

Linked data privacy

SVETLANA JAKŠIĆ, JOVANKA PANTOVIĆ
and SILVIA GHILEZAN†

University of Novi Sad, Trg Dositeja Obradovica 6, Novi Sad, Serbia

Received 14 October 2013; revised 11 November 2014

Web of Linked Data introduces common format and principles for publishing and linking data on the Web. Such a network of linked data is publicly available and easily consumable. This paper introduces a calculus for modelling networks of linked data with encoded privacy preferences.

In that calculus, a network is a parallel composition of users, where each user is named and consists of data, representing the user's profile, and a process. Data is a parallel composition of triples with names (resources) as components. Associated with each name and each triple of names are their privacy protection policies, that are represented by queries. A data triple is accessible to a user if the user's data satisfies the query assigned to that triple.

The main contribution of this model lies in the type system which together with the introduced query order ensures that static type-checking prevents privacy violations. We say that a network is well behaved if

- access to a triple is more restrictive than access to its components and less restrictive than access to the user name it is enclosed with,
- each user can completely access their own profile,
- each user can update or partly delete profiles that they own (can access the whole profiles), and
- each user can update the privacy preference policy of data of another profile that they own or write data to another profile only if the newly obtained profile stays fully accessible to their owner.

We prove that any well-typed network is well behaved.

1. Introduction

The more data connects with other sources of information, the more its value increases. Having that in mind, an initiative to establish a generic format for connecting the Semantic Web was born a decade ago, and has grown into a collection of recommendations for publishing data on the Web (Bizer 2009; Klyne and Carroll 2004). The Web of Linked Data is expected to expand into a huge graph of linked data, based on the use of URIs for names of anything, use of HTTP URIs for reading the names, and making connections between data on the Web in Resource Description Format (RDF). Publishing data online in an open standard, such as RDF, and interlinking data sources aims to transform Web

† This work was partly supported by the Serbian Ministry of Education, Science and Technological Development (projects ON174026 and III44006) and COST Action IC1201.

of Documents to more (re)usable, machine readable, Web of Data. Assuming that the Web of Linked Data is a reality, the data can be queried by W3 recommended SPARQL Query (Klyne and Carroll 2004; Prud'hommeaux and Seaborne 2008) and consumed by a higher-level programming language.

Another great merit of the Web of Linked Data is its exposure to public consumption. Even though public availability brings a great advantage to users of such data, not all data are produced for public usage. This gives rise to the question of privacy of linked data since the lack of privacy protection mechanisms often discourages people from publishing data on the Web of Linked Data. Addressing this issue requires a clear explanation for the intuition of the notion of privacy. Privacy may not include private status of some data only, but also the significance or no significance of data for some group and ability of readers to understand the data properly (Stanković *et al.* 2009). In Westin (1967), the privacy is defined as 'the ability to control who has access to information and to whom that information is communicated'. In this sense, we deem that privacy of data is protected in case:

- An owner (creator) of data can always access their own data.
- An owner of data can control access to their data, i.e. they can create conditions for other consumers to access parts (or all) of their data. Only consumers that fulfil the settled conditions can access the data constituents. If a data set (corresponding to an owner) is completely accessible to different users, they are all considered as the data owners and the data can be updated by any of them.

In this paper, we introduce a formal model of linked data that can statically detect run-time errors due to privacy violations.

We introduce a new calculus for modelling the web of linked data. Resources (URIs) are denoted by names from a specified set. Data are parallel composition of triples of names (describing links between resources). Processes are π -calculus processes (Milner 1999; Milner *et al.* 1992), without input and output capabilities, and with introduced capabilities that describe the interaction between processes and data. Networks are parallel compositions of user names enclosing data and processes.

In order to enable each owner of data to control privacy of the data, we assign a query to each user name and also a query to each data triple. This approach is similar to the one proposed in Sacco and Passant (2011b). We say that a user can access a triple if the user's data satisfies the query assigned to the triple. However, an owner's privacy will be violated:

- if a user updated a query of a triple with a query that was not satisfied by the owner's data or
- if a user wrote a data triple to an owner's data and the triple query was not satisfied by the owner's data.

In this scenario, we say that a network is well behaved if:

- access to a triple is more restrictive than access to its components and less restrictive than access to the user name it is enclosed with,
- each user can access each triple of its own data,

- each user can update or partly delete data that they own (can access all its triples), and
- each user can update a query assigned to a triple or write data to a user only if the newly obtained data stays accessible to its owner.

We introduce a simple yet sufficiently powerful type assignment system which, together with the introduced query order, is able to statically check if a network is well behaved. Since the queries are partially ordered the type system enforces that the query of a triple is

- bounded above by the queries of triple components and
- bounded below by the query of the user name it is enclosed with.

The type system ensures that a process can modify a triple’s query only if the query, assigned to the name it is enclosed with, admits this. Finally, we prove the main result, that a well-typed network is well behaved.

We clarify the developed typed calculus informally by the following example.

1.1. Example

In order to achieve a clear explanation of the calculus presented in this paper, the example is written in the corresponding language, instead of original RDF and SPARQL format.

Suppose that there are two users, Alice and Bob, running in parallel

$$N_{\text{Alice}} \mid N_{\text{Bob}}$$

where each user has their own profile (corresponding to RDF data) D_{user} and a process P_{user} running on behalf of that user. This will be written as

$$N_{\text{user}} = \text{user}[D_{\text{user}} \parallel P_{\text{user}}], \quad \text{user} \in \{\text{Alice}, \text{Bob}\}.$$

Data are represented as parallel composition of triples, where each component is a name (resource).

Since the main goal of this calculus is to protect privacy of data, we associate each resource (user name or triple component) and each triple with a privacy protection policy. A novelty we propose is to take the same apparatus to query data and to check if a profile (data) satisfies a privacy protection policy. Therefore, we introduce a function \mathcal{T} to assign a query (privacy protection policy) to each name and also assign a query to each data triple. For example, we consider the following data:

user	Profile (Data)
Alice	(Alice, affiliation, www.uns.ac.rs) ^{U_{Alice}} post ₁ ^{U_1} post ₂ ^{U_2} post ₃ ^{U_3}
Bob	(Bob, friend, Cindy) ^{U_{Bob}} (Bob, research, types) ^{U_{Bob}}

Alice is a professor and she writes three posts about her scientific work, private life and teaching. It is assumed that the user Alice has the privacy protection policy $\mathcal{T}(\text{Alice}) = U_{\text{Alice}}$ with $U_{\text{Alice}} = (\text{Alice}, \text{affiliation}, \text{www.uns.ac.rs})$. We also assume that each triple component policy is less restrictive than the triple policy (a user can create a triple

only from names that are accessible to that user). For example, let

$$T(\text{affiliation}) = T(\text{www.uns.ac.rs}) = \exists x.\exists y.\exists z.(x, y, z).$$

The query $\exists x.\exists y.\exists z.(x, y, z)$ is satisfied by a profile, if there is any triple in the profile data, and it will thus be the privacy policy of public names and public data. The users with profiles that contain $(\text{Alice}, \text{affiliation}, \text{www.uns.ac.rs})^U$, like Alice, can access her profile (for an arbitrary U). This is the case because such a profile satisfies the given query U_{Alice} . We denote by \models the relation that states that data satisfy query. We will write

$$(\text{Alice}, \text{affiliation}, \text{www.uns.ac.rs})^U \models (\text{Alice}, \text{affiliation}, \text{www.uns.ac.rs})$$

and the same will hold for all data that contain the triple,

$$(\text{Alice}, \text{affiliation}, \text{www.uns.ac.rs})^U | D' \models (\text{Alice}, \text{affiliation}, \text{www.uns.ac.rs}).$$

We say that the names *affiliation* and *www.uns.ac.rs* are public since it holds

$$D \models \exists x.\exists y.\exists z.(x, y, z)$$

for any D .

It is natural to require that each user has access to its own profile (i.e. to each triple in the profile), which is for Alice provided by conditions

$$D_{\text{Alice}} \models U_{\text{Alice}} \wedge D_{\text{Alice}} \models U_1 \wedge D_{\text{Alice}} \models U_2 \wedge D_{\text{Alice}} \models U_3.$$

This property will be written as

$$\text{readable}(D_{\text{Alice}}, D_{\text{Alice}}) = D_{\text{Alice}}.$$

Let the privacy protection policies of its single posts (triples) be the following:

Triple	Privacy Protection Policy
$post_1$	$U_1 = \exists x.(x, \text{research}, \text{types}) \oplus \exists x.(x, \text{affiliation}, \text{www.uns.ac.rs})$
$post_2$	$U_2 = \exists x.(x, \text{guest}, \text{www.uns.ac.rs}) \oplus \exists x.(x, \text{affiliation}, \text{www.uns.ac.rs})$
$post_3$	$U_3 = \exists x.(\text{Alice}, \text{professor}, x) \oplus \exists x.(x, \text{friend}, \text{Cindy})$ $\oplus \exists x.(x, \text{affiliation}, \text{www.uns.ac.rs})$

The first post written by Alice is about her research and she wants to share it with everyone with the same affiliation and with those whose research interest is types:

$$post_1 = (\text{paper}, \text{published}, \text{journal})^{U_1}.$$

The present type system protects privacy of a single resource. We want to guarantee that a name that is a component of a triple has a less restrictive policy than the triple. This means that the permission to access a triple can only be given to processes that have permissions to access all its components. For this purpose, we introduce an order \preceq between privacy protection policies (i.e. between queries). For example, it holds that

$$(\text{Alice}, \text{affiliation}, \text{www.uns.ac.rs}) \preceq \exists x.\exists y.\exists z.(x, y, z).$$

We will later prove that $U \preceq V$ implies $D \models U \Rightarrow D \models V$.

Bob has permission to read $post_1$ and $post_3$ because

$$D_{\text{Bob}} \models U_1 \text{ and } D_{\text{Bob}} \models U_3,$$

while he cannot read $post_2$. The part of D_{Alice} that Bob can access is

$$\text{readable}(D_{\text{Bob}}, D_{\text{Alice}}) = post_1^{U_1} | post_3^{U_3}.$$

Let the process running on behalf of Bob be

$$P_{\text{Bob}} = \text{READ}_{\text{Alice}}(\exists x.(x, \text{published}, \text{journal}), \chi).Q.$$

This process looks for all the triples from D_{Alice} that are readable to him and that satisfy the given query. Since only $post_1$ fulfills both conditions, the network reduces as follows:

$$N_{\text{Alice}} | N_{\text{Bob}} \rightarrow N_{\text{Alice}} | \text{Bob}[D_{\text{Bob}} || Q\{post_1/\chi\}].$$

Alice may decide to change the privacy protection policy of $post_2$ in order to let Bob see her post about the party, by running the following process:

$$P_{\text{Alice}} = \text{UPDATE}_{\text{Alice}}(post_2, U_2 \oplus (\text{Bob}, \text{friend}, \text{Cindy})).R.$$

It will be well typed in our calculus since

$$U_{\text{Alice}} \preceq U_2 \oplus (\text{Bob}, \text{friend}, \text{Cindy})$$

and hence

$$D_{\text{Alice}} \models U_{\text{Alice}} \Rightarrow D_{\text{Alice}} \models U_2 \oplus (\text{Bob}, \text{friend}, \text{Cindy}).$$

It means that Alice still has access to $post_2$ and she can still control who has access to her data.

1.2. Structure of the paper

The remainder of the paper is organized as follows. In Section 2, we introduce formally the calculus, including the syntax and the operational semantics, that describes the behaviour of a network of linked data which is used by a higher-level language. The operational semantics uses some auxiliary functions and relations that specify how data and processes interact within the network. In order to obtain a suitable basis for studying privacy preference issues, we introduce a type system, which is the content of Section 3. We formalize, in the language of this type system, when we consider that a network is well behaved. The main results are collected in Section 4, with all the technicalities to prove the subject reduction and the main properties. Finally, Section 5 discusses related papers and Section 6 gives a brief summary with certain directions for future work.

2. Language

In this section, we formally introduce the calculus.

Table 1. Syntax of query pattern and data.

$U ::=$	Query pattern	$D ::=$	Data
(u, u, u)	Triple pattern	\emptyset	Empty data
$ U \oplus U$	Choice	$ (a, a, a)^U$	Triple with query pattern
$ \exists x.U$	Exists	$ D D$	Parallel

2.1. Syntax

2.1.1. Names. We assume that we are given an infinite set Names of URIs(names, resources) ranged over by a, b, \dots , and an infinite set Variables of name variables ranged over by x, y, \dots . We let u, v range over the set Names \cup Variables.

2.1.2. Query patterns. The syntax of query pattern is given in Table 1. As a query pattern, we take a triple of names or name variables, choice of query patterns or an existing query pattern. We have chosen to leave out the join query since it would make the syntax and the operational semantics more complicated without contributing to privacy properties. The operator \exists is binding and we say that a query pattern is well formed if it contains no occurrences of free variables. Hereinafter, we observe only such query patterns. Query patterns are ranged over by U, V, W, \dots

2.1.3. Data. The syntax of data is given in Table 1. Data is an empty data, a triple of names with an associated query pattern or a parallel composition of data. A query pattern is assigned to each triple of names, representing its privacy protection policy. Data variables are ranged over by χ, ψ, \dots . We let δ, ϵ range over data and data variables.

2.1.4. Processes. The syntax of processes is given in Table 2. The first four process expressions, $0, P + P, P|P$ and $\text{rec}X.P$ are regular π -calculus processes. We additionally have:

- $\text{READ}_u(U, \chi).P$ that expresses a process that reads from the user named u the data D found by the query U and then behaves like $P\{^D/\chi\}$;
- $\text{WRITE}_u(\delta).P$ that expresses a process that writes the data δ to the user named u and then behaves like P ;
- $\text{DELETE}_u(U).P$ that expresses a process that deletes from the user named u the data found by the query U and then behaves like P ;
- $\text{SELECT}_u(\exists x.U).P$ that expresses a process that collects substitutions and then behaves like the parallel composition of processes obtained by applying these substitutions to P . Each substitution has the property that when applied to the given query pattern $\exists x.U$, it finds a data triple of user named u that satisfies it; and
- $\text{UPDATE}_u(U_1, U_2).P$ that expresses a process that finds the data of user named u identified by the query pattern U_1 , change by the query pattern U_2 all the query patterns of triples in the found data, and then behaves like P .

Process variables are ranged over by X, Y, \dots

Table 2. Syntax of processes and networks.

$P ::=$	Process	$N ::=$	Network
0	Termination	$a[D \parallel P]$	User
$ P + P$	Choice	$ N N$	Parallel
$ P P$	Parallel		
$ \text{rec}X.P$	Recursion		
$ X$	Process variable		
$ \text{READ}_u(U, \chi).P$	Read		
$ \text{WRITE}_u(\delta).P$	Write		
$ \text{DELETE}_u(U).P$	Delete		
$ \text{SELECT}_u(\exists x.U).P$	Select		
$ \text{UPDATE}_u(U, U).P$	Update		

2.1.5. *Networks.* The syntax of *networks* is given in Table 2. A network is a user name that encloses data and process, or a parallel composition of networks. We say that a *network is well formed* if all its users have different names.

2.2. Operational semantics

2.2.1. *Satisfaction of query patterns and queries.* Before defining the operational semantics we need to formalize two notions. The first one is a deductive system which presents a tool for checking whether some data triple satisfy a query pattern (in the absence of a join query, data that can satisfy a query pattern can only be a single data triple).

Definition 2.1 (satisfaction of a query pattern). We say that a triple T^U satisfies query pattern V , written $T^U \models V$, if it can be deduced by the following rules:

$$\begin{array}{c}
 \text{[Q-TRIPLE]} \\
 T^U \models T
 \end{array}
 \qquad
 \begin{array}{c}
 \text{[Q-CHOICE L]} \\
 \frac{D \models U}{D \models U \oplus V}
 \end{array}
 \qquad
 \begin{array}{c}
 \text{[Q-CHOICE R]} \\
 \frac{D \models V}{D \models U \oplus V}
 \end{array}
 \qquad
 \begin{array}{c}
 \text{[Q-EXISTS]} \\
 \frac{D \models U\{^a/x\}}{D \models \exists x.U}
 \end{array}$$

The second one is a method for checking whether some profile (data) satisfies an ask query.

Definition 2.2 (satisfaction of ask query). We say that data D satisfies an ask query of a query pattern U , written $D \models U$, if it can be deduced by the following rules

$$\begin{array}{c}
 \text{[Q-ASK]} \\
 \frac{D' \models U}{D'|D'' \models U}
 \end{array}
 \qquad
 \begin{array}{c}
 \text{[Q-SCONG]} \\
 \frac{D' \models U \quad D' \equiv_D D''}{D'' \models U}
 \end{array}$$

We sometimes write that D satisfies a query U , for short. Informally, a data satisfies a query if there is a triple in the data that satisfies the corresponding query pattern.

2.2.2. *Auxiliary functions.* In Table 3, we introduce auxiliary functions for data manipulation:

- $\text{readable}(D_a, D_b)$ takes two data (profiles) as arguments and returns the parallel composition of triples of the second profile that a process running on behalf of the

Table 3. Definitions of data manipulation functions.

$readable(D, \emptyset)$	$= \emptyset$
$readable(D, (a, b, c)^U)$	$= \begin{cases} (a, b, c)^U & \text{if } D \models U, \\ \emptyset & \text{otherwise} \end{cases}$
$readable(D, D_1 D_2)$	$= readable(D, D_1) readable(D, D_2)$
$read(U, \emptyset)$	$= \emptyset$
$read(U, (a, b, c)^V)$	$= \begin{cases} (a, b, c)^V & \text{if } (a, b, c)^V \models U, \\ \emptyset & \text{otherwise} \end{cases}$
$read(U, D_1 D_2)$	$= read(U, D_1) read(U, D_2)$
$delete(U, \emptyset)$	$= \emptyset$
$delete(U, (a, b, c)^V)$	$= \begin{cases} \emptyset & \text{if } (a, b, c)^V \models U, \\ (a, b, c)^V & \text{otherwise} \end{cases}$
$delete(U, D_1 D_2)$	$= delete(U, D_1) delete(U, D_2)$
$select(\exists x.U, \emptyset)$	$= \emptyset$
$select(\exists x.U, (a, b, c)^V)$	$= \begin{cases} \{a/x\} & \text{if } (a, b, c)^V \models U\{a/x\}, \\ \{b/x\} & \text{if } (a, b, c)^V \models U\{b/x\}, \\ \{c/x\} & \text{if } (a, b, c)^V \models U\{c/x\}, \\ \emptyset & \text{otherwise} \end{cases}$
$select(\exists x.U, D_1 D_2)$	$= select(\exists x.U, D_1) \cup select(\exists x.U, D_2)$
$update(U, \emptyset, V)$	$= \emptyset$
$update(U, (a, b, c)^{V'}, V)$	$= \begin{cases} (a, b, c)^V & \text{if } (a, b, c)^{V'} \models U, \\ (a, b, c)^{V'} & \text{otherwise} \end{cases}$
$update(U, D_1 D_2, V)$	$= update(U, D_1, V) update(U, D_2, V)$

first profile can access. We will say that the first profile’s user can fully access (or owns) the second profile if $readable(D_a, D_b) = D_b$.

- $read(U, D)$ takes a query and data as arguments and returns the parallel composition of the data triples that satisfy the query;
- $delete(U, D)$ takes a query and data as arguments and, contrary to $read$, returns the parallel composition of all the triples that do not satisfy the query;
- $select(\exists x.U, D)$ takes an existing query and data as arguments and returns a set of substitutions. The substitutions are those that applied to the query induce that the data satisfy the query. Sometimes we denote a substitution with s and a set of substitutions with S ;
- $update(U, D, V)$ takes a query, a profile and another query as arguments and returns the profile with privacy protection policies updated on triples that satisfy the first query. The new privacy protection policies are given by the second query.

2.2.3. *Auxiliary relation.* Besides auxiliary functions, we introduce also an auxiliary relation \rightsquigarrow_a , in order to describe interaction between processes and data that occurs on behalf of the user $a[D_a \parallel U_a]$. It is given in Table 4.

Table 4. Interaction rules.

$\frac{[I\text{-CHOICE}]}{(P, D) \rightsquigarrow_a (P', D')}{(P + Q, D) \rightsquigarrow_a (P', D')}$	$\frac{[I\text{-PARALLEL}]}{(P, D) \rightsquigarrow_a (P', D')}{(P Q, D) \rightsquigarrow_a (P' Q, D')}$	$\frac{[I\text{-RECURSION}]}{(P\{\text{rec}^X.P/X\}, D) \rightsquigarrow_a (P', D')}{(\text{rec}X.P, D) \rightsquigarrow_a (P', D')}$
$\frac{[I\text{-READ}]}{D' = \text{read}(U, \text{readable}(D_a, D))}{(\text{READ}_b(U, \chi).P, D) \rightsquigarrow_a (P\{D'/\chi\}, D)}$	$\frac{[I\text{-WRITE}]}{(\text{WRITE}_b(D').P, D) \rightsquigarrow_a (P, D D')}$	$\frac{[I\text{-DELETE}]}{D' = \text{delete}(U, D)}{(\text{DELETE}_b(U).P, D) \rightsquigarrow_a (P, D')}$
$\frac{[I\text{-SELECT}]}{S = \text{select}(\exists x.U, \text{readable}(D_a, D))}{(\text{SELECT}_b(\exists x.U).P, D) \rightsquigarrow_a (\prod_{s \in S} P_s, D)}$	$\frac{[I\text{-UPDATE}]}{D' = \text{update}(U, D, V)}{(\text{UPDATE}_b(U, V).P, D) \rightsquigarrow_a (P, D')}$	

Table 5. Structural congruence.

<p>(Data) $(D_1 D_2) D_3 \equiv_D D_1 (D_2 D_3)$ $D_1 D_2 \equiv_D D_2 D_1$ $D \emptyset \equiv_D D$</p> <p>(Processes) $(P_1 P_2) P_3 \equiv_P P_1 (P_2 P_3)$ $P_1 P_2 \equiv_P P_2 P_1$ $P 0 \equiv_P P$</p> <p style="padding-left: 20px;">$(P_1 + P_2) + P_3 \equiv_P P_1 + (P_2 + P_3)$ $P_1 + P_2 \equiv_P P_2 + P_1$</p> <p style="padding-left: 20px;">$D_1 \equiv_D D_2 \Rightarrow \text{WRITE}_a(D_1).P \equiv_P \text{WRITE}_a(D_2).P$</p> <p>(Networks) $(N_1 N_2) N_3 \equiv N_1 (N_2 N_3)$ $N_1 N_2 \equiv N_2 N_1$</p> <p style="padding-left: 20px;">$D_1 \equiv_D D_2 \wedge P_1 \equiv_P P_2 \Rightarrow a[D_1 \parallel P_1] \equiv a[D_2 \parallel P_2]$</p>

- In rules [I-Choice], [I-Parallel], [I-Recursion], [I-Read] and [I-Select] the data remain unchanged.
- In rules [I-Choice], [Parallel] and [I-Recursion], components that correspond to processes continue as in reductions in the π -calculus, while in [I-Delete], [I-Write] and [I-Update] processes follow up by the given (unchanged) continuations.
- In [I-Read], the process $\text{READ}_b(U, \chi).P$ interacts with the data D such that it finds the part D' of the data that is both readable to the profile D_a and satisfies the query U , and substitutes D' for χ in the continuation P .
- In [I-Select], interaction between $\text{SELECT}_b(\exists x.U).P$ and D results in the parallel composition of those processes obtained by applying to the continuation P the substitutions that are the result of $\text{select}(\exists x.U, D')$. The data argument D' of this function is the part of D that is readable to D_a .
- In [I-Write], the process $\text{WRITE}_b(D').P$ and the data D give, after interaction, the data D' composed in parallel with D .
- In [I-Delete] and [I-Update], reactions produce the data that is the result of the data manipulation functions delete and update, respectively.

2.2.4. *Reduction relation.* We assume \equiv to be the structural congruence, which is the least equivalence relation on networks that is closed with respect to α -conversion and satisfies the axioms given in Table 5.

In Table 6, we define the *reduction relation* on networks. There are two axioms, [R-User] and [R-Self]. The former one allows interaction between different users and it is determined by the interaction between the process of the first user and the data of the second user. The latter one describes self-interaction between user's process and its own data. The rules [R-Struct] and [R-Parallel] formalize the property that the reduction

Table 6. *Reduction relation.*

$\frac{[R\text{-USER}]}{a[D_a \parallel P] \mid b[D_b \parallel Q] \rightarrow a[D_a \parallel P'] \mid b[D \parallel Q]} \quad \begin{array}{l} (P, D_b) \rightsquigarrow_a (P', D) \quad a \neq b \\ N_1 \equiv N'_1 \quad N_1 \rightarrow N_2 \quad N_2 \equiv N'_2 \\ \hline N'_1 \rightarrow N'_2 \end{array}$	$\frac{[R\text{-SELF}]}{a[D_a \parallel P] \rightarrow a[D \parallel P']} \quad \begin{array}{l} (P, D_a) \rightsquigarrow_a (P', D) \\ N_1 \rightarrow N_2 \\ \hline N_1 \mid N \rightarrow N_2 \mid N \end{array}$
---	--

relation is defined up to structural congruence and closed with respect to the usual static contexts. We use \rightarrow^* to denote the reflexive and transitive closure of \rightarrow .

2.2.5. *Well-behaved networks.* Suppose we are given a function \mathcal{T} that assigns query patterns (privacy protection policies) to names. More precisely, each name will have a corresponding privacy protection policy identified by the query pattern. The following definition formalizes what we consider as well-behaved network, i.e. when we consider that privacy of data is completely protected.

Definition 2.3. Let $N \rightarrow^* a[D_a \parallel P] \mid N'$ and $\mathcal{T}(a) = U_a$. We say that N is *well behaved* if the following three assertions hold:

- i. if $D_a \equiv (a_1, a_2, a_3)^U \mid D'_a$ and $\mathcal{T}(a_i) = U_i, i \in \{1, 2, 3\}$, then
 -
 - $D \models U$ implies $D \models U_i$ for every $i \in \{1, 2, 3\}$,
- ii. $D \models U_a$ implies $D \models U$, and $\text{readable}(D_a, D_a) = D_a$, and
- iii. if $N' \equiv b[D_b \parallel Q] \mid N''$ with $\mathcal{T}(b) = U_b$ then
 - if $P \equiv \text{DELETE}_b(U).R$ or $P \equiv \text{UPDATE}_b(U, W).R$ then $\text{readable}(D_a, D_b) = D_b$, and
 - if $P \equiv \text{WRITE}_b(D').R$ or $P \equiv \text{UPDATE}_b(U, W).R$ and $(P, D_b) \rightsquigarrow_a (P, D'_b)$ then $\text{readable}(D'_b, D'_b) = D'_b$.

Less formally, a network is well behaved if:

- access to a triple is less restrictive than access to the user name it is enclosed with,
- access to a triple is more restrictive than access to its components,
- each user can completely read their own profile,
- each user can update or partly delete profiles that they own (can read the whole profile), and
- each user can update the privacy preference policy of data of another profile that they own or write data to another profile only if the newly obtained profile stays accessible (fully readable) to their owner.

Example 2.1. Even if we extended Bob’s profile such that Bob became an owner of the profile of Alice, we would treat the network with the processes

$$\begin{aligned} P_{\text{Bob}} &\equiv \text{UPDATE}_{\text{Alice}}(\text{post}_2, (\text{Bob}, \text{research}, \text{types})) \text{ or} \\ P_{\text{Bob}} &\equiv \text{WRITE}_{\text{Alice}}(\text{post}_4^{(\text{Bob}, \text{research}, \text{types})}) \end{aligned}$$

as ill behaved, since after reduction Alice’s profile would become

$$D_{\text{Alice}} \equiv \dots | \text{post}_i^{(\text{Bob}, \text{research}, \text{types})} | \dots$$

and Alice could not access her own triple post_i (for $i = 2, 4$).

3. Type system

Aiming to prevent privacy violations, we introduce a type system and prove that well-typed networks are well behaved.

3.1. Query comparison

In order to compare privacy protection policies, we find it inevitable to introduce a relation on queries.

Definition 3.1 (query comparison relation). We define a partial order \preceq on queries by the following rules:

$$U \preceq U \oplus V \quad V \preceq U \oplus V \quad U\{^a/x\} \preceq \exists x.U \quad \frac{U\{^a/x\} \preceq V\{^a/x\}}{\exists x.U \preceq \exists x.V}$$

If $U \preceq V$ we say that U is *more restrictive* than V (or, equivalently, that V is *less restrictive* than U).

We use the relation \preceq in order to compare privacy protection policies. Before proving Theorem 3.1 we will prove two auxiliary lemmas.

Lemma 3.1. If $\exists x.U \preceq \exists x.V$ then $U\{^a/x\} \preceq V\{^a/x\}$ for every name a .

Lemma 3.2. If $U \preceq V$ then $U\{^a/x\} \preceq V\{^a/x\}$ and $\exists x.U \preceq \exists x.V$.

Proof. We will prove that $U\{^a/x\} \preceq V\{^a/x\}$ and the rest will hold by Definition 3.1.

The proof is by induction on the derivation of $U \preceq V$.

If $U \preceq U \oplus V$ then $U\{^a/x\} \preceq U\{^a/x\} \oplus V\{^a/x\}$, by definition of \preceq .

If $V \preceq U \oplus V$ then $V\{^a/x\} \preceq U\{^a/x\} \oplus V\{^a/x\}$, by definition of \preceq .

If $U\{^c/y\} \preceq \exists y.U$ then $(U\{^c/y\})\{^a/x\} \preceq (\exists y.U)\{^a/x\}$ since it is equivalent to $(U\{^a/x\})\{^c/y\} \preceq \exists y.(U\{^a/x\})$.

The induction hypothesis states that $U\{^b/y\} \preceq V\{^b/y\}$ implies $(U\{^b/y\})\{^a/x\} \preceq (V\{^b/y\})\{^a/x\}$. The latter relation is equivalent to $(U\{^a/x\})\{^b/y\} \preceq (V\{^a/x\})\{^b/y\}$, which implies that $\exists y.(U\{^a/x\}) \preceq \exists y.(V\{^a/x\})$ or equivalently $(\exists y.U)\{^a/x\} \preceq (\exists y.V)\{^a/x\}$.

We shall prove that $\exists y.U \preceq \exists y.V$ implies $(\exists y.U)\{^a/x\} \preceq (\exists y.V)\{^a/x\}$. If the relation $\exists y.U \preceq \exists y.V$ holds, then $U\{^b/y\} \preceq V\{^b/y\}$ for every name b . The induction hypothesis completes the proof. □

By proving the following lemma, we know that if a triple satisfies a query pattern then it satisfies all less restrictive query patterns. The same property is valid for the profiles and ask queries.

Theorem 3.1 (policy satisfaction). If $U \leq V$ and $D \models U$ then $D \models V$.

Proof. The proof is by induction on the derivation of $U \leq V$.
 If $U \leq U \oplus V$ and $D \models U$ then $D \models U \oplus V$ holds by [Q-ChoiceL].
 If $V \leq U \oplus V$ and $D \models V$ then $D \models U \oplus V$ holds by [Q-ChoiceR].
 If $U\{^a/x\} \leq \exists x.U$ and $D \models U\{^a/x\}$ then $D \models \exists x.U$ holds by [Q-Exists].
 Let $\exists x.U \leq \exists x.V$ and $D \models \exists x.U$. We can conclude from the definition of \models that there is a name a such that $D \models U\{^a/x\}$. It holds by Lemma 3.1 that $U\{^a/x\} \leq V\{^a/x\}$. By induction hypothesis, we have that $D \models V\{^a/x\}$. Finally, by the definition of \models we deduce $D \models \exists x.V$. \square

3.2. Typing

We distinguish four kinds of types, given by the following syntax:

$Name(U)$	name type	$Data(U)$	data type
$Process(U)$	process type	$Network$	network type.

They correspond to four kinds of the calculus primitives. The type $Name(U)$ denotes a name which has the privacy policy U . The data whose all triples and all triples' components have policies which are not more restrictive than U have type $Data(U)$. A process with the type $Process(U)$ does not update or delete data of users with more restrictive policies than U . $Network$ denotes the type of the network. A *type environment* Γ associates name and data variables with name and data types, i.e. we define:

$$\Gamma ::= \emptyset \mid \Gamma, x : Name(U) \mid \Gamma, \chi : Data(U).$$

The empty list is a well-formed environment. We denote by $dom(\Gamma)$ the set of all name and data variables that appear in Γ . For a well-formed environment Γ , we say that $\Gamma, x : Name(U)$ is well formed if $x \notin dom(\Gamma)$, and similarly, $\Gamma, \chi : Data(U)$ is well formed if $\chi \notin dom(\Gamma)$.

The four kinds of types induce the four forms of judgments:

$$\Gamma \vdash u : Name(U) \quad \Gamma \vdash D : Data(U) \quad \Gamma \vdash P : Process(U) \quad \Gamma \vdash N : Network.$$

The first three state that the name u , the data D and the process P are well typed under the environment Γ for the privacy protection policy U . The forth judgment states that the network N is well typed under Γ . We use the environment by the standard axioms:

[T-NAME VARIABLE]	[T-DATA VARIABLE]
$\Gamma, x : Name(U) \vdash x : Name(U)$	$\Gamma, \chi : Data(U) \vdash \chi : Data(U)$.

3.2.1. *Typing rules for names and data.* Typing rules for names and data, given in Table 7, state the following:

- If a query pattern (privacy protection policy) is assigned to a name by the function \mathcal{T} , then that name is well typed under any environment for the assigned query pattern. ([T-Name])

Table 7. Typing rules for names and data.

$\frac{[T\text{-NAME}]}{\Gamma \vdash a : \text{Name}(U)} \quad \begin{array}{l} T(a) = U \\ \Gamma \vdash \emptyset : \text{Data}(U) \end{array}$	$\frac{[T\text{-EMPTY DATA}]}{\Gamma \vdash \emptyset : \text{Data}(U)}$
$\frac{[T\text{-TRIPLE}]}{\Gamma \vdash (a_1, a_2, a_3)^U : \text{Data}(V)}$	$\frac{[T\text{-PARALLEL DATA}]}{\Gamma \vdash D_1 D_2 : \text{Data}(U)}$
$(\text{superscript } i \in I \text{ means for every } i \in I)$	

- Empty data is well typed under any environment and privacy protection policy. ([T-Empty Data])
- The privacy protection policy of a triple is more restrictive than the privacy protection policies of its subject, predicate and object (expressed by the relation $U \leq U_i$ for each $i \in \{1, 2, 3\}$). This ensures that access to the triple is forbidden to processes that do not have the permission to access any of its component names. The triple is well typed for a query V if the policy U of the triple is less restrictive than the query ($V \leq U$). ([T-Triple])
- Parallel composition of data is well typed for a privacy protection policy if the composed data have less restrictive privacy protection policies than the specified policy. ([T-Parallel Data])

3.2.2. Typing rules for processes. Typing rules for processes, given in Table 8, state the following:

- The terminating process and process variables are well typed under any environment and privacy protection policy. ([T-Termination], [T-Process Variable])
- Choice and parallel composition of two processes are well typed under an environment for a privacy protection policy if both processes are well typed under the same environment for the same privacy protection policy. ([T-Choice], [T-Parallel Process])
- A recursive process is well typed under an environment if its body is well typed for the same environment. ([T-Recursion])
- Read and select processes are well typed under an environment if their continuing processes are well typed under corresponding augmented environments. The enhancement is formed by adding an association between data variable or name variable with data or name type corresponding to the privacy protection policy of the location where we intend to read or select data. ([T-Read], [T-Select])
- A process that wants to write data well typed for W to a location well typed for U_u is well typed for V if U_u is more restrictive than W and continuation of the process is well typed for V . This ensure that users that have access to the whole profile will still have that privilege after the profile is extended with new data. ([T-Write])
- A process aiming to delete or update data of a user that is well typed for a privacy protection policy U_u are well typed for V if $V \leq U_u$ and the continuation is well typed for V . The update requires also the new privacy protection policy W to be

Table 8. Typing rules for processes.

$\frac{[\text{T-TERMINATION}]}{\Gamma \vdash 0 : \text{Process}(V)}$	$\frac{[\text{T-CHOICE}]}{\Gamma \vdash P_i : \text{Process}(V) \quad (i \in \{1,2\})} \quad \Gamma \vdash P_1 + P_2 : \text{Process}(V)$	$\frac{[\text{T-PARALLEL PROCESS}]}{\Gamma \vdash P_i : \text{Process}(V) \quad (i \in \{1,2\})} \quad \Gamma \vdash P_1 P_2 : \text{Process}(V)$
$[\text{T-PROCESS VARIABLE}] \quad \Gamma \vdash X : \text{Process}(V)$	$\frac{[\text{T-RECURSION}]}{\Gamma \vdash \text{rec}X.P : \text{Process}(V)}$	$\frac{[\text{T-READ}]}{\Gamma, \chi : \text{Data}(W) \vdash P : \text{Process}(V)} \quad \Gamma \vdash \text{READ}_u(U, \chi).P : \text{Process}(V)$
$[\text{T-WRITE}] \quad \Gamma \vdash u : \text{Name}(U_u) \quad \Gamma \vdash \delta : \text{Data}(W) \quad \Gamma \vdash P : \text{Process}(V) \quad U_u \leq W$	$\frac{\Gamma \vdash \text{WRITE}_u(\delta).P : \text{Process}(V)}{\Gamma \vdash \text{WRITE}_u(\delta).P : \text{Process}(V)}$	
$[\text{T-UPDATE}] \quad \Gamma \vdash u : \text{Name}(U_u) \quad \Gamma \vdash P : \text{Process}(V) \quad V \leq U_u \leq W$		$[\text{T-DELETE}] \quad \Gamma \vdash u : \text{Name}(U_u) \quad \Gamma \vdash P : \text{Process}(V) \quad V \leq U_u$
$\Gamma \vdash \text{UPDATE}_u(U, W).P : \text{Process}(V)$		$\Gamma \vdash \text{DELETE}_u(U).P : \text{Process}(V)$
$[\text{T-SELECT}] \quad \Gamma, x : \text{Name}(U_x) \vdash P : \text{Process}(V)$		$\Gamma \vdash \text{SELECT}_u(\exists x.U).P : \text{Process}(V)$

Table 9. Typing rules for networks.

$\frac{[\text{T-USER}] \quad \emptyset \vdash a : \text{Name}(U) \quad \emptyset \vdash D : \text{Data}(U) \quad \emptyset \vdash P : \text{Process}(U) \quad D \models U}{\emptyset \vdash a[D \parallel P] : \text{Network}}$	$\frac{[\text{T-PARALLEL NETWORK}]}{\emptyset \vdash N_i : \text{Network} \quad (i \in \{1,2\})} \quad \emptyset \vdash N_1 N_2 : \text{Network}$
---	---

less restrictive than U_u . This ensures that the data stays available to its owner after an update has been executed and that delete is to be done only by the owner of the data. ([T-Delete], [T-Update])

3.2.3. *Typing rules for networks.* Typing rules for networks, given in Table 9, state the following:

- A user with enclosed data (profile) and process is well typed, if the user name, the data and the process are well typed for the same privacy protection policy. A straightforward consequence of the type assignment rules is the following: if a user $a[D \parallel P]$ is well typed, then both the data D and the process P do not contain occurrences of free variables. ([T-User])
- Parallel composition of well-typed networks is a well-typed network. ([T-Parallel Network])

4. Properties

In the current section, we prove that the proposed type system guarantees the operational property of type preservation under reduction: all networks obtained by reduction starting in a typed network are again typed. First, we need a few preliminary lemmas in order to enable the reversal of the typing rules. The proposed type system is syntax directed, hence the proofs of the following lemmas are straightforward.

We introduce an extra meta-variable:

$$\omega ::= \text{Name}(U) \mid \text{Data}(U) \mid \text{Process}(U) \mid \text{Network}.$$

Lemma 4.1 (generation lemma for variables).

1. $\Gamma \vdash x : \omega \Rightarrow x : \omega \in \Gamma$ and $\omega = \text{Name}(U)$.
2. $\Gamma \vdash \chi : \omega \Rightarrow \chi : \omega \in \Gamma$ and $\omega = \text{Data}(U)$.

Lemma 4.2 (generation lemma for names and data).

1. $\Gamma \vdash a : \omega \Rightarrow \mathcal{T}(a) = U_a$ and $\omega = \text{Name}(U_a)$.
2. $\Gamma \vdash \emptyset : \omega \Rightarrow \omega = \text{Data}(U)$.
3. $\Gamma \vdash (a_1, a_2, a_3)^U : \omega \Rightarrow \omega = \text{Data}(V)$ and $\Gamma \vdash a_i : \text{Name}(U_i)$ and $V \preceq U \preceq U_i$ for every $i \in \{1, 2, 3\}$.
1. $\Gamma \vdash D_1 | D_2 : \omega \Rightarrow \omega = \text{Data}(U)$ and $\Gamma \vdash D_i : \text{Data}(U_i)$ and $U \preceq U_i$ for every $i \in \{1, 2\}$.

Lemma 4.3 (generation lemma for processes).

1. $\Gamma \vdash 0 : \omega \Rightarrow \omega = \text{Process}(V)$.
2. $\Gamma \vdash P_1 + P_2 : \omega \Rightarrow \omega = \text{Process}(V)$ and $\Gamma \vdash P_1 : \text{Process}(V)$ and $\Gamma \vdash P_2 : \text{Process}(V)$.
3. $\Gamma \vdash P_1 | P_2 : \omega \Rightarrow \omega = \text{Process}(V)$ and $\Gamma \vdash P_1 : \text{Process}(V)$ and $\Gamma \vdash P_2 : \text{Process}(V)$.
4. $\Gamma \vdash X : \omega \Rightarrow \omega = \text{Process}(V)$.
5. $\Gamma \vdash \text{rec}X.P : \omega \Rightarrow \omega = \text{Process}(V)$ and $\Gamma \vdash P : \text{Process}(V)$.
6. $\Gamma \vdash \text{READ}_u(U, \chi).P : \omega \Rightarrow \omega = \text{Process}(V)$ and $\Gamma, \chi : \text{Data}(W) \vdash P : \text{Process}(V)$.
7. $\Gamma \vdash \text{WRITE}_u(\delta).P : \omega \Rightarrow \omega = \text{Process}(V)$ and $\Gamma \vdash u : \text{Name}(U_u)$ and $\Gamma \vdash \delta : \text{Data}(W)$ and $\Gamma \vdash P : \text{Process}(V)$ and $U_u \preceq W$.
8. $\Gamma \vdash \text{SELECT}_u(\exists x.U).P : \omega \Rightarrow \omega = \text{Process}(V)$ and $\Gamma, x : \text{Name}(U_x) \vdash P : \text{Process}(V)$.
9. $\Gamma \vdash \text{UPDATE}_u(U, W).P : \omega \Rightarrow \omega = \text{Process}(V)$ and $\Gamma \vdash u : \text{Name}(U_u)$ and $\Gamma \vdash P : \text{Process}(V)$ and $V \preceq U_u \preceq W$.
10. $\Gamma \vdash \text{DELETE}_u(U).P : \omega \Rightarrow \omega = \text{Process}(V)$ and $\Gamma \vdash u : \text{Name}(U_u)$ and $\Gamma \vdash P : \text{Process}(V)$ and $V \preceq U_u$.

Lemma 4.4 (generation lemma for networks).

1. $\emptyset \vdash a[D \parallel P] : \omega \Rightarrow \omega = \text{Network}$ and $\emptyset \vdash a : \text{Name}(U)$ and $\emptyset \vdash D : \text{Data}(U)$ and $\emptyset \vdash P : \text{Process}(U)$ and $D \models U$.
2. $\emptyset \vdash N_1 | N_2 : \omega \Rightarrow \omega = \text{Network}$ and $\emptyset \vdash N_1 : \text{Network}$ and $\emptyset \vdash N_2 : \text{Network}$.

Lemma 4.5 (substitution).

1. If $\Gamma, \chi : \text{Data}(W) \vdash P : \text{Process}(V)$ and $\Gamma \vdash D : \text{Data}(W)$ then $\Gamma \vdash P\{D/\chi\} : \text{Process}(V)$.
2. If $\Gamma, x : \text{Name}(U_x) \vdash P : \text{Process}(V)$ and $\Gamma \vdash a : \text{Name}(U_x)$ then $\Gamma \vdash P\{a/x\} : \text{Process}(V)$.
3. If $\Gamma, X : \text{Process}(U) \vdash P : \text{Process}(V)$ and $\Gamma \vdash Q : \text{Process}(U)$ then $\Gamma \vdash P\{Q/X\} : \text{Process}(V)$.

Lemma 4.6. Let $\Gamma \vdash D : \text{Data}(V)$. If

1. $D' = \text{readable}(D_u, D)$,
2. $D' = \text{read}(U, D)$,
3. $D' = \text{delete}(U, D)$, or
4. $D' = \text{update}(U, D, W)$ and $V \preceq W$,

then $\Gamma \vdash D' : \text{Data}(V)$.

Proof. The proof is by induction on the structure of D . We write the proof for read, other cases are similar.

If $D = (a, b, c)^U$ then $V \leq U$ by Lemma 4.2.3. By definition of read, $D' = \emptyset$ or $D' = D$. In both cases, applying typing rules for data, we derive $\Gamma \vdash D' : \text{Data}(V)$.

Assume that $\Gamma \vdash D'_1 : \text{Data}(V)$ and $\Gamma \vdash D'_2 : \text{Data}(V)$ for $D'_1 = \text{read}(U, D_1)$, and $D'_2 = \text{read}(U, D_2)$. We shall prove $\Gamma \vdash D' : \text{Data}(V)$ for $D' = \text{read}(U, D_1 | D_2) = \text{read}(U, D_1) | \text{read}(U, D_2)$, which follows from the induction hypothesis and [T-Parallel Data].

□

Lemma 4.7. If $\emptyset \vdash N_1 : \text{Network}$ and $N_1 \equiv N_2$, then $\emptyset \vdash N_2 : \text{Network}$.

Finally, we prove that the proposed type system enjoys the preservation of types under reduction.

Theorem 4.1 (subject reduction). If $\emptyset \vdash N_1 : \text{Network}$ and $N_1 \rightarrow N_2$ then $\emptyset \vdash N_2 : \text{Network}$.

Proof. The proof is by induction on the derivation of $N_1 \rightarrow N_2$. We have two base cases, from the reduction rules [R-User] and [R-Self].

[R-User] Suppose $a \neq b$ and $\emptyset \vdash a[D_a \parallel P] \mid b[D_b \parallel Q] : \text{Network}$. We have to show $\emptyset \vdash a[D_a \parallel P'] \mid b[D \parallel Q] : \text{Network}$, which will follow from the typing rules if we prove $\emptyset \vdash P' : \text{Process}(U_a)$ and $\emptyset \vdash D : \text{Data}(U_b)$. The latter two will be proved by induction on the derivation of $(P, D_b) \rightsquigarrow_a (P', D)$. We consider the following five base cases.

[I-Read] $P = \text{READ}_b(U, \chi).P_1$, $P' = P_1\{D'/\chi\}$, $D' = \text{read}(U, \text{readable}(D_a, D_b))$, $D = D_b$: By Lemma 4.3.6, $\emptyset \vdash P : \text{Process}(U_a)$ implies $\chi : \text{Data}(W) \vdash P_1 : \text{Process}(U_a)$. We conclude by Lemma 4.6.1–2 and Lemma 4.5.1 that $\emptyset \vdash P_1\{D'/\chi\} : \text{Process}(U_a)$.

[I-Write] $P = \text{WRITE}_b(D').P_1$, $P' = P_1$, $D = D_b | D'$: From $\emptyset \vdash P : \text{Process}(U_a)$, by Lemma 4.3.7, it holds that $\emptyset \vdash P_1 : \text{Process}(U_a)$ and $\emptyset \vdash D' : \text{Data}(W)$ and $U_b \leq W$. Applying the typing rule [T-Parallel Data] we get $\emptyset \vdash D : \text{Data}(U_b)$.

[I-Select] $P = \text{SELECT}_b(\exists x.U).P_1$, $P' = \prod_{s \in S} P_1 s$, $S = \text{select}(\exists x.U, \text{readable}(D_a, D_b))$, $D = D_b$: By Lemma 4.3.8, Lemma 4.5.2 and [T-Parallel Process], $\emptyset \vdash P' : \text{Process}(U_a)$.

[I-Update] $P = \text{UPDATE}_b(U, W).P_1$, $P' = P_1$, $D = \text{update}(U, D_b, W)$: By Lemma 4.3.9, $\emptyset \vdash P : \text{Process}(U_a)$ implies $\emptyset \vdash b : \text{Name}(U_b)$ and $\emptyset \vdash P_1 : \text{Process}(U_a)$ and $U_a \leq U_b \leq W$. According to Lemma 4.6.4 it follows that $\emptyset \vdash D : \text{Data}(U_b)$.

[I-Delete] $P = \text{DELETE}_b(U).P_1$, $P' = P_1$, $D = \text{delete}(U, D_b)$: By Lemma 4.3.10, $\emptyset \vdash P : \text{Process}(U_a)$ implies $\emptyset \vdash P_1 : \text{Process}(U_a)$, and by Lemma 4.6.3 we get $\emptyset \vdash D : \text{Data}(U_b)$.

Assume the induction hypothesis: $(P, D_b) \rightsquigarrow_a (P', D)$ implies $\emptyset \vdash P' : Process(U_a)$ and $\emptyset \vdash D : Data(U_b)$. Applying the typing rules [T-Parallel Process] and [T-Recursion], we obtain

If $(P + Q, D_b) \rightsquigarrow_a (P', D)$ then $\emptyset \vdash P' : Process(U_a)$ and $\emptyset \vdash D : Data(U_b)$.

If $(P|Q, D_b) \rightsquigarrow_a (P'|Q, D)$ then $\emptyset \vdash P'|Q : Process(U_a)$ and $\emptyset \vdash D : Data(U_b)$.

If $(recX.P_1, D_b) \rightsquigarrow_a (P', D)$ and $P = P_1\{^{recX.P_1}/X\}$ then $\emptyset \vdash P' : Process(U_a)$ and $\emptyset \vdash D : Data(U_b)$.

[R-Self] Suppose $\emptyset \vdash a[D_a \parallel P] : Network$. We have to show that $\emptyset \vdash a[D_a \parallel P'] : Network$ i.e. $\emptyset \vdash P' : Process(U_a)$ for $(P, D_a) \rightsquigarrow_a (P', D)$. The proof is similar to the previous case.

Suppose that $\emptyset \vdash N_1 : Network$ and $N_1 \rightarrow N_2$ imply $\emptyset \vdash N_2 : Network$.

[R-Parallel] If $\emptyset \vdash N_1 \mid N : Network$ and $N_1 \mid N \rightarrow N_2 \mid N$, by Lemma 4.4.2 and the typing rule [T-Parallel Network], $\emptyset \vdash N_2 \mid N : Network$.

[R-Struct] If $N_1 \equiv N'_1$, $N_2 \equiv N'_2$, $N'_1 \rightarrow N'_2$ and $\emptyset \vdash N'_1 : Network$. By Lemma 4.7, $\emptyset \vdash N_1 : Network$, and by the induction hypothesis we conclude $\emptyset \vdash N_2 : Network$. Applying now Lemma 4.7 to $N_2 \equiv N'_2$, we get $\emptyset \vdash N'_2 : Network$. □

Corollary 4.1. If $\emptyset \vdash N_1 : Network$ and $N_1 \rightarrow^* N_2$ then $\emptyset \vdash N_2 : Network$.

Lemma 4.8. If $\emptyset \vdash a[D_a \parallel P_a] : Network$ and $\emptyset \vdash a : Name(U_a)$, then $readable(D_a, D_a) = D_a$.

Proof. The proof follows by induction on the structure of D_a . If $D_a = T^V$ then, by Lemma 4.2.3, $U_a \leq V$. From Lemma 4.4.1 it holds that $T^V \models U_a$, by Theorem 3.1 we conclude $T^V \models V$, and therefore $readable(D_a, T^V) = T^V$. Assume that $D_a = D'|D''$ and $readable(D_a, D') = D'$ and $readable(D_a, D'') = D''$. Then $readable(D_a, D) = readable(D_a, D'|D'') = readable(D_a, D') \mid readable(D_a, D'') = D'|D'' = D_a$. □

Theorem 4.2. If $\emptyset \vdash N : Network$, then N is well behaved.

Proof. Let $N \rightarrow^* a[D_a \parallel P] \mid N'$, $\mathcal{T}(a) = U_a$, and $\emptyset \vdash Network$. We can conclude, by Corollary 4.1 and Lemma 4.4.2, that $\emptyset \vdash a[D_a \parallel P] : Network$ and $\emptyset \vdash N' : Network$.

- i. Assume that $D_a \equiv (a_1, a_2, a_3)^U \mid D'_a$ and $\mathcal{T}(a_i) = U_i$, $i \in \{1, 2, 3\}$. It holds by Lemma 4.4.1 that $\emptyset \models (a_1, a_2, a_3)^U \mid D'_a : Data(U_a)$. By Lemma 4.2.3, $U_a \leq U \leq U_i$ for every $i \in \{1, 2, 3\}$. We conclude by applying the Theorem 3.1.
- ii. $readable(D_a, D_a) = D_a$ is straightforward consequence of the Lemma 4.8 and Lemma 4.4.2.
- iii. – Assume that $N' \equiv b[D_b \parallel Q] \mid N''$ and $(P \equiv DELETE_b(U).R$ or $P \equiv UPDATE_b(U, W).R)$ and $\mathcal{T}(a) = U_a$. By Lemma 4.4.2, $\emptyset \vdash P : Process(U_a)$, and by Lemma 4.3.9–10 $U_a \leq U_b$. We proceed by induction on the structure of D_b . If $D_b = T^U$ then, by Lemma 4.2, $U_b \leq U$. From $U_a \leq U_b \leq U$, by Theorem 3.1, since $D_a \models U_a$ it follows $D_a \models U$, and therefore also $readable(D_a, T^U) = T^U$. Induction shows $readable(D_a, D'_b \mid D''_b) = readable(D_a, D'_b) \mid readable(D_a, D''_b) = D'_b \mid D''_b = D_b$.

- Assume that $N' \equiv b[D_b \parallel Q] \mid N''$ and $(P \equiv \text{WRITE}_b(D').R$ or $P \equiv \text{UPDATE}_b(U, W).R)$ and $(P, D_b) \rightsquigarrow_a (P, D'_b)$ and $\mathcal{T}(b) = U_b$. It follows from [I-Write] and [I-Update] that $D'_b \equiv D_b \mid D'$ or $D'_b \equiv \text{update}(U, D_b, W)$. According to Lemma 4.6.4 and in the former case [T-Parallel Data] it holds $\emptyset \vdash D'_b : \text{Data}(U_b)$. We conclude by Theorem 4.1 that $D'_b \models U_b$ and from $U_b \leq W$ we have $D'_b \models W$. Therefore, $\text{readable}(D'_b, D'_b) = D'_b$.

□

5. Related work

The Linked Data is currently present on the Web in many areas: media, publications, life sciences, geographic data and user generated content (Berners-Lee *et al.* 2001; Bizer 2009; Bizer *et al.* 2009; Heath 2011; Heath and Bizer 2001). In the present calculus, we have mostly focused on privacy issues for user generated content. There is a great number of references on technologies and tools for the Linked Data. Here we name RDF (Brickley and Guha 2004; Klyne and Carroll 2004), SPARQL (Haris *et al.* 2011; Prud'hommeaux and Seaborne 2008), FOAF (Friend of a Friend, a common vocabulary that describes RDF data, <http://www.foaf-project.org/>), SIOC exporters (Linked Data wrappers for blogging engines, content management systems and discussion forums, <http://sioc-project.org/exporters>), Zemanta (provides tools for the semiautomated enrichment of blog posts with data-level links pointing to DBpedia, Freebase, MusicBrainz, and Semantic Crunch-Base), HyperTwitter and Twarql.

Horne and Sassone (2011b); Horne *et al.* (2011) provide an abstract syntax in order to capture Linked Data structures, queries and updates which are then internalized in a process calculus. The abstract syntax of RDF of Klyne and Carroll (2004) was used as base for the introduced model of stored Linked Data. The interaction of the stored data in RDF format, SPARQL queries and applications that consume the queried data is described by operational semantics. We have introduced the user profiles and privacy policies as suggested in Sacco and Passant (2011a,b); Stanković *et al.* (2009) where the authors propose the authentication of users based on their FOAF profiles and SPARQL query as privacy preference checkers.

In Dezani *et al.* (2012), the authors study provenance for Linked Data and introduce a typed calculus for modelling interaction between processes and Linked Data, tracing where the data has been published and who published it. The syntax of our calculus is similar to the one of Dezani *et al.* (2012) in the sense that both calculi distinguish between processes and data and describe their interaction. In the present calculus, processes run on behalf of data and both the data and the process are enclosed with a user name (like in named graphs), while in the other calculus linked data is open. Since the two calculi model orthogonal problems, triples are decorated with completely different annotations, while the tool for checking whether a data satisfy a query is the same. The semantics differ quite a lot since in our model it is essential that the operations are performed on the entire observed data (think of updating privacy policy: it must be updated on all triples satisfying some condition i.e. query) while other related calculi do not have to guarantee this.

In this paper, by means of a type system, we verify preservation of privacy properties from Sacco and Passant (2011a,b), where the vocabulary for fine-grained privacy preference control and a tool for privacy preference management were defined, as well as several more general properties. Basic type systems for process calculi could be found in Hennessy (2007) and Sangiorgi and Walker (2001). Type systems were successfully used to analyse a wide range of security properties (authenticity, security, safety, secrecy, privacy...), here we mention the following: in Abadi and Blanchet (2003) and Abadi *et al.* (2006) computational secrecy control for asynchronous π -calculus (Merro and Sangiorgi 1998) by introducing types for public and private channels and proving computational soundness theorem; in Fournet *et al.* (2003) the type system for objective join-calculus guarantees that private labels are accessed only from the body of a class used to create the object; in Fournet *et al.* (2007) policy conformance in a distributed system is verified with a type system; in Haack and Jeffrey (2005) authenticity and secrecy of well-typed protocols in timed spi-calculus. In Horne and Sassone (2011a) a type and a subtype system verify consistency of URI names usage. The most related type assignment systems are (Dezani *et al.* 2010, 2008), where the safety properties of data in XML format have been proved. The type system from the present paper is equipped with query comparison relation which is tailored especially for the investigated kind of privacy control.

6. Conclusion

We have introduced a core language of processes that interact with data in RDF format, modelling a fragment of SPARQL and a higher order language that consumes the data. These processes together with data are enclosed under named users which are put in parallel, representing a network of users interacting with each other. We define the desired privacy properties of the network by defining well-behaved network. What distinguishes this work from others analysing variety of security properties is the fact that it does not feature any additional means for privacy control beside those that are already present in the syntax of the calculus, i.e. privacy policies are expressed as queries and policy satisfaction comes to query satisfaction with users profiles in RDF format. We have studied a type system guaranteeing some privacy properties by proving that well-typed network is well behaved. The key of the type system simplicity lies in the introduced query comparison relation and the chosen set of processes. The calculus is equipped only with processes essential for the privacy study done in the paper. With the extension of the process language we would get more complicated semantics and the type system without significant impact on the privacy properties which we have discussed.

This paper contributes to the expanding trend of building the Web of Linked Data. To the best of our knowledge, this is the first typed calculus that tackles privacy protection for Linked Data. The proposed typed calculus provides a ground model for the development of type checkers for high level languages for Linked Data. When the Web of Linked Data becomes significantly spread out, the natural step forward will be the study of formal models of applications that consume the Web of Linked Data.

Acknowledgement

We wish to thank the anonymous reviewers who have provided suggestions for improving the paper. We are grateful to Mariangiola Dezani-Ciancaglini for reading and providing valuable comments on an early version of the paper.

References

- Abadi, M. and Blanchet, B. (2003). Secrecy types for asymmetric communication. *Theoretical Computer Science* **3** (298) 387–415.
- Abadi, M., Corin, R. and Fournet, C. (2006). Computational secrecy by typing for the pi calculus. In: Kobayashi, N. (ed.) APLAS. *Springer Lecture Notes in Computer Science* **4279** 253–269.
- Berners-Lee, T., Hendler, J. and Lassila, O. (2001). The semantic web. *Scientific Am.* **284** (5) 35–43.
- Bizer, C. (2009). The emerging web of linked data. *IEEE Intelligent Systems* **24** 87–92.
- Bizer, C., Heath, T. and Berners-Lee, T. (2009). Linked data - the story so far. *International Journal on Semantic Web and Information Systems* **5** (3) 1–22.
- Brickley, D. and Guha, R. V. (2004). RDF Vocabulary Description Language 1.0: RDF Schema. W3C, MIT, Cambridge, MA. REC-rdf-schema-20040210.
- Dezani-Ciancaglini, M., Ghilezan, S., Jakšić, S. and Pantović J. (2010). Types for role-based access control of dynamic web data. In: WFLP'10. *Lecture Notes in Computer Science* **6559** 1–29.
- Dezani-Ciancaglini, M., Ghilezan, S., Pantović, J. and Varacca, D. (2008). Security types for dynamic web data. *Theoretical Computer Science* **402** (2–3) 156–171.
- Dezani-Ciancaglini, M., Horne, R. and Sassone, V. (2012). Tracing where and who provenance in Linked Data: A calculus. *Theoretical Computer Science* **464** 113–129.
- Fournet, C., Gordon, A. and Maffeis, S. (2007). A type discipline for authorization in distributed systems. In: CSF IEEE Computer Society 31–48.
- Fournet, C., Laneve, C., Maranget, L. and Rémy, D. (2003). Inheritance in the join calculus. *Journal of Logic and Algebraic Programming* **57** (1-2) 23–69.
- Haack, C. and Jeffrey, A. (2005). Timed spi-calculus with types for secrecy and authenticity. In: Abadi, M. and de Alfaro, L. (eds.) CONCUR. *Springer Lecture Notes in Computer Science* **3653** 202–216.
- Haris, S., Seaborne, A. and Prud'hommeaux, E. (2011). SPARQL 1.1 query language. W3C, MIT, Cambridge, MA. WD-sparql11-query-20110512.
- Heath, T. (2011) Linked data - welcome to the data network. *IEEE Internet Computing* **15** (6) 70–73.
- Heath, T. and Bizer, C. (2001). Linked data: Evolving the web into a global data space. *Synthesis Lectures on the Semantic Web: Theory and Technology* **1** (1) 1–136.
- Hennessy, M. (2007). *A Distributed Pi-Calculus*, Cambridge University Press.
- Horne, R. and Sassone, V. (2011a). A typed model for linked data. *Technical Report*, available online at <http://eprints.ecs.soton.ac.uk/21996/5/paper.pdf>.
- Horne, R. and Sassone, V. (2011b). A verified algebra for linked data. In: Mousavi, M. R. and Ravara, A. (eds.) FOCLASA. *Electronic Proceedings in Theoretical Computer Science* **58** 20–33.
- Horne, R., Sassone, V. and Gibbins, N. (2011). Operational semantics for SPARQL update. In: *1st Joint International Semantic Technology Conference*, 4–7th December 2011, Hangzhou, China 1–16.
- Klyne, G. and Carroll, J. (2004). Resource description framework: Concepts and abstract syntax. W3C, MIT, Cambridge, MA, REC-rdf-concepts-20040210.

- Merro, M. and Sangiorgi, D. (1998). On asynchrony in name-passing calculi. In: Larsen, K. G., Skyum, S. and Winskel, G. (eds.) ICALP. *Springer Lecture Notes in Computer Science* **1443** 856–867.
- Milner, R. (1999). *Communicating and Mobile Systems: The π -Calculus*, Cambridge University Press.
- Milner, R., Parrow, J. and Walker, D. (1992). A calculus of mobile processes, part I and II. *Information and Computation* **100** (1) 1–40.
- Prud'hommeaux, E. and Seaborne, A. (2008). SPARQL query language for RDF. W3C, MIT, Cambridge, MA. REC-rdf-sparql-query-20080115.
- Sacco, O. and Passant, A. (2011a). A privacy preference ontology (ppo) for linked data. In: *Proceedings of the Linked Data on the Web Workshop (LDOW2011)*, available online at <http://ceur-ws.org/Vol-813/>.
- Sacco, O. and Passant, A. (2011b). A privacy preference manager for the social semantic Web. In: *Proceedings of the 2nd Workshop on Semantic Personalized Information Management: Retrieval and Recommendation* 42–53.
- Sangiorgi, D. and Walker, D. (2001). *The π -Calculus: A Theory of Mobile Processes*, Cambridge University Press.
- Stanković, M., Passant, A. and Laublet, P. (2009). Directing status messages to their audience in online communities. In: COIN@AAMAS&IJCAI&MALLOW. *Springer Lecture Notes in Computer Science* **6069** 195–210.
- Westin, A. (1967). *Privacy and Freedom*, New York: Atheneum.