Collaborative Proposal: SI2-SSI: Title Goes Here

PI:

Provenance is metadata that describes the history of a digital object: where it came from, how it came to be in its present state, who or what acted upon it, etc. I wouldn't have "etc." in the opening sentence. – Aaron Provenance is widely recognized as an essential means to document and cite computational It also adds value to data and experiments by providing a means to experimental reproducibility, facilitating precise tracking of data sets and computational artifacts, and enhancing data validation. There exists a community devoted to defining, formalizing, and standardizing provenance [**?**, **?**, **?**, **?**, **?**] as well as designing and developing systems that capture and record provenance [**?**, **?**, **?**, **?**, **?**]. Despite the needs of the scientific community and the activity of the provenance community, there has been only limited scientific impact from the advances in our understanding and management of provenance. At the Workshop on the Theory and Practice of Provenance held in 2011 [**?**] Aaron: Given that we are up to what, TAPP 13, and all the activity, it indeed seems odd that there has been limited scientific impact from provenance work.. Repsonse: I think the number refers to the year I believe TAPP 2014 is the 6th TAPP workshop, the community agreed that facilitating and encouraging adoption of provenance was one of the most significant challanges ahead [**?**].

Most existing provenance systems require users to adopt a particular tool set in order to benefit from provenance. For example, Starflow requires that you use the Python language [**?**], Trio requires that you use the Trio Database Management System [**?**], Kepler [**?**], Vistrails [**?**, **?**], Pasoa [**?**] and myriad other workflow engines all offer provenance support if you use a particular workflow engine, and PASS requires that you run a modified operating system [**?**]. This approach presents two problems to the adoption of provenance technology by scientists. First, it requires scientists to learn a new technology, since provenance collection is generally not available in the environments with which they are most familiar. Second, provenance collected from one technology cannot be easily integrated with provenance collected from a different technology even if they both collect provenance and are used in combination by the scientist to accomplish a task. Our goal is to bridge the gap between the "use-my-system" provenance solutions that exist and the reality of scientists who work in multiple environments, use whatever languages and tools they choose, develop their own computational tools, use a variety of extant data sources, and who would rather spend their time doing their scientific work than in learning a new technology.

The Harvard group, led by Seltzer, has many years of experience developing the provenance aware storage system (PASS) and integrating it with a number of other provenance solutions. The team from Mount Holyoke College (MHC), led by Lerner, has been working on two tools to support data provenance. The first is a library used to collect provenance from R [**?**], a language that is widely used by scientists in disciplines ranging from... to... to do data analysis. The second is a visualization tool that has been designed to be language-independent and has so far been used with provenance collected from R and from Little-JIL [**?**]. The MHC team has been working with the team from Harvard Forest (HF), led by Boose and Ellison, for over ten years, on studying the provenance needs of environmental scientists, working closely on the development and evaluation of the R library in the past year. This proposal combines these related efforts to bring a powerful collection of integrated provenance tools and capabilities into the hands of domain scientists in a manner that makes them easy to adopt and use. The end result will include a library for use by any software developer to collect and use provenance, a demonstration of that library in both the R and Little-JIL Barb: How important is Little-JIL to this proposal? languages, and multiple science applications MIS: Can we be more specific about the applications using these libraries. The different teams serve as platforms and testbeds for one another: The MHC team is a testbed and critical audience for the Harvard libraries, while the Harvard Forest team is a testbed and critical audience for the R and Little-JIL tools. The final test is the value that added provenance capabilities bring to the scientific results produced by the Harvard Forest team.

Accomplishing the goals of this proposal requires fundamental research in the following areas.

Reconciling provenance at multiple semantic granularities. Different layers of a software stack manipulate different kinds of objects. Databases manipulate tuples using relational algebraic operations; operating systems manipulate files and processes; languages operate on variables and operators; workflow engines manipulate objects and messages; and biologists experiment at scales ranging from base pairs and genes to planetary ecosystems. We will develop models and mechanisms that relate objects at different layers to facilitate capturing and querying their provenance.

Capturing detailed provenance at the level of a programming language. Most provenance is captured at a coarse grain, such as files or components in a workflow. Understanding how a file came into existence or what a workflow component does requires more detail. We will develop an approach that collects sufficient provenance data to provide this detail while scaling well as the volume of data increases.

Defining precisely aspects of provenance required for different use cases. Provenance can be used in myriad ways, and a system must capture different information depending on the intended use, which might vary during the project's lifecycle. We will formalize the relationship between what must be collected and the uses to which the provenance will be applied.

Making provenance accessible to scientists. Building a successful provenance system is more than a technology problem. It depends on collecting the right data and presenting it to the scientist in a way that is meaningful and useful. For example, the most common provenance query we encounter in talking with scientists is the ancestry query, "Where did this item come from?" This question is ill-posed: from what point in time should we return an answer? Understanding the context of this question and the data being queried will aid in determining the *right* provenance to return. We will work closely throughout this project to ensure that the software we develop addresses the real needs of real scientists.

To address these research questions, we will focus our efforts on the following technologies:

API Design: The Harvard team developed the Disclosed Provenance API as part of its layered provenance architecture. This provides a good starting point, but is currently insufficient to handle the interaction of related objects at different layers and the multi-granularity provenance required for this proposal.

RDataTracker library: The R library developed by the MHC and HF teams uses a combination of instrumentation by the scientist, introspection, and source code analysis. We plan to continue this work, simplifying the scientist's instrumentation task, improving the precision of the provenance collected, in a manner consistent with important aspects of R, such as its support for lazy evaluation and using the API developed by the Harvard team.

Scientific applications using provenance: The HF team will focus on the provenance use cases, using RDataTracker and other tools integrated with the Harvard library to help address the question of the value of provenance to the working scientists.

Intellectual Merit: This para seems off target to me. In my experience, the Intellectual Merit should focus on the core disciplinary (here CS) questions of interest at a fairly high (theory) level. "Synthesizing research" is not quite the right flavor. Resolving provenance relationships, semantics, and granularity seem more on target. So they should probably be the emphasis here. – Aaron The intellectual merit lies in bridging the gap between *provenance capture* and *use*. That is, while many existing systems capture provenance, there are few documented cases of provenance use, suggesting that fundamental research is required to bridge the gap between *data* and *information*; that is the major thrust of this work. Resolving the provenance relationships between different semantic levels and object granularities is the cornerstone of this work. As existing systems focus exclusively on a single semantic level or unit of granularity, this problem remains open today and is critical to enable the use of provenance in heterogeneous environments.

Broader Impact: The potential impact of this work is enormous. Although we focus on applications in environmental science I can provide data/platforms from 'omics (proteomics/transcriptomics) as well. I would think that breadth would be appreciated by reviewers. – Aaron; Absolutely! (MIS), the tools that we develop and insight we gain are general-purpose. By making the tools widely available, we expect that they can and will be adopted by scientists in other disciplines. Other colleagues at Harvard in physics and

imaging have also expressed interest in such capabilities. Seltzer's group at Harvard is currently engaged in a data mining project for which provenance is being recorded; we anticipate converting to the tools developed in this project. The R library has already been described at meetings and is gaining interest from users outside of Harvard Forest and we expect to present workshops at ecology meetings to promote its use.

This work also supports NSF's mission of broadening participation in computer science as Mount Holyoke is a women's college and therefore women undergraduates will be involved in this research. We also expect to fund students to work at Harvard Forest in the summer, where there has been a successful REU program for 25 years.

1 Provenance and Science: A Critical Need

As computational scientists, we need provenance every time we analyze data. A profound and tragic example of this need comes from the now-debunked study by Potti et al [**?**] (retraction [**?**]), reporting a gene expression signature that could guide the choice of chemotherapy for cancer patients. After this method was put into clinical practice, serious flaws were found in the analysis, and the test proved to be worthless. The Institute of Medicine (IOM) initiated an investigation to determine what went wrong, and what should be done to avoid such serious errors in the future. A member of the IOM investigating committee said, "many of the problems could have been avoided, or at least better identified, if there was a clear record of where and when different versions of the primary and derived research data existed". In other words, had a provenance system been in place, it may have prevented this tragedy.

Another example was discussed by Ellison et al (2006). The relationship between increasing atmospheric concentration of carbon dioxide (CO_2) and its relationship to rising global temperatures is well understood and not especially controversial ¡**cite IPCC 2014 report here**¿. Forests in the northern hemisphere remove large amounts of CO_2 from the atmosphere when deiduous trees leaf out in the spring and photosynthesize throughout the summer. In the winter, when leaves are shed and branches are bare, cellular respiration predominates, and CO_2 is released back to the atmosphere. The annual net ecosystem exchange (NEE) is the cumulative amount of CO_2 taken up or released by forests. Precise and accurate measurements of NEE are critical not only so researchers can understand whether forests can buffer anthropogenic activities and ameliorate change but also are used to set prices in regional, national, and international carbon markets and to define the value of forests for carbon offsets ¡**this needs a reference**¿.

NEE is measured using eddy covariance methods ¡**cite Barford et al. 2001**¿; measurements are taken at frequencies of 5-20Hz, averaged over 15-60 minute intervals, and integrated over an entire year. The accuracy of NEE estimates depend on atmospheric conditions, however, and as much as 75% of eddy-covariance data streams may be unreliable or missing; these data gaps are filled using a variety of investigator-dependent algorithms. Precise provencance is needed to identify data gaps and modeled values; to allow for propagation of uncertainty in annual integration and estimation; and to allow for models to be re-run as additional data accumulate, gap-filling algorithms change and improve, or intersite comparisons are attempted using common methods ¡**some citations useful**¿

Could keep the following or use similar examples. Points we may want to make are the following:

- Having just the source code may not be enough. Random numbers, user input, data downloaded from the Internet, etc. must be included.
- Source code describes what might happen. A detailed provenance trace describes what actually happened.
- A provenance trace may be easier to understand than source code.

The previous examples illustrate today's reality in computational science: vital provenance data is rarely even collected, let alone used to validate, authenticate, and reproduce scientific results. Most fundamentally, this gap identifes the need for a provenance system to track what computational scientists do: the programs run, their parameters, inputs and outputs. And we need to be able to do so in a domain scientist's *native* environment – forcing the use of particular workflow engines or systems makes the barrier to entry too high.

Beyond these basic capabilities, it would be helpful to be able to annotate data and workflows, textually, verbally (with audio), and visually (with video) to explain why experiments were conducted the way they were. It would also be beneficial to capture provenance when exploring the data, moving dynamically through the space of possible analyses following interesting leads as they emerge. Scientists frequently run predefined workflows repeatedly - hundreds or thousands of times - with different parameter settings to systematically explore the analysis space. Similarly, many analytical tools include a stochastic element, so it is useful to run a workflow repeatedly with the same parameters but different random number sequences, to characterize the distribution of results. Finally, scientists need a way to track changes in third party or external data and resources that are incorporated in analyses. It is vital to track whether changes in these inputs affect results.

Research questions:

The most fundamental research question we plan to address is how to bring provenance to the scientists, making it easily available and accessible to the scientists.

- What information should be collected?
- What should the scientist's role be in provenance collection?
- What is the best way to scale up?
- How will scientists use provenance once they have it?

Potential uses of provenance:

- Short-Term (months)
  - Develop and troubleshoot scripts
  - Examine derivation and use of particular data values
- Mid-Term (years)
  - Understand original analysis
  - Reproduce original analysis
  - Examine derivation and use of particular data values
  - Better use data in subsequent analyses
  - Better use analysis in other applications
- Long-Term (decades)
  - Same as mid-term, but replication may no longer be possible
  - Source code and data provenance will be key to understanding original analysis

In designing the project, we chose an initial set of applications and use cases that we believe can be solved and whose solutions will help us and others in the field move on to more challenging cases. The initial applications we've selected are: ???.

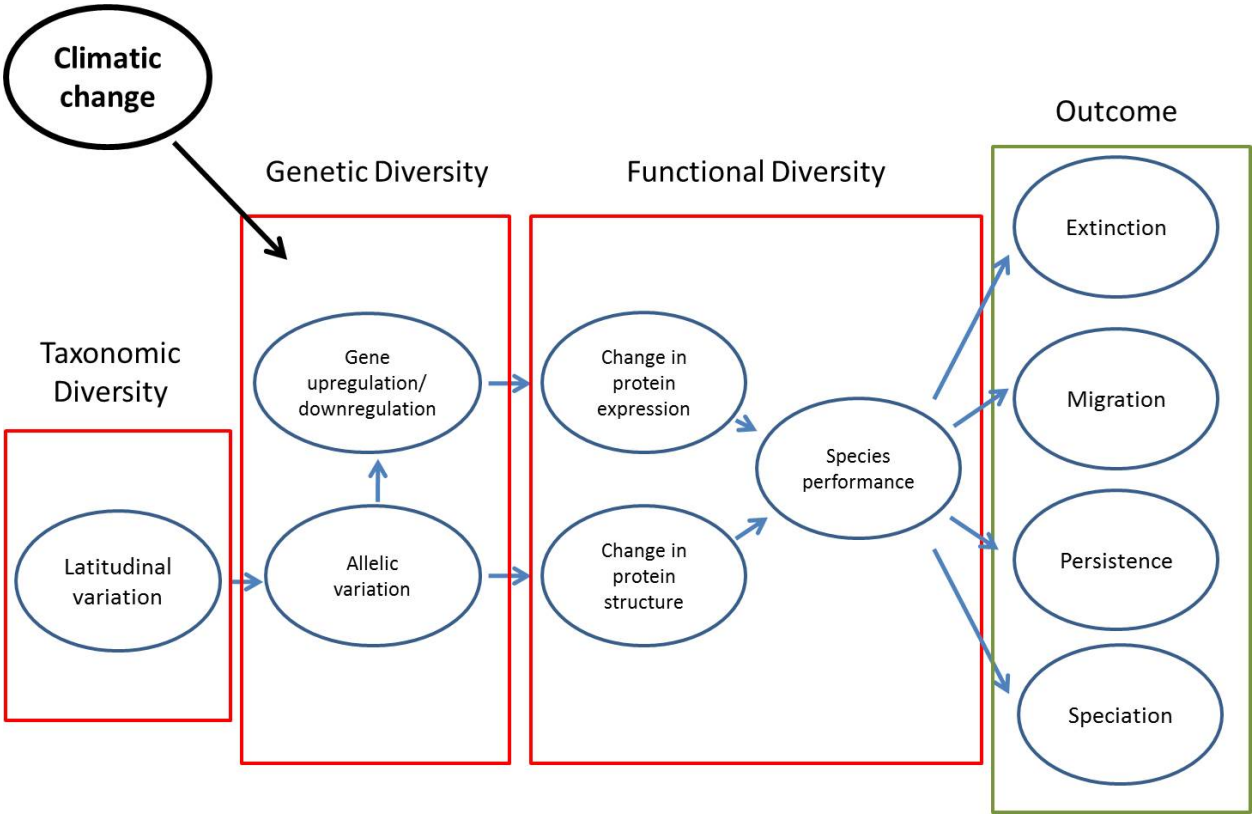

Figure 1: Caption goes here.

2 Provenance in Environmental Science

2.1 Use Case 1 - Modeling hydrology Emery/Barbara to write this one

2.2 Use Case 2 - Modeling forest carbon flux and estimating annual net ecosystem exchange

2.3 Use Case 3 - Using 'omics to forecast evolutionary responses to climate change

As the Earth's climate changes globally, habitats that are home to distinct species are changing too. The organisms that live and thrive in a particular location can respond to locally changing climates in one of four ways: they can move (migrate), stay put and acclimate (persist), go extinct, or evolve into new species. ¡**insert "climate cascade" figure 2.3 here**¿. Most research in this area has focused on how or when organisms might track climate change by migrating to new habitats (e.g., north or south, or up mountains, to remain in thermal equilibrium) ¡**general review of SDMs**¿, but new research suggests that genetic variability within a species may allow individual lineages to acclimate and persist in place. A research team co-led by co-PI Ellison and supported by NSF Dimensions of Biodiversity award 1136646 is using genomic, proteomic, and transcriptomic data collected from ants throughout eastern North America to study how evolution of heat-shock proteins and temperature-induced changes in protein expression and protein conformation affects performance of individual species in different climates.

Joint analyses of genomic, proteomic, and transcriptomic data are uncommon and present novel bioinformatic challenges that would benefit from provenance. At a minimum, each of these sets of 'omic data

(Gigabytes-to-Terabytes) must be assembled, checked, and annotated using standard methods, but different standards apply to different 'omes. Heat-shock proteins are conserved across animals and plants ¡**citation**¿, but ant 'omes are much more poorly studied than those of (e.g.) people, mice, or rats, and the data assembly pipeline requires much more "hands-on" work and (re)checking than a "standard" BLAST search for, say, a new human cancer gene. And completing the genome, proteome, or transcriptome of even a single species of ant is just the first step. Modeling protein conformations and their responses to different climate-change scenarios requires extensive custom coding (and debugging) to ensure reliable results. At each stage, collection of detailed provenance would ensure reproducibility of analysis, reliability of results, and information useful to track uncertainty through the process and to associate levels of uncertainty with the different potential outcomes for individual species in a changing climate.

3 Research Plan

Need to subdivide into 2 sections. One R-specific, one for the interoperability toolkit (currently here)

The team at Harvard is responsible for the computational research necessary to produce the software artifacts to be incorporated by ISB researchers as described above. While the Harvard team has extensive experience producing provenance solutions in tools, languages, and operating systems, their projects to date all require that users adopt a particular tool set or environment. We refer to that model of provenance provision as "bringing the users to the tools"; our goal is to bring tools to the users by producing a provenance toolkit, easily integrated into any desired platform. As described in the introduction, designing such a toolkit requires fundamental research in four areas: reconciling provenance at multiple semantic granularities, identifying what provenance is required for what capabilities, returning appropriate provenance in response to queries, and API design.

The first two subsections that follow present the foundational motivation, experience, and model for the research. We next present a brief architectural description of the provenance library. The subsequent four sections then outline our research approach to each of our four challenges.

3.1 Layering in PASS

The layered PASS architecture addresses three specific problems in provenance integration: data/provenance consistency, stacking, and cycles.

Data/provenance consistency allows a system to make concrete statements about the semantic consistency provenance provides. *Complete* provenance means that the provenance present in a system describes all the transformations applied to the data it describes. *Correct* provenance means that all the transformations present in the provenance were actually applied to the data. PASS ensures complete provenance via its *Disclosed Provenance API* (DPAPI), which transmits data and provenance atomically, and its use of write-ahead provenancing (WAP, akin to write-ahead logging), which guarantees that even after a crash, a description of every transformation to a piece of data is present in the persistent provenance store. PASS ensures correctness using hashing to identify cases where provenance was written to a persistent store, but the data described by that provenance has not yet made it to the persistent store and may have been lost in a failure. Using a transaction-like mechanism during recovery, PASS provides both completeness and correctness. We will retain WAP and an integrated data/provenance API in this work.

The DPAPI does not, however, capture the rich set of relationships among objects at different semantic levels, such as containment, encapsulation, specialization, and generalization. Therefore, one of the challenges to be addressed is to identify the requisite relationships and then develop a more general and flexible API to express those relationships, retaining the atomic flow of data and provenance and the WAP capabilities.

In building provenance-aware applications and components, we quickly realized that some components are both consumers and producers of data and therefore must consume, transmit, and produce

provenance as well. Thus, the provenance architecture must facilitate *stacking* such that a component can accept provenance from another component, augment that provenance, and then transmit it to yet another component. For example, a data pipeline that ingests a file containing both organism and gene data, transforming it into a per-organism and per-gene databases, must accept provenance about the file, retain that provenance, and produce provenance about the various tables and tuples it creates, linking the different objects correctly. Thus, like the DPAPI, the API we design here must stacking provenance: receiving, augmenting, and then transmitting provenance between software tools and layers. Doing so in a way that preserves key properties of provenance and facilitating semantically meaningful queries across software components remains an open problem.

Making connections between objects becomes more challenging in distributed environments where objects bear different names. The Second Provenance Challenge [**?**] identified object naming and name resolution as one of the most challenging aspects of provenance interoperability. Resolving such naming challenges is critical for avoiding provenance cycles. Provenance must form a directed acyclic graph. Since provenance represents ancestry, it is crucial that the graph is acyclic, or else it would suggest that multiple items were related both through ancestry and descendency relationships,which is problematic. However, systems that infer provenance from observed events (e.g., PASS) will produce cyclic graphs unless care is taken to avoid them. For example, a process tracking file-level provenance that reads and writes from/to the same file creates cyclic dependencies. Versioning objects is the obvious way to deal with cycles, and PASS has evaluated multiple versioning approaches [**?**], but none of them can adequately address the challenge that arises in avoiding cycles that can occur between objects created at different layers of software and different granularities.

Addressing this problem relies on developing solutions to the following challenges: we need to identify the range of relationships to express, design a means of expressing those relationships, and resolve names between the various layers. Furthermore, we need to develop a formal model of provenance that will let us make precise statements about the semantics of provenance and therefore define how provenance from different levels can and must interact. Current models [**?**, **?**, **?**, **?**] are insufficient to capture the broad range of relationsihps that we have identified to date.

3.2 A Graphical Model of Provenance

In the database world, provenance results only from the execution of queries. In this constrained world, provenance can be expressed in terms of two operators that form a semi-ring [**?**]. In the wild, provenance results from any computation and cannot be so cleanly expressed. Thus, we model provenance as a graph.

A provenance graph is a labeled graph whose nodes represent both objects and agents and whose edges represent relationships among those nodes. We will begin with a set of edges whose meaning is precisely defined and then extend the model to include arbitrary edges.

3.2.1 The Basic Model

This is different than how we talk about provenance graphs. We have several types of nodes that fall into 2 broad categories: operations and data. We also have 2 edge types: control flow and data flow. Our types could easily be attributes in your model, but I'm not sure what originator corresponds to in our view of provenance. Perhaps originator corresponds to an operation node? On the other hand, you propose a much richer set of edge types than we have. How would a richer set like this affect R provenance or improve its usefulness?

The model derives from our work integrating provenance systems, developing formal methods of representing security and privacy properties, and querying provenance graphs, but we anticipate extending the model as necessary. A provenance graph consists of vertices, attributes, labels and edges. Edges are directed and bear a single label; thus every edge has a source vertex, a label and a destination vertex.

Provenance Graph $G_P := (V_P, A_P, L_P, E_P)$

- Vertices V_P
- Attributes A_P
- Labels L_P
- Edges $E_P := \{src : V_P, lbl : L_P, dst : V_P\}$

A vertex has two or more attributes associated with it, where an attribute is a name:value pair. The two required attributes are *id* and *originator*, where *id* is an object identifier and *originator* uniquely identifies the agent responsible for creation of the vertex. Together, the *id* and *originator* form a unique key.

The basic model specifices a required set of labels and attributes that have semantic meaning with respect to graph construction and query semantics. The extended model makes it possible for any provenance-aware agent to specify its own additional labels and attributes, so long as it respects the semantics of the basic model.

The *attributes* defined in the basic model are:

- id: vertex identifier, unique within an originator
- originator: agent responsible for creation of a graph element
- name: a string used to describe the graph element
- time: the creation time of a graph element

The *labels* defined in the basic model are:

- input: the destination vertex is an input to the source vertex; in other words, the source depends upon the destination
- version: the destination vertex is a previous version of the source vertex
- contained: the source is contained within the destination; for example the destination might represent an entire file, while the source represents a piece of that file
- instantiates: the source is an instance of the object represented by the destination
- maps: the source is an alternate representation for the destination object
- controlled_by: the source is controlled by the destination

Figure 3.2.1 shows examples of several different edge labels.

3.2.2 The Query Model

In our model, any query on a directed acyclic graph constitutes a provenance query; however there are several queries that we find common across domains. We have developed a path-oriented query language (PQL) [**?**] that is sufficiently powerful to express such queries. The basic query model is that of traversal across edges, with optional matching on attributes. Rather than going into the details of our specific query language here (the interested reader can refer to our reference guide [**?**]), we instead outline those queries we have found most common and demonstrate how they are, in fact, straightforward graphical queries.

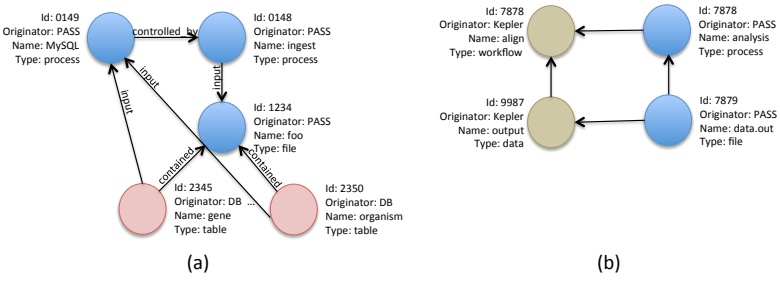

Figure 2: Edge type examples. Figure A depicts a data ingest process that reads an input data file and then spawns a database (MySQL) and inserts the data into relational tables in the database. Figure B depicts a process that is the instantiation of a workflow template. The actual file produced by the execution of the process maps to the one described by the workflow template.

The Ancestry Query: By far, the most common query we have encountered is the ancestry query, "From where did this object come?" This query is a transitive closure of the graph from a given node through all "ancestor nodes." In the basic model, for ancestry queries, we consider input edges, version edges, and controlled_by edges, however in any extended model a user or agent may specify additional edge labels (either from the basic model or from the extended model) as representing ancestry. The proper query result is still a transitive closure, just one along more edges.

Friend Queries: After shadowing scientific users in different fields, we found that users frequently wanted to identify objects (in our case, files) that had undergone processing similar to some other object. We call such queries "friend" queries. The specification of a friend query typically begins with an object in the provenance tree and its "immediately-relevant path." The immediately-relevant path may be specified manually or determined using the techniques described below in Section 3.6. Although most users wanted a simple path for such friend queries, we could also begin with any designated subtree. The chosen path is then abstracted to allow for different inputs undergoing identical processing. The query requests all paths matching the abstract path. In the data ingest use case described above, the query that asks for the friends of a particular data product should return all the data products produced by the same ingest process.

Repeated Trial Queries: Repeated trials are similar to friends. Once again, we begin with a specific object and its immediately relevant ancestry path. This time, we want all such paths beginning with the same input, experiencing "similar" transformations, where the transformations differ only in attributes or a computational module. For example, the ISB team frequently repeats the same analysis experimenting with alternate analysis modules. Given the output of such an analysis, a repeated trial query should return all the data outputs of the analysis, including those that used different parameters or different analysis modules.

3.3 Library Architecture

The provenance library consists of a set of modules that collect provenance, construct the graphical representation as described above, and persist the provenance to a backend database. As our goal is to support a variety of programming languages and database backends, the library includes a collection of language bindings and database adapters. Figure 3.3 shows the architecture and components of the Core Provenance Library.

3.4 Provenance Reconciliation

Provenance reconciliation is the process of integrating provenance contributed by different software agents. Most existing provenance systems operate at a single level of abstraction: the system call layer, a workflow specification, or the high-level constructs of a particular application. The provenance collectable in each of these layers is different, and all of it can be important. The research challenge lies in expressing the relationship among these different entities correctly and in such a way to facilitate queries both within each semantic level as well as across levels. Most current systems fail to account for the different levels of abstraction at which users need to reason about their data and processes, therefore these systems cannot integrate data provenance across layers and cannot answer questions that require such an integrated view.

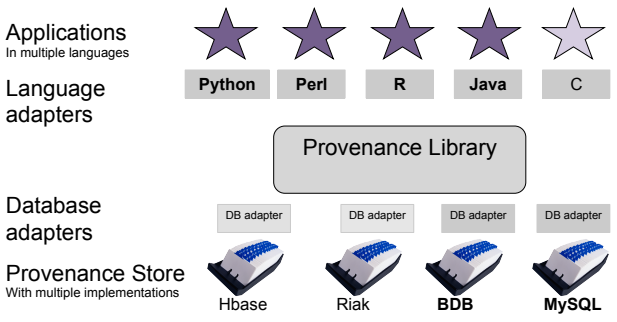

Figure 3: Core Provenance Library Architecture. The databases and language adapters shown in bold are ones that are needed for our applications.

There are two fundamental approaches to provenance reconciliation. One approach, where every software system interacts directly with every other software system in the manner specified by each system, produces an n^2 problem. The other approach requires that the software artifacts agree how to express provenance relationships and use a common API. We are adopting the second approach, because it is significantly more scalable and requiring the use of a common API allows applications to express the relationship between objects they manage and those managed by other software artifacts.

One of the key challenges in managing this interaction is properly accounting for versions. Although versioning has been fundamental to PASS since its inception, it is only more recently that others have recognized the criticality of versioning [**?**, **?**, **?**]. Versioning becomes increasingly complicated when different software layers are managing related items and independently creating versions. For example, consider the data ingest scenario. The objects that are obtained from external sources are frequently flat, structured files. Early in the ingest process we transform those flat files into database tables. The modules that manipulate the database will collect provenance on tables and will need to create versions of those tables and track relationships among specific versions. At the same time, we may download new versions of the source files. Should the modules manipulating database tables need to refer back to the original files, it is critical that we do so using appropriate version numbers. While manageable with only two layers of software, as the number of layers of software increases and the relationships among the entities being managed becomes more complex, the problem becomes more challenging. We will draw on our experience managing versions in PASS and layered applications to develop algorithms that ensure proper version handling.

3.5 Provenance in support of Use Cases

At the most recent Workshop on the Theory and Practice of Provenance [**?**], the participants identified myriad uses for provenance: increasing trust in an artifact, validating data currency, recovery, repeatability and exploration, annotation propagation, citation, specification of update semantics, debugging, schema integration, security auditing, data synchronization, self-adjusting computation, and documenting probabilistic data. Given this wide-ranging list of uses, it is no wonder that different systems choose to capture widely varying amounts of provenance. The challenge we will address is to provide the flexibility to capture provenance to address a wide range of use cases and to specify with precise models matching the provenance to collect with the use case. For example, in some cases, organizations need to track where their data goes [**?**], while in other cases, organizations are strictly prohibited from tracking such information [**?**, **?**, **?**]. Current approaches to provenance collection are haphazard in that there is no specification of what data must be collected to facilitate any particular use case.

We have a three-pronged approach to this problem. First, we will formalize the various use cases and their associated provenance needs. Next, we will define a security and privacy model on graphical data that supports the needs of our users. Last, we will develop and revise our programmatic APIs throughout the project period, to meet the needs of our users, the use cases, and the privacy policies. We defer discussion of the API to Section 3.7, discussing the first two approaches below.

Cheney's work on provenance traces [**?**] provides a good starting point for this work. Cheney' work provides a formal model that shows how provenance traces created from the execution of nested relational calculus programs produce the three types of provenance described in the literature: why, where, and how provenance [**?**]. The challenge for us is to map use cases to these three provenance types, identify gaps, use cases for which the provenance we need is none of why, where or how provenance, and then extend the model to incorporate those. We have found the model effective at tackling other problems, such as the security issues described below, which makes us optimistic that it can be adapted for this problem.

The Harvard team has been studying the area of provenance security and privacy and will apply graph summarization techniques to provenance graphs to provide query capabilities under privacy policies. We have built a prototype system that generates provenance traces automatically from the execution of a

program written in nested relational calculus (NRC). We then produce graph summaries from those traces that hides sensitive information. By allowing queries only on the summarized graph, we are able to hide information.

Our summarization techniques draw directly upon our graphical provenance model, described in Section 3.2. A provenance graph describes a particular provenance trace, which corresponds to a particular execution. A summary graph represents some number of provenance traces; the more traces it represents, the more information it hides. We create summary graphs by augmenting our basic graphical model with cardinalities on vertices, edges, attributes, and labels. We specify cardinalities via a lower and upper bound. Thus a vertex in a summary graph with one-to-one correspondence with a vertex in the original provenance graph will have cardinality 1:1, and a vertex that might maps to all vertices in the original graph might have cardinality 1:infinity. The greater the range in cardinality, the more information we can hide, but the query results become less precise.

Our goal in this proposal is to transform the summarization and proof techniques we have for provenance graphs from NRC programs into real programs and to develop summarizations that implement the specific policies required by our sample applications. For example, imagine that one of Gaggle's Geese records each time it exports a data product to a user. Now, let's say that that component is used in a system whose privacy policy makes it unacceptable to collect or release such information. The original provenance graph in Figure 3a could be summarized into Figure 3b to obscure that information. While Figure 3b shows that the data has in fact been input to something, the graph reveals nothing about the identity or number of such objects. The key technical challenges to be addressed here are: 1. developing or selecting a language in which to express the security policies; 2. automatically producing summaries from those policies; and 3. querying on those summaries.

3.6 Responding to provenance queries

Provenance trees can grow quite large – the compilation of the `am_utils` suite produces a provenance graph containing hundreds of thousands of nodes, and large-scale analyses that need to model every data point easily produce millions of nodes. The sheer number of nodes and the fact that the graph continues to grow as data products are used to produce new data products introduces a second question, "How much of the tree should a query return?" Consider the simple "ancestry query" that asks where an object came from. The KMO example from Section **??** uses a database that has been in use for approximately five years. Had we been tracking provenance all this time and then asked for the provenance of an element of that database, the result could return the provenance describing each of the twenty-six versions of the current (third) release of the database. Alternately, an accurate, but probably unhelpful result might include only the last modification. In reality, the query is under-specified and it is usually not reasonable to require that the user know the provenance tree in sufficient detail to specify precisely how far back they would like the result to go. We will develop techniques to return an appropriate amount of provenance automatically in such cases.

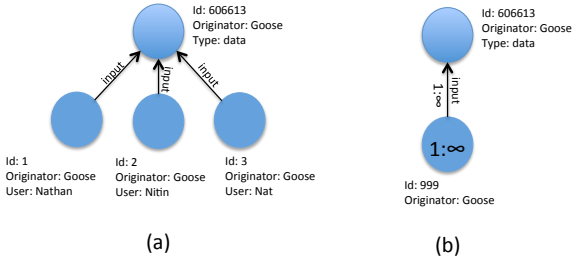

Figure 4: Summarization. The graph on the left (a) shows that the data product with ID 606613 was used as input to three users. This information is obscured in graph (b) by replacing the three nodes with a single node and making the cardinalities of both the node and connecting edge range from one to infinity.

We propose to use properties of the graph itself to determine the amount of provenance a query needs. That is, for any ancestry query, there exists a subset of the entire ancestry tree that is of interest to a user; our goal is to determine that sub-tree of interest. Preliminary work suggests that we can rank nodes in

a provenance tree based upon their likelihood of appearing in any ancestry query and then use discontinuities in that ranking to identify "good" places to truncate the query result. Our work to date suggests that if we pick our ranking metric correctly, discontinuities in that rank represent semantically differentiated points. For example, when we applied the ranking to queries on the provenance tree produced by the first provenance challenge, we found that the first discontinuity corresponded well to the graph from the original workflow specification. The second discontinuity produces an ancestry tree including the workflow and the compilation of the tools used to run the workflow. The third discontinuity includes the installation of the compilation environment itself. Our work to date convinces us that we have two potentially good metrics, SubRank and ProvRank, and that with further research, we can compute them efficiently and construct algorithms to return "the right amount of provenance."

SubRank is a direct measure of the frequency of a node within the space of lineage query. For each provenance object, simply count the number of lineages in which it appears. Computing this translates directly into counting the number of descendants of the node (since the node appears in the ancestry query of every descendant) and adding one (since every node is in its own ancestry). Though simple, this metric appears to work well.

ProvRank derives from the PageRank algorithm of Internet search [**?**]. While SubRank measures an object's frequency of appearing in an ancestry result directly, PageRank computes it via theoretical simulation. If we think of PageRank as a traversal-oriented query process in which the traversal rule is to choose equiprobably from among the outgoing edges of each node, then we can view ProvRank as a traversal-oriented query process in which the traversal rule is to visit every parent. In this case, the ProvRank is the probability that the traversal engine considers a node at a particular instant. When we apply this traversal-approach to a provenance graph, we find that it produces results similar to, but subtly different from those produced using SubRank.

The challenges in this proposal are to evaluate SubRank and ProvRank in more depth, develop alternative metrics that work better, and devise robust threshholding algorithms that can use these metrics to return "the right" portion of an ancestry tree to a user.

A second, and perhaps even more important use of these truncated ancestries is to "attach provenance" to exported data products. As discussed in Section **??**, the exported data products from ISB will be accompanied by provenance, but the full provenance tree will be too large. Using our ranking and truncation approach, we believe we can automatically produce "the right" provenance to transmit with data products.

3.7 API Design

While designing an API is typically considered an engineering task, our challenge here lies in designing an API simple and flexible enough to be adopted in a wide range of use cases, but powerful enough to support the range of use cases to which people want to apply provennace. The PASS Disclosed Provenance API (DPAPI), which is documented in prior work [**?**], provides a good starting point, but is not scalable (object identifiers are tied to operating system resource) and does not capture the semantically meaningful relationships that we need to capture.

The two fundamental abstractions in the DPAPI are pnode numbers, which uniquely identify a collection of provenance, and versions. Pnode numbers are persistent object identifiers. Unlike many provenance systems, the DPAPI assumes that objects are mutable and uses versioning to distinguish these mutations. We will retain these two core concepts, but as a pnode number/version pair uniquely identifies an object in the DPAPI, our new API will incorporate the notion of an originator as described in Section 3.2. The originator identifies the source of a piece of provenance, which we call a *provenance record*. In essence, the originator is provenance of a provenance record. To avoid the potential for infinite recursion when recording provenance of provenance, we simply trust that the originator is accurately recorded; making concrete guarantees of that accuracy is beyond the scope of this work.

The DPAPI always transmits data with provenance. When data is read, the provider of the data must provide its pnode number and version; when data is written, the writer transmits a *provenance bundle*, a colletion of provenance records describing the pnodes on which the write depends. However, the DPAPI does not attach any inherent meaning to the relationships between the different pnodes in a provenance bundle. The API that we develop in this work will use the basic model as described earlier to ascribe specific semantic relationships between different pnodes. The library will then leverage those relationships to accurately resolve the versioning challenges described in Section 3.4 and the inter-layer cycle challenges that arise in a layered system.

4 Engineering Management and Sustainability Plan

By virtue of the geographical separation of ISB and Harvard, the project will be conducted by a distributed workgroup. Seltzer brings 15 years experience with such distributed workgroups (Berkeley DB has always been and continues to be developed by a global workforce). Our plan is to use internal processes and tools that can be easily incorporated into existing software packages.

We present the details of both our software and team management in the supplementary *Management Plan* document and focus the discussion here on software sustainability.

4.1 Sustainability Plan

We are planning for a two-pronged approach to software sustainability: first, as a standalone entity distributed by Harvard, and second, through the existing applications developer communities that exist for Gaggle.

Seltzer's group has a longstanding track record at producing and maintaining software. Berkeley DB, originally distributed by the University of California at Berkeley, traveled with Seltzer to Harvard, where db-1.86 was released, and was then commercialized by Seltzer and Keith Bostic as Sleepycat Software, which is now owned and maintained by Oracle Corporation. In subsequent work, her group developed an instructional operating system, OS161 [**?**], that has been in active use in tens of Universities around the world. Through gift funding, Seltzer's group has maintained and extended the software for the past ten years. More recently, her group has continued to produce releases of the PASS kernel, both as a customized Linux kernel and as a virtual machine installation. By continuing to use artifacts that the group produces in educational and research activities, the software continues to be maintained.

Supporting Seltzer's efforts at releasing software, Harvard's School of Engineering and Applied Sciences has recently announced a new program in Applied Computational Science [**?**]. Initially an educational program, the curriculum includes a significant software engineering component, and the long-range plans for the program include personnel for software support. Seltzer served on the initial planning team for the program and continues to serve on its advisory board. Our hope is that by the end of the award plan for this grant, the library developed can become part of the Applied Computational Science software portfolio.

The ISB team also has a longstanding record at maintaining software for the long term. Gaggle has been in use for over five years and has an active user and developer community. We will leverage this active community and the team at ISB to ensure that provenance support becomes a standard Gaggle feature. Additionally, we will nurture and encourage the existing community of developers that produce Gaggle-enabled tools.

ISB also has a track record in the longevity of its data products. Goodman's group has public databases they have maintained for over a decade, predating Goodman's arrival at ISB. Once we have built provenance support into those data products, the ongoing sustenance of the data products themselves will sustain the provenance capabilities as well. It is precisely this Institute culture of software and data release and sustenance that makes ISB the ideal scientific collaborator for this work.

5 Project Plan

The software can be viewed as six separate components: three application implementations, the provenance library, a set of language bindings, and a set of database adapters. The latter three together comprise the *Core Provenance Library* (CPL). We expect to issue at least three CPL releases and two releases of each application.

Our development strategy is to construct a vertical slice of the project as quickly as possible to put something in end-user hands quickly to obtain user feedback, which we will incorporate into later releases. We will begin by spending the first year addressing everything needed to enable provenance in Gaggle and Gaggle applications. Our rationale is that Gaggle is already widely-used, and we can gain significant leverage and experience early with it. We have users ready to use the new features and developers ready to do the work. In addition, capturing provenance for interactive use is one area in which there is little current work. Finally, the system is entirely in Java, which means we can focus on only one language and one database (MySQL) for the first year. Thus, we will direct the entire team towards this goal. We can achieve this early milestone by leveraging our API experience in PASS as well as significant portions of the PASS infrastructure for provenance recording and storage. Thus, at the end of year one, we expect to have the first release of both the CPL and provenance-enabled Gaggle (PE-Gaggle).

Once we have deployed the first release of PE-Gaggle, we will simultaneously solicit user and developer feedback, monitor the growth of provenance stores, and begin work on other CPL components. As the Data Ingest Project and Metabolic Network Inference project require bindings in Perl and Python, we will begin the next development phase with bindings for those languages. During year two, we will issue minor CPL releases that introduce additional language support. In parallel, we will begin to address the challenges in formalizing and enabling provenance collection for different uses. The ISB team will begin work on provenance integration for Data Ingest Pipeline and Metabolic Network Inference.

By the middle to end of year two, we expect to have significant user feedback from Gaggle users, which will lead to the detailed design of the second version of the CPL. We will first address API changes to minimize the impact to the other two applications.

The next major release will include the revised API, the full set of language bindings, any database adapters that have been requested, and support for all the features discussed in this proposal. By the end of year 3, we will have the second CPL release and first release of all applications and potentially a second release of Gaggle.

In the year four, we will solicit feedback from all application users and design and deploy the third major version of the CPL, as well as second (and possibly third) release of applications. By the end of this proposal, we expect the CPL to be stable and ready for adoption into a broad range of tools from different fields.

6 Education and Outreach

Harvard's emergent program in Applied Computational Science (IACS) [**?**] and Seltzer's own graduate courses provide the foundations of our educational plan. The IACS program is a master's degree program for both terminal master's students as well as Ph.D students in domain sciences whose work requires a significant computational component. The program requires two core courses in computer science, one in software engineering and one in Computing Foundations for Computational Science, as well as multiple electives.. As part of the software engineering course, students must undertake significant projects, and we will actively recruit students to use our provenance library in their work.

The program also includes several computer science electives and we expect that Seltzer's undergraduate information management course will be a popular elective among students (she regularly has graduate students from domain sciences audit the course). This course also has a final project component where

students are asked to identify a real world data management problem and design and implement a solution to the problem. Traditionally, a few of the projects from this course become long term useful artifacts. In the last offering, one student developed a scheduling tool for use by the atheletic department that was deployed this past spring. Another student developed a new social networking paradigm that he is attempting to commercialize, and a third student developed a room scheduling tool that we are hoping to use within SEAS. When the provenance library is ready for use, we will encourage students in this course to undertake projects in computational science using the provenance library.

These provenance tools will also be integrated into the Harvard Forest Summer Research Program in Ecology (an NSF REU site since 1993, currently supported by DBI 10-03938 "Ecological data-model fusion and environmental forecasting for the 21st century). For the last four years, two undergraduates have worked with co-PI Lerner and Senior Investigator Boose on developing RDataTracker. In 2013 and 2014, other students in this REU site (20-30/year) use R for data analysis and we have been working with these students to integrate RDataTracker and provenance tools into their analysis. In August 2014, we will be submitting a renewal proposal for our REU program. The theme for the next five years will be collection, use, and analysis of environmental "big data", and will integrate provenance tools throughout all undergraduate summer research projects.

7 Results from Prior NSF Support

7.1 PQL: A Path Query Language

Margo Seltzer was PI on a 1–year NSF SGER grant number 0849392, NSF 08-1, "PQL: A Path Query Language" with a budget of $130,000. This grant funded preliminary work on the design and implementation of a new query language, particularly amenable to searching graphical structures, such as those produced by recording provenance or lineage [**?**]. We now have a formal semantics of the language and a prototype implementation. Since completion of the funded work, we have designed an update language to augment the query language.

7.2 Support for Atomic Sequences of File System Operations

Margo Seltzer was Co-PI on CSR-PDOS NSF grant number 0614784 with Erez Zadok of SUNY Stonybrook, "CSR—PDOS: Support for Atomic Sequences of File System Operations" (total budget $685,837; Harvard allocation $202,947). The joint team explored the use of atomic file system operations to enhance the functionality and ease of development of sophisticated services. The provenance-aware storage system, built by Seltzer and her students [**?**], is a sample of such a service. The Harvard team has developed a collection of provenance-aware systems including a versioning file system, a network-attached file system, a workflow engine, an interpreter and a browser [**?**, **?**, **?**, **?**].

7.3 Harvard Forest Long Term Ecological Research

Aaron Ellison was co-PI and Emery Boose is Senior Investigator on DEB 0620443 (" LTER IV: Integrated studies of the drivers, dynamics, and consequences of landscape change in New England", $4.92M, 6 years, ending October 2012). Begun in 1988, the Harvard Forest LTER (HFR) is an ongoing integrated research and educational program that examines responses of forest dynamics to natural and human disturbances and environmental changes over broad spatial and temporal scales. HFRs ¿60 scientists from seven institutions investigate past, present, and future ecological patterns and processes in New England. HFRs observations and experiments test fundamental ecological hypotheses; long-term studies continually illustrate that hypotheses derived from short-term studies, experience, or intuition are often rejected as unanticipated factors, events and processes alter trajectories of ecological dynamics.

Central HFR findings HFR (see also http://harvardforest.fas.harvard.edu/research/LTER) include:

- Historical legacies of land use and biotic conditions interacting with long-term environmental change, and natural and human-induced disturbances condition ecological patterns and processes (e.g., Foster and Aber 2004, Thompson et al. 2011);
- Climatic change and disturbance together can trigger abrupt ecological shifts by causing the loss of foundation species that control biotic and environmental conditions and ecosystem processes (e.g., Albani et al. 2010, Ellison et al. 2010, Orwig et al. 2013);
- Ecosystem trajectories have large inter-annual variability (e.g., Urbanski et al. 2007);
- Strong biogeochemical resiliency to disturbance maintains ecosystem functions despite disturbance-induced changes in system structure (e.g. Ollinger et al. 2008, Melillo et al. 2010, Finzi et al. 2011);
- Effective ecological interpretation and management depend on integrated study of human/natural systems through retrospective study, decadal measurements, experiments, and modeling (Foster and Aber 2004)
- Scientists must engage early with decision-makers to span the science and policy boundary (Foster et al. 2010, Lambert 2010). HFR research is documented in ¿600 peer-reviewed publications and a synthesis volume (Foster & Aber 2004). Application of these results and insights has directed development of state, regional, and national policies for forest conservation and management. HFR also is the NEON Domain 1 core site; plays a major role in LTER leadership, strategic planning, and network-wide studies; and has been involved with two ULTRA planning grants.

HFR research is documented in ¿600 peer-reviewed publications and a synthesis volume (Foster & Aber 2004). Application of these results and insights has directed development of state, regional, and national policies for forest conservation and management. HFR also is the NEON Domain 1 core site; plays a major role in LTER leadership, strategic planning, and network-wide studies; and has been involved with two ULTRA planning grants.

citations are entered as comments in the tex file