A Process Calculus Integrating Data Protection by Design and Default

Chinmayi Prabhu Baramashetru
Department of Informatics
University of Oslo, Oslo, Norway
cpbarama@ifi.uio.no

S. Lizeth Tapia Tarifa
Department of Informatics
University of Oslo, Oslo, Norway
sltarifa@ifi.uio.no

Olaf Owe
Department of Informatics
University of Oslo, Oslo, Norway
olaf@ifi.uio.no

***Abstract*—The European General Data Protection Regulation (GDPR) focuses on giving more control to personal data owners. It imposes several restrictions on organizations to obtain valid consent from users for processing their personal data. In this paper, we propose a formal consent framework satisfying core GDPR provisions and try to solve the ambiguity around GDPR compliance. We use a variant of Pi-calculus to model privacy-aware systems incorporating conditions of consent, storage limitation, and purpose-based processing. We provide operational semantics that respect user consent through the data processing life cycle. The theory is explained with an example, and we also present a prototypic implementation in Maude, yielding an environment for program simulation to verify consent specifications within the programs.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Software systems often collect and analyze user data to tailor experiences to individual preferences. Extensive interaction of users with software applications has resulted in frequent and broader dissemination of information beyond ways users can comprehend. While personalized services can enhance user satisfaction, it often raises concerns about privacy and data protection. To control this rampant flow of information online and offline data protection regulations such as GDPR came into practice. The GDPR applies to all organizations that handle personal data about EU residents, regardless of the location, and several conditions have to be satisfied to be GDPR-compliant. However, extracting legal requirements and mapping them into software functionality is complex and error-prone. For instance, one of the core provisions of GDPR is to ensure that the processing of personal data is according to user's valid will or, in GDPR terms, also referred to as user's consent, this consent as per the GDPR must be freely given, specific, informed, and unambiguous, but in practice; one might say that this is seldom the case; often, consent is uninformed and presented in unclear forms, which leads users to accept the conditions of the provider without a choice. On the other hand, failure to provide proof of valid consent by the organization may result in noncompliance with GDPR. This higher restriction on consent and data processing requirements has raised a need for the organization to revisit the design and development of its software system to guarantee GDPR compliance. In this current work, we formally model a distributed system with the notion of consent and the other core provisions of GDPR. We choose pi-calculus as the starting point, which provides a formal and expressive framework to study the dynamic behaviour of distributed systems. **CP:** revise the contributions bit more here Intuitively, we associate processes with entities, the data flow within the scope of the entity process must comply with the concerned user's consent. We capture the consent in the form of user policies, which capture the notion of purposes, entities, actions, and respective retention time enforcing purpose based processing of data. We monitor personal data flow within various entities and check against the consent available at the state. To achieve this, we provide operational semantics in the form of user interaction rules and system rules.

II. GDPR Requirements

The GDPR establishes a unified framework for data protection across the EU and it is currently stretched over 11 chapters consisting 99 articles.However, in this paper, we mainly focus on a few articles that have the potential to be designed within our calculus. We discuss such articles in this section from two perspectives: the data controllers(i.e., organizations that process personal data) and the data subjects(i.e., users whose personal data is being collected and processed).

The Perspective of Data Subjects: The GDPR primarily aims to enhance transparency and communication, granting data subjects greater control over their data. Art. [12-23] outlines rights that service providers must enable for data subjects while processing their personal data. Art. 15 (Right of Access) emphasizes that users play a vital role in the processing of personal data. Users are entitled to be informed about specific details regarding the handling of their personal data, including 1) the purposes of processing, 2) recipients of their data, 3) storage duration, 4) access to any personal data concerning them, the ability to obtain a copy, rectify inaccuracies, and object to processing. As a result, these metadata details should be dynamically accessible within the system at all times. For instance, if a user denies processing for certain purposes, data access within the system must adhere to these restrictions throughout the data life cycle. Prominently, under Article 17 (Right to be forgotten) for any reason, if a user requests data erasure or if specific consent is no longer valid, the data controller is obligated to promptly delete all data, including backups, snapshots, and replicas, without undue delay.

The Data Controller Perspective: Art. [24-43], specifies the measures to be taken by controllers and processors when handling user's personal data. Art. 24 (Responsibility of the Controller) states that the controller is solely responsible for having technical measures in place to handle the collected data by the regulation. Art. 25 (Data Protection by Design and Default) specifies that all information systems should be designed, configured, and processed with data protection as a fundamental goal. Art. 6.1 (Conditions for Consent) specifies that the data subject's consent to handle personal data for one or more purposes is essential for lawful processing. Art. 7 specifies that consent needs to be informed and freely given before processing personal data. The terms and conditions presented to receive consent from data subjects should be clear, intelligible, and understandable and users can modify their consent at any time, and data controllers should facilitate this choice. Art. 5 Sec. 1(b) (Purpose Limitation) imposes restrictions on systems for collecting a vast amount of data for ambiguous and broadly classified purposes. It states that data controllers are compelled to use personal data only for specific, well-defined purposes and cannot process further for alternative purposes. Additionally, Sec. 1(e) of Art. 5 mentions that personal data shall only be stored if necessary and imposes storage limitations based on purposes.

III. GDPR Challenges and Solutions

In this section, we discuss the main technical challenges in implementing the requirements discussed in Sec. II. In particular, we focus on the challenges with the potential to be addressed via language design principles. In contrast to well-explored problems for incorporating GDPR rules within a distributed system, the challenges discussed in this section are less explored by the research community [8].

a) ***Challenge 1: Contextual awareness of consent*** *:* As we know, consent is one of the lawful bases for the GDPR, but managing consent within its contextual framework presents a complex challenge. Consent typically aligns with specific purposes, services, and entities, which can shift over time. Factors such as an organization's adoption of new data handling methods, changes in the intended purpose, or users modifying their consent status, including withdrawals, consent expiration after the retention period, new business partners, and service collaborators can all alter the contextual landscape of consent. Existing systems lack such integration in their system design and development, thus failing to keep track of internal and external contextual changes in consent over time.

b) ***Challenge 2: Data processed as personal data.****:* This can happen when non-personal data can be associated with an identifier or when such data is combined with other pieces of data to be associated with an individual. For example, entity E_1 can handle non-personal data D, which in the process of data handling is transformed into identifiable personal data D'. However, E_1 does not have authorization from the data owner to process such personal data (data leak), creating a violation of the users' consent preference.

c) ***Challenge 3: Personal data with multiple owners.****:* Although the GDPR doesn't explicitly address processing data owned by multiple subjects, it's crucial to consider data handling in such scenarios. As an example, let us consider a loan application of combined personal data D of two data subjects, Alice and Bob. If Alice allows the handling of her personal data for a set of purposes P_1 and Bob allows the handling of his personal data for a set of purposes P_2, it is unclear how to proceed with the handling of D.

d) ***Challenge 4: Unclear terminology for personal data handing****:* As discussed in Sec. II, GDPR terminology, concerning data handling (e.g., collect, store, use, delete, transfer), lacks specific definitions. It is unclear how services should directly enforce requirements associated with such terms.

e) ***Challenge 5: Ambiguity in data deletion and degree of compliance****:* GDPR mandates that no personal data can be stored indefinitely within the system, but it does not specify when the controller should delete the data, i.e., within seconds, hours, days, or even months. For a company like Google, which collects user data at scale upon deletion request, it takes up to 6 months to delete data from all its subsystems. The notion of deletion is unclear and indirectly affects the degree of compliance achieved. If the system weak-compliant or strong-compliant is affected by the efficiency of deletion. Additionally, most storage systems or databases do not consider consent or metadata such as purposes, access, and entities during deletion it follows key-value storage, but these GDPR-related metadata helps efficiently delete personal data based on consent retraction or expired retention.

IV. Design Principles of the Calculus

In this section, we motivate the main privacy-related aspects that we will incorporate in our calculus, introduce our language syntax inspired by π calculus formally, followed by labeled transition semantics that formalises our privacy-specific data handling features. We refer to calculus as CCAL.
Data controllers (DC) and *data processors (DP)* can be understood as *entities*, which are identifiable organizations, and organizational units that have or can handle data. CCAL will include the declaration of a set of entities, these entities are domain-specific. Each entity represents a session with a process running within its scope. The system consists of parallel compositions of many of these entity sessions denoting the compositionality of organization. In object-oriented programming, this could be envisioned as interfaces or classes abstracting the entities. Entity can be DC or DP and we can model how information flows between them by checking against the consent settings provided by the users.
Data subjects (DS) can be understood as *users* whose personal data is being handled by entities. As noted in Sec. II, users should be able to express consent on their personal data handling and CCAL will include unique names that represent users so that we can check how a service behaves when interacting with the users. In CCAL, personal data will be associated with specific IDs, transforming them into private

values, these IDs are user names, also it can handle non-personal data freely and restrict the handling of personal data. In particular, the language can handle personal data with multiple owners (where each owner has different consent).We declare the users within the calculus and can update their consent anytime in the configuration.

Purposes can be understood as the reasons why personal data is being handled. CCAL will include the declaration of a set of purposes as names. We enforce purpose based processing throughout the calculus, for e.g., personal data will be associated with specific purposes in terms of private runtime tags and purposes are captured in consent via user policies as well. All the tagged data at runtime is handled based on the purposes attached to them against the global consent.
Consent can be understood as an *informed agreement* that users give to allow the handling of their personal data. CCAL will include the addition and removal of consent via *privacy policies*. Explicitly stating what actions entities can perform for certain purposes. Actions will be stated using the vocabulary in the GDPR (e.g., collect, transfer, use, store, delete). The language used to capture privacy policies is a simplified version of the language presented in [2]. Consent also captures retention time for the data handling, and since it is a runtime element, users may modify it anytime during the lifetime of a service. To capture such flexibility, CCAL will include operations that users can perform to add/remove consent and add/update retention at any time, mimicking the enforcement of Article [7] in the GDPR. See Sec. II.
Lawfulness of processing of personal data is a fundamental principle of the GDPR, as noted in Sec. II. All the elements described above will help CCAL capture how personal data should be handled on a basis of user's consent. The novelty of the calculus resides in the incorporation of privacy properties, including the notion of private data, the inclusion of entity encapsulation, user interaction, purpose-based processing, consent, and data retention (more details in further sections).

A. Syntax

Our calculus is a conservative extension of the π calculus. Hence we assume the reader is familiar with the basic π calculus mechanisms. We want to model systems that follow a privacy-by-design paradigm. Systems are composed of communicating agents which can be treated as DC/DP. Each entity includes running processes, privacy handling capabilities, and the capability of communicating private data with other entities and /or other systems. Interactions between these entities are modeled following the consent available from the DS. We show interactions between processes as channel-based message passing. The syntax of the calculus is given in figure 1. We extend the standard π calculus with the following,(although names, expressions, variables, and constant symbols have similarities we keep them separate). We separate normal channels, localized channels, users, entities, purposes and database channels, to provide detailed formal modeling of the consent. We assume the following countable sets: We refer to a set of channels $\mathcal{E}$ ranged over by $e, f, g \dots$ as entity channels; set $\mathcal{P}$ for purpose channels ranged over by $p_1, p_2, p_3 \dots$; $\mathcal{U}$ as user channels ranged over by $u_1, u_2, u_3 \dots$; and $\mathcal{K}$ as database channels ranged over by $k, l, m \dots$. As mentioned in the introduction, channels can be uniquely associated with an entity/user. The set of channels $\mathcal{C}$ is formed by coupling a channel name with an entity/user name and it is ranged over by $a^e, a^f, a^g, a^{u_1}, a^{u_2}, \dots$, Hence the set over channels are not just the plain names but a coupled channel. For e.g., an entity name doc is associated with channel a resulting in a coupled channel a^{doc}. Similarly, user $alice$ is associated with channel b making a^{alice}. We use the notion of localized channels based on [**?**] [**?**] by tagging output action with the entity/user name where the exchange is allowed to happen.

Example 1: Consider an interaction between a doctor entity and a nurse entity, when *doctor* wants to send patient data through channel a, since our system is distributed these channels are not bound to any process and are non-located channels and passing the private values on such public channels is quite an abstraction of reality. Hence, we tag the capability of the channel during output action with entity *nurse* to specify where the exchange message is supposed to take place. This is inspired by the idea of location type where a resource is available to an entity at a particular location.

$$doc \; [\![\; a^{nurse}!(patient\_data).P]\!] \;\; || \;\; nurse \; [\![a?(x).Q]\!]$$

Our identifiers are ranged over by $v, w \dots$ comprised of normal channels, database channels, variables, and coupled compound channels. Whereas our terms, ranged over by $r, s \dots$, comprise everything in identifiers and data values. A system consists of a parallel composition of entity and user sessions that can share channels and pass messages. An entity session $e[\![\; P \;]\!]$ represents an entity e executing a session with process P running inside the session. Note that, input channels cannot be used as variables and not coupled with entity names. This is how we can localize the input channels to the entity they belong to. On the other hand output actions must mention the entity names. For example, $e[\![a^f!(...) \; .P \;]\!] \; || \; f[\![a?(...) \; .Q \;]\!]$ defines that an entity e is trying to communicate with a private channel that belongs to entity f whereas if you see the entity f has input channel a which does not mention any capability reason being if the channel is localized within the entity f that means the channel belongs to entity f there is developed location types based calculus proposed in [5] and used in [**?**] for an RBAC schema-based calculus. We follow a similar approach. Processes 0, $P|Q$, $*P$, $(\nu)P$, $a?(r)$, $w!(s)$ are the standard π calculus constructs depicting the inactive process, parallel composition, process replication, channel restriction, input action and output action. We will omit the trailing nil process in the rest of the paper.

a) User interaction process expressions: Now we explain other privacy-specific process expressions within the calculus: the term $addCon(\rho)$ defines a process adding consent in the form of a privacy policy ρ to the respective user consent specification, whereas the term $rmvCon(\rho)$ removes the user's consent from the consent specification. Similarly, we have terms $addret(m)$ and $updRet(m)$ adding and updating retention in the consent respectively. Note that these are user-specific constructs meaning only the respective users can (in their session) add or remove his/her policy and retention.

expressions:	$m ::= v \mid x$
data values:	$\alpha \in \chi ::= m \mid m$ **tag** t
tags:	$t \in \gamma ::= \langle \widehat{x}, \widehat{p} \rangle$
policy:	$\rho \in \delta ::= \langle \mathcal{E}, \mathcal{A}, \mathcal{P} \rangle$
actions:	$use, collect, store, trans, delete \in \mathcal{A}$
entities:	$e, f, g, h \cdots \in \mathcal{E}$
users:	$u_1, u_2, u_3 \cdots \in \mathcal{U}$
purposes:	$p_1, p_2, p_3, \cdots \in \mathcal{P}$
channels:	$a^e, b^f, c^g \cdots \in \mathcal{C}$
database names:	$k, l, \cdots \in \mathcal{K}$
variables:	$x, y, z, \cdots \in \mathcal{V}$
terms:	$r, s, \cdots \in \chi \cup \mathcal{K} \cup \mathcal{V} \cup \mathcal{N} \cup \gamma \cup \delta$
identifiers:	$v, w \cdots \in \mathcal{C} \cup \mathcal{N} \cup \mathcal{V} \cup \mathcal{K}$

$$\begin{array}{ll} \text{Processes: } P ::= & 0 \mid P \mid\mid Q \mid \; *P \mid (\nu a)P \mid a?(r).P \mid \\ & v!(r).P \mid addCon(\rho).P \mid rmvCon(\rho).P \mid \\ & addRet(m).P \mid updRet(m).P \mid \\ & k \leftarrow store(\alpha).P \mid k \leftarrow delete(\alpha).P \mid \\ & k \leftarrow getdata(t).P \end{array}$$

$$\begin{array}{lll} S ::= & (G_d, \langle \mathcal{C}, \mathcal{D} \rangle \;) \blacktriangleright E & \textit{(system)} \\ E ::= & 0 & \textit{(empty)} \\ & e \llbracket\, P \,\rrbracket & \textit{(entity process)} \\ & u \llbracket\, P \,\rrbracket & \textit{(user process)} \\ & E \mid\mid E & \textit{(par composition)} \\ & (\nu a^e)E & \textit{(entity restriction)} \\ & & \\ \mathcal{C} ::= & u \rightarrow \langle \Theta, \mathcal{R} \rangle & \textit{(user consent)} \\ & \widehat{\mathcal{C}} & \textit{(consent set)} \\ \Theta ::= & \beta \rightarrow \widehat{\langle \eta, p \rangle} & \textit{(consent binding)} \\ & \widehat{\Theta} & \textit{(consent binding set)} \\ & & \\ \mathcal{D} ::= & u \rightarrow (\widehat{v}_t) & \textit{(user data binding)} \\ & \widehat{\mathcal{D}} & \textit{(data binding set)} \end{array}$$

Fig. 1: Syntax of CCAL. For BNF definitions over multiple lines, each line contributes an alternative.

b) Database process expressions : We envision storing, retrieving, and deleting data in a database through process expressions to explicitly model if the right entity can perform these operations. Where $k \leftarrow store(\alpha).P$ defines a process as sending personal data α on a database channel k to store the data, $k \leftarrow delete(\alpha).P$ sends the personal data via the database channel, which deletes the data associated to users from the database and to retrieve data from database term $k \leftarrow getdata(t)$ is used where the entity sends valid tags to get the respective data for the users associated with the tag. Note that this will be a dual action where the entity sends a request to get data of a particular set of users and from the database side it sends the data and receiving is done from standard input expression $a?(x)$ on the entity side.

c) Personal Data: In our calculus, the property of data protection acts as a foundation upon which the processes communicate. Systems are composed of various entity sessions running parallel. To build this privacy model within our systems, we need to clarify how personal data flow is processed between various agent sessions. To envision purpose-based processing of personal data, we introduce tagged data values α, with a constructor m **tag** t to associate personal data with identifiers and purposes. Here tags t of the form $\langle \widehat{x}, \widehat{p} \rangle$ (where $\widehat{x}$ evaluates to $\widehat{u}$) are attached to expression m , so that the value becomes private. Note that the tag $\langle \widehat{x}, \{p, q\} \rangle$ on some data expresses that the data can be used for purpose p or q. The tag $\langle \widehat{x}, \emptyset \rangle$ expresses that data cannot be used for any purposes. We syntactically abuse this tag construct so that it can be applied on sets, lists and tuples, e.g., (”John Smith”, 25, ”Oslo”) tag ⟨john, Registration⟩. This explicit tagging could be replaced by automatic and implicit tagging, given a strategy, e.g., for detecting users and purposes. However, such an extension is beyond the scope of this paper. As a compensation the language allows explicit tagging to be provided to the programmer.

d) Consent Specification: The syntax of the calculus allows us to define the entities denoted by $\mathcal{E}$, purposes $\mathcal{P}$, actions $\mathcal{A}$ and users $\mathcal{U}$. A privacy policy ρ is defined as a set of tuples $(\mathcal{E}', \mathcal{A}', \mathcal{P}')$ where the entities $\mathcal{E}'$ are a subset of declared $\mathcal{E}$, purposes $\mathcal{P}'$ are a subset of the declared purposes $\mathcal{P}$, and actions $\mathcal{A}'$ are a subset of the predefined actions in the syntax of the calculus i.e., **use**, **collect**, **store**, **trans** and **delete**.

Intuitively these *actions* express the legal actions permitted to an entity on some personal data α. Action **use** gives read access to personal data α through the channel, **collect** gives collect access for active substitution of the personal data received over a shared to a variable of the current process, **store** allows to store personal data to database, **delete** allows deleting the personal data in the database and **transfer** allows to send the data over the output channel to the entity tagged on output capability of the channel. Note that we specify a distinct syntactic notion of a variable within the calculus: For example, in $a?(x).P$, upon receiving a personal data value on channel a, **use** allows reading the value on the channel (without active substitution) **collect** access allows to substitute the value to a variable x and continue as process P.

Our consent specification is a run-time element, which is globally available in the configuration. Any personal data in the system is handled following the consent specification available at that state. Consent specification mainly aims to capture the privacy preferences of each user; consent $\mathcal{C}$ consists of $u \rightarrow \langle \Theta, \mathcal{R} \rangle$ where u is a user, Θ is a policy map representing a set of bindings $\beta \rightarrow \widehat{\langle e, p \rangle}$ that mainly records for every action β, a set of pairs of entity and purposes expressing that entity e is allowed to perform action β for the user u and for the purpose p. Here, empty policy mapping i.,e $\beta \rightarrow \emptyset$ states that no entity or purposes are allowed to perform this action. $\mathcal{R}$ in consent is the retention time of the data specified by the user i.,e any data handling post retention time should be disallowed in the system. We record retention mainly in days.

e) Privacy-Aware Database Specification: We define a system database as the following, Let database $\mathcal{D}$ consist of set of mapping $u \rightarrow (\widehat{v}_t)$ from the user to all tagged private values, here we address challenge 5 from Sec[III] by making the database equipped with purposes through tags. Here user could be a database key or identifier of the user, which can adapted based on the database system in place. Such details on the design architecture of the database are out of the scope of this paper. We have three process expressions to mainly request the database i.,e to store, delete, and retrieve data from the database as discussed earlier.

f) Global Time Model : We define a global time model within our system for modeling and analysis of retention. In our calculus, a clock G_{clk} gives access to a time model. A clock can be chronometric or Logical. The former is related to physical time whereas the latter is not. In this paper, we model time as discrete-time logical clocks. Logical clocks are defined originally by .. **CP:** add a reference here

Definition 1 (Global Clock): We define global clock G_{clk} as a tuple $G_{clk} = \langle \mathcal{T}, \prec, \mathcal{L}, \Lambda, t \rangle$ where $\mathcal{T}$ is a set of Instances, $\prec$ is a quasi order relation on instances $\mathcal{T}$, $\mathcal{L}$ is set of labels , $\Lambda : \mathcal{T} \rightarrow \mathcal{L}$ is a mapping, t is a symbol for time unit, usually considered as tick for logical clocks. The ordered set $\langle \mathcal{T}, \prec \rangle$ is a temporal structure associated with the global clock. $\prec$ is a total, transitive and irreflexive binary relation on $\mathcal{T}$.

A discrete-time logical clock is a clock with a discrete set of instances $\mathcal{T}$. Since $\mathcal{T}$ is discrete, it can be indexed with natural numbers that follow the ordering on $\mathcal{T}$. Let $idx : \mathcal{T} \rightarrow \mathbb{N}_{>0}$, $\forall I \in \mathcal{T}$, $idx(I) = k$ if and only if i is the k^{th} instance of i . We define a discrete time date clock using our global clock to model retention as follows, note that using our system clock one can define their clock based on the time granularity required for eg., days, months, years, etc.

Definition 2 (Date Clock): Let G_d be a date clock, such that $G_d = \langle \mathcal{T}_d, \prec_d, \mathcal{L}_d, \Lambda_d, t_d \rangle$, and $G_d(k)$ denotes the k^{th} instant in $\mathcal{T}_d$;$\forall i \in \mathcal{T}$ the index $idx(i) = k$ where $i = G_d(k)$. For simplicity, we assume i_c as the current instant of the clock, $'i$ be the unique immediate predecessor of i in T_d and i' be the unique immediate successor of i in T_d.

g) Events: An event is a flexible low-level synchronization primitive within the calculus that is used to construct a form of communication. The occurrence of an event might trigger processes. We specify a special time tick event $e_{\#T}$, which will triggered after a period of specified simulation time T. This T can be a bus cycle, processor cycle, or a real-world clock that generates events within the system's clock. In our date clock $G_d = \langle \mathcal{T}_d, \prec_d, \mathcal{L}_d, \Lambda_d, t_d \rangle$ when the time unit t_d is a tick event $e_{\#T}$ then the time advances in the clock for a next date. We explain more on time advance semantics later using figure 5.

V. Operational Semantics

We want to bring the discussion to the operational semantics of our calculus stating that statically we do not have all the information within our system known, since our system is based on a consent specification paradigm and is interacting, open, and dynamically granted by the user at runtime. Even though knowing the system through static methods is interesting from a theoretical point of view, it may not be usable in a dynamic consent environment in practice. In this section, we introduce the operational semantics of the system via labeled transition rules which allows us to highlight the interactions with the environment. In addition, the LTS provides operational semantics for any minimal system that can implement the GDPR specifications based on our consent specification during runtime. The standard way to represent the interactions within a system is by labeling the system's evolution. Hence we define a labeled transition system $S \xrightarrow{\mu} S^{'}$, where system S will execute the action indicated by label μ and evolve to system $S^{'}$, letting μ range over the labels. The operational semantics come with enriched consent specification and runtime checks to avoid illegal handling of personal data. The labels that can be passed to perform actions are extended with that of standard π calculus. We distinguish four kinds of labels: standard communication labels, user interaction labels, and database interaction labels and time events.

$\mu ::= \gamma \mid e_{\#T} \mid a^e!(r) \mid a?(r) \mid a \leftarrow addCon(\rho) \mid a \leftarrow rmvCon(\rho) \mid a \leftarrow addRet(v) \mid a \leftarrow updRet(v) \mid k \leftarrow store(\alpha) \mid k \leftarrow delete(t) \mid k \leftarrow getdata(t)$

The label μ represents the internal actions of the system. Standard communication labels $a?(r)$ and $a^e!(r)$ show the actions of sending/receiving term r onto/from the channel a. The user interaction labels are as follows, $a \leftarrow addCon(\rho)$ defines the action of sending a policy ρ onto the channel a to add consent, and label $a \leftarrow rmvCon(\rho)$ shows the action of removing consent by sending the policy ρ on channel a
At the center of the transition rules is the reduction relation $\rightarrow$ on extended processes, mentioned in section **??**, which models the steps of computation. **CP:** add reduction rules here
Since the database is the one giving access to store, delete, and access of personal data, we have three database interaction labels $k \leftarrow store(\alpha)$, $k \leftarrow delete(t)$ and $k \leftarrow getdata(t)$.

A. Auxiliary functions

In our calculus, we introduce several auxiliary functions that support our operational semantics and are mainly used for privacy-related checks. We use the following functions to extract users and purposes from the tags. Functions $users(t)$ extract the users and $purpose(t)$ purposes, respectively.

$$users(t) = \{u \mid t = \langle \widehat{u}, \widehat{p} \rangle \wedge u \in \widehat{u} \}$$
$$purpose(t) = \{p \mid t = \langle \widehat{u}, \widehat{p} \rangle \wedge p \in \widehat{p}\}$$

Let function $\mathcal{A}(\mathcal{C}, e, u, \widehat{p})$ return a set of allowed actions for entity e, user u, purposes $\widehat{p}$ in consent specification $\mathcal{C}$.

$$\mathcal{A}(\mathcal{C}, e, u, \widehat{p}) = \{a \mid \mathcal{C}(u) = \langle \Theta, \mathcal{R} \rangle \wedge \exists p \in \widehat{p} \quad (e, p) \in \Theta(a)\}$$

We define a function $Act(\mathcal{C}, \eta, t)$ that, for any given tags and entity, extracts the allowed actions granted by the users in the consent $\mathcal{C}$. If the tags t are empty, it implies that the data in question is non-private and all actions, i.e., use, collect, transfer, store, are granted.

$$Act(\mathcal{C}, e, t) = \begin{cases} \bigcap_{i=1}^{n} \mathcal{A}(\mathcal{C}, e, u_i, \widehat{p}), \text{if } t = \langle \widehat{u}, \widehat{p} \rangle \wedge \widehat{p} \neq \emptyset \\ \{\mathit{use}, \mathit{collect}, \mathit{trans}, \mathit{store}\}, \text{if } t = \emptyset \\ \emptyset, \quad \text{otherwise} \end{cases}$$

We have functions exclusive to database operations. Note that our database requests can be imagined in the form of queries, and we do not define the database process further to show the encoding but give an abstract idea data operation within the database. Let $datamap(u, w, \mathcal{D})$ be the function that maps an entry to the database for a user. Since our database is a mapping of users and their respective values associated, a new private value w with its tags is added to the set of values v_t.

$$datamap(u, w, \mathcal{D}) = u \to \{\widehat{v}_t \cup w\} \text{ s.t } \mathcal{D}(u) = \widehat{v}_t \wedge w = v'_t$$

$$adddata(\widehat{u}, w, \mathcal{D}) = \bigcup_{i=1}^{n} datamap(u_i, w, \mathcal{D}), \text{ s.t } \widehat{u} = u_1 \dots u_n$$

we define function $adddata(\widehat{u}, w, \mathcal{D})$ which adds the mapping for all the users $\widehat{u}$ with same data value.

we define functions $delmap(u, \mathcal{D})$ and $deldata(\widehat{u}, \mathcal{D})$ to delete the mapping of the users from the database. This works analogously to adding the data mappings.

$$delmap(u, \mathcal{D}) = u \to \emptyset \ s.t \ \exists u \wedge \mathcal{D}(u) = \widehat{v}_t$$

$$deldata(\widehat{u}, \mathcal{D}) = \bigcup_{i=1}^{n} delmap(u_i, \mathcal{D}), s.t \ \widehat{u} = u_1 \dots u_n$$

We also have function $ret(\widehat{u}, \mathcal{C})$ to extract the retention of the users $\widehat{u}$ from their respective consent.

The rules defining the LTS transitions $S \xrightarrow{\mu} S'$, are given in four separate parts, first we give standard transition rules in figure 2, the database interaction rules in 3, the user interaction rules in figure 4, and global time transition rules in figure 5. The overall structure is similar to the standard π calculus with our privacy extensions. According to rule [L-INP], the input prefix $a?(x)$ may interact with the input action $a?(w)$, the rules apply for both personal and non-personal data rule [L-INP] denotes the process of sending data on the channel a, since the data comes with runtime tags in our semantics, using $tags(v_t)$ function tags of the data is extracted and respectively the users and purposes attached with the tag. Premises mainly check if the bounded entity e listening on the channel a has access *use* and *collect* in order to receive the value over the channel and perform active substitution of data within the local scope. Action credentials are checked for tags received against the users consent.

Retention is checked if the current instant of the Global clock $G_d(i_c)$ is less than or equal to the retention in the consent for all users $\widehat{u}$. If the access is not available or the retention date is invalid in the consent, the rule gets blocked and does not proceed further. Note that if the data has empty tags attached, indicating it's public data, the communication proceeds with privacy checks being true, and all the premises hold. Rule [L-OUT] the output prefixed $a^f!(v_t)$ process may react with the output action $a^f!(v_t)$ and continue as process P if all the privacy checks are satisfied. To send privacy value v_t over the channel, we have to specify the entity location where communication has to happen a^f. Action credentials i.e, *use*, *transfer* are checked for the tags attached, current entity e against the consent $\mathcal{C}$ and retention is checked w.r.t global clock and since we are transferring the data to entity f we check if the entity is able to *use*, *collect* the data in the consent. Communication arises in rule [L-COMM] by means of a τ action obtained by a synchronisation of an $a?(w)$ action with a $a^f!(w)$ action. Rule [L-RES] states that actions are closed under the restriction operation provided that the restricted name does not occur freely in any action. Note that a restriction of a channel does not inhibit the use or mentioning of other names. Rule [L-PAR] states that the actions are closed under the parallel composition provided that there is no name conflict between the bounded names of the actions and the free names of the parallel composition. Rules [L-SPLIT] [L-REPL] are standard rules that assume alpha conversion.

Figure 3 shows the database interaction rules, rule [L-STOREDATA] with database process expression prefix observes the label $k \leftarrow store(w)$, checks for action credentials, validity of retention and as a result stores the data in database, if the same data with multiple users is received data is stored for all the users individually. Rule [L-DELDATA] shows the process interacting with $k \leftarrow delete(w)$ label performs privacy checks and deletes the data from the respective users. whereas rule [L-GETDATA] shows a process inetraction with label $k \leftarrow getdata(w)$ gets the runtime tags attached with users and purposes, upon satisfying privacy checks process will have a dual **CP:** check getdata rule

Figure 4 shows user interaction rules, users in our system are program elements and are statically available. Rule [L-ADDPOL] defines a process where user adds consent via prefix $x \leftarrow addCon(\rho)$ where the users identity is validated and sends the policy ρ via channel x after the process interacts with the label, the policy is added to the consent specification. Note that here, the respective user from the session is matched against the user identifier in the consent specification. The rule [L-REMPOL] removes the policy bindings from the consent specification respectively. The rules [L-ADDRET] and [L-UPDRET] add and update the retention in the consent specification within the respective user session.

Figure 5 defines two rules specifying the global clock semantics. Semantics for the time model does not explicit the computation time of each operation. In particular, two instants of global date clock G_d are independent of computation time in between. When the time unit t_d has a tick event, the system interacts with a time event label $e_{\#T}$ the rule [L-clocktick] advances the clock by adding new instant $\mathcal{T}'_d$, label $\mathcal{L}'_d$ and new mapping Λ'_d. For each clock advancement, the retention date of all the users in the database is checked against the system clock, if the retetion check does not satisafy we have rule

(L-Inp)
$$\frac{\alpha = v_t \quad \widehat{u} = Users(t) \quad \widehat{p} = Purp(t) \\ \{use, collect\} \subseteq Act(e, \widehat{u}, \widehat{p}, \mathcal{C}) \quad G_d(i_c) \leq ret(\widehat{u}, \mathcal{C})}{(G_d, \langle \mathcal{C}, \mathcal{D} \rangle) \triangleright e\llbracket a?(x)\ .P \rrbracket \xrightarrow{a?(w)} (G_d, \langle \mathcal{C}, \mathcal{D} \rangle\ ) \triangleright e\llbracket P\{w/x\} \rrbracket}$$

(L-out)
$$\frac{\widehat{u} = Users(t) \quad \widehat{p} = Purp(t) \quad \{use, transfer\} \subseteq Act(e, \widehat{u}, \widehat{p}, \mathcal{C}) \\ \{use, collect\} \subseteq Act(f, \widehat{u}, \widehat{p}, \mathcal{C}) \quad G_d(i_c) \leq ret(\widehat{u}, \mathcal{C})}{(G_d, \langle \mathcal{C}, \mathcal{D} \rangle) \triangleright e\llbracket a^f!(v_t).P \rrbracket\ [5pt] \xrightarrow{a^f!(v_t)} (G_d, \langle \mathcal{C}, \mathcal{D} \rangle\ ) \triangleright e\llbracket P \rrbracket}$$

(L-comm)
$$\frac{(G_d, \langle \mathcal{C}, \mathcal{D} \rangle) \triangleright E \xrightarrow{a?(w)} (G_d, \langle \mathcal{C}, \mathcal{D} \rangle) \triangleright E' \quad (G_d, \langle \mathcal{C}, \mathcal{D} \rangle) \triangleright T \xrightarrow{a^f!(w)} (G_d, \langle \mathcal{C}, \mathcal{D} \rangle) \triangleright T'}{(G_d, \langle \mathcal{C}, \mathcal{D} \rangle) \triangleright E \mid\mid T \xrightarrow{\tau} (G_d, \langle \mathcal{C}, \mathcal{D} \rangle) \triangleright E' \mid\mid T'}$$

(L-Par)
$$\frac{(G_d, \langle \mathcal{C}, \mathcal{D} \rangle) \triangleright E \xrightarrow{\mu} (G_d, \langle \mathcal{C}', \mathcal{D}' \rangle) \triangleright E' \quad bc(\mu) \cap fc(T) = \emptyset}{(G_d, \langle \mathcal{C}, \mathcal{D} \rangle) \triangleright E \mid\mid T \xrightarrow{\mu} (G_d, \langle \mathcal{C}', \mathcal{D}' \rangle) \triangleright E' \mid\mid T}$$

(L-Res)
$$\frac{(G_d, \langle \mathcal{C}, \mathcal{D} \rangle) \triangleright E \xrightarrow{\mu} (G_d, \langle \mathcal{C}, \mathcal{D} \rangle) \triangleright E' \quad a^e \notin Fc(\mu)}{(G_d, \langle \mathcal{C}, \mathcal{D} \rangle) \triangleright (\nu a^e)E \xrightarrow{\mu} \langle \mathcal{C}, \mathcal{D} \rangle \triangleright (\nu a^e)E'}$$

(L-Repl)
$$\frac{(G_d, \langle \mathcal{C}, \mathcal{D} \rangle) \triangleright e\llbracket\ P \mid *P\ \rrbracket \xrightarrow{\mu} (G_d, \langle \mathcal{C}', \mathcal{D}' \rangle) \triangleright E}{(G_d, \langle \mathcal{C}, \mathcal{D} \rangle) \triangleright e\llbracket *P\ \rrbracket \xrightarrow{\mu} (G_d, \langle \mathcal{C}', \mathcal{D}' \rangle) \triangleright E}$$

(L-split)
$$\frac{(G_d, \langle \mathcal{C}, \mathcal{D} \rangle) \triangleright e\llbracket\ P\ \rrbracket \mid\mid e\llbracket\ Q\ \rrbracket \xrightarrow{\mu} (G_d, \langle \mathcal{C}', \mathcal{D}' \rangle) \triangleright E}{(G_d, \langle \mathcal{C}, \mathcal{D} \rangle) \triangleright e\llbracket\ P \mid Q\ \rrbracket \xrightarrow{\mu} (G_d, \langle \mathcal{C}', \mathcal{D}' \rangle) \triangleright E}$$

Fig. 2: Standard execution rules in CCAL

(L-Storedata)
$$\frac{\alpha = v_t \quad \widehat{u} = Users(t) \quad \widehat{p} = Purp(t)\{use, store\} \subseteq Act(e, \widehat{u}, \widehat{p}, \mathcal{C}) \\ \mathcal{D}' = adddata(\widehat{u}, w, \mathcal{D}) \quad G_d(i_c) \leq ret(\widehat{u}, \mathcal{C})}{(G_d, \langle \mathcal{C}, \mathcal{D} \rangle) \triangleright e\llbracket k \leftarrow store(\alpha).P \rrbracket \xrightarrow{k?\leftarrow store(w)} (G_d, \langle \mathcal{C}, \mathcal{D}' \rangle\ ) \triangleright e\llbracket P\{w/\alpha\} \rrbracket}$$

(L-Deldata)
$$\frac{\alpha = v_t \quad \widehat{u} = Users(t) \quad \widehat{p} = Purp(t)\{use, delete\} \subseteq Act(e, \widehat{u}, \widehat{p}, \mathcal{C}) \\ \mathcal{D}' = deldata(\widehat{u}, w, \mathcal{D}) \quad G_d(i_c) \leq ret(\widehat{u}, \mathcal{C})}{(Gclk, \langle \mathcal{C}, \mathcal{D} \rangle) \triangleright e\llbracket k \leftarrow delete(\alpha).P \rrbracket \xrightarrow{k?\leftarrow delete(w)} (Gclk, \langle \mathcal{C}, \mathcal{D}' \rangle\ ) \triangleright e\llbracket P\{w/\alpha\} \rrbracket}$$

(Lin-Getdata)
$$\frac{t = \langle \widehat{u}, \widehat{p} \rangle \quad \{use, collect\} \subseteq Act(e, u, \widehat{p}, \mathcal{C}) \quad \delta = getdata(u, \widehat{p}, e, \mathcal{C}) \\ G_d(i_c) \leq ret(u, \mathcal{C})}{(G_d, \langle \mathcal{C}, \mathcal{D} \rangle) \triangleright e\llbracket k \leftarrow getdata(t).P \rrbracket \xrightarrow{k\leftarrow getdata(w)} (G_d, \langle \mathcal{C}, \mathcal{D} \rangle\ ) \triangleright e\llbracket P\{w/\alpha\} \rrbracket}$$

(Lout-Getdata)
$$\frac{\alpha = v_t\ \{use, collect\} \subseteq Act(e, u, \widehat{p}, \mathcal{C}) \quad t = \langle u, \widehat{p} \rangle \\ G_d(i_c) \leq ret(u, \mathcal{C})}{(G_d, \langle \mathcal{C}, \mathcal{D} \rangle) \triangleright e\llbracket k \rightarrow getdata(\alpha).P \rrbracket \xrightarrow{k\rightarrow getdata(w)} (G_d, \langle \mathcal{C}, \mathcal{D} \rangle\ ) \triangleright e\llbracket P \rrbracket}$$

Fig. 3: Execution rules for database query statements in PAOL.

(L-AddPol)
$$\frac{(u) == (v) \quad \Theta' = InsertPol(\Theta, \rho)}{(G_d, \langle (u \rightarrow \langle \Theta, \mathcal{R} \rangle); \mathcal{C}, \mathcal{D} \rangle) \triangleright v\llbracket x \leftarrow addCon(\rho).P \rrbracket \xrightarrow{x\leftarrow addCon(w)} (G_d, \langle (u \rightarrow \langle \Theta', \mathcal{R} \rangle); \mathcal{C}, \mathcal{D} \rangle \rangle) \triangleright e\llbracket P\{w/\rho\} \rrbracket}$$

(L-RemPol)
$$\frac{(u) == (v) \quad \Theta' = RemPol(\Theta, \rho)}{(G_d, \langle (u \rightarrow \langle \Theta, \mathcal{R} \rangle); \mathcal{C}, \mathcal{D} \rangle) \triangleright v\llbracket x \leftarrow remCon(\rho).P \rrbracket \xrightarrow{x\leftarrow remCon(w)} (G_d, \langle (u \rightarrow \langle \Theta', \mathcal{R} \rangle); \mathcal{C}, \mathcal{D} \rangle) \triangleright e\llbracket P\{w/\rho\} \rrbracket}$$

(L-AddRet)
$$\frac{(u) == (v) \quad \mathcal{R}' = w}{(G_d, \langle (u \rightarrow \langle \Theta, \emptyset \rangle); \mathcal{C}, \mathcal{D} \rangle) \triangleright v\llbracket x \leftarrow addRet(r).P \rrbracket \xrightarrow{x\leftarrow addRet(w)} (G_d, \langle (u \rightarrow \langle \Theta, \mathcal{R}' \rangle); \mathcal{C}, \mathcal{D} \rangle \rangle) \triangleright e\llbracket P\{w/r\} \rrbracket}$$

(L-UpdRet)
$$\frac{(u) == (v) \quad \mathcal{R}' = w}{(G_d, \langle (u \rightarrow \langle \Theta, \mathcal{R} \rangle); \mathcal{C}, \mathcal{D} \rangle) \triangleright v\llbracket x \leftarrow updRet(r).P \rrbracket \xrightarrow{x\leftarrow updRet(w)} (G_d, \langle (u \rightarrow \langle \Theta, \mathcal{R}' \rangle); \mathcal{C}, \mathcal{D} \rangle \rangle) \triangleright e\llbracket P\{w/r\} \rrbracket}$$

Fig. 4: Execution rules for user-interaction statements in PAOL.

rule [L-Clocktick delete] which deletes of user data with expired retention dates, meaning if the retention of any users within the database is greater than the current instant of the global clock, then such user's data will be deleted from the database.

B. Thoughts on Error Handling when lacking consent

The semantics of the calculus does not include runtime error handling. If a runtime error occurs in one of the processes, the execution of the process will stop, and the process thread will get stuck. Since our calculus is mainly concerned with consent-based processing of user data, errors might occur at runtime due to a lack of consent for handling personal data. One alternative to overcome such errors is the use of a *dynamic consent interaction dialogue with the user*, where a dialogue box is introduced asking for the required consent from the

$$\text{(L-ClockTick)}\quad \frac{t_d = e_{\#T} \quad \mathcal{U} = alluser(\mathcal{D}) \quad RetOk(\mathcal{U}, \mathcal{D}, G_d(i_c))}{(\langle \mathcal{T}_d, \prec_d, \mathcal{L}_d, \Lambda_d, t_d\rangle, \langle \mathcal{C}, \mathcal{D}\rangle) \triangleright E \xrightarrow{e_{\#T}} (\langle \mathcal{T}'_d, \prec_d, \mathcal{L}'_d, \Lambda'_d, t_d\rangle, \langle \mathcal{C}, \mathcal{D}\rangle) \triangleright E}$$

$$\text{(L-ClockTick delete)}\quad \frac{t_d = e_{\#T}\ \mathcal{U} = alluser(\mathcal{D})\ \neg RetOk(\mathcal{U}, \mathcal{D}, G_d(i_c)))\ \mathcal{D}' = DelExpired(\mathcal{U})}{(\langle \mathcal{T}_d, \prec_d, \mathcal{L}_d, \Lambda_d, t_d\rangle, \langle \mathcal{C}, \mathcal{D}\rangle) \triangleright E \xrightarrow{e_{\#T}} (\langle \mathcal{T}'_d, \prec_d, \mathcal{L}'_d, \Lambda'_d,\ t_d\rangle, \langle \mathcal{C}, \mathcal{D}'\rangle) \triangleright E}$$

Fig. 5: Execution rules for user-interaction statements in PAOL.

user. Regarding GDPR, this could be reflected by opt-in/opt-out requests, that user allow dynamically at runtime, we can further refine the options to: 1) *Allow-once:* for temporary consent, with a new dialogue next time when consent is needed. 2) *Allow-Always:* for changing the consent so that it is used for further processing. Failure to apply any rule within the LTS will result in runtime errors that can be modeled within our calculus. The process in the syntax can be extended with three different errors: standard errors, access errors, and retention errors.

$$P ::= \text{Err} \mid \text{accesErr} \mid \text{retErr}$$

We give error semantics example for two LTS rules for input message in fig 6. However, the rest of the rules are straightforward and can be written in a similar style. To explain error semantics, we consider the [L-InpAccessErr] rule, listening on a channel upon receiving personal data premises checks for credentials for right access i.e., use collect and lack of access in current consent results in runtime access error. Similarly, if the global clock has exceeded user retention, results in a retention error given in the rule [L-InpRetErr].

C. Reflection on semantics by example: E-Health System

Nowadays, many user-centric services are delivered through smart systems and internet-connected devices. It is crucial to model these pervasive systems within the healthcare sector, given the extensive and sensitive nature of information managed by E-health systems. This includes individual details like medical history, diagnostic reports, treatment plans, etc. E-health systems often employ sensors and monitors to effortlessly collect patient data, with continuous monitoring and extracting insights; while all of this makes the user interaction with the hospital system easier, it still fails to empower individuals privacy. We model an E-health system using CCAL, which mainly involves user's personal data flow within the systems by current consent among various entities. Let *Doctor*, *Nurse*, *Patient* and *Lab* be the various system components. The workflow of the E-health scenario is as follows: 1) User Alice is registered in the system and can manage consent at any stage in processing. 2) Alice provides her blood samples to a lab to generate some results. 3) Lab collects the sample data, performs computation, saves reports in the storage system, and sends a copy of them to the Nurse. 4) Nurse collects the lab reports, performs analysis, and sends them to the Doctor. 5) Doctor retrieves patient details from the storage, receives an analysis report from the Nurse, and performs further diagnosis.

For this concrete scenario, we assume global date clock G_d in the current configuration is dated 31st March 2023 and $\mathcal{C}_h$ and $\mathcal{D}_h$ as the global consent and database given as follows,

$$\begin{aligned}
\mathcal{C}_h \quad &\triangleq \quad (alice \rightarrow \{use \rightarrow \langle (doctor, trt), (nurse, trt)\rangle,\\
&\quad collect \rightarrow \langle \{(doctor, trt), (nurse, trt)\rangle\},\\
&\quad store \rightarrow \langle \{(doctor, trt), (nurse, trt)\rangle\},\\
&\quad trans \rightarrow \langle \{(doctor, trt), (nurse, trt)\rangle\},\\
&\quad \text{1st April 2023})\\
\mathcal{D}_h \quad &\triangleq \quad (alice \rightarrow \{Id_{t1}, address_{t2}\}, bob \rightarrow \{reports_{t3}\})
\end{aligned}$$

Here user Alice consent allows entities *doctor* , *nurse* to *use*, *collect* data for purpose of *trt*. The database currently stores data values *Id*, *Address* along with their private tags. We model the system based on the specifications as follows,

$$\begin{aligned}
\mathit{System} \quad &\triangleq \quad (G_d, \langle \mathcal{C}_h, \mathcal{D}_h\rangle) \triangleright Doc \mid Nur \mid Patient \mid Lab\\
\mathit{Patient} \quad &\triangleq \quad alice\ [\![a \leftarrow addCon(\rho_1).a \leftarrow addRet(r)\\
&\quad b^{lab}!(sample\ \textbf{tag}\ \{alice\}, \{trt, lab\})]\!]\\
\text{Lab} \quad &\triangleq \quad l[\![b?(x\ \textbf{tag}\ t).\\
&\quad c^{nur}!(rep\ \textbf{tag}\ \{alice\}, \{trt, lab\})]\!]\\
\text{Nur} \quad &\triangleq \quad nur[\![c?(y\ \textbf{tag}\ u).k \leftarrow store(y\ \textbf{tag}\ u)\\
&\quad d^{doc}!(analysis\ \textbf{tag}\ (\{alice\}, \{trt, lab\}))]\!]\\
\text{Doc} \quad &\triangleq \quad doc[\![d?(z).k \leftarrow getdata((\{alice\}, \{trt\})).\\
&\quad k?(x).k \leftarrow delete(y\ \textbf{tag}\ u)]\!]
\end{aligned}$$

Here doctor, nurse, and lab are entities in the system and the patient is the user. Patient alice updated her consent by sending her policy ρ_1 and retention r via channel a. Here the user interaction rules [L-ADDPOL] and [L-ADDRET] are applied respectively updating the global consent and sends her tagged personal data sample to the lab via channel b. Lab l gets a sample x **tag** t from a patient on channel b, lab has both use and collect rights for trt purpose in the consent and retention checks holds hence rule [L-Inp], applies, after local computation process sends lab reports rep **tag** $(\{alice\}, \{trt, lab\})$ to nurse via channel c and respective actions, use and transfer and retention validity checks are done by [L-out]. Nurse receives data from the lab via channel c, privacy checks are done by rule [L-out], nurse stores the data to database via rule [L-STOREDATA]performs analysis and sends the analysis to the entity doctor via channel d. Here again the respective rules checks for valid privacy requirements. The doctor receives an analysis from the Nurse and retrieves other data of the user from the database applying rule [L-GETDATA] however, when doctor tries to delete the data in database via database channel k action use and delete are checked against the consent via rule [L-DELDATA], and fails to apply the rule stopping the consent violation. With such a modeling example can help understand and reason about consent needed from the users to perform or analyse the data and provide services accordingly.

(L-InpAccessErr)
$$\frac{\begin{array}{c}\alpha = v_t \quad \widehat{u} = Users(t) \quad \widehat{p} = Purp(t) \\ \neg(\{use, collect\} \subseteq Act(e, \widehat{u}, \widehat{p}, \mathcal{C})) \quad G_d(i_c) \leq ret(\widehat{u}, \mathcal{C})\end{array}}{(G_d, \langle \mathcal{C}, \mathcal{D} \rangle) \triangleright e[\![a?(x)\ .P]\!] \xrightarrow{a?(w)} (G_d, \langle \mathcal{C}, \mathcal{D} \rangle\ ) \triangleright e[\![\text{accessErr}]\!]}$$

(L-InpRetErr)
$$\frac{\begin{array}{c}\alpha = v_t \quad \widehat{u} = Users(t) \quad \widehat{p} = Purp(t) \\ \{use, collect\} \subseteq Act(e, \widehat{u}, \widehat{p}, \mathcal{C}) \quad \neg(G_d(i_c) \leq ret(\widehat{u}, \mathcal{C}))\end{array}}{(G_d, \langle \mathcal{C}, \mathcal{D} \rangle) \triangleright e[\![a?(x)\ .P]\!] \xrightarrow{a?(w)} (G_d, \langle \mathcal{C}, \mathcal{D} \rangle\ ) \triangleright e[\![\text{RetErr}]\!]}$$

Fig. 6: Error Execution rules for Input statement

The example also showcases when the data belongs to multiple users, how the rule checks for the consent from each user.

VI. Implementation

Maude is a powerful tool for implementing various semantics to experiment with quick prototypes for trying examples and proving properties. It is a high-level language and high-performance system supporting equational and rewriting logic computation. We use Maude as a metalanguage in which the syntax and semantics of the CCAL can be formally defined. It is easier to scale the implementation based on the executable in Maude. We have formalized the operational semantics, which provides a prototypic environment for our specification. We validate the running example from the case study with two versions, consented and unconsented specifications. The full executable in Maude and the running example can be downloaded from https://github.com/Chinmayiprabhu/CCALexecutable.git.

VII. Related Work

Notions such as privacy by design (PbD) [4], data protection by design and default [11], and legal compliance appear quite often in the literature; despite much ongoing work in this direction, the research community agrees about the need of practical guidance on such notions [8], [18]. Schneider indicates that privacy principles, such as PbD, go through various levels of abstraction from their conceptual models during design time until software implementation and do not entirely guarantee privacy [14]. Hence, there is a need for privacy-aware languages that help capture software design models with data privacy constructs that facilitate the check of privacy principles. The purpose based processing data is vital to privacy and formal approaches such as privacy-aware role-based access control models consider purposes, roles, and obligations to specify the access control policies and role-based access control (RBAC) models enriched with purpose-awareness [3], [9], [19] for enterprise data handling have been previously well-explored; these authorization models fail to capture the current regulatory requirements and consider only organizational interests and not user preferences. In existing language-based approaches, a programming language with privacy principles is checked either statically or at runtime. Researchers have previously explored information flow analysis to make programs comply with privacy policies, many of these approaches use static techniques [10], [15], [17], which will not be enough to fully capture GDPR compliance, since elements, such as consent change at runtime. Some runtime checking work, as in [7], [16], the former considers a subset of GDPR elements that we address and the latter has built in consent management operations but are not considering process calculi. There are well-explored works using pi-calculus in the state-of-the-art reasoning security and privacy properties. π-calculus serves as both a modeling language for systems and a tool for validating various properties. For instance, applied pi calculus, as discussed in [1], facilitates the specification and automatic analysis of security protocols Type system have been employed with π calculus in number of papers to reason about access control and policy authorisations however the exploration of the GDPR concepts is not prevalent in these works.. Closest to our work is the Privacy Calculus which proposes a modelling language incorporating the notions like purpose, consent and privacy policies. Nevertheless, in contrast to our approach, Privacy Calculus relies on static techniques to model consent, since consent is dynamic given on the fly static approaches will not be enough to reason about compliance. They also do not consider retention and deletion of data. In the direction of a general study of GDPR and legal compliance, Ranise and Siswantoro [13] propose an approach for privacy-aware automated legal compliance checking by using tools for policy analysis on efficient SMT solvers. Unfortunately, this was proposed before GDPR and did not consider crucial elements like consent, actions, entities, and other GDPR requirements. Piras et al. [12] propose the design of an architecture abiding by GDPR requirements. However, this approach lacks implementation details on compliance and does not address data interoperability at lower system operations, unlike our approach, where enforcement of data subjects' preferences and monitoring inconsistency with consent is handled on a data level, guaranteeing a concrete notion of compliance.

VIII. Conclusion

This paper proposes a modeling language that integrates privacy features into core π calculus and explores how syntax and semantics can be used to ensure personal data handling by design and default. The motivation behind incorporating privacy constructs, along with existing challenges in the field, is explored, emphasizing their integration into calculus for enforcing consent-based data processing. Operational semantics, offering such abstractions, are presented, and an evaluation is conducted through a healthcare example and a Maude-based simulation environment that allows us to simulate programs and experiment with our new calculus features. Our currently proposed semantics are fixed to a concrete interpretation of GDPR terminology. Exploring how to parametrize the transition rules concerning desired personal data handling checks according to different data privacy legislations and domain expertise would be interesting. The proposed data privacy checks introduce notable runtime overheads for every transition rule handling data, prompting future investigation into reducing checkpoints while still ensuring compliance. Techniques like behavioral types [6] could be explored to statically approximate the required checks and offer a type system for the calculus. To demonstrate correctness, the paper aims to prove bisimulation equivalence for systems.

References

[1] M. Abadi, B. Blanchet, and C. Fournet. The applied pi calculus: Mobile values, new names, and secure communication. *Journal of the ACM (JACM)*, 65(1):1–41, 2017.

[2] C. P. Baramashetru, S. L. Tapia Tarifa, O. Owe, and N. Gruschka. A policy language to capture compliance of data protection requirements. In *Integrated Formal Methods, IFM 2022, Proceedings*, volume 13274 of *Lecture Notes in Computer Science*, pages 289–309. Springer, 2022.

[3] J.-W. Byun, E. Bertino, and N. Li. Purpose based access control of complex data for privacy protection. In *Proceedings of the tenth ACM symposium on Access control models and technologies*, pages 102–110. ACM, 2005.

[4] A. Cavoukian and M. Chibba. Advancing privacy and security in computing, networking and systems innovations through privacy by design. In *Proceedings of the 2009 conference of the Centre for Advanced Studies on Collaborative Research*, pages 358–360. ACM, 2009.

[5] M. Hennessy and J. Riely. Information flow vs. resource access in the asynchronous pi-calculus. *ACM Transactions on Programming Languages and Systems*, 24(5):566–591, 2002.

[6] H. Hüttel, I. Lanese, V. T. Vasconcelos, L. Caires, M. Carbone, P.-M. Deniélou, D. Mostrous, L. Padovani, A. Ravara, E. Tuosto, H. T. Vieira, and G. Zavattaro. Foundations of session types and behavioural contracts. *ACM Comput. Surv.*, 49(1), 2016.

[7] F. Karami, D. A. Basin, and E. B. Johnsen. DPL: A language for GDPR enforcement. In *35th IEEE Computer Security Foundations Symposium, CSF 2022*, pages 112–129. IEEE, 2022.

[8] M. Kutyłowski, A. Lauks-Dutka, and M. Yung. GDPR – Challenges for Reconciling Legal Rules with Technical Reality. In *Computer Security – ESORICS 2020*, Lecture Notes in Computer Science, pages 736–755. Springer, 2020.

[9] A. Masoumzadeh and J. B. D. Joshi. PuRBAC: Purpose-Aware Role-Based Access Control. In *On the Move to Meaningful Internet Systems: OTM 2008*, Lecture Notes in Computer Science, pages 1104–1121. Springer, 2008.

[10] A. C. Myers and B. Liskov. Protecting privacy using the decentralized label model. *ACM Transactions on Software Engineering and Methodology*, 9(4):410–442, 2000.

[11] E. Network and I. S. Agency. *Privacy and data protection by design: from policy to engineering.* Publications Office, 2014.

[12] L. Piras et al. DEFeND Architecture: A Privacy by Design Platform for GDPR Compliance. In *Trust, Privacy and Security in Digital Business*, Lecture Notes in Computer Science, pages 78–93. Springer, 2019.

[13] S. Ranise and H. Siswantoro. Automated Legal Compliance Checking by Security Policy Analysis. In *Computer Safety, Reliability, and Security*, Lecture Notes in Computer Science, pages 361–372. Springer, 2017.

[14] G. Schneider. Is Privacy by Construction Possible? In *Leveraging Applications of Formal Methods, Verification and Validation. Modeling*, Lecture Notes in Computer Science, pages 471–485. Springer, 2018.

[15] S. Sen, S. Guha, A. Datta, S. K. Rajamani, J. Tsai, and J. M. Wing. Bootstrapping Privacy Compliance in Big Data Systems. In *2014 IEEE Symposium on Security and Privacy*, pages 327–342. IEEE, 2014.

[16] S. Tokas and O. Owe. A formal framework for consent management. In *Formal Techniques for Distributed Objects, Components, and Systems - FORTE 2020*, volume 12136 of *Lecture Notes in Computer Science*, pages 169–186. Spinger, 2020.

[17] S. Tokas, O. Owe, and T. Ramezanifarkhani. Static checking of GDPR-related privacy compliance for object-oriented distributed systems. *J. Log. Algebraic Methods Program.*, 125:100733, 2022.

[18] C. Utz, M. Degeling, S. Fahl, F. Schaub, and T. Holz. (Un) informed consent: Studying GDPR consent notices in the field. In *Proceedings of the 2019 ACM SIGAC conference on computer and communications security*, pages 973–990. ACM, 2019.

[19] N. Yang, H. Barringer, and N. Zhang. A Purpose-Based Access Control Model. In *Third International Symposium on Information Assurance and Security*, pages 143–148, 2007.