PSpec: A Formal Privacy Language for Data Analytics

Abstract
Business organizations regularly collect customer data to improve their services. Organizations may want to share data within themselves or even with third-parties to maximize data utility. Since business data contain lots of customer information, organizations must respect customers’ privacy expounded by privacy laws. In this paper, we present the formal privacy specification language PSpec (Privacy Specification) for data usage restrictions. Comparing with previous works, PSpec specializes in data analytics, and also provides explicit support for data desensitization and association to balance data privacy and utility. We moreover present redundancy and conflict analysis algorithms to help data owners write PSpec policies. To evaluate PSpec and our policy analysis algorithms, we carry out a case study on TPC-DS benchmark and several experiments on synthetic rules. The results demonstrate the applicability of PSpec and the efficiency of the policy analysis algorithms.

1. Introduction
As more and more personal data are being collected by organizations, privacy issues are becoming increasingly important in modern society. In the past decade, privacy specification languages have been developed to formalize the text-based privacy policies. Since organizations may have different privacy requirements, it is important to have a flexible language to specify various privacy requirements across departments. Typical works include EPAL [3] and XACML [19]. Meanwhile, customer information collected by business organizations is invaluable. Organizations can make profits by sharing such information with third parties. For instance, manufacturers may predict sale trends by analyzing sales information of a small retail company at cost. Advertisers may also pay for user information in social networks. Opportunities are abundant if data can be shared with third parties. Demands for sharing business data with third parties are becoming more viable.

However, business data contain lots of private information such as identities and personal preferences. Misusing such data can cause severe privacy breaches and even violate privacy laws. More importantly, it immediately tarnishes the organization’s reputation by attracting unwanted publicity. Business organizations could not be more concerned about protecting customers’ privacy when they share custom information within themselves or with business partners. Privacy protection however is not easy [4, 22]. Several mechanisms have been proposed to protect privacy from adversaries.

A natural idea for privacy protection is by anonymization [28]. Through anonymization, data related to private information are removed or generalized. Privacy hence can be protected. Anonymization techniques however depend on data contents. Dates of birth and shopping histories, for instances, require different techniques. When the privacy and utility requirements change, anonymization techniques need be fine-tuned to fulfill new requirements. Anonymization subsequently can be tedious and cost-ineffective. Releasing anonymized data may be impractical for diverse business data and privacy requirements.

A more recent and promising privacy protection mechanism is differential privacy [9, 10]. In differential privacy, business data are not shared. A carefully designed curator resolves a pre-determined set of queries from third parties instead. Through noisy answers, one can argue that private information cannot be leaked through the curator. Lots of researches are devoted to designing curators for various data and queries. Designing differentially private curators however requires tedious analysis and mathematical insights. Adopting differential privacy in organizations with varying business data can be too challenging at present.

From data owners’ viewpoint, a more desirable and practical approach is sharing information under privacy specifications. Similar to differential privacy, a curator is used to mediate data and queries. It however only enforces privacy specifications on queries. Privacy specifications instead protect privacy by depicting data usage restrictions. If a query
does not access the data according to the restrictions in privacy specifications, the curator immediately rejects the query. Privacy specifications are written by legal experts to adhere to privacy laws and policies. The practical approach effectively shifts the responsibility of privacy protection to privacy specifications [27].

Privacy specifications may involve complicated conditions and operations. Consider the sales information of an online retail company. The company may be willing to share the amounts of all transactions last year. It may be hesitant to share the amounts of all transactions associated with an IP address last year. Nevertheless, the company may release such sensitive information after desensitization. For instance, transaction amounts associated with truncated IP addresses can be shared, as well as differentially private average transaction amounts associated with full IP addresses. Privacy specifications can be very complex and even confusing. A formal foundation is certainly needed.

In order to describe intricate privacy specification, we present the formal privacy language PSpec (Privacy Specification) for specifying data usage restrictions in data analytics. Its formal semantics help developers implement curators easily and correctly. PSpec moreover abstracts away details of underling data models and desensitize operations. Its specifications can be enforced by existing data analytics systems like Hive and Spark-SQL. PSpec moreover can employ techniques from differential privacy as desensitize operations to share aggregated information.

After privacy specifications are written, it is equally important to ensure that the specifications are semantically sound. Thanks to its formal semantics, PSpec specifications can be analyzed to identify potential flaws in privacy policies. Two useful analyses are redundancy analysis and consistency analysis [1]. Intuitively, a rule is redundant if it has no effect on the entire specification; a set of rules are inconsistent if they may issue restrictions that cannot be satisfied simultaneously. A redundant rule wastes computation resources. Inconsistent rules indicate contradictory privacy policies. Both should be identified and carefully re-examined. Yet privacy policies are interdependent on data association. Manually identifying such problems among intricate real-world privacy policies is hard. We develop novel techniques to analyze specifications using SMT solvers.

Inconsistency among privacy specifications allowing data association is not straightforward to formulate. Traditional inconsistency considers privacy rules with only one data category. When data analysts wish to access two different data, privacy rules for the two data categories are triggered. If requirements of the triggered rules cannot be satisfied simultaneously, this is inconsistency. For data analytics, privacy rules with multiple data categories are common. Different desensitize requirements can be specified for different combinations of data categories. When multiple data are accessed, several privacy rules with different desensitize requirements can be triggered. It is easy to have unsatisfiable desensitize requirements when data association is allowed. Such unsatisfiable requirements precisely indicate that the requested data cannot be desensitized to protect privacy. They should not be seen as (classical) inconsistency. For analysis with data association, inconsistency need be defined properly to avoid such false alarms. We propose a new definition for data analytics.

The main contributions of this paper are as follows:

- We present PSpec, a formal privacy language for data analytics with explicit support for data association and desensitization.
- We generalize the formulation of inconsistency for privacy policy specifications allowing data association.
- We present redundancy and inconsistency analysis algorithms for PSpec to analyze the PSpec policy.
- Finally, we carried out a case study on TPC-DS [21] benchmark and performed extensive experiments on synthetic rules.

The rest of the paper is organized as follows. Section 2 and Section 3 present the language design and formal semantics of PSpec respectively. Section 4 gives redundancy and conflict analyses for PSpec. Section 5 reports our primary implementation and evaluation. Section 6 further discusses related works. Finally, Section 7 concludes this paper.

2. PSpec Language

We consider the scenario with the following participants:

- The data owner such as a retail company, who shares data with third-party analysts under agreements.
- The data analyst, who performs data analysis over the shared data.
- The data analytics system, which manages the data and allows the analyst to submit queries for data analysis.

Queries submitted by the data analyst may violate the data owner’s privacy policy. The primary goal of PSpec is to allow the legal team of the data owner to specify enforceable data usage restrictions such that only the privacy-compliant queries are authorized to access the data.

In the remainder of this section, we introduce the design of PSpec. As in EPAL [3], we separate PSpec into vocabulary and policy. Vocabulary defines a set of abstract concepts. Policy contains a set of PSpec rules.

2.1 Vocabulary

The data owner defines a set of user categories and data categories in vocabulary. A user category represents a role. A data category represents a privacy-related data concept. Both user and data categories are hierarchical. In a category hierarchy, a parent category is the generalization of its child categories; a child category is a specialization of its parent
category. When a parent category is referred, all its descend-ants are also referred except those excluded explicitly.

Fig. 1 shows category hierarchies for a retail company as an example. The top-level user category is Analyst repre-senting all data analysts. Analyst is divided into Report Ana-lyst, Marketing Analyst, and Advertise Analyst based on the tasks assigned to them. Data categories are classified as Key Attribute, Quasi Identifier, and Sensitive Attribute. Key At-tribute are the attributes that can uniquely identify an indi-vidual, such as Name and Phone. Quasi Identifier is the at-tributes that may identify individuals by association, such as the combination of Birth, Zip, and Gender [28]. Finally, Sen-sitive Attribute carries sensitive personal information, such as customers’ sales records.

The data owner also specifies supported desensitize opera-tions for data categories. Desensitize operations can be used to make sensitive data category ready for access. For example, desensitize operations for Price can be aggregate opera-tions such as Min, Max, and Avg; they can be the truncated operation for Zip. A data category automatically inherits its ancestors’ desensitize operations.

2.2 Policy

Before introducing details, we discuss data access issues pertaining to data analytics. In data analytics, accessing multiple pieces of information together may not be equivalent to accessing each piece separately because of association. For example, accessing a customer’s address and salary sepa-rately may be acceptable. Yet accessing them together causes a severe privacy breach. Instead of forbidding accessing any sensitive information, desensitization is useful to balance between privacy and utility in data analytics. For instance, full IP addresses may locate individuals; truncated IP addresses may be acceptable and useful to data analysis.

The data owner defines a set of PSpec rules to regulate data access. Informally, each rule states under what restrictions can a user category access certain data categories to-gether. We require a user category to satisfy all the applicable rules. The reason is that usually only a part of the business data is related to customers’ privacy. This allows the data owner to focus on privacy-related data.

The rule grammar is shown in Fig. 2. A rule contains a scope and restrictions (separated by =>). The scope is defined by User-Ref and Data-Assoc (short for Data Association). User-Ref refers a user category (<user>) to specify applicable user categories. Data-Assoc specifies applicable association of data categories by a sequence of Data-Refs. Each Data-Ref refers a data category (<data>) with an ac-tion. An action specifies how a data category is accessed. We distinguish two types of actions since they have different privacy implications. The action projection means that data categories are output by a query. The action condition denotes that data categories are used in the control flow of a query. The actions also form a hierarchy where access is the parent of projection and condition. For Data-Assoc, data categories referred in each Data-Ref must be disjoint. That is, no referred data category is the parent of another.

A rule can forbid the query or specify desensitize require-ments by restrictions. A restriction is a sequence of Desensi-tize with the same length as the data association of the rule. Each Desensitize specifies a set of admissible desensitize opera-tions (<operation>) required for the corresponding data category (0 denotes no desensitization requirement). The corresponding data category must support the specified desensitize operations as defined in vocabulary. A query must satisfy one of the restrictions in an applicable rule.

Example 2.1. Consider the privacy policy of a retail company. First, the company forbids any access to customer names by Analyst. We use the following PSpec rule:

\[ r_1: \text{Analyst, } [\text{access Name}] \Rightarrow \text{forbid} \]

Second, the company further requires the sales data should be aggregated when outputting with personal information.

\[ r_2: \text{Analyst, } [\text{projection All exclude Sensitive Attribute, projection Price}] \Rightarrow \{[\text{min}, \text{max}, \text{avg}, \text{sum}] \} \]

In \( r_2 \), we define a data association of length two. The first data access refers to the projection on all but sensitive data categories. The second data access refers to the projection on sales price. The rule \( r_2 \) allows direct access to all but sensitive data but requires Price to be aggregated by one of the avg, max, min, or sum desensitize operations.

Finally, the company forbids any access to quasi-identifiers such as birth, gender, and zip together [28].

\[ r_3: \text{Analyst, } [\text{access Birth, access Gender, access Zip}] \Rightarrow \text{forbid} \]

Note that \( r_3 \) only forbids access to them together. One can access any one or two of them freely.

2.3 Discussion

A data category is desensitized when one of the specified desensitize operations is performed. We do not support se-quences of desensitize operations since it would complicate PSpec rules and queries. To enforce sequences of desensitize operations, the data owner can define a new operation that combines all required operations. Also note that PSpec does not model access purpose directly since it would be difficult to verify the claimed purpose mechanically. We thus assume each data analyst is assigned a user category by her tasks.

PSpec is a privacy language for specifying data usage restrictions in data analytics. It is designed to be

- usable for legal teams with little technical background;
- suitable for specifying restrictions in data analytics; and
- enforceable by existing data analytics systems.

To achieve usability, PSpec is designed as a high level language based on abstract concepts to leave out details of the underlying data models and queries. To be suitable for data analytics, PSpec provides explicit support for data association and desensitization. To attain enforceability, we only
include minimal elements in PSpec, and leave out others in privacy policies like purpose, obligation, and condition. However, it is straightforward to extend PSpec with these elements if necessary. To enforce the PSpec policy within a data analytics system, one simply needs to analyze each submitted query to identify how each data category is accessed with what operation, and check the compliance according to the semantics discussed in the next section.

3. Formal Semantics

In this section, we present the formal semantics of PSpec, which lay the foundation for policy enforcement and analysis. Recall that PSpec specifies restrictions on data analytics, thus the semantics are defined against queries.

3.1 Rule

Recall that on the data owner’s side, a hierarchy is used to organize each of the user, data, and action categories. A policy rule may refer to internal nodes in hierarchies. In the following, we use the concepts associated with leaf categories of these hierarchies to define the semantics of a rule.

Let \( U_{\text{user}}, U_{\text{data}} \) and \( U_{\text{act}} \) be the universal set of all user categories, data categories, and action categories referred by leaf categories of the user hierarchy, the data hierarchy and the action hierarchy, respectively. We use \( u \in U_{\text{user}}, \ d \in U_{\text{data}}, \ a \in U_{\text{act}} \) to denote a leaf concept of the user, data, and action categories, respectively. Especially, we denote the pair \( \langle d, a \rangle \) a data access. Let \( U_{da} = U_{data} \times U_{act} \) be the set of all data accesses with respect to \( U_{data} \) and \( U_{act} \). An associated data access \( \tau \) of size \( n \) is an \( n \)-tuple \( \tau = (\tau_1, \tau_2, \ldots, \tau_n) \), where \( \tau_i \in U_{da} \) for \( 1 \leq i \leq n \).

The associated data accesses referred by a rule \( r \) must have the same size, denoted as \( |r| \). Denote \( U_{da}^{|r|} \) the set of all associated data accesses of length \( |r| \).

Given a rule \( r \), we use \( r.\text{User} \subseteq U_{\text{user}} \) and \( r.\text{Asso} \subseteq U_{da}^{|r|} \) to denote the set of user categories and the set of associated data accesses applicable to \( r \), respectively. In the grammar of Fig. 2, \( r.\text{User} \) is specified by User-Ref. It corresponds to the leaves of the subtree rooted in the specified data category minus the leaves of the subtree rooted in any of the excluded data categories. The set \( r.\text{Asso} \) is specified by Data-Assoc in the grammar. Let \( r.\text{Asso} \) be the set of data accesses specified by the \( i \)-th Data-Ref of Data-Assoc, \( r.\text{Asso}_i \) be the Cartesian product of \( r.\text{Asso} \) for \( 1 \leq i \leq n \).

Let \( U_{op} \) be the universal set of all available desensitize operations. We use \( \Delta \subseteq U_{op} \) to denote a set of desensitize operations, and \( \delta \in U_{op} \) a desensitize operation. The empty desensitize operation is denoted by \( \delta_0 \in U_{op} \). Given an associated data access \( \tau = (\langle d_1, a_1 \rangle, \langle d_2, a_2 \rangle, \ldots, \langle d_n, a_n \rangle) \) of size \( n \), a restriction \( \xi \) is an \( n \)-tuple \( (\Delta_1, \Delta_2, \ldots, \Delta_n) \), where \( \Delta_i \subseteq U_{op} \) specifies the set of desensitize operations. An operation in \( \Delta_i \) must be taken to protect \( d_i \) against \( a_i \). Without loss of generality, we assume the rule \( r \) contains \( m \) restrictions.

Given a rule \( r \), we use \( r.\text{Res} \) to denote the set of restrictions specified in \( r \). Each restriction \( \xi_i \in r.\text{Res} \) is specified by Restriction_\text{i} in the grammar.

Definition 3.1 (Rule). A rule \( r \) is a triple \( r = (r.\text{User}, r.\text{Asso}, r.\text{Res}) \), where

- \( r.\text{User} \subseteq U_{\text{user}} \) is the set of user categories,
- \( r.\text{Asso} \subseteq U_{da}^{|r|} \) is the set of associated data accesses, and
- \( r.\text{Res} = \{\xi_1, \xi_2, \ldots, \xi_m\} \) is the set of restrictions. Especially, \( r.\text{Res} = \emptyset \) if \( r \) is a forbidden rule.

In the grammar of Fig. 2, \( r.\text{User} \) is specified by User-Ref. It corresponds to the leaves of the subtree rooted in the specified data category minus the leaves of the subtree rooted in any of the excluded data categories. The set \( r.\text{Asso} \) is specified by Data-Assoc in the grammar. Let \( r.\text{Asso}_i \) be the set of data accesses specified by the \( i \)-th Data-Ref of Data-Assoc, \( r.\text{Asso}_i \) be the Cartesian product of \( r.\text{Asso} \) for \( 1 \leq i \leq n \).

Let \( U_{op} \) be the universal set of all available desensitize operations. We use \( \Delta \subseteq U_{op} \) to denote a set of desensitize operations, and \( \delta \in U_{op} \) a desensitize operation. The empty desensitize operation is denoted by \( \delta_0 \in U_{op} \). Given an associated data access \( \tau = (\langle d_1, a_1 \rangle, \langle d_2, a_2 \rangle, \ldots, \langle d_n, a_n \rangle) \) of size \( n \), a restriction \( \xi \) is an \( n \)-tuple \( (\Delta_1, \Delta_2, \ldots, \Delta_n) \), where \( \Delta_i \subseteq U_{op} \) specifies the set of desensitize operations. An operation in \( \Delta_i \) must be taken to protect \( d_i \) against \( a_i \). Without loss of generality, we assume the rule \( r \) contains \( m \) restrictions.

Given a rule \( r \), we use \( r.\text{Res} \) to denote the set of restrictions specified in \( r \). Each restriction \( \xi_i \in r.\text{Res} \) is specified by Restriction_\text{i} in the grammar.

Example 3.1. Consider the following rule:

\[ r_4: \text{Report Analyst, [projection Address,projection Price]} \Rightarrow \{(), (sum, avg, min, max)\} \]
Obviously, \( |r_4| = 2 \), and

\[
\begin{align*}
  r_4.User &= \{ \text{Report Analyst} \} \\
  r_4.Aso &= \{ (\text{State, projection}), (\text{Price, projection}), \\
  &\quad (\text{City, projection}), (\text{Price, projection}), \\
  &\quad (\text{Street, projection}), (\text{Price, projection}) \} \\
  r_4.Res &= \{ \{ \{ \cup, \{ \text{sum, avg, min, max} \} \} \} \\
\end{align*}
\]

Note that \( \cup \) is used if there is no desensitize requirement.

### 3.2 Query

Similar as for a rule, we use the leaf categories to define the semantics of a data analytics query (or a query for short). This is not a limitation in practice. Non-leaf categories can be easily decomposed into a set of leaf categories.

The semantics of a query is defined from the information flow point of view [25, 26]. A query \( q \) is divided into a set of data leakages. A data leakage is a pair \( ((d, a), \delta) \) where \( (d, a) \) is a data access and \( \delta \) is a desensitize operation. A data leakage \( ((d, a), \delta) \) means that the data \( d \) is leaked through the action channel \( a \) after the desensitize operation \( \delta \). Let \( |q| \) be the number of data leakages in \( q \). We extract the data accesses and desensitize operations from the data leakages of \( q \), each as a \( |q| \)-tuple.

**Definition 3.2 (Query).** A query \( q \) is a triple \( q = (q.user, q.asso, q.des) \), where

- \( q.user \in \cup_user \) is the user who submits \( q \),
- \( q.asso \in \cup_{asso}^{|q|} \) is the associated data accesses in \( q \), and
- \( q.des = (\delta_1, \delta_2, \ldots, \delta_{|q|}) \) is the desensitize operations.

**Example 3.2.** Consider a report analyst submits a query:

\[
q: \text{SELECT State, City, avg(Price) from t_addr}
\]

where “t_addr” is a table name. Obviously, \( |q| = 3 \), and

\[
\begin{align*}
  q.user &= \text{Report Analyst}, \\
  q.asso &= (\text{State, projection}), (\text{City, projection}), \\
  &\quad (\text{Price, projection}), \\
  q.des &= (\delta_0, \delta_0, \text{avg}).
\end{align*}
\]

Note that \( \delta_0 \) is used if one wants to access data without any desensitize.

**Definition 3.3.** Given two associated data accesses \( \tau \) and \( \tau' \), we say \( \tau \) is included in \( \tau' \) (written \( \tau \preceq \tau' \)) if all data accesses contained in \( \tau \) are also contained in \( \tau' \).

If we treat both \( \tau \) and \( \tau' \) as a set, \( \tau \preceq \tau' \) just means \( \tau \) is a subset of \( \tau' \). Note that a smaller-size associated data access usually means larger scope.

**Definition 3.4.** A rule \( r \) is applicable to a query \( q \) (or equivalently, \( q \) triggers \( r \)) if

- \( q.user \in r.User \), and
- there exists at least one associated data access \( \tau \) in \( r.Aso \) such that \( \tau \preceq q.asso \).

The first condition in above definition performs user scope check, and the second condition performs the scope check of associated data access. The rule is applicable to the query only if both checks are passed. For the ease of following discussion, we say a user category \( u \) triggers a rule \( r \) if \( u \in r.User \), and an associated data access \( \tau \) triggers \( r \) if there exists some \( \tau_r \in r.Aso \) such that \( \tau_r \preceq \tau \).

Moreover, if there exists an \( \tau \in r.Aso \) such that the second condition holds, we further define \( q.des|\tau \) as the projection of \( q.des \) to \( \tau \), obtained by removing the desensitize operations in \( q.des \) whose corresponding data access does not occur in \( \tau \). The size of the tuple \( q.des|\tau \) equal to \( |r| \).

Note that there may be more than one \( \tau \) satisfying the second condition holds. With different \( \tau \), the projection \( q.des|\tau \) may be different. We thus define

\[
q.des|_\tau = \{ q.des|\tau \mid \tau \in r.Aso \text{ and } \tau \preceq q.asso \}
\]

**Definition 3.5.** A \( n \)-tuple of desensitize operations \( \bar{\delta} = (\delta_1, \delta_2, \ldots, \delta_n) \) satisfies a restriction \( \bar{\xi} = (\Delta_1, \Delta_2, \ldots, \Delta_n) \) if \( \delta_i \in \Delta_i \) for \( 1 \leq i \leq n \).

**Definition 3.6 (Rule Satisfaction).** A query \( q \) satisfies a rule \( r \) iff either \( r \) is not applicable to \( q \) or for any \( \delta \in q.des|_r \), there exists at least one restriction \( \xi \in r.Res \), such that \( \delta \) satisfies \( \xi \).

Intuitively, a query satisfies a triggered rule if the data categories accessed by the query are properly desensitized.

**Example 3.3.** Consider the rule \( r_4 \) in Example 3.1 and the query \( q \) in Example 3.2. \( r_4 \) is applicable to \( q \) since \( q.user \in r_4.User \) and two former associated data accesses in \( r_4.Aso \) are included in \( q.asso \). Hence \( q.des|_r \in \{ \{ \delta_0, \text{avg} \} \} \). Since \( \delta_0, \text{avg} \) satisfies the constraint in \( r_4.Res, r_4 \) is satisfied by \( q \) (Definition 3.6).

**Definition 3.7 (Policy Satisfaction).** A query \( q \) satisfies a PSpec policy iff \( q \) satisfies all rules of the policy.

### 3.3 Symbolic Encoding

Since our policy analysis algorithms heavily rely on SMT solving techniques, we present in this subsection an symbolic encoding scheme for PSpec rules.

We use a tuple of \( |r| \) variables \( \{ x_r^1, x_r^2, \ldots, x_r^{|r|} \} \) to denote the associated data accesses of a rule \( r \), where \( x_r^i \in U_{op} \) is called a rule variable for \( 1 \leq i \leq |r| \).

**Definition 3.8.** Let \( X_r = \{ x_r^1, x_r^2, \ldots, x_r^{|r|} \} \) be the set of rule variables of \( r \). A restriction \( \bar{\xi} = (\Delta_1, \Delta_2, \ldots, \Delta_{|r|}) \) in \( r \) can be symbolically encoded as a logical formula over \( X_r \)

\[
\phi_{\bar{\xi}} = \bigwedge_{1 \leq i \leq |r|} (x_r^i \in \Delta_i)
\]
The restriction formula of a rule \( r \) is

\[
\phi_r = \bigvee_{\delta \in r.\text{Res}} \phi_\delta
\]

Especially, \( \phi_r = \text{false} \), if \( r \) is a forbidden rule.

Note that a tuple \( \delta \) of desensitize operations of length \(|r|\)

\( \) can be considered as a valuation to the variables \( X_r \). Then we have the following lemma.

**Lemma 1.** A tuple of desensitize operations \( \delta \) satisfies one of the restrictions in a rule \( r \) if \( \phi_r(\delta) \) evaluates to true.

**Example 3.4.** Consider the rule \( r_4 \) in Example 3.1, where \(|r_4| = 2\). Let \( x^1_r, x^2_r \) be rule variables of \( r_4 \). The logical formula of \( r_4 \) is

\[
\phi_{r_4} = (x^1_r \in \mathbb{U}_{op}) \land (x^2_r \in \{\text{sum}, \text{avg}, \text{min}, \text{max}\}) = x^2_r \in \{\text{sum}, \text{avg}, \text{min}, \text{max}\}
\]

The desensitize operations executed by the query \( q \) in Example 3.2 is \((\delta_0, \text{avg})\). It does satisfy the constraint, since \( \phi_{r_4}(\delta_0, \text{avg}) = \text{true} \).

### 4. Policy Analysis

After introducing the formal semantics, we discuss some useful policy analysis algorithms for PSpec, namely redundancy analysis and consistency analysis.

#### 4.1 Redundancy Analysis

Since a query should satisfy all rules in a PSpec policy, some rules may thus be redundant. The redundant rules not only cost unnecessary time for policy enforcement, but also may represent potential errors or unintended side-effects in a policy [1]. In this subsection, we discuss redundancy analysis for the PSpec policy.

**Definition 4.1** (Rule Redundancy). A rule \( r' \) is said redundant with respect to another rule \( r \) if for any query \( q \) such that \( q \) satisfies \( r \), \( q \) also satisfies \( r' \).

With the inclusion relation between associated data accesses (Definition 3.3), we define a similar relation between the scopes of two rules. Note again that a less-size associated data access usually means larger scope.

**Definition 4.2.** Given two rules \( r \) and \( r' \), we say the scope of \( r \) subsumes the scope of \( r' \), denoted as \( \text{scope}(r) \supseteq \text{scope}(r') \), if \( r.' \text{User} \subseteq r. \text{User} \), and

\[
\forall \tau' \in r.' \text{Asso}, \exists \tau \in r. \text{Asso}, \text{s.t. } \tau \subseteq \tau'.
\]

**Lemma 2.** Given two rules \( r \) and \( r' \) with \( \text{scope}(r) \supseteq \text{scope}(r') \), we have \(|r| \leq |r'|\); and for any query \( q \), if \( r' \) is triggered, \( r \) is also triggered.

Moreover, we can further facilitate the check of scope inclusion with the following lemma:

**Lemma 3.** Given two rules \( r \) and \( r' \) such that \( r.' \text{User} \subseteq r. \text{User} \), \( \text{scope}(r) \supseteq \text{scope}(r') \) iff

\[
\forall 1 \leq i \leq |r|, \exists 1 \leq j \leq |r'|, \text{ s.t. } r. \text{Asso}_i \supseteq r.' \text{Asso}_j.
\]

**Example 4.1.** Consider \( r_4 \) in Example 3.1 and the following rule:

\[
r_5: \text{Analyst}, \text{[Access Price]} \Rightarrow \{\text{avg, sum}\}
\]

We have \(|r_5| = 1\), and

\[
\]

Comparing \( r_5 \) to \( r_4 \), we have \( r_4. \text{User} \subseteq r_5. \text{User} \); and for all associated data accesses \( \tau' \) in \( r_4. \text{Asso} \), there exists an associated data access \( \tau = \{\text{(Price, projection)}\} \) in \( r_5. \text{Asso} \), such that \( \tau \subseteq \tau' \). Thus \( \text{scope}(r_5) \supseteq \text{scope}(r_4) \) (Definition 4.2). Obviously, whenever Report Analyst outputs Address and Price together \( (r_4 \) is triggered), \( r_5 \) must be triggered as well.

Definition 4.2 makes it possible to compare the scopes of two rules. We now compare the restrictions of two rules. The basic idea is to check their restriction formulas to see if one restriction implies another one. The hardness is that these two formulas are defined over two sets of rule variables. We thus need a mechanism to link these variables together.

**Definition 4.3.** Let \( r \) and \( r' \) be two rules, \( X_r \) and \( X_{r'} \) their sets of rule variables, \( \phi_r \) and \( \phi_{r'} \) their restriction formulas, respectively. For any variable \( x^i_r \in X_r \) \((1 \leq i \leq |r|)\) and \( x^j_{r'} \in X_{r'} \) \((1 \leq j \leq |r'|)\), we say \( x^i_r \) matches \( x^j_{r'} \) iff for any \( \tau \in r. \text{Asso} \), there exists \( \tau' \in r'. \text{Asso} \), such that \( \tau'[j] = \tau[i] \), i.e., \( r. \text{Asso} \subseteq r'. \text{Asso} \). We denote all matched variables in \( r' \) for \( x^i_r \) as matched\( (x^i_r, r') \).

**Definition 4.4.** Given two rules \( r \) and \( r' \) that \( \text{scope}(r) \supseteq \text{scope}(r') \), a variable mapping from \( r \) to \( r' \) is a function \( \theta_{r'}: X_r \rightarrow X_{r'} \) such that \( \forall x^i_r \in X_r, \theta_{r'}(x^i_r) \in \text{matched}(x^i_r, r') \).

Note that the mapping from \( X_r \) to \( X_{r'} \) is possible since \(|r| \leq |r'|\). As a convention, we denote all possible variable mappings from \( r \) to \( r' \) as \( \Theta_{r'} \). Let \( \phi_r \) be the restriction formula of \( r \), we denote \( \theta_{r'}(\phi_r) \) the formula obtained by replacing each \( x^i_r \) in \( \phi_r \) with \( \theta_{r'}(x^i_r) \) for \( 1 \leq i \leq |r| \).

**Definition 4.5** (Rule Implication). Given two rules \( r \) and \( r' \), we say \( r \) implies \( r' \) (denoted as \( r \implies r' \)), if

- \( \text{scope}(r) \supseteq \text{scope}(r') \), and
- \( \forall x^1_{r'}, \ldots, x^{l'_{r'}} \in X_{r'} \) \((\bigwedge_{\theta_{r'} \in \Theta_{r'}} (\theta_{r'}(\phi_r)) \rightarrow \phi_{r'} \).

The following theorems state some properties of rule implication.

**Theorem 1.** Implication relation over rules is a partial order.
Theorem 2. A rule \( r' \) is redundant with respect to another rule \( r \) iff \( r \implies r' \).

Example 4.2. Consider the rules \( r_4 \) and \( r_5 \) in Example 4.1. From previous discussion, we already know \( \text{scope}(r_5) \supseteq \text{scope}(r_4) \). Let \( X_{r_5} = \{ x_1^{r_5} \} \) and \( X_{r_4} = \{ x_1^{r_4}, x_2^{r_4} \} \), the restriction formulas of \( r_5 \) and \( r_4 \) are

\[
\phi_{r_5} = x_1^{r_5} \in \{ \text{avg}, \text{sum} \} \\
\phi_{r_4} = x_2^{r_4} \in \{ \text{avg}, \text{sum}, \text{min}, \text{max} \}
\]

Observing \( r_5 \).\text{Asso} and \( r_4 \).\text{Asso}, we know that \( x_1^{r_5} \) matches only \( x_2^{r_4} \). The only variable mapping is \( \theta_{r_4}^{r_5} = \{ x_1^{r_5} \rightarrow x_2^{r_4} \} \). Thus, to check whether \( r_5 \implies r_4 \), it suffices to check the following formula:

\[
\forall x_1^{r_4}, x_2^{r_4}, x_2^{r_5} \in \{ \text{avg}, \text{sum} \} \rightarrow x_2^{r_5} \in \{ \text{avg}, \text{sum}, \text{min}, \text{max} \}
\]

Obviously, the above formula is true, and thus we have \( r_5 \implies r_4 \). From Theorem 2, we know \( r_4 \) is redundant w.r.t. \( r_5 \).

With the definitions above, it is straightforward to derive a redundancy analysis algorithm for the PSpec policy. Specifically, for each pair of rules \( r \) and \( r' \), if \( r' \implies r \) then \( r \) is reported redundant with respect to \( r' \). Due to its simplicity, the details are omitted here.

4.2 Consistency Analysis

After discussing redundancy analysis for PSpec, we now introduce consistency analysis. Since we require all rules in a policy be satisfied, it is thus possible that some rules can never be satisfied simultaneously, which leads to inconsistency.

4.2.1 Consistency Definition

Data association makes the consistency analysis much more subtle and complicated. Traditional consistency considers privacy rules with only one data category. With this traditional definition, two rules are inconsistent if they can be triggered by a same query, while their restrictions cannot be satisfied simultaneously. To directly extend the classical consistency definition to analysis with data associations, we have the following definition.

Definition 4.6 (Weak Inconsistency). Two rules \( r \) and \( r' \) are weak inconsistent if there exist a user category \( u \) and an associated data access \( \pi \) such that \( \pi \) is inaccessible for \( u \) w.r.t. \( r \) and \( r' \), i.e., there is no query \( q \) such that \( q.user = u \), \( q.asso = \pi \) and \( q \) satisfies both \( r \) and \( r' \).

This weak inconsistency definition is problematic for privacy language. Consider following two rules:

\[
\begin{align*}
\text{rg}: \text{Analyst, [project Zip, access Price]} & \Rightarrow \{\} \cdot \{\text{min, max}\} \\
\text{rr}: \text{Analyst, [access Birth, access Price]} & \Rightarrow \{\} \cdot \{\text{avg}\}
\end{align*}
\]

Clearly, when \( \text{Analyst} \) accesses \( \text{Zip, Birth} \) and \( \text{Price} \) together, there is no way to satisfy \( \text{rg} \) and \( \text{rr} \) simultaneously. But in practice, this may just be the user’s intention. Note that data associations with various length usually have different privacy implications. From the point view of privacy protection, if a query attempts to access more data categories, it turns to be more dangerous. Thus the data owner may specify above two rules on purpose to forbid the simultaneous access of these three data categories.

The weak inconsistency definition also has efficiency problem. To decide whether two rules are weak inconsistent, one needs to check not only the associated data accesses in the scope of these rules, but also others (for example, the associated access to \( \text{Address, Birth} \) and \( \text{Price} \) in above example). Given \( N \) data categories and \( M \) (\( M = 2 \) PSpec) action categories, the total number of different associated data access is as large as \( 2^N \times M \). The efficiency problem is further worsened when we consider the inconsistency of a set of, rather than two, rules. Thus, from the implementation point of view, weak inconsistency can lead to a very inefficient analysis algorithm.

We propose a novel inconsistency definition for analysis with data association. The basic idea is to predetermined a seed rule, and then define the consistency with respect to this seed rule.

Definition 4.7. A rule \( r \) is inconsistent with respect to an non-forbidden rule \( r_s \) if there exist a user category \( u_s \in r_s \).\text{User} and an associated data access \( \pi_s \in r_s \).\text{Asso} such that \( \pi_s \) is inaccessible for \( u_s \) w.r.t. \( r_s \) and \( r \). The rule \( r_s \) is called the seed rule.

The above definition can be naturally extended to sets.

Definition 4.8 (Inconsistency). A rule set \( R \) is inconsistent w.r.t. \( r_s \) if there exist a user category \( u_s \in r_s \).\text{User} and an associated data access \( \pi_s \in r_s \).\text{Asso} such that \( \pi_s \) is inaccessible for \( u_s \) w.r.t. the rules in \( R \cup \{r_s\} \).

Intuitively, for a non-forbidden rule \( r_s \), the defined data accesses \( \delta_s \in r_s \).\text{Asso} should be accessible if properly desensitized. However, accessing \( \delta_s \) could trigger other rules, which may in turn render \( \delta_s \) inaccessible and lead to inconsistency. In practice, we are only interested in the minimal inconsistent rule set as follows.

Definition 4.9 (Minimal Inconsistency). A rule set \( R \) is minimally inconsistent w.r.t. \( r_s \) if:

1. \( R \) is inconsistent w.r.t. \( r_s \), and
2. \( \forall R' \subseteq R, R' \) is consistent w.r.t. \( r_s \).

Example 4.3. Consider the rules \( r_6 \) and \( r_7 \). No matter \( r_6 \) or \( r_7 \) being the seed rule, there is no associated data access in the seed rule that triggers another rule. Rules \( r_6 \) and \( r_7 \) are thus consistent.

4.2.2 Consistency Analysis as Satisfiability Problems

In this subsection, we formulate the consistency analysis as satisfiability problems.
Definition 4.10 (Candidate Rule). Given a seed rule \( r_s \), we say a rule \( r \) is a candidate rule of \( r_s \) if there exist a user category \( u_s \in r_s.User \) and an associated data access \( \delta_s \in r_s.Asso \) such that both \( u_s \) and \( \delta_s \) trigger \( r \). Denote \( cand(r_s) \) the set of all candidate rules of \( r_s \).

Apparently, for any minimal inconsistent rule set \( R \) w.r.t. a seed rule \( r_s \), \( R \subseteq cand(r_s) \).

Definition 4.11. Given a seed rule \( r_s \) and a non-forbidden rule set \( R \subseteq cand(r_s) \), \( trig_{\delta}^6(r_s, R) \) is the set of associated data accesses in \( r_s \) that trigger all rules in \( R \).

Let \( r_s \) be a seed rule, and \( r \) a rule such that \( r \in cand(r_s) \). Let \( X_r \) and \( X_{r_s} \) be the set of rule variables of \( r \) and \( r_s \), respectively. If an associated data access \( \tau_s \) of \( r_s \) triggers \( r \), there must exist an associated data access \( \tau \) of \( r \), such that \( \tau \subseteq \tau_s \) (Definition 3.4). Then we can construct an interval mapping \( I^{\tau}_{\tau_s} \) from \[1, |r|\] to \[1, |r_s|\] as: \( \forall i \in [1, |r|], I^{\tau}_{\tau_s}[i] = j \in [1, |r_s|] \) such that \( \tau[i] = \tau_s[j] \).

Definition 4.12. Let \( \tau \in r.Asso \) and \( \tau_s \in r_s.Asso \) be two associated data accesses that satisfy \( \tau \subseteq \tau_s \), a variable mapping with respect to \( \tau \) and \( \tau_s \) is a function \( \varphi^{\tau}_{\tau_s} : X_r \to X_{r_s} \) such that \( \forall x_r^{i} \in X_r, \varphi^{\tau}_{\tau_s}(x_r^{i}) = x_{r_s}^{j} \) with \( j = I^{\tau}_{\tau_s}(i) \).

Let \( \phi_r \) be the restriction formula of \( r \), we denote \( \varphi^{\tau}_{\tau_s}(\phi_r) \) the formula obtained by replacing \( x_r^{i} \) with \( \varphi^{\tau}_{\tau_s}(x_r^{i}) \) for \( 1 \leq i \leq |r| \).

Definition 4.13. The association formula for an associated data access \( \tau_s \in trig_{\delta}^6(r_s, R) \) is

\[
\psi_{\tau_s} = \phi_{r_s} \land \bigwedge_{\forall \tau \in r.Asso} \bigwedge_{\forall \tau_s \in r_s.Asso} \varphi^{\tau}_{\tau_s}(\phi_r)
\]

where \( \phi_{r_s} \) and \( \phi_r \) are restriction formulas of \( r_s \) and \( r \), respectively, \( \varphi^{\tau}_{\tau_s} \) is the variable mapping with respect to \( \tau \) and \( \tau_s \).

If \( \psi_{\tau_s} \) is unsatisfiable, \( \tau_s \) is just an associated data access that is inaccessible w.r.t. the rules in \( R \cup \{r_s\} \). In other words, if \( \psi_{\tau_s} \) is unsatisfiable, the rule set \( R \) is inconsistent with respect to \( r_s \) (Definition 4.8).

Theorem 3. Given a seed rule \( r_s \) and a non-forbidden rule set \( R \subseteq cand(r_s) \), the rule set \( R \) is minimal inconsistent w.r.t. \( r_s \) iff the following holds:

- \( r_s.User \cap \bigcap_{\tau \in R} \tau.User \neq \emptyset \),
- there exists \( \tau_s \in trig_{\delta}^6(r_s, R) \) such that its association formula \( \psi_{\tau_s} \) is unsatisfiable, and
- \( \forall \tau \subseteq \tau_s, \tau_s \) is consistent w.r.t. \( r_s \).

Remark that \( \tau_s \in trig_{\delta}^6(r_s, R) \) in the second condition is important. If \( \tau_s \notin trig_{\delta}^6(r_s, R) \), there must exist a rule \( r \) in \( R \) that is not triggered by \( \tau_s \). Let \( R' = R \setminus \{r\} \). Then from the unsatisfiability of \( \psi_{\tau_s} \), we can never conclude that \( R \) is minimal inconsistent (since \( R' \) is inconsistent and \( R' \subseteq R \)).

Corollary 1. Given a seed rule \( r_s \) and a non-forbidden rule set \( R \subseteq cand(r_s) \), if either \( r_s.User \cap \bigcap_{\tau \in R} \tau.User \neq \emptyset \), or \( trig_{\delta}^6(r_s, R) = \emptyset \), then the rule set \( R \) and all its supersets cannot be minimal inconsistent w.r.t. \( r_s \).

Example 4.4. Consider the rule \( r_6 \) and the following rule:

\[
\begin{align*}
r_8 & \colon \text{Analyst, } \{\text{projection Price}\} \Rightarrow \{\text{avg, sum}\}
\end{align*}
\]

where \( r_6 \) is set as the seed rule. Obviously, \( r_6.User \cap r_8.User \neq \emptyset \). Let \( u_s \) be any user category in \( r_6.User \cap r_8.User \), and \( \tau_s = \langle \{\text{Zip, projection}\}, \{\text{Price, projection}\} \rangle \) in \( r_8.Asso \). The only associated data access \( \tau = \langle \{\text{Price, projection}\} \rangle \) in \( r_8.Asso \) satisfies \( \tau \leq \tau_s \). By Definition 3.4, \( u_s \) and \( \tau_s \) together trigger \( r_8 \). Thus \( r_8 \) is a candidate rule of \( r_6 \).

Let \( x_{r_6}^1, x_{r_6}^2 \) be rule variables of \( r_6 \), and \( x_{r_8}^1 \) the rule variable of \( r_8 \). Apparently,

\[
\begin{align*}
\phi_{r_6} & = \text{true} \land (x_{r_6}^1 \in \{\text{min, max}\}) \\
\phi_{r_8} & = x_{r_8}^1 \in \{\text{avg, sum}\}
\end{align*}
\]

Moreover, \( trig_{\delta}^6(r_6, \{r_8\}) = \{\tau_s\} \). The variable mapping w.r.t. \( \tau \) and \( \tau_s \) is \( \varphi^{\tau}_{\tau_s} = x_{r_6}^1 \to x_{r_8}^1 \}. \) The association formula for \( \tau_s \) is

\[
\psi_{\tau} = (\text{true} \land (x_{r_6}^1 \in \{\text{min, max}\}) \land (x_{r_6}^2 \in \{\text{avg, sum}\})
\]

Apparently \( \psi_{\tau_s} \) is unsatisfiable, thus the rule set \( \{r_8\} \) is minimal inconsistent w.r.t. \( r_6 \).

4.2.3 Analysis Algorithm

The purpose of inconsistency analysis on a policy is to find all minimal inconsistency rule sets w.r.t. each non-forbidden rule (as the seed rule) in the policy.

According to Definition 4.9, to find all minimal inconsistent rule sets w.r.t. a seed rule \( r_s \), it suffices to only consider the subsets of the candidate rules \( cand(r_s) \) of \( r_s \). Moreover, all subsets of the candidate rules \( cand(r_s) \) constitute a set containment lattice. An example lattice of four candidate rules is shown in Fig. 3a, where each node represents a subset of these candidate rules. We apply the levelwise search technique [18] to find in this lattice all minimal inconsistent rule sets. The efficiency of the levelwise search process relies on the search space pruning. For example, if the rule set \( \{r_1, r_2\} \) (denoted as “12” in the figure) is pruned (Fig. 3b), all its super sets are pruned from the search as well, which greatly reduces the search efforts.

Our levelwise search algorithm for finding all minimal inconsistency rule sets is shown in Fig. 4. The function LEVELWISESEARCH takes as inputs a seed rule \( r_s \) and a set \( R \) of all candidate rules. The algorithm divides the search space (i.e. all subsets of \( R \)) into levels, and iteratively processes each level.

The variable \( L_l \) stores the subsets of \( R \) at the \( l \)-th level. For each subset \( R_l \) in \( L_l \), the algorithm checks whether \( R_l \) is minimal inconsistent (line 6) using Theorem 3. According
1) as follows: GENERATE is initialized as a set (using G next level. Following the levelwise search framework, tent or not. The algorithm then continues its search in the tent. And note that the minimality of \( R_l \) is automatically guaranteed by the nature of levelwise search, i.e., if any subset \( R'_l \subseteq R_l \) is minimal inconsistent, then \( R'_l \) (and all its supersets) would be pruned. Otherwise, the algorithm checks if \( R_l \) is prunable (line 8) using Corollary 1, i.e., whether \( R_l \) and all its supersets cannot be minimal inconsistent. If so, \( R_l \) is then pruned from \( L_l \).

After above two procedures, the supersets of the remaining rule sets in \( L_l \) are unknown for being minimal inconsistent or not. The algorithm then continues its search in the next level. Following the levelwise search framework, \( L_{l+1} \) is initialized as a set (using GENERATENEXTLEVEL at line 10) as follows:

\[
L_{l+1} = \{ R | |R| = l + 1 \land \forall R' \subseteq R_c (|R'| = l \rightarrow R' \in L_l) \}.
\]

5. Implementation and Evaluation

In this section, we report our implementation of PSpec. We discuss a case study on TPC-DS benchmark [21] and experiments on synthetic rules.

5.1 Implementation

We have implemented PSpec with XML. Specifically, we defined an XML schema definition (XSD) file, and implemented an XML parser with Java to parse the user-provided XML files. We have also implemented all policy analysis algorithms with Java. We use Z3 SMT solver\(^1\) for policy analysis. To help the data owner write and analyze PSpec specifications, we implemented a GUI-based policy editing tool. Our implementation is available on github\(^2\).

5.2 Case Study on TPC-DS

TPC-DS benchmark is a general benchmark for decision support systems. It models a national-wide retail company. The benchmark includes a database schema (7 fact tables and 17 dimension tables) and 99 queries. Some TPC-DS tables contain customers’ private information, such as Customer, Customer_demographics. To ensure these privacy-related data are properly used, we have made a sample PSpec policy. In the case study, we focus on privacy aspects and possible violations in TPC-DS queries.

5.2.1 Vocabulary

We assume only one user category named Analyst. The data hierarchy is shown in Fig. 5. Most data categories are self-explanatory. We use abbreviations when meanings are clear. Similar to previous work [17, 28], privacy-related information is classified as KA (Key Attribute), QI (Quasi Identifier) and SA (Sensitive Attribute). They are further divided into finer-grained categories corresponding to columns in the TPC-DS tables.

Table 1 shows supported desensitize operations. There are only a handful of desensitize operations since the schema of TPC-DS is well designed. PSpec rules can directly refer these data categories without desensitize operations.

5.2.2 PSpec Rules

We write PSpec rules to ensure privacy related data is properly used while keeping the data still useful to data analysis (Table 2). The general goal of these rules is to prevent Analyst from accessing and projecting private information for individuals. The first rule forbids the access of KA (Key Attribute) since KA can directly identify individuals. The rules 2 to 6 forbid the projection of certain data categories since these data categories may identify individuals by linking with public datasets [28]. The rules 7 to 9 further forbid any access to certain combinations of data categories since these combinations have a good chance to identify individuals. Finally, the rules 10 to 13 require some data categories must be desensi-

---

\(^1\)http://z3.codeplex.com/

\(^2\)Link to repository removed for double blind review.
5.3 Experimental Evaluation

To evaluate the performance of the policy analysis algorithms, we performed experiments on synthetic rules. All experiments are performed on a laptop with a 2.7 GHz Intel Core i5 CPU and 8GB of RAM. The maximum memory of JVM is set to 4GB. Each experiment is performed 3 times. The average time is reported.

The policies used in the experiments are randomly generated by a policy generator as follows. We first randomly generate a vocabulary containing $N_u$ user categories and $N_d$ data categories. Both of them are organized as three-level hierarchies. All data categories support the same set of desensitize operations of size $N_{ds}$. We then randomly generate $N_r$ rules. For each rule, we randomly pick up a user category, $n_d$ data categories (1 ≤ $n_d$ ≤ $Max_r$), and $n_{rcs}$ restrictions (1 ≤ $n_{rcs}$ ≤ $Max_{rcs}$). Default values for these parameters are listed in Table 3.

We designed several sets of experiments to evaluate the performance of our policy analysis algorithms under different settings. In each set of experiment, only one parameter varies and others take the default values. The experimental results are shown in Table 4.

### Rules

In the first set of experiments, we change the number of rules $N_r$ to evaluate the basic performance. The number of rules has a direct impact on the performance of both algorithms. The running time for consistency analysis grows more rapidly since its complexity is exponential in the number of rules. The results also show all algorithms finish in reasonable time and hence fit for practical use.

### User and Data Categories

In the following two sets of experiments, we change $N_u$ and $N_d$ to evaluate the impact of rule overlaps. Since rules are randomly generated, fewer categories imply more scope overlaps among the randomly generated rules. Rule overlaps have a significant impact on consistency analysis since its efficiency heavily relies on search space pruning. For redundancy analysis, the impact is relatively small since its worst-case time complexity is much lower than consistency analysis.

### Data Associations

In the fourth set of experiments, we change $Max_r$ to evaluate the impact of data associations. Recall that the size of data association in each rule is randomly chosen from 1 to $Max_r$.

Increasing $Max_r$ directly affects redundancy analysis since the time for both scope inclusion test and satisfiability solving increases. For consistency analysis, the time decreases significantly with the increase of $Max_r$, since longer data associations imply fewer candidate rules for each seed rule on average.

### Desensitize Operations

In the fifth set of experiments, we change the number of desensitize operations $N_{ds}$. Increasing the number of desensitize operations has a slight impact on both algorithms since formulas become more complex and the time for satisfiability solving increases.

---

### Table 2: PSpec rules for TPC-DS

<table>
<thead>
<tr>
<th>ID</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Analyst,[access KA]=&gt;forbid</td>
</tr>
<tr>
<td>2</td>
<td>Analyst,[projection Street]=&gt;forbid</td>
</tr>
<tr>
<td>3</td>
<td>Analyst,[projection Name]=&gt;forbid</td>
</tr>
<tr>
<td>4</td>
<td>Analyst,[projection Birth,projection Address, projection Gender] =&gt;forbid</td>
</tr>
<tr>
<td>5</td>
<td>Analyst,[projection Birth,projection Address, projection Marital] =&gt;forbid</td>
</tr>
<tr>
<td>6</td>
<td>Analyst,[projection Birth,projection Address, projection Education] =&gt;forbid</td>
</tr>
<tr>
<td>7</td>
<td>Analyst,[access F_Name,access L_Name]=&gt;forbid</td>
</tr>
<tr>
<td>8</td>
<td>Analyst,[access S_Name,access S_Name, access Suite] =&gt;forbid</td>
</tr>
<tr>
<td>9</td>
<td>Analyst,[access B_Day,access B_Month, access B_Year] =&gt;forbid</td>
</tr>
<tr>
<td>10</td>
<td>Analyst,[access Zip]=&gt;[[truncate]]</td>
</tr>
<tr>
<td>11</td>
<td>Analyst,[access QI,access Vehicle]=&gt;[[],{isZero}]</td>
</tr>
<tr>
<td>12</td>
<td>Analyst,[access QI,access Vehicle]=&gt;[[],{range}]</td>
</tr>
<tr>
<td>13</td>
<td>Analyst,[access QI,access Price] =&gt;[[],{sum,count,avg,min,max}]</td>
</tr>
</tbody>
</table>

### Table 3: Experiment Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_u$</td>
<td>the number of user categories</td>
<td>15</td>
</tr>
<tr>
<td>$N_d$</td>
<td>the number of data categories</td>
<td>60</td>
</tr>
<tr>
<td>$N_r$</td>
<td>the number of rules</td>
<td>600</td>
</tr>
<tr>
<td>$N_{ds}$</td>
<td>the number of desensitize operations</td>
<td>15</td>
</tr>
<tr>
<td>$Max_r$</td>
<td>the maximum size of data association</td>
<td>5</td>
</tr>
<tr>
<td>$Max_{rcs}$</td>
<td>the maximum number of restrictions</td>
<td>5</td>
</tr>
</tbody>
</table>
In the final set of experiments, we change $Max_{res}$ to evaluate the impact of restrictions. Similar to the previous set, increasing the number of restrictions slightly impacts the running time of both algorithms since the time needed for satisfiability solving increases.

6. Related Works

Privacy Languages. Several privacy languages have been proposed to formalize text-based policies, including P3P [7], EPAL [3], XACML [19], and P-RBAC models [23]. Several works have also been proposed to formalize privacy laws, including pLogic [15], contextual integrity [5], and PrivacyLFP [8]. These works serve for general purposes. PSpec is designed as a specific language suitable for big data analytics systems with explicitly support for data association and desensitization.

Sen et al. [27] recently proposed LEGALEASE, a privacy language for encoding privacy policies in big data systems. But LEGALEASE is designed for handling general data usages, that is, store, use and share. It may not be most suitable for online data analytics. For instance, it has no flexible support for specifying desensitizations. Moreover, the work [27] lacks necessary policy analysis algorithms for redundancy and consistency. Such algorithms are useful to analyze unintended effects after a policy is drafted.

Policy Analysis. Policy analysis has been extensively studied in past decades. These works range from system configuration policies [1], firewall policies [2, 14], to access control policies [13, 16]. Although the intuition of redundancy and consistency is similar, the policy analysis algorithms in this paper are pertaining to PSpec. They are further complicated by data association.

Privacy Extensions for Programming Languages. In order to automatically enforce privacy requirements on programs, several extensions for programming languages have been proposed, including Jeeves [29] for enforcing privacy policies, and Fuzz [24] and DFuzz [11] for certifying differentially private [9] programs. There are many language extensions to enforce security policies, such as Jif [20], Joana [12], and FINE [6]. Different from these works, PSpec is a new privacy language for data analytics. It is not an extension to existing programming languages.

7. Conclusion

In this paper, we present the privacy language PSpec for specifying data usage restrictions in data analytics. To facilitate the user reason and analyze the PSpec policy, we also introduce two policy analysis algorithms, namely redundancy analysis and conflict analysis.

In the future, we plan to perform more real world case studies to further evaluate the applicability of PSpec. Moreover, we also plan to explore other policy analysis algorithms to help users write and analyze their policies.

### Table 4: Experimental Results

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Parameter</th>
<th>Redundancy</th>
<th>Consistency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rules</td>
<td>$N_r=200$</td>
<td>878ms</td>
<td>298ms</td>
</tr>
<tr>
<td></td>
<td>$N_r=400$</td>
<td>1,237ms</td>
<td>1,155ms</td>
</tr>
<tr>
<td></td>
<td>$N_r=600$</td>
<td>1,690ms</td>
<td>3,914ms</td>
</tr>
<tr>
<td></td>
<td>$N_r=800$</td>
<td>2,445ms</td>
<td>3,467ms</td>
</tr>
<tr>
<td></td>
<td>$N_r=1000$</td>
<td>4,646ms</td>
<td>14,215ms</td>
</tr>
<tr>
<td>User Categories</td>
<td>$N_u=5$</td>
<td>2,607ms</td>
<td>14,731ms</td>
</tr>
<tr>
<td></td>
<td>$N_u=10$</td>
<td>2,067ms</td>
<td>4,046ms</td>
</tr>
<tr>
<td></td>
<td>$N_u=15$</td>
<td>2,113ms</td>
<td>4,596ms</td>
</tr>
<tr>
<td></td>
<td>$N_u=20$</td>
<td>2,043ms</td>
<td>1,726ms</td>
</tr>
<tr>
<td></td>
<td>$N_u=25$</td>
<td>2,426ms</td>
<td>1,923ms</td>
</tr>
<tr>
<td>Data Categories</td>
<td>$N_d=20$</td>
<td>2,628ms</td>
<td>7,597ms</td>
</tr>
<tr>
<td></td>
<td>$N_d=40$</td>
<td>4,348ms</td>
<td>4,931ms</td>
</tr>
<tr>
<td></td>
<td>$N_d=60$</td>
<td>3,015ms</td>
<td>3,269ms</td>
</tr>
<tr>
<td></td>
<td>$N_d=80$</td>
<td>5,763ms</td>
<td>2,127ms</td>
</tr>
<tr>
<td></td>
<td>$N_d=100$</td>
<td>1,980ms</td>
<td>1,715ms</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Parameter</th>
<th>Redundancy</th>
<th>Consistency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$Max_{r}=1$</td>
<td>40ms</td>
<td>48,018ms</td>
</tr>
<tr>
<td></td>
<td>$Max_{r}=3$</td>
<td>2,132ms</td>
<td>8,772ms</td>
</tr>
<tr>
<td></td>
<td>$Max_{r}=5$</td>
<td>2,564ms</td>
<td>3,004ms</td>
</tr>
<tr>
<td></td>
<td>$Max_{r}=7$</td>
<td>5,680ms</td>
<td>1,557ms</td>
</tr>
<tr>
<td></td>
<td>$Max_{r}=9$</td>
<td>4,560ms</td>
<td>2,243ms</td>
</tr>
<tr>
<td>Desensitize Operations</td>
<td>$N_{ds}=5$</td>
<td>1,594ms</td>
<td>2,923ms</td>
</tr>
<tr>
<td></td>
<td>$N_{ds}=10$</td>
<td>2,603ms</td>
<td>2,586ms</td>
</tr>
<tr>
<td></td>
<td>$N_{ds}=15$</td>
<td>2,770ms</td>
<td>4,755ms</td>
</tr>
<tr>
<td></td>
<td>$N_{ds}=20$</td>
<td>3,983ms</td>
<td>5,421ms</td>
</tr>
<tr>
<td></td>
<td>$N_{ds}=25$</td>
<td>2,296ms</td>
<td>4,697ms</td>
</tr>
<tr>
<td>Restrictions</td>
<td>$Max_{res}=1$</td>
<td>1,593ms</td>
<td>2,515ms</td>
</tr>
<tr>
<td></td>
<td>$Max_{res}=3$</td>
<td>1,109ms</td>
<td>2,692ms</td>
</tr>
<tr>
<td></td>
<td>$Max_{res}=5$</td>
<td>2,533ms</td>
<td>3,831ms</td>
</tr>
<tr>
<td></td>
<td>$Max_{res}=7$</td>
<td>3,627ms</td>
<td>3,115ms</td>
</tr>
<tr>
<td></td>
<td>$Max_{res}=9$</td>
<td>5,787ms</td>
<td>3,732ms</td>
</tr>
</tbody>
</table>
References


