Under consideration for publication in Math. Struct. in Comp. Science

Linked Data Privacy

Svetlana Jakšić, Jovanka Pantović, Silvia Ghilezan [†]

University of Novi Sad, Trg Dositeja Obradovica 6, Serbia

Received 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.

Contents

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

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 (Klyne and Caroll 2004; Bizer 2009). 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 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 Caroll 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 modeling 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, without input and output capabilities, and with introduced capabilities that describe the interaction between processes and data. Networks are parallel composition 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 2011-2). 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

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

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

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

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 $\mathscr{T}$ 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	$(\texttt{Alice},\texttt{affiliation},\texttt{www.uns.ac.rs})^{U_{\texttt{Alice}}}\vert post_1^{U_1}\vert post_2^{U_2}\vert post_3^{U_3}$
Bob	$(\texttt{Bob},\texttt{friend},\texttt{Cindy})^{U_{\texttt{Bob}}}\vert(\texttt{Bob},\texttt{research},\texttt{types})^{U_{\texttt{Bob}}}$

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 $\mathscr{T}(\texttt{Alice}) = U_{\texttt{Alice}}$ with $U_{\texttt{Alice}} = (\texttt{Alice}, \texttt{affiliation}, \texttt{www.uns.ac.rs})$. 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

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

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 $(\texttt{Alice}, \texttt{affiliation}, \texttt{www.uns.ac.rs})^U$, like Alice, can access her profile (for an arbitrary U). This is the case because such a profile satisfies the given query $U_{\texttt{Alice}}$. We denote by $\models$ the relation that states that data satisfy query. We will write

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

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

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

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

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

This property will be written as

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

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

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

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:

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

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 $\preccurlyeq$ between privacy protection policies (i.e. between queries). For example, it holds that

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

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

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

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

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

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

Let the process running on behalf of Bob be

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

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

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

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:

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

It will be well typed in our calculus since

$$U_{\texttt{Alice}} \preccurlyeq U_2 \oplus (\texttt{Bob},\texttt{friend},\texttt{Cindy})$$

and hence

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

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 organised 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.

Table 1. Syntax of query pattern and data

			query pattern				**data**
U	$::=$	(u,u,u)	triple pattern	D	$::=$	$\emptyset$	empty data
	$\mid$	$U \oplus U$	choice		$\mid$	$(a,a,a)^U$	triple with query pattern
	$\mid$	$\exists x.U$	exists		$\mid$	$D \mid D$	parallel

2. Language

In this section, we formally introduce the calculus.

2.1. *Syntax*

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

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 exists 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 $U,V,W,\ldots$.

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 $\chi,\psi,\ldots$. We let δ,ε range over data and data variables.

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

— $\mathtt{READ}_u(U,\chi).P$ that expresses a process that reads from the user named u the data D found by the query U and then behaves like $P\{^D/_\chi\}$;
— $\mathtt{WRITE}_u(\delta).P$ that expresses a process that writes the data δ to the user named u and then behaves like P;
— $\mathtt{DELETE}_u(U).P$ that expresses a process that deletes from the user named u the data found by the query U and then behaves like P;
— $\mathtt{SELECT}_u(\exists x.U).P$ 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
— $\mathtt{UPDATE}_u(U_1,U_2).P$ 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 $X,Y,\ldots$

Table 2. Syntax of processes and networks

P	::=		**process**	N	::=		**network**
		$\mathtt{0}$	termination			$a[D \parallel P]$	user
	$\mid$	$P+P$	choice		$\mid$	$N \mid N$	parallel
	$\mid$	$P \mid P$	parallel				
	$\mid$	$\mathtt{rec}X.P$	recursion				
	$\mid$	X	process variable				
	$\mid$	$\mathtt{READ}_u(U,\chi).P$	read				
	$\mid$	$\mathtt{WRITE}_u(\delta).P$	write				
	$\mid$	$\mathtt{DELETE}_u(U).P$	delete				
	$\mid$	$\mathtt{SELECT}_u(\exists x.U).P$	select				
	$\mid$	$\mathtt{UPDATE}_u(U,U).P$	update				

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*

Satisfaction of query patterns and queries Before defining the operational semantics we need to formalise 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:

$$\text{[Q-TRIPLE]}\ \ T^U \models T \qquad \text{[Q-CHOICEL]}\ \ \frac{D \models U}{D \models U \oplus V} \qquad \text{[Q-CHOICER]}\ \ \frac{D \models V}{D \models U \oplus V} \qquad \text{[Q-EXISTS]}\ \ \frac{D \models U\{{}^{a}/_{x}\}}{D \models \exists x.U}.$$

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

$$\text{[Q-ASK]}\ \ \frac{D' \models U}{D' \mid D'' \models U} \qquad \text{[Q-SCONG]}\ \ \frac{D' \models U \ \ D' \equiv_D D''}{D'' \models U}.$$

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.

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

— $\mathsf{readable}(D_a, D_b)$ 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 first profile can access. We will say that the first profile's user can fully access (or owns) the second profile if $\mathsf{readable}(D_a, D_b) = D_b$.

Table 3. Definitions of data manipulation functions

$$
\begin{array}{lcl}
\mathsf{readable}(D,\emptyset) & = & \emptyset \\
\mathsf{readable}(D,(a,b,c)^U) & = & \begin{cases} (a,b,c)^U & \text{if } D \models U, \\ \emptyset & \text{otherwise} \end{cases} \\
\mathsf{readable}(D,D_1|D_2) & = & \mathsf{readable}(D,D_1)|\mathsf{readable}(D,D_2)
\end{array}
$$

$$
\begin{array}{lcl}
\mathsf{read}(U,\emptyset) & = & \emptyset \\
\mathsf{read}(U,(a,b,c)^V) & = & \begin{cases} (a,b,c)^V & \text{if } (a,b,c)^V \models U, \\ \emptyset & \text{otherwise} \end{cases} \\
\mathsf{read}(U,D_1|D_2) & = & \mathsf{read}(U,D_1)|\mathsf{read}(U,D_2)
\end{array}
$$

$$
\begin{array}{lcl}
\mathsf{delete}(U,\emptyset) & = & \emptyset \\
\mathsf{delete}(U,(a,b,c)^V) & = & \begin{cases} \emptyset & \text{if } (a,b,c)^V \models U, \\ (a,b,c)^V & \text{otherwise} \end{cases} \\
\mathsf{delete}(U,D_1|D_2) & = & \mathsf{delete}(U,D_1)|\mathsf{delete}(U,D_2)
\end{array}
$$

$$
\begin{array}{lcl}
\mathsf{select}(\exists x.U,\emptyset) & = & \emptyset \\
\mathsf{select}(\exists x.U,(a,b,c)^V) & = & \begin{cases} \{^a/_x\} & \text{if } (a,b,c)^V \models U\{^a/_x\}, \\ \{^b/_x\} & \text{if } (a,b,c)^V \models U\{^b/_x\}, \\ \{^c/_x\} & \text{if } (a,b,c)^V \models U\{^c/_x\}, \\ \emptyset & \text{otherwise} \end{cases} \\
\mathsf{select}(\exists x.U,D_1|D_2) & = & \mathsf{select}(\exists x.U,D_1) \cup \mathsf{select}(\exists x.U,D_2)
\end{array}
$$

$$
\begin{array}{lcl}
\mathsf{update}(U,\emptyset,V) & = & \emptyset \\
\mathsf{update}(U,(a,b,c)^{V'},V) & = & \begin{cases} (a,b,c)^V & \text{if } (a,b,c)^{V'} \models U, \\ (a,b,c)^{V'} & \text{otherwise} \end{cases} \\
\mathsf{update}(U,D_1|D_2,V) & = & \mathsf{update}(U,D_1,V)|\mathsf{update}(U,D_2,V)
\end{array}
$$

— $\mathsf{read}(U,D)$ takes a query and data as arguments and returns the parallel composition of the data triples that satisfy the query;
— $\mathsf{delete}(U,D)$ 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;
— $\mathsf{select}(\exists x.U,D)$ takes an exists 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;
— $\mathsf{update}(U,D,V)$ 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.

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.

— 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

Table 4. Interaction rules

$$\text{[I-CHOICE]}\ \frac{(P,D) \rightsquigarrow_a (P',D')}{(P+Q,D) \rightsquigarrow_a (P',D')} \qquad \text{[I-PARALLEL]}\ \frac{(P,D) \rightsquigarrow_a (P',D')}{(P|Q,D) \rightsquigarrow_a (P'|Q,D')} \qquad \text{[I-RECURSION]}\ \frac{(P\{^{\mathtt{rec}X.P}/_X\},D) \rightsquigarrow_a (P',D')}{(\mathtt{rec}X.P,D) \rightsquigarrow_a (P',D')}$$

$$\text{[I-READ]}\ \frac{D' = \mathsf{read}(U, \mathsf{readable}(D_a, D))}{(\mathtt{READ}_b(U,\chi).P,D) \rightsquigarrow_a (P\{^{D'}/_\chi\},D)} \qquad \text{[I-WRITE]}\ (\mathtt{WRITE}_b(D').P,D) \rightsquigarrow_a (P,D|D') \qquad \text{[I-DELETE]}\ \frac{D' = \mathsf{delete}(U,D)}{(\mathtt{DELETE}_b(U).P,D) \rightsquigarrow_a (P,D')}$$

$$\text{[I-SELECT]}\ \frac{S = \mathsf{select}(\exists x.U, \mathsf{readable}(D_a,D))}{(\mathtt{SELECT}_b(\exists x.U).P,D) \rightsquigarrow_a (\prod_{s\in S} Ps, D)} \qquad \text{[I-UPDATE]}\ \frac{D' = \mathsf{update}(U,D,V)}{(\mathtt{UPDATE}_b(U,V).P,D) \rightsquigarrow_a (P,D')}$$

Table 5. Structural congruence

(Data)	$(D_1\|D_2)\|D_3 \equiv_D D_1\|(D_2\|D_3) \quad D_1\|D_2 \equiv_D D_2\|D_1 \quad D\|\emptyset \equiv_D D$
(Processes)	$(P_1\|P_2)\|P_3 \equiv_P P_1\|(P_2\|P_3) \qquad P_1\|P_2 \equiv_P P_2\|P_1 \qquad P\|\mathtt{0} \equiv_P P$
	$(P_1+P_2)+P_3 \equiv_P P_1+(P_2+P_3) \qquad P_1+P_2 \equiv_P P_2+P_1$
	$D_1 \equiv_D D_2 \Rightarrow \mathtt{WRITE}_u(D_1).P \equiv_P \mathtt{WRITE}_u(D_2).P$
(Networks)	$(N_1 \mid N_2) \mid N_3 \equiv N_1 \mid (N_2 \mid N_3) \qquad N_1 \mid N_2 \equiv N_2 \mid N_1$
	$D_1 \equiv_D D_2 \wedge P_1 \equiv_P P_2 \Rightarrow a[D_1 \parallel P_1] \equiv a[D_2 \parallel P_2]$

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 $\mathtt{READ}_b(U,\chi).P$ 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 $\mathtt{SELECT}_b(\exists x.U).P$ and D results in the parallel composition of those processes obtained by applying to the continuation P the substitutions that are the result of $\mathsf{select}(\exists x.U, D')$. The data argument D' of this function is the part of D that is readable to D_a.

— In [I-Write], the process $\mathtt{WRITE}_b(D').P$ 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.

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] formalise the property that the reduction 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 $\rightarrow$.

Table 6. Reduction relation

[R-USER] $\dfrac{(P,D_b) \rightsquigarrow_a (P',D) \quad a \neq b}{a[D_a \parallel P] \mid b[D_b \parallel Q] \to a[D_a \parallel P'] \mid b[D \parallel Q]}$	[R-SELF] $\dfrac{(P,D_a) \rightsquigarrow_a (P',D)}{a[D_a \parallel P] \to a[D \parallel P']}$
[R-STRUCT] $\dfrac{N_1 \equiv N_1' \quad N_1 \to N_2 \quad N_2 \equiv N_2'}{N_1' \to N_2'}$	[R-PARALLEL] $\dfrac{N_1 \to N_2}{N_1 \mid N \to N_2 \mid N}$

Well-behaved networks Suppose we are given a function $\mathscr{T}$ 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 formalises what we consider as well-behaved network, i.e. when we consider that privacy of data is completely protected.

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

(i) if $D_a \equiv (a_1, a_2, a_3)^U | D_a'$ and $\mathscr{T}(a_i) = U_i$, $i \in \{1,2,3\}$, 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 $\mathsf{readable}(D_a, D_a) = D_a$, and
(iii)if $N' \equiv b[D_b \parallel Q] \mid N''$ with $\mathscr{T}(b) = U_b$ then

- if $P \equiv \mathtt{DELETE}_b(U).R$ or $P \equiv \mathtt{UPDATE}_b(U,W).R$ then $\mathsf{readable}(D_a, D_b) = D_b$, and
- if $P \equiv \mathtt{WRITE}_b(D').R$ or $P \equiv \mathtt{UPDATE}_b(U,W).R$ and $(P, D_b) \rightsquigarrow_a (P, D_b')$ then $\mathsf{readable}(D_b', D_b') = D_b'$.

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

$$P_{\mathtt{Bob}} \equiv \mathtt{UPDATE}_{\mathtt{Alice}}(post_2, (\mathtt{Bob}, \mathtt{research}, \mathtt{types})) \text{ or}$$
$$P_{\mathtt{Bob}} \equiv \mathtt{WRITE}_{\mathtt{Alice}}(post_4^{(\mathtt{Bob},\mathtt{research},\mathtt{types})})$$

as ill-behaved, since after reduction `Alice`'s profile would become

$$D_{\mathtt{Alice}} \equiv \ldots | post_i^{(\mathtt{Bob},\mathtt{research},\mathtt{types})} | \ldots$$

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 $\preccurlyeq$ on queries by the following rules:

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

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

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

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

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

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

The proof is by induction on the derivation of $U \preccurlyeq V$.
If $U \preccurlyeq U \oplus V$ then $U\{^a/_x\} \preccurlyeq U\{^a/_x\} \oplus V\{^a/_x\}$, by definition of $\preccurlyeq$.
If $V \preccurlyeq U \oplus V$ then $V\{^a/_x\} \preccurlyeq U\{^a/_x\} \oplus V\{^a/_x\}$, by definition of $\preccurlyeq$.
If $U\{^c/_y\} \preccurlyeq \exists y.U$ then $(U\{^c/_y\})\{^a/_x\} \preccurlyeq (\exists y.U)\{^a/_x\}$ since it is equivalent to $(U\{^a/_x\})\{^c/_y\} \preccurlyeq \exists y.(U\{^a/_x\})$.

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

We shall prove that $\exists y.U \preccurlyeq \exists y.V$ implies $(\exists y.U)\{^a/_x\} \preccurlyeq (\exists y.V)\{^a/_x\}$. If the relation $\exists y.U \preccurlyeq \exists y.V$ holds, then $U\{^b/_y\} \preccurlyeq V\{^b/_y\}$ 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 $U \preccurlyeq V$ and $D \models U$ then $D \models V$.

Proof. The proof is by induction on the derivation of $U \preccurlyeq V$.
If $U \preccurlyeq U \oplus V$ and $D \models U$ then $D \models U \oplus V$ holds by [Q-ChoiceL].
If $V \preccurlyeq U \oplus V$ and $D \models V$ then $D \models U \oplus V$ holds by [Q-ChoiceR].
If $U\{^a/_x\} \preccurlyeq \exists x.U$ and $D \models U\{^a/_x\}$ then $D \models \exists x.U$ holds by [Q-Exists].

Let $\exists x.U \preccurlyeq \exists x.V$ 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 $U\{^a/_x\} \preccurlyeq V\{^a/_x\}$. By induction hypothesis, we have that $D \models V\{^a/_x\}$. Finally, by the definition of $\models$ we deduce $D \models \exists x.V$. □

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]
$\Gamma, x : Name(U) \vdash x : Name(U)$	$\Gamma, \chi : Data(U) \vdash \chi : Data(U)$

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 $\mathscr{T}$, then that name is well typed under any environment for the assigned query pattern. ([T-Name])

— 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 $U \preccurlyeq U_i$ 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 $(V \preccurlyeq U)$. ([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])

Table 7. Typing rules for names and data

$$\text{[T-NAME]}\ \frac{\mathscr{T}(a) = U}{\Gamma \vdash a : \mathit{Name}(U)} \qquad \text{[T-EMPTY DATA]}\ \Gamma \vdash \emptyset : \mathit{Data}(U)$$

$$\text{[T-TRIPLE]}\ \frac{\Gamma \vdash a_i : \mathit{Name}(U_i)^{\ (i\in\{1,2,3\})} \quad V \preccurlyeq U \preccurlyeq U_i^{\ (i\in\{1,2,3\})}}{\Gamma \vdash (a_1,a_2,a_3)^U : \mathit{Data}(V)} \qquad \text{[T-PARALLEL DATA]}\ \frac{\Gamma \vdash D_i : \mathit{Data}(U_i)^{\ (i\in\{1,2\})} \quad U \preccurlyeq U_i^{\ (i\in\{1,2\})}}{\Gamma \vdash D_1 | D_2 : \mathit{Data}(U)}$$

(superscript $i \in I$ means for every $i \in I$)

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-Parallell 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 $V \preccurlyeq U_u$ and the continuation is well typed for V. The update requires also the new privacy protection policy W to be 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])

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

Table 8. Typing rules for processes

$$\text{[T-Termination]}\quad \Gamma \vdash \mathtt{0} : \mathit{Process}(V)$$

$$\text{[T-Choice]}\quad \frac{\Gamma \vdash P_i : \mathit{Process}(V)\ ^{(i\in\{1,2\})}}{\Gamma \vdash P_1 + P_2 : \mathit{Process}(V)}$$

$$\text{[T-Parallel Process]}\quad \frac{\Gamma \vdash P_i : \mathit{Process}(V)\ ^{(i\in\{1,2\})}}{\Gamma \vdash P_1 | P_2 : \mathit{Process}(V)}$$

$$\text{[T-Process Variable]}\quad \Gamma \vdash X : \mathit{Process}(V)$$

$$\text{[T-Recursion]}\quad \frac{\Gamma \vdash P : \mathit{Process}(V)}{\Gamma \vdash \mathtt{rec}X.P : \mathit{Process}(V)}$$

$$\text{[T-Read]}\quad \frac{\Gamma, \chi : \mathit{Data}(W) \vdash P : \mathit{Process}(V)}{\Gamma \vdash \mathtt{READ}_u(U,\chi).P : \mathit{Process}(V)}$$

$$\text{[T-Write]}\quad \frac{\Gamma \vdash u : \mathit{Name}(U_u) \quad \Gamma \vdash \delta : \mathit{Data}(W) \quad \Gamma \vdash P : \mathit{Process}(V) \quad U_u \preccurlyeq W}{\Gamma \vdash \mathtt{WRITE}_u(\delta).P : \mathit{Process}(V)}$$

$$\text{[T-Select]}\quad \frac{\Gamma, x : \mathit{Name}(U_x) \vdash P : \mathit{Process}(V)}{\Gamma \vdash \mathtt{SELECT}_u(\exists x.U).P : \mathit{Process}(V)}$$

$$\text{[T-Update]}\quad \frac{\Gamma \vdash u : \mathit{Name}(U_u) \quad \Gamma \vdash P : \mathit{Process}(V) \quad V \preccurlyeq U_u \preccurlyeq W}{\Gamma \vdash \mathtt{UPDATE}_u(U,W).P : \mathit{Process}(V)}$$

$$\text{[T-Delete]}\quad \frac{\Gamma \vdash u : \mathit{Name}(U_u) \quad \Gamma \vdash P : \mathit{Process}(V) \quad V \preccurlyeq U_u}{\Gamma \vdash \mathtt{DELETE}_u(U).P : \mathit{Process}(V)}$$

Table 9. Typing rules for networks

$$\text{[T-User]}\quad \frac{\emptyset \vdash a : \mathit{Name}(U) \quad \emptyset \vdash D : \mathit{Data}(U) \quad \emptyset \vdash P : \mathit{Process}(U) \quad D \models U}{\emptyset \vdash a[D \parallel P] : \mathit{Network}}$$

$$\text{[T-Parallel Network]}\quad \frac{\emptyset \vdash N_i : \mathit{Network}\ ^{(i\in\{1,2\})}}{\emptyset \vdash N_1 \mid N_2 : \mathit{Network}}$$

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:

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

Lemma 4.1 (Generation lemma for variables).

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

Lemma 4.2 (Generation lemma for names and data).

1 $\Gamma \vdash a : \omega \Rightarrow \mathscr{T}(a) = U_a$ and $\omega = \mathit{Name}(U_a)$.
2 $\Gamma \vdash \emptyset : \omega \Rightarrow \omega = \mathit{Data}(U)$.
3 $\Gamma \vdash (a_1, a_2, a_3)^U : \omega \Rightarrow \omega = \mathit{Data}(V)$ and $\Gamma \vdash a_i : \mathit{Name}(U_i)$ and $V \preccurlyeq U \preccurlyeq U_i$ for every $i \in \{1,2,3\}$.
4 $\Gamma \vdash D_1 | D_2 : \omega \Rightarrow \omega = \mathit{Data}(U)$ and $\Gamma \vdash D_i : \mathit{Data}(U_i)$ and $U \preccurlyeq U_i$ for every $i \in \{1,2\}$.

Lemma 4.3 (Generation lemma for processes).

1 $\Gamma \vdash \mathtt{0} : \omega \Rightarrow \omega = \mathit{Process}(V)$.
2 $\Gamma \vdash P_1 + P_2 : \omega \Rightarrow \omega = \mathit{Process}(V)$ and $\Gamma \vdash P_1 : \mathit{Process}(V)$ and $\Gamma \vdash P_2 : \mathit{Process}(V)$.
3 $\Gamma \vdash P_1 | P_2 : \omega \Rightarrow \omega = \mathit{Process}(V)$ and $\Gamma \vdash P_1 : \mathit{Process}(V)$ and $\Gamma \vdash P_2 : \mathit{Process}(V)$.
4 $\Gamma \vdash X : \omega \Rightarrow \omega = \mathit{Process}(V)$.
5 $\Gamma \vdash \mathtt{rec}X.P : \omega \Rightarrow \omega = \mathit{Process}(V)$ and $\Gamma \vdash P : \mathit{Process}(V)$.
6 $\Gamma \vdash \mathtt{READ}_u(U,\chi).P : \omega \Rightarrow \omega = \mathit{Process}(V)$ and $\Gamma, \chi : \mathit{Data}(W) \vdash P : \mathit{Process}(V)$.

7 $\Gamma \vdash \texttt{WRITE}_u(\delta).P : \omega \Rightarrow \omega = \mathit{Process}(V)$ and $\Gamma \vdash u : \mathit{Name}(U_u)$ and $\Gamma \vdash \delta : \mathit{Data}(W)$ and $\Gamma \vdash P : \mathit{Process}(V)$ and $U_u \preccurlyeq W$.
8 $\Gamma \vdash \texttt{SELECT}_u(\exists x.U).P : \omega \Rightarrow \omega = \mathit{Process}(V)$ and $\Gamma, x : \mathit{Name}(U_x) \vdash P : \mathit{Process}(V)$.
9 $\Gamma \vdash \texttt{UPDATE}_u(U,W).P : \omega \Rightarrow \omega = \mathit{Process}(V)$ and $\Gamma \vdash u : \mathit{Name}(U_u)$ and $\Gamma \vdash P : \mathit{Process}(V)$ and $V \preccurlyeq U_u \preccurlyeq W$.
10 $\Gamma \vdash \texttt{DELETE}_u(U).P : \omega \Rightarrow \omega = \mathit{Process}(V)$ and $\Gamma \vdash u : \mathit{Name}(U_u)$ and $\Gamma \vdash P : \mathit{Process}(V)$ and $V \preccurlyeq U_u$.

Lemma 4.4 (Generation lemma for networks).

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

Lemma 4.5 (Substitution).

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

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

1 $D' = \mathsf{readable}(D_a, D)$,
2 $D' = \mathsf{read}(U, D)$,
3 $D' = \mathsf{delete}(U, D)$, or
4 $D' = \mathsf{update}(U, D, W)$ and $V \preccurlyeq W$,

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

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 $V \preccurlyeq U$ by Lemma 4.2.3. By definition of read, $D' = \emptyset$ or $D' = D$. In both cases, applying typing rules for data, we derive $\Gamma \vdash D' : \mathit{Data}(V)$.
Assume that $\Gamma \vdash D'_1 : \mathit{Data}(V)$ and $\Gamma \vdash D'_2 : \mathit{Data}(V)$ for $D'_1 = \mathsf{read}(U, D_1)$, and $D'_2 = \mathsf{read}(U, D_2)$. We shall prove $\Gamma \vdash D' : \mathit{Data}(V)$ for $D' = \mathsf{read}(U, D_1 | D_2) = \mathsf{read}(U, D_1) | \mathsf{read}(U, D_2)$, which folllows from the induction hypothesis and [T-Parallel Data].

□

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

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

Theorem 4.1 (Subject reduction). If $\emptyset \vdash N_1 : \mathit{Network}$ and $N_1 \to N_2$ then $\emptyset \vdash N_2 : \mathit{Network}$.

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

[R-User] Suppose $a \neq b$ and $\emptyset \vdash a[D_a \parallel P] \mid b[D_b \parallel Q] : \mathit{Network}$. We have to show $\emptyset \vdash a[D_a \parallel P'] \mid b[D \parallel Q] : \mathit{Network}$, which will follow from the typing rules if we prove $\emptyset \vdash$

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

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

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

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

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

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 $(\mathtt{rec}X.P_1, D_b) \rightsquigarrow_a (P', D)$ and $P = P_1\{^{\mathtt{rec}X.P_1}/_X\}$ 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 $N_1 \to N_2$ imply $\emptyset \vdash N_2 : Network$.
[R-Parallel] If $\emptyset \vdash N_1 \mid N : Network$ and $N_1 \mid N \to N_2 \mid N$, by Lemma 4.4.2 and the typing rule [T-Parallel Network], $\emptyset \vdash N_2 \mid N : Network$.
[R-Struct] If $N_1 \equiv N_1'$, $N_2 \equiv N_2'$, $N_1' \to N_2'$ 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 $N_1 \to^* N_2$ 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 $\mathsf{readable}(D_a, D_a) = D_a$.

Proof. The proof follows by induction on the structure of D_a. If $D_a = T^V$ then, by Lemma 4.2.3, $U_a \preccurlyeq V$. 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 $\mathsf{readable}(D_a, T^V) = T^V$. Assume that $D_a = D'|D''$ and $\mathsf{readable}(D_a, D') = D'$ and $\mathsf{readable}(D_a, D'') = D''$. Then $\mathsf{readable}(D_a, D) = \mathsf{readable}(D_a, D'|D'') = \mathsf{readable}(D_a, D')|\mathsf{readable}(D_a, D'') = D'|D'' = D_a$. □

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

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

(i) Assume that $D_a \equiv (a_1, a_2, a_3)^U | D'_a$ and $\mathscr{T}(a_i) = U_i$, $i \in \{1,2,3\}$. It holds by Lemma 4.4.1 that $\emptyset \models (a_1, a_2, a_3)^U | D'_a : \mathit{Data}(U_a)$. By Lemma 4.2.3, $U_a \preccurlyeq U \preccurlyeq U_i$ for every $i \in \{1,2,3\}$. We conclude by applying the Theorem 3.1.

(ii) $\mathsf{readable}(D_a, D_a) = D_a$ 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 ($P \equiv \mathtt{DELETE}_b(U).R$ or $P \equiv \mathtt{UPDATE}_b(U,W).R$) and $\mathscr{T}(a) = U_a$. By Lemma 4.4.2, $\emptyset \vdash P : \mathit{Process}(U_a)$, and by Lemma 4.3.9-10 $U_a \preccurlyeq U_b$. We proceed by induction on the structure of D_b. If $D_b = T^U$ then, by Lemma 4.2, $U_b \preccurlyeq U$. From $U_a \preccurlyeq U_b \preccurlyeq U$, by Theorem 3.1, since $D_a \models U_a$ it follows $D_a \models U$, and therefore also $\mathsf{readable}(D_a, T^U) = T^U$. Induction shows $\mathsf{readable}(D_a, D'_b | D''_b) = \mathsf{readable}(D_a, D'_b) \mid \mathsf{readable}(D_a, D''_b) = D'_b \mid D''_b = D_b$.

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

□

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 ((Heath and Bizer 2001; Heath 2011; Bizer *et al.* 2009; Bizer 2009; Berners-Lee 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 (Klyne and Caroll 2004; Brickley and Guha 2004), SPARQL (Prud'hommeaux and Seaborne 2008; Haris *et al.* 2011), 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 in (Horne and Sassone 2011-2) provide an abstract syntax in order to capture Linked Data structures and queries, which are then internalised in a process calculus. The abstract syntax of RDF of (Klyne and Caroll 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 2011-1; Sacco and Passant 2011-2; 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 2011-1; Sacco and Passant 2011-2), 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 (Sangiorgi and Walker 2001; Hennessy 2007). 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 *et al.* 2006; Abadi and Blanchet 2003) 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 guaranties 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 2011-1) a type and a subtype system verify consistency of URI names usage. The most related type assignment systems are (Dezani *et al.* 2008; Dezani *et al.* 2010), 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 and in general.

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. and Corin, R. and Fournet, C. (2006) Computational Secrecy by Typing for the pi Calculus. In Naoki Kobayashi, editor, *APLAS*, *Lecture Notes in Computer Science* 4279, 253–269. Springer.

Barendregt, H. P. (1992) Lambda Calculi with Types. In Samson Abramsky and Dov M. Gabbay and Thomas S. E. Maibaum *Handbook of Logic in Computer Science*, 117–309. Oxford University Press.

Bizer, C. (2009) The emerging Web of Linked Data, *IEEE Inteligent Systems.*, **24**, 87–92.

Bizer, C. and Heath, T. and Berners-Lee, T. Linked Data - the Story So Far (2009). *International Journal on Semantic Web and Information Systems*, **5** (3):1–22.

Berners-Lee, T. and Hendler, J. and Lassila, O. (2001) The Semantic Web, *Scientific Am.*, **284** (5), 35–43.

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. and Ghilezan, S. and Jakšić, S. and Pantovič 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. and Ghilezan, S. and Pantović, J. and Varacca, D. (2008) Security Types for Dynamic Web Data, *Theoretical Computer Science*, **402** (2-3):156–171.

Dezani-Ciancaglini, M. and Horne, R. and Sassone, V. (2012) Tracing where and who provenance in Linked Data: a calculus, *Theoretical Computer Science*, **464**:113–129.

Girard, J.-Y. (1987) Linear Logic, *Theoretical Computer Science*, **50**, 1–112.

Fournet, C. and Gordon, A. and Maffeis, S. (2007) A Type Discipline for Authorization in Distributed Systems, In *CSF*, 31–48. IEEE Computer Society.

Fournet, C. and Laneve, C. and 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 Martín Abadi and Luca de Alfaro, editors, *CONCUR*, *Lecture Notes in Computer Science* 3653, 202–216. Springer.

Haris, S. and 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. (2011) Programming Languages and Principles for Read-Write Linked Data. Ph.D. Thesis, *Electronics and Computer Science*, University of Southampton.

Horne, R. and Sassone, V. (2011) 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. (2011) A Verified Algebra for Linked Data. In Mohammad Reza Mousavi and António Ravara, editors, *FOCLASA*, *EPTCS* 58, 20–33.

Horne, R. and 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 Caroll, J. (2004) Resource Description Framework: Concepts and Abstract Syntax. W3C, MIT, Cambridge, MA, REC-rdf-concepts-20040210.

Maffeis, S. and Abadi, M. and Fournet, C. and Gordon, A. D. (2008) Code-Carrying Authorization. In Sushil Jajodia and Javier López, editors, *ESORICS*, *Lecture Notes in Computer Science* 5283, 563–579. Springer.

Merro, M. and Sangiorgi, D. (1998) On Asynchrony in Name-Passing Calculi. In Kim Guldstrand Larsen, Sven Skyum, and Glynn Winskel, editors, *ICALP*, *Lecture Notes in Computer Science* 1443, 856–867. Springer.

Milner, R. (1999) Communicationg and Mobile Systems: the π-Calculus. *Cambridge University Press*.

Milner, R. and 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. (2011) A Privacy Preference Ontology (ppo) for Linked Data. In *Proceedings of the Linked Data on the Web Workshop (LDOW2011)*.

Sacco, O. and Passant, A. (2011) A Privacy Preference Manager for the Social Semantic Web, *Proceedings of the second Workshop on Semantic Personalized Information Management: Retrieval and Recommendation 2011*, 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*, *Lecture Notes in Computer Science* 6069, 195–210. Springer.

Weitzner, D. (2007): Beyond Secrecy: New Privacy Protection Strategies for Open Information Spaces. IEEE Internet Computing, **11** (5), 94–96.

Westin, A. (1967): Privacy and Freedom. New York: Atheneum.