A Semantics-Based Approach to Privacy Languages

Ninghui Li[*] Ting Yu[†] Annie I. Antón[‡]

Abstract

A key reason for the slow adoption of the Platform for Privacy Preferences (P3P) is the lack of a formal semantics. Without a formal semantics, a P3P policy may be semantically inconsistent and may be interpreted and represented differently by different user agents. In this paper, we redress these problems by proposing a relational formal semantics for P3P policies, which precisely models the relationships between different components of P3P statements (i.e., collected data items, purposes, recipients and retentions) during online information collection. Based on this semantics, we present SemPref, a simple, efficient and expressive semantics-based preference language. Unlike previously proposed preference languages, SemPref queries the meaning of a privacy policy rather than its syntactic representation. The proposed formal semantics and preference language are an important step towards improving P3P, making it more comprehensible to enterprises and individual users, and ultimately accelerating the large-scale adoption of P3P across the Internet.

1 Introduction

Privacy is increasingly a major concern that prevents Internet users from fully enjoying the convenience, variety and flexibility offered by e-services. Many Internet users tend to avoid websites that ask for personal information because they fear potential misuses of their private information [12]. Effective protection of individuals' privacy is in the best interest of Internet users as well as e-service providers. The W3C's Platform for Privacy Preferences Project (P3P) [23] is one major effort to improve today's online privacy practices. P3P enables websites to encode their data-collection and data-use practices in a machine-readable XML format, known as P3P policies [11]. The W3C also has a working draft on APPEL (A P3P Preference Exchange Language) [20], which allows users to specify their privacy preferences. Ideally, through the use of P3P and APPEL, a user's agent should be able to check a website's privacy policy against the user's privacy preferences, and automatically determine when the user's private information can be disclosed. In short, P3P and APPEL are designed to enable users to play an active role in controlling their private information.

After proposed, P3P has received broad attention from both industry and the research community. For example, according to [13], 21% of the top 100 most visited web sites post or in the process of post P3P privacy policies. Nevertheless, P3P adoption has been slow to date. Overall, only a small portion of websites post their P3P policies; according to W3C, as of May 2005, only 445 websites were compliant with the P3P 1.0 specification [24]. A major problem that hinders P3P adoption is that a P3P policy may be interpreted and represented differently by different user agents. Companies are thus reluctant to provide P3P policies on

[*]Department of Computer Sciences and CERIAS (Center for Education and Research in Information Assurance and Security), Purdue University, ninghui@cs.purdue.edu.

[†]Computer Science Department, North Carolina State University, yu@csc.ncsu.edu.

[‡]Computer Science Department, North Carolina State University, anton@csc.ncsu.edu.

their websites, fearing that the policies may be misrepresented [10, 22]. Quoting from CitiGroup's position paper [22], "The same P3P policy could be represented to users in ways that may be counter to each other as well as to the intent of the site." "... This results in legal and media risk for companies implementing P3P that needs to be addressed and resolved if P3P is to fulfill a very important need."

APPEL, the preference language documented in a W3C working draft [20], is also plagued with problems. For example, expressing one's privacy preferences in APPEL is a highly error-prone process. A seemingly correct APPEL privacy preference often behaves in a counterintuitive manner. The designers of APPEL even made mistakes in the APPEL specification [20]. Quoting from the position paper by the Joint Research Center (JRC) of the European Commission [14], "a preference exchange language is a very necessary part of P3P. However, there are various problems with the preference expression language (APPEL). Constructing the logic of matching patterns is very complex, and involves various inherent contradictions." (In Section 3, we give a detailed analysis of APPEL.)

Recently, work has focused on improving the usability and clarity of P3P and APPEL. Many authors have expressed various limitations of P3P and APPEL, and have made some suggested improvements, see, e.g., [3, 14, 15, 16, 21]. Agrawal et al. [3] proposed XPref, an XPath-based P3P preference language, to remedy APPEL's ambiguity and expressiveness limitations. Additionally, a variety of privacy policy tools (e.g., Privacy Bird [5] and JRC P3P Proxy [6]) have been developed, providing user-friendly interfaces for P3P and APPEL policy design. In [7], P3P designers identify several potential inconsistency problems within a P3P policy, and propose guidelines to deal with these problems. Also in [7] are some guidelines for user agents to present privacy policies. However, the previously mentioned problems have only been alleviated to a limited extent by these solutions.

The fundamental reason underlying the aforementioned technical difficulties is that the need for a semantics was apparently overlooked in the initial design of P3P and APPEL. We recognize the impact of this claim given that, to the best of our knowledge, there have been no formal semantics to ground P3P and APPEL. This lack of a formal and precise semantics allows inconsistent P3P policies to be written and allows inconsistent representations of P3P policies to users because it leaves too much freedom for user agents to misinterpret P3P policies. design. Similarly, APPEL's design is purely syntax-based; it is based on the XML-encoding of information, rather than the information itself. Because P3P allows the same policy to be encoded differently, a preference specified in APPEL may accept one encoding and reject another. The XPath-based preference language, XPref [3], has the same problem.

In this paper, we propose a semantics-based approach for analyzing and designing privacy policy and preference languages. We develop a relational formal semantics for P3P. A P3P policy is essentially a collection of statements, each of which describes the purpose, retention and recipient of a piece of collected data. However, the current P3P specification is missing a well-defined model of the relationships between purposes, retentions, recipients and the collected data. Moreover, a P3P statement may be semantically inconsistent and/or in conflict with other P3P statements in the same policy. Our formal semantics for P3P transforms each P3P policy into a database that has five relations: data purpose, data recipient, data retention, data collection, and data category. Integrity constraints are introduced to maintain a P3P policy's semantic consistency. We observe that a formal semantics by itself does not eliminate the problem of potential misrepresentation of P3P policies. Standards and guidelines for consistent user interface and vocabulary representation are also necessary. However, a formal semantics for P3P is a necessary step, without which there is little hope for completely solving P3P's misrepresentation and ambiguity problems.

In this paper we also propose SemPref, a semantics-based preference language for P3P. Unlike previously proposed preference languages, SemPref queries the meaning of a privacy policy (which is represented as a database) rather than its syntactical representation. As a result, SemPref has a concise and clear syn-

Example 1. *A P3P Example Statement*

```
<STATEMENT>
 <PURPOSE><admin required="opt-in"/></PURPOSE>
 <RECIPIENT><public/></RECIPIENT>
 <RETENTION><indefinitely/></RETENTION>
 <DATA-GROUP>
  <DATA ref="#user.home-info.postal"></DATA>
 </DATA-GROUP>
</STATEMENT>
```

```
stmt(
 purpose:  {admin(opt-in)}
 recipient:  {public}
 retention:  {indefinitely}
 data:  {#user.home-info.postal}
)
```

Figure 1: An Example P3P Statement. The XML representation appears on the left side and a more succinct representation on the right side.

tax, making it much easier to understand. It offers a declarative semantics as well as a simple and efficient evaluation algorithm.

The remainder of this paper is organized as follows. Section 2 provides an overview of P3P and presents a formal semantics for it. Section 3 analyzes the pitfalls and limitations of APPEL. Section 4 describes SemPref, a semantics-based privacy preferences exchange language. The relevant work is discussed in Section 5. Finally, a summary and our plans for future work appear in Section 6.

2 A Formal Semantics for P3P

Although at first sight P3P appears to be a simple XML-based language, developing a formal semantics for it is quite challenging. There are often several ways to interpret a particular P3P policy and the P3P specification does not clearly state which way is the correct one. There are also many ways for a P3P policy to be semantically inconsistent as we now discuss.

2.1 An Overview of P3P's Syntax

Each P3P policy is specified by one `POLICY` element that includes the following major elements.

One `ENTITY` element: identifies the legal entity making the representation of privacy practices contained in the policy.

One `ACCESS` element: indicates whether the site allows users to access the various kinds of information collected about them.

One `DISPUTES-GROUP` element: contains one or more `DISPUTES` elements that describe dispute resolution procedures to be followed when disputes arise about a service's privacy practices.

Zero or more `EXTENSION` elements: contain a website's self-defined extensions to the P3P specification.

One or more `STATEMENT` elements: describe data collection, use and storage. A `STATEMENT` element specifies the data (e.g. user's name) and the data categories (e.g. user's demographic data) being collected by the site, as well as the purposes, recipients and retention of that data.

There are two kinds of P3P statements. The first kind contains the `NON-IDENTIFIABLE` element, which is used to indicate that either no information will be collected or information will be anonymized during collection. The second kind does not contain the `NON-IDENTIFIABLE` element; this is the commonly used one. In this paper, we will focus on the second kind of P3P statements.

Figure 1 gives an example of the second kind of statements. (We also show a more succinct representation on the right side. Due to space limitation, we will use this succinct representation in the rest of the paper.) Each such statement contains the following:

One PURPOSE element, which describes for which purpose(s) the information will be used. It contains one or more pre-defined values such as admin, individual-analysis and historical. A purpose value can have an optional attribute required, which takes one of the following values: 'opt-in', 'opt-out', and 'always'. The value 'opt-in' means that data may be used for this purpose only when the user affirmatively requests this use. The value 'opt-out' means that data may be used for this purpose unless the user requests that it not be used in this way. The value 'always' means that users cannot opt-in or opt-out of this use of their data. Therefore; in terms of strength of data usage, 'always' $>$ 'opt-out' $>$ 'opt-in'. In Figure 1, PURPOSE is opt-in.

One RECIPIENT element, which describes with whom the collected information will be shared. It contains one or more pre-defined values such as ours, delivery and public. A recipient value can have an optional attribute required, which is similar to that of a PURPOSE element. In Figure 1, RECIPIENT is public.

One RETENTION element, which describes for how long the collected information will be kept. It contains exactly one of the following pre-defined values: no-retention, stated-purpose, legal-requirement, business-practices and indefinitely. In Figure 1, the RETENTION value is indefinitely.

One or more DATA-GROUP elements, which specify what information will be collected and used. Each DATA-GROUP element contains one or more DATA elements. Each DATA element has two attributes. The mandatory attribute ref identifies the data being collected. For example, '#user.home-info.telecom.telephone' identifies a user's home telephone number. The optional attribute indicates whether or not the data collection is optional. A DATA element may also contain a CATEGORIES element, which describes the kind of information this data item is, e.g., financial, demographic and health. In Figure 1, DATA is postal info.

Zero or one CONSEQUENCE element, which contains human-readable contents that can be shown to users to explain the data usage practice's ramifications and why the usage is useful.

2.2 Towards a formal semantics of P3P

Clearly, statements comprise the core of a P3P policy, as they specify a website's data-collection and data-use practices. They are also the most complicated parts of a P3P policy. In this paper, we limit the scope of the formal semantics to statements. To develop a formal semantics for P3P statements, we must first determine the relationships among the four major components (purpose, recipient, retention and data) of a P3P statement.

In the statement in Figure 1, the three components (purpose, recipient and retention) all refer to the same data item '#user.home-info.postal'; however, for the statement to have a precise meaning, one must also determine how these components interact. We consider two interpretations. In the first interpretation, all three components are related, i.e., the purpose, the recipient and the retention are about *one* data usage. In Figure 1, the postal information will be used for the administration purpose (technical support of the website and its computer system), the information will be shared with the public, and it will be stored indefinitely. This interpretation seems counterintuitive because in the P3P statement there is no need to share the data with the public for the administration purpose. Furthermore, it is not clear whether this data usage is required or optional, since the 'required' attribute has the 'opt-in' value for purpose but the default 'always' value for recipient. The explanation [8] for this statement, provided by one of the P3P architects, is that the data item '#user.home-info.postal' will always be collected and shared with the public. Additionally, if the user chooses to opt-in, their postal information will be used for the admin purpose. In other words, whether the individual's postal information will be shared with the public does not depend upon whether or not the information is used for the admin purpose.

This leads us to the second interpretation, in which purpose, recipient and retention are considered orthogonal. In this interpretation, a P3P statement specifies three relations: the purposes for which a data item will be used, the recipients with whom a data item will be shared, and how long the data item will be stored. Even though these relations are specified in the same statement, they are not necessarily about a single data usage. Given this *data-centric* interpretation, the following three P3P policies will have the same meaning in the sense that the three relations derived from them are the same.

Example 2. *Three P3P policies that have the same meaning.*

Policy 1:
```
stmt( data: {#user.home-info.telecom},
      purpose: {individual-analysis,
                telemarketing(opt-in)},
      recipient: {ours},
      retention: {stated-purpose} )
stmt( data: {#user.bdate(optional)},
      purpose: {individual-analysis,
                telemarketing(opt-in)},
      recipient: {ours},
      retention: {stated-purpose} )
```

Policy 2:
```
stmt( data: {#user.home-info.telecom,
                    #user.bdate(optional)},
       purpose: {individual-analysis},
      recipient: {ours},
      retention: {stated-purpose})
stmt( data: {#user.home-info.telecom,
                    #user.bdate(optional)},
       purpose: {telemarketing(opt-in)},
      recipient: {ours},
      retention: {stated-purpose} )
```

Policy 3:
```
stmt( data: {#user.home-info.telecom,
             #user.bdate(optional)},
      purpose: {individual-analysis,
                telemarketing(opt-in)},
      recipient: {ours},
      retention: {stated-purpose} )
```

As we will see in Section 3, the fact that the same meaning may be encoded in several different ways makes it very difficult to correctly express privacy preferences in a syntax-based preference language such as APPEL. One representation can be accepted by a preference, but another representation could be rejected by the same preference. A preference language based on the formal semantics proposed herein will help ensure the same meaning will always be handled the same way.

We adopt this data-centric interpretation in the rest of this paper for the following reasons. First, this appears to be the intention of P3P's designers, as it is consistent with the P3P specification and the explanation we were given [8].[1] Second, it is simpler than the first interpretation. It is not clear that end users can distinguish the differences among the three example policies in Example 2. Third, the size of the representation of such a semantics is smaller than one that is usage-centric. We discuss semantics that take other interpretations in Section 2.6.

In the statements in Example 2, the data item '#user.bdate' is 'optional' but the purpose 'individual-analysis' is 'always'. This seems counterintuitive: if the collection of the data is optional, why is it always used for a certain purpose? According to Cranor [8], this means that the collection of '#user.bdate' is optional, i.e, the user can choose not to provide the information. However, once the user provides it, it will be used for 'individual analysis'. The user cannot opt out of this purpose. This explanation is consistent with the data-centric interpretation in which the components are orthogonal. It suggests that data collection is independent of data usage. In addition to the three dimensions: purpose, recipient, and retention, there is a

[1]In fact, if the first interpretation is intended, it is more intuitive for the 'required' attribute to be associated with the whole statement rather than with both purpose and recipient.

Relation name	Field name	Domain of the field	Key for the relation
d-purpose	*data* *purpose* *required*	URI references to data items The P3P-defined purpose values {opt-in, opt-out, always}	(*data*, *purpose*)
d-recipient	*data* *recipient* *required*	URI references to data items The P3P-defined recipient values {opt-in, opt-out, always}	(*data*, *recipient*)
d-retention	*data* *retention*	URI references to data items The P3P-defined retention values	(*data*)
d-collection	*data* *optional*	URI references to data items {required, optional}	(*data*)
d-collection	*data* *category*	URI references to data items The P3P-defined category values	(*data*, *category*)

Figure 2: The schema for the five relations in the data-centric semantics for P3P.

fourth dimension about whether data collection is required or optional. Therefore, we need a fourth relation in the P3P semantics to specify this.

In a P3P statement, each `DATA` element has a set of categories associated with it. Some categories are implicitly specified by the base P3P data schema whereas some others are specified explicitly in the statement. We need another (fifth) relation to store the categories with which a data item is associated, as this is not included in any of the other four relations.

2.3 A data-centric semantics for P3P

We now propose a formal semantics for P3P policies. Recall that a P3P statement determines three relations, specifying the purpose, recipient, and retention associated with data items. We also need a fourth relation to specify whether the data collection is required or optional. Finally, we need a relation to store the categories (i.e., financial, health, demographic, etc.) associated with a data item. Thus, in the semantics, every P3P policy's data usage part is mapped onto five relations. (See any standard textbook on databases for an introduction to relational databases.) The schemas for the five relations are given in Figure 2.

One can consider the semantics of a P3P policy as a database consisting of five tables (the previous subsection provides the justification and rationale for why we need these five tables). For example, the three policies in Example 2 all have the same semantics, given by the semantic database in Figure 3. (Note that The d-category relation is constructed according to the P3P specification, which specifies that one's telecom information belongs to the Physical Contact Information category and one's date of birth information belongs to the Demographic and Socioeconomic Data category.)

Given a set of P3P statements, it is straightforward to translate them into the data-centric semantics. Intuitively, for each data item in a statement, we pair it with each purpose, recipient and the retention in the statement, then insert the resulting pairs into corresponding relations.

2.4 Potential semantic inconsistencies in P3P policies

In general, any combinations of the values for purpose, recipient and retention are allowed in P3P. However, in a practical setting, semantic dependencies arise naturally between these values, making some of the combinations invalid. A P3P policy using invalid combinations is thus semantically inconsistent. This problem has been recognized [7, 21], and P3P's designers are beginning to address some of these conflicts [7].

d-purpose	*data*	*purpose*	*required*
	`#user.home-info.telecom`	`individual-analysis`	`required`
	`#user.home-info.telecom`	`telemarketing`	`opt-in`
	`#user.bdate`	`individual-analysis`	`required`
	`#user.bdate`	`telemarketing`	`opt-in`
d-recipient	*data*	*recipient*	*required*
	`#user.home-info.telecom`	`ours`	`required`
	`#user.bdate`	`ours`	`required`
d-retention:	*data*	*retention*	
	`#user.home-info.telecom`	`stated-purpose`	
	`#user.bdate`	`stated-purpose`	
d-collection:	*data*	*optional*	
	`#user.home-info.telecom`	`required`	
	`#user.bdate`	`optional`	
d-category:	*data*	*category*	
	`#user.home-info.telecom`	`'Physical Contact Information'`	
	`#user.bdate`	`'Demographic and Socioeconomic'`	

Figure 3: The semantic database for the three P3P policies in Example 2, which have the same meaning.

Nonetheless, many places where potential conflicts may occur have not been previously identified. We now identify some additional classes of potential semantic inconsistencies in P3P.

Issue 1. *A P3P policy may be inconsistent because multiple retention values apply to one data item.*

P3P allows one data item to appear in multiple statements, which introduces a semantic problem. Recall that in each P3P statement, only one retention value can be specified, even though multiple purposes and recipients can be used. The rationale behind this is that retention values are mutually exclusive, i.e., two retention values conflict with each other. For instance, *no-retention* means that "Information is not retained for more than a brief period of time necessary to make use of it during the course of a single on-line interaction"[11]. And *indefinitely* means that "Information is retained for an indeterminate period of time"[11]. One data item cannot have both retention values. However, allowing one data item to appear in multiple statements makes it possible for multiple retention values to apply to one data item.

Issue 2. *A statement may have conflicting purposes and retention values.*

Consider a statement in a P3P policy that collects users' postal information for the purpose *historical* with retention *no-retention*. Clearly, if the postal information is going to be "... archived or stored for the purpose of preserving social history ...", as described by the *historical* purpose, it will conflict with *no-retention*, which requires that the collected information "... MUST NOT be logged, archived or otherwise stored"[11].

Issue 3. *A statement may have conflicting purposes and recipients.*

Consider a statement that includes all the purpose values (e.g., history, admin, telemarketing, individual-analysis, etc.) but only the recipient value *delivery* (delivery services). This does not make sense as one would expect that at least *ours* should be included in the recipients.

Issue 4. *A statement may have conflicting purposes and data items.*

Certain purposes imply the collection and usage of some data items. This has been recognized by the P3P designers and reflected in the guidelines for designing P3P user agents [7]. For example, suppose a statement contains purpose contact but does not collect any information from physical or online. Then the statement is inconsistent because, in order to contact a user, "the initiator of the contact would possess a data element identifying the individual .... This would presuppose elements contained by one of the above categories"[7].

We suggest that all semantic inconsistency instances be identified and specified in the P3P specification. Completion of this work requires a detailed analysis of the vocabulary, which is beyond the scope of the current paper; ideally by the individuals who design and use these vocabularies.

2.5 Integrity constraints in the semantics

We handle the aforementioned semantic problems by employing integrity constraints in the semantics. If a P3P policy is translated into a semantics database that violates these constraints, then this policy is invalid. Such a set of integrity constraints can benefit both end users and websites. First, policies that are semantically inconsistent can be automatically rejected by user agents. Thus, the design of preference languages is simplified because we only need to handle semantically consistent policies. Second, a website may also use integrity constraints to detect semantic inconsistency in their policies and fix them promptly, to avoid confusing Internet users.

In the formal semantics for P3P, we specify the following classes of integrity constraints:

Data-Centric Constraints. The keys in the relations imply four functional dependency constraints. For example, in the d-purpose relation, a pair (*data item*, *purpose value*) can only have one *required* value. This is a reasonable constraint because the three required choices are mutually exclusive. It does not make sense for the purpose of a data item to be, for example, both opt-in and required. The same constraint is also applied to the d-recipient relation and d-retention relation. Similarly, a website cannot specify that the collection of a data item is both optional and required. Therefore, the d-collection relation requires that no more than one optional value be associated with a data item. These constraints are implied by the definition of the semantics and do not need to be explicitly specified.

Data Hierarchy Constraints. Data items in P3P are organized into hierarchies. Each data item can be viewed as a node in a tree. For example, the data item '#user.home-info' is a child node of the data item node '#user', and is the parent node of the three data items: '#user.home-info.postal', '#user.home-info.telecom' and '#user.home-info.online'. When a data collection statement refers to a node in the tree, it means all data in the subtree rooted at the node may be collected.

This introduces the potential for a semantics conflict. For example, if the collection of '#user.home-info' is required, which means that the collection of all data items under '#user.home-info' is required, then it does not make sense for the collection of '#user.home-info.online' to be optional. However, it may be reasonable for the collection of '#user.home-info' to be optional, but the collection of '#user.home-info.online' to be required, which would mean that the collection of '#user.home-info.postal' and '#user.home-info.telecom' are optional.

Based on the above observation, we define the following constraint. For any two tuples $\mathsf{d\text{-}purpose}(d_1, p_1, r_1)$ and $\mathsf{d\text{-}purpose}(d_2, p_2, r_2)$, if d_1 is more specific than d_2 and $p_1 = p_2$, then $r_1 \geq r_2$. ('always' $>$ 'opt-out' $>$ 'opt-in'). Similar constraints apply to the d-recipient relation and the d-collection relation.

The data hierarchy constraint for the d-retention relation is as follows. For any two tuples $\mathsf{d\text{-}retention}(d_1, r_1)$ and $\mathsf{d\text{-}retention}(d_2, r_2)$, if d_1 is more specific than d_2, then $r_1 \geq r_2$ ('indefinitely'

$>$ 'business-practices', 'legal-requirement', 'stated-purpose' $>$ 'no-retention'; the middle three values are incomparable).

Semantic Vocabulary Constraints. As discussed in Section 2.4, many contradictions may arise if we carefully examine the semantics of P3P's pre-defined values. These can be specified as integrity constraints. For example, we may use the following integrity constraints to address some of the semantic inconsistencies among purposes, recipients and retentions.

Constraint 1. *If a data item is collected for the purpose* `historical`*, then its retention cannot be* `no-retention`.
$\forall p \in \textsf{d-purpose}\ \neg\exists r \in \textsf{d-retention}, p.data = r.data \wedge p.purpose = \texttt{historical} \wedge r.retention = \texttt{no}-\texttt{retention}$

Constraint 2. *If a data item is shared with public fora, then its retention should be* `indefinitely`. *(From [11].)*
$\forall s \in \textsf{d-recipient}\ r \in \textsf{d-retention}, (s.data = r.data \wedge s.recipient = \texttt{public}) \rightarrow r.retention = \texttt{indefinitely}$

Identifying all the constraints to handle all such inconsistencies is beyond the scope of this paper, but these examples demonstrate the need for individuals who design these vocabularies to conduct a detailed analysis.

2.6 Discussion

Because of the lack of clear specification, we have to make some judgment calls in defining a formal semantics for P3P. We make these decisions based on the information we obtained from P3P architects and the rationales for these decisions are documented in this paper.

The proposed semantics takes a strong step in the formal study of online privacy policies; however, it is not intended to be the only interpretation of P3P to be accepted by everybody. Rather, we view the proposed semantics as a starting point for a standard semantics for P3P. We now explore alternative designs and related issues.

Alternative Semantics for P3P. As mentioned in Section 2.2, other alternative semantics certainly exist. For example, instead of a data-centric semantics, one can also design a purpose-centric semantics [1], where a data item along with a purpose determines other elements (i.e., recipients and retention) in a P3P statement. Similar to the data-centric semantics, a purpose-centric semantics may be modeled as two relations $\textsf{dp-recipient}(data, purpose, recipient)$ and $\textsf{dp-retention}(data, purpose, retention)$, where $data$ and $purpose$ form a primary key for both relations. Integrity constraints can be defined accordingly.[2]

The rationale of this purpose-centric semantics is obvious. In practice, certain data are sometimes used for multiple purposes. Depending on the specific purposes, the data may be shared by different parties and may be kept for different periods of time. In some sense, the purpose-centric semantics represents a fine-grained interpretation of P3P whereas the data-centric semantics is relatively coarse-grained. The two semantics reflect a tradeoff between expressiveness and ease of management. We choose the data-centric semantics due to its simplicity. Further a coarse-grained semantics enables users to act more conservatively

[2] In fact, as shown in section 2.2, some tricky issues may arise when one tries to handle the *required* attribute of purposes and recipients. Thus, more integrity constraints may need to be introduced. A detailed discussion about this topic is outside the scope of this paper.

when disclosing information to a website, without worrying about complex relationships between the major components of P3P statements.

Vocabulary Issues. Besides providing a set of pre-defined values, P3P also allows websites to define their own data schemas and categories, so that privacy policies can be better tailored to fit specific applications. However, Internet users and websites often interact from different domains. They may not have any pre-existing knowledge about each other. We currently lack a mechanism that allows two parties to *dynamically* agree on a common vocabulary for the data schema and category definitions. Without such a mechanism, websites' self-defined data schemas will not be understood by user agents. In order to protect their privacy, users may have to reject any policies involving non-standard data schemas, or design very complex rules, hoping to cover all possible self-defined data schemas from a website.

3 An Analysis of APPEL

P3P enables websites to express their privacy policies. To support users, the creators of P3P designed APPEL (A P3P Preference Exchange Language) [20]. APPEL allows users to specify privacy preference rules, termed rulesets. The ruleset can then be used by the user agent to make automated or semi-automated decisions regarding whether a P3P enabled website's policies are acceptable to the user. A main design goal of APPEL is to allow users to import preference rulesets created by other parties and to transport their own ruleset files between multiple user agents.

Many authors have noted that APPEL is complex and problematic [3, 14, 15, 25]. In this section, we analyze APPEL's pitfalls and the rationales for some of the design decisions embedded in APPEL. The main objective for our analysis is to ensure we design a preference language that avoids the APPEL's pitfalls but preserves the desirable functionalities in APPEL.

3.1 An overview of APPEL

Privacy preferences are expressed as a *ruleset* in APPEL. A ruleset is an ordered set of rules. An APPEL evaluator evaluates a ruleset against a P3P policy.[3] A rule includes the following two parts:

- A behavior, which specifies the action to be taken if the rule fires. It can be *request*, implying that a P3P policy conforms to preferences specified in the rule body and should be accepted. We call this an *accept* rule. It can be *block*, implying that a P3P policy violates the user's privacy preferences and should be rejected. We call this a *reject* rule. It can also be *limit*, which can be interpreted as accept with warning.
- A number of expressions, which follow the XML structure in P3P policies. An expression may contain subexpressions. An expression is evaluated to TRUE or FALSE by matching (recursively) against a target XML element. A rule *fires* if the expressions in the rule evaluate to TRUE.

Every APPEL expression has a *connective* attribute that defines the logical operators between its subexpressions and subelements of the target XML element to be matched against. A connective can be: *or*, *and*, *non-or*, *non-and*, *or-exact* and *and-exact*. The default connective is *and*, which means that all subexpressions must match against some subelements in the target XML element, but the target element may contain subelements that do not match any subexpression. The connective *and-exact* further requires that

[3]The APPEL specification allows arbitrary XML elements not related to P3P to be included together with the P3P policy for evaluation. Since the users may not even know about them, it is not clear how they write preferences dealing with them. In this paper, we assume that only a P3P policy is evaluated against a ruleset.

any subelement in the target match one subexpression. The *or* connective means that at least one subexpression matches a subelement of the target. The *or-exact* connective further requires that all subelements in the target matches some subexpressions.

When evaluating a ruleset against a P3P policy, each rule in a ruleset is evaluated in the order in which it appears. Once a rule evaluates to true, the corresponding behavior is returned.

The following subsections examine the pitfalls and limitations of APPEL.

3.2 Semantic inconsistencies of APPEL

Because of APPEL's syntax-based design, two P3P policies that have the same semantic meanings but are expressed in syntactically different ways may be treated differently by one APPEL ruleset. This deficiency in APPEL has been identified before [16]. We now show an example APPEL rule.

```
01 <appel:RULE behavior="block">
02  <p3p:POLICY>
03   <p3p:STATEMENT>
04    <p3p:DATA-GROUP>
05     <p3p:DATA ref=
06      "#user.home-info.telecom"/>
07     <p3p:DATA ref="#user.bdate"/>
08    </p3p:DATA-GROUP>
09   </p3p:STATEMENT>
10  </p3p:POLICY>
11 </appel:RULE>
```

This is a reject rule, since the behavior (on line 1) is "block". The body of this rule has one expression (lines 2-10) for matching a P3P policy. This expression contains one subexpression (lines 3-9), which in turns contain one subexpression (lines 4-8). The outmost expression (lines 2-10) uses the default **and** directive; therefore, it matches a P3P policy only if the policy contains at least one statement that matches the enclosed expression (lines 3-9). Overall, this APPEL rule says that a P3P policy will be rejected if it contains a `STATEMENT` element that mentions both the user's birthday and the user's home telephone number.

This rule rejects Policies 1 and 2 in Example 3, but not Policy 3. In Policy 3, the two data items are mentioned in different statements and no statement mentions both data items. This is clearly undesirable, as the three statements have the same semantics. This problem is a direct consequence of that fact that APPEL is designed to query the representation of a P3P policy, rather the semantics of the policy.

The same problem exists in XPref [3], since it is also syntax-based.

3.3 The subtlety of APPEL's connectives

The meaning of an APPEL rule depends very much on the connective used in the expressions. However, the connectives are difficult to understand and use. The APPEL designers made mistakes using them in the first example in the APPEL specification [20]. Consider the following example taken from [20].

Example 3. *The user does not mind revealing click-stream and user agent information to sites that collect no other information. However, she insists that the service provides some form of assurance.*

The APPEL rule used in [20] for the above example is as follows:

```
01 <appel:RULE behavior="request"
02   description="clickstream okay">
03 <p3p:POLICY>
04  <p3p:STATEMENT>
05   <p3p:DATA-GROUP
06     appel:connective="or-exact">
07    <p3p:DATA
08     ref="#dynamic.http.useragent"/>
09    <p3p:DATA
10     ref="#dynamic.clickstream.server"/>
11   </p3p:DATA-GROUP>
12  </p3p:STATEMENT>
13  <p3p:DISPUTES-GROUP>
14   <p3p:DISPUTES service="*"/>
15  </p3p:DISPUTES-GROUP>
16 </p3p:POLICY>
17 </appel:RULE>
```

The above APPEL rule is an accept rule; its body has one outmost expression (lines 3-16) to match a P3P policy. The expression contains two subexpressions, matching different elements in a policy. The expression denoted by the `p3p:POLICY` element (lines 3–16) does not have the 'connective' attribute; therefore, the default **and** connective is used, which means that as long as the two included expressions, i.e., `p3p:STATEMENT` (lines 4-12) and `p3p:DISPUTES-GROUP` (lines 13-15), match some parts in the P3P policy, the rule accepts the policy. The expression denoted by `p3p:DATA-GROUP` uses the **or-exact** connective, it matches a `DATA-GROUP` element if the `DATA` elements contained in the element is a non-empty subset of {#dynamic.http.useragent, #dynamic.clickstream.server}.

Overall, this rule means that a P3P policy will be accepted if it contains a `STATEMENT` element that mentions only the two specified data items and a `DISPUTES-GROUP` element.

Observe that this rule does not express the preference, as it does not take into consideration the fact that a P3P policy can have multiple statements. Subsequently, a policy that only mentions "#dynamic.http.useragent" in the first statement can be accepted by the rule, even if the next statement collects and uses other user data.

One may try to fix this problem by using the **and-exact** connective on line 2, which means that each element contained in the P3P policy must match one of the expressions within the APPEL rule. For such a rule to work as intended, the `p3p:POLICY` expression (lines 3–16) must contain two additional sub-expressions: `p3p:ENTITY` and `p3p:ACCESS`. Otherwise, no P3P policy will be accepted because of the existence of `ENTITY` and `ACCESS` elements. However, this fix still does not work. A P3P policy may optionally contain an `EXTENSION` element. Even when such a policy collects only '#dynamic.http.useragent' and '#dynamic.clickstream.server', it will not be accepted by the above rule, due to the semantics of **and-exact**. On the other hand, including a sub-expression `p3p:EXTENTION` cannot fix the problem, as a policy without extensions will not be accepted in this case.

Another approach to fix the problem is to use the **or-exact** connective on line 3 and to include subexpressions for `p3p:ENTITY`, `p3p:ACCESS` and `p3p:EXTENSION`. Recall that the **or-exact** connective means that all elements in the policy must match some subexpressions, but not every subexpression is required to match some element in the policy. However, this means that the `p3p:DISPUTES-GROUP` element also becomes optional. A P3P policy that does not have the `DISPUTES-GROUP` element will also be accepted. This is not the preference described in Example 3.

In fact, as far as we can see, there is no way to correctly specify the preference in Example 3 in APPEL. One source of difficulty is that one has to intermingle statements and other aspects (e.g., dispute procedures) of a P3P policy in a preference rule. This is caused by APPEL's syntactic nature and the fact that statements and dispute procedures are all immediate children of a `POLICY` element. If conditions about data usages and other aspects of P3P policies may be specified separately, it is possible to specify the conditions on data usage in Example 3 using the **or-exact** connective.

3.4 The subtlety of accept and reject

APPEL uses both accept and reject rules. This means that one preference rule does not need to completely decide whether to accept or reject a policy. Unfortunately, this introduces another level of subtlety into APPEL. Consider the following example, which was used by Agrawal et al. [3] to illustrate the limitation of APPEL:

Example 4. *Only the purposes current and pseudo-analysis are acceptable.*

There are three ways to interpret the above preference. Let us classify P3P policies into two classes. Class one contains all policies in which the only purposes mentioned are a subset of {'current', 'pseudo-analysis'}. Class two contains all other policies. The three possible interpretations are:

1. Class two policies should be rejected. Class one policies may or may not be accepted, depending on other rules.
2. Class one policies should be accepted. Class two policies may or may not be rejected, depending on other rules.
3. Class one policies should be accepted, and class two policies should be rejected.

The three interpretations are different. From the wording of the preference, interpretation 1 seems to be the correct interpretation. Agrawal et al. [3] did not differentiate the three interpretations. They started by implementing interpretation 2 and discussed two approaches and the reasons why they did not work. The first approach is to use the default **and** connective in the `p3p:POLICY` element, which accept policies that it should not. The second approach is to use the **and-exact** connective, which rejects good policies. The reasons why these two approaches fail are similar to the example discussed in Section 3.3.

Additionally, Agrawal et al. did not discuss the approach of using the **or-exact** connective, which would correctly express interpretation 2 of the preference. Instead, they concluded that one must specify preferences using reject rules and switched to implement interpretation 1. Because one can only specify what occurs in an expression, one has to explicitly list all the purposes other than *current* and *pseudo-analysis* in a reject rule. They pointed out that such an approach is verbose and hard to understand, and more importantly, this approach is not robust since websites may use extensions to define additional purposes.

4 SemPref: A Semantics-Based Privacy Preference Language

In this section, we present SemPref, a semantics-based privacy preference language that seeks to avoid APPEL's pitfalls. To accomplish this, we must first understand where those pitfalls originate. Although many authors agree that APPEL is complex and problematic [3, 14, 15, 25], most authors have not analyzed the reasons for these problems. We argue that the problems with APPEL come directly from its syntax-based design. It is designed to match the XML representation of a P3P policy, rather than the *underlying meaning* of a P3P policy.

It is fundamentally more difficult to express privacy preferences in a syntax-based preference language compared with doing so in a semantics-based language. To express one's preferences, one starts by thinking about meanings of what policies should be accepted or rejected. Using a syntax-based language, one also needs to think about how these meanings may be encoded syntactically in P3P and try to cover all the possible representations of the same meaning.

4.1 Design Desiderata

Before presenting SemPref, we first provide a set of desiderata for designing SemPref.

Semantic consistency. Given any preference encoded in the preference language, two P3P policies that have the same semantic meaning should have the same result. Syntax-based languages such as APPEL and XPref violate this requirement.

Efficient evaluation. Evaluating a preference against a P3P policy should be efficient, i.e., worst-case time complexity is low-degree polynomial in the size of the P3P policy and the preference ruleset.

Declarative semantics. In addition to algorithms for checking a preference ruleset against a P3P policy, there should also be a declarative semantics defining those P3P policies that are accepted or rejected by a preference ruleset.

Ease of use. It should be relatively easy for end users to specify preference rules and understand them. APPEL has been shown to be very hard to use [3, 14, 15, 16]; and XPref requires first learning another language, XPath, a complex language that is the subject of many books.

Expressive power. Common preferences should be expressible in the language.

Support for partial specification of preferences. The preference language should allow a policy to be neither accepted nor rejected. This is one of the design decisions made in APPEL. The motivation is as follows. Because of the large number of possible combinations, it is very difficult (if not impossible) to create a complete specification covering all possible cases. If a user is required to come up with a complete specification before starting to use the tool, then the adoption will be very difficult. On the other hand, if a user can start by specifying some data usage practices to be acceptable and others to be unacceptable, and defer the decision about other cases to the time of accessing websites using those policies, the adoption will be much easier.

Some of the above considerations are unavoidably subjective, e.g., ease of use. Some considerations are mutually conflicting, e.g., there is always a tradeoff between expressive power and ease of use.

4.2 SemPref: Syntax, Semantics, and Examples

The BNF syntax of SemPref is provided in Figure 4. There are two kinds of rules in SemPref: accept rules and reject rules. Each rule has a body, which is a list of constraints. Each constraint consists of zero or more predicates; specify conditions on data, purpose, recipient, etc. We note that he above syntax is used in this paper for conciseness and ease of presentation. It can be equivalently encoded by using standards such as XML. To understand the semantics of SemPref, it helps to consider the relation data-usage that has eight fields: data, data-category, collection, purpose, purpose-required, recipient, recipient-required and retention. Each P3P policy P defines the table data-usage(P) in this relation. The table data-usage(P) can be obtained by doing a natural join across the five tables in the P3P semantics.

Given a constraint, a tuple of data-usage either satisfies it or does not. The relationship among multiple predicates in a constraint is logical AND. For example, given the constraint [

$$
\begin{aligned}
\langle\text{list of X}\rangle &::= \langle\text{X}\rangle \mid \langle\text{X}\rangle\ \langle\text{list of X}\rangle && (1)\\
\langle\text{set of X}\rangle &::= \text{“\{”}\ \langle\text{list of X}\rangle\ \text{“\}”} && (2)\\
\langle\text{ruleset}\rangle &::= \langle\text{list of rule}\rangle && (3)\\
\langle\text{rule}\rangle &::= \langle\text{accept-rule}\rangle \mid \langle\text{reject-rule}\rangle && (4)\\
\langle\text{accept-rule}\rangle &::= \text{“accept” “:”}\ \langle\text{list of constraint}\rangle\ \text{“;”} && (5)\\
\langle\text{reject-rule}\rangle &::= \text{“reject” “:”}\ \langle\text{list of constraint}\rangle\ \text{“;”} && (6)\\
\langle\text{constraint}\rangle &::= \text{“[”}\ [\langle\text{data-pred}\rangle \mid \langle\text{category-pred}\rangle\ ]\ [\langle\text{collection-pred}\rangle\ ]\ [\langle\text{purpose-pred}\rangle\ ] \\
&\qquad [\langle\text{pr-pred}\rangle\ ]\ [\langle\text{recipient-pred}\rangle\ ]\ [\langle\text{rr-pred}\rangle\ ]\ [\langle\text{retention-pred}\rangle\ ]\ \text{“]”} && (7)\\
\langle\text{data-pred}\rangle &::= \text{“(” “data” (“}\in\text{” | “}\notin\text{”)}\ \langle\text{set of data-value}\rangle\ \text{“)”} && (8)\\
\langle\text{collection-pred}\rangle &::= \text{“(” (“collection = optional” | “data-optional = required”) “)”} && (9)\\
\langle\text{category-pred}\rangle &::= \text{“(” “category” (“}\in\text{” | “}\notin\text{”)}\ \langle\text{set of category-value}\rangle\ \text{“)”} && (10)\\
\langle\text{purpose-pred}\rangle &::= \text{“(” “purpose” (“}\in\text{” | “}\notin\text{”)}\ \langle\text{set of purpose-value}\rangle\ \text{“)”} && (11)\\
\langle\text{pr-pred}\rangle &::= \text{“(” “purpose-required” (“}\in\text{” | “}\notin\text{”)}\ \langle\text{subset of \{opt-in opt-out, always\}}\rangle\ \text{“)”} && (12)\\
\langle\text{recipient-pred}\rangle &::= \text{“(” “recipient” (“}\in\text{” | “}\notin\text{”)}\ \langle\text{set of recipient-value}\rangle\ \text{“)”} && (13)\\
\langle\text{rr-pred}\rangle &::= \text{“(” “recipient-required” (“}\in\text{” | “}\notin\text{”)}\ \langle\text{subset of \{opt-in, opt-out, always\}}\rangle\ \text{“)”} && (14)\\
\langle\text{retention-pred}\rangle &::= \text{“(” “required” (“}\in\text{” | “}\notin\text{”)}\ \langle\text{set of retention-value}\rangle\ \text{“)”} && (15)
\end{aligned}
$$

Figure 4: Syntax of SemPref in BNF. Symbols inside quotation are terminals. Elements inside square brackets ([]) are optional. The first two definitions, ⟨list of X⟩ and ⟨set of X⟩, are macros.

`(data ∈ {user.bdate}) (purpose ∈ {individual-analysis})]`, which contains two predicates, a tuple of data-usage satisfies it if its data field is `user.bdate` and its purpose field is `individual-analysis`. An empty constraint is viewed as the constant "true"; every tuple satisfies it.

A constraint C thus also defines a table of the data-usage relation, denoted by data-usage(C), by selecting all the tuples that satisfy the predicate from the universal table of the data-usage relation. The universal table contains all the possible tuples for the data-usage relation, e.g., with all data items, purpose values, recipient values, etc.

An accept rule with $C_1, C_2, \cdots, C_k$ in its body means that a policy P is accepted by the rule if and only if:

$$\textsf{data-usage}(P) \subseteq \bigcup_{i=1}^{k} \textsf{data-usage}(C_i) \tag{1}$$

In other words, the rule-body, $C_1, C_2, \cdots, C_k$, also defines a table of the data-usage relation, given by $\textsf{data-usage}(C_1) \cup \cdots \cup \textsf{data-usage}(C_k)$. (Therefore, the relationship between the constraints $C_1, C_2, \cdots, C_k$ is logical OR.) A policy is accepted by an accept rule if and only if the data-usage table specified by the policy is contained in the table given by the rule-body.

Each accept rule defines a maximal acceptable set of data-collection and data-use practices. Multiple accept rules in one ruleset defines multiple maximal acceptable sets. Consider the following two accept rules, each having a single constraint in its body:

```
accept : [ (data ∈ {user.bdate}) ];
accept : [ (data ∈ {user.home-info}) ];
```

These two rules say that collecting and using only the user's birthday is acceptable, collecting and using only the user's home-info is also acceptable, but these two rules do not accept a policy that collects both information items. Such a policy will be accepted by the following accept rule, provided that the policy does not collect other information.

```
 accept : [ (data ∈ {user.bdate}) ]
          [ (data ∈ {user.home-info}) ];
```

Another way to understand the behavior of an accept rule is as follows: a policy P is accepted if every tuple in data-usage(P) satisfies at least one constraint (i.e., satisfying all the predicates in the clause) in the body of the rule.

A reject rule having $C_1, C_2, \cdots, C_k$ in its body means that a P3P policy P is rejected if and only if

$$\textsf{data-usage}(P) \cap \left(\bigcup_{i=1}^{k} \textsf{data-usage}(C_i) \right) \neq \emptyset \tag{2}$$

In other words, a policy P is rejected if data-usage(P) overlaps with the data-usage table specified by the rule-body. Thus, the body of a reject rule defines a set of data usages that are considered unacceptable.

Another way to understand the behavior of a reject rule is as follows: a policy P is rejected if any tuple in data-usage(P) satisfies at least one constraint in the rule-body. Multiple reject rules can be merged into one by combining all the constraints into one rule body.

We now use the following examples to illustrate how to express preference rules in SemPref.

Example 5. A user's preferences are as follows:

1. Policies that collect only click-stream and user agent information are acceptable. (From [20])

   ```
   accept :
    [(data∈{#dynamic.http.useragent,
          #dynamic.clickstream.server})];
   ```

2. The purposes current and pseudo-analysis are acceptable. The purpose individual-analysis is also acceptable, as long as the recipient is ours. (From [3])

   ```
   accept :
   [(purpose ∈{current,pseudo-analysis})]
   [(purpose ∈ {individual-analysis})
    (recipient ∈ {ours}) ];
   ```

   This rule contains two constraints, defining two maximal acceptable sets.

3. Reject policies that give out personal information to third parties, but only when the collection of such information is required *or* the data sharing is always.

   ```
   reject :
    [(data-category∈{physical,
                   demographic,uniqueid})
     (collection = required)
     (recipient ∉ {ours})]
    [(data-category∈{physical,
                   demographic,uniqueid})
     (recipient ∉ {ours})
     (recipient-required ∈ {always} ];
   ```

   This rule contains two constraints, describing two kinds of data usage that should be rejected.

4. Rejects if the P3P policy includes contact or telemarketing purposes unless those purposes are opt-in.

   ```
   reject :
    [(purpose∈{contact,telemarketing})
     (purpose-optional∈{opt-out,always})];
   ```

4.3 Dealing with conflicts

When a ruleset has both accept rules and reject rules, conflict may arise. A policy may be accepted by one rule but rejected by another rule. APPEL uses ordered rules to resolve conflicts. When two rules are in conflict, the earlier one overrides the latter one. As a result, the semantics of one rule cannot be completely determined without looking at earlier rules and understanding how they interact. This also makes it very difficult to import preferences from more than one source (which was one of APPEL's original objectives). When one combines two preference rulesets sequentially, the semantic of the second one is completely changed. Other researchers also question the adequacy of using ordered rules for preferences [25].

Our proposed approach for dealing with conflicts is to detect conflicts, inform users about those conflicts, help users rewrite rules to avoid conflicts, but allow conflicting rules to exist in one ruleset if the user chooses to do so. In this way, a ruleset partitions the P3P policy space into four sub-spaces: accepted, rejected, both accepted and rejected, and neither accepted nor rejected. A user may specify how the latter two cases should be handled. An obvious choice is to have a default action. For example, whenever a policy is not explicitly accepted by a ruleset, the user agent should reject it by default. On the other hand, a user may specify to be prompted to make a case-by-case decision for the latter two situations. When a policy is both accepted and rejected, the user may be given the opportunity to make a more permanent decision by revising the rules.

Tools for authoring preferences should perform a check to determine whether inconsistencies exist, and ask the user to revise preferences until the ruleset is consistent. In SemPref, an accept rule conflicts with a reject rule if there exists a tuple of data-usage that could satisfy one constraint in the former rule and one constraint in the latter rule. It is straightforward to check whether two constraints can both be satisfied.

Consider a ruleset consisting of the four rules in Example 5; 1 and 2 are accept rules; 3 and 4 are reject rules. Rule 3 does not conflict with rule 1, because it is impossible to satisfy both `(data ∈ {#dynamic.http.useragent, #dynamic.clickstream.server})` and `(data-category ∈ {physical, demographic, uniqueid})`. However, rule 3 conflicts with rule 2.

If one decides that rule 2 should override rule 3, one can revise rule 3 into the following:

```
reject :
 [(data-category∈{physical,
                demographic,uniqueid})
  (collection = required)
  (purpose ∉ {current, pseudo-analysis,
                  individual-analysis})
  (recipient ∉ {ours})]
 [(data-category∈{physical,
                demographic,uniqueid})
  (purpose ∉ {current, pseudo-analysis,
                   individual-analysis})
  (recipient ∉ {ours})
  (recipient-required ∈ {always} ];
```

Knowing which way the user wants to resolve the conflict, a tool may, in many cases, be able to generate candidate changes and ask the user to select from them. The simple structure of SemPref makes this possible. The details of such algorithms are beyond the scope of the current paper.

4.4 Evaluation algorithms for SemPref

Given a ruleset in SemPref and a P3P policy, the evaluation procedure is to evaluate each rule against the policy one by one. In Figure 5, we describe the algorithms for checking an accept rule against a policy as

```
01 // takes two parameters: the body of the accept rule and the policy.
02 check-accept(rule-body, policy)
03    foreach tuple $t$ in data-usage (policy)
04       pass = no;
05       foreach constraint $c$ in rule-body
06          if $t$ is subsumed by $c$ then pass = yes;
07       if (pass == no) then return no;  // don't accept
08    return yes;                         // accept
09
10 // takes two parameters: the body of the reject rule and the policy.
11 check-reject(rule-body, policy)
12    foreach tuple $t$ in data-usage (policy)
13       foreach constraint $c$ in rule-body
14          if ($t \wedge c$) is satisfiable  // reject even if $t$ is not subsumed to $c$
15          then return yes;   // reject
16    return no;               // don't reject
```

Figure 5: Evaluation algorithms for SemPref. Note the difference between line 6 and line 14. For accept, the policy should be fully contained in the rule-body. For reject, the policy should not overlap with the rule-body.

well as for checking a reject rule.

The algorithms in Figure 5 involve checking whether a data-usage tuple is subsumed by a constraint and whether the conjunction of a data-usage tuple and a constraint is satisfiable. Observe that a data-usage tuple can be translated into a constraint in a straightforward manner. Further observe that such checking can be performed in linear time, because the kinds of predicates involved are very simple.

The size of data-usage(P) is linear in the size of the P3P policy, assuming that total number of recipient values is a constant (P3P has 6 recipient values) and the number of categories associated with any data item is bounded by a constant. Furthermore, the algorithms in Figure 5 have worst-case time complexity $O(MN)$, where M is the size of data-usage(P) and N is the size of the ruleset. Therefore, evaluating a SemPref ruleset against a P3P policy is fairly efficient.

4.5 Analysis of SemPref

The most important strengths of SemPref are its semantics-based design and its simplicity. We now evaluate SemPref with respect to the desiderata presented in Section 4.1.

1. SemPref is semantically consistent; this follows straightforwardly from its semantics-based design.
2. Evaluation of SemPref is efficient; it takes time worst-case $O(MN)$, where M is the size of the P3P policy and N is the size of the preference ruleset.
3. SemPref has a simple, declarative semantics, given by equations 1 and 2. A user only needs to specify what constraints a preference should have, instead of how to evaluate those constraints for a given policy.
4. Compared with APPEL and XPref, SemPref is much easier to learn and understand. The semantics-based design of SemPref avoids arcane connectives in APPEL and the learning of another nontrivial language (XPath) in the case of XPref. In addition, SemPref rules are highly structured. Each SemPref rule has zero or more constraints, and each constraint has zero or more predicates. Inputting a

predicate amounts to input three values, i.e., the field name, the predicate, and a value set. Thus it is straightforward to build a simple GUI for authoring them.

5. SemPref can express most preferences in [20] and [3]. Since SemPref deals only with data collection and data usage information that are encoded in `STATEMENT` elements in P3P policies, one cannot use SemPref to express preferences involving constraints on things such as dispute resolution procedures, which are not encoded in `STATEMENT` elements. The only examples in [3, 20] that SemPref cannot express are of this kind. It is straightforward to extend SemPref to express constraints involving such information. For example, the preference in Example 3, which cannot be expressed in APPEL, can be expressed in the following extension of SemPref.

```
accept :
  [(data∈{#dynamic.http.useragent,
        #dynamic.clickstream.server})
   (disputes-type ∉ {no-resolution}) ];
```

   We omit further discussion of such extensions due to space limitation.

6. SemPref supports both accept and reject rule; in addition, it uses a simple conflict resolution mechanism, avoiding complicated interactions among different rules.

Given that our semantics for P3P uses a relational database, a natural preference language would be based on a subset of SQL. We did not take this approach for the following reasons. First, writing simple preferences in SQL uses a nontrivial subset of SQL. Although expressing a reject rule in SQL is relatively straightforward, expressing an accept rule needs to check the containment of two tables, which is not a basic SQL query. Second, the SemPref evaluation algorithms are simpler, making it possible to have a small and efficient user agent. In general, one can view SemPref as a macro language for writing database queries. Because it is targeted to databases having a specific schema and a specific class of queries, it can be made simpler and more accessible than a general language such as SQL. If in the future it is desirable to make the preference language more expressive, it may be worthwhile to investigate whether a subset of SQL should be used.

5 Related Work

A detailed description of P3P and APPEL can be found at [9, 11, 20]. Several APPEL implementations have been developed, including Privacy Bird from AT&T Labs-Research [5], which can be integrated into users' web browsers, and a java-based implementation from JRC [6, 16]. Agrawal et al. [2] designed a server-centric architecture for P3P, where user privacy preferences are matched with websites' P3P policies at the server side. In this architecture, privacy policies are stored in a relational database, and users' APPEL preferences are translated into SQL queries. Though database techniques are used for privacy preference matching, this approach is still syntax based. Relational databases are only used as a means for storing the XML representation of policies, and preference matching is still done by matching the representation of policies. No formal semantics is defined for P3P in [2].

Many researchers have noted the limitations of P3P and APPEL [14, 15, 3, 21, 25]. Hogben [14, 15] identified the limitations of P3P in terms of cookie management, user interfaces and vocabularies. The ambiguity and awkwardness of APPEL was also pointed out in [14, 15]. Hogben suggested investigating XPath as an alternative preference language. Schunter et al. [21] also showed the ambiguity of P3P and argued that the current P3P specification lacks a clear guideline for policy design and interpretation. Suggested solutions included augmented consent models, more specific element definitions and a simplified syntax.

Our work extends previous work by showing that the lack of a clear formal semantics is the fundamental reason for a variety of problems in P3P and APPEL.

Agrawal et al. [3] showed the limitations of APPEL with a series of plausible examples. They then proposed an XPath-based privacy preference language, XPref. XPref only uses a small subset of the XPath specification. Therefore, it can be efficiently evaluated. Meanwhile, XPref has many advantages over APPEL in terms of clarity, ease of use and expressiveness. On the other hand, XPref is still a syntax-based preference language and, thus, cannot overcome APPEL's problems completely. The work reported in this paper is in part inspired by the analysis of APPEL in [3].

As evidenced by the recent JetBlue Airways case, it is becoming increasing important for enterprises to effectively *enforce* their privacy policies in addition to simply specifying them. In [1], Agrawal et al. proposed a set of principles for designing databases that enforce a company's privacy policy. Karjoth et al. [17, 19] proposed a privacy-centric access control language (E-P3P and its successor EPAL), and designed an architecture for privacy policy enforcement in the entire life cycle of customers' information, based on the principle of separation of duty. Because of the dynamic nature of enterprise authorization, Karjoth et al. [18] also investigated the translation from enterprise authorization policies to P3P policies. Such a translation will help enterprises keep their privacy promises consistent with their privacy practices. Although a formal model is designed for EPAL in [19], it is focused on the information flow of an enterprise's internal operation. The semantics proposed in this paper concentrate on private information collection during online transaction. Therefore, some key concepts of EPAL such as action hierarchies and user hierarchies are not applicable in our model.

Though websites are increasingly posting their P3P policies, the dominant majority of online privacy policies are published in textual files. Compared to P3P policies, textual privacy policies usually cover a much larger scope of companies' privacy practices, and tend to be more ambiguous and incomplete. Current textual privacy policy analysis involves extensive inputs from application/domain experts [4]. Computer-aid semi-automated analysis is a promising means to increase its efficiency and accuracy. Techniques from multiple communities, such as databases, natural language processing and software engineering, may be applied.

6 Conclusion

Although P3P has received broad attention since it was proposed, its adoption has been slow. A fundamental reason for its slow adoption is that it lacks a formal and precise semantics. Consequently, a P3P policy may be inherently ambiguous and confusing to both end users and user agents. The neglect of semantics also results in APPEL, a purely syntax-based preference language, being overly complex and highly error-prone.

In this paper, we focus on the semantics of privacy languages and show that the semantics-based approach has several important advantages. We develop a data-centric relational semantics, in which a P3P policy is modeled as a relational database. This semantics is both simple and intuitive. In the process of creating the semantics, we have identified various ambiguities and semantic problems in P3P. Additionally, we have designed SemPref, a semantics-based preference language. Unlike previous preference languages, SemPref queries the meaning of a privacy policy instead of its syntactical representation. Therefore, SemPref offers a declarative semantics and a concise syntax, so that it can be adopted by end users without requiring a long learning curve.

This paper represents our first attempt at a formal study of online privacy policies. As discussed, many challenging issues remain to be addressed. We plan to extend our semantics-centered approach to areas including textual privacy policy modeling, privacy negotiation, privacy policy enforcement and privacy practice auditing.

Acknowledgment

This work is supported by NSF IIS-0430274. We would like to thank Lorrie Faith Cranor for answering our questions about P3P. We thank the anonymous reviewers for their helpful comments.

References

[1] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Hippocratic databases. In *Proceedings of the 24th International Conference on Very Large Databases*. ACM Press, August 2002.

[2] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Implementing P3P using database technology. In *Proceedings of the 19th International Conference on Data Engineering*, March 2003.

[3] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. An XPath-based preference language for P3P. In *Proceedings of the Twelfth International World Wide Web Conference (WWW2003)*, pages 629–639. ACM Press, May 2003.

[4] Annie I. Antón, Julia B. Earp, and Angela Reese. Analyzing web site privacy requirements using a privacy goal taxonomy. In *Proceedings of the 10th Anniversary IEEE Joint Requirements Engineering Conference (RE'02)*, pages 605–612, Essen, Germany, September 2002.

[5] AT&T Privacy Bird. http://privacybird.com.

[6] JRC P3P Resource Centre. http://p3p.jrc.it.

[7] Lorrie Cranor. P3P user agent guidlines, May 2003. P3P User Agent Task Force Report 23.

[8] Lorrie Faith Cranor. Personal communication.

[9] Lorrie Faith Cranor. *Web Privacy with P3P*. O'Reilly, 2002.

[10] Lorrie Faith Cranor and Joel R. Reidenberg. Can user agents acurately represent privacy notices?, August 2002. Discussion draft 1.0.

[11] Massimo Marchiori et al. The Platform for Privacy Preferences 1.0 (P3P1.0) Specification, April 2002. W3C Recommendation.

[12] UCLA Center for Communication Policy. The UCLA Internet report: Year three. Available at http://ccp.ucla.edu/pages/internet-report.asp.

[13] Joshua Freed. P3P Outreach Project, 2002.

[14] Giles Hogben. A technical analysis of problems with P3P v1.0 and possible solutions, November 2002. Position paper for W3C Workshop on the Future of P3P. Available at http://www.w3.org/2002/p3p-ws/pp/jrc.html.

[15] Giles Hogben. Suggestions for long term changes to P3P, June 2003. Position paper for W3C Workshop on the Long Term Future of P3P. Available at http://www.w3.org/2003/p3p-ws/pp/jrc.pdf.

[16] Giles Hogben, Tom Jackson, and Marc Wilikens. A fully compliant research implementation of the P3P standard for privacy protection: Experiences and recommendations. In *Proceedings of the 7th European Symposium on Research in Computer Security (ESORICS 2002)*, volume 2502 of *LNCS*, pages 104–125. Springer, October 2002.

[17] Gunter Karjoth and Matthias Schunter. A privacy policy model for enterprises. In *Proceedings of the 15th IEEE Computer Security Foundations Workshop (CSFW-15 2002)*, pages 271–281. IEEE Computer Society Press, June 2002.

[18] Gunter Karjoth, Matthias Schunter, and Els Van Herreweghe. Translating privacy practices into privacy promises – how to promise what you can keep. In *Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2003*, pages 135–146. IEEE Computer Society Press, June 2003.

[19] Gunter Karjoth, Matthias Schunter, and Michael Waidner. Platform for enterprise privacy practices: Privacy-enabled management of customer data. In *Proceedings of the Second International Workshop on Privacy Enhancing Technologies (PET 2002)*, number 2482 in LNCS, pages 69–84. Springer, 2003.

[20] Marc Langheinrich. A P3P Preference Exchange Language 1.0 (APPEL1.0). W3C Working Draft, April 2002.

[21] Matthias Schunter, Els Van Herreweghen, and Michael Waidner. Expressive privacy promises — how to improve the platform for privacy preferences (P3P). Position paper for W3C Workshop on the Future of P3P. Available at http://www.w3.org/2002/p3p-ws/pp/ibm-zuerich.pdf.

[22] Daniel M. Schutzer. Citigroup P3P position paper. Position paper for W3C Workshop on the Future of P3P. Available at http://www.w3.org/2002/p3p-ws/pp/ibm-zuerich.pdf.

[23] W3C. Platform for privacy preferences (P3P) project. http://www.w3.org/P3P/.

[24] W3C. Web sites using P3P. http://www.w3c.org/P3P/compliant_sites.

[25] Rigo Wenning. Minutes of the P3P 2.0 workshop, July 2003. Available at http://www.w3.org/2003/p3p-ws/minutes.html.