EasyChair Preprint

№ 7596

Type Checking Conditional Purpose-Based Privacy Policies in the π-Calculus

Georgios V. Pitsiladis

March 17, 2022

Type Checking Conditional Purpose-Based Privacy Policies in the π-calculus

Georgios V. Pitsiladis

National Technical University of Athens,
School of Applied Mathematical and Physical Science
Heroon Polytechniou 9, 15780 Zografou,Greece
g.v.pitsiladis@gmail.com

Abstract. This paper presents a formal system which builds upon the privacy framework defined in [8], able to statically infer the read, write, access, and disclose permissions needed by a given process of a variant of the π-calculus and then check if they are consistent with a given privacy policy. The syntax and semantics of the framework is extended to support granting permissions after checking for condition satisfaction. In addition, the proofs of the extended framework's safety are outlined.

Keywords: privacy, privacy policies, type systems, π-calculus

1 Introduction and Related Work

This paper presents some work carried out in the context of a diploma thesis [13] in the School of Applied Mathematical and Physical Science of the National Technical University of Athens, supervised by Prof. Petros Stefaneas.

Privacy is a concept that has been studied –in some form or another– since Aristotle, but has been acknowledged as a human right that needs protection over the last century [3]. Although its precise nature has not been clarified, it has been related to personal autonomy [17], the ability to maintain social relations [14] and human dignity [4]; several cases of breaches of privacy have been identified and classified [15]. In recent years, the advancement of technology has posed a great threat to privacy [1, 16]. As a result, privacy enforcement in the digital age needs relevant tools that protect user privacy and detect potential or actual breaches.

A long-term goal that follows from these concerns is to have sound and efficient formal systems that can be used as easily as possible to reason about privacy-related properties of information systems and enforce privacy requirements. A methodology towards such a system is described in [7] and some steps to the direction it indicates are taken in [8] and its extensions [6]; they define a framework (a privacy policy language, a variant of the π-calculus and a type system) that can statically infer the read, write, access, and disclose permissions needed by a given process of the π-calculus and then check if they are consistent with a given privacy policy. This paper extends those results with the notion of

condition, which captures the intuition that some permission may be granted to entities after checking that some condition is fulfilled (eg a person has a specific age).

Much work has been done lately on the subject of formal methods on privacy, apart from the line of work of [8]. [5] defines a framework which uses type checking and a custom variant of the π-calculus, in order to reason about data on the Web, particularly the data expressed with standards such as RDF. [11] defines a rather expressive formal system based on epistemic logic, tailored to reasoning about the privacy policies of social networks.

Moreover, since privacy policies share some common properties with access control policies, there have been attempts to extend access control policies, in order to be usefully applicable for privacy purposes. Such an extension is P-RBAC [9, 10, 2], where the notion of condition employed here stems from. It also provides reasons for splitting Groups of [8] in separate notions of Users and Roles.

2 The Privacy Policy Language

As in [6], privacy policies are defined over the sets of Basic Types $\mathcal{D}$, which represent the (sensitive) data of users, Purposes $\mathcal{PU}$, Users $\mathcal{U}$ and Roles $\mathcal{R}$. We also use the set of Groups $\mathcal{G} \stackrel{\text{def}}{=} \mathcal{U} \cup \mathcal{R}$.

Notation. Variables of $\mathcal{D}$ will be denoted by t, of $\mathcal{PU}$ by u, of $\mathcal{U}$ by U, $\mathcal{R}$ by R, of $\mathcal{G}$ by G.

We will need a language for expressing conditions. The syntax of the one presented here is identical to LC_0 of Core P-RBAC [9], although the semantics differ in some respects to be discussed later on. The condition language LC_0 presupposes a set of context variables $\mathcal{X}$. Each $X \in \mathcal{X}$ has a finite domain of possible values D_X, which is equipped with the operators $=, \neq$. Conditions (denoted by c) are expressed as follows:

1. If $X \in \mathcal{X}$, $op \in \{=, \neq\}$, $u \in D_X$, then $(X\ op\ v)$ is an atomic condition. $(X = v)$ is satisfied by v, while $(X \neq v)$ is satisfied by $D_X \setminus v$.
2. If c_1, c_2 are conditions, $c_1 \wedge c_2$ is a condition. $c_1 \wedge c_2$ is satisfied when both c_1 and c_2 are satisfied.

If $X_1, \ldots, X_n$ (ordered by their names), are the context variables that appear in c, then the set $D(c) = \prod_{i=1}^{n} D_{X_i}$ is called the domain of c. $D(c)$ is naturally split in two equivalence classes, $D^+(c) = \{v \in D(c) \mid c \text{ is satisfied by } v\}$ (positive condition domain) and $D^-(c) = \{u \in D \mid c \text{ is not satisfied by } v\}$ (negative condition domain). Hereafter, each condition will be semantically identified with its positive domain[1].

[1] It is trivial to check that the relation $c_1 \sim c_2 \stackrel{\text{def}}{\Leftrightarrow} D^+(c_1) = D^+(c_2)$ is an equivalence relation, so each condition will hereafter represent its equivalence class under $\sim$.

The set of privacy-related Permissions `Perm` (its elements will be denoted by the letter p) that can be granted to entities contains two kinds of elements: basic or unconditional permissions and conditional permissions. Basic permissions are the same as in [6]: `read`, `write`, `access` and `disc` G; they give entities the right to read, write, access and disclose data within group G. Conditional permissions are formed by an unconditional permission, followed by the keyword "if" and a condition in LC_0.

Hierarchies, denoted by the letter H, supply partial orderings of groups and assign them purposes. Their syntax is as follows:

$$H ::= \epsilon \quad | \quad U : \tilde{u}[\epsilon] \quad | \quad R : \tilde{u}[\tilde{H}] \ ,$$

where $\tilde{H}$ is a non-empty hierarchy set, ϵ is the empty hierarchy and we require that $R \notin \text{groups}(\tilde{H})$. In $G : \tilde{u}[\tilde{H}]$, purposes $\tilde{u}$ are also implicitly assigned to all $G \in \text{groups}(\tilde{H})$. The function $\text{groups}(\tilde{H})$ collects all the groups that appear in a set of hierarchies. We also define the functions $\text{root}(H)$, which returns the root of H as a set, and $u \to H$, which adds the purpose u to the root of H.

Notation. We write $G[\ ]$ for $G : \emptyset[\epsilon]$, $G : \tilde{u}$ for $G : \tilde{u}[\epsilon]$ and $G[\tilde{H}]$ for $G : \emptyset[\tilde{H}]$.

Privacy policies have the form

$$\mathcal{P} ::= t \gg H, \pi \quad | \quad \mathcal{P}; \mathcal{P} \ ,$$

with the operator ; being associative and commutative and π being a permission assignment function: $\pi \ : \ \mathcal{PU} \times \mathcal{G} \to \wp(\texttt{Perm})$. We require that the same basic type does not appear twice in a policy.

We now turn to define some useful ordering relations between elements of our policies. We will use the functions $\text{vars}(c)$, which collects all context variables appearing in c, and $c\restriction_{\tilde{X}}$, which strips c of every atomic condition whose context variable is not contained in $\tilde{X}$ (we require that $\text{vars}(c) \cap \tilde{X} \neq \emptyset$).

Definition 1.

1. $c_1 \leq c_2 \overset{def}{\Leftrightarrow} \text{vars}(c_2) \subseteq \text{vars}(c_1) \ \wedge \ D^+(c_1\restriction_{\text{vars}(c_2)}) \subseteq D^+(c_2)$
2. $p_1 \leq p_2 \overset{def}{\Leftrightarrow} p_1 = p_2 \ \vee \ ((p_1 = p \text{ if } c) \ \wedge \ (p_2 = p))$
   $\vee \ ((p_1 = p \text{ if } c_1) \ \wedge \ (p_2 = p \text{ if } c_2) \ \wedge \ (c_1 \leq c_2))$
3. $\tilde{p_1} \lesssim \tilde{p_2} \overset{def}{\Leftrightarrow} \forall p \in \tilde{p_1} \exists p' \in \tilde{p_2} : p \leq p'$

The above definition captures the intuition that we can make a condition stricter by diminishing its positive domain or by making it depend on more context variables, we can make a permission stricter by adding a condition and we can make a permission set stricter by removing a permission or by making a permission in it stricter. A direct consequence of the definition is that $\leq$ is a partial order on conditions and on permissions and that $\lesssim$ is a preorder on permission sets.

Note that in case $\tilde{p_1}$ and $\tilde{p_2}$ do not contain conditional permissions, we obtain $\tilde{p_1} \lesssim \tilde{p_2} \Leftrightarrow \tilde{p_1} \subseteq \tilde{p_2}$. This, along with our definitions of privacy policy language

elements, implies that our privacy policy language is backwards compatible with the one of [6].

Although LC_0 comes directly from Core P-RBAC, there is a fundamental difference in its semantics: in our framework, by virtue of $\exists$ in the definition of $\lesssim$, permission p on data t will be granted to user or role G for purpose u under policy $t \gg H, \pi(u, G) = \{p \text{ if } c_1, p \text{ if } c_2\}$ if either one of c_1 and c_2 holds (as if there was a disjunction between them); on the contrary, the corresponding construct of Core P-RBAC will grant the permission only if both c_1 and c_2 hold (as if there was a conjunction between them) [9]. Our framework's behaviour is similar to that of Normalized Permission Assignment Sets in Conditional P-RBAC [10]. As noted in [9], this can lead to unintended consequences in uninformed use of the framework (imagine a case where $c_1 = (\text{UserAge} \neq \text{under18})$ and $c_2 = (\text{UserrConsent} = \text{Yes})$; using c_2, one may bypass the age check).

Example 1. Suppose we wish to examine the privacy policy of an online sales company. The company is divided in three departments: administration, orders and marketing. The orders department is divided in purchase and shipping departments. Of course, the company needs to communicate with its clients for sales, but it also communicates with other companies, in order to exchange data about customers for marketing purposes; the second case happens only for customers over 18 years old who have given their consent. Moreover, administration periodically collects and statistically analyses sales and some customer data. We will focus on a user named Bob and, in particular, on his address. The special thing about Bob is that he has given his friend Alice permission to use his address for her purchases.

We model the above assumptions as follows:

The context variables we need are Bob's age and his consent with regard to the exchange of his data with third parties for marketing purposes.

$$\mathcal{X} = \{\text{Bob.Age}, \text{Bob.Consent}\}, \quad D_{\text{Consent}} = \{\text{Yes}, \text{No}\},$$
$$D_{\text{Age}} = \{0-12, 13-17, 18-30, 31-50, 51-70, \text{over70}\} \ .$$

Bob's sensitive data is his address, so $\mathcal{D} = \{\text{Bob.Address}\} \cup \mathcal{X}$.

The set of users is $\mathcal{U} = \{\text{Alice}, \text{Bob}\}$, while for the set of roles we have the departments of the company, the role of client, a role that unifies company and clients and a role that unifies third parties and the marketing department.

$$\mathcal{R} = \{\, \text{Company}, \text{OrderDept}, \text{AdminDept}, \text{PurchaseDept}, \text{ShippingDept},$$
$$\text{MarketingDept}, \text{ThirdParty}, \text{Comp\&Clients}, \text{Clients} \,\}$$

The intuitive relation of groups in $\mathcal{G} = \mathcal{U} \cup \mathcal{R}$ is illustrated in figure 1.

As for purposes, we have $\mathcal{PU} = \{\text{analysis}, \text{purchase}, \text{marketing}\}$.

In order to specify a privacy policy, we will need permission assignment functions, one for each element of $\mathcal{D}$. Bob's age can be accessed and read by the purchase department to check if he is old enough to make purchases; it can be accessed and read by the marketing department to check if his data may be

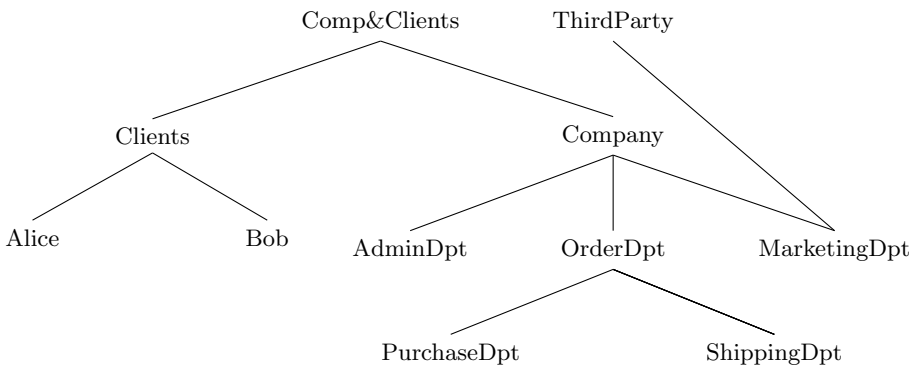

Fig. 1. The groups of $\mathcal{G}$ and their relation. A group connected with another group above it is supposed to inherit its permissions.

forwarded to third parties for marketing purposes; it can finally be read and accessed by administration for statistical analysis. Of course, Bob has complete permissions on his own data. Same holds for Bob's consent regarding the exchange of his data with third parties for marketing purposes, except that there is no reason for the purchase department to have any permission on it.

$$\pi_{\text{B.Age}}(u,G) = \begin{cases} \{\texttt{access},\texttt{read}\} & \text{if } (u,G) = (\text{analysis},\text{AdminDpt}) \\ \{\texttt{access},\texttt{read}\} & \text{if } (u,G) = (\text{purchase},\text{PurchaseDpt}) \\ \{\texttt{access},\texttt{read}\} & \text{if } (u,G) = (\text{marketing},\text{MarketingDpt}) \\ \left\{\begin{matrix}\texttt{read},\texttt{write},\texttt{access},\\ \texttt{disc}\ \text{Comp\&Clients}\end{matrix}\right\} & \text{if } G = \text{Bob} \end{cases}$$

$$\pi_{\text{B.Consent}}(u,G) = \begin{cases} \{\texttt{access},\texttt{read}\} & \text{if } (u,G) = (\text{marketing},\text{MarketingDpt}) \\ \{\texttt{access},\texttt{read}\} & \text{if } (u,G) = (\text{analysis},\text{AdminDpt}) \\ \left\{\begin{matrix}\texttt{read},\texttt{write},\texttt{access},\\ \texttt{disc}\ \text{Comp\&Clients}\end{matrix}\right\} & \text{if } G = \text{Bob} \end{cases}$$

During purchases, the purchase department can access Bob's address and disclose it within the order department, provided that Bob is over 18 years old. The shipping department can receive and read the address in order to send him the products he ordered, but it is not obliged to check his age. The marketing department can access Bob's address and disclose it to third parties for the purpose of marketing, provided that Bob is over 18 years old and has given his consent to do so. Alice can read, access and disclose Bob's address within the company during her purchases. Observe that administration enjoys no permission on Bob's address. We write c_0 for (B.Age $\neq 0-17$) and c_1 for (B.Consent $=$ Yes).

$$\pi_{\text{B.Address}}(u, G) = \begin{cases} \left\{\begin{array}{l}\texttt{access if } c_0,\\ \texttt{disc OderDpt if } c_0\end{array}\right\} & \text{if } u = \text{purchase},\\ & \quad G = \text{PurchaseDpt}\\ \{\texttt{access}, \texttt{read}\} & \text{if } u = \text{purchase},\\ & \quad G = \text{ShippingDpt}\\ \left\{\begin{array}{l}\texttt{access if } c_0,\\ \texttt{disc ThirdParty if } c_1 \wedge c_0\end{array}\right\} & \text{if } u = \text{marketing},\\ & \quad G = \text{MarketingDpt}\\ \left\{\begin{array}{l}\texttt{read}, \texttt{write}, \texttt{access},\\ \texttt{disc Comp\&Clients}\end{array}\right\} & \text{if } G = \text{Bob}\\ \{\texttt{read}, \texttt{access}, \texttt{disc Comp\&Clients}\} & \text{if } u = \text{purchase},\\ & \quad G = \text{Alice} \end{cases}$$

Finally, we need to specify a hierarchy; the tree of figure 1 with Comp&Clients as its root will suffice for that. We also specify the purposes towards which each group can act.

$$\begin{aligned} H = &\text{Comp\&Clients}\,[\\ &\quad \text{Clients} : \{\text{purchase}\}\,[\text{Alice}[\,], \text{Bob}[\,]],\\ &\quad \text{Company}\,[\\ &\qquad \text{AdminDept} : \{\text{analysis}\}\,,\\ &\qquad \text{OrderDept} : \{\text{purchase}\}\,[\text{PurchaseDept}[\,], \text{ShippingDept}[\,]],\\ &\qquad \text{MarketingDept} : \{\text{marketing}\}]] \end{aligned}$$

Thus, we obtain the following privacy policy:

$$\begin{aligned} \mathcal{P}_{\text{Bob}} = \;& \text{B.Age} \gg H, \pi_{\text{B.Age}}\ ;\ \text{B.Consent} \gg H, \pi_{\text{B.Consent}}\\ & ;\ \text{B.Address} \gg H, \pi_{\text{B.Address}} \end{aligned}$$

3 The π-calculus

We will use the variant of π-calculus described in [6], with an extension to support condition checking.

We assume there exist three basic sets of entities: an infinitely countable set of Names $\mathcal{N}$, ranged over by x, y, z, a, b, a set of basic types, a set of purposes and a set of groups. We identify the sets of basic types, purposes and groups with those defined in the previous section. We include each context variable domain as a subset of $\mathcal{N}$. We also define a set of Types $\mathcal{T}$, ranged over by T; it contains all basic types, all context variables and each element of the form $G[T]$. Each name is mapped to a type (explicitly or implicitly), something that restricts usage of the channel represented by the name: in particular, a name of type $G[T]$ can be

used by processes that "belong" to group G (in a sense that will be clear later) to exchange messages of type T.

Programmes of π-calculus are defined in two levels: processes (denoted by P) and systems (denoted by S):

$$\begin{aligned} P ::= {} & \mathbf{0} \quad | \quad P1 \mid P2 \quad | \quad !P \quad | \quad (\nu\, x : T)P \quad | \quad x(y : T).P \\ & | \quad \overline{x}\,\langle y\rangle .P \quad | \quad [x = y](P1; P2) \quad | \quad [\![x = y]\!]P \quad | \quad [\![x \neq y]\!]P \\ S ::= {} & \mathbf{0} \quad | \quad S1 \mid S2 \quad | \quad (\nu\, x : T)S \quad | \quad (\nu\, R)S \quad | \quad (\nu\, G)P\,\langle u\rangle \end{aligned}$$

The processes $\mathbf{0}$, $P1 \mid P2$, $!P$, $(\nu\, x : T)P$, $x(y : T).P$ and $\overline{x}\,\langle y\rangle .P$ are standard constructs of π-calculus; we require that y in $x(y : T).P$ and x in $(\nu\, x : T)P$, $(\nu\, x : T)S$ are not context variable values, because we think of context variable values as global constants of our systems. The process $[x = y](P1; P2)$ checks if the names x and y, with y being a context variable value, are equal; if so, it proceeds as P_1, else it proceeds as P_2. The processes $[\![x = y]\!]P$, $[\![x \neq y]\!]P$, with y being a context variable value, assume there has been a condition check and continue as P. These two constructs are intended to be used only by the framework itself, acting as a "tag" that a condition check –which otherwise would have been forgotten– has taken place. The systems $\mathbf{0}$, $(\nu\, x : T)S$, $S_1 \mid S_2$ act like the respective processes, albeit at the level of systems. The system $(\nu\, G)P\,\langle u\rangle$ binds G and declares that a process is running on behalf of group (user or system) G for the purpose u. Finally, the system $(\nu\, R)S$ binds R and declares that the system is running on behalf of an entity that belongs to role R.

Notation. For brevity, $[x = y]P \stackrel{\text{def}}{\equiv} [x = y](P; \mathbf{0})$ and $[x \neq y]P \stackrel{\text{def}}{\equiv} [x = y](\mathbf{0}; P)$.

Notation. We write $\text{fn}(P)$ and $\text{fn}(S)$ for the names free in a process P and a system S, $\text{bn}(P)$ and $\text{bn}(S)$ for the names bound in a process P and a system S, $\text{fg}(T)$, $\text{fg}(P)$ and $\text{fg}(S)$ for the groups bound in a type T, a process P and a system S. Note that free groups of processes occur only within types and context variable values can never be bound.

As in [6], we use labelled transition semantics. We first define labels (denoted by l) as $l \ ::= \ \tau \ | \ x(y : T) \ | \ \overline{x}\,\langle y\rangle \ | \ (\nu\, y : T)\bar{x}\,\langle y\rangle$, where τ corresponds to the internal action, $x(y : T)$ to input, $\overline{x}\,\langle y\rangle$ to output and $(\nu\, y : T)\overline{x}\,\langle y\rangle$ to restricted output. The functions $\text{fn}(l)$ and $\text{bn}(l)$ collect the free and bound names of label l; all names in labels are free, except for the bound y in labels of the form $(\nu\, y : T)\overline{x}\,\langle y\rangle$. We also define the relation dual between labels:

$$\text{dual}(l_1, l_2) \stackrel{\text{def}}{\Leftrightarrow} \{l_1, l_2\} = \{x(y : T), \bar{x}\,\langle y\rangle\} \vee \{l_1, l_2\} = \{x(y : T), (\nu\, y : T)\bar{x}\,\langle y\rangle\} \quad .$$

The labelled transition semantics is presented in figure 2, using the meta-notation $F \ ::= \ P \ | \ S$. The symbol $(\nu\, \text{bn}(l_1) \cup \text{bn}(l_2))$ in (**??**) is to be read as follows (ϵ being the empty string):

$$(\nu\, \text{bn}(l_1) \cup \text{bn}(l_2)) \equiv \begin{cases} \epsilon & \text{if } \{l_1, l_2\} = \{x(y : T), \bar{x}\,\langle y\rangle\} \\ (\nu\, y : T) & \text{if } \{l_1, l_2\} = \{x(y : T), (\nu\, y : T)\bar{x}\,\langle y\rangle\} \end{cases} \quad .$$

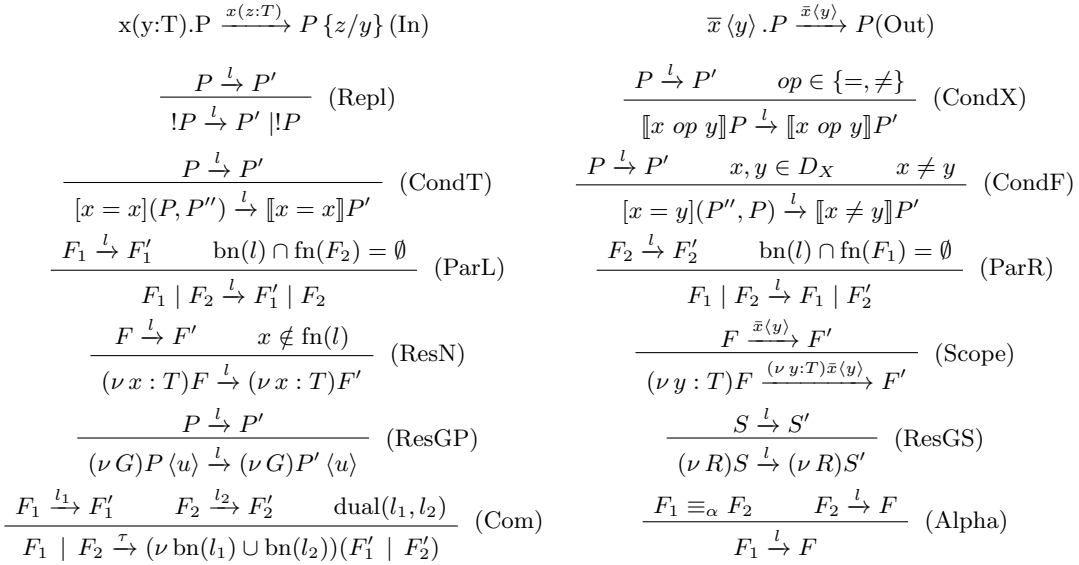

Fig. 2. The rules of labelled transition semantics.

Example 2. We build on example 1 by giving examples of π-calculus systems that may run in the company's environment.

1. During a purchase, system S_{Alice} runs in Alice's browser; it discloses Bob's address to the company. At the same time, systems $S_{\text{PurchaseDpt}}$, running on behalf of the purchase department, and $S_{\text{ShippingDpt}}$, running on behalf of the shipping department, are waiting to receive the address, process it accordingly and then restart: $S_{\text{PurchaseDpt}}$ checks if Bob is an adult and, if so, forwards the address; $S_{\text{ShippingDpt}}$ receives and reads it in order to finalise the purchase and proceed to shipping. T_1 stands for Comp&Clients[B.Address].

$$S_{\text{Alice}} = (\nu\,\text{Alice})\overline{\text{sendaddr}}\,\langle\text{addr}\rangle\,.\mathbf{0}\,\langle\text{purchase}\rangle$$
$$S_{\text{PurchaseDpt}} = (\nu\,\text{PurchaseDpt})!\text{[bobage} \neq 0 - 17]\text{sendaddr(addr} : T_1).\ \overline{\text{order}}\,\langle\text{addr}\rangle\,.\mathbf{0}\,\langle\text{purchase}\rangle$$
$$S_{\text{ShippingDpt}} = (\nu\,\text{ShippingDpt})!\text{order(addr} : T_1).\text{addr(a} : \text{B.Address}).\mathbf{0}\,\langle\text{purchase}\rangle$$
$$S_1 = (\nu\,\text{Comp\&Clients})(\nu\,\text{sendaddr} : \text{Comp\&Clients}[T_1])(\ (\nu\,\text{Clients})S_{\text{Alice}} \mid ((\nu\,\text{Company})(\nu\,\text{OderDpt})\ (\nu\,\text{order} : \text{OrderDpt}[T_1])(S_{\text{PurchaseDpt}} \mid S_{\text{ShippingDpt}})))$$

2. The system S_2 runs on behalf of the marketing department for marketing purposes. It checks if Bob is an adult, reads the value of his consent and forwards his address to a third party. Note that it does not stop if Bob has declared that he refuses to share his data with third parties –we expect type

checking to detect this. T_2 stands for ThirdParty[ThirdParty[B.Address]].

$$\begin{aligned} S_2 = (\nu\,\text{ThirdParty})(\nu\,\text{Comp\&Clients})(\nu\,\text{sendtotp} : T_2)(\nu\,\text{Company}) \\ (\nu\,\text{MarketingDpt})[\text{bobage} \neq 0 - 17]\text{readc}(\text{cons} : \text{B.Consent}). \\ (\nu\,\text{addr} : \text{ThirdParty[B.Address]})\overline{\text{sendtotp}}\,\langle\text{addr}\rangle\,.\mathbf{0}\,\langle\text{analysis}\rangle \end{aligned}$$

4 The Type System

The type system aims to infer the permissions needed by a process or system of the π-calculus. Of course, when inferred, these permissions can –and will– be compared with privacy policies for compliance. What follows is an extension of the type system of [6] with rules for condition checking; type checking is performed via Γ-Environments and infers Δ-Environments for processes and Θ-Interfaces for systems.

Γ-Environments (denoted by Γ) store information concerning the groups "known" to a system and the mapping of names to types in processes and systems. Their syntax is as follows:

$$\Gamma \;::=\; \emptyset \quad | \quad x : T \quad | \quad G \quad | \quad \Gamma_1 \cdot \Gamma_2 \;\;,$$

with the operator $\cdot$ being associative and commutative; we require that each name and each group appears in a Γ-Environment at most once. We define the function $\text{dom}(\Gamma)$, which collects all the names and groups appearing in Γ and we write $\Gamma \subseteq \Gamma'$ when $\text{dom}(\Gamma) \subseteq \text{dom}(\Gamma')$ and Γ, Γ' map their common names to the same types.

Δ-Environments (denoted by Δ) store information concerning the permissions exercised on basic types in a process. Their syntax is as follows:

$$\Delta \;::=\; \emptyset \quad | \quad t : \tilde{p} \quad | \quad \Delta_1 \cdot \Delta_2 \;\;,$$

with the operator $\cdot$ being associative and commutative; we require that each basic type appears in a Δ-Environment at most once.

Θ-Interfaces (denoted by Θ) store information concerning the basic types present in a system; for each basic type, they store tuples consisting of a) the groups some entity belongs to, along with a purpose towards which it acts, b) a permission set. Their syntax is as follows:

$$\Theta \;::=\; \emptyset \quad | \quad t \gg \left\langle H^{\downarrow}, \tilde{p} \right\rangle \quad | \quad \Theta_1 \,;\, \Theta_2 \;\;,$$

with the operator ; being associative and commutative. $H^{\downarrow}$ is a special type of linear hierarchy called Interface Hierarchy: $H^{\downarrow} ::= G[u] \;\mid\; G[H^{\downarrow}]$. We define the functions $\text{dom}(\Theta)$, which collects all the basic types appearing in Θ, and $\text{purpose}(H^{\downarrow})$, which returns the purpose in $H^{\downarrow}$.

We need to define some auxiliary functions:

Definition 2.

1. $c \oplus p = \begin{cases} p' \text{ if } c \wedge c' & \text{if } p = p' \text{ if } c' \\ p \text{ if } c & \text{otherwise} \end{cases}$, $\quad c \oplus \tilde{p} = \bigcup_{p \in \tilde{p}} \{c \oplus p\}$

2. $c \oplus \Delta = \begin{cases} (c \oplus \Delta_1) \cdot (c \oplus \Delta_2) & \text{if } \Delta = \Delta_1 \cdot \Delta_2 \\ t : (c \oplus \tilde{p}) & \text{if } \Delta = t : \tilde{p} \\ \emptyset & \text{otherwise} \end{cases}$

3. $\Delta_r(T) = \begin{cases} t : \texttt{read} & \text{if } T = t \\ t : \texttt{access} & \text{if } T = G[t] \\ \emptyset & \text{otherwise} \end{cases}$, $\quad \Delta_w(T) = \begin{cases} t : \texttt{write} & \text{if } T = G[t] \\ t : \texttt{disc}\, G & \text{if } T = G[G'[t]] \\ \emptyset & \text{otherwise} \end{cases}$

4. $\Delta_1 \uplus \Delta_2 = \{t : \tilde{p_1} \cup \tilde{p_2} \mid t : \tilde{p_1} \in \Delta_1, t : \tilde{p_2} \in \Delta_2\}$, *where we suppose for brevity that* $t : \emptyset \in \Delta$ *if there is no other* $\tilde{p}$ *such that* $t : \tilde{p} \in \Delta$

5. $G[u] \odot \Delta = \begin{cases} (G[u] \odot \Delta_1); (G[u] \odot \Delta_2) & \text{if } \Delta = \Delta_1 \cdot \Delta_2 \\ t \gg \langle G[u], \tilde{p} \rangle & \text{if } \Delta = t : \tilde{p} \\ \emptyset & \text{otherwise} \end{cases}$

$$G \odot \Theta = \begin{cases} (G \odot \Theta_1); (G \odot \Theta_2) & \text{if } \Theta = \Theta_1; \Theta_2 \\ t \gg \left\langle G[H^{\downarrow}], \tilde{p} \right\rangle & \text{if } \Theta = t \gg \left\langle H^{\downarrow}, \tilde{p} \right\rangle \\ \emptyset & \text{otherwise} \end{cases}$$

We use three kinds of typing judgements: $\Gamma \vdash x \triangleright T$ states that x has type T under Γ; $\Gamma \vdash P \triangleright \Delta$ states that P is well-typed under Γ and produces Δ; $\Gamma \vdash S \triangleright \Theta$ states that S is well-typed under Γ and produces Θ. Type checking is performed according to the rules in figure 3.

$$\frac{\mathrm{fg}(T) \subseteq \mathrm{dom}(\Gamma)}{\Gamma \cdot x : T \vdash x \triangleright T} \text{ (Name)} \qquad \frac{\Gamma \vdash P \triangleright \Delta}{\Gamma \vdash !P \triangleright \Delta} \text{ (Rep)}$$

$$\frac{\Gamma \cdot y : T \vdash P \triangleright \Delta \qquad \Gamma \vdash x \triangleright G[T]}{\Gamma \vdash x(y : T).P \triangleright \Delta \uplus \Delta_{\mathrm{r}}(T)} \text{ (In)} \qquad \frac{\Gamma \vdash P \triangleright \Delta \qquad \Gamma \vdash x \triangleright G[T] \qquad \Gamma \vdash y \triangleright T}{\Gamma \vdash \bar{x}\langle y \rangle .P \triangleright \Delta \uplus \Delta_{\mathrm{w}}(G[T])} \text{ (Out)}$$

$$\frac{\Gamma \vdash P_1 \triangleright \Delta_1 \qquad \Gamma \vdash P_2 \triangleright \Delta_2}{\Gamma \vdash P_1 \mid P_2 \triangleright \Delta_1 \uplus \Delta_2} \text{ (ParP)} \qquad \frac{\Gamma \vdash S_1 \triangleright \Theta_1 \qquad \Gamma \vdash S_2 \triangleright \Theta_2}{\Gamma \vdash S_1 \mid S_2 \triangleright \Theta_1; \Theta_2} \text{ (ParS)}$$

$$\frac{\Gamma \cdot x : T \vdash P \triangleright \Delta}{\Gamma \vdash (\nu\, x : T)P \triangleright \Delta} \text{ (ResNP)} \qquad \frac{\Gamma \cdot x : T \vdash S \triangleright \Theta}{\Gamma \vdash (\nu\, x : T)S \triangleright \Theta} \text{ (ResNS)}$$

$$\frac{\Gamma \cdot G \vdash P \triangleright \Delta}{\Gamma \vdash (\nu\, G)P\langle u \rangle \triangleright G[u] \odot \Delta} \text{ (ResGP)} \qquad \frac{\Gamma \cdot R \vdash S \triangleright \Theta}{\Gamma \vdash (\nu\, R)S \triangleright R \odot \Theta} \text{ (ResGS)}$$

$$\frac{\Gamma \vdash P \triangleright \Delta \qquad \Gamma \vdash x \triangleright X \qquad \Gamma \vdash y \triangleright X \qquad op \in \{=, \neq\}}{\Gamma \vdash [\![x \; op \; y]\!]P \triangleright (X \; op \; y) \oplus \Delta} \text{ (CondA)}$$

$$\frac{\Gamma \vdash [\![x = y]\!]P_1 \triangleright \Delta_1 \qquad \Gamma \vdash [\![x \neq y]\!]P_2 \triangleright \Delta_2}{\Gamma \vdash [x = y](P_1; P_2) \triangleright \Delta_1 \uplus \Delta_2} \text{ (CondB)}$$

$$\Gamma \vdash \mathbf{0} \triangleright \emptyset \text{ (Nil)}$$

Fig. 3. The rules of the type system.

Rules (CondA) and (CondB) can be combined to yield rule (Cond):

$$\frac{\Gamma \vdash P_1 \triangleright \Delta_1 \qquad \Gamma \vdash P_2 \triangleright \Delta_2 \qquad \Gamma \vdash x \triangleright X \qquad \Gamma \vdash y \triangleright X}{\Gamma \vdash [x = y](P_1; P_2) \triangleright ((X \;=\; y) \oplus \Delta_1) \uplus ((X \;\neq\; y) \oplus \Delta_2)} \qquad \text{(Cond)}$$

Now that we can infer the permissions needed by a system of the π-calculus, it is natural to wish to test them for compliance with a privacy policy. We say that a Θ-Interface Θ satisfies a privacy policy $\mathcal{P}$ (notation: $\mathcal{P} \vDash \Theta$), if

$$\forall t \gg \left\langle H^{\downarrow}, \tilde{p} \right\rangle \in \Theta \; \exists t \gg H, \pi \in \mathcal{P} : \tilde{p} \lesssim \mathrm{perms}_{\pi}(H, \mathrm{groups}(H^{\downarrow}), \mathrm{purpose}(H^{\downarrow})) \;\; ,$$

where

$$\mathrm{perms}_{\pi}(H, \tilde{G}, u) = \begin{cases} \pi(u, G) \cup \left(\bigcup\limits_{\substack{H_i \in \tilde{H} \\ \mathrm{root}(H_i) \subseteq \tilde{G}}} \mathrm{perms}_{\pi}(u \to H_i, \mathrm{groups}(H_i) \cap \tilde{G}, u) \right) \\ \qquad\qquad \text{if } H = G : \tilde{u}[\tilde{H}], G \in \tilde{G}, u \in \tilde{u} \\ \bigcup\limits_{\substack{H_i \in \tilde{H} \\ \mathrm{root}(H_i) \subseteq \tilde{G}}} \mathrm{perms}_{\pi}(H_i, \mathrm{groups}(H_i) \cap \tilde{G}, u) \\ \qquad\qquad \text{if } H = G : \tilde{u}[\tilde{H}], G \in \tilde{G}, u \notin \tilde{u} \\ \emptyset \qquad\qquad \text{if } \tilde{G} = \emptyset \text{ or } H = \epsilon \\ \bot \qquad\qquad \text{otherwise} \end{cases}$$

The function $\mathrm{perms}_{\pi}(H, \tilde{G}, u)$ finds all the paths stemming from the root of the hierarchy H that contain groups from $\tilde{G}$. If the group at some point in the path is permitted to act towards purpose u, the permissions of $\pi(u, G)$ are added to the resulting permission set; of course, in this case, all following points in its path(s) –being lower in the hierarchy– are associated with u.

We say that the system S is compliant with, or conforms to, or respects, a privacy policy $\mathcal{P}$ (notation: $\mathcal{P} \vdash S$) if there are Γ and Θ such that $\Gamma \vdash S \triangleright \Theta$ and $\mathcal{P} \vDash \Theta$.

Example 3. We build on example 2 by giving the results of type checking for the systems presented there.

1. Due to space constraints, we will only show a small portion of the typing judgements: the ones concerning $S_{\mathrm{PurchaseDpt}}$. We use the Γ-Environments

$$\begin{aligned} \Gamma_1 \quad &= 0 - 17 : \mathrm{B.Age} \cdot \mathrm{bobage} : \mathrm{B.Age} \cdot \mathrm{address} : T_1 \\ \Gamma_{\mathrm{PD}} &= \Gamma_1 \cdot \mathrm{sendaddr} : \mathrm{Comp\&Clients}[T_1] \cdot \mathrm{Comp\&Clients} \cdot \mathrm{Company} \\ &\qquad \cdot \mathrm{OrderDpt} \cdot \mathrm{order} : \mathrm{OrderDpt}[T_1] \cdot \mathrm{PurchaseDpt} \\ \Gamma'_{\mathrm{PD}} &= \Gamma_{\mathrm{PD}} \cdot \mathrm{PurchaseDpt} \;\; . \end{aligned}$$

   and obtain

$$\Gamma'_{\mathrm{PD}} \cdot \mathrm{addr} : T_1 \vdash \mathbf{0} \triangleright \emptyset \qquad \text{(Nil)}$$

$$\Gamma'_{\mathrm{PD}} \cdot \mathrm{addr} : T_1 \vdash \overline{\mathrm{order}}\,\langle \mathrm{addr}\rangle\,.\mathbf{0} \triangleright \mathrm{B.Address} : \{\mathtt{disc}\ \mathrm{OrderDpt}\} \qquad (\mathbf{??})$$

$$\Gamma'_{\mathrm{PD}} \vdash \mathrm{sendaddr}(\mathrm{addr} : T_1).\overline{\mathrm{order}}\,\langle \mathrm{addr}\rangle\,.\mathbf{0} \triangleright \mathrm{B.Address} : \{\mathtt{disc}\ \mathrm{OrderDpt}, \mathtt{access}\} \qquad (\mathbf{??})$$

$$\Gamma'_{\mathrm{PD}} \vdash [\mathrm{bobage} \neq 0 - 17]\mathrm{sendaddr}(\mathrm{addr} : T_1).\overline{\mathrm{order}}\,\langle \mathrm{addr}\rangle\,.\mathbf{0} \triangleright \mathrm{B.Address} : \{\mathtt{disc}\ \mathrm{OrderDpt}\ \mathtt{if}\ c_0, \mathtt{access}\ \mathtt{if}\ c_0\} \qquad (\mathrm{Cond})$$

$$\Gamma'_{\mathrm{PD}} \vdash\, !\,[\mathrm{bobage} \neq 0 - 17]\mathrm{sendaddr}(\mathrm{addr} : T_1).\overline{\mathrm{order}}\,\langle \mathrm{addr}\rangle\,.\mathbf{0} \triangleright \mathrm{B.Address} : \{\mathtt{disc}\ \mathrm{OrderDpt}\ \mathtt{if}\ c_0, \mathtt{access}\ \mathtt{if}\ c_0\} \qquad (\mathbf{??})$$

$$\Gamma_{\mathrm{PD}} \vdash S_{\mathrm{PurchaseDpt}} \triangleright \mathrm{B.Address} \gg \langle\ \mathrm{PuchaseDpt}[\mathrm{purchase}], \{\mathtt{disc}\ \mathrm{OrderDpt}\ \mathtt{if}\ c_0, \mathtt{access}\ \mathtt{if}\ c_0\}\ \rangle \qquad (\mathbf{??})$$

Using the type checking rules, we can continue this process and infer that $\Gamma_1 \vdash S_1 \triangleright \Theta_1$, with

$$\begin{aligned}\Theta_1 = \ & \mathrm{B.Address} \gg \langle \mathrm{Comp\&Clients}[\mathrm{Clients}[\mathrm{Alice}[\mathrm{purchase}]]], \{\mathtt{disc}\ \mathrm{Comp\&Clients}\}\rangle \\ & ;\mathrm{B.Address} \gg \langle \mathrm{Comp\&Clients}[\mathrm{Company}[\mathrm{OrderDpt}[\mathrm{PurchaseDpt}[\mathrm{purchase}]]]], \\ & \qquad \{\mathtt{access}\ \mathtt{if}\ c_0, \mathtt{disc}\ \mathrm{OrderDpt}\ \mathtt{if}\ c_0\}\rangle \\ & ;\mathrm{B.Address} \gg \langle \mathrm{Comp\&Clients}[\mathrm{Company}[\mathrm{OrderDpt}[\mathrm{ShippingDpt}[\mathrm{purchase}]]]], \\ & \qquad \{\mathtt{access}, \mathtt{read}\}\rangle\ .\end{aligned}$$

Then, it is easy to see that $\mathcal{P}_{\mathrm{Bob}} \vDash \Theta_1$, which means that S_1 respects Bob's privacy.

2. In a similar fashion to above, we can infer that $\Gamma_2 \vdash S_2 \triangleright \Theta_2$, with

$$\begin{aligned}\Gamma_2 = \ & \mathrm{age} : \mathrm{B.Age} \cdot 0 - 17 : \mathrm{B.Age} \cdot \mathrm{readc} : \mathrm{Comp\&Clients}[\mathrm{B.Consent}], \\ \Theta_2 = \ & \mathrm{B.Address} \gg \langle \mathrm{Comp\&Clients}[\mathrm{ThirdParty}[\mathrm{Company}[\mathrm{MarketingDpt}[\\ & \qquad \mathrm{marketing}]]]], \{\mathtt{disc}\ \mathrm{ThirdParty}\ \mathtt{if}\ c_0\}\rangle \\ & ;\ \mathrm{B.Consent} \gg \langle \mathrm{Comp\&Clients}[\mathrm{ThirdParty}[\mathrm{Company}[\mathrm{MarketingDpt}[\\ & \qquad \mathrm{marketing}]]]], \{\mathtt{read}\ \mathtt{if}\ c_0\}\rangle\end{aligned}$$

We have

$$\mathrm{perms}_{\pi_{\mathrm{B.Address}}}(H, \{\mathrm{Comp\&Clients}, \mathrm{ThirdParty}, \mathrm{Company}, \mathrm{MarketingDpt}\}, \mathrm{marketing}) = \{\mathtt{access}\ \mathtt{if}\ c_0, \mathtt{disc}\ \mathrm{ThirdParty}\ \mathtt{if}\ c_0 \wedge c_1\}$$

and $\{\mathtt{disc}\ \mathrm{ThirdParty}\ \mathtt{if}\ c_0\} \not\lesssim \{\mathtt{access}\ \mathtt{if}\ c_0, \mathtt{disc}\ \mathrm{ThirdParty}\ \mathtt{if}\ c_0 \wedge c_1\}$, so we conclude that S_2 does not conform to $\mathcal{P}_{\mathrm{Bob}}$, as expected.

5 Safety Proofs

By safety of our typing system we mean that when a system of the π-calculus is judged to respect a privacy policy, it does not actually try to exercise some

permission not allowed by the policy; we also mean that the type system and the π-calculus semantics are compatible, ie a system that has been proved compliant by type checking will continue to be so after an arbitrary number of transitions. We prove our framework's safety as done in [8]. We only state propositions and give hints on their proofs; the proofs themselves can be found in [13].

Definition 3.

1. $\Delta_1 \lesssim \Delta_2 \overset{def}{\Leftrightarrow} \forall t : \tilde{p} \in \Delta_1 \exists t : \tilde{p}' \in \Delta_2 : \tilde{p} \lesssim \tilde{p}'$
2. $\Theta_1 \lesssim \Theta_2 \overset{def}{\Leftrightarrow} \mathrm{dom}(\Theta_1) = \mathrm{dom}(\Theta_2)$
$$\wedge \ \forall t \gg \left\langle H^{\downarrow}, \tilde{p} \right\rangle \in \Theta_1 \exists t \gg \left\langle H^{\downarrow}, \tilde{p}' \right\rangle \in \Theta_2 : \tilde{p} \lesssim \tilde{p}'$$

It is trivial to check that the relations defined above are preorders and that the auxiliary functions of definition 2 respect the relations of definitions 1 and 3. Proposition 1 is a direct consequence of the definitions of $\lesssim$ and $\vDash$:

Proposition 1. $\mathcal{P} \vDash \Theta_1 \ \wedge \ \Theta_2 \lesssim \Theta_1 \ \Rightarrow \ \mathcal{P} \vDash \Theta_2$

Notation. For the rest of this section, we use the meta-notation $F \ ::= \ P \mid S$ and $\Xi \ ::= \ \Delta \mid \Theta$.

Lemma 1. *If F is well-typed under Γ, then* $\mathrm{fn}(F) \cup \mathrm{fg}(F) \subseteq \mathrm{dom}(\Gamma)$.

Proof. By inspection of the typing rules.

Lemma 2. *If x is not a context variable value,*

$$\Gamma \cdot x : T \vdash F \triangleright \Xi \ \Rightarrow \ \Gamma \cdot y : T \vdash F\{y/x\} \triangleright \Xi$$

Proof. The case of $x, y \notin \mathrm{bn}(F)$ can be proved by induction on the structure of F. The general case follows from the next corollary.

Corollary 1. *Type checking respects a-equivalence, ie*

$$\Gamma \vdash F_1 \triangleright \Xi \ \wedge \ F_1 \equiv_\alpha F_2 \ \Rightarrow \ \Gamma \vdash F_2 \triangleright \Xi$$

Proof. By induction on the structure of F, using the case of $x, y \notin \mathrm{bn}(F)$ above.

The following two lemmas can be proved by induction on the structure of the process/system and by inspection of rule (**??**).

Lemma 3 (Strengthening). $\Gamma \cdot z : T \vdash x \triangleright T \ \wedge z \neq x \ \Rightarrow \ \Gamma \vdash x \triangleright T$
and also $\Gamma \cdot z : T \vdash F \triangleright \Xi \ \wedge z \notin \mathrm{fn}(F) \Rightarrow \ \Gamma \vdash F \triangleright \Xi$

Lemma 4 (Weakening). $\Gamma \vdash x \triangleright T \ \wedge \ y \notin \mathrm{dom}(\Gamma) \ \Rightarrow \Gamma \cdot y : T \vdash x \triangleright T$
and also $\Gamma \vdash F \triangleright \Xi \ \wedge \ y \notin \mathrm{dom}(\Gamma) \ \Rightarrow \Gamma \cdot y : T \vdash F \triangleright \Xi$

Theorem 1.

$$\Gamma \vdash F_1 \triangleright \Xi \ \wedge \ F_1 \xrightarrow{l} F_2 \ \Rightarrow \ \Gamma' \vdash F_2 \triangleright \Xi' \ \wedge \ \Xi' \lesssim \Xi \ \wedge \ \Gamma \subseteq \Gamma'$$

Proof. By induction on transition rules, using the lemmas above and corollary 1; observe that $\mathrm{dom}(\Gamma') \setminus \mathrm{dom}(\Gamma) \subseteq \mathrm{fn}(F_2) \setminus \mathrm{fn}(F_1)$.

The fact that a system is deemed compliant to a policy by the type system does not by itself imply something about its safety. We need to formalise what it means to perform a prohibited action. This is achieved by the following definition:

Definition 4.

1. *Consider a privacy policy $\mathcal{P}$, a Γ-Environment Γ and a system*

   $$S \equiv (\nu\, G_1)(\nu\, \boldsymbol{x}_1 : \boldsymbol{T}_1)\,(\cdots((\nu\, G_n)(\nu\, \boldsymbol{x}_n : \boldsymbol{T}_n)[\![\boldsymbol{y}\ \boldsymbol{op}\ \boldsymbol{z}]\!]P\,\langle u\rangle \mid P'\,\langle u'\rangle)\cdots \mid S_1) \quad ,$$

   with the notation $(\nu\, \boldsymbol{x} : \boldsymbol{T})$ standing for zero or finite $(\nu\, x_i : T_i)$ and $[\![\boldsymbol{y}\ \boldsymbol{op}\ \boldsymbol{z}]\!]$ standing for zero or finite $[\![y_j\ op_j\ z_j]\!]$.
   S will be called an error with respect to $\mathcal{P}$ and Γ (notation: $\mathrm{error}_{\mathcal{P},\Gamma}(S)$), if one of the following holds, where $\tilde{p} = \bigcup_{i=1}^{n} \pi(u, G_i)$ and $\Gamma' = \Gamma \cdot \boldsymbol{G} \cdot \boldsymbol{x}_i : \boldsymbol{T}_i$:
   (a) S is not well-typed under Γ, ie there is no Θ such that $\Gamma \vdash S \triangleright \Theta$.
   (b) There are $t \in \mathcal{D}$ and context variables $\boldsymbol{Y}$ such that $\mathcal{P} = t \gg H, \pi; \mathcal{P}'$, $\Gamma' \vdash \boldsymbol{y} \triangleright \boldsymbol{Y}$, $P = x(y : t).P''$ and $\{\mathtt{read}\ \mathrm{if}\ (\boldsymbol{Y}\ \boldsymbol{op}\ \boldsymbol{z})\} \not\lesssim \tilde{p}$.
   (c) There are $t \in \mathcal{D}$ and context variables $\boldsymbol{Y}$ such that $\mathcal{P} = t \gg H, \pi; \mathcal{P}'$, $\Gamma' \vdash \boldsymbol{y} \triangleright \boldsymbol{Y}$, $\Gamma \vdash y \triangleright t, P = x\,\langle y\rangle\,.P''$ and $\{\mathtt{write}\ \mathrm{if}\ (\boldsymbol{Y}\ \boldsymbol{op}\ \boldsymbol{z})\} \not\lesssim \tilde{p}$.
   (d) There are $t \in \mathcal{D}$ and context variables $\boldsymbol{Y}$ such that $\mathcal{P} = t \gg H, \pi; \mathcal{P}'$, $\Gamma' \vdash \boldsymbol{y} \triangleright \boldsymbol{Y}$, $P = x(y : G[t]).P''$ and $\{\mathtt{access}\ \mathrm{if}\ (\boldsymbol{Y}\ \boldsymbol{op}\ \boldsymbol{z})\} \not\lesssim \tilde{p}$.
   (e) There are $t \in \mathcal{D}$ and context variables $\boldsymbol{Y}$ such that $\mathcal{P} = t \gg H, \pi; \mathcal{P}'$, $\Gamma' \vdash \boldsymbol{y} \triangleright \boldsymbol{Y}$, $\Gamma \vdash x \triangleright G[G'[t]], P = x\,\langle y\rangle\,.P''$, $\{\mathtt{disc}\, G\ \mathrm{if}\ (\boldsymbol{Y}\ \boldsymbol{op}\ \boldsymbol{z})\} \not\lesssim \tilde{p}$.
2. *Consider a privacy policy $\mathcal{P}$ and a system S. S will be called an error with respect to $\mathcal{P}$ (notation: $\mathrm{error}_{\mathcal{P}}(S)$), if for all Γ-Environments Γ we have $\mathrm{error}_{\mathcal{P},\Gamma}(S)$.*

Proposition 2. $\mathrm{error}_{\mathcal{P},\Gamma}(S) \ \wedge\ \Gamma \vdash S \triangleright \Theta \ \Rightarrow\ \mathcal{P} \nvDash \Theta$

Proof. By inspecting the cases (1b), (1c), (1d) (1e) of the definition of $\mathrm{error}_{\mathcal{P},\Gamma}(S)$.

Theorem 2. $\mathcal{P} \vdash S \ \wedge\ S \xrightarrow{l}{}^{*} S' \ \Rightarrow\ \neg\mathrm{error}_{\mathcal{P}}(S')$

Proof. Follows from theorem 1 and propositions 1 and 2.

6 Future Work

We have managed to extend the framework of [8] with the notion of condition, keeping it backwards compatible with its version of [6]. In the future, we are planning to work in the direction of obtaining a provingly correct efficient executable implementation of our framework; this can aid in i) applying it to usecases at greater scale, which can assist in spotting difficulties in their widespread use, and ii) automating the proofs of its safety as far as possible, which can assist in easily extending it while still possessing such proofs. Some first steps in this

direction are presented in [12]. Moreover, we want to make policies more realistic by adding obligations and giving a hierarchical structure to purposes and data, in the spirit of Universal P-RBAC [10].

As noted in [7], the computational model of the π-calculus may need to be replaced by something more agile that can model cases where data is internally processed (eg aggregated) by entities or when entities need to change their privacy properties dynamically.

References

1. Berman, J., Mulligan, D.: Privacy in the digital age: Work in progress. Nova L. Rev. 23, 551 (1998)
2. Byun, J.W., Bertino, E., Li, N.: Purpose based access control of complex data for privacy protection. In: Proceedings of the tenth ACM symposium on Access control models and technologies. pp. 102–110. ACM (2005)
3. DeCew, J.: Privacy. In: Zalta, E.N. (ed.) The Stanford Encyclopedia of Philosophy. Spring 2015 edn. (2015)
4. Floridi, L.: On human dignity as a foundation for the right to privacy. Philosophy & Technology pp. 1–6 (2016)
5. Jakšic, S., Pantovic, J., Ghilezan, S.: Linked data privacy. Mathematical Structures in Computer Science pp. 1–21 (2015)
6. Kokkinofta, E., Philippou, A.: Type checking purpose-based privacy policies in the π-calculus. In: International Workshop on Web Services and Formal Methods. pp. 122–142. Springer (2014)
7. Kouzapas, D., Philippou, A.: A methodology for a formal approach in privacy, to appear
8. Kouzapas, D., Philippou, A.: Type checking privacy policies in the π-calculus. In: International Conference on Formal Techniques for Distributed Objects, Components, and Systems. pp. 181–195. Springer (2015)
9. Ni, Q., Bertino, E., Lobo, J., Brodie, C., Karat, C.M., Karat, J., Trombeta, A.: Privacy-aware role-based access control. ACM Transactions on Information and System Security (TISSEC) 13(3), 24 (2010)
10. Ni, Q., Lin, D., Bertino, E., Lobo, J.: Conditional privacy-aware role based access control. In: European Symposium on Research in Computer Security. pp. 72–89. Springer (2007)
11. Pardo, R., Schneider, G.: A formal privacy policy framework for social networks. In: International Conference on Software Engineering and Formal Methods. pp. 378–392. Springer (2014)
12. Pitsiladis, G.V.: Implementing type checking of π-calculus processes for privacy in maude, submitted
13. Pitsiladis, G.V.: Type checking privacy policies in the π-calculus and its executable implementation in Maude. Diploma thesis, National Technical University of Athens (2016), supervised by Stefaneas P., in Greek
14. Rachels, J.: Why privacy is important. Philosophy & Public Affairs pp. 323–333 (1975)
15. Solove, D.J.: A taxonomy of privacy. University of Pennsylvania law review pp. 477–564 (2006)
16. Tene, O.: Privacy: The new generations. International Data Privacy Law 1(1), 15–27 (2011)
17. Warren, S.D., Brandeis, L.D.: The right to privacy. Harvard law review pp. 193–220 (1890)