endNode) takes the start and end nodes of a selection, and returns the nearest common ancestor. These operations correspond to \texttt{element::cloneNode(false)}, \texttt{element::cloneNode(true)}, and \texttt{range::commonAncestorContainer} in the FireFox API [15].

\textsc{Copy-DF}(node) performs a shallow copy of the node, and then queries the RDFa triple store for metadata concerning the parameter node. It then marks up the copied node with copies of all the RDFa that applies to the parameter node.

The algorithm takes a selection, specified by the start node and the end node. The algorithm finds the common node, and recurses up through the common node’s parents until it reaches the top level of the document. It wraps the newly-copied node in any layers of provenance metadata that the common node was already wrapped in, thus preserving the nested semantics. Next, it wraps the new node in a new layer of provenance metadata that describes the copy being performed. Finally, the algorithm walks the section of the tree which is selected in a manner that preserves any relevant metadata while cutting away unselected material.

To do this, we note that the selection of the text partitions each level of the DOM tree into five segments, the segment before the selection begins, the segment containing the node which the selection begins in, the segment containing all nodes that are completely inside the selection, the segment containing the single node in which the selection ends, and the segment containing any nodes which are entirely
after the endpoint of the selection. ¹

In order to copy only the relevant parts of the tree, the copy algorithm needs to treat each of these segments differently. First, the algorithm iterates through the toplevel node’s children until it finds the node of the subtree which contains of the start of the selection (even if the start of the selection is arbitrarily deep). It does not add these nodes to the selection’s subtree, effectively deleting them. Next, the algorithm special-cases the node of the subtree containing the start of the selection. After this, the algorithm recursively copies all nodes completely contained by the selection into the copied node. When the end of the selection is encountered, the algorithm special-cases this node as well. Finally, the algorithm terminates without copying anything after the selection ends, effectively deleting those nodes as well.

By separating the process of copying the node which contains the start of the selection, we can make simplifying assumptions. The nodes in this particular subtree can be split into three sections; those before the start of the selection, the node of the subtree containing the start of the selection, and the nodes after the start of the selection. ² The procedure thus iterates until it finds the node containing the start of the selection. All nodes before this are discarded. The procedure recurses on the node containing the subtree of the selection start, and deep copies all nodes after the node containing the selection start. The base case for recursion is if the procedure is called on the node that actually is the node of the start of the selection. If this is the case, the node is copied into the selection. If it is a text node, only the selected text is copied.

¹This is in the general case; any situations in which this is not the case (the selection spans an entire document, the selection is entirely contained within one node, etc. are special-cased.)

²This is a guaranteed invariant, because Copy only calls this procedure when it is the case, and the operation of Copy-start preserves the invariant.
Copying the node containing the end node is similar, except it is the search for the start node we skip. The procedure iterates over all nodes, deep copying each of them. When it finds the subtree of the end node, it recurses. It ignores all nodes after the subtree of the end of the selection.

```plaintext
COPY(startNode, endNode)
1 commonNode ← COMMON-NODE(startNode, endNode)
2 newNode ← COPY-PARENTS(commonNode)
3 for i ← 0 to commonNode.childNodes.length
4   do if cNode.contains(startNode)
5      then COPY-START(commonNode.childNodes[i], startNode)
6      BREAK
7 for i ← i to commonNode.childNodes.length
8   do if cNode.contains(endNode)
9      then COPY-END(commonNode.childNodes[i], endNode)
10 else DEEP-COPY(commonNode.childNodes[i], newNode)

COPY-PARENTS(node)
1 newNode ← COPY-DOCFRAG(node)
2 while node.parentNode
3   do node ← node.parentNode
4      childNew ← newNode
5      newNode ← COPY-DOCFRAG(node)
6      newNode.appendChild(childNew)
7     childNew ← newNode
8     newNode ← document.createElement('div')
9     newNode.appendChild(childNew)
10 Add elements that have relevant metadata to newNode
11 return newNode

COPY-START(node, startNode)
1 newNode ← COPY-DOCFRAG(node)
2 for i ← 0 to node.childNodes.length
3   do if node.childNodes[i].contains(startNode)
4      then newNode.appendChild(COPY-START(node.childNodes[i], startNode))
5      BREAK
6 return newNode
```
A man, a plan, a canal, Panama.

Figure 5-1: The original example document.

COPY-END(node, endNode)
1  newNode ← COPY-DOCFRAG(node)
2  for i ← 0 to node.childNodes.length
3      do newNode.appendChild(DEEP-COPY(node.childNodes[i]))
4      if node.childNodes[i].contains(endNode)
5      then newNode.appendChild(COPY-END(node.childNodes[i], endNode))
6          BREAK
7  return newNode

5.3 Example of Operation

In these examples, we are investigating the proper operation of the copy procedure on a sample XHTML document. The additional complexity in this section’s sample document is necessary to demonstrate corner cases that do not arise when examining the main scenario’s document. The sample document which we will be excerpting from contains the phrase “A man, a plan, a canal, Panama.” It is split into DIVs of the structure (((A man,)(a plan,))((a canal,)(Panama.))). The outer and first inner DIVs are annotated with provenance information.
Example 1  In the first example, the selection only exists in the second text node of the first DIV. Thus, we start from this node and recurse upwards until we find an annotated node (the first DIV). The copied node is thus this DIV, with the text node trimmed to match the selection. Note that we entirely omit the first text node, as it does not contain any of the selected text.

Example 2  In this example, our selection spans two nodes, so we start from the common node (the second DIV). Starting from this point, we recurse upwards until we find an annotated node (the outer DIV). We copy the body and the DIV into the target node. We then trim the text nodes to match the selection, and copy both of them into the DIV.

Example 3  In this example, our selection is of a small portion of text, but this portion of text spans a key boundary in the document. Thus, the common node is the outer DIV tag. Since this tag is annotated, we do not need to recurse upwards. We next go downwards from the common node, discarding any nodes outside of the
Figure 5-4: An annotated selection spanning both DIVs.

A man, a plan, a canal, Panama.

...selection range. This is why the first node of the first DIV and the second node of the second DIV are discarded.

5.4 Paste Operation

By encapsulating all of the provenance information in the copied node, we have made the copied content self contained; all relevant information regarding the provenance of a quotation is contained inline with the quote’s XHTML. As a result, the “paste” operation simply inserts the copied XHTML into the document at the specified location.

This implementation of the “Paste” operation is limited to inserting copied XHTML in the top level of the document; we cannot insert copied XHTML inside existing Document Fragments. This is because of the semantics specified above; any copies associated with the parent DOM nodes of content have happened to that content. There are two methods of enabling pastes inside existing Document Fragments, but both are beyond the scope of this thesis. One is to “split” any existing Document Fragments we are pasting into in two at the point of the paste, thus effectively preserving the paste into the toplevel restriction. The second is to timestamp all Document Fragments with the time of creation, and extend the semantics such that only more
recent copies apply.
Chapter 6

License Generation

The license generator takes an annotated composite document as input, and produces a list of possible licenses that the composite document may be relicensed under. Doing this requires a detailed breakdown of the components of the document and license information about each component. This information is specified by the DocumentFragment ontology as described above. In order to accurately present licensing information for the composite document, the verifier must have a method of computing acceptable licensing terms from the aggregation of individual assertions about component document fragments. In order to do this, the code must extract the appropriate metadata, combine it, and calculate the possible options available to the end user.

Applicable License Terms For the purposes of producing licenses for aggregate documents, we will require that all licensing assertions in Document Fragments must be consistent, even if the assertion is associated was made by an excerpter, rather than the author of original content. The line between creating new content and excerpting
existing content is very fine, and difficult to determine algorithmically. The act of selecting and combining several works may qualify as a creative act in itself, so we will consider and combine all licensing assertions contained in all Document Fragments of the composite document.

In other situations, it may be very useful to track the exact sequence of copy operations that resulted in a document fragment ending up in its final location, even if there was no creation involved in the copy operation. For example, a full audit trail would be useful in determining where a hypothetical infringement first occurred, and determining if said infringement was intentional.

For example, in the following document, each assertion (CC-BY, CC-BY-NC, and CC-BY-NC-SA) will be aggregated into the final result of CC-BY-NC-SA.

```html
...<meta property='cc:license' content='.../by-nc-sa/2.5'/>
...
<div class='prov:DocumentFragment'>
 <meta property='prov:rights' content='.../by-nc/2.5'/>
 ...
 <div class='prov:DocumentFragment'>
  <meta property='prov:rights' content='.../by/2.5'/>
 ...
</div>
</div>
```

Combining Licensing Terms A key component of the Creative Commons model is that the licenses form a partial ordering; simply by looking at the license of a current work, we can determine all possible terms it may be relicensed under. [16]

I have used this partial ordering to create an automated Creative Commons reasoning engine. Implemented in JavaScript, it can take arbitrary numbers of Creative
Commons assertions and aggregate them into a single license that would be appropriate for distributing a composite work. If there is a fundamental conflict in licensing terms, the aggregator reports that the aggregation may not be redistributed under any license.

```
AGGREGATE(licenses)
1  noPurposes ← false
2  nc ← false
3  sa ← false
4  for each l in licenses
5    do if l.sa
6      then sa ← true
7      if !l.nc ∧ nc
8      then noPurposes ← true
9      if this.sa
10     then if l.nc∧nc
11     then noPurposes ← true
12
13    else if sa
14      then if l.nc∧nc
15      then noPurposes ← true
16      this.nc ← this.nc ∨ lic.nc
17
18  return noPurposes, nc, sa
```
Presenting Licensing Terms  Given that the licenses form a partial ordering, all the verifier has to do is present the end user with the most permissive license, and let the end user make the decision of how much further to restrict the composite document. For example, if the document contains CC-BY and CC-BY-NC fragments, as shown in Scenario A, the tool can present the user with a CC-BY-NC license. The user can then choose to either retain that license, or move to a CC-BY-NC-SA license.

Going Further  Embedded provenance information facilitates performing computations about a composite document. By including information about the purposes content may be used for in the DocumentFragment class, we can create a framework for computing these purposes for composite documents. For example, combining this framework with a Creative Commons reasoning engine produced the Creative Commons verifier. This idea is not limited to Creative Commons licenses, however. One could also combine it with a reasoning engine which combines security clearances, to produce policy-aware tools that can calculate the appropriate level of data classification. One could combine it with a tool that is aware of assertions regarding data reuse and create a tool that checks for violations of citizens’ privacy. [17]
Chapter 7

Outcome

As we have now developed several important technological components of inline document provenance, let us revisit the core scenario, and see how using the provenance ontology and associated tools allows us to make automated policy calculations.

7.1 Source 1

The professor begins, as before, by visiting the web site where the paper is located. Because the author wants his material to be reused under Creative Commons licenses, he has included metadata to this effect.

```html
...  
<meta property='dc:author' content='John Doe'/>  
<meta property='cc:license' content='.../by/2.5'/>  
...  
Facebook is evil.  
...  
```
The "copy" operation starts at the text node containing the selected text. It then walks up the chain of ancestors, adding any provenance information to the copied data. In this case, the only provenance information resides in the global metadata. A new DocumentFragment RDF node is created, and populated with this metadata. This markup is added to the clipboard.

```
<div class='prov:DocumentFragment'>
  <span property='prov:sourceURI' content='.../paper.html'/>
  <span property='prov:license' content='.../by/2.5'/>  
  Facebook is evil.
</div>
```

The professor then creates a blank document, and pastes in the copied markup.

### 7.2 Source 2

The blog posting is annotated in the same way as the paper, so the source document has the following markup:

```
...
  <meta property='dc:author' content='Richard Doe'/>
  <meta property='cc:license' content='.../by-nc/2.5'/>
...
  Facebook is a necessary evil.
```

The second copy operation acts as described above, yielding the following copied XHTML:
7.3 License Calculation

After the professor pastes the second excerpt into his template, the composite document has the following structure:

...<meta property='author' content='Richard Doe'/>
...

Now, let us juxtapose two opposing views on Facebook:
<div class='prov:DocumentFragment'>
  <span property='prov:sourceURI' content='.../paper.html'/>
  <span property='prov:license' content='.../by/2.5'/>
  Facebook is evil.
</div>
<div class='prov:DocumentFragment'>
  <span property='prov:sourceURI' content='.../blog.html'/>
  <span property='prov:license' content='.../by-nc/2.5'/>
  Facebook is a necessary evil.
</div>
...

The professor next clicks the “verify” bookmarklet on his toolbar. This tool queries the RDFa database for all triples of the form < *, prov:license, * >, and receives two results. The verification bookmarklet aggregates the objects of these triples, combining CC-BY and CC-BY-NC. This yields a result of CC-BY-NC, which is presented to the end user.
7.4 Conclusion

7.4.1 Future Work

The applications of inline provenance metadata extend far beyond copyright. Inline metadata will be useful in any situation that requires after-the-fact auditing to accomplish policy goals. For example, the Transparent Accountable Datamining initiative [17] is currently investigating the use of auditing to increase accountability in the government use of personal information. In the future, the government could tag all data with information about its sources; this would allow policymakers to distinguish between data acquired via the Patriot Act and data acquired via normal police investigations.

The Privacy Act creates an environment in which provenance tracking is extremely useful. The Privacy Act restricts government use of data to the purposes it was originally intended for. This maps well to the license field in the copyright scenario. Data could be tagged at the time of collection with an RDF list of allowable use purposes. In addition, the "chain of custody" of data is also important for determining if data was used for a disallowed purpose. The nesting semantics of Document Fragments could be used in this situation to establish an audit trail. Later, an automated reasoner could combine the chain of custody with the allowable purposes to calculate policy results. The Transparent Accountable Datamining Initiative [17] has investigated this problem domain extensively.

The Fair Credit Reporting Act is another example of a large system which needs to be very concerned about the use of data. For example, medical data may not be used to make decisions regarding employment, credit, or insurance transactions. [18]
By tagging this data proactively, the credit industry could limit their liability from lawsuits regarding the improper use of data.

7.4.2 Contributions

In this paper, I have described several tools that enable programs to aid end users in making policy decisions:

- A document format that can represent information about the sources of its content.

- A method of excerpting from these documents that allows programs to trace the excerpt back to the source.

- A Creative Commons reasoning engine.

- A tool that uses these components to recommend permissible licenses for a composite document.
Appendix A

Creative Commons Reasoner

```javascript
if(typeof CCCombiner == 'undefined'){
    CCCombiner = new Object();
}

CCCombiner.CCLicense = function(uri){
    this.noPurposes = false;
    if(uri == "http://creativecommons.org/licenses/by/2.5/"){
        this.nc = false;
        this.sa = false;
    } else if (uri == "http://creativecommons.org/licenses/by-nc/2.5/"){
        this.nc = true;
        this.sa = false;
    } else if (uri == "http://creativecommons.org/licenses/by-sa/2.5/"){
        this.nc = false;
        this.sa = true;
    } else if (uri == "http://creativecommons.org/licenses/by-nc-sa/2.5/"){
        this.nc = true;
        this.sa = true;
    } else {
        alert(uri);
        throw RangeError;
    }
}

CCCombiner.CCLicense.prototype.combine = function(lic){
    if(lic.sa){
        this.sa = true;
    }
```
if(lic.nc == false && this.nc == true){
    this.noPurposes = true;
}
if(this.sa){
    if(lic.nc == true && this.nc == false){
        this.noPurposes = true;
    }
}
else if(this.sa){
    if(lic.nc == true && this.nc == false){
        this.noPurposes = true;
    }
}
this.nc = this.nc || lic.nc;
}

CCCombiner.CCLicense.prototype.toURI = function(){
    if (this.noPurposes == true){
        return "No Allowed Purposes";
    }
    if(this.nc == false && this.sa == false){
        return "http://creativecommons.org/licenses/by/2.5/";
    } else if (this.nc == true && this.sa == false){
        return "http://creativecommons.org/licenses/by-nc/2.5/";
    } else if (this.nc == false && this.sa == true){
        return "http://creativecommons.org/licenses/by-sa/2.5/";
    } else if (this.nc == true && this.sa == true){
        return "http://creativecommons.org/licenses/by-nc-sa/2.5/";
    }
    return "Error";
}
/**
 * Creative Commons Checker
 * By Harvey Jones <harveyj@mit.edu>
 * Modified from Simple CC license thing
 * By Ben Adida - ben@mit.edu
 * licensed under GPL v2
 */

// Loading the PED javascript.
if (typeof(__PED_BASE) == 'undefined'){
    __PED_BASE = 'http://people.csail.mit.edu/harveyj/';
}

var PED = new Object();
PED.url = __PED_BASE + 'ped.js';
PED.load = function(){
    var s2 = document.createElement("script");
    s2.type = 'text/javascript';
    s2.src = __PED_BASE + 'cc-combiner.js';
    document.getElementsByTagName('head')[0].appendChild(s2);

    var s = document.createElement("script");
    s.type = 'text/javascript';
    s.src = PED.url;
    document.getElementsByTagName('head')[0].appendChild(s);
}
PED.load();

PED CALLBACK DONE LOADING = function() {
  var lic = new CCCombiner.CCLicense("http://creativecommons.org/licenses/by/2.5/");

  var cc = new RDFA.Namespace('cc', 'http://web.resource.org/cc');
  var cclicense = new RDFA.CURIE(cc, "license");

  triples = PED.generalGetTriples(null, cclicense, null);
  for(prop in triples){
    lic.combine(new CCCombiner.CCLicense(triples[prop].object));
  }

  triples = PED.generalGetTriples(null, PED.allowedPurposes, null);
  for(prop in triples){
    lic.combine(new CCCombiner.CCLicense(triples[prop].object));
  }
  alert("The most permissive license you could redistribute this content under is " + lic.toURI());
}

50
Appendix C

PED.js

// Set up RDFa
if (typeof(__RDFA_BASE) == 'undefined'){
   __RDFA_BASE = 'http://people.csail.mit.edu/harveyj/';
}

var RDFA = new Object();
RDFA.url = __RDFA_BASE + 'rdfa.js';
RDFA.load = function(){
   var s = document.createElement("script");
   s.type = 'text/javascript';
   s.src = RDFA.url;
   document.getElementsByTagName('head')[0].appendChild(s);
}

RDFA.load();

RDFA.CALLBACK_DONELOADING = function() {
   // Set up PED namespace
   PED.callbacks = new Array();

   PED.prov = new RDFA.Namespace('prov', 'http://www.w3.org/2007/03/provenance');
   PED.rdf = new RDFA.Namespace('prov', 'http://www.w3.org/1999/02/22-rdf-syntax-ns');
   PED.cc = new RDFA.Namespace('cc', 'http://web.resource.org/cc');
   PED.dc = new RDFA.Namespace('dc',

   51
'http://purl.org/dc/elements/1.1/');

PED.rdfType = new RDFA.CURIE(PED.rdf, "type");
PED.documentFragment = new RDFA.CURIE(PED.prov, 'DocumentFragment');
PED.creator = new RDFA.CURIE(PED.prov,'creator');
PED.copier = new RDFA.CURIE(PED.prov,'copier');
PED.creationTime = new RDFA.CURIE(PED.prov,'creationTime');
PED.copyTime = new RDFA.CURIE(PED.prov,'copyTime');
PED.sourceURI = new RDFA.CURIE(PED.prov,'sourceURI');
PED.allowedPurposes = new RDFA.CURIE(PED.prov, 'allowedPurposes');

PED.license = new RDFA.CURIE(PED.cc, 'license');

PED.creator = new RDFA.CURIE(PED.dc, 'creator');
PED.date = new RDFA.CURIE(PED.dc, 'date');

RDFA.parse();
}

RDFA.CALLBACK_DONE_PARSING = function() {
    PED.CALLBACK_DONE_LOADING();
}

// Set up PED object
if(typeof PED == 'undefined'){
    var PED = new Object();
}

// dummy callbacks in case they're not defined
if (!PED.CALLBACK_DONE_LOADING)
    PED.CALLBACK_DONE_LOADING = function() {};

// Input: URI
// Returns: String of .ttl representation of that DOM node’s RDFA
PED.nodeToTurtle = function(node){
    if(node._RDFASUBJECT){
        return PED.toTurtle(node._RDFASUBJECT);
    } else {
        return "";
    }
// Input: Subject
// Returns: String of .ttl representation of that subject
PED.toTurtle = function(subj){
    if(PED.isBNodeURI(subj)){
        ttl = "[" + PED.contentsToTurtle(subj) + "]";
    } else {
        subj = RDFA.CURIE.getAbsoluteURI(subj);
        ttl = "" + subj + ">";
        ttl += PED.contentsToTurtle(subj);
    }
    return ttl + "\n."
}

PED.contentsToTurtle = function(subj){
    ttl = ""
    subj = RDFA.CURIE.getAbsoluteURI(subj);
    for(var prop in RDFA.triples[subj]){  
        ttl += " " + prop + " ";
        triples = RDFA.triples[subj][prop];
        for(i in triples){
            if(triples[i].object_literal_p){
                ttl += ", " + triples[i].object + "\n";
            } else if(PED.isBNodeURI(triples[i].object)){
                ttl += "[" + PED.contentsToTurtle(triples[i].object) + "] \n";
            } else {
                ttl += "" + triples[i].object + ">";
            }
        }
    }
    return ttl;
}

// Input: URI
// Returns: HTML table representation of document fragment located at URI
PED.subjToTable = function(subj){
    tbl = document.createElement("table");
    tbl.appendChild(PED.makeRow("Creator: ", PED.getField(subj, PED.creator)));
    for(var prop in RDFA.triples[subj]){  
        triples = RDFA.triples[subj][prop];
        for(i in triples){
            if(triples[i].object_literal_p){
                triples[i].object += "\n";
            } else if(PED.isBNodeURI(triples[i].object)){
                triples[i].object += "[" + PED.contentsToTurtle(triples[i].object) + "] \n";
            } else {
                triples[i].object += ">";
            }
        }
    }
    return tbl;
}
tbl.appendChild(PED.makeRow("Copier: ", PED.getField(subj, PED.copier)));
tbl.appendChild(PED.makeRow("Creation Time: ", PED.getField(subj, PED.creationTime)));
tbl.appendChild(PED.makeRow("Copy Time: ", PED.getField(subj, PED.copyTime)));
tbl.appendChild(PED.makeRow("Source URI: ", PED.getField(subj, PED.sourceURI)));
tbl.appendChild(PED.makeRow("Allowed Purposes: ", PED.getField(subj, PED.allowedPurposes)));

    return tbl;
}
function PED.getAllContainingFragments(node) {
  var frags = new Array();
  node = node.parentNode;
  while (node) {
    // Go up DOM Tree
    if (PED.isDocumentFragment(node)) {
      frags.push(node);
    }
    node = node.parentNode;
  }
  return frags;
}

function PED.getAllFragments(node) {
  dftn = new PED.DocumentFragmentTreeNode();
  dftn.childNodes = PED.getAllChildFragments(node);

  if (PED.isDocumentFragment(node)) {
    dftn.dfNode = node;
  } else {
    dftn.dfNode = null;
  }
  return dftn;
}

function PED.getAllChildFragments(node) {
  var frags = new Array();
  for (var i = 0; i < node.childNodes.length; i++) {
    if (PED.isDocumentFragment(node.childNodes[i])) {
      var childDFTN = new PED.DocumentFragmentTreeNode();
      childDFTN.dfNode = node.childNodes[i];
      childDFTN.childNodes = PED.getAllChildFragments(node.childNodes[i]);
      frags.push(childDFTN);
    } else {
      // Additional code here
    }
  }
  return frags;
}
result = PED.getAllChildFragments(node.childNodes[i]);
for(var j = 0; j < result.length; j++){
frags = frags.concat(result[j]);
}
return frags;
}

// Tree abstraction that contains DocumentFragments
PED.DocumentFragmentTreeNode = function(){
    this.dfNode = null;
    this.childNodes = new Array();
}

// Input: DocumentFragmentTreeNode
// Output: String representing the tree associated with this node
PED.treeToString = function(dftn){
    var retStr;
    if(dftn.dfNode == null){
        retStr = "";
        for(var i = 0; i < dftn.childNodes.length; i++){
            retStr += PED.treeToString(dftn.childNodes[i]) + "\n";
        }
        return retStr;
    } else {
        retStr = "";
        retStr +=" DocumentFragment";
        for(var i = 0; i < dftn.childNodes.length; i++){
            retStr += PED.treeToString(dftn.childNodes[i]) + "\n";
        }
        return retStr + " }\n";
    }
}

// Input: DocumentFragmentTreeNode
// Output: Table representing the tree associated with this node
PED.treeToTable = function(dftn){
    var ret = document.createElement("div");
if(dftn.dfNode){
    subj = dftn.dfNode._RDFASUBJECT;
    tbl = PED.subjToTable(subj);
} else {
    tbl = document.createElement("table");
}

for(var i = 0; i < dftn.childNodes.length; i++){
    b = document.createElement("input");
    id = dftn.childNodes[i].dfNode.id;
    b.setAttribute("type", "button");
    b.setAttribute("value", id);
    b.setAttribute("onclick", PED.genCallback(id));
    d = document.createElement("div");
    d.setAttribute("id", id+"div");
    d.appendChild(b);
    ret.appendChild(PED.makeGenericRow("Contains", d));
}
ret.appendChild(tbl);
return ret;
}

PED.flattenTree = function(dftn){
    var nodes = new Array();
    nodes = nodes.concat(dftn.dfNode)
    for(i in dftn.childNodes){
        nodes = nodes.concat(PED.flattenTree(dftn.childNodes[i]));
    }
    return nodes;
}

// Input: Subject, predicate, object
// Output: All RDFa triples that match this pattern
// If subject, predicate, or object are undefined, treat as wildcard
PED.generalGetTriples = function(subject, predicate, object) {
    var trips = new Array();
    var subject = subject ? RDFA.CURIE.getAbsoluteURI(subject) : "undefined";
    var predicate = predicate ?
        RDFA.CURIE.getAbsoluteURI(predicate) : "undefined";
    var object = object ? RDFA.CURIE.getAbsoluteURI(object) : "undefined";
    for(i in trips){
        if(subject in this && predicate in this && object in this) {
            return true;
        }
    }
    return false;
}
for (subj in RDFA.triples) {
    if (subject === "undefined" || subj === subject) {
        for (pred in RDFA.triples[subj]) {
            if (predicate === "undefined" || pred === predicate) {
                for (i in RDFA.triples[subj][pred]) {
                    if (!object || RDFA.triples[subj][pred][i].object === object) {
                        trips.push(RDFA.triples[subj][pred][i]);
                    }
                }
            }
        }
    } else {
        return trips;
    }
}

/** Internal utility functions */
PED.makeSpanProperty = function(property, content) {
    var node = document.createElement("span");
    node.setAttribute("property", property);
    node.setAttribute("content", content);
    return node;
}

PED.makeRow = function(label, content) {
    return PED.makeGenericRow(label, document.createTextNode(content));
}

PED.makeGenericRow = function(label, content) {
    var row = document.createElement("tr");
    var labelCell = document.createElement("td");
    var contentCell = document.createElement("td");
    var labelNode = document.createTextNode(label);
    var contentNode = document.createTextNode(content);
    labelCell.appendChild(labelNode);
    contentCell.appendChild(contentNode);
    row.appendChild(labelCell);
    row.appendChild(contentCell);
    return row;
}
row.appendChild(labelCell);
row.appendChild(contentCell);

labelCell.appendChild(labelNode);
contentCell.appendChild(content);

return row;
}

// Gets the table associated with node with id
// Puts it into the right place
PED.genCallback = function(id){
    PED.callbacks[id] = function() {
        var divel = document.getElementById(id + "div");
        if(divel.childNodes.length == 1){
            var tbl = PED.treeToTable(PED.getAllFragments(document.getElementById(id)));
            divel.appendChild(tbl);
        } else {
            divel.removeChild(divel.childNodes[1]);
        }
    }
    return "PED.callbacks['" + id + "']();";
}

// Returns .ttl rep of subject's property
PED.getField = function(subj, prop){
    var triple = RDFA.getTriples(subj, prop);
    if(triple){
        if(triple[0].object_literal_p){
            return triple[0].object;
        } else {
            return PED.toTurtle(triple[0].object);
        }
    } else {
        return "(null)";
    }
}
PED.isBNodeURI = function(subj){
    if(typeof subj == "string"){
        return subj.substring(0, 3) == "[_:
            && subj.substring(subj.length-1, subj.length) == "]";
        } else {
            return false;
        }
    }
}

/*** Copy functionality */
PED.copySelection = function(){
    var sel = window.getSelection().getRangeAt(0);
    var node = PED.copy(sel);
    PED.copyPopup(node);
}

// Retrieves the creator of the current document.
PED.getDocumentCreator = function(){
    return PED.generalGetTriples(document.URL, PED.creator)[0].object
}

// Retrieves the CC license of the current document.
PED.getDocumentLicense = function(){
    return PED.generalGetTriples(document.URL, PED.license)[0].object
}

// Core copy functionality,
// Input: range
// Output: DOM node containing copied markup
PED.copy = function(range){
    var anc = range.commonAncestorContainer;
    var startNode = range.startContainer;
    var endNode = range.endContainer;
    var top;

    var newNode = document.createElement('div');
    newNode.setAttribute('about', 'dummy');
    newNode.setAttribute('class', 'prov:DocumentFragment');
    newNode.appendChild(PED.makeSpanProperty('prov:copier',

60
newNode.appendChild(PED.makeSpanProperty('prov:sourceURI',
document.URL));
newNode.appendChild(PED.makeSpanProperty('prov:creator',
PED.getDocumentCreator()));
newNode.appendChild(PED.makeSpanProperty('prov:allowedPurposes',
PED.getDocumentLicense()));

PED.dumpNS(newNode);

if(PED.isDocumentFragment(anc)){
    top = anc;
    for(var node = anc; node != null; node = node.parentNode){
        if(PED.isDocumentFragment(node)){
            newTop = PED.cloneDF(node);
            newTop.appendChild(top)
        }
    }
    top = PED.cloneDF(anc);
} else if(anc.nodeType == Node.TEXT_NODE){
    top = document.createElement('div');
} else {
    top = anc.cloneNode(false);
}

newNode.appendChild(top)

if(startNode == endNode){
    if(startNode.nodeType == Node.TEXT_NODE){
        sel = document.createTextNode(startNode.substringData(range.startOffset,
            range.endOffset - range.startOffset));
        top.appendChild(sel);
    } else {
        top.appendChild(startNode);
    }
    return newNode;
}

var i;
for(i = 0; i < anc.childNodes.length; i++){
if (PED.nodeContains(anc.childNodes[i], startNode)) {
    top.appendChild(PED.copyStart(anc.childNodes[i], range));
    break;
}
}

for (i = i + 1; i < anc.childNodes.length; i++) {
    if (PED.nodeContains(anc.childNodes[i], endNode)) {
        top.appendChild(PED.copyEnd(anc.childNodes[i], range))
    } else {
        top.appendChild(anc.childNodes[i].cloneNode(true));
    }
}
return newNode;
}

PED.dumpNS = function(node) {
    for (var i in RDFA.namespaces) {
        if (RDFA.namespaces[i].uri != document.location) {
            node.setAttribute("xmlns:" + RDFA.namespaces[i].prefix, RDFA.namespaces[i].uri);
        }
    }
}

PED.copyStart = function(node, range) {
    var startNode = range.startContainer;

    if (node == startNode) {
        if (node.nodeType == Node.TEXT_NODE) {
            return document.createTextNode(node.substringData(range.startOffset, node.length - range.startOffset))
        } else {
            return node;
        }
    }
    newNode = PED.cloneDF(node);
    for (var i = 0; i < node.childNodes.length; i++) {
        if (PED.nodeContains(node.childNodes[i], startNode)) {

        }
    }
}
newNode.appendChild(PED.copyStart(node.childNodes[i], range));
break;
}
}
for(; i < node.childNodes.length; i++){
    newNode.appendChild(node.childNodes[i].cloneNode(true));
}
return newNode;
}

PED.copyEnd = function(node, range){
    var endNode = range.endContainer;
    if(node == endNode){
        if(node.nodeType == Node.TEXT_NODE){
            return document.createTextNode(node.substringData(0, range.endOffset));
        } else {
            return node;
        }
    }
    var newNode = PED.cloneDF(node);
    for(var i = 0; i < node.childNodes.length; i++){
        if(PED.nodeContains(node.childNodes[i], endNode)){
            newNode.appendChild(PED.copyEnd(node.childNodes[i], range));
            break;
        } else {
            newNode.appendChild(node.childNodes[i].cloneNode(true));
        }
    }
    return newNode;
}

PED.cloneDF = function(node){
    var newNode = node.cloneNode(false);
    if(node._RDFA_SUBJECT){
        PED.dumpRDFA(node._RDFA_SUBJECT, newNode);
    }
    return newNode;
}
// Dumps all RDFa assertions with subject subject into node
PED.dumpRDFA = function(subject, node){
    var triples = PED.generalGetTriples(subject, null, null);
    for(var i = 0; i < triples.length; i++){
        var pred = RDFA.CURIE.getAbsoluteURI(triples[i].predicate);
        var objTrips = PED.generalGetTriples(triples[i].object, null, null);
        if(pred == "http://www.w3.org/1999/02/22-rdf-syntax-ns#type"){
            node.setAttribute("class", triples[i].object);
        } else if(triples[i].object_literal_p){
            // Literal
            var el = document.createElement("span");
            el.setAttribute("content", triples[i].object);
            el.setAttribute("property", RDFA.CURIE.getAbsoluteURI(triples[i].predicate));
        } else if(objTrips == null){
            // URI
            var el = document.createElement('a');
            el.setAttribute("rel", RDFA.CURIE.getAbsoluteURI(triples[i].predicate));
            el.setAttribute("href", triples[i].object);
        } else {
            // BNode
            var el = document.createElement('span');
            PED.dumpRDFA(triples[i].object, el);
            el.setAttribute("rel", RDFA.CURIE.getAbsoluteURI(triples[i].predicate));
        }
        node.appendChild(el);
    }
}

// Returns true if container exists in node's subtree
PED.nodeContains = function(container, node){
    while(node){
        if(node == container){
            return true;
        }
        node = node.parentNode;
    }
    return false;
}
// Appends a textbox containing the copied markup onto the current document
PED.copyPopup = function(node){
    var br = document.createElement("br");
    var outbox = document.createElement("textarea");
    outbox.setAttribute('id', 'outbox');
    outbox.rows = 50;
    outbox.cols = 100;
    outbox.value = PED.serializeNode(node);
    document.body.appendChild(br);
    document.body.appendChild(outbox);
}

PED.serializeNode = function(node){
    var xmlDocument = document.implementation.createDocument('', 'span', null);
    xmlDocument.documentElement.appendChild(node);
    var xmlSerializer = new XMLSerializer();
    var markup = xmlSerializer.serializeToString(xmlDocument);
    return markup
}

PED.newWin = function(node){
    var win = window.open(""");
    alert("Here we go!");
    win.document.body.appendChild(node);
}
Bibliography


