Tracing where and who provenance in Linked Data: a calculus

Mariangiola Dezani-Ciancaglini[☆a], Ross Horne[☆b], Vladimiro Sassone[c]

[a]*Dipartimento di Informatica, Università di Torino, Italy*
[b]*Institute of Computer Science, Romanian Academy, Iaşi, Romania*
[c]*Electronics and Computer Science, University of Southampton, United Kingdom*

Abstract

Linked Data provides some sensible guidelines for publishing and consuming data on the Web. Data published on the Web has no inherent truth, yet its quality can often be assessed based on its provenance. This work introduces a new approach to provenance for Linked Data. The simplest notion of provenance – viz., a named graph indicating where the data is now – is extended with a richer provenance format. The format reflects the behaviour of processes interacting with Linked Data, tracing where the data has been published and who published it. An executable model is presented based on abstract syntax and operational semantics, providing a proof of concept and the means to statically evaluate provenance driven access control using a type system.

Keywords: Operational Semantics, Type Systems, Linked Data

1. Introduction

The Web challenges traditional perspectives on data. Traditionally, data is stored in centralised databases with clear ownership. Only trusted experts have access to the mechanisms for manipulating data and the boundaries of each data set are fixed. Linked Data shatters these assumptions, by removing the boundaries between datasets and reducing barriers for publishing data.

Linked Data is a movement that is pushing data onto the Web [7]. To reflect the diversity of data a light data format is introduced. The data format is based on triples of Uniform Resource Identifiers (URIs). Triples of URIs are versatile. A wide variety of data sources can be lifted to collections of triples. Furthermore, the use of URIs as standardised global identifiers allows data from one source to refer to data in another source. Several protocols are then employed to consume and publish data collaboratively. With many contributors and locations, tracking the provenance of data becomes a significant challenge.

In the Web of Linked Data, anyone can form a triple. If someone makes a statement and publishes it as a triple on the Web, it does not mean that the triple can be trusted.

[☆]The first author is supported by the MIUR Project IPODS. The second author is supported by CNCS-UEFISCDI Project PN-II-ID-PCE-2011-3-0919.

The degree of trustworthiness of the triple will depend on the trustworthiness of the individuals involved in producing the triple and the judgement of the consumer of the triple. Thus this work argues that a consumer of Linked Data is most concerned with the provenance of triples. The provenance of any values inside triples is secondary. For Linked Data the big concern is the provenance of a statement such as “Yoshihito Toyama is affiliated with Tohoku University,” rather than the provenance of identifier for Tohoku University.

The problem of provenance tracking for Linked Data is well known. A basic provenance mechanism called a named graph is widely supported [13, 53]. A named graph extends triples with an extra location, which indicates where the triple is located. This is the simplest kind of where provenance. From a triple in a named graph, decisions can be made based on where the triple is located now. A query may specifically ask for triples lifted from the BBC News feed for Asia.

A named graph only captures ‘where now’ provenance. Harth, Polleres and Decker argue that a more social provenance model is required [33]. By a social provenance model, they mean that the context should record who provenance. The ‘where now’ provenance of named graphs can be extended with ‘who now’ provenance, by also identifying the agent who published the data. This work captures such social provenance, by extending the notion of a named graph with pairs.

This work extends named graphs one step further, by tracing provenance history. The protocols for Linked Data allow triples to be retrieved from locations and written to other locations [51]. Thus a history of ‘where and who’ provenance can be accumulated. Each time an agent publishes data in a location, the agent and location can be recorded in the provenance history of the triple. Furthermore, the data may be processed locally, by the agent. Recording the operations that were applied to the data provides a notion of ‘how’ provenance.

This work provides a calculus of processes which use, consume and publish Linked Data tracing data provenance traces. The calculus demonstrates that the proposed provenance format tracks the provenance of data according to the processes modelled. It allows the claim that the provenance format introduced is suitable for Linked Data to be evaluated.

The calculus goes one step further, by exploiting the provenance format to enhance Linked Data protocols. The model introduces a logic for provenance patterns. The logic can be used to specify precise queries over Linked Data. The calculus is then typed to ensure that an access control policy based on the provenance format can be maintained. A type system ensures the integrity of the policy of each location. Such policies can improve the reliability of Linked Data. A first key result is a subject reduction theorem, which verifies that such policies can be guaranteed by the static analysis of processes. Further formal results prove that the policies described are indeed captured by the model. The implication is that this static analysis can be applied to programs which follow the Linked Data protocols modelled.

The present paper is organised as follows. Section 2 introduces provenance and Linked Data. The syntax and the operational semantics of the calculus are the contents of Section 3 and of Section 4, respectively. Section 5 introduces a type system whose properties are shown in Section 6. Related work is discussed in Section 7 and some conclusions are drawn in Section 8. The electronic version of this paper contains the

URIs mentioned as active links.

2. Linked Data: Guiding Principles and Provenance

Some ubiquitous Web standards are employed to support Linked Data. These standards enable the decentralised identification of resources, the transfer of data, the representation of data and the exploitation of data. These technologies are briefly summarised, and an overview of how these technologies meet the guidelines for Linked Data is provided.

URIs are identifiers for resources on the Web. A URI may be used in HTTP to supports the fundamental operations for publishing and consuming data at that URI. URIs may also be used in the Resource Description Framwork (RDF), to identify resources in data. RDF is a standardised loosely structured data format that allows the resources identified by URIs to be described by their relationship to other URIs. Finally, SPARQL, a protocol and RDF query language, standardises some basic mechanisms for exploiting RDF.

The guiding principles of Linked Data were outlined by Berners-Lee [5]. Firstly, in data use URIs to name things; as opposed to a local identifier scheme specific to a dataset. Secondly, make use of the HTTP protocol, so that the URIs can be looked up. This process of looking up a URI using HTTP, called dereferencing, should provide useful information using standards, specifically RDF and SPARQL. Finally, include other dereferenceable URIs in the dereferenced data, so that more things can be discovered. The idea is that distributed data published following these guidelines can be used collectively, without prior coordination.

Information about the publications of Yoshihito Toyama is readily available as Linked Data. There is of course a URI for the home page of Toyama at Tohoku University. However, the Web page does not return Linked Data, so is not dereferenceable. Fortunately, there are many dereferenceable URIs for Toyama. One such URI can be dereferenced to obtain the following data.

subject	property	object
Toyama	*akt:type*	*akt:Person*
rkpres:CS132820	*akt:has-author*	*Toyama*

The above data is in the standard format of RDF triples. Furthermore the data contains dereferenceable URIs. Thus the URI for the paper, *rkpres:CS132820*, can be dereferenced to obtain some more information. The data returned includes the following data.

subject	property	object
rkpres:CS132820	*akt:has-author*	*Toyama*
rkpres:CS132820	*cites*	*rkpres:CS323375*

Notice that these triples have a different provenance, the first two triples came from dereferencing a URI for Toyama, the second two triples came from dereferencing a URI for one of his papers. Indeed the same triple appeared twice with different provenance traces.

The source of these triples can be traced further. Three of these triples originated from a data source published at URI source1. This source can be traced back to a data dump made by the website CiteSeer. Thus the origin of the last two triples above might be traced as the following sequence of two URIs followed by an operation *lift*. Thus *lift#* is the mapping of a function, which transforms some raw data into RDF, into the provenance format. The left most location is the most recent.

$$\textit{rkpres:CS132820} \cdot \text{source1} \cdot \textit{lift\#}$$

Furthermore, we can trace the agents who wrote this data. In this case, an agent acting on behalf of *CiteSeer* lifted the data and published the data as RDF. Another agent *RKBexplorer* owned the process that republished part of this information for the above paper. Thus the agents can be included in the provenance trace. Notice that the agent that wrote the data and the location where the data was published form a pair.

$$(\textit{RKBexplorer}, \textit{rkpres:CS132820}) \cdot (\textit{CiteSeer}, \text{source1}) \cdot \textit{lift\#}$$

There are several operations involved in the scenario in this section. There is the process owned by agent *CiteSeer*, that obtained the data and published it in a location. There is the process owned by *RKBexplorer*, that obtained data from several locations, including the dump above. The data was then filtered for information about a URI and the resulting data was published at the URI, so that the URI could be dereferenced. This work introduces a calculus which captures the behaviour of processes in this scenario whilst automatically tracking the provenance of triples.

Notice that including the entire trace is stronger than the approach offered by named graphs [13]. A named graph just provides a current location for a triple, it cannot track the provenance history. With the extended provenance format we can determine whether a triple originated with the *ACM*, *CiteSeer* or *DBLP* as well as the current location. This is useful since these are trustworthy sources for data on academic publications. We may also only trust data that was most recently handled by *RKBexplorer*, since *RKBexplorer* is a reliable agent for gathering Linked Data. These, and many more, provenance patterns can be handled by this work.

The language suggested by this work is a high level language, where interaction with provenance is primitive. At times, several low level operations are covered by a single high level operation. However, there should be no doubt that the language encompasses the guidelines for publishing Linked Data. The provenance format is a serious contribution to the area of Linked Data, extending existing approaches for tracing the provenance of triples.

3. A Syntax for Capturing Provenance in Linked Data

The methods employed here are purely syntactic. This section introduces the basic atoms and grammars required to discuss both a provenance format and systems in which the provenance represented can be traced.

3.1. The Syntax for a Format for Provenance Traces

The basis of the provenance format are identifiers representing where and who. Both identifiers are readily provided by the Linked Data architecture. Using these identifiers, a provenance format is proposed, which is evaluated throughout this work for its effectiveness in tracking the provenance of triples.

agents: α, β functions: f, g

locations: ℓ, m location variable: a, b

location or variable: $\lambda ::= \ell \mid a$

provenance traces:

$$\begin{array}{rcll} p & ::= & \epsilon & \text{empty} \\ & \mid & (\alpha, \ell) & \text{who-where} \\ & \mid & f\# & \text{why} \\ & \mid & p \cdot p & \text{concatenation} \\ & \mid & p \vee p & \text{disjunction} \end{array}$$

$\sigma ::= \alpha \mid ?\_ \quad \Lambda ::= \lambda \mid ?\_ \quad \phi ::= f \mid ?\_$

patterns:

$$\begin{array}{rcll} \pi & ::= & \epsilon & \text{empty} \\ & \mid & (\sigma, \Lambda) & \text{who-where} \\ & \mid & \phi\# & \text{why} \\ & \mid & \pi \cdot \pi & \text{concatenation} \\ & \mid & \pi \vee \pi & \text{disjunction} \\ & \mid & \pi^{*} & \text{Kleene star} \\ & \mid & \top & \text{top} \end{array}$$

Figure 1: The provenance format and pattern syntax.

Identifiers for where, who and why. This work refers to URIs as locations, ranged over by ℓ, m in definitions. *Toyama* denotes one of several dereferenceable URIs for Yoshihito Toyama.

Entities that run processes, are referred to as agents. It is assumed here that each agent has an identifier, ranged over by α, β, which identifies 'who.' Harth et al. note that the Web has a built in mechanism for identifying agents via the Domain Name System (DNS) [33]. There are obvious issues with using DNS to identify agents, particularly since the correspondence between DNS identifiers and agents is not necessarily one-to-one. The exact choice of identifier is perpendicular to this work.

This work assumes that there are some basic functions which can be applied to data, ranged over by f, g. These functions can be recorded in the provenance format where they appear as the function names followed by #. There are various mechanisms these functions could represent, so the details are not provided in this work. The point is that some basic 'why' provenance can be recorded.

The provenance format. Figure 1 introduces the provenance format, which traces 'who' and 'where' provenance with some basic 'why' provenance information. A who-where provenance pair indicates that a particular agent published data in a particular location. That data may be retrieved and published in another location by another agent. Each time the trace is extended. Disjunction can be used in provenance traces to represent that the data from several sources have been combined. Why they were combined may also be indicated.

The following trace represents that initially there were two pieces of data. One piece of data was published by agent *ACM* in *acm_1*, another piece was published by agent *CiteSeer* in *cs_1*. Agent *RKBExplorer* consumes both pieces of data, applies the function *Clean* to the combination of both pieces of data, and publishes it in location *Toyama*.

$$(\mathit{RKBExplorer}, \mathit{Toyama}) \cdot \mathit{Clean}\# \cdot ((\mathit{ACM}, \mathit{acm\_1}) \vee (\mathit{CiteSeer}, \mathit{cs\_1}))$$

The pattern language. Figure 1 introduces the syntax of patterns. The basic atoms of the patterns are who-where pairs and functions extended with wild cards. The wild data variable: x, y

queries:

$$\begin{array}{rll} Q ::= & (\lambda\,\lambda\,\lambda)^{\pi} & \text{triple pattern} \\ \mid & Q \oplus Q & \text{choose} \\ \mid & Q \otimes Q & \text{tensor} \\ \mid & \exists a.Q & \text{exists} \\ \mid & *Q & \text{iteration} \end{array}$$

stored data:

$$\begin{array}{rll} D ::= & (\ell\,\ell\,\ell)^{p} & \text{tracked triple} \\ \mid & D \mid\mid D & \text{composition} \\ \mid & 0 & \text{empty data} \end{array}$$

expressions:

$$\begin{array}{rll} e ::= & x & \text{variable} \\ \mid & D & \text{data} \\ \mid & f(e) & \text{application} \\ \mid & e \mid\mid e & \text{composition} \end{array}$$

Figure 2: The syntax of stored data queries and expressions.

cards state that any agent, location or function can be matched. The pattern language for provenance traces is based on Kleene algebras. The operations are concatenation, and disjunction of patterns as well as a Kleene star, which allows a pattern to be matched an arbitrary number of times. There is also a top element, which can match any pattern.

The following examples are useful patterns which can be expressed. Using iteration and disjunction, the pattern $((\mathit{Glaser}, ?\_) \vee (\mathit{Millard}, ?\_))^{*}$ guarantees that only agents *Glaser* or *Millard* ever wrote this data. The pattern $\top \cdot (?\_, \mathit{acm\_1})$ states that data originated from the ACM periodical data. The pattern $\top \cdot (\mathit{L3S}, ?\_) \cdot (?\_, \mathit{Toyama}) \cdot \top$ states that at some point data published in location *Toyama* was republished by agent *L3S*.

Note. Glaser, Millard and the organisation L3S published the real Linked Data used in examples.

3.2. Mechanisms for Manipulating Linked Data

This section introduces the standards for Linked Data, extended with the provenance formats from the previous section. Query mechanism for picking out patterns in Linked Data are extended with provenance patterns, so that provenance traces can be exploited in queries. Finally, expressions that manipulate data are suggested. See Fig. 2.

Annotating triples with provenance. RDF is based on triples of URIs, which resemble simple sentences in natural language. The first component is the subject, the second the property and the third the object. There are some further features of RDF including literal values and blank nodes [36], however this work focuses only on URIs.

For stored data the syntax of triples is decorated with a provenance. The provenance represents the history of where, who and why the triple was obtained. Furthermore, the most recent provenance indicates where the triple is now. An example is:

$$(\mathit{ipl{:}Toyama87}\ \mathit{dc{:}creator}\ \mathit{Toyama})^{(\mathit{L3S},\mathit{dipl{:}Toyama87})\cdot(\mathit{DBLP},\mathit{source4})}$$

Annotating queries with patterns. The standard for querying RDF is SPARQL. A formal syntactic model for SPARQL queries has been provided [38]. For brevity, this work takes the most relevant subset of SPARQL and adapts it to this setting, represented as the queries in Fig. 2. The constructs form a commutative Kleene algebra with existential quantifiers over triples annotated with provenance patterns. The disjunction gives a choice of queries, the tensor allows more than one triple to be identified, the existential quantifiers allow locations to be discovered. The Kleene star allows many instances of a triple to be matched.

Queries are used in this setting to test whether some data matches a pattern. This is used in the calculus to consume data which matches only that pattern. The following query demands that two triples are discovered. The subject of both triples must be the same URI. Furthermore, the provenance patterns ensures that the triples were originally posted by an agent on behalf of the ACM.

$$*\exists a.\Big((a\ \mathit{dc{:}creator}\ \mathit{Toyama})^{\top\cdot(\mathit{ACM},?\_)} \otimes (a\ \ \mathit{journal}\ \ \mathit{ipl})^{\top\cdot(\mathit{ACM},?\_)}\Big)$$

Expressions over data. To allow data to be manipulated, functions mapping stored data to stored data are introduced. The functions are used in the syntax of expressions to represent the manipulation of data. Future work would introduce a calculus of functions to precisely specify the transformation performed, facilitating a more detailed analysis of why provenance.

3.3. A Syntax for Process Configurations

This section introduces processes that manipulate data. Systems then model a combination of data decorated by provenance traces and processes run by particular agents. Firstly, policies for URIs must be explained since they appear in processes. Policies control the access of an agent to a location based on the provenance of data.

The syntax of policies. According to the syntax of systems, Fig. 3, agents interact with data by means of three operations: getting, deleting and inserting. Therefore it is sensible to design location policies prescribing which agents can read and modify their data.

For example the location *dipl:Toyama87* can allow anybody to get the data inserted by *Glaser* and originally posted by an agent representing the *ACM*. This can be represented by the access triple:

$$\langle ?\_, \mathit{Glaser}, \top \cdot (\mathit{ACM}, ?\_)\rangle$$

The same location can allow *Millard* to delete only data inserted by himself and at some point published at location *acm_1*, while *Glaser* can delete arbitrary data. This is represented by the following set of access triples:

$$\{\langle \mathit{Millard}, \mathit{Millard}, \top \cdot (?\_, \mathit{acm\_1}) \cdot \top\rangle, \langle \mathit{Glaser}, ?\_, \top\rangle\}$$

Lastly only *Millard* and *Glaser* can insert data in the location *dipl:Toyama87*, and while *Millard* must take the data from *DBLP*, *Glaser* can take data from any source. This is expressed by the set of insert pairs:

$$\{\langle \mathit{Millard}, (?\_, \mathit{DBLP}) \cdot \top\rangle, \langle \mathit{Glaser}, \top\rangle\}$$

process variable: X

policy of locations: $\texttt{Loc}(\mathcal{R}, \mathcal{D}, \mathcal{I})$

processes:

P	$::=$	0	termination
	$\mid$	$\texttt{get}\,(Q, x)\,.P$	consume
	$\mid$	$\texttt{del}\,(Q, x)\,.P$	delete
	$\mid$	$\texttt{ins}\,(\lambda, e)\,.P$	publish
	$\mid$	$P + P$	choose
	$\mid$	$\exists a\colon \texttt{Loc}(\mathcal{R}, \mathcal{D}, \mathcal{I}).P$	select location
	$\mid$	X	process variable
	$\mid$	$\texttt{rec}X.P$	recursion

access triples: $\kappa ::= \langle \sigma, \sigma, \pi \rangle$

insert pairs: $\iota ::= \langle \sigma, \pi \rangle$

sets of access triples: $\mathcal{R}, \mathcal{D}$

sets of insert pairs: $\mathcal{I}$

systems:

S	$::=$	$\mathbf{0}$	termination
	$\mid$	D	stored data
	$\mid$	$\alpha\,[P]$	agent
	$\mid$	$S \parallel S$	parallel

Figure 3: The syntax of processes and systems.

These examples justify that the location policies are built from three sets (see Fig. 3):

1. the set $\mathcal{R}$ of access triples that controls get access to data;
2. the set $\mathcal{D}$ of access triples that controls delete access to data;
3. the set $\mathcal{I}$ of insert pairs that prescribes the data insertion policy.

The syntax of processes. Processes suggest a high level programming language that combines HTTP operations with queries and expressions. This allows sequences of interactions with Linked Data, including dereferencing, to be expressed. The syntax of processes is defined in Fig. 3.

There are three operators for interacting with Linked Data – get, delete and insert. These operations model both HTTP operations and query mechanisms [51]. Get retrieves some data, but also restricts the data using a query pattern. The resulting data is then passed on to a continuation process, since the data variable binds occurrences in the continuation process. I.e. the process P is the continuation of the process $\texttt{get}\,(Q, x)\,.P$.

Since queries include provenance patterns, the provenance of the data retrieved is also selected. This allows the most basic dereference operation to be realised, as shown below. Notice that a dereferenceable URI for Toyama appears in the provenance pattern, indicating the location to dereference. The query pattern asks for anything, so all data in that location can be retrieved.

$$\texttt{get}\left(*\exists a.\exists b.\exists c.(a\ b\ c)^{(?\_, Toyama)\cdot\top}\ , x \right).Display\_Data(x)$$

The data variable binds occurrences in the continuation process. It is assumed that the process *Display_Data* does something useful with the data retrieved. Several examples of get with more precise queries are provided throughout this work.

Delete has the same form as get. The only difference is the data retrieved which matches the query is removed, whereas get persists the data. Insert consists of an expression, a location and a continuation process. Insert is used to publish the data which results from evaluating the expression at the location, before continuing.

The other operations are used to control the flow of processes. The choice operator allows one of two processes to be selected. The existential operator binds a location variable. When a location variable in a query is bound, the location discovered is unified with the continuation. This allows locations to be passed from data to continuation processes. Notice that the bound location variable is annotated with a location policy. This is necessary to guarantee that the variable is replaced by a location of that policy.

The process variable and fixed point operator allow a process to behave recursively. Finally, there is a process representing termination.

The syntax of systems. The state of systems is expressed using the syntax in Fig. 3. Systems indicate the agent that runs a process. This information is required for tracing who provenance. The processes of several agents can be composed in parallel with stored data. Two examples of processes which refer to the agent that runs them are provided.

Suppose that an agent α has made some contribution to data located at *pubs* . The following pattern removes all data this agent has created. Both data that was written directly to the location by the agent, and data in the location that was touched by the agent at some point in the past is removed.

$$\alpha\left[\mathtt{del}\left(*\exists a.\exists b.\exists c.(a\; b\; c)^{(\alpha,pubs)\cdot\top\vee\left((?\_,pubs)\cdot\top\cdot(\alpha,?\_)\cdot\top\right)}, x\right).0\right]$$

Suppose that an agent β periodically moves information from one location to another location (the origin location *acm_1* and the target location *pubs*). The agent obtains citations from the source location. The agent then removes citations with a provenance that indicates that the agent himself had obtained them from the source location and inserted them in the target location. Finally, the agent inserts new triples in place of the triples removed.

$$\beta\left[\mathtt{rec}X.\left(\begin{array}{l}\mathtt{get}\left(*\exists a.\exists b.(a\; cites\; b)^{(?\_,acm\_1)\cdot\top}, x\right).\\ \mathtt{del}\left(*\exists a.\exists b.(a\; cites\; b)^{(\beta,pubs)\cdot(?\_,acm\_1)\cdot\top}, y\right).\mathtt{ins}\left(pubs, x\right).X\end{array}\right)\right]$$

Notice that the triples inserted will have the same provenance as the triples removed in the previous step. This means that on the next iteration of the recursion, these triples will be removed and replaced by more up to date triples. The effect is that β maintains a copy of citation data from the source location at the target location.

The next section defines the behaviour of the above systems precisely.

4. An Operational Semantics for Provenance Tracking in Linked Data

Operational semantics captures the behaviours of the systems modelled by the syntax of the previous section. The behaviour of systems depends on the evaluation of provenance patterns, queries and expressions, which are formalised using deductive systems. Given the mechanisms defined in this section, substantial examples that track provenance in Linked Data can be executed.

4.1. Pre-order on Patterns and Satisfaction of Patterns

Let fix finite sets $\mathcal{A}$ of agent names, $\mathcal{L}$ of location names and $\mathcal{F}$ of function names. The mapping re associates to each pattern π which only contains names in $\mathcal{A}\cup\mathcal{L}\cup\mathcal{F}$ a regular expression on the alphabet $\Sigma = \{(\alpha, \ell) \mid \alpha \in \mathcal{A} \,\&\, \ell \in \mathcal{L}\} \cup \{f\# \mid f \in \mathcal{F}\}$, where the notation for regular expressions is as in [2]. The main purpose of this mapping is to reduce the wild cards in patterns to well understood regular expressions:

$$\mathsf{re}(\epsilon) = \epsilon \qquad \mathsf{re}((\sigma, \Lambda)) = \bigl|_{\alpha\in\mathsf{reA}(\sigma),\ell\in\mathsf{reL}(\Lambda)} (\alpha, \ell) \qquad \mathsf{re}(\phi\#) = \mathsf{reF}(\phi)$$

$$\mathsf{re}(\pi \cdot \pi') = \mathsf{re}(\pi)\mathsf{re}(\pi') \quad \mathsf{re}(\pi \vee \pi') = \mathsf{re}(\pi) \mid \mathsf{re}(\pi') \quad \mathsf{re}(\pi^*) = \mathsf{re}(\pi)^* \quad \mathsf{re}(\top) = \Sigma^*$$

where

$$\mathsf{reA}(\sigma) = \begin{cases} \{\alpha\} & \text{if } \sigma = \alpha, \\ \mathcal{A} & \text{if } \sigma = ?\_ \end{cases} \qquad \mathsf{reL}(\Lambda) = \begin{cases} \{\ell\} & \text{if } \Lambda = \ell, \\ \mathcal{L} & \text{if } \Lambda = ?\_ \end{cases}$$

$$\mathsf{reF}(\phi) = \begin{cases} f\# & \text{if } \phi = f, \\ \bigl|_{f\in\mathcal{F}} f\# & \text{if } \phi = ?\_ \end{cases}$$

Following the semantic subtyping approach [14] the *pre-order* $\leq$ *on patterns* is defined by:

$$\pi \leq \pi' \text{ only if } \mathsf{re}(\pi) \subseteq \mathsf{re}(\pi')$$

where the sets of names are those occurring in π, π' and the inclusion between regular expressions stands for the inclusion between the generated languages. The pre-order on patterns can then be decided in exponential time for the worst case; however there exist algorithms which are efficient in practice [2].

The mappings reA, reL and reF make $?\_$ the top of agents, locations and why functions. Similarly, re makes $\top$ the top of patterns:

$$\alpha \leq ?\_ \qquad \lambda \leq ?\_ \qquad f\# \leq ?\_\# \qquad \pi \leq \top$$

Since provenance traces are a subset of patterns, satisfaction of patterns is a special case of the above pre-order.

A provenance p satisfies a pattern π (notation $p \Vdash \pi$) if $\mathsf{re}(p) \subseteq \mathsf{re}(\pi)$.

The following proposition can be easily shown by the definitions of pre-order and pattern satisfaction.

Proposition 4.1. *$\pi \leq \pi'$ if and only if, for all p, $p \Vdash \pi$ implies $p \Vdash \pi'$. In particular $p \leq \pi$ if and only if $p \Vdash \pi$.*

$$\frac{p \Vdash \pi}{C^p \models C^\pi}\,\lceil\text{QAx}\rceil \qquad \frac{D \models Q_0}{D \models Q_0 \oplus Q_1}\,\lceil\text{QChL}\rceil \qquad \frac{D \models Q_1}{D \models Q_0 \oplus Q_1}\,\lceil\text{QChR}\rceil$$

$$\frac{D_0 \models Q_0 \quad D_1 \models Q_1}{D_0 \parallel D_1 \models Q_0 \otimes Q_1}\,\lceil\text{QT}\rceil \qquad \frac{D \models Q\{{}^{\ell}/_{a}\}}{D \models \exists a.Q}\,\lceil\text{QE}\rceil$$

$$\frac{}{\models *Q}\,\lceil\text{QW}\rceil \qquad \frac{D \models Q}{D \models *Q}\,\lceil\text{QD}\rceil \qquad \frac{D \models *Q \otimes *Q}{D \models *Q}\,\lceil\text{QC}\rceil$$

Figure 4: Satisfaction of queries

4.2. Satisfaction of Queries

Figure 4 presents a deductive system for deciding whether some data satisfies a query. The definition adapts the most relevant subset of the model of a SPARQL query presented in [38].

The axioms for queries hold when a stored triple, matches a triple demanded by the query. Triples of locations and of locations variables are denoted by C. In this axiom C is a triple of locations. The stored triple is annotated with a provenance, while the query triple is annotated with a pattern. The axiom is therefore dependent on the provenance matching the pattern, as defined in the previous section.

Further operators enable an expressive query language. The tensor rule is a language-based approach to joining queries. The rule ensures that both parts of a query are simultaneously answered using distinct resources. Traditional join semantics give a denotational set of all solutions to a query, which is a mismatch for this operational approach [52]. Instead, a particular solution is selected for execution. The linear restriction, which ensures disjointness of the resources used, is suited to a concurrent distributed setting [38].

The rules for choose and exists provide more flexibility in queries by selecting one of several ways in which a query can be evaluated. Iteration employs weakening, dereliction and contraction to enable a pattern to be answered an unbounded number of times.

4.3. Expression Evaluation

A big step operational semantics is provided for expressions, though the relation $\Downarrow$ defined in Fig. 5. Functions are recorded in the provenance of triples only when they affected the triples.

Raw stored data is simply stored data without the provenance annotation. The functions from stored data to stored data are defined with respect to their underlying functions from raw stored data to raw stored data. More precisely assume that for each function f there is a corresponding raw function $|f|$ from raw stored data to raw stored data. The raw function may be undefined if some of the triples in the data are

$$\frac{}{D \Downarrow D}\ \lceil\mathrm{id}\rceil \qquad \frac{|f|(\Pi_{i=1}^{m} C_i) \downarrow \Pi_{j=1}^{n} \hat{C}_j \quad q = f\# \cdot \bigvee_{i=1}^{m} p_i}{f(\Pi_{i=1}^{m} C_i^{p_i} \parallel D) \Downarrow \Pi_{j=1}^{n} \hat{C}_j^{q} \parallel D}\ \lceil\mathrm{fun}\rceil$$

$$\frac{e_0 \Downarrow D_0 \qquad e_1 \Downarrow D_1}{e_0 \| e_1 \Downarrow D_0 \| D_1}\ \lceil\mathrm{par}\rceil$$

Figure 5: A big step operational semantics for expressions.

unused. The notation $\downarrow$ (inspired by big step semantics) is used for the evaluation of raw functions.

For more accurate why and how provenance the nature of these functions over data need to be more precisely defined. It would be possible to base the functions on the calculus of SPARQL Updates [40]. Each update is associated with a proof that explains why an update holds. The proofs should however be considered up to equivalence, which greatly increases the cost of evaluating provenance patterns.

A model of how provenance for updates in databases is provided by Green et al. [29]. Green et al. employ a provenance structure that accounts for the data deleted and the data inserted by updates. Similar provenance structures could be considered here.

4.4. Policies

The pre-order on agents, locations, why functions and patterns (see Sec. 4.1) naturally induces the component-wise pre-order on access triples and insert pairs. We can then also compare location policies: a smaller policy must allow all that is allowed by a bigger policy. This agrees with standard subtyping. In this way each location can be ascribed all policies that are bigger than its own. This can be achieved by asking that all triples and pairs in the bigger policy have corresponding bigger triples and pairs in the smaller policy. This leads to the definition:

$$\mathtt{Loc}(\mathcal{R}', \mathcal{D}', \mathcal{I}') \leq \mathtt{Loc}(\mathcal{R}, \mathcal{D}, \mathcal{I}) \text{ if}$$
$$\forall x \in \mathcal{X}\ \exists x' \in \mathcal{X}'\ .\ x \leq x', \quad \text{where } \mathcal{X} \text{ ranges over } \mathcal{R}, \mathcal{D}, \mathcal{I}.$$

For example the following policy

$$\mathtt{Loc}(\{\langle ?\_, \mathit{Glaser}, \top\rangle\}, \{\langle ?\_, ?\_, \top \cdot (?\_, \mathit{acm\_1}) \cdot \top\rangle, \langle \mathit{Glaser}, ?\_, \top\rangle\}, \{\langle ?\_, \top\rangle\})$$

is smaller than the policy of location *dipl:Toyama87* (see page 7). The bottom policy is $\mathtt{Loc}(\{\langle ?\_, ?\_, \top\rangle\}, \{\langle ?\_, ?\_, \top\rangle\}, \{\langle ?\_, \top\rangle\})$ and there is no top policy.

4.5. Reduction Rules for Systems

The reduction rules for systems are presented in Fig. 6, where we assume standard structural congruences, i.e. associativity and commutativity of $\|$ and $+$, and neutrality of $\mathbf{0}$ for $\|$. This completes the definition of the operational semantics, hence more substantial examples are provided.

$$\frac{S_0 \longrightarrow S_1}{S_0 \parallel S_2 \longrightarrow S_1 \parallel S_2}\lceil\text{context}\rceil \qquad \frac{D \models Q}{\alpha\,[\mathtt{get}\,(Q, x)\,.P] \parallel D \longrightarrow \alpha\left[P\{^{D}/_{x}\}\right] \parallel D}\lceil\text{get}\rceil$$

$$\frac{\alpha\left[P\{^{\mathtt{rec}X.P}/_{X}\}\right] \parallel D \longrightarrow S}{\alpha\,[\mathtt{rec}X.P] \parallel D \longrightarrow S}\lceil\text{rec}\rceil \qquad \frac{e \Downarrow \Pi_{i=1}^{m} C_i^{p_i}}{\alpha\,[\mathtt{ins}\,(\ell, e)\,.P] \longrightarrow \alpha\,[P] \parallel \Pi_{i=1}^{m} C_i^{(\alpha,\ell)\cdot p_i}}\lceil\text{ins}\rceil$$

$$\frac{D \models Q}{\alpha\,[\mathtt{del}\,(Q, x)\,.P] \parallel D \longrightarrow \alpha\left[P\{^{D}/_{x}\}\right]}\lceil\text{del}\rceil \qquad \frac{\alpha\,[P_1] \parallel D \longrightarrow S}{\alpha\,[P_1 + P_2] \parallel D \longrightarrow S}\lceil\text{choose}\rceil$$

$$\frac{\alpha\left[P\{^{\ell}/_{a}\}\right] \parallel D \longrightarrow S \quad \mathcal{T}(\ell) \leq \mathtt{Loc}(\mathcal{R}, \mathcal{D}, \mathcal{I})}{\alpha\,[\exists a\colon \mathtt{Loc}(\mathcal{R}, \mathcal{D}, \mathcal{I}).P] \parallel D \longrightarrow S}\lceil\text{select}\rceil$$

Figure 6: Reduction rules for systems

Get and delete and insert. The get rule and delete rule are similar. Both rules match some data that occurs in parallel with the process containing the query pattern. The resulting data is substituted in the continuation process. The difference between the two is that for get the data in the context persists, whereas for delete the data is consumed.

The insert rule evaluates an expression to obtain some data. The data is then inserted in the location indicated. This is done by placing the data in parallel with the process and updating the provenance to indicate that most recently the data was written to the location by the given agent.

The following scenario demonstrates the execution of get, delete and insert operators. Suppose that two agents identified by *Sassone* and *Horne* contribute triples about Toyama. In the system configuration below *Sassone* has already contributed a triple and *Horne* is ready to contribute some misleading information. The misleading information is about a city classified as a Core City in Japan, also called Toyama. Mistaken identity of resources with similar properties is a well known problem for Linked Data [23].

$$\begin{array}{l}(\textit{Toyama akt:has-affiliation Tohoku\_University})^{(\textit{Sassone},\textit{Toyama})} \parallel \\ \textit{Horne}\,[\mathtt{ins}\,(\textit{Toyama}, (\textit{Toyama\_City akt:type Core\_city})^{\epsilon})\,.0]\end{array}$$

Consider a third agent identified by *Dezani*. This agent trusts contributions made by *Sassone* about Toyama, but notices that *Horne* has made a mistake. Using a provenance pattern the agent *Dezani* can remove the work of *Horne*, without affecting the contribution of *Sassone* and other agents. Furthermore, the contribution removed is reposted in the correct location.

$$\textit{Dezani}\left[\mathtt{del}\left(*\exists a.\exists b.\exists c.(a\;b\;c)^{(\textit{Horne},\textit{Toyama})\cdot\top}, x\right).\mathtt{ins}\,(\textit{Toyama\_City}, x)\,.0\right]$$

After three steps, the following data are obtained:

$$\begin{array}{l}(\textit{Toyama akt:has-affiliation Tohoku\_University})^{(\textit{Sassone},\textit{Toyama})} \parallel \\ (\textit{Toyama\_City akt:type Core\_city})^{(\textit{Dezani},\textit{Toyama\_City})\cdot(\textit{Horne},\textit{Toyama})}\end{array}$$

Notice that the correctly contributed triple has not been touched. The provenance of the new triple records the agents who affected its current and prior locations.

Note. Agents can lie about provenance only by inserting new data with a misleading provenance. This can be avoided by restricting the provenance of data to the the empty provenance within insert commands in the initial configuration of an agent. Intermediate operational steps, may include data with richer provenance, but the operational semantics ensure that this provenance is properly tracked.

Passing locations to continuations. Notice that the target location of insert can be a variable. This allows a name discovered by a query to be used. This effect is achieved by using existential quantifiers at the level of processes. The rule substitutes a location in place of the variable such that the process can perform a transition.

The rule also checks whether the substituted location has a suitable policy, though the function $\mathcal{T}$ which associates policies to locations. This function is also used in the type system in the next section. Details are not specified for where in a system this function is stored; however it is clear for security that it cannot be stored client-side where it would be available to agents. Thus type checking and execution must occur server-side in a system that implements this model.

The following example demonstrates an existential quantifier that binds a location variable in a query and an insert. Assume that $\mathtt{Loc}_0$ is some policy with the insert pair $\langle \alpha, \top \cdot (\textit{ACM}, ?\_)\rangle$, which allows α to insert triples originating with the *ACM*.

$$\begin{array}{l}\alpha\left[\exists a\colon \mathtt{Loc}_0.\Big(\mathtt{del}\left((a\ \textit{akt:has-author Toyama})^{(?\_,\textit{Toyama})\cdot\top}, x\right).\mathtt{ins}\,(a, x)\,.0\Big)\right] \parallel \\ (\textit{rkpres:CS132820 akt:has-author Toyama})^{(\textit{Glaser},\textit{Toyama})\cdot(\textit{ACM},\textit{acm\_1})}\end{array}$$

Assuming that the location *rkpres:CS132820* has the policy $\mathtt{Loc}_0$, this location is substituted to a when evaluating the delete. So the the system evolves to the following state.

$$\alpha\left[\mathtt{ins}\left(\begin{array}{l}\textit{rkpres:CS132820}, \\ (\textit{rkpres:CS132820 akt:has-author Toyama})^{(\textit{Glaser},\textit{Toyama})\cdot(\textit{ACM},\textit{acm\_1})}\end{array}\right).0\right]$$

The reduction of the insert command gives the following data:

$$(\textit{rkpres:CS132820 akt:has-author Toyama})^{(\alpha,\textit{rkpres:CS132820})\cdot(\textit{Glaser},\textit{Toyama})\cdot(\textit{ACM},\textit{acm\_1})}$$

An example of exists binding pattern in continuations is presented in the next section.

4.6. Dereferencing Revisited

This section returns to the scenario initially described in Section 2. The data and processes defined realise the scenario. This demonstrates the power of the calculus for precisely modelling and evaluating Linked Data.

Consider how the data was published. Initially some agent *CiteSeer* lifts the data from some source. The lifted data is then published at *cs_1* . This results in the following stored data, along with a huge amount of similarly annotated data.

$$(\mathit{Toyama}\ \mathit{akt{:}type}\ \mathit{akt{:}Person})^{(\mathit{CiteSeer},\mathit{cs\_1})\cdot \mathit{Lift\#}} \parallel$$
$$(\mathit{rkpres{:}CS132820}\ \mathit{akt{:}has\text{-}author}\ \mathit{Toyama})^{(\mathit{CiteSeer},\mathrm{source1})\cdot \mathit{Lift\#}}$$

Now some agent *RKBexplorer* dereferences the data and extracts the data that immediately refers to Toyama. The agent is also allowed to draw from another source *acm_1* , which for now is empty. The agent them publishes the combination of any data from both sources in the location *Toyama* . An agent that achieves this is presented below.

$$\mathit{RKBexplorer}\left[\begin{array}{l} \mathtt{get}\left(*\exists a,b.\left((\mathit{Toyama}\ a\ b)^{(?\_,\mathit{cs\_1})\cdot\top} \oplus (a\ b\ \mathit{Toyama})^{(?\_,\mathit{cs\_1})\cdot\top}\right), x\right). \\ \mathtt{get}\left(*\exists a,b.\left((\mathit{Toyama}\ a\ b)^{(?\_,\mathit{acm\_1})\cdot\top} \oplus (a\ b\ \mathit{Toyama})^{(?\_,\mathit{acm\_1})\cdot\top}\right), y\right). \\ \mathtt{ins}\,(\mathit{Toyama}, x \parallel y)\,.0 \end{array}\right]$$

Reducing the parallel composition of the above data and agent gives the following data:

$$(\mathit{Toyama}\ \mathit{akt{:}type}\ \mathit{akt{:}Person})^{(\mathit{RKBexplorer},\mathit{Toyama})\cdot(\mathit{CiteSeer},\mathit{cs\_1})\cdot \mathit{Lift\#}} \parallel$$
$$(\mathit{rkpres{:}CS132820}\ \mathit{akt{:}has\text{-}author}\ \mathit{Toyama})^{(\mathit{RKBexplorer},\mathit{Toyama})\cdot(\mathit{CiteSeer},\mathrm{source1})\cdot \mathit{Lift\#}}$$

Note that the original data is retained, so it now appears in two locations.

Consider an agent *Sassone* who consumes this data. The agent dereferences *Toyama* to discover one publication. The location of the publication is passed to the continuation process. The continuation dereferences the publication to find a paper cited by the original paper. The location of the cited paper is then passed to the continuation. The continuation process then dereferences the cited paper to find whether an author of the paper was also Toyama. The data consumed is a proof of a self-citation. The process is expressed as follows.

$$\mathit{Sassone}\left[\exists a\colon \mathtt{Loc}_1.\left(\begin{array}{l} \mathtt{get}\left((a\ \mathit{akt{:}has\text{-}author}\ \mathit{Toyama})^{(?\_,\mathit{Toyama})\cdot\top}, x\right). \\ \exists b\colon \mathtt{Loc}_1.\left(\begin{array}{l} \mathtt{get}\left((a\ \mathit{cites}\ b)^{(?\_,a)\cdot\top}, y\right). \\ \mathtt{get}\left((b\ \mathit{akt{:}has\text{-}author}\ \mathit{Toyama})^{(?\_,b)\cdot\top}, z\right). \\ \mathit{Demonstrate\_Self\_Citation}\,(x \parallel y \parallel z) \end{array}\right) \end{array}\right)\right]$$

Assume that $\mathtt{Loc}_1$ contains the access triple $\langle \mathit{Sassone}, ?\_, \top\rangle$ for reading, which gives *Sassone* full read access. Reducing the parallel composition of the agent *Sassone* with some data *D* matching the above queries gives:

$$D \parallel \mathit{Sassone}\,[\mathit{Demonstrate\_Self\_Citation}\,(D)]$$

Thus key processes for publishing and consuming Linked Data are captured.

5. A Type System for Provenance Based Access Control

A type system for the calculus is defined. The type system guarantees that access control policies for data are respected by processes run by agents. The access control policies are based on the provenance of the data. More precisely the typing rules assure that:

$$wf(\epsilon) \qquad \frac{\vdash_L \ell : \mathtt{Loc}(\mathcal{R}, \mathcal{D}, \mathcal{I}) \quad (\alpha, p) \in \mathcal{I} \quad wf(p)}{wf((\alpha, \ell) \cdot p)} \qquad \frac{wf(p)}{wf(f\# \cdot p)} \qquad \frac{wf(p) \quad wf(q)}{wf(p \vee q)}$$

Figure 7: Well-formed provenance traces

$$\frac{}{\vdash_D 0 : \pi} \lfloor D0 \rfloor \qquad \frac{wf(p) \quad p \Vdash \pi}{\vdash_D C^p : \pi} \lfloor Dt \rfloor \qquad \frac{\vdash_D D_0 : \pi \quad \vdash_D D_1 : \pi}{\vdash_D D_0 \parallel D_1 : \pi} \lfloor D\| \rfloor$$

Figure 8: Typing rules for stored data

1. the provenance traces of the tracked triples agree with the location policies;
2. getting, deleting and inserting operations are always done by agents that are authorised by the location policies, as formalised in Theorem 6.9.

For the first point it is handy to define well-formed provenance traces, see Fig. 7. A well-formed provenance is such that the provenance that follows each who-where pair appears for the agent in the policy of the location.

Location types are policies. The *principal type* of a location is the policy given by the function $\mathcal{T}$. A location must have all location types that are greater than or equal to the principal type of that location. This is achieved by the following axiom:

$$\frac{\mathcal{T}(\ell) \leq \mathtt{Loc}(\mathcal{R}, \mathcal{D}, \mathcal{I})}{\vdash_L \ell : \mathtt{Loc}(\mathcal{R}, \mathcal{D}, \mathcal{I})} \lfloor L\ell \rfloor$$

Since the location policies relate agents and patterns, patterns type stored data, queries and expressions. Figure 8 gives the rules for stored data: the empty data can have any pattern, a tracked triple with a well-formed provenance has all the patterns satisfied by its provenance and composed data must have the patterns of their components. Note that by rule $\lfloor Dt \rfloor$ each triple has many patterns, the smallest one being the provenance itself of the triple.

Figure 9 gives the typing rules for queries, where each triple pattern has all the provenance patterns that are bigger than or equal to its own provenance pattern.

In order to type expressions, environments (ranged over by Γ) associating data variables to patterns are needed, see Fig. 10. The only interesting rule is $\lfloor Ef \rfloor$: since in

$$\frac{\pi \leq \pi'}{\vdash_Q C^\pi : \pi'} \lfloor Qt \rfloor \qquad \frac{\vdash_Q Q : \pi}{\vdash_Q \exists a.Q : \pi} \lfloor Q\exists \rfloor \qquad \frac{\vdash_Q Q : \pi}{\vdash_Q *Q : \pi} \lfloor Q* \rfloor$$

$$\frac{\vdash_Q Q_0 : \pi \quad \vdash_Q Q_1 : \pi}{\vdash_Q Q_0 \oplus Q_1 : \pi} \lfloor Q\oplus \rfloor \qquad \frac{\vdash_Q Q_0 : \pi \quad \vdash_Q Q_1 : \pi}{\vdash_Q Q_0 \otimes Q_1 : \pi} \lfloor Q\otimes \rfloor$$

Figure 9: Typing rules for queries

$$\frac{}{\Gamma, x:\pi \vdash_E x:\pi}\lfloor Ev \rfloor \qquad \frac{\vdash_D D:\pi}{\Gamma \vdash_E D:\pi}\lfloor Et \rfloor \qquad \frac{\Gamma \vdash_E e:\pi}{\Gamma \vdash_E f(e): f\# \cdot \pi \vee \pi}\lfloor Ef \rfloor$$

$$\frac{\Gamma \vdash_E e_0:\pi \quad \Gamma \vdash_E e_1:\pi}{\Gamma \vdash_E e_0 \parallel e_1:\pi}\lfloor E\| \rfloor \qquad \frac{\Gamma \vdash_E e:\pi \quad \pi \leq \pi'}{\Gamma \vdash_E e:\pi'}\lfloor Esub \rfloor$$

Figure 10: Typing rules for expressions

evaluating $f(D)$ not all the provenance traces of the tracked triples in D are prefixed by $f\#$ (see rule $\lfloor\text{fun}\rfloor$ in Fig. 5), only the pattern $f\#\cdot\pi\vee\pi$ can be deduced for $f(e)$ knowing that e has pattern π.

For typing processes, environments (ranged over by Θ) associating data variables to patterns and location variables to location types are needed. To derive for a location all locations types that are bigger than or equal to the principal location type of that location, a weakening of rule $\lfloor L\ell \rfloor$ and a standard axiom are handy:

$$\frac{\vdash_L \ell : \mathsf{Loc}(\mathcal{R},\mathcal{D},\mathcal{I})}{\Theta \vdash_L \ell : \mathsf{Loc}(\mathcal{R},\mathcal{D},\mathcal{I})}\lfloor L\ell w \rfloor \qquad \frac{\mathsf{Loc}(\mathcal{R}',\mathcal{D}',\mathcal{I}') \leq \mathsf{Loc}(\mathcal{R},\mathcal{D},\mathcal{I})}{\Theta, a : \mathsf{Loc}(\mathcal{R}',\mathcal{D}',\mathcal{I}') \vdash_L a : \mathsf{Loc}(\mathcal{R},\mathcal{D},\mathcal{I})}\lfloor La \rfloor$$

Processes are built from get, delete and insert operations on locations. Each operation can be allowed or disallowed by location policies according to the agent who is willing to act. For this reason, processes are typed by agents, meaning that a process typed by an agent contains only operations that the agent is authorised to do (see Fig. 11). In rules $\lfloor Pg \rfloor$ and $\lfloor Pd \rfloor$ the patterns of the queries must start with a who-where pair, so that the queries are limited to exactly one location (σ can be ?_ and therefore the triples can be inserted by an arbitrary agent). The condition $\langle\alpha,\sigma,\pi\rangle \in \mathcal{R}$ assures that the getting agrees with the location policy. Similarly for $\langle\alpha,\sigma,\pi\rangle \in \mathcal{D}$. The rule for typing insertion simply checks that α is allowed to insert data with provenance π in the location λ by the location policy.

Since systems do not contain free variables, environments to type them are not needed, see Fig. 12. Rule $\lfloor S\,d \rfloor$ validates well-typed stored data. The typing of an agent checks that the process can be typed by the agent name, see rule $\lfloor S\,\alpha \rfloor$.

5.1. Examples of Typed Systems

The examples of systems in the previous section are revisited to consider the effect of typing.

Consider the first scenario in Sec. 4.5. There are three agents interacting with data in a location. Assume that the location *Toyama* has the insert pair $\langle \mathit{Horne}, \epsilon\rangle$ and the access triple $\langle \mathit{Dezani}, \mathit{Horne}, \top\rangle$ for deleting. Also assume that the location *Toyama_City* has the insert pair $\langle \mathit{Dezani}, \top\rangle$. These assumptions allow the system to be typed. The agent *Horne* can write any fresh data to the location and the agent *Dezani* can remove any data contributed by *Horne*. However without the insert pair $\langle \mathit{Horne}, \epsilon\rangle$, the system is not well typed, so the agent *Horne* could not have inserted the misleading triple.

Consider the second scenario in Sec. 4.5. Assume that the location *Toyama* has access triple $\langle \alpha, ?\_, \top\rangle$ for deleting. Then the agent α is well typed. Notice that the

$$\frac{\begin{array}{c}\Theta, x : (\sigma, \lambda) \cdot \pi \vdash_P P : \alpha \quad \langle \alpha, \sigma, \pi \rangle \in \mathcal{R} \\ \Theta \vdash_L \lambda : \mathtt{Loc}(\mathcal{R}, \mathcal{D}, \mathcal{I}) \quad \vdash_Q Q : (\sigma, \lambda) \cdot \pi\end{array}}{\Theta \vdash_P \mathtt{get}\,(Q, x)\,.P : \alpha} \lfloor Pg \rfloor$$

$$\frac{\begin{array}{c}\Theta, x : (\sigma, \lambda) \cdot \pi \vdash_P P : \alpha \quad \langle \alpha, \sigma, \pi \rangle \in \mathcal{D} \\ \Theta \vdash_L \lambda : \mathtt{Loc}(\mathcal{R}, \mathcal{D}, \mathcal{I}) \quad \vdash_Q Q : (\sigma, \lambda) \cdot \pi\end{array}}{\Theta \vdash_P \mathtt{del}\,(Q, x)\,.P : \alpha} \lfloor Pd \rfloor$$

$$\frac{\Theta \vdash_L \lambda : \mathtt{Loc}(\mathcal{R}, \mathcal{D}, \mathcal{I}) \quad \Theta \vdash_P P : \alpha \quad \Gamma \vdash_E e : \pi \quad \Gamma \subseteq \Theta \quad (\alpha, \pi) \in \mathcal{I}}{\Theta \vdash_P \mathtt{ins}\,(\lambda, e)\,.P : \alpha} \lfloor Pi \rfloor$$

$$\frac{}{\Theta \vdash_P 0 : \alpha} \lfloor P0 \rfloor \qquad \frac{}{\Theta \vdash_P X : \alpha} \lfloor Pv \rfloor \qquad \frac{\Theta \vdash_P P : \alpha}{\Theta \vdash_P \mathtt{rec} X.P : \alpha} \lfloor Pr \rfloor$$

$$\frac{\Theta \vdash_P P_0 : \alpha \quad \Theta \vdash_P P_1 : \alpha}{\Theta \vdash_P P_0 + P_1 : \alpha} \lfloor P{+} \rfloor \qquad \frac{\Theta, a\colon \mathtt{Loc}(\mathcal{R}, \mathcal{D}, \mathcal{I}) \vdash_P P : \alpha}{\Theta \vdash_P \exists a\colon \mathtt{Loc}(\mathcal{R}, \mathcal{D}, \mathcal{I}).P : \alpha} \lfloor P\exists \rfloor$$

Figure 11: Typing rules for processes

$$\frac{}{\vdash_S \mathbf{0}} \lfloor S\,\mathbf{0} \rfloor \qquad \frac{\vdash_D D : \pi}{\vdash_S D} \lfloor S\,d \rfloor \qquad \frac{\vdash_P P : \alpha}{\vdash_S \alpha\,[P]} \lfloor S\,\alpha \rfloor \qquad \frac{\vdash_S S \quad \vdash_S S'}{\vdash_S S \parallel S'} \lfloor S \parallel \rfloor$$

Figure 12: Typing rules for systems

annotation on the select quantifier is sufficient to guarantee that the location discovered by the query is of the correct type to enable the insert to be triggered. Thus assuming that the location *rkpres:CS132820* has the insert pair $\langle \alpha, \top \cdot (ACM, ?\_) \rangle$, the system will reduce since the dynamic check in the reduction rule enforces that the location discovered matches the annotation. Thus the static typing relies on the dynamic type check for that location.

Consider the scenario in Sec. 4.6. Assume that the locations *acm_1* and *cs_1* have the access triple $\langle RKBexplorer, ?\_, \top \rangle$ for getting, and that the location *Toyama* has the insert pair $\langle RKBexplorer, (?\_, cs\_1) \cdot \top \vee (?\_, acm\_1) \cdot \top \rangle$ and the access triple $\langle Sassone, RKBexplorer, \top \rangle$ for getting. Then both agents are well typed. For agent *Sassone* the dynamic checks in the reduction rules assure that the discovered locations indeed allow *Sassone* to read, as indicated by the annotations.

5.2. Type Inference

The given type assignment system can be easily made syntax directed in order to get a type inference algorithm. It is enough:

- to replace in the second clause of the definition of well-formed provenance traces and in rules $\lfloor Pg \rfloor$, $\lfloor Pd \rfloor$, and $\lfloor Pi \rfloor$ the membership condition with a condition which takes subtyping into account,
- to eliminate the subtyping in rules $\lfloor L\ell \rfloor$, $\lfloor La \rfloor$, $\lfloor Qt \rfloor$,

$$\frac{\vdash_L \ell : \mathtt{Loc}(\mathcal{R},\mathcal{D},\mathcal{I}) \quad (\alpha, p) \in_{\leq} \mathcal{I} \quad wf(p)}{wf((\alpha,\ell)\cdot p)} \qquad \frac{}{\vdash_L \ell : \mathcal{T}(\ell)}\lfloor L\ell' \rfloor$$

$$\frac{}{\Theta, a : \mathtt{Loc}(\mathcal{R},\mathcal{D},\mathcal{I}) \vdash_L a : \mathtt{Loc}(\mathcal{R},\mathcal{D},\mathcal{I})}\lfloor La' \rfloor \qquad \frac{wf(p)}{\vdash_D C^p : p}\lfloor Dt' \rfloor$$

$$\frac{}{\vdash_Q C^\pi : \pi}\lfloor Qt' \rfloor \qquad \frac{\vdash_Q Q_0 : \pi_0 \quad \vdash_Q Q_1 : \pi_1}{\vdash_Q Q_0 \oplus Q_1 : \pi_0 \vee \pi_1}\lfloor Q\oplus' \rfloor$$

$$\frac{\vdash_Q Q_0 : \pi_0 \quad \vdash_Q Q_1 : \pi_1}{\vdash_Q Q_0 \otimes Q_1 : \pi_0 \vee \pi_1}\lfloor Q\otimes' \rfloor \qquad \frac{\Gamma \vdash_E e_0 : \pi_0 \quad \Gamma \vdash_E e_1 : \pi_1}{\Gamma \vdash_E e_0 \parallel e_1 : \pi_0 \vee \pi_1}\lfloor E\|' \rfloor$$

$$\frac{\begin{array}{c}\Theta, x : (\sigma,\lambda)\cdot\pi \vdash_P P : \alpha \quad \Theta \vdash_L \lambda : \mathtt{Loc}(\mathcal{R},\mathcal{D},\mathcal{I}) \\ \vdash_Q Q : (\sigma,\lambda)\cdot\pi \quad \langle\alpha,\sigma,\pi\rangle \in_{\leq} \mathcal{R}\end{array}}{\Theta \vdash_P \mathtt{get}\,(Q,x)\,.P : \alpha}\lfloor Pg' \rfloor$$

$$\frac{\begin{array}{c}\Theta, x : (\sigma,\lambda)\cdot\pi \vdash_P P : \alpha \quad \Theta \vdash_L \lambda : \mathtt{Loc}(\mathcal{R},\mathcal{D},\mathcal{I}) \\ \vdash_Q Q : (\sigma,\lambda)\cdot\pi \quad \langle\alpha,\sigma,\pi\rangle \in_{\leq} \mathcal{D}\end{array}}{\Theta \vdash_P \mathtt{del}\,(Q,x)\,.P : \alpha}\lfloor Pd' \rfloor$$

$$\frac{\Theta \vdash_L \lambda : \mathtt{Loc}(\mathcal{R},\mathcal{D},\mathcal{I}) \quad \Theta \vdash_P P : \alpha \quad \Gamma \vdash_E e : \pi \quad \Gamma \subseteq \Theta \quad (\alpha,\pi) \in_{\leq} \mathcal{I}}{\Theta \vdash_P \mathtt{ins}\,(\lambda, e)\,.P : \alpha}\lfloor Pi' \rfloor$$

Figure 13: Syntax directed versions of provenance well-formedness and of typing rules

- to build unions in rules $\lfloor Q\oplus \rfloor$, $\lfloor Q\otimes \rfloor$, $\lfloor E\| \rfloor$, and
- to cancel rule $\lfloor Esub \rfloor$.

More precisely the new clause of provenance well-formedness and the new typing rules are shown in Fig. 13, where for $\mathcal{X}$ ranging over $\mathcal{R}$, $\mathcal{D}$ and $\mathcal{I}$, we define:

$$x \in_{\leq} \mathcal{X} \text{ if there is } x' \in \mathcal{X} \text{ such that } x \leq x'$$

in agreement with the pre-order on policies defined in Sec. 4.4. In this way each location is typed with the least policy, and each data, query and expression is typed with the least provenance pattern. This assures that the new membership condition is satisfied whenever possible. The unique provenance pattern of a query is used to type the data variable that is bound in a get or in a del operator, i.e. if $\vdash_Q Q : \pi$, then in the inference of a type for $\mathtt{get}\,(Q,x)\,.P$ or $\mathtt{del}\,(Q,x)\,.P$, the premise $x : \pi$ is used for getting a type for P.

This shows that typing of closed systems is decidable when the pre-order on access tripes and insert pairs is decidable, which in turn is assured by the decidability of the pre-order on patterns (see Sec. 4.1).

6. Properties

This section verifies that the type system is correctly defined. An essential subject reduction theorem verifies that typing can be statically checked. Finally, it is proven that the location policies are indeed enforced for well-typed systems.

This section starts as usual with inversion lemmas whose proofs are standard.

Lemma 6.1 (Inversion Lemma for Data)**.** *1. If $\vdash_D C^p : \pi$, then $wf(p)$ and $p \Vdash \pi$.*

2. If $\vdash_D D_0 \parallel D_1 : \pi$, then $\vdash_D D_0 : \pi$ and $\vdash_D D_1 : \pi$.

Lemma 6.2 (Inversion Lemma for Queries)**.** *1. If $\vdash_Q C^\pi : \pi'$, then $\pi \leq \pi'$.*

2. If $\vdash_Q Q_0 \oplus Q_1 : \pi$, then $\vdash_Q Q_0 : \pi$ and $\vdash_Q Q_1 : \pi$.

3. If $\vdash_Q Q_0 \otimes Q_1 : \pi$, then $\vdash_Q Q_0 : \pi$ and $\vdash_Q Q_1 : \pi$.

4. If $\vdash_Q \exists a.Q : \pi$, then $\vdash_Q Q : \pi$.

5. If $\vdash_Q *Q : \pi$, then $\vdash_Q Q : \pi$.

Lemma 6.3 (Inversion Lemma for Expressions)**.** *1. If $\Gamma \vdash_E x : \pi$, then $\Gamma = \Gamma', x : \pi'$ and $\pi' \leq \pi$.*

2. If $\Gamma \vdash_E D : \pi$, then $\vdash_D D : \pi$.

3. If $\Gamma \vdash_E f(e) : \pi$, then $\Gamma \vdash_E e : \pi'$ and $f\# \cdot \pi' \vee \pi' \leq \pi$.

4. If $\Gamma \vdash_E e_0 \parallel e_1 : \pi$, then $\Gamma \vdash_E e_0 : \pi$ and $\Gamma \vdash_E e_1 : \pi$.

Lemma 6.4 (Inversion Lemma for Processes)**.** *1. If $\Theta \vdash_P \texttt{get}(Q, x).P : \alpha$, then $\Theta, x : (\sigma, \lambda) \cdot \pi \vdash_P P : \alpha$ and $\Theta \vdash_L \lambda : \texttt{Loc}(\mathcal{R}, \mathcal{D}, \mathcal{I})$ and $\vdash_Q Q : (\sigma, \lambda) \cdot \pi$ and $\langle \alpha, \sigma, \pi \rangle \in \mathcal{R}$.*

2. If $\Theta \vdash_P \texttt{del}(Q, x).P : \alpha$, then $\Theta, x : (\sigma, \lambda) \cdot \pi \vdash_P P : \alpha$ and $\Theta \vdash_L \lambda : \texttt{Loc}(\mathcal{R}, \mathcal{D}, \mathcal{I})$ and $\vdash_Q Q : (\sigma, \lambda) \cdot \pi$ and $\langle \alpha, \sigma, \pi \rangle \in \mathcal{D}$.

3. If $\Theta \vdash_P \texttt{ins}(\lambda, e).P : \alpha$, then $\Theta \vdash_L \lambda : \texttt{Loc}(\mathcal{R}, \mathcal{D}, \mathcal{I})$ and $\Theta \vdash_P P : \alpha$ and $\Gamma \vdash_E e : \pi$ and $\Gamma \subseteq \Theta$ and $\langle \alpha, \pi \rangle \in \mathcal{I}$.

4. If $\Theta \vdash_P P_0 + P_1 : \alpha$, then $\Theta \vdash_P P_0 : \alpha$ and $\Theta \vdash_P P_1 : \alpha$.

5. If $\Theta \vdash_P \texttt{rec}X.P : \alpha$, then $\Theta \vdash_P P : \alpha$.

6. If $\Theta \vdash_P \exists a \colon \texttt{Loc}(\mathcal{R}, \mathcal{D}, \mathcal{I}).P : \alpha$, then $\Theta, a \colon \texttt{Loc}(\mathcal{R}, \mathcal{D}, \mathcal{I}) \vdash_P P : \alpha$.

Lemma 6.5 (Inversion Lemma for Systems)**.** *1. If $\vdash_S D$, then $\vdash_D D : \pi$, for some π.*

2. If $\vdash_S \alpha[P]$, then $\vdash_P P : \alpha$.

3. If $\vdash_S S \parallel S'$, then $\vdash_S S$ and $\vdash_S S'$.

The proof of subject reduction (Theorem 6.8) is based on the agreement between the typing of stored data and the pre-order on patterns, the satisfaction of queries and the reduction of expressions, as formalised in the following Lemma.

Lemma 6.6 (Key)**.** *1. If $\vdash_D D : \pi$ and $\pi \leq \pi'$, then $\vdash_D D : \pi'$.*

2. *If* $\vdash_Q Q : \pi$ *and* $D \models Q$ *and* $\vdash_S D$*, then* $\vdash_D D : \pi$.

3. *If* $\vdash_E e : \pi$ *and* $e \Downarrow D$*, then* $\vdash_D D : \pi$.

Proof. (1). By induction on the definition of $\vdash_D$ using Proposition 4.1 for rule $\lfloor Dt \rfloor$.
(2). By induction on the definition of $\models$.

Rule $\lceil$QAx$\rceil$ $\quad \dfrac{p \Vdash \pi'}{C^p \models C^{\pi'}}$

By Lemma 6.2(1) $\vdash_Q C^{\pi'} : \pi$ implies $\pi' \leq \pi$. Proposition 4.1 and $p \Vdash \pi'$ give $p \Vdash \pi$. By Lemmas 6.5(1) and 6.1(1) $\vdash_S C^p$ implies $wf(p)$. So $\vdash_D C^p : \pi$ can be derived using rule $\lfloor Dt \rfloor$.

Rule $\lceil$QChL$\rceil$ $\quad \dfrac{D \models Q_0}{D \models Q_0 \oplus Q_1}$

By Lemma 6.2(2) $\vdash_Q Q_0 \oplus Q_1 : \pi$ implies $\vdash_Q Q_0 : \pi$ and $\vdash_Q Q_1 : \pi$. The induction applied to $\vdash_Q Q_0 : \pi$ and $D \models Q_0$ gives $\vdash_D D : \pi$.

Rule $\lceil$QChL$\rceil$ $\quad \dfrac{D_0 \models Q_0 \quad D_1 \models Q_1}{D_0 \parallel D_1 \models Q_0 \otimes Q_1}$

By Lemma 6.2(3) $\vdash_Q Q_0 \otimes Q_1 : \pi$ implies $\vdash_Q Q_0 : \pi$ and $\vdash_Q Q_1 : \pi$. Induction gives $\vdash_D D_0 : \pi$ and $\vdash_D D_1 : \pi$ and then $\vdash_D D_0 \parallel D_1 : \pi$ can be derived using rule $\lfloor D\parallel \rfloor$.
The remaining cases are similar and simpler.
(3). By induction on $\Downarrow$. The only interesting case is

$$\frac{|f|(\Pi_{i=1}^m C_i) \downarrow \Pi_{j=1}^n \hat{C}_j \quad q = f\# \cdot \bigvee_{i=1}^m p_i}{f(\Pi_{i=1}^m C_i^{p_i} \parallel D) \Downarrow \Pi_{j=1}^n \hat{C}_j^q \parallel D}$$

By Lemma 6.3(3) $\vdash_E f(\Pi_{i=1}^m C_i^{p_i} \parallel D) : \pi$ implies $\vdash_E \Pi_{i=1}^m C_i^{p_i} \parallel D : \pi'$ and $f\# \cdot \pi' \vee \pi' \leq \pi$. Lemmas 6.3(2) and 6.1(2) give $\vdash_D C_i^{p_i} : \pi'$ for all i $(1 \leq i \leq m)$ and $\vdash_D D : \pi'$. Lemma 6.1(1) implies $wf(p_i)$ and $p_i \Vdash \pi'$ for all i $(1 \leq i \leq m)$, which allow to derive $wf(q)$ and $\bigvee_{i=1}^m p_i \Vdash \pi'$ and then $q \Vdash f\# \cdot \pi'$ and $q \Vdash \pi$. Lastly $\vdash_D \hat{C}_j^q : \pi$ for all j $(1 \leq j \leq n)$ using rule $\lfloor Dt \rfloor$. Point (1) gives $\vdash_D D : \pi$ and $\vdash_D \Pi_{j=1}^n \hat{C}_j^q \parallel D : \pi$ can be derived using rule $\lfloor D\parallel \rfloor$. $\square$

The following substitution lemma has a simple proof since data variables in processes can only occur inside expressions.

Lemma 6.7 (Substitution Lemma)**.** *If* $\Theta, x : \pi \vdash P : \alpha$ *and* $\vdash D : \pi$*, then* $\Theta \vdash P\{^D/_x\} : \alpha$.

The preservation of typing under reduction can now be shown.

Theorem 6.8 (Subject Reduction)**.** *If* $\vdash_S S$ *and* $S \longrightarrow S'$*, then* $\vdash_S S'$.

Proof. By induction on $\longrightarrow$.

Rule $\lceil$Pg$\rceil$ $\quad \dfrac{D \models Q}{\alpha\,[\mathtt{get}\,(Q, x)\,.P] \parallel D \longrightarrow \alpha \left[P\{^D/_x\}\right] \parallel D}$

By Lemma 6.5(3) $\vdash_S \alpha[\mathtt{get}(Q,x).P] \parallel D$ implies $\vdash_S \alpha[\mathtt{get}(Q,x).P]$ and $\vdash_S D$. By Lemma 6.5(2) $\vdash_S \alpha[\mathtt{get}(Q,x).P]$ implies $\vdash_P \mathtt{get}(Q,x).P : \alpha$. Then $x : (\sigma,\ell)\cdot\pi \vdash_P P : \alpha$ and $\vdash_L \ell : \mathtt{Loc}(\mathcal{R},\mathcal{D},\mathcal{I})$ and $\vdash_Q Q : (\sigma,\ell)\cdot\pi$ and $\langle\alpha,\sigma,\pi\rangle \in \mathcal{R}$ by Lemma 6.4(1). By Lemma 6.6(2) $\vdash_D D : (\sigma,\ell)\cdot\pi$, which implies $\vdash_P P\{^D/_x\} : \alpha$ by Lemma 6.7. Lastly $\vdash_S \alpha\left[P\{^D/_x\}\right] \parallel D$ can be derived using rules $\lfloor S\,\alpha\rfloor$, $\lfloor S\,d\rfloor$ and $\lfloor S\parallel\rfloor$.

Rule $\lceil \mathrm{Pi}\rceil$ $$\frac{e \Downarrow \Pi_{i=1}^{m} C_i^{p_i}}{\alpha[\mathtt{ins}(\ell,e).P] \longrightarrow \alpha[P] \parallel \Pi_{i=1}^{m} C_i^{(\alpha,\ell)\cdot p_i}}$$

By Lemma 6.5(2) $\vdash_S \alpha[\mathtt{ins}(l,e).P]$ implies $\vdash_P \mathtt{ins}(l,e).P : \alpha$. Then $\vdash_L \ell : \mathtt{Loc}(\mathcal{R},\mathcal{D},\mathcal{I})$ and $\vdash_P P : \alpha$ and $\vdash_E e : \pi$ and $(\alpha,\pi) \in \mathcal{I}$ by Lemma 6.4(3). By Lemma 6.6(3) $\vdash_D \Pi_{i=1}^{m} C_i^{p_i} : \pi$. This implies $wf(p_i)$ and $p_i \Vdash \pi$ for all i $(1 \le i \le m)$ by Lemma 6.1(2) and (1). The well-formedness $wf((\alpha,\ell)\cdot p_i)$ follows from $wf(p_i)$ and $\vdash_L \ell : \mathtt{Loc}(\mathcal{R},\mathcal{D},\mathcal{I})$ and $(\alpha,\pi) \in \mathcal{I}$. Then $\vdash_S C_i^{(\alpha,\ell)\cdot p_i}$ for all i $(1 \le i \le m)$ can be derived using rules $\lfloor Dt\rfloor$ and $\lfloor S\,d\rfloor$. Lastly $\vdash_S \alpha[P] \parallel \Pi_{i=1}^{m} C_i^{(\alpha,\ell)\cdot p_i}$ can be derived using rules $\lfloor S\,\alpha\rfloor$ and $\lfloor S\parallel\rfloor$.

□

This section ends by showing that reducing a well-typed system:

1. If an agent α gets a tracked triple $C^{(\beta,\ell)\cdot p}$, then the getting policy of ℓ contains the triple $\langle\alpha,\beta,p\rangle$.
2. If an agent α deletes a tracked triple $C^{(\beta,\ell)\cdot p}$, then the deleting policy of ℓ contains the triple $\langle\alpha,\beta,p\rangle$.
3. If an agent α inserts a tracked triple C^p, then the inserting policy of ℓ contains the pair $\langle\alpha,p\rangle$.

More precisely the following theorem holds:

Theorem 6.9. *1. If $\vdash_S \alpha[\mathtt{get}(Q,x).P]$ and $C^{(\beta,\ell)\cdot p} \parallel D \models Q$, then $\vdash_L \ell : \mathtt{Loc}(\mathcal{R},\mathcal{D},\mathcal{I})$ and $\langle\alpha,\beta,p\rangle \in \mathcal{R}$.*

2. If $\vdash_S \alpha[\mathtt{del}(Q,x).P]$ and $C^{(\beta,\ell)\cdot p} \parallel D \models Q$, then $\vdash_L \ell : \mathtt{Loc}(\mathcal{R},\mathcal{D},\mathcal{I})$ and $\langle\alpha,\beta,p\rangle \in \mathcal{D}$.

3. If $\vdash_S \alpha[\mathtt{ins}(\ell,e).P]$ and $e \Downarrow C^p \parallel D$, then $\vdash_L \ell : \mathtt{Loc}(\mathcal{R},\mathcal{D},\mathcal{I})$ and $\langle\alpha,p\rangle \in \mathcal{I}$.

Proof. (1). By Lemmas 6.5(2) and 6.4(1) $\vdash_S \alpha[\mathtt{get}(Q,x).P]$ implies $x : (\sigma,\ell')\cdot\pi \vdash_P P : \alpha$ and $\vdash_L \ell' : \mathtt{Loc}(\mathcal{R},\mathcal{D},\mathcal{I})$ and $\vdash_Q Q : (\sigma,\ell')\cdot\pi$ and $\langle\alpha,\sigma,\pi\rangle \in \mathcal{R}$. By Lemmas 6.6(2) and 6.1(2), $C^{(\beta,\ell)\cdot p} \parallel D \Vdash Q$ and $\vdash_Q Q : (\sigma,\ell')\cdot\pi$ give $\vdash_D C^{(\beta,\ell)\cdot p} : (\sigma,\ell')\cdot\pi$. This gives $(\beta,\ell)\cdot p \Vdash (\sigma,\ell')\cdot\pi$ by Lemma 6.1(1), which implies $\beta \le \sigma$, $\ell = \ell'$ and $p \le \pi$ by Proposition 4.1 and definition of $\le$. Lastly $\mathtt{Loc}(\mathcal{R},\mathcal{D},\mathcal{I}) \le \mathtt{Loc}(\mathcal{R} \cup \{\langle\alpha,\beta,p\rangle\},\mathcal{D},\mathcal{I})$ and then $\vdash_L \ell : \mathtt{Loc}(\mathcal{R} \cup \{\langle\alpha,\beta,p\rangle\},\mathcal{D},\mathcal{I})$ can be derived using rule $\lfloor L\ell\rfloor$.
(2). Similar to the proof of (1).
(3). By Lemmas 6.5(2) and 6.4(3) $\vdash_S \alpha[\mathtt{ins}(\ell,e).P]$ implies $\vdash_L \ell : \mathtt{Loc}(\mathcal{R},\mathcal{D},\mathcal{I})$ and $\vdash_P P : \alpha$ and $\vdash_E e : \pi$ and $\langle\alpha,\pi\rangle \in \mathcal{I}$. By Lemmas 6.6(3) and 6.1(2), $e \Downarrow C^p \parallel D$ and $\vdash_E e : \pi$ imply $\vdash_D C^p : \pi$. This gives $p \Vdash \pi$ by Lemma 6.1(1) and $p \le \pi$ by Proposition 4.1. Lastly $\mathtt{Loc}(\mathcal{R},\mathcal{D},\mathcal{I}) \le \mathtt{Loc}(\mathcal{R},\mathcal{D},\mathcal{I} \cup \{\langle\alpha,p\rangle\})$ and then $\vdash_L \ell : \mathtt{Loc}(\mathcal{R},\mathcal{D},\mathcal{I} \cup \{\langle\alpha,p\rangle\})$ can be derived using rule $\lfloor L\ell\rfloor$. □

Thus a well-typed system reduces by respecting the access control dictated by the location policies. More precisely if the operational semantics had rules that did dynamic checks that the policy is satisfied, then these checks would always succeed by reducing well-typed systems.

7. Related Work

The Web of Linked Data, or simply '*Linked Data*' [57], which draws experience from ideas explored for the '*Semantic Web*,' is a composite and exciting movement of ideas, applications and techniques arising around the Web as we know it. It originates from the desire of moving away from a web of documents to a web of data. This is a web where links are not simply a technical device to reach documents, but rather a way to attach meaning to data and establish semantic connections between pieces of information. Its characterising features include the use of dereferenceable URIs to represent atomic information and RDF to represent their relationships [5].

So far the Linked Data community has chiefly focussed on publishing large datasets, importing them from various sources to RDF, and on designing applications that make use of such data in tools of popular impact. Most of the theoretical work has been devoted to developing *ontologies* and their formulation using *description logics* [41]. An equally considerable effort has been dedicated to the study of the Web as a science [6], including (social) *network dynamics* (cf. e.g. [3]).

Ours is among the first papers to propose a language-based semantics for Linked Data. Indeed, to the best of our knowledge, the first formal operational model of (computation over) Linked Data is [37], followed by [40] and [39, 38]. Our calculus here is original in its all provenance elements, but bears a close resemblance to that of [37] for its linked data part. An alternative formal model of dereferencing URIs in linked data is provided by Jeffrey and Patel-Schneider [42].

As a research theme, provenance covers a very wide spectrum of problems, techniques and approaches. With a recent flourishing of activity, and most of its literature published in the last four years, it is a field in flux, rather complex to review systematically. A comprehensive survey and full literature analysis is therefore beyond our present scope, and can be found e.g. in [46]. Here only the main components of the provenance movement are mentioned, as well as those items of work more directly related to the present paper.

Historically, provenance emerged from issues in *databases* (cf. [19] for an overview), where the need arose to characterise the source of information [62] so as to justify the answer to complex queries [11]. In that context, provenance developed an elegant mathematical theory, which is arguably the pinnacle of its theoretical development, where symbolic polynomials on semi-rings are used to represent abstractly computations and their sources (cf., e.g., [30]).

The research on provenance moved out of its databases origins to find wider and deeper applications in workflow systems for eScience and for web computing (cf. [58] for a survey). This represented a significant extension in scope, which brought to a consolidation of ideas and a generalisation of techniques, as well as the design of pilot systems and infrastructures, including Kepler [8], VDL [21], Taverna [64] and PA-SOA [31], and the formulation of the concept of 'provenance-aware' application [45].

In particular, provenance became a concept in distributed computing, where it developed a need for standardisation [47] and interchange of data [48] and processes [43]. Among other trends, provenance recently acquired a trust and security dimension (cf. [4]), which is relevant to this work.

According to its most liberal definition, the provenance of a piece of data is the process that led to that piece of data. A concrete approach to this notion represents provenance via directed acyclic graphs, where nodes are data and edges are data derivations [44, 47]. Our work is compatible with such a kind of model, yet we use traces that can be viewed as trees and leave the generalisation to graphs to future work. The 'why-' 'where-' and 'how-provenance' notions from databases [62, 22, 11] are also well represented in our calculus. An alternative approach that focusses on the use of user-provided metadata to record the provenance information is the 'provenance-as-annotations' paradigm [54].

The present work is more closely related to the application to provenance of formal methods and analysis techniques. Buneman *et al* in [10] study the expressive power of provenance in database queries, Cheney *et al* [17] introduce a formal notion of provenance traces and study some of its properties, including computability, consistency and fidelity issues. Cheney *et al* [18] argue that dependency analysis techniques provide a formal foundation for forms of provenance that are intended to show how the output of a query depends on its input. A formal account of the interchange provenance data model of [47] has been given in [44]. A precursor to this paper is [59], where Sassone and Souilah present a π-calculus where the provenance channel communication is traced. As here, the provenance model is a tree, whilst the data model only includes names like in the π calculus.

Provenance for Linked Data has already been considered in the literature. A first line of work is concerned with representing provenance using RDF and to query and reason over provenance using Linked Data techniques (see, e.g., [63]). Various papers deal with annotated RDF, and consider also annotations providing information on provenance. Udrea *et al* [60] present a formal declarative semantics for RDF annotated by members of a partially ordered set. Buneman and Kostylev [12] develop an algebra for computing annotations on inferred triples. Zimmermann et al [65] present a detailed and systematic approach for combining multiple annotation domains into a new single complex domain. Closer to our interests, Carroll *et al* [13] introduced *named graphs* as a first approach to where provenance for RDF triples. In [25] Flouris *et al* rely on coloured RDF triples represented as quadruples to capture 'where' provenance. Halpin and Cheney [32] consider a provenance model for SPARQL queries and updates to data stores involving named graphs, whose purpose is to provide a record of how the raw data in a dataset has changed over time. Our work extends named graphs is several significant ways.

The standard approach to the semantics of SPARQL is denotational, see for example [52]. In this paper we use instead the operational semantics of [39], which better fits with the reduction rules of our calculus. We remark that the algebraic axiomatisation of SPARQL queries in [39] provides a starting point for our future investigation of provenance semirings for our calculus.

A significant amount of work has concerned provenance and security. One strand deals with securing access to provenance information (cf., e.g., [49, 9]). Hasan *et al*

[35] show how to provide strong integrity and confidentiality assurances for data provenance information. In [16] Cheney formalises what it means for a provenance tracking system to successfully disclose some information that users require while obfuscating other sensitive information. Acar et al. [1] develop a core calculus for provenance in programming languages and discuss some solutions to the disclosure and obfuscation problems. Chong [20] presents a formal system to control undesired indirect disclosure of provenance trace. Rosenthal *et al* [56] introduce '*attribute-based access control*' to specify policies of access control to provenance, whilst [15, 50] focus on confidentiality of provenance by controlling respectively 'user view' and queries. A different research line treats of integrity [34, 24, 26] and non-repudiability [27] of provenance traces. Golbeck [28] exploits provenance to implement trust-based filtering of web content, whilst Vaughan *et al* [61] use evidence-based audits for language-based security. We believe our calculus provides a powerful and flexible framework to investigate questions such as these, which is proposed as future work. The reader is remanded to [55] for further information about open problems and current provenance research.

8. Conclusion

The provenance format introduced in the work is clean and simple. It is however a significant extension of existing provenance formats for Linked Data. It provides a comprehensive account of where and who provenance, and records all agents who have published the data and where the data was written. In line with existing approaches to provenance for Linked Data, the provenance is recorded at the level of triples (rather than URIs).

Our examples use Linked Data published on the Web at the time of writing. They represent realistic scenarios, and are provided to explain how the demands of the application are addressed. The examples benefit from the formal syntax and the operational semantics of the calculus they are expressed in, and this enables an unambiguous discussion of the ideas explored.

The calculus presents some fresh ideas for new high level languages for Linked Data. Some high level constructs are suggested that combine explicit dereferencing of URIs with queries over the data obtained and the continuation process that uses the data. Furthermore, the queries are extended with patterns that exploit provenance, and demonstrate that the ideas in this paper can be usefully integrated with several existing languages. The framework for operational semantics employed is concise and extensible. Thus further features for tackling problems in Linked Data [36] can be combined with this work easily. Some basic how provenance is suggested by means of functions. By recording the functions applied to data in the provenance format, judgements can be made about the quality of data depending on whether reliable functions were applied. More detailed why provenance could be recorded by indicating a proof of why some data is transformed into some other data.

The present calculus leaves a significant number of open issues to investigate. Location policies are assumed to be fixed and available for centralised checks. Moreover agents cannot lie and provenance cannot be forgotten. A calculus which addresses these realistic challenges should allow location policies to change over the time and be stored in locations themselves, while agents may be untrusted. This scenario requires a reputation system for agents, and for locations to dynamically check agent requests against their policies, whilst taking the reputation of agents into account.

The calculus provides a credible and flexible framework for future developments, some of which are indicated in the previous section. Among the several avenues for future work, five priorities are anticipated. Firstly, a calculus of transformations over Linked Data should be specified to provide a detailed account of why provenance. Also, provenance traces are expected to be extended to directed acyclic graphs. This entails formulating a suitable syntax for graphs as well as a treatable logic for querying them. Security figures among the most interesting challenges for both Linked Data and provenance. A proposal is to focus on controlling the access to provenance information. This will involve equipping the calculus with mechanisms and primitives to specify suitable access control policies, as well as the analysis of how information may covertly flow from (public) provenance trees to (private) data following [20] and [16]. The calculus is proposed as a formal platform to develop trust-based assessment and filtering on the Web of Linked Data.

Acknowledgements. The present version of this paper strongly improved on the submitted one thanks to many useful referee suggestions.

References

[1] U. A. Acar, A. Ahmed, J. Cheney, and R. Perera. A core calculus for provenance. In *POST'12*, volume 7215 of *LNCS*, pages 410–429. Springer, 2012.

[2] A. V. Aho and J. D. Ullman. *Principles of Compiler Design*. Addison-Wesley, 1977.

[3] R. Albert, H. Jeong, and A.-L. Barabasi. Internet: Diameter of the World-Wide Web. *Nature*, 401:130–131, 1999.

[4] D. Artz and Y. Gil. A survey of trust in computer science and the semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):58 – 71, 2007.

[5] T. Berners-Lee. Linked Data – W3C design issues, 2006. http://www.w3.org/DesignIssues/LinkedData.html.

[6] T. Berners-Lee, W. Hall, J. A. Hendler, K. O'Hara, N. Shadbolt, and D. J. Weitzner. A framework for web science. *Foundations and Trends in Web Science*, 1(1):1–130, 2006.

[7] C. Bizer. The emerging Web of Linked Data. *IEEE Intelligent Systems*, 24:87–92, 2009.

[8] S. Bowers, T. McPhillips, B. Ludscher, S. Cohen, and S. Davidson. A model for user-oriented data provenance in pipelined scientific workflows. In *IPAW'06*, volume 4145 of *LNCS*, pages 133–147. Springer, 2006.

[9] U. Braun, A. Shinnar, and M. I. Seltzer. Securing provenance. In *HotSec*. USENIX Association, 2008. http://www.usenix.org/events/hotsec08/tech/full_papers/braun/braun.pdf.

[10] P. Buneman, J. Cheney, and S. Vansummeren. On the expressiveness of implicit provenance in query and update languages. *ACM Transactions on Database Systems*, 33:1–47, 2008.

[11] P. Buneman, S. Khanna, and W. C. Tan. Why and where: A characterization of data provenance. In *ICDT'01*, volume 1973 of *LNCS*, pages 316–330. Springer, 2001.

[12] P. Buneman and E. Kostylev. Annotation algebras for RDFS. In *SWPM'10*, volume 670 of *CEUR Workshop Proceedings*, pages 1–6. CEUR-WS.org, 2010.

[13] J. J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named graphs, provenance and trust. In *WWW'05*, pages 613–622. ACM Press, 2005.

[14] G. Castagna and A. Frisch. A gentle introduction to semantic subtyping. In *PPDP'05*, pages 198-208, ACM Press, (full version) and *ICALP'05*, volume of LNCS, pages 30-34, Springer, (summary), 2005. Joint ICALP-PPDP keynote talk.

[15] A. Chebotko, X. Fei, C. Lin, S. Lu, and F. Fotouhi. Storing and querying scientific workflow provenance metadata using an RDBMS. In *eScience'07*, pages 611–618. IEEE, 2007.

[16] J. Cheney. A formal framework for provenance security. In *CSF'11*, pages 281–293. IEEE, 2011.

[17] J. Cheney, U. A. Acar, and A. Ahmed. Provenance traces. Technical report, University of Edinburgh, 2008. arXiv:0812.0564v1.

[18] J. Cheney, A. Ahmed, and U. A. Acar. Provenance as dependency analysis. *Mathematical Structures in Computer Science*, 21(6):10301–1337, 2011.

[19] J. Cheney, L. Chiticariu, and W.-C. Tan. Provenance in databases: Why, how, and where. *Foundations and Trends in Databases*, 1(4):379–474, 2009.

[20] S. Chong. Towards semantics for provenance security. In *TAPP'09*. USENIX Association, 2009. `http://www.usenix.org/events/tapp09/tech/full_papers/chong/chong.pdf`.

[21] B. Clifford, I. Foster, J.-S. Voeckler, M. Wilde, and Y. Zhao. Tracking provenance in a virtual data grid. *Concurrency and Computation: Practice and Experience*, 20(5):565–575, 2008.

[22] Y. Cui, J. Widom, and J. L. Wiener. Tracing the lineage of view data in a warehousing environment. *ACM Transactions on Database Systems*, 25(2):179–227, 2000.

[23] L. Ding, J. Shinavier, Z. Shangguan, and D. McGuinness. SameAs networks and beyond: Analyzing deployment status and implications of owl:sameAs in Linked Data. In *ISWC'10*, volume 6496 of *LNCS*, pages 145–160. Springer, 2010.

[24] M. Factor, E. Henis, D. Naor, S. Rabinovici-Cohen, P. Reshef, S. Ronen, G. Michetti, , and M. Guercio. Authenticity and provenance in long term digital preservation: modeling and implementation in preservation aware storage. In *TAPP'09*. USENIX Association, 2009. `http://www.usenix.org/events/tapp09/tech/full_papers/factor/factor.pdf`.

[25] G. Flouris, I. Fundulaki, P. Pediaditis, Y. Theoharis, and V. Christophides. Coloring RDF triples to capture provenance. In *ISWC'09*, volume 5823 of *LNCS*, pages 196–212. Springer, 2009.

[26] L. M. Gadelha Jr and M. Mattoso. Kairos: An architecture for securing authorship and temporal information of provenance data in grid-enabled workflow management systems. In *eScience'08*, pages 597–602. IEEE, 2008.

[27] A. Gehani, M. Kim, and J. Zhang. Steps toward managing lineage metadata in grid clusters. In *TAPP'09*. USENIX Association, 2009. `http://www.usenix.org/events/tapp09/tech/full_papers/gehani/gehani.pdf`.

[28] J. Golbeck. Combining provenance with trust in social networks for semantic web content filtering. In *IPAW'06*, volume 4145 of *LNCS*, pages 101–108. Springer, 2006.

[29] T. J. Green, G. Karvounarakis, Z. G. Ives, and V. Tannen. Update exchange with mappings and provenance. In *VLDB'07*, pages 675–686. ACM Press, 2007.

[30] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *PODS'07*, pages 31–40. ACM Press, 2007.

[31] P. Groth, S. Jiang, S. Miles, S. Munroe, V. Tan, S. Tsasakou, and L. Moreau. An architecture for provenance systems. Technical report, University of Southampton, 2006. `http://eprints.ecs.soton.ac.uk/13216/`.

[32] H. Halpin and J. Cheney. Dynamic provenance for SPARQL updates using named graphs. In *TAPP'11*, pages 4:1–4:6. USENIX Association, 2011. `http://static.usenix.org/event/tapp11/tech/final_files/Halpin.pdf`.

[33] A. Harth, A. Polleres, and S. Decker. Towards a social provenance model for the web. In *PrOPr'07*, 2007. `http://axel.deri.ie/publications/harth-etal-2007.pdf`.

[34] R. Hasan, R. Sion, and M. Winslett. The case of the fake Picasso: Preventing history forgery with secure provenance. In *FAST'09*, pages 1–14. USENIX Association, 2009. `http://www.usenix.org/events/fast09/tech/full_papers/hasan/hasan.pdf`.

[35] R. Hasan, R. Sion, and M. Winslett. Preventing history forgery with secure provenance. *ACM Transactions on Storage*, 5(4):12:1–12:43, Dec. 2009.

[36] R. Horne. *Programming Languages and Principles for Read–Write Linked Data*. PhD thesis, School of Electronics and Computer Science, University of Southampton, 2011.

[37] R. Horne and V. Sassone. A typed model for Linked Data. Technical report, University of Southampton, 2011. `http://eprints.ecs.soton.ac.uk/21996/`.

[38] R. Horne and V. Sassone. A verified algebra for Linked Data. In *FOCLSA'11*, volume 58 of *EPTCS*, pages 20–33, 2011.

[39] R. Horne and V. Sassone. A verified algebra for read-write Linked Data. Technical report, University of Southampton, 2012. `http://eprints.soton.ac.uk/273248/`.

[40] R. Horne, V. Sassone, and N. Gibbins. Operational semantics for SPARQL Update. In *STC'11*, volume 7185 of *LNCS*, pages 242–257. Springer, 2011.

[41] I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible SROIQ. In *KR'06*, pages 57–67. AAAI Press, 2006.

[42] A. Jeffrey and P. Patel-Schneider. Integrity constraints for Linked Data. In *DL'11*, volume 745 of *CEUR Workshop Proceedings*, pages 521–531. CEUR-WS.org, 2011.

[43] G. Klyne and P. Groth. PROV-AQ: Provenance access and query. Technical report, W3C Working Draft, 2012. `http://www.w3.org/TR/prov-aq`.

[44] N. Kwasnikowska, L. Moreau, and J. Van den Bussche. A formal account of the open provenance model. Technical report, University of Southampton, 2010. `http://eprints.ecs.soton.ac.uk/21819/`.

[45] S. Miles, P. T. Groth, S. Munroe, and L. Moreau. PrIMe: A methodology for developing provenance-aware applications. *ACM Transactions on Software Engineering and Methodology*, 20(3):8, 2011.

[46] L. Moreau. The foundations for provenance on the web. *Foundations and Trends in Web Science*, 2(2-3):99–241, 2010.

[47] L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. Groth, N. Kwasnikowska, S. Miles, P. Missier, J. Myers, B. Plale, Y. Simmhan, E. Stephan, and J. Van den Bussche. The open provenance model core specification (v1.1). *Future Generation Computer Systems*, 27(6):743–756, 2010.

[48] L. Moreau and P. Missier. The PROV data model and abstract syntax notation. Technical report, W3C Working Draft, 2012. `http://www.w3.org/TR/prov-dm`.

[49] K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. I. Seltzer. Provenance-aware storage systems. In *USENIX'06*, pages 43–56. USENIX Association, 2006. http://static.usenix.org/event/usenix06/tech/muniswamy-reddy.html.

[50] M. Nagappan and M. Vouk. A model for sharing of confidential provenance information in a query based system. In *IPAW'08*, volume 5272 of *LNCS*, pages 62–69. Springer, 2008.

[51] C. Ogbuji. *SPARQL 1.1 Graph Store HTTP Protocol*. W3C Working Draft, 2012. http://www.w3.org/TR/sparql11-http-rdf-update/.

[52] J. Pérez, M. Arenas, and C. Gutierrez. Semantics and complexity of SPARQL. *ACM Transactions on Database Systems*, 34(3):16:1–16:45, 2009.

[53] A. Polleres, C. Feier, and A. Harth. Rules with contextually scoped negation. In *SWC'06*, volume 4011 of *LNCS*, pages 332–347. Springer, 2006.

[54] A. Powell, M. Nilsson, A. Naeve, P. Johnston, and T. Baker. DCMI abstract model. Technical report, Dublin Core Metadata Initiative, 2007. http://www.dublincore.org/documents/abstract-model.

[55] Provenance Working Group. http://www.w3.org/2011/prov/wiki/Main_Page.

[56] A. Rosenthal, L. Seligman, A. Chapman, and B. Blaustein. Scalable access controls for lineage. In *TAPP'09*. USENIX Association, 2009. http://www.usenix.org/events/tapp09/tech/full_papers/rosenthal/rosenthal.pdf.

[57] N. Shadbolt, T. Berners-Lee, and W. Hall. The semantic web revisited. *IEEE Intelligent Systems*, 21(3):96–101, 2006.

[58] Y. Simmhan, B. Plale, and D. Gannon. A survey of data provenance in e-science. *SIGMOD Record*, 34:31–36, 2005.

[59] I. Souilah, A. Francalanza, and V. Sassone. A formal model of provenance in distributed systems. In *TAPP'09*. USENIX Association, 2009. http://www.usenix.org/events/tapp09/tech/full_papers/souilah/souilah.pdf.

[60] O. Udrea, D. R. Recupero, and V. S. Subrahmanian. Annotated RDF. *ACM Transaction on Computational Logic*, 11(2):10:1–10:41, 2010.

[61] J. A. Vaughan, L. Jia, K. Mazurak, and S. Zdancewic. Evidence-based audit. In *CSF'08*, pages 177–191. IEEE, 2008.

[62] Y. R. Wang and S. E. Madnick. A polygen model for heterogeneous database systems: The source tagging perspective. In *VLDB'90*, pages 519–538. Morgan Kaufmann, 1990.

[63] J. Zhao, C. Goble, R. Stevens, and S. Bechhofer. Semantically linking and browsing provenance logs for E-science. In *ICSNW'04*, volume 3226 of *LNCS*, pages 158–176. Springer, 2004.

[64] J. Zhao, C. Goble, R. Stevens, and D. Turi. Mining taverna's semantic web of provenance. *Concurrency and Computation: Practice and Experience*, 20(5):463–472, 2008.

[65] A. Zimmermann, N. Lopes, A. Polleres, and U. Straccia. A general framework for representing, reasoning and querying with annotated semantic web data. *Journal of Web Semantics*, 11:72–95, 2012.