Privacy Policies and Their Enforcement in Composite Service Environment

der

Dissertation

Dem Promotionsausschuss der
Technische Universität Hamburg-Harburg
zur Erlangung des akademischen Grades
Doktor-Ingenieurin (Dr.-Ing.)

vorgelegte Dissertation

von

Assadarat Khurat

aus

Bangkok, Thailand

2014

1. Gutachter: Prof. Dr. Dieter Gollmann
2. Gutachter: Prof. Dr. rer. nat. habil. Ralf Möller
3. Gutachter: Prof. Dr.-Ing. Hannes Federrath
Prüfungsausschussvorsitzender: Prof. Dr. Sibylle Schupp

Tag der mündlichen Prüfung: 26.05.2014

urn:nbn:de:gbv:830-tubdok-12703

Hamburg University of Technology
Security in Distributed Applications
http://www.sva.tu-harburg.de/
Harburger Schloßstraße 20
21079 Hamburg
Germany

Declaration

I, Assadarat Khurat, solemnly declare that I have written this dissertation independently, and that I have not made use of any aid other than those acknowledged in this dissertation. Neither this dissertation, nor any other similar work, has been previously submitted to any examination board.

Hamburg, May 31, 2014

Assadarat Khurat

Abstract

In their early stage, online services were independent providing services all alone. Online services requiring customers to enter their data through the services' websites forced privacy issue to be considered. Nowadays, online services tend to cooperate with other existing services to compose new ones. Privacy thus becomes more important due to the increasing amount of user data being collected, stored and shared. Technically, privacy protection can be considered from two perspectives, i.e. before and after the users use the services. From the first viewpoint, it is beneficial for the users that their privacy preferences can be compared automatically with the services' privacy policies that describe what the services will do with their data. P3P (Platform for Privacy Preferences) is a privacy policy language designed to describe this information so that the users can decide whether to use a service or not. After the users have used the services, which means some user data have been given to the services, the actual enforcement on data usage of online services can be achieved by access control systems. How an access control system reacts to an access request is determined according to policies where XACML (eXtensible Access Control Markup Language) is one of the most widely used policy languages for access control.

The change of services from the single to the composite paradigm impacts on both viewpoints. First, P3P was designed from a single service perspective. For instance, its recipient element cannot express prospective services that will be combined into a composite service. In addition, when a composite service consists of e.g. three single services, the matching of users' privacy preferences with privacy policies of this composite service has to be done three times, one for each service member. It is thus beneficial to have a privacy policy for the new composite service so that the matching can be done only once. However, there is no mechanism provided in P3P to acquire the privacy policy of composite services. Moreover, there may be conflicts between the P3P policies of service members. P3P does not provide any mechanism regarding this issue either. Furthermore, the P3P policy language has some potential semantic inconsistencies from some combination of values between some elements in the policy.

In this thesis, we extend P3P to be suitable for composite services. To mitigate ambiguities in the P3P policy language, we propose a formal semantics for P3P using OWL (Web Ontology Language) and define constraints to verify potential semantic inconsistencies. In addition, we define constraints for conflict verification occurring from P3P policies of service members. These constraints are specified, instead of logical constraints, in special classes to capture the constraint violations. This way allows us to know the reason for the conflicts of a P3P policy by checking if the policy, viewed as an individual in our model, is inferred as an instance of any of those special classes. All defined constraints can be expressed in the special classes. We have implemented a P3P verification tool and verified five

hundred P3P policies collected from actual websites. The verification result shows that more than half of these P3P policies have conflicts. In addition, a combining algorithm is proposed such that we can automatically derive the P3P policy of a composite service.

Secondly, the problem affecting access control systems using XACML as their policy language in a composite service environment is that policy administration can be more complex and policy enforcement can be less efficient. Since there are multiple services cooperating with each other to perform a composite service, there might be the case where a member service normally provides an entire data object, e.g. an XML document, whereas the other member services require only a part of that data object. In order to preserve the collection limitation principle of the OECD Privacy Guidelines, policies at the data provider service governing the data access must be individualized for each data consumer service, which is a burden for policy administration. In addition, this situation could increase the computational cost of authorization decision evaluation in XACML. Because XACML was designed in such a way that it can evaluate one requested resource at a time, its evaluation process, thus, can only be performed for the entire document or for a node in the document. Should only a part of the document be accessed, separate evaluations have to be performed for every node in that part. Therefore, a composite service can highly increase the computational overhead of access control system using XACML especially when the XPath language is used to query nodes in an XML document.

We propose a way to express XACML policies that eases policy administration and a mechanism for reducing the overhead performance costs by applying a post-processing concept. The policy expression is achieved by using meaningful names representing parts of a document instead of all nodes and information details on allowed elements in each part. From this way of expression, our mechanism can provide one time XACML policy evaluation. If the evaluation result is permit, the requested resource will be filtered according to the information details such that only allowed data items are revealed. Therefore, fine-grained access control can also be preserved. We illustrate our proposed solution via a location-based service scenario. The implementation result shows that our mechanism brings better XACML evaluation performance when the number of nodes are high, and especially when XPath is used.

Abstrakt

Zu Beginn leisteten Online-Dienste ihren Dienst unabhängig und losgelöst von anderen existierenden. Um einen bestimmten Dienst bereitzustellen, war es für den Benutzer notwendig, seine personenbezogenen Daten in die Webseite des Dienstes einzugeben. Dieses machte es notwendig, den Datenschutz näher zu betrachten. Heute streben Online-Dienste danach, mit schon existierenden Diensten zusammen zu wirken, und so neue Dienste bereit zu stellen. Aufgrund der anwachsenden Menge an Benutzerdaten, die dabei erhoben, gespeichert und bereitgestellt werden, wird der Datenschutz dabei immer wichtiger. Aus technischer Sicht gibt es zwei Fälle zur Sicherung der Benutzerdaten, zum einen vor der Benutzung und desweiteren nach der Benutzung des Dienstes. Im ersten Fall ist es wichtig, dass die Datenschutzvorgaben der Benutzer automatisch mit den Datenschutzrichtlinien (Policies) des Services verglichen werden können, welche beschreiben, was mit den Benutzerdaten geschehen wird. Hierzu wird P3P (Platform for Privacy Preferences), eine Sprache zur Beschreibung der Datenschutzrichtlinien (P3P-Policy-Language), verwendet. Der Benutzer kann auf Grund der P3P-Policies entscheiden, ob er den Dienst nutzen möchte oder nicht. Im zweiten Fall, mit der Nutzung eines Dienstes, werden personenbezogene Daten des Benutzers zum Online-Dienst gesendet. Um den Datenschutz für Online-Dienste sicherzustellen, können Zugriffskontrollsysteme verwendet werden. Wie das Zugriffskontrollsystem auf eine Anfrage reagiert, wird durch Policies beschrieben. Hier ist XACML (eXtensible Access Control Markup Language) eine der bekanntesten und weit verbreitetsten Sprachen und gilt als Standard zur Beschreibung von Zugriffskontrollrichtlinien.

Der Wechsel vom Einzeldienst zum Diensteverbund betrifft beide oben genannten Fälle. P3P wurde für Einzeldienste entwickelt. So kann beispielsweise das “Recipient"-Element einer P3P-Policy keinen Zusammenschluss mehrerer der in Frage kommenden Dienste beschreiben. Desweiteren erhöht sich der Aufwand zum Abgleich der Datenschutzvorgaben und der Datenschutzrichtlinien. Besteht beispielsweise ein Diensteverbund aus drei Einzeldiensten, so muss der Abgleich für jeden der beteiligten Teildienste erneut durchgeführt werden, im Beispiel also drei Mal. Es wäre von Vorteil, eine einzelne Datenschutzrichtlinie für neue Verbunddienste ableiten zu können, mit denen dann der Abgleich zwischen Datenschutzvorgaben und -richtlinien nur einmal durchgeführt werden muss. P3P bietet keine Mechanismen an, um Datenschutzrichtlinien von Verbunddiensten zu erbringen. Außerdem kann es zwischen den P3P-Policies einzelner Teildienste des Diensteverbundes zu Konflikten führen. P3P bietet hierfür ebenso keine Mechanismen an. Auch wurde festgestellt, dass P3P-Policy-Language potenziell semantische Inkonsistenzen von Wertekombinationen zwischen einigen Elementen der Richtlinie beinhaltet.

In dieser Dissertation wird P3P angemessen erweitert, um Dienstzusammenschlüsse zu unterstützen. Um Uneindeutigkeiten in P3P-Policy-Language zu verringern, wird eine formale Semantik mittels

OWL (Web Ontology Language) vorgeschlagen und Bedingungen für eine Verifizierung von potentiell semantischen Inkonsistenzen definiert. Zusätzlich werden Bedingungen zur Konfliktprüfung definiert, die durch P3P-Policies der Teildienste des Diensteverbundes hervorgerufen werden. Anstatt die Bedingungen ein logisch zu definieren, werden spezielle Klassen erstellt, mit denen der Grund jedes Verstoßes der Bedingungen genau erfasst werden kann. Diese Vorgehensweise, jede Policy als Individuum in unserem Modell zu betrachten, und als Instanz von den speziellen Klassen abzuleiten, ermöglicht es, die Konfliktgründe einer P3P-Policy zu erkennen. Jede definierte Bedingung kann als Klasse ausgedrückt werden. Ein Konfliktprüfungsmechanismus wurde implementiert, mit dem in akzeptabler Rechenzeit 500 P3P-Policies von einer existierenden Webseite entnommen wurden. Das Testergebnis zeigt, dass über die Hälfte der P3P-Policies Konflikte aufwerfen. Zusätzlich wird ein Verbindungsalgorithmus vorgeschlagen, mit dem die Erstellung der P3P-Policies eines Dienstzusammenschlusses ermöglicht wird.

Zweitens, die Schwierigkeiten, die XACML basierte Zugriffskontrollen betreffen, ist eine komplexere Verwaltung und eine ineffizientere Durchsetzung der Regeln. Der Umstand, dass mehrere Dienste in einem Diensteverbund zusammenarbeiten, kann dazu führen, dass ein Teildienst ein vollständiges Datenobjekt (z.B. ein XML Dokument) zur Verfügung stellt, während andere Teildienste nur einen Teil des Objektes benötigen. Um die OECD Datenschutzrichtlinie zur Datenvermeidung einzuhalten, müssen die Regeln der Datenquelle an die Bedürfnisse der Datensenke individuell angepasst werden, dies bedeutet einen hohen Verwaltungsaufwand. Zusätzlich wird der Berechnungsaufwand im Zusammenhang mit der Zugriffskontrolle mit XACML erhöht. Dies ist der Fall, weil XACML entwickelt wurde, um nur eine Entscheidung gleichzeitig zu berechnen. Die Berechnung der Zugriffsentscheidung kann nur für das gesamte Dokument erfolgen oder einen Knoten im Dokument. Soll nur auf einen Teil des Dokumentes zugegriffen werden, müssen separate Evaluierungen für jeden Knoten dieses Teils durchgeführt werden. Daher wird im Fall eines Dienstzusammenschlusses die Zugriffsberechnung der XACML Policy einen erhöten Aufwand erzeugen, besonders wenn XPath-Sprache zum Abfragen von Knoten in einem XML-Dokument verwendet werden.

Wir zeigen einen Weg auf, die XACML-Policies auszudrücken, der sowohl die benötigte Verwaltung vereinfacht als auch den Berechnungsmehraufwand durch ein Nachberechnungskonzept verringert. Der Ausdruck der XACML-Policies wird in mehrere Teile gegliedert, die durch aussagekräftige Namen repräsentiert werden. Durch einen erlaubten Zugriff auf einen solchen Teilbereich werden nicht mehr alle, sondern nur solche Informationen zugänglich gemacht, die zum jeweiligen Teilbereich gehören. Auf diese Weise benötigt der Mechanismus nur eine einmalige XACML-Policies überprüfung. Bei einer Zugangsgewährung werden die angeforderten Daten den Informationsdetails nach gefiltert, sodass nur erlaubte Datenobjekte dem Anfordernden offengelegt werden. Die fein abgestufte Zugriffskontrolle bleibt erhalten. Die vorgestellte Lösung wird durch ein ortsbezogenes Diensteszenario (location-based service scenario) näher erläutert. Die Implementierung des Mechanismus zeigt im Ergebnis eine bessere XACML-Überprüfungseffizienz bei einer hohen Zahl an Knoten, speziellen, wenn XPath Verwendung finden.

Acknowledgement

During the time of my doctoral work, I have been supported from many people in different aspects. I would like to dedicate this section to mention their generosities.

The first and foremost person I would like to thank is my supervisor, Prof. Dr. Dieter Gollmann. He has given me tremendous advices, productive comments and proofreading my thesis. Moreover, I have learnt from him what a good adviser and researcher looks like.

I would like to thank Prof. Dr. Ralf Möller and Prof. Dr. Hannes Federrath to be the oral examiners, and Prof. Dr. Sibylle Schupp to be a chairman of the examination committee.

Special thanks go to Dr. Jorge Cuellar for giving me an important opportunity to work at Siemens and Dr. Jörg Abendroth, my supervisor at NSN, for great advices both in academic and living sides. I would like to thank my colleagues at NSN i.e. Robert Seidl, Dr. Manfred Schäfer, Dr. Michael Marhöfer, Markus Bauer Hermann and Gerald Meyer for make me experiencing in a very good team work. They also show me how a good leader and coworker should be. Furthermore, I would like to thank my colleagues at SVA department; Dr. Hannah Baker for making our office a nice place to work, and Tobias Jeske, Harald Sauff, Marina Krotofil and Dr. Wolfram Reinken for taking care of office facilities.

I also would like to thank Dr. Boontawee Suntisrivaraporn from SIIT for fruitful discussions, Sanarwee Kanluan; my graduate student for her contributions, and Matthias Obst for German version of my abstract.

Living in Germany would never be easy and fun without my friends both in Munich; Dr. Anja Jerichow, Panida and Andreas Licht, Andreas Ginther, Pornphen Tangtisanon and Dr. Ktawut Tappayuthpijarn, and in Hamburg; Patcharaporn Piritburana, Dr. Clark Lee, Thitinat Mangkang, Wilairat Kersten, Chutima Andres, and Dr. Sudchai and Tipaporn Boonto.

Finally I am deeply grateful to my parents and sisters who have supported me for whatever they can especially during my sickness time. No word can truly represent how thankful I feel for them. I love you all.

Hamburg, May 31, 2014

Assadarat Khurat

Contents

1. Introduction

In the beginning of the online services era, most services were single and independent, employing and developing proprietary technology to serve their customers. Hence, the main communication messages for service provisioning of these online services were between users and service providers. Nowadays, there is strong competition in the online market to increase and expand the number of customers. This is an incentive for developing new and better services, which should be fast enough and better serve user demands. Service providers, therefore, attempt to combine their services with other existing services to obtain a new service, so called a composite service. For example, a weather forecast service may combine with a text-to-voice service to become a new service that can provide weather forecasts in voice. Another example is Facebook that provides ways of communication between its users. The users can choose to communicate only within their own community by such ways as posting notes and photos, and sending messages. Combining with games providers by giving, e.g. some personal information and friends' list of its users, Facebook can provide games that the users choose to play only with their friends. This service combination concept is not limited to services in the Internet only but also includes services in the mobile world such as IMS (IP Multimedia Subsystem) based services and location-based services which became one of the most heavily used services due to the proliferation of smart phones. This concept has been encouraged by the European Union, e.g. in a project called SPICE (Service Platform for Innovative Communication Environment) [SPI] supported in the Sixth Framework Programme (FP6). An aim of the SPICE project was to provide a platform supporting the creation of composite services which can be accessible via heterogeneous environments and terminal devices. With this platform a service composition can be created on-the-fly by querying existing services stored in a repository.

Our research stems from the SPICE project. In order to provide a service, most of the online services collect and store personal data of their users such as name, addresses, photos, telephone numbers, etc. For instance, online shopping services require name and address of their customers for product delivery. A user normally registers with more than one online service and is likely to increase the usage which reveals more of the user's data. These collected data include personal data, hence it is essential to consider what services will do with the data to provide privacy protection. This has been increasingly a concern as seen from the enactment of privacy legislations.

Principally, privacy laws govern the processing of personal data that the entities controlling these data are obliged to comply with. Such regulations have been enacted in various countries, e.g. United States, United Kingdom, Germany, etc. In the United States, privacy laws have been published in several areas, e.g. the Health Insurance Portability and Accountability Act (HIPAA) Privacy Rule [Dep02] for health information, the Electronic Communications Privacy Act of 1986 [Pri86] and 2000 [Pri00]

for provisions of access, use, disclosure, interception and privacy protections of electronic communications, and the Privacy Act of 1974 [Pri74] for protecting personally identifiable information about individuals maintained in systems of records by federal agencies. For the countries in the European Union, their national privacy regulations must comply with certain EU Directives. The EU Directives related to privacy are 95/46/EC [Dir95] for protection of individuals on personal data processing in general, 2002/58/EC [Dir02] complementing 95/46/EC in the electronic communications sector and 2006/24/EC [Dir06] amending 2002/58/EC on retention of data for the purpose of the investigation, detection and prosecution of serious crime. In the United Kingdom, the Data Protection Act of 1998 [DPA98] was issued with the intention to conform with the EU Directive 95/46/EC. In Germany, the Federal Data Protection Act or Bundesdatenschutzgesetz (BDSG) [BDS06] implements EU Directive 95/46/EC. The Telecommunications Act (TKG) [TKG04] serves to transpose several EU Directives including the privacy related ones, i.e. Directive 2002/58/EC in the telecommunications area, and the Telemediengesetz (TMG) [TMG07] regulates the electronic information and communication services that are not telecommunication services. At the international level, the OECD (Organisation for Economic Co-operation and Development) has adopted a guideline providing eight basic principles for privacy protection of personal data transferred across countries [OEC80].

In order to comply with the legal requirements, a technical implementation to provide privacy protection is by using policy. There are a few policy languages dedicated for privacy such as P3P (Platform for Privacy Preferences) [CLM+02b, CDE+06], EPAL (Enterprise Privacy Authorization Language) [AHK+03], S4P (SecPAL for Privacy) [BMB10], PPL (PrimeLife Policy Language) [ABV+09] and XACML (eXtensible Access Control Markup Language) [OAS05b]. Note that XACML was not defined specifically for privacy but it is capable of doing so.

According to Article 10 of the EU Directive 95/46/EC that requires the data controller (service provider) to inform the data subject (user) on data collection at least about the identity of the controller and purposes of data usage, P3P (among the above privacy policies) is the only privacy policy being used in real applications that can fulfill this requirement. It describes the data practices of websites in a machine-readable format which enables the users to decide whether to use the service or not after comparing the policy with their preferences. The P3P specification, however, does not define any enforcement mechanism. The data practices stated by websites using P3P policy, therefore, are considered as privacy promises. In order to further provide privacy protection after data have been collected, a real enforcement is needed. An existing technology handling this task is called access control which governs access to resources such as data and services. How the access control system reacts to the access request is again determined according to policies. Among the policy languages mentioned above, XACML, which is one of the most widely used access control language due to its capabilities and implementation availability as open-source, is the most promising one. Our research, hence, focuses on P3P and XACML policies.

1.1. Motivation

The emergence of the composite service paradigm though providing better and more convenient ways for service provisioning introduces some problems at the same time to P3P and access control using XACML. P3P is a W3C standard for expressing privacy policies. It uses a predefined vocabulary to describe policies. However, certain terms in this vocabulary, e.g. *Recipient*, can represent recipient types from the perspective of a single entity, but not in a composite service assembled on the fly. We thus extend the vocabulary so that P3P can be employed in the composite service scenario. In addition, the flexible syntax of the P3P policy language and some combinations of terms in the vocabulary may cause ambiguous and conflicting meaning as criticized in [YLA04, Cra03, SHW02, KSHW03, LYA03]. In a nutshell, P3P has a taxonomy (vocabulary with structure) but not yet an ontology (taxonomy with relationships, constraints and rules). Therefore, we propose to use a semantic web ontology, i.e. OWL (Web Ontology Language), as our solution to mitigate the ambiguities and to check for conflicts. Moreover, when combining several services together, their privacy policies may not be compatible. We also employ the OWL ontology to check for incompatibilities.

The problems affecting an access control system using XACML as its policy language in a composite service environment relate to policy administration and policy enforcement. In general, a service providing information has to specify individual policies for its users. With the composite service environment, the users of the service become other services who may require different parts of the information provided. In order to conform to the privacy principle "limiting collection" that only necessary data for the specified purpose shall be collected, the service has to specify more specific individual policies since different parts of data are needed by different services. This situation is a burden for policy administration. In addition, it could increase the computational expense of the authorization decision process of XACML when the provided information is a hierarchical resource, e.g. an XML document. This is because XACML was designed in such a way that it can evaluate a request for a resource at a time, which means the authorization decision evaluation can be performed either for the whole document or for a single node in it. To allow only the necessary information to be revealed, each node in the required part must be evaluated one by one which can increase the computational overheads of policy enforcement. We propose a mechanism to ease policy administration and at the same time to improve efficiency of XACML authorization evaluation by employing a post-processing concept. Our implementation is shown by a widely used service, location-based service.

Our work stems from the EU project SPICE where technology employed in the IT sector relies on standards. Thus, it is inevitable to take existing standards into consideration. Therefore in this research, we aim for our solution to rely on standards as much as possible.

1.2. Contributions

This thesis introduces solutions improving privacy technologies to provide privacy protection in the composite service paradigm. These technologies are P3P and access control systems using XACML. Our primary contributions are:

- An extended P3P policy that can be used in the composite service scenario.
- A proposed formal semantics for P3P (using OWL) together with defined constraints that can mitigate semantic inconsistency problems by enabling policy verification to check policy validity for internal conflicts of a policy, and can check policy compatibility between policies from different services. The latter verification helps in avoiding a combination of services with incompatible privacy policies which will fail at run-time resulting in recomposing the services again.
- The privacy policies of composite services can be determined for further service compositions employing our policy combining mechanism.
- We improve the efficiency of XACML policy evaluation of the composite service when access is requested for hierarchical resources.

We have published our solutions in [KGA10, KS11] and [KA08] where [KGA10] proposed a formal semantics for P3P based on data–purpose centric interpretation, [KS11] proposed an ontology for P3P using OWL, and [KA08] proposed a mechanism to improve XACML policy evaluation performance.

1.3. Thesis Structure

The organization of the thesis is as follows:

Chapter 2 provides the foundation of this thesis. It describes composite services characteristics with their supporting technologies and analyses existing privacy policy languages.

Chapter 3 describes P3P and XACML policies in detail and analyses their limitations when being employed in a composite service paradigm.

Chapter 4 provides the background knowledge for our proposed solutions that are Semantic Web Ontology for P3P, and Presence Information and Common Policy for access control using XACML.

Chapter 5 presents our solution regarding P3P policies using OWL. The ideas on extending, interpreting and combining P3P policies, and defining its ontology and constraints are elaborated. The experimental results of our implementation of a P3P verification tool are described and analysed. An example showing our combining mechanism is also given. The comparisons between the proposed work and the related work are discussed.

Chapter 6 presents concepts and prototype implementation of our solution for access control system using XACML. The outcome of the implementation is analysed. The comparisons of the proposed work with the others are also evaluated.

Chapter 7 concludes the thesis with our contributions and discusses future work.

2. Preliminaries

In the online world where sensitive information about users is collected and stored, concerns about privacy have increased. In this chapter we describe online services and the existing technologies deployed to preserve user privacy, i.e. privacy policies expressing services' privacy practices and privacy policies for access control. The privacy policies are analysed in the last section.

2.1. Online Composite Services

In online services, end-users normally interact with services (web servers) via a web browser. The web servers may ask for required information from other services to construct a sophisticated answer according to the users' requests. Services that consist of two or more services are called *composite services*. Fig. 2.1 shows a composite service comprising two service providers. A standard technology that has been used to enable the interactions between services is Web Services [ACKM04, New02] which stems from the SOA (Service Oriented Architecture) [Erl06, MB06] concept. SOA defines a service as a function that is well-defined, self-contained and loosely coupled components that can be reused and remixed with other service components both in the same and different domains.

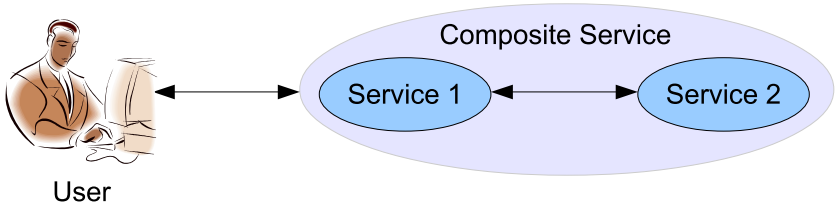


Figure 2.1.: A Composite Service Model

Implementing the SOA concept, Web Services provides a standard way for accessing services from heterogeneous platforms. Web Services technology consists of several standards from W3C and OASIS such as WSDL (Web Services Description Language) [CMRW07, CHM+07], UDDI (Universal Description, Discovery, and Integration) [BCC+04], WS-BPEL (Web Services Business Process Execution Language) [OAS07] and WS-CDL (Web Services Choreography Description Language) [W3C05]. Each service describes in a WSDL file which functions it can perform, how the service can be called, and what data it requires and returns; it publishes this information to a UDDI registry from where a service requester can search and retrieve the WSDL file of the service it wants. Once the required service is found, the service requester sends a request to the service provider in the format as stated in the service provider's WSDL file. Messages between service requester and provider

are conveyed using the SOAP [GHM+07a, GHM+07b] protocol which is an XML-based protocol for exchanging structured information. Fig. 2.2 shows an architecture of Web Services. WS-BPEL and WS-CDL are Web Services standards at the B2B (business-to-business) level. The former describes a workflow of executed Web Services in order from the perspective of one party while the latter describes coordinated interaction between parties involved.

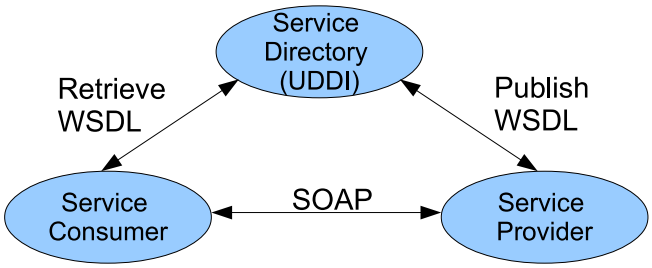


Figure 2.2.: Web Services Architecture

More recently a new technique called “mashup” that stems from Web 2.0 [O’R05], has drawn the attention of researchers [LHSL07, Nes08, NFPS09] as a mechanism for Web-based service composition that uses data from multiple sources to create new services. The Web 2.0 concept does not have a hard boundary. Its core principles include that the web can be a platform to run applications, users are allowed to control their contents, and the web uses new methods to share the contents more easily [Whi09]. Compared to SOA, a mashup is more in the presentation layer as a user interface is provided. In other words, it makes service composition easier for users without deep technology knowledge.

Service composition can be distinguished in two cases. First, when services are considered as software components stored in a repository waiting to be combined with other services. In this case, user data are not yet collected, thus user consent is not concerned. Second, when services are considered as active services where user data are already collected and stored at the services, user consent must be considered for service composition of these services.

2.2. Privacy Policy Languages in Online Services

Many online services collect their users’ personal data such as names, addresses, photos, telephone numbers, etc. The usage of these sensitive data must be carefully controlled in order to preserve the privacy of the users. There are several techniques and concepts for privacy protection established in different domains. For instance, the anonymity concept which aims to make an individual’s identity unknown is employed in, e.g. internet commerce when the users want to buy restricted goods and services. Another technique that supports privacy protection is by using privacy policies to express services’ privacy practices and their enforcement. For the Web Services, though privacy protection is defined as one of the goals of the Web Services Architecture Requirements [ABFG04], there is no specific standard for privacy until now; privacy policy can be contained in the policy assertion of WS-Policy [VOH+07].

Services normally publish their privacy policies in natural language which is usually very long and in legal and technical terms which are quite hard for the end-users to understand. Some naive end-users may agree with this policy without even reading it which may result in unwanted situations later. Privacy policies in machine-readable format thus are specified. They are aimed to be compared with users' privacy preferences as shown in Fig. 2.3. Privacy policies can be employed in two aspects, i.e.

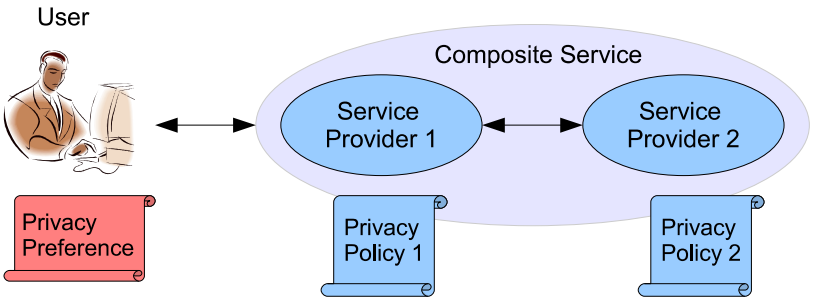


Figure 2.3.: Privacy Policies and Preferences of a Composite Service

prior and after service usage. For the former, the machine-readable privacy policies can be automatically compared with users' privacy preferences. For the latter, after the data were collected, an access control system is used to maintain control over the data usage where policy is employed to express how the system reacts to an access request.

In this section we provide an overview of the existing privacy policy languages which are P3P (Platform for Privacy Preferences), XACML (eXtensible Access Control Markup Language), EPAL (Enterprise Privacy Authorization Language), S4P (SecPAL for Privacy) and PPL (PrimeLife Policy Language).

2.2.1. P3P

The Platform for Privacy Preferences (P3P) [CLM+02b, CDE+06] specification defines P3P policy and the mechanisms for associating policies with Web resources. P3P policy is an XML-based policy language specified by W3C (World Wide Web Consortium) to enable websites to describe their privacy practices in a standard format, XML. Basically a P3P policy describes data usage from a business perspective of the websites on what data will be collected and for which purpose, how long the data may be kept and to whom the data may be distributed to. These privacy practices are interpreted by P3P user agents who will compare them with the pre-defined user preferences written in, e.g. APPEL[1] (A P3P Preference Exchange Language) [CLM02a], which is a language for describing user preferences regarding P3P policy. The comparison result helps the users to (automatically) decide on service usage. With this mechanism, users do not need to read the privacy policies themselves at every site they visit. The P3P specification does not define any enforcement mechanism. The data practices stated by the

[1] APPEL is not a standard. It is only a W3C working draft

websites using P3P policy, therefore, are considered as privacy promises. The P3P specification defines mechanisms to associate P3P policy with URIs transmitted over the HTTP protocol in version 1.0 and with P3P attribute to be used with WSDL in version 1.1.

2.2.2. XACML

XACML (eXtensible Access Control Markup Language) [OAS05b] is an XML-based policy language for access control, standardized by OASIS (Organization for the Advancement of Structured Information Standards). It is one of the most widely used access control policy languages [Com08] due to the fact, among others, that it is a general language and several developers have implemented XACML code and made it available as open-source such as Sun's XACML [Sun] and HERAS [HERa]. Its core specification defines XACML policy and XACML context (request and response messages). XACML policies describe who (subject) is permitted or denied to perform some actions (action) on resources (resource) when conditions are evaluated to true. XACML requests are evaluated against XACML policies in which the evaluation results are replied as XACML responses. XACML defines combining algorithms for the rule level and policy level to derive the result from several rules and policies respectively.

2.2.3. EPAL

EPAL (Enterprise Privacy Authorization language) [AHK+03] is an XML-based policy language established by IBM to express privacy policies. It is designed for internal use of an enterprise. EPAL policy describes who (user-category) is allowed or denied to do some actions (action) on the resources (data-category) for some purposes (purpose) if certain conditions and obligations are fulfilled. It is thus capable of being used in access control systems. EPAL policy is evaluated in order from the top rule. This process stops when a rule is satisfied. Therefore the order of rules in the policy is significant.

2.2.4. S4P

S4P (SecPAL for Privacy) [BMB10] is a declarative language, based on the SecPAL (Security Policy Assertion Language) [BFG10] policy language, to express services' privacy policies and users' privacy preferences. SecPAL is an authorization language established by Microsoft Research, using constraint logic programming, for decentralized systems. Both privacy policies and privacy preferences consist of a set of assertions and a query. S4P extends SecPAL with two modal verbs, "may" and "will". The privacy policies consist of a may-query expressing permissions the services ask from the users, and will-assertions describing promises the services give to the users. For the privacy preferences, there are may-assertions stating what permissions the users allow, and a will-query telling what promises the users ask the services for. The matching between a service's privacy policy and a user's privacy preference are performed in two steps; first all permissions that the service asks (i.e. may-query in the privacy policy) must be allowed by the user (i.e. may-assertions in the privacy preference), and second all promises that the user asks (i.e. will-query in the privacy preference) must be promised by the service as stated in will-assertions in the privacy policy.

2.2.5. PPL

PPL (PrimeLife Policy Language) [ABV+09, TNBN10, Par10] is a policy language that can provide access control and usage control. It is specified by PrimeLife (Privacy and Identity Management in Europe for Life) [Prib] which is the follow-up European project of PRIME [PRIa] (Privacy and Identity Management for Europe) aiming to enable privacy protection in the Internet of the citizen in real life. The PPL language expresses a) policy for access control employing XACML; b) Data Handling Policy (DHP), composed by data controller (service provider) describing what the data controller will do with the data collected from the data subject (user); c) Data Handling Preferences (DHPrefs), composed by data subject (user) describing how the collected data should be treated by the data controller after the data are collected; and d) Sticky policy describing the agreement about the permissions and restrictions on the requested data to control further data usage that the requesting data controller must comply with. This agreement is the result of the matching between data handling policies and data handling preferences. The PPL policy for access control extends XACML with some features such as credentials [CMN09] support, abstractions (abbreviations to represent complex information), recursive conditions (as a fundamental of policy evaluation in delegation cases) and dialog (information on which condition can be disclosed so that the requester knows what information shall be contained in the request). The Sticky policy is employed for controlling data usage of further data controllers [BNP10].

2.3. Privacy Policy Languages Comparison

Among the above privacy policy languages P3P, S4P and PPL are capable of expressing privacy practices. We choose to focus our research on P3P since it is the only one being used in real applications. In addition, S4P is too abstract in that it does not define a vocabulary of e.g. action, purpose of usage and data which makes it hard to be employed.

In the context of access control policy in which XACML, EPAL and PPL are options, we choose XACML to be further investigated. This is because PPL is based on XACML and EPAL is very similar but less expressive than XACML. For example, functions of EPAL's condition are a subset of XACML's functions, a policy level combining mechanism to obtain the result from multiple policies is not specified, and only one rule level combining algorithm (i.e. first-applicable) is employed to resolve the final result of a policy evaluation process. Though, two additional elements are defined, i.e. "vocabulary" describing elements of privacy policies in a specific enterprise so that other enterprises can understand, and "categories" (for user and data). XACML could implement these elements without changing the language itself [And06].

3. P3P and XACML Policies and Analysis in Composite Service Environment

In this chapter, we describe in detail the policy languages P3P (the Platform for Privacy Preferences) and XACML (eXtensible Access Control Markup Language) and then analyse the challenges arising when employing P3P and XACML in composite service environments.

3.1. P3P Policy

The Platform for Privacy Preferences (P3P) [CLM+02b, CDE+06, P3Pd] is an XML-based policy language specified by W3C to enable websites to describe their privacy practices in a standard format. It was specified based on several privacy principles and guidelines such as EU Directive 95/46/EC and the OECD Privacy Protection Guidelines [OEC80]. The main content of the P3P policy language describes what websites may do with the data collected from users such as the purpose of data usage and how long they will keep the data. In practice, users can retrieve the P3P policies of a website via the HTTP protocol as depicted in Fig. 3.1. In this figure, a user wants to visit a website via a client

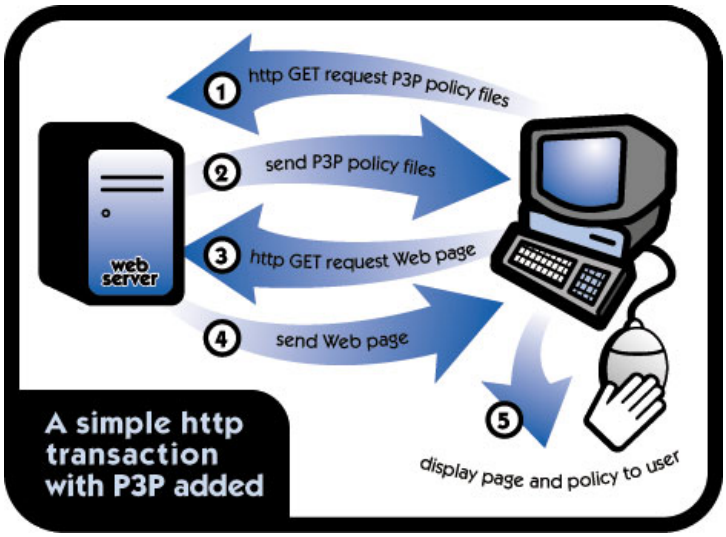


Figure 3.1.: P3P Policy Retrieval Using HTTP Protocol [P3Pc]

embedded with a P3P user agent. The client retrieves P3P policy files of the website. Then the P3P user agent compares the obtained policy files with the user preferences and notifies the user about the comparison result. This helps the user deciding whether to use the website or not. In our example, the user decides to visit the website (note that the decision making process can be automated). Therefore, the client retrieves the web page of the website.

The P3P specification defines the P3P policy language (syntax and predefined vocabularies), mechanisms for associating policies with web resources, e.g. by HTTP, and a standard set of data elements in a hierarchy which all P3P user agents should understand. The P3P policy language was designed for the single service paradigm. The first version of the P3P specification is 1.0 (W3C Recommendation) and the current version is P3P 1.1 (Working Group Note).

The P3P user agents are normally installed at the users' side, e.g. Internet Explorer and Privacy Bird [Pric, BCKM04]. The Privacy Bird helps users to specify their privacy preferences and informs about matchings with the websites' privacy practices via the bird icon. It can be installed with Internet Explorer web browser version IE5, IE6, IE7, and IE8 (WindowsXP 32bit). Another tool that helps users to search websites that match users' preferences is Privacy Finder [Prid]. With search results having valid P3P policies, Privacy Finder can also display some elements of the policies in a table where different colors are used to express data collection and usage. A work proposed to implement server-centric P3P employing database technology for the matching is [AKSX03a]. Some implementations supporting P3P policy creation are IBM P3P policy editor [P3Pb], JRC policy workbench [P3Pe] and P3P builder [P3Pa] (web-based). Cranor et. al have studied P3P deployment on websites in 2003 [CBK03, BCK03] and 2006 [CES+08] and found that P3P adoption was increasing overall. The P3P adoption rate of e-commerce websites is higher than others. This result corresponds to the surveys from Beatty et. al [BRDM07] in 2005 and Jensen et. al [JSJP07] in 2006. However, due to different sources of websites searching, the survey from Reay et. al [RBDM07] differs from the others observing that P3P adoption is stagnant in 2005.

In the following, we present the P3P policy language, P3P criticisms and its existing semantic inconsistencies problems, and analyze the challenges when employing P3P in a composite services environment.

3.1.1. P3P Policy Language

In this section, we describe the syntax of the P3P policy language and the predefined vocabularies.

P3P Syntax

The XML schema for P3P policy documents are shown in Listing 3.1. The main elements of a P3P policy consist of *Entity*, *Access*, *Disputes-Group*, and *Statement* elements. The *Entity* element identifies a legal entity, i.e. the service or website issuing this policy. The *Access* element indicates the ability of users to access their data. The *Disputes-Group* describes the resolution procedure when disputes about these privacy practices occur.

Listing 3.1: XML Schema for P3P Policy Documents [CLM+02b]

```
<!-- ************* POLICY ************* -->
<element name='POLICY'>
 <complexType>
  <sequence>
   <element ref='p3p:EXTENSION' minOccurs='0' maxOccurs='unbounded'/>
   <element ref='p3p:TEST' minOccurs='0'/>
   <element ref='p3p:ENTITY'/>
   <element ref='p3p:ACCESS'/>
   <element ref='p3p:DISPUTES-GROUP' minOccurs='0'/>
```

```
   <element ref='p3p:STATEMENT' maxOccurs='unbounded'/>
   <element ref='p3p:EXTENSION' minOccurs='0' maxOccurs='unbounded'/>
  </sequence>
  <attribute name='discuri' type='anyURI' use='required'/>
  <attribute name='opturi' type='anyURI' use='optional'/>
  <attribute name='name' type='ID' use='required'/>
  <attribute ref='xml:lang' use='optional'/>
 </complexType>
</element>

<!-- ********** STATEMENT ********** -->
<element name='STATEMENT'>
 <complexType>
  <sequence>
   <element ref='p3p:EXTENSION' minOccurs='0' maxOccurs='unbounded'/>
   <element name='CONSEQUENCE' minOccurs='0' type='string'/>
   <choice>
    <sequence>
     <element ref='p3p:PURPOSE'/>
     <element ref='p3p:RECIPIENT'/>
     <element ref='p3p:RETENTION'/>
     <element name='DATA-GROUP' type='p3p:data-group-type'
              maxOccurs='unbounded'/>
    </sequence>
    <sequence>
     <element name='NON-IDENTIFIABLE'/>
     <element ref='p3p:PURPOSE' minOccurs='0'/>
     <element ref='p3p:RECIPIENT' minOccurs='0'/>
     <element ref='p3p:RETENTION' minOccurs='0'/>
     <element name='DATA-GROUP' type='p3p:data-group-type' minOccurs='0'
              maxOccurs='unbounded'/>
    </sequence>
   </choice>
   <element ref='p3p:EXTENSION' minOccurs='0' maxOccurs='unbounded'/>
  </sequence>
 </complexType>
</element>

<!-- ********** PURPOSE ********** -->
<element name='PURPOSE'>
 <complexType>
  <sequence>
   <element ref='p3p:EXTENSION' minOccurs='0' maxOccurs='unbounded'/>
   <choice maxOccurs='unbounded'>
    <element name='current' type='p3p:purpose-value'/>
    <element name='admin' type='p3p:purpose-value'/>
    <element name='develop' type='p3p:purpose-value'/>
    <element name='tailoring' type='p3p:purpose-value'/>
    <element name='pseudo-analysis' type='p3p:purpose-value'/>
    <element name='pseudo-decision' type='p3p:purpose-value'/>
    <element name='individual-analysis' type='p3p:purpose-value'/>
    <element name='individual-decision' type='p3p:purpose-value'/>
    <element name='contact' type='p3p:purpose-value'/>
    <element name='historical' type='p3p:purpose-value'/>
    <element name='telemarketing' type='p3p:purpose-value'/>
    <element name='other-purpose'>
     <complexType mixed='true'>
      <attribute name='required' use='optional' type='p3p:required-value'/>
     </complexType>
    </element>
   </choice>
   <element ref='p3p:EXTENSION' minOccurs='0' maxOccurs='unbounded'/>
  </sequence>
 </complexType>
</element>

<complexType name='purpose-value'>
```

```
 <attribute name='required' use='optional' type='p3p:required-value'
            default='always'/>
</complexType>

<!-- ********** RECIPIENT ************ -->
<element name='RECIPIENT'>
 <complexType>
  <sequence>
   <element ref='p3p:EXTENSION' minOccurs='0' maxOccurs='unbounded'/>
   <choice maxOccurs='unbounded'>
    <element name='ours'>
     <complexType>
      <sequence>
       <element ref='p3p:recipient-description' minOccurs='0'
                maxOccurs='unbounded'/>
      </sequence>
     </complexType>
    </element>
    <element name='same' type='p3p:recipient-value'/>
    <element name='other-recipient' type='p3p:recipient-value'/>
    <element name='delivery' type='p3p:recipient-value'/>
    <element name='public' type='p3p:recipient-value'/>
    <element name='unrelated' type='p3p:recipient-value'/>
   </choice>
   <element ref='p3p:EXTENSION' minOccurs='0' maxOccurs='unbounded'/>
  </sequence>
 </complexType>
</element>

<complexType name='recipient-value'>
 <sequence>
  <element ref='p3p:recipient-description' minOccurs='0'
           maxOccurs='unbounded'/>
 </sequence>
 <attribute name='required' use='optional' type='p3p:required-value'/>
</complexType>

<!-- ********** RETENTION ************ -->
<element name='RETENTION'>
 <complexType>
  <sequence>
   <element ref='p3p:EXTENSION' minOccurs='0' maxOccurs='unbounded'/>
   <choice>
    <element name='no-retention' type='p3p:retention-value'/>
    <element name='stated-purpose' type='p3p:retention-value'/>
    <element name='legal-requirement' type='p3p:retention-value'/>
    <element name='indefinitely' type='p3p:retention-value'/>
    <element name='business-practices' type='p3p:retention-value'/>
   </choice>
   <element ref='p3p:EXTENSION' minOccurs='0' maxOccurs='unbounded'/>
  </sequence>
 </complexType>
</element>

<!-- ************** DATA ************** -->
<complexType name='data-group-type'>
 <sequence>
  <element ref='p3p:EXTENSION' minOccurs='0' maxOccurs='unbounded'/>
  <element name='DATA' type='p3p:data-in-statement' maxOccurs='unbounded'/>
  <element ref='p3p:EXTENSION' minOccurs='0' maxOccurs='unbounded'/>
 </sequence>
 <attribute name='base' type='anyURI'
            use='optional' default='http://www.w3.org/TR/P3P/base'/>
</complexType>

<complexType name='data-in-statement' mixed='true'>
 <sequence minOccurs='0' maxOccurs='unbounded'>
```

```
  <element ref='p3p:CATEGORIES'/>
 </sequence>
 <attribute name='ref' type='anyURI' use='required'/>
 <attribute name='optional' use='optional' default='no' type='p3p:yes_no'/>
</complexType>
```

The *Statement* element describes how the websites/services may deal with the collected data. This element is the part that brings semantic ambiguities. A policy can contain one or more *Statement* elements. The major elements in a *Statement* are *Purpose*, *Recipient*, *Retention* and *Data-Group*. *Purpose* describes the purpose of data usage. *Recipient* expresses to whom the data may be distributed. *Retention* indicates how long the data will be retained. The *Data-Group* element contains a list of data (*Data* elements) which the services may collect and possibly their categories (*Categories* element).

In a P3P statement, there can be one *Retention* value, multiple *Data* elements, and multiple *Purpose* and *Recipient* values. The data standard set is grouped in four sets; *dynamic* (data elements that do not have fixed values e.g. searchtext), *user* (general information about the user), *thirdparty* (data of related third party e.g. delivery address of users' friend) and *business* (data relevant to services). Some *Data* elements can be placed in more than one group. The base data structure of P3P is shown in Appendix A. For better understanding, from now on we will use *Data Categories* instead of mere *Categories*.

In addition, the *Purpose* and *Recipient* elements have an attribute called *Required* that indicates which purpose of data usage, and which data recipient are optional or mandatory. The value of the *Required* attribute can be `always` (default value) or `opt-out` or `opt-in`. For the purpose of data usage, `always` means this purpose value is always required, `opt-out` means the data may be used for this purpose unless the users request not to use the data in this way, and `opt-in` means the data may be used for this purpose only when the users agree so. For the data recipient, `always` means the data will be distributed to this recipient, `opt-out` means the data may be distributed to this recipient unless the users request not to do so, and `opt-in` means the data may be distributed to this recipient only when the users agree to. Similar to the *Purpose* and *Recipient* elements, the *Data* element also has an attribute to indicate which data are optional or mandatory to be collected. This attribute is called *Optional* and the value of this attribute is either `no` (default value) when the data are needed by the service or `yes` when the data are optional.

An excerpt of a P3P policy from Walmart[1] (`http://www.walmart.com/w3c/globalp3ppolicy.xml`) is shown in Fig. 3.2. In the first statement, Walmart allows a user to create an account by collecting the user's login (Email address and password) and the user's home contact information. The collected information will be used to contact the user if agreed and will be stored at Walmart indefinitely. In the second statement, Walmart may conduct a survey and contest via its website or email so it collects user's name, user's login and user's home contact information. If the user chooses to participate, the collected data will be used for research and development and contact purposes and will be stored at Walmart as long as required by the stated purposes.

[1] last checked on 31st December 2012

```
<Policy>
    <Entity>
       <Data-Group>
          <Data ref="#business.name">walmart.com</Data>
            ...
       </Data-Group>
    </Entity>
       ...
    <Statement>
       <Purpose><current/><contact required="opt-in"/></Purpose>
       <Recipient><ours/></Recipient>
       <Retention><indefinitely/></Retention>
       <Data-Group>
          <Data ref="#user.login"/>
          <Data ref="#user.home-info"/>
       </Data-Group>
    </Statement>
    <Statement>
       <Purpose>
          <current/>
          <develop required="opt-in"/>
          <contact required="opt-in"/>
       </Purpose>
       <Recipient><ours/></Recipient>
       <Retention><stated-purpose/></Retention>
       <Data-Group>
          <Data ref="#user.name"/>
          <Data ref="#user.login"/>
          <Data ref="#user.home-info"/>
       </Data-Group>
    </Statement>
        ...
 </Policy>
```

Figure 3.2.: A P3P Policy of Walmart.com

Predefined Vocabularies

The P3P specification defines values of P3P elements. In this section, we only address values of elements in our research focus, i.e. within the *Statement* element. The defined types of the *Purpose*, *Data Categories*, *Retention* and *Recipient* elements are given in Tables 3.1, 3.2, 3.3 and 3.4 respectively.

3.1.2. P3P Criticisms

In the early stage of the P3P framework, two other protocols were also provided; "negotiation" and "data transfer". The former is a mechanism to establish an agreement between websites and users on the collection of the users' data. During the negotiation, a website's privacy practice expressed as a P3P policy is checked by the P3P user agent against the user preferences. The latter mechanism enables automatic data exchange after an agreement is achieved. However after receiving several criticisms related to legal aspects [law98, GR00, Cla98], e.g. that P3P does not address privacy protection completely as laws stated such as some OECD principles ("data quality" that the data should be accurate, complete and kept up-to-date), technical aspects [EJ00, Thi00, LS98, Hog02, Hog03] e.g. the vocab-

Table 3.1.: Descriptions of Values of the *Purpose* Element [CDE+06]

Purpose Values	**Descriptions**
`current`	Completion and Support of Activity For Which Data Was Provided: Information may be used by the service provider to complete the activity for which it was provided, whether a one-time activity such as returning the results from a Web search, or a recurring activity such as providing a subscription service.
`admin`	Web Site and System Administration: Information may be used for the technical support of the Web site and its computer system.
`develop`	Research and Development: Information may be used to enhance, evaluate, or otherwise review the site, service, product, or market.
`tailoring`	Information may be used to tailor or modify content or design of the site where the information is used only for a single visit to the site and not used for any kind of future customization.
`pseudo-analysis`	Pseudonymous Analysis: Information may be used to create or build a record of a particular individual or computer that is tied to a pseudonymous identifier, without tying identified data to the record. This profile will be used to determine the habits, interests, or other characteristics of individuals for purpose of research, analysis and reporting, but not for identifying specific individuals.
`pseudo-decision`	Pseudonymous Decision: Information may be used to create or build a record of a particular individual or computer that is tied to a pseudonymous identifier, without tying identified data to the record. This profile will be used to determine the habits, interests, or other characteristics of individuals to make a decision that directly affects that individual, but not for identifying specific individuals.
`individual-analysis`	Individual Analysis: Information may be used to determine the habits, interests, or other characteristics of individuals and combine it with identified data for the purpose of research, analysis and reporting.
`individual-decision`	Individual Decision: Information may be used to determine the habits, interests, or other characteristics of individuals and combine it with identified data to make a decision that directly affects that individual.
`contact`	Contacting Visitors for Marketing of Services or Products: Information may be used to contact the individual, through a communications channel other than voice telephone, for the promotion of a product or service.
`historical`	Historical Preservation: Information may be archived or stored for the purpose of preserving social history as governed by an existing law or policy.
`telemarketing`	Contacting Visitors for Marketing of Services or Products Via Telephone: Information may be used to contact the individual via a voice telephone call for promotion of a product or service.
`other-purpose`	Other Uses: Information may be used in other ways not captured by the above definitions.

Table 3.2.: Descriptions of Values of the *Data Categories* Element [CDE+06]

Category Values	**Descriptions**
`physical`	Physical Contact Information: Information that allows an individual to be contacted or located in the physical world – such as telephone number or address.
`online`	Online Contact Information: Information that allows an individual to be contacted or located on the Internet – such as email. Often, this information is independent of the specific computer used to access the network. (See the category "Computer Information")
`uniqueid`	Unique Identifiers: Non-financial identifiers, excluding government-issued identifiers, issued for purposes of consistently identifying or recognizing the individual. These include identifiers issued by a Web site or service.
`purchase`	Purchase Information: Information actively generated by the purchase of a product or service, including information about the method of payment.
`financial`	Financial Information: Information about an individual's finances including account status and activity information such as account balance, payment or overdraft history, and information about an individual's purchase or use of financial instruments including credit or debit card information. Information about a discrete purchase by an individual, as described in "Purchase Information", alone does not come under the definition of "Financial Information".
`computer`	Computer Information: Information about the computer system that the individual is using to access the network – such as the IP number, domain name, browser type or operating system.
`navigation`	Navigation and Click-stream Data: Data *passively* generated by *browsing* the Web site – such as which pages are visited, and how long users stay on each page.
`interactive`	Interactive Data: Data *actively* generated from or reflecting *explicit interactions* with a service provider through its site – such as queries to a search engine, or logs of account activity.
`demographic`	Demographic and Socioeconomic Data: Data about an individual's characteristics – such as gender, age, income, postal code, or geographic region.
`content`	Content : The words and expressions contained in the body of a communication – such as the text of email, bulletin board postings, or chat room communications.
`state`	State Management Mechanisms: Mechanisms for maintaining a stateful session with a user or automatically recognizing users who have visited a particular site or accessed particular content previously – such as HTTP cookies.
`political`	Political Information: Membership in or affiliation with groups such as religious organizations, trade unions, professional associations, political parties, etc.
`health`	Health Information: information about an individual's physical or mental health, sexual orientation, use or inquiry into health care services or products, and purchase of health care services or products.
`preference`	Preference Data: Data about an individual's likes and dislikes – such as favorite color or musical tastes.
`location`	Location Data: Information that can be used to identify an individual's current physical location and track them as their location changes – such as GPS position data.
`government`	Government-issued Identifiers: Identifiers issued by a government for purposes of consistently identifying the individual.
`other-category`	Other: Other types of data not captured by the above definitions; A human readable explanation should be provided within these instances.

Table 3.3.: Descriptions of Values of the *Retention* Element [CDE+06]

Retention Values	Descriptions
`no-retention`	Information is not retained for more than a brief period of time necessary to make use of it during the course of a single online interaction.
`stated-purpose`	Information is retained to meet the stated purpose. Sites MUST have a retention policy that establishes a destruction time table. The retention policy MUST be included in or linked from the site's human-readable privacy policy.
`legal-requirement`	Information is retained to meet the stated purpose but the retention period is longer because of a legal requirement or liability. Sites MUST have a retention policy that establishes a destruction time table. The retention policy MUST be included in or linked from the site's human-readable privacy policy.
`business-practices`	Information is retained under a service provider's stated business practices. Sites MUST have a retention policy that establishes a destruction time table. The retention policy MUST be included in or linked from the site's human-readable privacy policy.
`indefinitely`	Information is retained for an indeterminate period of time.

Table 3.4.: Descriptions of Values of the *Recipient* Element [CDE+06]

Recipient Values	Descriptions
`ours`	Ourselves and/or entities acting as our agents or entities for whom we are acting as an agent
`same`	Legal entities following our practices
`delivery`	Delivery services possibly following different practices (may use data other than stated purpose including delivery services with unknown data practices)
`other-recipient`	Legal entities following different practices
`unrelated`	Unrelated third parties: Legal entities whose data usage practices are not known by the original service provider.
`public`	Public fora

ulary used in P3P such as *Purpose* values should be more precise, and social aspects [Coy99, Coy00] e.g. that P3P was designed not to protect users' data privacy but rather to facilitate the websites to collect the data, the P3P Working Group decided to remove these two modules. The reasons behind are that the negotiation process is too complex [EJ00, Pre06] and for the data transfer [Gro99] there was little implementation support from industry and some criticism concerning automatic data exchange that may allow privacy breaches. Lack of actual enforcement according to the stated practices in a P3P policy is one of the significant issues in all the above criticisms.

In addition, several issues related to semantic ambiguities of the P3P policy language were discussed in [YLA04, Cra03, SHW02, KSHW03, LYA03]. Some of them were clarified and addressed in the latest version, v1.1 of the P3P specification [CDE+06]. We describe them in detail in the next section.

3.1.3. Potential Semantic Inconsistencies of P3P Policies

We analyze and categorize causes of P3P policy ambiguities into syntax issues and predefined vocabularies issues as follows.

P3P Policy Syntax

The potential semantic inconsistencies in P3P policies arising from its syntax can be classified in three parts.

Multiple Statements : P3P allows multiple statements in a policy. This syntactic flexibility potentially causes semantic conflicts. For instance, a data item can be mentioned in different statements, assigning different *Retention* values to it. As *Retention* values are mutually exclusive, it is not sensible to allow such multiple values.

Data Hierarchy : P3P defines its base data schema in a hierarchy (shown in Appendix A). It is possible in a policy that two data items, where one is a descendant of the other, are specified with incompatible attribute values. Since the descendant inherits properties of its ancestor, it does not make sense when the ancestor's attributes are more restrictive than the descendant's. An example of this conflict is when the ancestor data is required to be collected but its descendant is optional.

Optional Attributes : P3P defines optional attributes expressing whether *Data*, *Purpose* and *Recipient* elements are required or optional. With some combinations of these attributes, ambiguous meanings arise when, e.g. *Data* element is required while *Purpose* and *Recipient* elements are optional. It is unclear whether or not the data is collected in the first place.

Predefined Vocabularies

With the predefined values of *Purpose, Retention, Recipient* and *Data Categories* elements, some combination of values are semantically inconsistent. Consider, e.g. a statement containing *Purpose* value `develop` and *Retention* value `no-retention`. This introduces a conflict since `develop` is used *"... to enhance, evaluate, or otherwise review the site, service, product, or market ..."* (see Table 3.1) which requires to store data for longer than a brief period of time as stated by `no-retention`. For the inconsistent meaning occurring from the combination of values between *Purpose* and *Data Categories*, the

User Agent Guidelines section in the P3P 1.1 specification already specified some constraints to check these semantic inconsistencies.

Due to the potential semantic inconsistencies mentioned above, there is a need for an explicit formal semantics for the P3P policy language to address this problem.

3.1.4. Analysis of P3P Policy in Composite Services

When employing the P3P policy language, which was designed for single services, in a composite service scenario, some challenges occur. First, existing *Recipient* values cannot cover all of the data recipients in a composite service scenario. P3P defined *Recipient* value to describe types of recipients the data may be distributed to. In a composite service, data necessary for service provisioning are transferred between service members. Thus the services receiving data become data recipients of the data sender service. However, from the definitions of *Recipient* values in Table 3.4, it could be implied that the *Recipient* values can be determined only when the data recipient services are known by the data sender service. In case of dynamic service composition such as in the SPICE project where service descriptions are stored in a repository waiting for being queried to be combined with other services according to e.g. targets and goals of the desired composite services, these other services may include services that the data sender service does not know initially. Therefore, none of the existing *Recipient* values can cope with this situation.

Second, when combining several services together, there might be conflicts between their privacy policies. For example, when the privacy policy of Service A (data sender service) does not allow sending data needed by Service B (data recipient service) in providing a composite service, this composite service cannot function. P3P does not provide any mechanism regarding this issue.

Third, information obtained from P3P policy is insufficient for checking conflicts between privacy policies. To be able to check the feasibility of a successful service combination, the shared data are an important key. In some applications, the shared data are not only data collected from the users but also the processed data generated by the services such as location information. However, the P3P specification does not mention this type of data at all.

Fourth, no mechanism to acquire the privacy policy of composite services is provided. Matching between users' preferences and privacy practices of the websites are normally done at the user agent, i.e. at the user side. Though with the current single service approach this matching for composite services could be done one by one, i.e. checking users' preferences with the P3P policy from each member service individually, having a privacy policy for a composite service can bring additional benefits. The user agents can check the matching only once which is more appropriate compared to the current approach regarding to the computational capability of the end-users.

Last, the potential semantic inconsistencies occurring from syntax and some vocabularies of the P3P policy language as described in Section 3.1.3 will also propagate to the policy of a composite service having that service provider as a member. Therefore, we first check some semantic inconsistencies before performing policy combination.

With these challenges, we propose extensions to the P3P policy language so that it is suitable for

composite services, and mechanisms a) to check potential semantic inconsistencies of the policy, b) to check compatibility between policies of services in composite services and c) to determine the privacy policy of a composite service. These proposed solutions are described in Chapter 5.

3.2. eXtensible Access Control Markup Language

Access control is a concept that is used to restrict the access to resources, e.g. files and services, by an entity. How the access control system reacts to an access request depends on the policies specified beforehand. XACML (eXtensible Access Control Markup Language) [OAS05b] is a policy language for access control standardized by OASIS (Organization for the Advancement of Structured Information Standards). It has been widely utilized in many applications [Com08] such as SPICE [SPI], ESOE [ESO], RAMP [RAM], etc. The current version is XACML 2.0 which we will describe in this section. The recent version, XACML 3.0, is under review at the time of writing. XACML is an XML-based language consisting of XACML policy and XACML context. The XACML policy expresses who is permitted or denied to do some actions on specified resources under some conditions. For example, a facebook user can set his XACML policy to allow only other facebook users in his friend list to see his photos. The XACML context is composed of XACML request and XACML response in which the former contains a request message to access resources and the latter contains result of the corresponding request after being evaluated with appropriate XACML policies.

3.2.1. XACML Data-Flow Model

A data-flow model introducing the major entities of the XACML domain and their interactions was defined as shown in Fig. 3.3. From this model, the initial condition is that the XACML policies are already defined and made available by Policy Administration Points (PAP) (1). Once there is a request to access a resource (2), a Policy Enforcement Point (PEP) forwards the request to the Context Handler in its original format (3). The Context Handler creates an XACML request out of the original one and presents it to a Policy Decision Point (PDP) (4). When some additional attributes are required, the PDP queries these attributes from the Context Handler (5). The Context Handler provides all relevant attributes (i.e. subjects, resources and environments) supplied by Policy Information Points (PIP) and resource content to the PDP (6-10). After evaluating all relevant attributes with applicable policies, the PDP sends an XACML response including the authorization decision back to the Context Handler (11) where a response in the original format is constructed and sent back to the PEP (12). The PEP enforces the policy according to the result (not shown in the figure) and fulfills obligations (13) if there are any. The policy evaluation at the PDP is done for a resource per time, e.g. for a document, an attribute or a service.

3.2.2. XACML Syntax

In this section, the syntax of the XACML language i.e. XACML policy and XACML context will be briefly described.

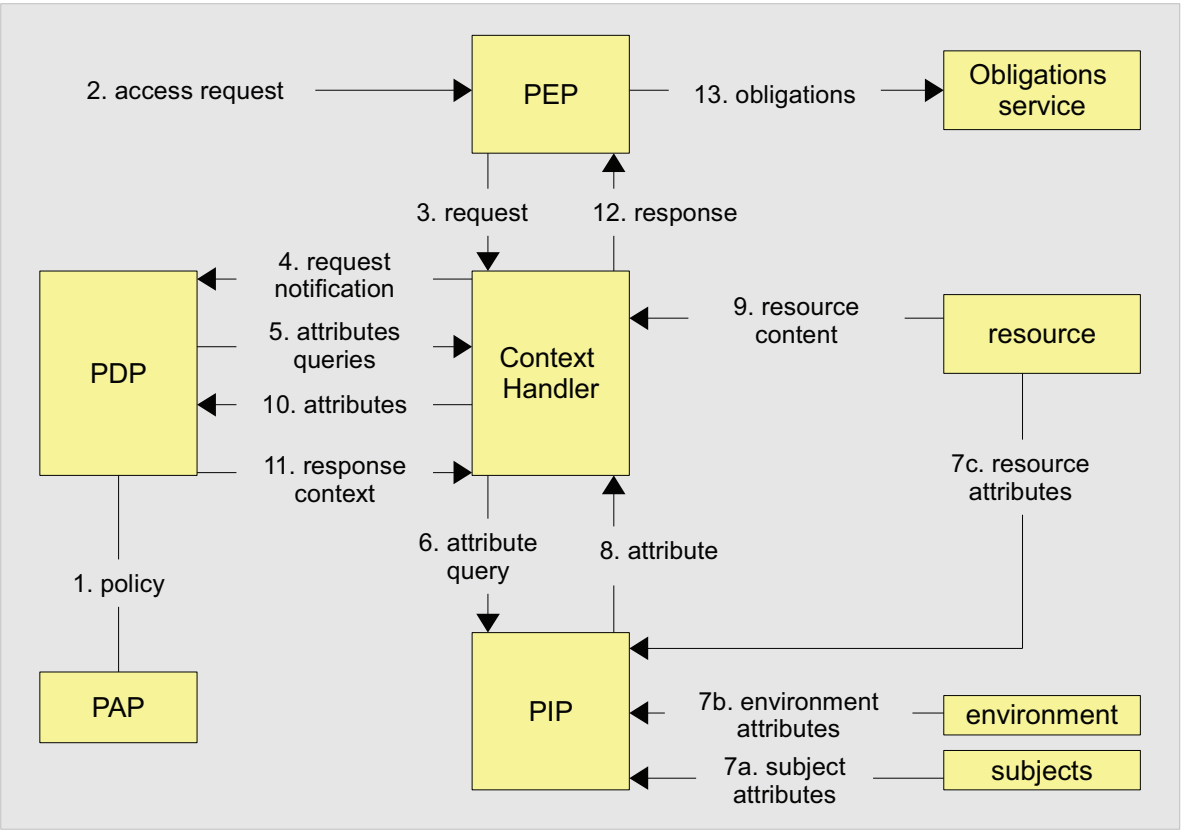


Figure 3.3.: XACML Data-Flow Model [OAS05b]

XACML Policy

Fig. 3.4 depicts the XACML policy structure. The root element of an XACML policy is either a *PolicySet* or a *Policy*. A *PolicySet* can contain a *Target* element, one or more *Policy* and/or *PolicySet* elements. A *Policy* is composed mainly of a *Target* element, a set of *Rule* elements and an optional *Obligations* element. The main components of a *Rule* are a *Target* element, an *Effect* attribute and a *Condition* element. An *Obligations* element contains a set of *Obligation* elements. Each *Obligation* consists of an *ObligationId* attribute identifying the obligation, a *FulfillOn* attribute, and zero or more *AttributeAssignment* elements. The *AttributeAssignment* is composed of an *AttributeId* attribute, a *DataType* attribute and its value.

The *Target* element of *PolicySet*, *Policy*, and *Rule* contains a set of simple conditions (i.e. a match function) that check specified values against the values of *Subject*, *Resource*, *Action*, and *Environment* elements in the request. If these conditions are met, their *PolicySet*, *Policy*, and *Rule* respectively apply to the given request. If the *Target* element is not present, it means that this XACML policy is applicable with all requests. The *Condition* element expresses more complex conditions such as arithmetic functions (e.g. add and subtract), and numeric comparison functions (e.g. greater-than and less-than). If no *Condition* element is present, it means that the *Condition* is evaluated to `true`. *Policy-*

```
<PolicySet PolicyCombiningAlg=" ">
   <Target> ... </Target>
   <Policy RuleCombiningAlg=" ">
      <Target> ... </Target>
      <Rule Effect=" ">
         <Target> ... </Target>
         <Condition> ... </Condition>
      </Rule>
      <Rule> ... </Rule>
         ...
      <Obligations>
         <Obligation ObligationId=" " FulfillOn=" ">
            <AttributeAssignment AttributeId=" " DataType=" ">
               ...
            </AttributeAssignment>
               ...
         </Obligation>
         <Obligation> ... </Obligation>
             ...
      </Obligations>
   </Policy>
   <Policy> ... </Policy>
      ...
</PolicySet>
```

Figure 3.4.: An Example of XACML Policy Structure

Set and *Policy* must have a *PolicyCombiningAlgId* and a *RuleCombiningAlgId* attributes respectively since each *Policy* and *Rule* can return an authorization result. Therefore, a mechanism is needed to derive one final decision result when *PolicySet* contains multiple *Policy* elements and when *Policy* contains multiple *Rule* elements. XACML 2.0 defines four combining algorithms, i.e. Deny-overrides, Permit-overrides, First-applicable, and Only-one-applicable. All combining algorithms can be used at *PolicySet* while only the first three can be used at *Policy*. The *Effect* and *FulfillOn* attributes have a value either `Permit` or `Deny`.

XACML Context

In Fig. 3.5, the XACML context structure is shown with an XACML request on the left and an XACML response on the right-hand side. The XACML request contains information about the request in form of a set of XACML *Attribute* elements classified in *Subject*, *Resource*, *Action*, and *Environment* categories. Each *Attribute* has an identifier, a data type, and a value specified by an *AttributeId* attribute, a *DataType* attribute and a *AttributeValue* element respectively. These *Attribute(s)* are verified with the appropriate policy by the PDP. After the evaluation, the decision result is conveyed via an XACML response as a value of the *Decision* element. In case there is an *Obligation* specified in the policy that is matched with the decision result, this *Obligation* is also contained in the response such that the PEP can act according to it.

```
<Request>
    <Subject>
        <Attribute AttributeId=" " DataType=" ">
           <AttributeValue> ... </AttributeValue>
        </Attribute>
            ...
    </Subject>
    <Resource>
        <Attribute AttributeId=" " DataType=" ">
           <AttributeValue> ... </AttributeValue>
        </Attribute>
            ...
    </Resource>
    <Action>
        <Attribute AttributeId=" " DataType=" ">
           <AttributeValue> ... </AttributeValue>
        </Attribute>
            ...
    </Action>
    <Environment>
        <Attribute AttributeId=" " DataType=" ">
           <AttributeValue> ... </AttributeValue>
        </Attribute>
            ...
    </Environment>
</Request>
```

```
<Response>
   <Result>
      <Decision>
         ...
      </Decision>
      <Status>
         ...
      </Status>
      <Obligations>
         ...
      </Obligations>
   </Result>
</Response>
```

Figure 3.5.: XACML Request (left) and Response (right) Structure

3.2.3. XACML Policy Evaluation

According to the core XACML specification, only one resource can be requested in an XACML request. When there is an authorization request, the Policy Decision Point (PDP) will find out which XACML policy is applicable for the request by checking the *Target* of the root element of the XACML policy. For the applicable policy that has *PolicySet* and *Policy* as its root element, its underlaying *Policy* and *Rule* elements are evaluated further respectively. The evaluation result of a *PolicySet* depends on the evaluation of its *Target*, *Policy* and policy combining algorithm. When there are multiple *Policy* elements in a *PolicySet*, the evaluation result of each *Policy* will be combined into one result according to the policy combining algorithm. The evaluation result of a *Policy* depends on the evaluation of its *Target*, *Rule* and rule combining algorithm. If there are multiple *Rule* elements in a *Policy*, the evaluation result of each *Rule* will be combined into one result according to the rule combining algorithm. The evaluation result of a *Rule* depends on the evaluation of its *Target* and *Condition*. If the *Condition*, which is a Boolean expression, is evaluated to `true`, the value specified in the *Effect* attribute is returned.

In some situations, there is a requirement for some actions that must be performed in conjunction with the policy evaluation, e.g. logging. This process can be specified in an *Obligations* element. Only the *Obligation* whose *FulfillOn* value corresponds to the authorization result (i.e. either `permit` or `deny`) will be performed which will be done at the Policy Enforcement Point (PEP). What to perform

is contained in a set of *AttributeAssignment* elements.

3.2.4. Multiple Resource Profile

The XACML specification defined several profiles so that other technologies can make use of it such as the RBAC (Role-Based Access Control) profile [OAS05a], SAML (Security Assertion Markup Language) profile [OAS05e], etc. In this section, the Multiple Resource profile [OAS05d] that we will use in our work will be described.

In some cases, the requester may want to access several resources implying that several XACML requests and responses would be transmitted between the PEP and the PDP. To avoid this cumbersome procedure, the Multiple Resource profile [OAS05d] was defined. This profile defines three ways of creating a single XACML request context for access to multiple resources, how the result of such a request is returned in one XACML response, and how the result corresponding to the request for an entire hierarchical resource shall be constructed. These three ways are by a) using multiple *Resource* elements in the request, b) identifying nodes (resources) by XPath, and c) identifying nodes (resources) by the "scope" attribute. The "scope" attribute has a value either "Children" or "Descendants" to state whether all child nodes or all sub-nodes respectively of the node specified in the single request are also requested to be accessed. We use the last way to create a single request for access to multiple resources in our work.

Since the PDP can evaluate the access request for one resource per time, there is a suggestion (not required) in this profile to map the single access request for multiple resources to individual requests, where each individual request represents an access request for a node in the requested data part of the XML document.

3.2.5. XML Access Control and XACML

Since XACML also supports protected resources that are XML data, we review some background of research in this area.

XML access control is concerned with limiting access to XML data such that only authorized users can access permitted elements and/or attributes of XML documents. The research in XML access control can be generally considered in two areas [LY09], i.e. a) specification of access control policy and b) efficient enforcement of the XML access control. Access control policy for XML data can be modeled as a set of rules [DVPS00, BCFM00] which are typically specified as who (*subject*) is permitted (*sign* +) or prohibited (*sign* -) to perform actions (*action*) on an element (*object*) and whether this authorization is also applied to sub-elements of this element (*propagation_type*). This rule is specified directly at both the structure (schema level) and content (element level) of XML documents. While [DVPS00] concerns one action type, [BCFM00] proposed several action types. An XML-based policy language from IBM called XACL (XML Access Control Language) [KH00] was proposed which is the first work to introduce actions (provisional authorization) that must be done so that the request will be authorized. This work has been integrated into XACML.

A straight forward way of enforcing XML access control is by evaluating queries at the node-level of an XML document. Typically, XML access control policies use some query languages such as XPath

to specify the requested XML data part, which is a burden on the overall performance. To improve efficiency of XML access control enforcement, [MTKH03, LLLL04, CCLF07] proposed to pre-check the queries whether an input query is fully permitted, fully denied or partially permitted before sending it to an XML engine. The first two types of queries can be quickly processed. While [MTKH03] lets the latter query be processed by an XML engine, [LLLL04, CCLF07] rewrite the query such that it requests only the allowed part. Another method to improve the efficiency of XML access control is view-based [SF02, FCG04, YSLJ02, ZZSZ05]. This scheme creates a view, according to the access control policies, that contains only permitted nodes for each user. When requests arrive, queries will be run against the views. Therefore, there is no need to evaluate the request against the policies anymore. This method, however, has high costs for storage and maintenance when updating. [YSLJ02, ZZSZ05], hence, proposed to shorten the view based on the accessibility locality, i.e. nodes close to each other are likely to have the same accessibility. [SF02] solves another problem when prohibited nodes have permitted child nodes by introducing "dummy" nodes to protect the prohibited nodes.

3.2.6. Analysis of XACML Policy in Composite Services

When an access control system using XACML as its policy language is employed in a composite service environment, policy administration can be more complex and policy enforcement can be less efficient. For example, consider a PresenceInfo service providing presence documents (XML document) and a RestaurantFinder service that uses location data, residing in the presence document [Pet05], to provide a list of restaurants nearby. In general, the PresenceInfo service allows users who are the owners of the presence information to specify policies about who is allowed to access users' presence information and from which elements of their presence document. When the PresenceInfo and RestaurantFinder services are combined, if the users would like to use this combined service, they have to specify a policy for the RestaurantFinder service. However to comply with the "limiting collection" concept that only necessary data shall be revealed, the policy for the RestaurantFinder service must be specified for only specific parts, i.e. location data. Again, when the PresenceInfo service combines with other services who requires other parts of the presence document, the policies for the other services must be individualized for each service.

Furthermore, this situation could increase the computational cost of authorization decision evaluation of XACML. To provide only location data to the RestaurantFinder service, the authorization decision evaluation will be performed for each node in the location data part one by one. This case can highly increase computational overheads of the access control system, especially when the XPath Expression is used to query nodes in an XML document.

We, thus, propose a way to express XACML policies that eases policy administration and a mechanism that enables the process of authorization decision to be evaluated only one time to reduce the computational cost. Our mechanism employs a post-processing concept to filter out the unauthorized data part before sending back the response to the requester. This post-processing also facilitates reducing the load of the PDP which is expected to increase when combining more services.

4. Background on Semantic Web Ontologies, Presence Information and Common Policy Framework

This chapter describes the background knowledge on formal semantic technology, i.e. Semantic Web Ontologies, Presence Information, and Common Policy needed for a better understanding of our proposed solutions.

4.1. Semantic Web Ontologies

In general, ontology is a "philosophical discipline, namely the branch of philosophy which deals with the nature and structure of 'reality' " [SS09]. In computer science, the term has been employed in Knowledge Representation (KR) which is an area in Artificial Intelligence (AI). The main goal of KR is to represent knowledge in a domain in such a way that interpretation of and inference with the knowledge can be automated. One of the most prominent ways of KR to represent the knowledge in a domain is by using logic-based formalisms.

The World Wide Web started as a relatively simple artifact with its contents mostly in form of hypertext and hypermedia. These contents are interwoven together via hyperlinks which help in navigation and browsing. However, the original Web is not equipped with a mechanism to provide "meanings" to its contents and links among these contents. This brings difficulties in sharing content or web resources, which is increasingly occurring in web applications.

The Semantic Web aims at allowing Web information to be more systematically exploited and more accessible by both humans and machines. This aim is (partially) achieved by making the knowledge about the domain concerning a particular website explicit and accessible. RDF (Resource Description Framework) [KC04] specified by W3C, is a language for describing web resources and relationships between them in form of a collection of *triples*, each consisting of a *subject*, a *predicate* (representing a relationship) and an *object*. Though RDF can provide structured annotations of web resources, it does not provide sufficient meaning of the terms used in the annotations. W3C, hence, defined RDF Schema (RDF Vocabulary Description Language) [BG04] as an extension of RDF by adding features (e.g. `rdfs:Class`, `rdfs:subClassOf`, `rdfs:subPropertyOf`[1]) as needed in defining an ontology. RDF Schema is therefore recognized as an ontology language. Due to some limitations of the expressive power of RDF Schema that cannot describe, e.g. cardinality constraints and quantification restrictions, more expressive ontology languages were proposed. Among others, W3C defined a stan-

[1] where rdfs: represents `http://www.w3.org/2000/01/rdf-schema#`.

dard web ontology language called OWL (Web Ontology Language) [BHH+04], which is built on top of RDF and has Description Logics (DLs) as its logical underpinning. The DLs provide a trade-off between expressivity and scalability compared, e.g. to first-order logic that is very expressive but is semidecidable[2] and propositional logic that is decidable but less expressive. The current version is OWL 2.0 [MPSP09, GHM+08].

4.1.1. Web Ontology Language

In one view, Web Ontology Language (OWL) can be seen as a combination of XML (the syntax) and a logical formalism known as Description Logic (the semantics) [BCM+07]. In general, a Description Logic (DL) knowledge base is composed of a set of terminological axioms (TBox) and a set of assertions about *individual* objects (ABox). The TBox describes general knowledge in a domain via *concepts*, *roles*, and relationships between them. The ABox describes specific knowledge about individual objects of the domain.

An ontology is merely a tuple of TBox and ABox. In OWL, the *concepts* and *roles* are called *classes* and *properties* respectively. We use these corresponding terms interchangeably in this thesis. In this section, we give some background on OWL 2.0 sufficient for describing our ontological solution to the P3P problem in Section 5.3.

Concepts/Classes, Roles/Properties and Individuals

To develop an ontology, one begins by defining *individuals* that are a finite set of names representing particular individual objects in the domain, *concepts* that are a finite set of names representing group of individuals that have something in common, and *roles* that are a finite set of names representing relationships between individuals.

Complex concepts can be built from atomic concepts and roles using OWL constructors[3] shown in Table 4.1 where the intersectionOf, unionOf, and complementOf are concept constructors; and the someValuesFrom, allValuesFrom, minCardinality, maxCardinality, and exactCardinality constructors are used to restrict roles.

Besides the complex concepts, we can make statements about how concepts or roles are related to each other by terminological axioms. Some concept and role axiom schemata in OWL syntax, DL syntax and their satisfiability condition are shown in the upper part of Table 4.2. The first three axioms (subClassOf, equivalentClasses, and disjointClasses) are used to establish relationships between concepts and the rest (subPropertyOf, propertyDomain, propertyRange, dataPropertyDomain, and dataPropertyRange) are used to establish relationships and restrictions for roles. The subClassOf axioms can be used to create a concept hierarchy.

A role hierarchy can be created by subPropertyOf axioms. Roles usually connect individuals to individuals where propertyDomain and propertyRange can be used to restrict the first and the second

[2] A logical system is semidecidable if for an argument expressible in the language of the system, there is a procedure that gives a positive answer in a finite time iff the argument is valid, but might not terminate if the argument is not valid. Unlike the logical system that is decidable, its procedure always terminates.

[3] A full list of constructors is available at `http://www.w3.org/TR/2012/REC-owl2-quick-reference-20121211`.

individual to be instances of the specified concept respectively. Roles can have some characteristics[4] to be such as *symmetric*, *reflexive*, *functional*, *transitive*, etc. Since our work uses only *functional* and *transitive*, we describe these two characteristics here. A role can be defined that it is functional and transitive via functionalProperty and transitiveProperty axioms. The former means that each individual can connect to at most one individual via the specified role. The latter means if an individual name a is connected by a role name r to an individual name b that is connected by r to an individual name c, then a is also connected by r to c.

OWL also provides another type of role called *data property* that relates individuals to data values where dataPropertyDomain can be used, similarly as propertyDomain, to restrict individuals connected by a role to be instances of the specified class; and dataPropertyRange to restrict the data values to be in the specified data range. The datatypes used to specify data ranges are mostly from those of XML Schema.

Table 4.1.: Pertinent OWL 2.0 Constructors in Description Logic Syntax

OWL Syntax	DL Syntax	Semantics
Thing	$\top$	$\Delta^{\mathcal{I}}$
Nothing	$\bot$	$\emptyset$
intersectionOf (C D)	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
unionOf (C D)	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
complementOf (C)	$\neg C$	$\Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$
someValuesFrom(r C)	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
allValuesFrom (r C)	$\forall r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y \in \Delta^{\mathcal{I}} : (x,y) \in r^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$
minCardinality (n r)	$\geq n\ r$	$\{x \in \Delta^{\mathcal{I}} \mid n \leq \sharp\{y : (x,y) \in r^{\mathcal{I}}\}\}$
minCardinality (n r C)	$\geq n\ r.C$	$\{x \in \Delta^{\mathcal{I}} \mid n \leq \sharp\{y : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}\}$
maxCardinality (n r)	$\leq n\ r$	$\{x \in \Delta^{\mathcal{I}} \mid n \geq \sharp\{y : (x,y) \in r^{\mathcal{I}}\}\}$
maxCardinality (n r C)	$\leq n\ r.C$	$\{x \in \Delta^{\mathcal{I}} \mid n \geq \sharp\{y : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}\}$
exactCardinality (n r C)	$= n\ r.C$	$\{x \in \Delta^{\mathcal{I}} \mid n = \sharp\{y : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}\}$

Knowledge about individuals can be specified in OWL through *concept assertions* and *role assertions* in the bottom part of Table 4.2 where the former is used to state that an individual is an instance of a particular concept and the latter is used to state that an individual is connected by a role to an individual.

DL Syntax and Semantics

In this section, we provide syntax and semantics of Description Logics corresponding to the previous section. Let N_C be a set of concept names and N_R be a set of role names. The set of concept descriptions is the smallest set such that $\top, \bot$, and each concept name $A \in N_C$ is an concept description, and if C and D are concept descriptions and $r \in N_R$, then $C \sqcap D$, $C \sqcup D$, $\neg C$, $\forall r.C$, $\exists r.C$, $\leq n\ r.C$, $\geq n\ r.C$ and

[4] A full list of role characteristics is available at `http://www.w3.org/TR/2012/REC-owl2-primer-20121211/`

Table 4.2.: Pertinent OWL 2.0 Axioms in Description Logic Syntax

OWL Syntax	DL Syntax	Satisfiability Condition
subClassOf (C D)	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
equivalentClasses (C D)	$C \equiv D$	$C^{\mathcal{I}} = D^{\mathcal{I}}$
disjointClasseses (C D)	$C \sqsubseteq \neg D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$
subPropertyOf (r s)	$r \sqsubseteq s$	$r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$
functionalProperty (r)	$\top \sqsubseteq (\leq 1\ r)$	$\forall x,y,z \in \Delta^{\mathcal{I}} : (x,y) \in r^{\mathcal{I}} \cap (x,z) \in r^{\mathcal{I}} \Rightarrow y = z$
transitiveProperty (r)	Trans(r)	$\forall x,y,z \in \Delta^{\mathcal{I}} : (x,y) \in r^{\mathcal{I}} \cap (y,z) \in r^{\mathcal{I}} \Rightarrow (x,z) \in r^{\mathcal{I}}$
propertyDomain (r C)	$\exists r.\top \sqsubseteq C$	$\forall x,y \in \Delta^{\mathcal{I}} : (x,y) \in r^{\mathcal{I}} \Rightarrow x \in C^{\mathcal{I}}$
propertyRange (r C)	$\top \sqsubseteq \forall r.C$	$\forall x,y \in \Delta^{\mathcal{I}} : (x,y) \in r^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}$
dataPropertyDomain (p C)	$\exists p.\top \sqsubseteq C$	$\forall x,y \in \Delta^{\mathcal{I}} : (x,y) \in p^{\mathcal{I}} \Rightarrow x \in C^{\mathcal{I}}$
dataPropertyRange (p u)	$\top \sqsubseteq \forall p.u$	$\forall x,y \in \Delta^{\mathcal{I}} : (x,y) \in p^{\mathcal{I}} \Rightarrow y \in u^{\mathcal{I}}$
(Role axiom)	$r_1 \circ \ldots \circ r_n \sqsubseteq s$	$\forall y_0,\ldots,y_n \in \Delta^{\mathcal{I}} : (y_0,y_1) \in r_1^{\mathcal{I}} \cap \ldots \cap (y_{n-1},y_n) \in r_n^{\mathcal{I}}$ $\Rightarrow (y_0,y_n) \in s^{\mathcal{I}}$
(Concept assertion)	$a : C$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
(Role assertion)	$(a,b) : r$	$(a^{\mathcal{I}},b^{\mathcal{I}}) \in r^{\mathcal{I}}$

$= n\ r.C$ are concept descriptions.

The set-theoretic semantics of OWL stems from that of Description Logics and are defined by means of *interpretations*. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ comprises a non-empty set $\Delta^{\mathcal{I}}$, called the interpretation domain, and an interpretation function $\cdot^{\mathcal{I}}$ that maps each concept name A to a subset of $\Delta^{\mathcal{I}}$ ($A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$), each role name r to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ (a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$), and each individual name a to an element in the domain ($a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$). The interpretation of complex concepts is shown in the right column of Table 4.1.

The most general form of TBox axioms are $C \sqsubseteq D$ (i.e., subClassOf axiom), where C and D are concept descriptions. This form is called a general concept inclusion (GCI). A TBox $\mathcal{T}$ is a finite set of GCIs. An interpretation $\mathcal{I}$ satisfies a GCI $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. If $\mathcal{I}$ satisfies a GCI, we say that $\mathcal{I}$ is a *model* of this GCI. $\mathcal{I}$ is a model of a TBox $\mathcal{T}$, if it satisfies every GCI in $\mathcal{T}$. An axiom $C \equiv D$ is an abbreviation of two GCIs, $C \sqsubseteq D$ and $D \sqsubseteq C$.

A *role hierarchy* is a finite set of role inclusion axioms (RIAs). A RIA is of the form $r \sqsubseteq s$ (i.e., subPropertyOf axiom), where r and s are role names. An interpretation $\mathcal{I}$ satisfies a role hierarchy $\mathcal{R}$ iff $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$ for each $r \sqsubseteq s$ in $\mathcal{R}$. Such an interpretation is called a model of $\mathcal{R}$. To express propagation of roles, the form $r_1 \circ \ldots \circ r_n \sqsubseteq s$ can be used, where $\circ$ is a composition operator on binary relations. The satisfiability conditions of functional role, transitive role and its domain and range are shown in the right column of Table 4.2.

The *data property* of OWL relates individuals to data values that are in specific datatypes, so called *concrete domains* in Description Logics. When concrete domains $\mathcal{D}$ are used, the set of role names

is partitioned into a set of data properties and a set of ordinary roles. With concrete domains, an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty set $\Delta^{\mathcal{I}}$ i.e. the abstract domain of the interpretation, and an interpretation function $\cdot^{\mathcal{I}}$. The abstract domain and the given concrete domain $\Delta^{\mathcal{D}}$ must be disjoint, i.e., $\Delta^{\mathcal{D}} \cap \Delta^{\mathcal{I}} = \emptyset$. The mapping of the interpretation function is the same as previously described except that each data property p is now interpreted by partial functions from $\Delta^{\mathcal{I}}$ into $\Delta^{\mathcal{I}} \cup \Delta^{\mathcal{D}}$. The satisfiability conditions of domain and range of a data property are shown in the right column of Table 4.2. The range of a data property is described by a data range u instead of a concept description as of an ordinary role.

An ABox is a finite set of assertional axioms. A concept assertion is of the form $a : C$ asserting that a is an instance of C and a role assertion is of the form $(a,b) : r$ asserting that a is connected by r to b, where C is a concept description, r is a role name, and a and b are individual names. An interpretation $\mathcal{I}$ satisfies a concept assertion $a : C$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and satisfies a role assertion $(a,b) : r$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a model of an ABox $\mathcal{A}$ if it satisfies every axiom in $\mathcal{A}$. For a role assertion $(a,b) : r$, when this role is a data property p, the assertion is of the form $(a,v) : p$, where v is a data value (`Literal`).

A knowledge base $\mathcal{K}$ is a pair $(\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ is a TBox and $\mathcal{A}$ is an ABox. An interpretation $\mathcal{I}$ is a model of $\mathcal{K}$ (denoted by $\mathcal{I} \models \mathcal{K}$), if $\mathcal{I}$ is a model of $\mathcal{T}$ (denoted by $\mathcal{I} \models \mathcal{T}$) and $\mathcal{I}$ is a model of $\mathcal{A}$ (denoted by $\mathcal{I} \models \mathcal{A}$).

Reasoning Tasks

Besides formal semantics, which allow for knowledge to be represented in an unambiguous manner, Description Logic systems also provide automated reasoning tools with algorithms for solving inference problems for the ontology. Basic reasoning tasks for a TBox are *subsumption* to verify whether a concept is a subconcept of another concept and *satisfiability* to determine whether a concept is not the empty concept. For an ABox, the basic reasoning tasks are *consistency checking* to check whether an ABox is consistent with respect to a TBox, *instance checking* to determine whether a given individual is an instance of a specified concept, and *relation checking* to verify whether a given relation holds between two individuals. In our P3P ontology solution, all of these reasoning tasks except *relation checking* are involved. These reasoning tasks are formally defined as follows:

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base, where $\mathcal{T}$ is a TBox and $\mathcal{A}$ is an ABox.

Subsumption: A concept C is subsumed by a concept D with respect to $\mathcal{K}$ ($\mathcal{K} \models C \sqsubseteq D$), if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for every model $\mathcal{I}$ of $\mathcal{K}$.

Satisfiability: A concept C is satisfiable with respect to $\mathcal{K}$, if there exists a model $\mathcal{I}$ of $\mathcal{K}$ such that $C^{\mathcal{I}} \neq \emptyset$. In this case, $\mathcal{I}$ is called a model of C.

Consistency Checking: $\mathcal{A}$ is consistent with respect to $\mathcal{T}$, if there is an interpretation $\mathcal{I}$ that is a model of both $\mathcal{A}$ and $\mathcal{T}$. In other word, $\mathcal{K}$ is consistent if it has a model.

Instance Checking: An individual a is an instance of C with respect to $\mathcal{K}$ ($\mathcal{K} \models a : C$), if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ holds for every model $\mathcal{I}$ of $\mathcal{K}$.

Relation Checking: A pair of individuals (a,b) is an instance of a role name r with respect to $\mathcal{K}$ ($\mathcal{K} \models (a,b) : r$), if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$ holds for every model $\mathcal{I}$ of $\mathcal{K}$.

Open-World and Non-Unique Name Assumptions

Two important assumptions of OWL are the *open-world assumption* (OWA) and the *non-unique name assumption* (non-UNA). The first one assumes that what is not known to be true is simply unknown. For example, if there are only assertions about Policy1 that it collects a data item `Name` for purpose `develop`, (POLICY1,NAME):collects and (NAME,DEVELOP):collectedForPurpose. An OWL system cannot conclude that the Policy1 collects data for only `develop` purpose since the first assertion only states that `Name` is a data item that is collected by Policy1. Thus, there can be more data that Policy1 collects. The latter assumption means that two different names for individuals do not necessarily denote different individual objects. For instance, if we have two role assertions, $(a,b) : r$ and $(a,c) : r$, where r is a role name and a, b and c are individual names; and if r is a functional role, then an OWA system can only infer that b and c are mapped to the same object i.e., $b^{\mathcal{I}} = c^{\mathcal{I}}$.

Example

In this section, we would like to give a brief idea of how to describe an invalid policy and assertions in our proposed ontology. A more comprehensive explanation can be found in Chapter 5. Consider the domain of a P3P policy where a policy refers to the data that may be collected. Each of the collected data has an *Optional* attribute. This attribute must have only one value. Therefore, if any policy contains a data item that has the *Optional* attribute with multiple values, this policy is considered invalid. From this information, we can define the concepts (Policy, InvalidPolicy, CollectedData, and DataCollectionOptionality to represent any policies, any invalid policies, any collected data, and any *Optional* attribute values respectively), and roles (collects and optionality to define relationship between concepts Policy and CollectedData meaning that a policy collects some data, and between concepts CollectedData and DataCollectionOptionality meaning that collected data has an *Optional* attribute value respectively). In addition, the DataCollectionOptionality has two subconcepts, Required and Optional that are disjoint, to represent each *Optional* attribute value. The concept of an InvalidPolicy that is defined to be those "policies that collect some data items where their *Optional* attribute has more than one value" is

$$\mathsf{InvalidPolicy} \equiv \mathsf{Policy} \sqcap (\exists \mathsf{collects}.(\mathsf{CollectedData} \sqcap (\geq 2\ \mathsf{optionality}))).$$

An example of concept and role assertions of a Test policy that collects a data item `Name` that has both *Optional* attribute values `Required` and `Optional` is shown below:

$$\begin{aligned}
&\mathsf{TEST\text{–}POLICY} : \mathsf{Policy}\\
&(\mathsf{TEST\text{–}POLICY},\ \mathsf{TEST\text{–}NAME}) : \mathsf{collects}\\
&\mathsf{TEST\text{–}NAME} : \mathsf{Name}\\
&\mathsf{TEST\text{–}NAME} : (\exists \mathsf{optionality.Required})\\
&\mathsf{TEST\text{–}NAME} : (\exists \mathsf{optionality.Optional}).
\end{aligned}$$

4.2. Presence Information

We use presence information as the example of our solution for an access control system using XACML. Therefore, we shortly explain presence information and its data model of the presence document in this section. Presence information is data expressing ability and willingness of a user called "presentity" to communicate with other users such as location data and user status; "on-the-phone", "busy", etc. This information helps in increasing the success of communication establishment by selecting appropriate communication means.

The IETF (Internet Engineering Task Force) specified presence information in RFC 2778 [DRS00]. Its data format and data model are defined in RFC 3863 [SFK+04] and RFC 4479 [Ros06] respectively. The presence data model consists of three components, i.e. "service", "person" and "device". These data components are specified by a *tuple* element, a *person* element, and a *device* element respectively. They contain information called "presence attributes" that describe some aspects of them. The *service* component captures the services that can be used to interact with the presentity. The *person* component contains information about characteristics and status of the presentity. The *device* component contains information about the physical device that helps a user choosing which service to use for communication. There can be more than one *tuple*, *person*, and *device* elements in a presence document. In a presence document, location information can be presented as defined in RFC 4119 [Pet05] and RFC 5491 [WTT09] which is suggested to be placed within the *service* component.

Listing 4.1: Bob's Presence Document

```
<presence entity="pres:bob@example.com">
  <tuple id="eg92n8">
    <status>
      <basic>open</basic>
      <geopriv>
        <location-info>
          <civicAddress>
            <country>US</country>
            <A1>New York</A1>
            <A3>New York</A3>
            <A6>Broadway</A6>
            <HNO>123</HNO>
            <LOC>Suite 75</LOC>
            <PC>10027-0401</PC>
          </civicAddress>
        </location-info>
        <usage-rules>
          <retransmission-allowed>yes</retransmission-allowed>
```

```
          <retention-expiry>2013-06-23T04:57:29Z</retention-expiry>
        </usage-rules>
      </geopriv>
    </status>
    <class>email</class>
    <service-class><electronic/></service-class>
    <status-icon>http://example.com/mail.gif</status-icon>
    <contact>mailto:bob@example.com</contact>
  </tuple>
  <person id="p1">
    <activities>
      <note>Far away</note>
      <away/>
    </activities>
    <mood><angry/></mood>
    <place-is><audio><noisy/></audio></place-is>
    <place-type><residence/></place-type>
    <sphere>bowling league</sphere>
    <note xml:lang="en">Scoring 120</note>
    <timestamp>2012-08-13T16:09:44+05:00</timestamp>
  </person>
  <device id="pc147">
    <user-input>idle</user-input>
    <deviceID>mac:8asd7d7d70</deviceID>
  </device>
</presence>
```

An example of a presence document belonging to a user Bob is shown in Listing 4.1. The *service* component (*tuple*) describes that Bob's current location is at Broadway 123, New York, 10027-0401 (*geopriv*) and it is likely to reach him successfully by Email (*class* and *service-class*) with the address `bob@example.com` (*class*). His status icon is shown at http://example.com/mail.gif. The *person* component describes that he is away (*activities*) and has an angry mood (*mood*). He is in a noisy place (*place-is* and *place-type*) locating at a bowling league (*sphere*). The *device* component shows that Bob's device status is `idle` (*user-input*) and he is using a device with Media Access Control (MAC) address `mac:8asd7d7d70` (*deviceID*)

4.3. Common Policy

An important part of our solution related to an access control system using XACML is *Transformations* which is specified in Common Policy. To facilitate a better understanding of our solution which will be illustrated in Chapter 6, we briefly describe Common Policy [STM+07] and its syntax in this section.

Common Policy is a framework of authorization policies for controlling access to application-specific data, which is standardized by the IETF. It was originally specified for expressing privacy preferences in the combination of presence-specific policies called Presence Authorization Rules [Ros07] (governing access to presence information) and location-specific policies called Geolocation Policy [STC+11] (governing access to location data).

4.3.1. Common Policy Syntax

Generally, an authorization policy expresses conditions that must be fulfilled so that operations can be activated. Fig. 4.1 shows the main Common Policy language structure. The root element of Common Policy is a *RuleSet* which contains zero or more *Rule* elements. In a *Rule*, there are three main entities

```
<RuleSet>
    <Rule id=" ">
        <Conditions>
            <Identity> ... </Identity>
            <Sphere> ... </Sphere>
            <Validity> ... </Validity>
        </Conditions>
        <Actions> ... </Actions>
        <Transformations> ... </Transformations>
    </Rule>
      ...
</RuleSet>
```

Figure 4.1.: Common Policy Language Structure

which are a *Conditions* element, an *Actions* element and a *Transformations* element. Each *Rule* must have an *id* attribute. The main components of the *Conditions* are an *Identity*, a *Sphere* and a *Validity* element. The *Identity* element specifies the authenticated entity. The *Sphere* element can be employed to show the current state (e.g. work, home) of the owner of the requested data. The *Validity* element indicates the validity period of the rule. If all criteria in the *Conditions* part are fulfilled, the operations (so called permissions) determined in the *Actions* and the *Transformations* parts are performed. The *Transformations* part describes the operations that modify the result that is returned to the requester. It helps in reducing the granularity of the requested information.

Since Common Policy is a framework to describe authorization policies governing application-specific data, it does not define the *Transformations* part in detail. Common Policy lets each application specify its own *Transformations*. The *Transformations* of the presence information and location data are specified in Presence Authorization Rules [Ros07] and Geolocation Policy [STC+11] respectively. The example illustrating our proposed idea uses the *Transformations* part of both authorization languages. Therefore we describe the *Transformations* part of them in the following subsections.

4.3.2. Presence Authorization Rules

Presence Authorization Rules is an authorization policy governing access to presence information. It was specified by IETF in RFC 5025 [Ros07].

Transformations

Transformations describes which part of a presence document can be revealed according to permissions. The Presence Authorization Rules defines two types of permissions, i.e. the permissions for data component and the permissions for presence attributes. The permissions for providing access to the data components *service*, *person* and *device* are expressed by *provide-services*, *provide-persons* and *provide-devices* elements respectively. Which presence attributes of these data component are allowed to be accessed is elaborated in the permissions for providing access to presence attributes such as *provide-class*, *provide-sphere*, etc. Table 4.3 shows the former permissions on the left-hand side and their corresponding permissions for presence attributes on the right-hand side.

Table 4.3.: Transformations Permissions of Presence Authorization Rules

Permissions for Data Component Elements	Permissions for Presence Attributes
`provide-services`	`provide-class`, `provide-deviceID`, `provide-privacy`, `provide-relationship`, `provide-user-input`, `provide-note`, `provide-status-icon`, `provide-unknown-attribute`, `provide-all-attributes`
`provide-persons`	`provide-class`, `provide-place-type`, `provide-place-is`, `provide-mood`, `provide-activities`, `provide-sphere`, `provide-privacy`, `provide-note`, `provide-status-icon`, `provide-time-offset`, `provide-unknown-attribute`, `provide-user-input`, `provide-all-attributes`
`provide-devices`	`provide-class`, `provide-user-input`, `provide-note`, `provide-unknown-attribute`, `provide-all-attributes`

An example of *Transformations* is given in Fig. 4.2. This *Transformations* states that the outgoing presence document may contain only the *service* information (*provide-services*) and *person* information (*provide-persons*). The *device* information is forbidden to be revealed since no *provide-devices* element is presented in the *Transformations*. The *service* information may contain contact URI (*contact* element in the presence document) that matches with the specified URI schemes (*service-uri-scheme*) i.e. `sip` and `mailto`. The *all-persons* element means that if there are one or more *person* elements, all of them may be revealed. For the permissions for presence attributes (*provide-activities*, *provide-sphere*, *provide-class*, *provide-place-type*, *provide-privacy*, *provide-status-icon*, and *provide-note*), if their value is `true` the (*activities*, *sphere*, *class*, *place-type*, *privacy*, *status-icon*, and *note* elements are presented respectively in the presence document if there are any; otherwise if their value is `false`, they are forbidden.

4.3.3. Geolocation Policy

Geolocation Policy [STC+11] is an authorization policy to control access to location information. For presence information, the location information resides within *status* element of the presence document as depicted in Listing 4.1. There are two types of location data in presence document, i.e. civic location defined in RFC 4119 [Pet05] and RFC 5139 [TW08] and geodetic location defined in Geography Markup Language (GML) [CDL+04]. The example of presence document given in Listing 4.1 shows the civic location type.

Transformations

For the *Transformations* part of Geolocation Policy, a permission called *provide-location* was defined to indicate that the location data can be disclosed. Which type and granularity of the location data are revealed depends on child elements of the *provide-location* element. These child elements are *provide-civic* for the civic location and *provide-geo* for the geodetic location data type.

For the civic location, there are five levels of granularity specified, i.e. `country`, `region`, `city`,

```
<Transformations>
  <provide-services>
    <service-uri-scheme>sip</service-uri-scheme>
    <service-uri-scheme>mailto</service-uri-scheme>
  </provide-services>
  <provide-persons>
    <all-persons/>
  </provide-persons>
  <provide-activities>true</provide-activities>
  <provide-sphere>true</provide-sphere>
  <provide-class>true</provide-class>
  <provide-place-type>true</provide-place-type>
  <provide-privacy>true</provide-privacy>
  <provide-status-icon>true</provide-status-icon>
  <provide-note>true</provide-note>
    . . .
  <gp:provide-location profile="civic-transformation">
    <lp:provide-civic>building</lp:provide-civic>
  </gp:provide-location>
</Transformations>
```

Figure 4.2.: An Example of Transformations in Common Policy for Presence Information

`building` and `full` which are presented as values of the *provide-civic* element. For the geodetic location, the accuracy of this data is restricted by *radius* attribute, in meters, of the *provide-geo* element. An example of *Transformations* of Geolocation Policy is given in the lower part of Fig. 4.2 which allows the civic location to be revealed in `building` level.

5. Formal Semantics for P3P for Composite Services

According to our analysis of P3P policy in Chapter 3, some challenges occur when employing it in a composite services environment. First, existing *Recipient* values cannot cover all of the data recipients. Second, there might be conflicts between privacy policies of member services but P3P does not provide any mechanism regarding this issue. Third, in order to check conflicts between privacy policies of member services, more information is required. Fourth, no mechanism is provided to obtain the privacy policy of composite services. Last, there are some potential semantic inconsistencies of P3P policy which may propagate to the policy of the composite services.

From these challenges, our solutions are described as follow. In this chapter, we propose to enhance the P3P policy language so that it can be employed in composite services. The added elements are described in Section 5.1. We formalize P3P semantically with data–purpose base and define constraints to a) check semantic inconsistencies of a P3P policy and b) verify compatibility of services in a composite service through their P3P policies in Section 5.2. Our proposed formal semantics for P3P is realized by means of a semantic web ontology as illustrated in Section 5.3. We have performed an empirical evaluation of actual P3P policies with our implementation where the results are shown in Section 5.4. In addition, we propose a combining mechanism to determine privacy policies of composite services in Section 5.5. Finally we discuss some aspects of our proposed solutions and their related work in the last section.

5.1. P3P Extensions for Composite Services

To support composite services, we extend P3P in three points, i.e. a new *Recipient* value, a recipient list for each *Recipient* value, and a "Destruction Time Table" for retention policies. The former supports verifying the compatibility of services in a composite service and the latter two facilitate in combining mechanism to derive privacy policy of a composite service.

5.1.1. Recipient Value

The P3P specification uses the *Recipient* element to describe the types of legal entities the collected data may be distributed to. These data recipients may include, e.g. advertising companies, research institutes, government agencies, and online services. In a composite service, some data are shared between service members. The services receiving data thus become data recipients of the data sender service. The *Recipient* values can be determined when the data recipients are known by the data sender service (original service). With the dynamic characteristics of service composition, there might

be unknown services to be combined. Hence, none of the existing *Recipient* values can be used to capture this situation.

Therefore, we propose a new *Recipient* value called "`prospective-composite-service`" to fill this gap. The new value serves as a placeholder for future services in a composite service meaning that the data associated with this *Recipient* value may be distributed to data recipient services that have potential to be combined with the original service to become a composite service. However, using this *Recipient* value may introduce privacy breaches when users do not want to use the new service or the users do not trust the future services. As described in Section 3.1.1, the P3P specification allows a *Recipient* value to have a *Required* attribute where its value is `always` indicating that the distribution of the data to this recipient is mandatory, `opt-out` meaning that the distribution of the data to this recipient may be occurred unless the users request not to, or `opt-in` meaning that the distribution of the data to this recipient may be occurred only when the users agree to. To avoid privacy breaches when the *Recipient* value `prospective-composite-service` is employed, we propose that this *Recipient* value must have only `opt-in` as its *Required* attribute value which means that there must be a request (users' consent) from the users to use the new composite service before the users' data can be revealed.

With the `prospective-composite-service`, service providers can express their *Recipient* values for prospective services that are not known before enabling service composition. Hence, the new composite service can be successfully created. Moreover with the *Required* attribute value `opt-in`, user privacy can be supported at the same time.

5.1.2. Recipient List

The P3P specification defined *Recipient* values in a coarse-grained range, thus users do not know whom their data may be revealed to which may result in privacy breaches. Therefore, the actual recipients called "explicit-recipient" list for each *Recipient* value is required which enables users to better control their data disclosure.

In addition, the explicit-recipient list can support the determination of the *Recipient* value to derive the privacy policy of composite services. The complete algorithm will be described in Section 5.5. This process is not trivial since the P3P specification was defined in the single service paradigm which makes the determination of the existing *Recipient* value to be done from a single service perspective where each service has its own scope to determine. In order to specify the *Recipient* value of a data element in a composite service, it is necessary to find out the scope of *Recipient* values of the composite service in which the scope of all service members must be taken into account. Since each member has a different scope, some recipients of a *Recipient* value of a particular data of a member service may overlap with other member services which makes it hard to resolve the *Recipient* values of that data of the composite service. For better clarification, we describe the cumbersome determination of the *Recipient* value in the following scenario.

Let a composite service consist of service A and service B where service B has different privacy practices than service A and service B receives a data item from service A. For this data item, service A has one *Recipient* value `other-recipient` meaning that service A might distribute this data to

services that have different privacy practices from service A. From the perspective of service A the services that have different privacy practices from it may include service B, services having the same privacy practices as service B and/or services having different privacy practices from service B. However, after combining service A and service B, the *Recipient* values of this data for this composite service may be changed since the perspectives of both services must be taken into account. If the *Recipient* value for this data of service B is `ours` meaning that service B does not distribute the data to other services, the *Recipient* values of this data for the composite service can be determined into four possible cases i.e. a) `ours`, b) `ours` and `same`, c) `ours` and `other-recipient` and d) `ours`, `same` and `other-recipient`.

The first case occurs when the *Recipient* value `other-recipient` of service A includes only service B. Since now service B is a member service of the composite service, service B shall be classified in the *Recipient* value `ours` of the composite service. Then, no service is included in the *Recipient* value `other-recipient` and therefore it shall be deleted. In fact whether the *Recipient* value `other-recipient` of service A includes service B or not, the *Recipient* value `ours` must be one of the *Recipient* value of the composite service due to the *Recipient* value `ours` of service B. This means that the existence of service B in the *Recipient* value `other-recipient` of service A does not effect the determination of *Recipient* value of the composite service. For the sake of better understanding, we omit the situation when the *Recipient* value `other-recipient` of service A includes service B in the second, third and fourth cases.

The second case occurs when the *Recipient* value `other-recipient` of service A includes only services that have the same privacy practices as service B. Again, since service B is a member service of the composite service, the services that have the same privacy practices as service B shall be classified in the *Recipient* value `same` of the composite service instead. Then the *Recipient* value `other-recipient` includes no services and shall be deleted.

The third case occurs when the *Recipient* value `other-recipient` of service A includes only services that have different privacy practices as service B. Since these services have different privacy practices both from service A and service B, they shall be classified in the same *Recipient* value `other-recipient`.

The last case occurs when the *Recipient* value `other-recipient` of service A includes services that have both the same and different privacy practices as service B. The services having the same privacy practices as service B shall be classified in the *Recipient* value `same` and the services having different privacy privacy practices from service B shall be classified in the same *Recipient* value `other-recipient` of the composite service. This example shows various possible solutions for the determination of the *Recipient* values of the composite service. Which case the actual solution shall be depends on the knowledge about the exact recipient services that the data may be distributed to.

From the requirement of actual recipients and the difficulties of *Recipient* value determination, we propose that except `prospective-composite-service` and `public`, there must be a list of explicit-recipients attached to each *Recipient* value of a particular data item. The list contains recipient identifiers that must be unique in a URI (Uniform Resource Identifier) format. We expect that this extension

is not a difficult task for service providers since they already know the data recipients otherwise they cannot specify their *Recipient* values.

The defined lists for the *Recipient* values `ours`, `same`, `delivery`, `other-recipient` and `unrelated` are called "ours-list", "same-list", "delivery-list", "other-recipient-list" and "unrelated-list" respectively. The ours-list from single service contains its own recipient-id. We use P3P's extension mechanism to extend the explicit-recipient list of the *Recipient* values as below:

```
<xs:element name="explicit-recipient-list" type="p3p12:ExplicitRecipientListType"/>

<xs:complexType name="ExplicitRecipientListType"/>
  <xs:choice minOccurs="1" maxOccurs="unbounded">
    <xs:element name="ours-list" type="p3p12:ExplicitRecipientType"/>
    <xs:element name="same-list" type="p3p12:ExplicitRecipientType"/>
    <xs:element name="delivery-list" type="p3p12:ExplicitRecipientType"/>
    <xs:element name="other-recipient-list" type="p3p12:ExplicitRecipientType"/>
    <xs:element name="unrelated-list" type="p3p12:ExplicitRecipientType"/>
  </xs:choice>
</xs:complexType>

<xs:complexType name="ExplicitRecipientType"/>
  <xs:sequence maxOccurs="unbounded">
    <xs:element name="recipient-id" type="xs:anyURI"/>
  </xs:sequence>
</xs:complexType>

where p3p12 is an abbreviation of http://www.w3.org/2011/P3Pv12.
```

An example of a P3P policy in this part describing that the original service (BookShop) may send the data to services that have the same privacy practices i.e. CatalogExample and Walmart.com is shown as follow:

```
  <Recipient>
    <Extension>
      <explicit-recipient-list>
        <ours-list>
          <recipient-id>urn:recipient:id:BookShop</recipient-id>
        </ours-list>
        <same-list>
          <recipient-id>urn:recipient:id:CatalogExample</recipient-id>
          <recipient-id>urn:recipient:id:Walmart</recipient-id>
        </same-list>
      </explicit-recipient-list>
    </Extension>
  </Recipient>
```

5.1.3. Retention Duration

It is also not trivial to determine *Retention* values for a composite service. For example, though two services specify their *Retention* value as `stated-purpose`, it does not mean that they store the data for the same period of time. It is even harder to combine *Retention* values between `stated-purpose`, `legal-requirement` and `business-practices` from different services since they are incomparable.

A possible way to solve this problem is to know the exact length of the retention time. As shown in Table 3.3, P3P specifies that for `stated-purpose`, `legal-requirement` and `business-practices`, there must be a retention policy establishing a destruction time table included in or linked from the site's human-readable privacy policy but it does not further define what the retention policy looks like.

We thus propose to use the destruction time table to resolve the *Retention* value for composite services by demanding that the P3P policy from each service must provide destruction time in term of retention duration for the retention values `stated-purpose`, `legal-requirement` and `business-practices`. The retention duration expresses how long a service can keep a particular data after using it for the agreed purpose. This duration is in the `xs:duration` datatype [BPM04] format. As stated in the P3P specification for `stated-purpose`, the determination of the retention duration time depends also on the *Purpose* element. The algorithm to resolve the *Retention* value for composite services will be described in Section 5.5.

We also use P3P's extension mechanism to extend the retention duration of *Recipient* values as below:

```
<xs:element name="retentionDuration" type="durationType"/>

<xs:complexType name="durationType"/>
  <xs:choice>
    <xs:element name="stated-purpose-duration"
                type="p3p12:stated-purpose-durationType"/>
    <xs:element name="legal-requirement-duration" type=" xs:duration"/>
    <xs:element name="business-practices-duration" type=" xs:duration"/>
  </xs:choice>
</xs:complexType>

<xs:complexType name="stated-purpose-durationType"/>
  <xs:choice minOccurs="1" maxOccurs="unbounded">
    <xs:element name="current-duration" type="xs:duration"/>
    <xs:element name="admin-duration" type="xs:duration"/>
    <xs:element name="develop-duration" type="xs:duration"/>
    <xs:element name="tailoring-duration" type="xs:duration"/>
    <xs:element name="pseudo-analysis-duration" type="xs:duration"/>
    <xs:element name="pseudo-decision-duration" type="xs:duration"/>
    <xs:element name="individual-analysis-duration" type="xs:duration"/>
    <xs:element name="individual-decision-duration" type="xs:duration"/>
    <xs:element name="contact-duration" type="xs:duration"/>
    <xs:element name="historical-duration" type="xs:duration"/>
    <xs:element name="telemarketing-duration" type="xs:duration"/>
  </xs:choice>
</xs:complexType>
```

The example below shows an excerpt of the extended retention duration which expresses that the original service will store the data for one year for research and development (`develop`) and store the data for six months for marketing contact via telephone (`telemarketing`).

```
<Retention>
  <Extension>
    <retentionDuration>
      <stated-purpose-duration>
        <develop-duration>P1Y</develop-duration>
        <telemarketing-duration>P6M</telemarketing-duration>
      </stated-purpose-duration>
    </retentionDuration>
  </Extension>
</Recipient>
```

5.1.4. User Consent

User consent must be considered when a composite service is composed of active services in which user data are already collected and stored. Once users decide to use an active service, this implies that they agree on their data usage according to the privacy policy of the service and the service must employ user data as it promised. When this service offers a new service by cooperating with other services, it must provide a means (in corresponding to the *Required* attribute value `opt-in` of the *Recipient* value `prospective-composite-service`) such that users can compare their preferences with the privacy policies of those services before giving their consent if they agree on the data usage of the new service.

5.2. Data-purpose Centric Semantics for P3P

Since the P3P policy expresses privacy practices of services against user data, the data are an important key for P3P policy interpretation. The work from Yu et al. [YLA04] then proposed a formal semantics for P3P based on a data–centric view. The semantics of a P3P policy is considered as a relational database. From the main four components (data, purpose, recipient and retention) in a P3P statement, this view means that the data component is the main key of the relations among them. These relations, thus, are between a) data and purpose, b) data and recipient, and c) data and retention. However, this view does not consider the relations between purpose and recipient and retention components. It is arguable that the purpose of data usage is also an important key for P3P policy interpretation, i.e. how long the data should be retained and to whom the data should be distributed depends on the purpose of their usage. For example, data collected for historical purpose should be stored longer than for tailoring websites. Agrawal et al. [AKSX02] discussed this point and proposed a strawman for privacy-aware database systems using a data–purpose centric base. From the main four components as mentioned above, when the data–purpose centric view is applied, there can be two relations *i*) data–purpose and recipient and *ii*) data–purpose and retention.

Comparing between the data–centric and data–purpose centric approaches, the latter one provides more fine-grained interpretation of P3P, e.g. the same data with different purpose can have different *Retention* values while this is not allowed for the data–centric view. Moreover, the data–purpose centric interpretation also matches the EU Directive 95/46/EC Article 6 (e) [95/95] that requires the data to be stored no longer than necessary for the purposes for which the data were collected or for which they are further processed. We, therefore, choose to use both data and purpose as the keys in our formal semantics for P3P.

The following subsections describe constraints to verify possible semantic inconsistencies on internal conflicts of each service and conflicts between services.

5.2.1. Constraints for Single Services

Since the P3P policy language has potential semantic inconsistencies, we define constraints to verify the policy for each service. The constraints to verify semantic inconsistencies caused by the P3P policy syntax are described in the first three subsections. The constraints for pre-defined vocabularies are illustrated in pairs between *Purpose, Recipient, Retention* and *Data Categories* except the pair between *Retention* and *Data Categories* in the last five subsections.

Multiple Statements

P3P policy allows specifying more than one statement in a policy. Thus, multiple values can be assigned to some elements that should have only one value. These values are the values of *Retention* element, and the values of *Optional* and *Required* attributes. Under the data–purpose based interpretation, we define that in a policy there must be only one value of these entities for each data–purpose pair, otherwise the policy is considered invalid.

We define four constraints to check multiple values of *Retention* element, *Optional* attribute, and *Required* attribute of *Purpose* and *Recipient* elements as shown below. The constraint for *Optional* attribute, however, is defined for each data (not for each data–purpose pair as elsewhere) since this attribute only belongs to the *Data* element.

- In a policy, with the same data–purpose pair, its associated *Retention* element must have only one value.
- In a policy, with the same data, its associated *Optional* attribute must have only one value.
- In a policy, with the same data–purpose pair, the *Required* attribute of its *Purpose* element must have only one value.
- In a policy, with the same data–purpose pair, the *Required* attribute of its associated *Recipient* element must have only one value.

Data Hierarchy

The P3P specification defines its base data schema in a hierarchy. It would not make sense if a data item (e.g. *#user.bdate)* has more restrictions than its descendants (e.g. *#user.bdate.ymd.year*). For instance,

if the collection of *#user.bdate* is optional, the collection of *#user.bdate.ymd.year* can be optional or required. But if the collection of *#user.bdate* is required, the collection of *#user.bdate.ymd.year* must be required. These restrictions also apply with *Required* values of *Purpose* and *Recipient* elements and *Retention* value. Therefore, we define that a policy containing data where one is an ancestor of the other, the *Optional* value of the data and the *Required* values of *Purpose* and *Recipient* elements associated with the data must be equal or more restrictive than its ancestor where the data in the latter case must have the same *Purpose* value. In addition, the *Retention* value associated with the data must be equal or less restrictive than its ancestor when the data have the same *Purpose* value. According to their meaning, we define that the *Optional* value `no` is more restrictive than `yes`, the *Required* value `always` is more restrictive than `opt-out`, and `opt-out` is more restrictive than `opt-in`, and the *Retention* value `indefinitely` is less restrictive than `stated-purpose`, `business-practices` and `legal-requirement`; and `stated-purpose`, `business-practices` and `legal-requirement` is less restrictive than `no-retention`.

We define four constraints to check these restrictions for the *Optional* attribute, the *Required* attribute of *Purpose* and *Recipient* elements, and the *Retention* value as shown below:

- In a policy that contains data where one is an ancestor of the other, if the *Optional* value of the ancestor is *no* then the *Optional* value of the other must be *no*.
- In a policy that contains data where one is an ancestor of the other and the data have the same *Purpose* value, the *Required* value of the *Purpose* element associated with the data must be equal or more restrictive than the *Required* value of the *Purpose* element associated with its ancestor.
- In a policy that contains data where one is an ancestor of the other and the data have the same *Purpose* value, the *Required* value of the *Recipient* element associated with the data must be equal or more restrictive than the *Required* value of the *Recipient* element associated with its ancestor.
- In a policy that contains data where one is an ancestor of the other and the data have the same *Purpose* value, the *Retention* value associated with the data must be equal or less restrictive than the *Retention* value associated with its ancestor.

Optional Attributes

Due to unclear meanings of some combinations of optional attributes (*Optional* and *Required*) in the P3P specification, we define a constraint that for each data, if all of its associated *Purpose* values are optional (*Required* value of *Purpose* is `opt-in`), its collection must be optional (*Optional* value is `yes`). This is because for `opt-in`, the services may use the data only when the users explicitely agree to. Thus, before the users explicitely allow a service to use the data, the service should not collect the data.

- In a policy that contains data where all of its associated *Purpose* values have *Required* attributes *opt-in*, the *Optional* attribute of the data must be *yes*.

Purpose and Data Categories

In a policy, some defined *Purpose* values do not correspond to some *Data Categories*. For example, consider a statement containing *Purpose* value `individual-analysis` but no *Data* element from one of the `physical`, `online`, `financial`, `purchase` and `government` data categories. This does not make sense since the *Purpose* value `individual-analysis` requires identifiable data. Four constraints for these conflicts are listed in the User Agent Guidelines [CBK03] which has been appended to the P3P 1.1 specification. We apply these constraints to check potential conflicts of values between *Purpose* and *Data Categories* elements.

- A policy containing *Purpose* value `contact` must contain some *Data* element, associated with this *Purpose*, from one of `physical` and `online` categories.
- A policy containing *Purpose* value `telemarketing` must contain some *Data* element, associated with this *Purpose*, from `physical` category.
- A policy containing *Purpose* value `individual-analysis` must contain some *Data* element, associated with this *Purpose*, from `physical`, `online`, `financial`, `purchase` and `government` categories.
- A policy containing *Purpose* value `individual-decision` must contain some *Data* element, associated with this *Purpose*, from `physical`, `online`, `financial`, `purchase` and `government` categories.

Purpose and Recipient

Some defined *Purpose* values do not correspond to some *Recipient* values. For example, consider a statement containing *Purpose* values `admin`, `develop` or `tailoring` with *Recipient* values other than `ours`. This does not make sense because according to the meaning of these *Purpose* values, at least the *Recipient* value `ours` must be present. We define a constraint below:

- In a policy, when the *Purpose* value is one of `admin`, `historical`, `develop` and `tailoring`, its associated *Recipient* value must be at least `ours`.

Another constraint we define is for the new *Recipient* value `prospective-composite-service`. Since this value describes any services that may be combined with the original service for the purpose of providing a new service, its *Purpose* value must be `current`.

- In a policy, when the *Recipient* value is `prospective-composite-service`, its associated *Purpose* value must be `current`.

Note that a policy with *Recipient* value `public` can raise ethical concerns, e.g. when the *Purpose* values are `admin`, `develop` and/or `tailoring`. Though it is not inconceivable that the services distribute the data in public for the purpose of administration, development and tailoring, in some countries with strong privacy laws such as EU member states, it could be a problem since it is not necessary to expose the data in public to fulfill these data usage purposes.

Purpose and Retention

Some defined *Purpose* values do not correspond to some *Retention* values. For instance, consider a statement containing *Purpose* values `admin` and `develop` with the *Retention* value `no-retention`. This introduces a conflict since these purposes require to store data for longer than a brief period of time as stated by `no-retention`. We define a constraint below:

- In a policy that contains the *Purpose* values `admin`, `develop`, `pseudo-analysis`, `pseudo-decision`, `individual-analysis`, `individual-decision`, `contact`, `telemarketing` and `historical`, its associated *Retention* value must not be `no-retention`.

Similar as above ethical issue may be raised in a policy with *Retention* value `indefinitely` e.g., when the *Purpose* value is `tailoring`. Though it is not impossible that the services store the data forever for the purpose of tailoring their websites, it could be a problem since the services want to keep the data longer than necessary.

Retention and Recipient

In general, the *Retention* and *Recipient* values do not seem to be related to each other since how long the data will be kept does not rely on to whom the data may be sent and vice versa. However, when the *Recipient* value is `public` meaning that the data are disclosed in public such as bulletin boards or commercial CD-ROM, the appropriate *Retention* value is `indefinitely`. Thus, we define a constraint below:

- In a policy when the *Recipient* value is `public`, the *Retention* value of the same data–purpose association must be `indefinitely`.

Recipient and Data Categories

Similar to the previous section, *Recipient* values do not depend on *Data Categories* and vice versa. However, there is one case that does not make sense when there is a *Recipient* value *delivery* but there is no *Data Categories* `physical` in the same statement. This is because the physical address is necessary for product delivery. We define a constraint below:

- A policy containing *Recipient* value `delivery` must contain some *Data* element, associated with this *Recipient* value, from `physical` category.

Prospective-composite-service Optionality

As proposed earlier that the *Recipient* value `prospective-composite-service` must have *Required* attribute value `opt-in`. We check this value by defining a constraint as below:

- In a policy when the *Recipient* value is `prospective-composite-service`, its *Required* attribute value must be `opt-in`.

5.2.2. Constraints for Composite Services

In general, conflicts between two policies/rules occur when their effects on a given object differ, e.g. when for the same data–purpose pair, the values of *Retention* and *Recipient* elements and *Required* and *Optional* attributes from the first P3P policy differ from the values of the second policy. However, in the context of composite services, these discrepancies are not considered as conflicts since each service provides different functionalities and enforces its policy in its own domain. Therefore the usage of the same data in different domain can be different. The constraints we consider here are not about data usages discrepancies but rather about feasibility of successful combination of the composite service.

In a composite service environment, several single services and/or composite services cooperate with each other to perform a new service. Hence, output data of one service can be input data of other services. This may introduce conflicts when a service supposed to send data necessary for composite service provisioning that is an input to another service does not do so which results in the failure of this service combination. Thus, we first have to know which personal data (*shared data*) that services need are transferred between services. Then we further analyse privacy policies of the shared data for both data sender and data recipient services.

The essential shared data can be known e.g. by checking at WSDL file and WS-BPEL description of services described in Section 2.1. Since WSDL states which essential data are input and output data of services and WS-BPEL describes interaction between services, we know that the essential shared data are the matching data between output data of data sender and input data of data receiver. We define the shared data of a service in XML format where its XML schema is as follow:

```
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:p3p="http://www.w3.org/2002/01/P3Pv1"
  targetNamespace="http://www.w3.org/2011/P3Pv12/SharedData">

<import
    namespace="http://www.w3.org/2002/01/P3Pv1"
    schemaLocation="http://www.w3.org/2002/01/P3Pv1.xsd"/>

<xs:element name="SharedData">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="InputSharedData" type="p3p:data-group-type"/>
      <xs:element name="OutputSharedData" type="p3p:data-group-type"/>
     </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

An example of shared data is shown below:

```
<SharedData>
  <InputSharedData>
    <DATA ref="#user.name"/>
  </InputSharedData>
  <OutputSharedData>
    <DATA ref="#user.home-info"/>
  </OutputSharedData>
</SharedData>
```

We define constraints to check the conflicts for each service member of a composite service as follows:

Data collection

The essential shared data are obviously needed for service provisioning, thus their collection is necessary, i.e. the *Optional* attribute value of the data must be `no`. Otherwise, this composite service cannot function. We define a constraint as follow:

- In a policy that contains essential shared data, their *Optional* value must be `no`.

Purpose

For similar reasons as with data collection, the shared data are obviously needed to be used for service provisioning, the *Purpose* values of these data must contain at least `current` value. We define a constraint below:

- In a policy that contains essential shared data, their associated *Purpose* value must be at least `current`.

Recipient

Whether the privacy policy of a service that supposes to send data necessary for composite service provisioning allows these data to be shared, can be checked via the proposed *Recipient* value `prospective-composite-service` of those data. If the *Recipient* value of output shared data of the data sender service does not contain `prospective-composite-service`, the prospective composite service cannot function. This is because the data receiver service will not obtain the essential data for its processing. Therefore we define a constraint as below:

- In a policy that contains shared data that are the output data, their associated *Recipient* value must be at least `prospective-composite-service`.

5.3. Semantic Web Ontology for P3P

As described in the previous section, the original P3P specification fails to capture many characteristics and constraints of what it means by being "a sensible P3P policy". This is by and large due to the lack of an explicit formal semantics provided that the P3P specification has been designed and distributed using mere XML [CLM+02b, CDE+06]. In this section we propose to formalize P3P policy employing an OWL (Web Ontology Language) [BHH+04] ontology to systematically and precisely describe

the structures and constraints inherent in the P3P specification as well as the additional constraints proposed in the previous section.

The reasons why we employ OWL are threefold: a) the logical underpinning of OWL guarantees preciseness of the definitions and constraints, i.e. ambiguity is reduced; b) the OWL language is expressive enough to represent knowledge in the P3P policy domain with reasonable scalability (due to Description Logics); and c) an OWL reasoning tool can be applied to automatically verify consistency (of a particular P3P policy) and compatibility (of member services' P3P policies of a prospective composite service).

The core model and an ontology for P3P policy are described in Section 5.3.1 and 5.3.2. The defined constraints from Section 5.2.1 and 5.2.2, and an actual P3P policy are described in the P3P ontology in Section 5.3.3 and Section 5.3.4 respectively.

5.3.1. Core Ontology Model

Since we have decided to interpret P3P policy on a data–purpose centric base, our proposed ontology for P3P is designed according to this model. We have developed our proposed model as follows:

From the P3P specification described in Section 3.1, a policy comprises at least one *Statement*, which in turn comprises several main elements i.e. *Data*, *Purpose*, *Recipient*, and *Retention*. An obvious modeling choice is to define a class for each of these elements and relate them with appropriate properties/roles as shown in the left figure (Paradigm♯1) of Fig. 5.1 where each rectangular box represents a concept, each unlabeled arrow represents a subClassOf relation, and each labeled arrow with (some) represents a relation with a someValuesFrom constructor. To make sure that the purpose for one data is not grouped with another data, we propose here to flatten the original P3P statements such that each resulting *reified statement* has exactly one *Data* and one *Purpose*.

If the data–centric interpretation of the P3P policy should be adopted [YLA04], then Paradigm♯2 (on the right in Fig. 5.1) could be used. In this paradigm, data collected by a policy may be collected under different conditions. For instance, a birthdate (CollectedData) is retained for indefinitely (Retention) for a historical purpose (Purpose), and also collected with no retention (Retention) only for individual analysis (Purpose). It should be clear that *Retention* and *Purpose* values, among other things, must be grouped correctly, i.e. not intertwined. Otherwise, a misinterpretation may arise, namely "a birthdate is collected with no retention for the historical purpose", which is clearly unintuitive since it is impossible that the services will use the birthdate for historical purpose by retaining it for only users' current session. It means, in the context of P3P, that Purpose, Recipient and Retention must be grouped together under the class CollectionPractice, and each collected data may have relationships to more than one CollectionPractice.

Despite being relatively straightforward, the two modeling paradigms do not sufficiently support query answering and semantic constraint checking, e.g. to check if the *Retention* value is unique for a data–purpose pair. To this end, we design a third modeling paradigm which reflects our data–purpose centric interpretation of the P3P policy, i.e. Paradigm♯3 in Fig. 5.2. This paradigm improves the second paradigm by promoting the Purpose of data up one level in the ontological structure.

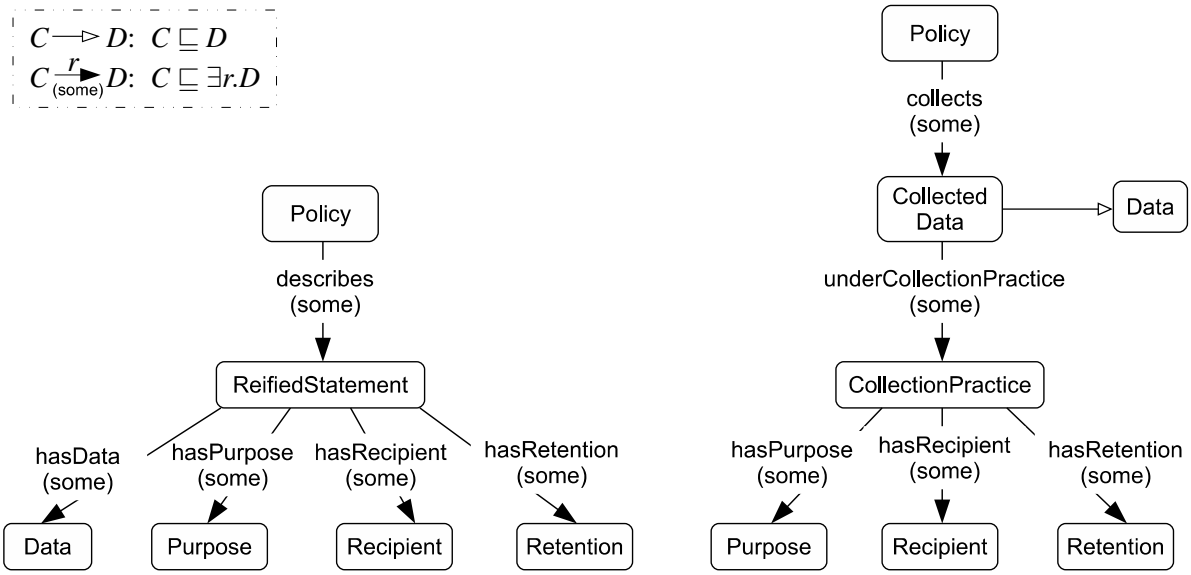


Figure 5.1.: *Paradigm♯1 (left)*; Data, Purpose, and Other Classes at the Same Level in the Ontology. *Paradigm♯2 (right)*; Data-Centric Modeling; Purpose and Other Classes Grouped Together at the Same Level.

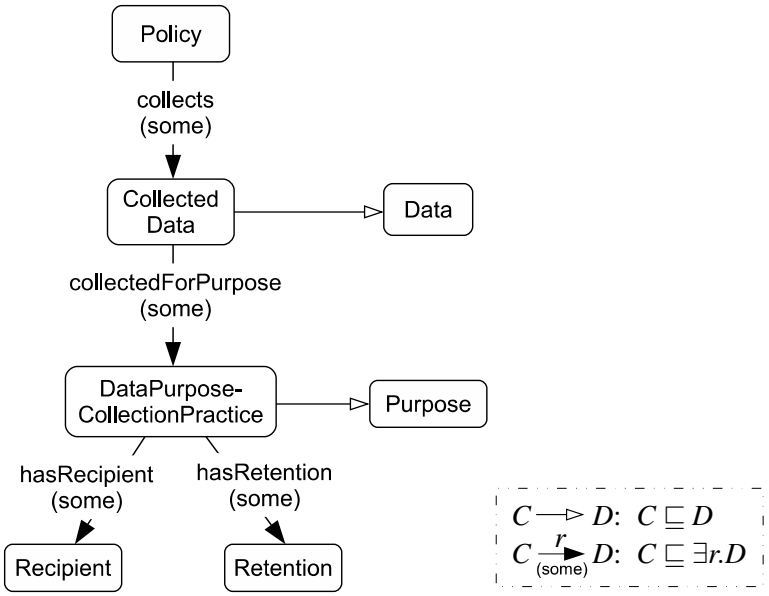


Figure 5.2.: *Paradigm♯3*; Data–Purpose Centric with the Other Elements Grouped Together at the Same Level.

5.3.2. An Ontology for P3P

From the ontology model in Fig. 5.2, we create an ontology for P3P policy as depicted in Fig. 5.3 where each labeled arrow with (only) represents a relation with a allValuesFrom constructor. We design this ontology in such a way that any P3P policy (translated to ontological individuals) can be verified

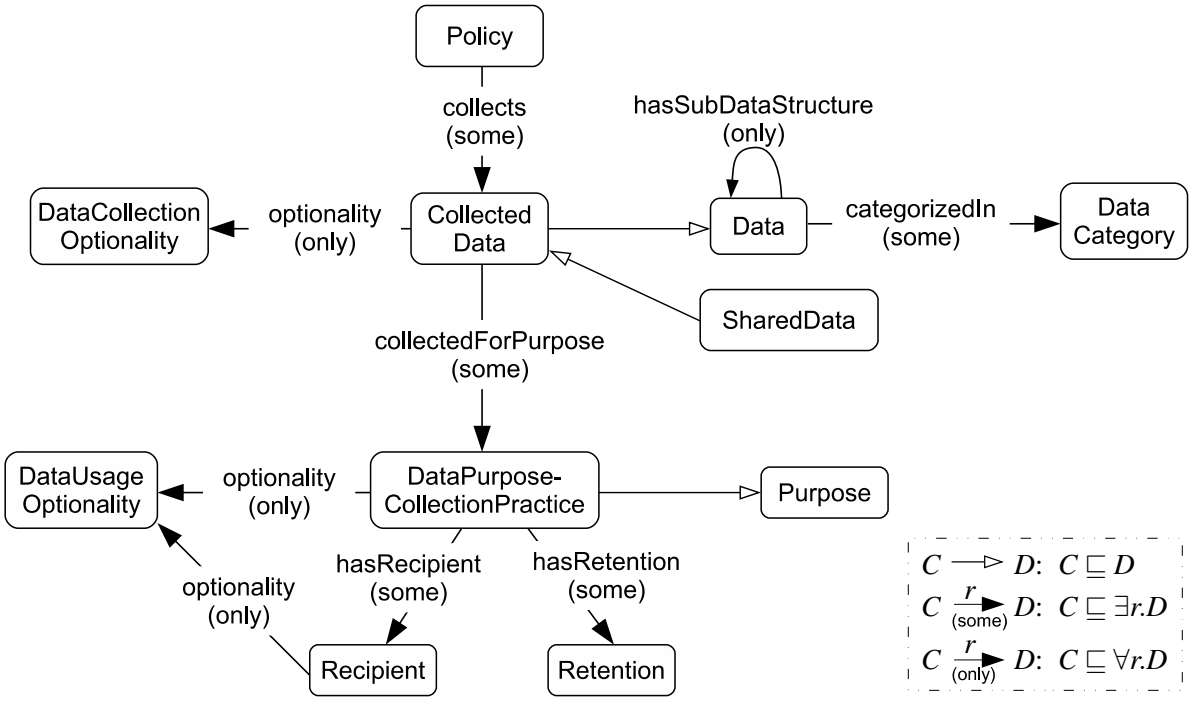


Figure 5.3.: An Ontology Model for P3P Policy

against it (with the help of an OWL reasoner) to check either a) whether a given policy is valid or not, or b) to verify the feasibility of a composite service through the policies of its member services, or both. If the policy is invalid in the former case or it is incompatible in the latter case, our ontology can provide the reason for what is wrong. This can be done by creating special classes that capture those constraint violations instead of specifying logical constraints in the ontology.

According to this design, a given P3P policy can be verified with the help of an OWL reasoner by a standard reasoning task, either *subsumption* or *instance checking*. If the former is used, a class will be created from the P3P policy. This P3P policy is considered invalid or incompatible when it is inferred as a subclass of those special classes. For the latter, individuals will be created from the P3P policy and the policy is considered invalid or incompatible when it is inferred as an instance of any special classes. We choose to employ the latter way in our model since we view a P3P policy as unique information that is obtained from a specific website.

The corresponding OWL definitions of the ontology in Fig. 5.3 in Description Logic syntax are given by α_1–α_5 in Fig. 5.4.

Concepts/Classes

In Fig. 5.2, besides the main P3P elements defined as classes (Policy, Data, Purpose, Recipient and Retention), the classes CollectedData and DataPurpose-CollectionPractice are defined in order to enable our data–purpose centric interpretation; the class CollectedData is a subclass of the class Data to represent the data collected by a policy for some purposes, and the class DataPurpose-CollectionPractice is a subclass of the class Purpose to represent the purposes for which the data are collected.

$$
\begin{array}{rrcl}
\alpha_1 & \text{Policy} & \sqsubseteq & \exists \text{collects}.\text{CollectedData} \\
\alpha_2 & \text{CollectedData} & \sqsubseteq & \text{Data} \sqcap (\exists \text{collectedForPurpose}.\text{DataPurpose–CollectionPractice}) \\
 & & & \quad \sqcap (\forall \text{optionality}.\text{DataCollectionOptionality}) \\
\alpha_3 & \text{DataPurpose–} & \sqsubseteq & \text{Purpose} \sqcap (\exists \text{hasRecipient}.\text{Recipient}) \\
 & \text{CollectionPractice} & & \quad \sqcap (\exists \text{hasRetention}.\text{Retention}) \\
 & & & \quad \sqcap (\forall \text{optionality}.\text{DataUsageOptionality}) \\
\alpha_4 & \text{Recipient} & \sqsubseteq & (\forall \text{optionality}.\text{DataUsageOptionality}) \\
\alpha_5 & \text{Data} & \sqsubseteq & (\forall \text{hasSubDataStructure}.\text{Data}) \sqcap (\exists \text{categorizedIn}.\text{DataCategory}) \\
 & & & \\
\rho_1 & \text{collects} & \sqsubseteq & \text{hasPart} \\
\rho_2 & \text{collectedForPurpose} & \sqsubseteq & \text{hasPart} \\
\rho_3 & \text{hasRecipient} & \sqsubseteq & \text{hasPart} \\
\rho_4 & \text{hasRetention} & \sqsubseteq & \text{hasPart} \\
\rho_5 & \text{hasSubDataStructure} & \sqsubseteq & \text{hasPart} \\
\rho_6 & \text{optionality} & \sqsubseteq & \text{hasPart} \\
\rho_7 & \text{hasPart} \circ \text{hasPart} & \sqsubseteq & \text{hasPart} \\
\rho_8 & \text{hasSubDataStructure} \circ \text{categorizedIn} & \sqsubseteq & \text{categorizedIn}
\end{array}
$$

Figure 5.4.: A Core Extract of the OWL Ontology for P3P Policies

In Fig. 5.3, the classes DataCollectionOptionality and DataUsageOptionality are created to represent *Optional* attributes of *Data* for the former, and of *Purpose* and *Recipient* elements for the latter. The class DataCategory represents *Data Categories* element.

For the classes Data, Purpose, Retention, Recipient, DataCollectionOptionality, DataUsageOptionality and DataCategory, we define subclasses that represent their possible values (not shown in the Fig. 5.3) as described in Section 3.1.1. These subclasses, except the subclasses of Data and DataCategory, are stated to be disjoint as they are intended to have different meanings. Note that we define the values of class DataCollectionOptionality as classes Optional and Required instead of the original values `yes` and `no` for better understanding.

Since the information on which data are shared and whether the shared data are input or output data is needed for constraints checking for composite services, we create a class SharedData to be a subclass of CollectedData to represent the shared data and define classes InputSharedData and OutputSharedData to be subclasses of SharedData to represent the input and output data respectively.

Roles/Properties

In Fig. 5.2, the roles collects, collectedForPurpose, hasRecipient and hasRetention relate individuals of the classes Policy to CollectedData, CollectedData to DataPurpose–CollectionPractice, DataPurpose–CollectionPractice to Recipient, and DataPurpose–CollectionPractice to Retention respectively. From the P3P policy schema which requires that there must be at least one *Data* item, and one *Purpose*, *Recipient* and *Retention* values in a P3P statement, we thus use existential quantification (someValuesFrom) to restrict these relationships (shown in Fig. 5.3 and in α_1–α_3 in Fig. 5.4). In addition, we also define a range for these roles to be CollectedData, DataPurpose–CollectionPractice, Recipient, and Retention classes respectively.

To realize the data–purpose centric viewpoint, it is mandatory that there is precisely one individual representing each data–purpose pair. What this means in our ontology is that CollectedData can have

at most one relationship to each Purpose subclass via the role collectedForPurpose, i.e.

$$(\leq 1\ \text{collectedForPurpose.Admin})$$
$$(\leq 1\ \text{collectedForPurpose.Current})$$
$$(\leq 1\ \text{collectedForPurpose.Historical})$$
$$(\leq 1\ \text{collectedForPurpose.IndividualAnalysis})$$
$$(\leq 1\ \text{collectedForPurpose.IndividualDecision})$$
$$(\leq 1\ \text{collectedForPurpose.PseudoAnalysis})$$
$$(\leq 1\ \text{collectedForPurpose.PseudoDecision})$$
$$(\leq 1\ \text{collectedForPurpose.Tailoring})$$
$$(\leq 1\ \text{collectedForPurpose.Telemarketing})$$
$$(\leq 1\ \text{collectedForPurpose.Contact})$$
$$(\leq 1\ \text{collectedForPurpose.Develop})$$
$$(\leq 1\ \text{collectedForPurpose.OtherPurpose}).$$

Besides the above roles, we define more roles to express relationships between defined classes depicted in Fig. 5.3 as below:

Optional Attributes: In the P3P specification, the *Data*, *Purpose* and *Recipient* elements have attributes named *Optional* for the former and *Required* for the two latter. Therefore, we define a role called optionality to relate these elements to their attribute values, i.e. from the classes CollectedData to DataCollectionOptionality, DataPurpose-CollectionPractice to DataUsageOptionality, and Recipient to DataUsageOptionality respectively (depicted in Fig. 5.3 and in α_2–α_4 in Fig. 5.4). Whenever these attributes are missing from an original P3P document, the default value as described in Section 3.1.1 has to be asserted explicitly in the ontology.

Since the *Required* attribute value of the *Purpose* value `current` and the *Recipient* value `ours` must be default value (`always`), we define that the class Current and Ours must have optionality value only Always, i.e.

$$\text{Current} \sqsubseteq \text{Purpose} \sqcap (\forall \text{optionality.Always})$$
$$\text{Ours} \sqsubseteq \text{Recipient} \sqcap (\forall \text{optionality.Always}).$$

P3P Base Data and Data Categories: The hierarchical structures of data in P3P are organized using an aggregation role hasSubDataStructure. The subclass relation is not used since a sub-data structure can be included by multiple super-data structures that are disjoint. The base data structure of P3P is shown in Appendix A. We define a range of the role hasSubDataStructure to be Data class. To specify a super-data structure that has a sub-data structure, we use a number restriction, maxCardinality. The P3P specification defines allowed categories for their base data. We define role categorizedIn having domain Data and range DataCategory to relate the data to their corresponding data category. The full list of which allowed categories data can belong to is shown in Appendix B.

An example below shows a class description of a super-data structure, Login, which has sub-data

structures Id and Password, and is categorized in UniqueID data category:

$$\begin{aligned} \mathsf{Login} \;\sqsubseteq\; \mathsf{Data} &\sqcap (\exists \mathsf{categorizedIn}.\mathsf{UniqueID}) \\ &\sqcap (\leq 1\ \mathsf{hasSubDataStructure}.\mathsf{Id}) \\ &\sqcap (\leq 1\ \mathsf{hasSubDataStructure}.\mathsf{Password}). \end{aligned}$$

To allow reasoning that a data D is subclass of its category C, we add a Class Inclusion:

$$(\exists \mathsf{categorizedIn}.C) \;\sqsubseteq\; C$$

for each data category C. This design enhances the modeling in [DDFP04, Hog05] by adding the role inclusion axiom ρ_8. In the presence of this axiom, any category of a sub-data structure is automatically propagated to its super-data structure.

Role Hierarchy: At the bottom of Fig. 5.4 are role axioms required for reasoning. The role hierarchy (ρ_1–ρ_6) specifies that hasPart is a superrole of every other role, and ρ_7 specifies that it is transitive. For convenience and conciseness, hasPart is often used to reach a nested part in the policy structure. Assume, for instance, that we are interested in referring to policies that collect some data for some purpose with retention value Indefinitely. A straightforward description would be:

$$(\exists \mathsf{collects}.(\exists \mathsf{collectedForPurpose}.(\exists \mathsf{hasRetention}.\mathsf{Indefinitely}))).$$

This cumbersome description can be shortened with our role hasPart to:

$$(\exists \mathsf{hasPart}.\mathsf{Indefinitely}).$$

Enhanced P3P Elements: In Section 5.1 we have extended P3P elements with *explicit-recipient list* and *retention duration*. These two elements are actually the concrete values of the *Recipient* and *Retention* values respectively. Therefore we relate the *Recipient* values to the explicit-recipients with the data property hasRecipientID and relate the *Retention* values with the data property retentionDuration. Since the explicit-recipients' value must be unique, we choose `Literal` as its datatype. The datatype of retention duration value is defined to be `xs:duration`. However, OWL 2 does not support this datatype. Thus, we decide to assign the retention duration value in seconds in which the proper datatype of this value is `xsd:unsignedLong`.

5.3.3. Constraints

With Paradigm♯3 (data–purpose centric), it is possible to realize the constraints pointed out in Section 5.2. There are generally two approaches for checking constraint violation in any given P3P policy. The first, and probably the most intuitive, approach is to translate each constraint into a logical expression (so called logical constraint) which then forms (part of) a definition in the ontology. For instance, the constraint to check multiple retention values for the same data–purpose pair can be specified on the role

hasRetention by using maxCardinality, i.e. ($\leq$ 1 hasRetention) or by specifying that role hasRetention is functional. Use of ontology reasoning on these logical constraints can help detecting inconsistencies whenever a P3P policy under consideration is invalid for some reasons. In other words, violation of any of the P3P constraints automatically triggers logical inconsistency in the ontology. This appeares quite a natural way to go at first, but it is deemed insufficient relative to our original aim to be able to explain what is wrong in a P3P policy. This is due to the fact that, once an ontology is inconsistent, nothing can be inferred about it.

An alternative approach is to allow arbitrary ontological structures depicted by a P3P policy of interest. For instance, instead of constraining ($\leq$ 1 hasRetention), we just omit it. Some parts of the ontology however may violate the constraints but will not raise logical inconsistency. We then define special classes with specific definitions to capture these constraint violations e.g. a class that capture a policy that has more than one retention values for a data–purpose pair. This approach not only can detect whether a given policy is valid, but also can provide the reasons for its invalidity.

OWL 2 is expressive enough to express special classes from all of the defined constraints in Section 5.2. We show these special classes which can be conceived as queries in the following two subsections. Since all classes can be expressed and our task is to verify a P3P policy according to these classes, query language is not used.

Constraints for Checking Invalid P3P Policies for Single Services

In our approach for checking P3P validity, the nineteen constraints for single services defined in Section 5.2.1 are translated to nineteen special OWL classes. We describe these classes (see definitions β_1–β_{19} in Fig. 5.5) as follows:

InvalidPolicy-MultipleRetention (β_1) represents the class of invalid P3P policies that have multiple retention values for the same data–purpose collection practice. Multiple retention values are captured with the help of *at-least* number restrictions.

InvalidPolicy-MultipleDataOpt (β_2) represents the class of invalid P3P policies that have two conflicting optionality values for the same data item, i.e. both `yes` and `no`.

InvalidPolicy-MulitplePurposeOpt (β_3) represents the class of invalid P3P policies that have two or more conflicting optionality values for the same data–purpose collection practice.

InvalidPolicy-MultipleRecipientOpt (β_4) represents the class of invalid P3P policies that have two or more conflicting optionality values for the same *Recipient* of a data item.

InvalidPolicy-DH-DataOpt (β_5) represents the class of invalid P3P policies where the optionality value of some data is more restrictive than the optionality value of their descendant data.

InvalidPolicy-DH-PurposeOpt (β_6) represents the class of invalid P3P policies where the optionality value of some data–purpose collection practice is more restrictive than the optionality value of the data–purpose collection practice of a descendant. In Fig. 5.5, an excerpt of axioms for

β_1 InvalidPolicy–MultipleRetention $\equiv$ Policy $\sqcap$ ($\exists$hasPart.(DataPurpose–CollectionPractice $\sqcap$ (≥ 2 hasRetention)))

β_2 InvalidPolicy–MultipleDataOpt $\equiv$ Policy $\sqcap$ ($\exists$collects.(CollectedData $\sqcap$ (≥ 2 optionality)))

β_3 InvalidPolicy–MultiplePurposeOpt $\equiv$ Policy $\sqcap$ ($\exists$hasPart.(DataPurpose–CollectionPractice $\sqcap$ (≥ 2 optionality)))

β_4 InvalidPolicy–MultipleRecipientOpt $\equiv$ Policy $\sqcap$ ($\exists$hasPart.(Recipient $\sqcap$ (≥ 2 optionality)))

β_5 InvalidPolicy–DH–DataOpt $\equiv$ Policy $\sqcap$ ($\exists$collects.(CollectedData $\sqcap$ ($\exists$optionality.Required) $\sqcap$ ($\exists$hasSubDataStructure.($\exists$optionality.Optional))))

β_6 InvalidPolicy–DH–PurposeOpt $\equiv$ Policy $\sqcap$ ($\exists$collects.(CollectedData $\sqcap$ (((($\exists$hasPart.(Admin $\sqcap$ ($\exists$optionality.OptIn))) $\sqcap$ ($\exists$subDataStructureOf.($\exists$hasPart.(Admin $\sqcap$ ($\exists$optionality.(Always $\sqcup$ OptOut)))))) $\sqcup$ (($\exists$hasPart.(Admin $\sqcap$ ($\exists$optionality.OptOut))) $\sqcap$ ($\exists$subDataStructureOf.($\exists$hasPart.(Admin $\sqcap$ ($\exists$optionality.Always)))))) $\sqcup$ ...))

β_7 InvalidPolicy–DH–RecipientOpt $\equiv$ Policy $\sqcap$ ($\exists$collects.(CollectedData $\sqcap$ (((($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.(Same $\sqcap$ ($\exists$optionality.OptIn))))) $\sqcap$ ($\exists$subDataStructureOf.($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.(Same $\sqcap$ ($\exists$optionality.(Always $\sqcup$ OptOut))))))))) $\sqcup$ (($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.(Same $\sqcap$ ($\exists$optionality.OptOut))))) $\sqcap$ ($\exists$subDataStructureOf.($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.(Same $\sqcap$ ($\exists$optionality.Always))))))))) $\sqcup$ ...))

β_8 InvalidPolicy–DH–Retention $\equiv$ Policy $\sqcap$ ($\exists$collects.(CollectedData $\sqcap$ (((($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.NoRetention))) $\sqcap$ ($\exists$subDataStructureOf.($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.(BusinessPractices $\sqcup$ StatedPurpose $\sqcup$ LegalRequirement $\sqcup$ Indefinitely)))))) $\sqcup$ (($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.(StatedPurpose $\sqcup$ BusinessPractices $\sqcup$ LegalRequirement)))) $\sqcap$ ($\exists$subDataStructureOf.($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.Indefinitely)))))) $\sqcup$ ...))

β_9 InvalidPolicy–OptionalAttributes $\equiv$ Policy $\sqcap$ ($\exists$collects.(($\exists$optionality.Required) $\sqcap$ ($\forall$collectedForPurpose.($\exists$optionality.OptIn))))

β_{10} InvalidPolicy–Purpose–DataCat1 $\equiv$ Policy $\sqcap$ ($\exists$collects.(($\exists$categorizedIn.($\neg$Online $\sqcap$ $\neg$Physical)) $\sqcap$ ($\exists$collectedForPurpose.Contact)))

β_{11} InvalidPolicy–Purpose–DataCat2 $\equiv$ Policy $\sqcap$ ($\exists$collects.(($\exists$categorizedIn.($\neg$Physical)) $\sqcap$ ($\exists$collectedForPurpose.Telemarketing)))

β_{12} InvalidPolicy–Purpose–DataCat3 $\equiv$ Policy $\sqcap$ ($\exists$collects.(($\exists$categorizedIn.($\neg$Financial $\sqcap$ $\neg$Government $\sqcap$ $\neg$Online $\sqcap$ $\neg$Physical $\sqcap$ $\neg$Purchase)) ($\exists$collectedForPurpose.IndividualDecision)))

β_{13} InvalidPolicy–Purpose–DataCat4 $\equiv$ Policy $\sqcap$ ($\exists$collects.(($\exists$categorizedIn.($\neg$Financial $\sqcap$ $\neg$Government $\sqcap$ $\neg$Online $\sqcap$ $\neg$Physical $\sqcap$ $\neg$Purchase)) ($\exists$collectedForPurpose.IndividualAnalysis)))

β_{14} InvalidPolicy–Purpose–Recipient1 $\equiv$ Policy $\sqcap$ ($\exists$hasPart.((Admin $\sqcup$ Develop $\sqcup$ Historical $\sqcup$ Tailoring) $\sqcap$ ($\forall$hasRecipient.($\neg$Ours))))

β_{15} InvalidPolicy–Purpose–Recipient2 $\equiv$ Policy $\sqcap$ ($\exists$hasPart.($\neg$Current $\sqcap$ ($\exists$hasRecipient.ProspectiveCompositeService)))

β_{16} InvalidPolicy–Purpose–Retention $\equiv$ Policy $\sqcap$ ($\exists$hasPart.(($\exists$hasRetention.NoRetention) $\sqcap$ (Admin $\sqcup$ Develop $\sqcup$ PseudoAnalysis $\sqcup$ PseudoDecision $\sqcup$ Historical $\sqcup$ IndividualAnalysis $\sqcup$ IndividualDecision $\sqcup$ Telemarketing $\sqcup$ Contact)))

β_{17} InvalidPolicy–Retention–Recipient $\equiv$ Policy $\sqcap$ ($\exists$hasPart.(($\exists$hasRetention.($\neg$Indefinitely)) $\sqcap$ ($\exists$hasRecipient.Public)))

β_{18} InvalidPolicy–Recipient–DataCat $\equiv$ Policy $\sqcap$ ($\exists$collects.(($\exists$categorizedIn.($\neg$Physical)) $\sqcap$ ($\exists$collectedForPurpose.($\exists$hasRecipient.Delivery))))

β_{19} InvalidPolicy–ProspectiveCSOpt $\equiv$ Policy $\sqcap$ ($\exists$hasPart.(ProspectiveCompositeService $\sqcap$ ($\exists$optionality.(OptOut $\sqcup$ Always))))

Figure 5.5.: Classes Capturing Constraint Violations of P3P Policies for Single Services

Purpose value `admin` is shown due to space limitation. To complete this class description, similar axioms for each *Purpose* value must be added. Appendix D shows the complete description of this invalid policy class.

InvalidPolicy-DH-RecipientOpt (β_7) represents the class of invalid P3P policies where the optionality value of some *Recipient* of some data–purpose collection practice is more restrictive than the optionality value of the *Recipient* of the data–purpose collection practice of a descendant. Fig. 5.5 shows an excerpt of axioms for *Purpose* value `admin` and *Recipient* value `same`. Similar axioms for each *Purpose* value and *Recipient* values must be added to complete this class description as illustrated in Appendix D.

InvalidPolicy-DH-Retention (β_8) represents the class of invalid P3P policies where the *Retention* value of some data–purpose collection practice is more restrictive than the *Retention* value of some data–purpose collection practice an ancester. Fig. 5.5 shows an excerpt of axioms for *Purpose* value `admin`. The complete description (after adding similar axioms for each *Purpose* value) of this invalid policy class is illustrated in Appendix D.

InvalidPolicy-OptinalAttributes (β_9) represents the class of invalid P3P policies that have incompatible optionality values between data collection and data–purpose usage. Incompatibility occurs when a data collection is required (i.e. its optionality is `no`), but all of its purpose optionality values are `opt-in`.

InvalidPolicy-Purpose-DataCat1 (β_{10}) represents the class of invalid P3P policies that collect some data that are not in `online` nor `physical` category for `contact` purpose.

InvalidPolicy-Purpose-DataCat2 (β_{11}) represents the class of invalid P3P policies that collect some non-`physical` data for the `telemarketing` purpose.

InvalidPolicy-Purpose-DataCat3 (β_{12}) represents the class of invalid P3P policies that collect some data that are not in `purchase`, `Government`, `financial`, `online` nor `physical` category for `individual- decision` purpose.

InvalidPolicy-Purpose-DataCat4 (β_{13}) represents the class of invalid P3P policies that collect some data that are not in `purchase`, `Government`, `financial`, `online` nor `physical` category for `individual- analysis` purpose.

InvalidPolicy-Purpose-Recipient1 (β_{14}) represents the class of invalid P3P policies that collect some data for any recipient values except `ours` for any of the purposes `admin`, `develop`, `historical` or `tailoring`.

InvalidPolicy-Purpose-Recipient2 (β_{15}) represents the class of invalid P3P policies that collect some data for recipient `prospective-composite-service` for the purposes other than `current`.

InvalidPolicy-Purpose-Retention (β_{16}) represents the class of invalid P3P policies that collect some data for the purposes `individual-analysis`, `individual-decision`, `pseudo-analysis`, `historical`, `pseudo-decision`, `admin`, `contact`, `telemarketing` or `develop` but does not retain it (`no- retention`).

InvalidPolicy-Retention-Recipient (β_{17}) represents the class of invalid P3P policies that collect some data for the `public` recipient but does *not* retain it `indefinitely`.

InvalidPolicy-Recipient-DataCat (β_{18}) represents the class of invalid P3P policies that collect some non-`physical` data for the `delivery` recipient.

InvalidPolicy-ProspectiveCSOpt (β_{19}) represents the class of invalid P3P policies that have recipient `prospective-composite-service` with optionality value `opt-out` or `always`.

Constraints for Checking Incompatible P3P Policies for Composite Services

According to three defined constraints for composite services proposed in Section 5.2.2, we create three additional OWL classes to capture the constraint violations β_{20}–β_{22} in Fig. 5.6 with their descriptions below:

$$
\begin{array}{lrcl}
\beta_{20} & \mathsf{IncompatiblePolicy\text{–}DataCollection\text{–}CS} & \equiv & \mathsf{Policy} \sqcap (\exists \mathsf{collects}.(\mathsf{SharedData} \sqcap (\exists \mathsf{optionality}.\mathsf{Optional}))) \\
\beta_{21} & \mathsf{IncompatiblePolicy\text{–}Purpose\text{–}CS} & \equiv & \mathsf{Policy} \sqcap (\exists \mathsf{collects}.(\mathsf{SharedData} \sqcap (\forall \mathsf{collectedForPurpose}.(\neg \mathsf{Current})))) \\
\beta_{22} & \mathsf{IncompatiblePolicy\text{–}Recipient\text{–}CS} & \equiv & \mathsf{Policy} \sqcap (\exists \mathsf{collects}.(\mathsf{OutputSharedData} \sqcap \\
 & & & \quad (\forall \mathsf{collectedForPurpose}.(\forall \mathsf{hasRecipient}.(\neg \mathsf{ProspectiveCompositeService})))))
\end{array}
$$

Figure 5.6.: Classes Capturing Constraint Violations of P3P Policies for Composite Services

IncompatiblePolicy-DataCollection-CS (β_{20}) represents the class of incompatible P3P policies to the composite service that collect shared data optionally.

IncompatiblePolicy-Purpose-CS (β_{21}) represents the class of incompatible P3P policies to the composite service that collect shared data for non-`current` purpose.

IncompatiblePolicy-Recipient-CS (β_{22}) represents the class of incompatible P3P policies to the composite service that collect some shared data that are output data for non-`prospective-composite-service`.

Due to the *open-world assumption* and *non-unique name assumption* adopted in OWL, additional information are required so that the axiom β_9–β_{14}, β_{18}, and β_{21}–β_{22} can correctly capture undesired policies. One solution is to add *closure assertions* that are shown in *Non-Category*, *Number of Purpose value*, *Number of Recipient value* and *Non-Prospective-Composite-Service* paragraphs of the following section.

5.3.4. Individuals and Ontological Assertions

A given P3P policy (in XML syntax) can easily be translated into an ontological individual with associated assertions, and then these assertions will be verified against the ontology. With our design the ontology would remain consistent in most cases, while the P3P individual may be "inferred" to be an instance of one or more of the InvalidPolicy and IncompatiblePolicy classes (β_1–β_{22}). The translated individual must have a structure corresponding to the ontology model. Besides the values obtained directly from the P3P document, some hidden information needed for completeness and for constraints checking must be added. We describe this information below:

Default Values: The P3P specification defines default values of some elements and attributes to be applied in case no value of these is explicitly specified in a P3P policy document. These values are data collection optionality, default value is Required; purpose and recipient optionality, default value is Always; and data category, default value is allowed categories based on each data item. An example of default value descriptions of data optionality and data category of a data Login, and purpose optionality of a purpose value where the data is collected for administration purpose are illustrated in a class description below:

$$\begin{aligned}\mathsf{Login} \sqcap\ & (\exists \mathsf{optionality.Required}) \\ & \sqcap\ (\exists \mathsf{categorizedIn.UniqueID}) \\ & \sqcap\ (\exists \mathsf{collectedForPurpose.}(\mathsf{Admin} \sqcap\ (\exists \mathsf{optionality.Always}))).\end{aligned}$$

Non-Category: Besides the information about in which categories a particular data item is categorized, we also need information on which categories a particular data item is not categorized in to be able to check data category related constraints (β_{10}–β_{13} and β_{18}). For example, the class β_{10} represents the class of invalid P3P policies that collect some data that are not in `online` nor `physical` category for the `contact` purpose. Due to the fact that each data item can belong to multiple categories, it is thus impossible to set all categories to be disjoint. Though the specified data are stated to belong to other categories (i.e. other than `online` and `physical`), the OWL reasoner cannot yet conclude that the specified data do not belong to the categories `online` and `physical` (due to the open-world assumption). Therefore, we must explicitly state the categories that the data do not belong to as well so that the OWL reasoner has sufficient information for its reasoning. The description expressing that the specified data item does not belong to categories `government`, `physical`, and `Computer` is illustrated below:

$$(\neg(\exists \mathsf{categorizedIn.}(\mathsf{Government} \sqcup \mathsf{Physical} \sqcup \mathsf{Computer}))).$$

Number of Purpose Value: The number of *Purpose* values for a data item is needed to check the Optional Attributes (β_9) constraint. The class β_9 represents the class of invalid P3P policies where data collection is required, but all of its purpose optionality are `opt-in`. Without the actual number of *Purpose* value, the OWL reasoner assumes that there could be more *Purpose* values (again due to the open-world assumption). Therefore it cannot infer the individuals that are actually compatible

with the requirements to be an instance of this class. Another constraint that requires this additional information is the Purpose-CompositeService (β_{21}). We use maxCardinality to describe the number of *Purpose* values. The description expressing that three is the maximum number of *Purpose* values is shown below:

$$(\leq 3\ \mathsf{collectedForPurpose}).$$

Data Hierarchy: The information on which data are a super-data structure of other data are needed to check data hierarchy related constraints (β_5–β_8). The description of an assertion indicating that the data `login` has the data `id` as its sub data structure is shown below:

$$((\mathsf{LOGIN}, \mathsf{ID}) :\ \mathsf{hasSubDataStructure}).$$

Retention Duration: We have defined retention duration as an extension of some *Retention* values i.e. `stated-purpose`, `legal-requirement` and `business-practices`. To relate these values with the retention duration, we define a data property called retentionDuration. Since the defined data type is `xs:duration` which is not supported by OWL2, we add the assertion for this element with its value in seconds. An example of the data property assertion describing that a data item having *Retention* value `StatedPurpose` will be stored for 600 seconds is shown below:

$$((\mathsf{STATEDPURPOSE},\ 600) : \mathsf{retentionDuration}).$$

Number of Recipient Value: The reason to add the amount of *Recipient* values per a data–purpose pair is the same as the *Number of Purpose Value*. This information facilitates for checking the Purpose-Recipient1 (β_{14}) constraint. We use maxCardinality to describe the number of *Recipient* values. The description expressing that two is the maximum number of *Recipient* values is shown below:

$$(\leq 2\ \mathsf{hasRecipient}).$$

Non-Prospective-Composite-Service: The information about that each data–purpose pair does not have `prospective-composite-service` is required to help in checking *Recipient* value of composite service constraint (β_{22}). Though all other *Recipient* values are mutually exclusive, i.e. can be set as disjoint, the `prospective- composite-service` has different scope than the other values. Therefore it cannot be set as disjoint with other values. With same reason as *Non-Category*, the description below must be added when a data–purpose pair does not contain `prospective-composite-service` recipient.

$$(\neg(\exists \text{hasRecipient.ProspectiveCompositeService})).$$

Explicit-Recipient List: We have defined the explicit-recipient list as an extension of some *Recipient* values, i.e. `ours`, `same`, `delivery`, `other-recipient` and `unrelated`. To relate these values with the actual data recipients, we define a data property named hasRecipientID. An example of the data

property assertion having ServiceA as an explicit-recipient of the recipient value `ours` is shown below:

$$((\mathsf{OURS}, \text{"\$ServiceA\$"}) : \mathsf{hasRecipientID}).$$

Shared Data: Since we define classes OutputSharedData and InputSharedData to be subclasses of SharedData, we add information on which data are a member of OutputSharedData and InputSharedData classes.

Example of Ontological Assertions of Walmart P3P Policy

For each P3P policy document, we create fresh ontological individuals with associated assertions. The P3P policy from `walmart.com` in Fig. 3.2 with our proposed extensions can be translated to ontological assertions as depicted in Fig. 5.7.

The assertions specify that WALMART-POLICY is a Policy that collects data items Login, HomeInfo and Name in which an individual is created for each data item i.e. WALMART-LOGIN, WALMART-HOMEINFO, and WALMART-NAME respectively. Login's collection is Required, its category is UniqueID and it is collected for purposes Current, Contact and Develop where an individual is created for each data–purpose pair, i.e. WALMART-LOGIN-CURRENT, WALMART-LOGIN-CONTACT and WALMART-LOGIN-DEVELOP. The purpose Current's optionality value is Always and the purpose optionality value of Contact and Develop is OptIn. For each data–purpose pair, an individual is created for each recipient and retention value associated to it. In this case, Walmart does not distribute the data Login collected for purpose Current to anyone so its recipient value is only Ours with recipient optionality Always and hasRecipientID value urn:recipient:id:Walmart. Then, an individual WALMART-LOGIN-CURRENT-OURS for recipient Ours is created. The data Login collected for purpose Current is stored Indefinitely and for StatedPurpose for ten minutes (600 seconds). Thus, individuals WALMART-LOGIN-CURRENT-INDEFINITELY and WALMART-LOGIN-CURRENT-STATEDPURPOSE are created for the retention values Indefinitely and StatedPurpose respectively. The remaining data–purpose pairs of the data Login and assertions for HomeInfo's and Name's collection are analogous. The parts that are not from the P3P policy document directly are presented in italic.

Axioms in Fig. 5.4 augmented with assertions in Fig. 5.7 form a knowledge base about the P3P privacy policy issued by `walmart.com`. With the help of an OWL reasoning tool, this is processed and thereby additional knowledge can be inferred as to which InvalidPolicy classes the policy individual WALMART–POLICY may belong to.

WALMART–POLICY : Policy
(WALMART–POLICY, WALMART–LOGIN) : collects
(WALMART–POLICY, WALMART–HOMEINFO) : collects
(WALMART–POLICY, WALMART–NAME) : collects
WALMART–LOGIN : Login
WALMART–LOGIN : (∃*optionality.Required*)
WALMART–LOGIN : (∃*categorizedIn.UniqueID*)
WALMART–LOGIN : (≤ *3 collectedForPurpose*)
WALMART–LOGIN : (¬(∃*categorizedIn.(Demographic* ⊔ *Government* ⊔ *Navigation* ⊔ *Financial* ⊔ *State* ⊔ *Preference* ⊔ *Interactive* ⊔ *Physical* ⊔ *Political* ⊔ *Content* ⊔ *Health* ⊔ *Location* ⊔ *Computer* ⊔ *Purchase* ⊔ *Online*)))
(WALMART–LOGIN, WALMART–LOGIN–CURRENT) : collectedForPurpose
WALMART–LOGIN–CURRENT : Current
WALMART–LOGIN–CURRENT : (∃*optionality.Always*)
WALMART–LOGIN–CURRENT : (≤ *1 hasRecipient*)
WALMART–LOGIN–CURRENT : (¬(∃*hasRecipient.ProspectiveCompositeService*))
(WALMART–LOGIN–CURRENT, WALMART–LOGIN–CURRENT–OURS) : hasRecipient
WALMART–LOGIN–CURRENT–OURS : Ours
WALMART–LOGIN–CURRENT–OURS : (∃*optionality.Always*)
(WALMART–LOGIN–CURRENT–OURS, *"$urn:recipient:id:Walmart$"*) : *hasRecipientID*
(WALMART–LOGIN–CURRENT, WALMART–LOGIN–CURRENT–INDEFINITELY) : hasRetention
WALMART–LOGIN–CURRENT–INDEFINITELY : Indefinitely
(WALMART–LOGIN–CURRENT, WALMART–LOGIN–CURRENT–STATEDPURPOSE) : hasRetention
WALMART–LOGIN–CURRENT–STATEDPURPOSE : StatedPurpose
(WALMART–LOGIN–CURRENT–STATEDPURPOSE, *600*) : *retentionDuration*
(WALMART–LOGIN, WALMART–LOGIN–CONTACT) : collectedForPurpose
WALMART–LOGIN–CONTACT : Contact
WALMART–LOGIN–CONTACT : (∃optionality.OptIn)
WALMART–LOGIN–CONTACT : (≤ *1 hasRecipient*)
WALMART–LOGIN–CONTACT : (¬(∃*hasRecipient.ProspectiveCompositeService*))
(WALMART–LOGIN–CONTACT, WALMART–LOGIN–CONTACT–OURS) : hasRecipient
WALMART–LOGIN–CONTACT–OURS : Ours
WALMART–LOGIN–CONTACT–OURS : (∃*optionality.Always*)
(WALMART–LOGIN–CONTACT–OURS, *"$urn:recipient:id:Walmart$"*) : *hasRecipientID*
(WALMART–LOGIN–CONTACT, WALMART–LOGIN–CONTACT–INDEFINITELY) : hasRetention
WALMART–LOGIN–CONTACT–INDEFINITELY : Indefinitely
(WALMART–LOGIN–CONTACT, WALMART–LOGIN–CONTACT–STATEDPURPOSE) : hasRetention
WALMART–LOGIN–CONTACT–STATEDPURPOSE : StatedPurpose
(WALMART–LOGIN–CONTACT–STATEDPURPOSE, *31536000*) : *retentionDuration*
(WALMART–LOGIN, WALMART–LOGIN–DEVELOP) : collectedForPurpose
WALMART–LOGIN–DEVELOP : Develop
WALMART–LOGIN–DEVELOP : (∃optionality.OptIn)
WALMART–LOGIN–DEVELOP : (≤ *1 hasRecipient*)
WALMART–LOGIN–DEVELOP : (¬(∃*hasRecipient.ProspectiveCompositeService*))
(WALMART–LOGIN–DEVELOP, WALMART–LOGIN–DEVELOP–OURS) : hasRecipient
WALMART–LOGIN–DEVELOP–OURS : Ours
WALMART–LOGIN–DEVELOP–OURS : (∃*optionality.Always*)
(WALMART–LOGIN–DEVELOP–OURS, *"$urn:recipient:id:Walmart$"*) : *hasRecipientID*
(WALMART–LOGIN–DEVELOP, WALMART–LOGIN–DEVELOP–STATEDPURPOSE) : hasRetention
WALMART–LOGIN–DEVELOP–STATEDPURPOSE : StatedPurpose
(WALMART–LOGIN–DEVELOP–STATEDPURPOSE, *7776000*) : *retentionDuration*
WALMART–HOMEINFO : HomeInfo ···
WALMART–NAME : Name ···

Figure 5.7.: Ontological Assertions Describing Walmart's P3P Policy

5.4. P3P Policy Verification Using Our P3P Ontology

An important benefit we obtain for free by using OWL is its reasoners in which we use for P3P policy verification. A P3P policy is verified as an invalid policy and/or incompatible policy when the policy individual is inferred (based on our P3P ontology) by an OWL reasoner as an instance of any InvalidPolicy classes and/or IncompatiblePolicy classes respectively.

In this section, we show our implementation for a P3P verification tool. With an input P3P policy file, this tool can verify the policy validity according to the proposed constraints and at the same time can verify (with additional shared data file) whether it is compatible when used in a composite service.

5.4.1. P3P Verification Tool

The architecture of our P3P verification tool is shown in Fig. 5.8. The implementation of this tool has been done in three steps; a) developing the ontology for P3P policy (TBox), b) creating ontological individuals (ABox) from an actual P3P policy, and c) querying verification results of the policy from an OWL reasoner with the given TBox and ABox. The second and third steps are implemented in Java programming language with the help of OWL API [owl]. These three steps are described below:

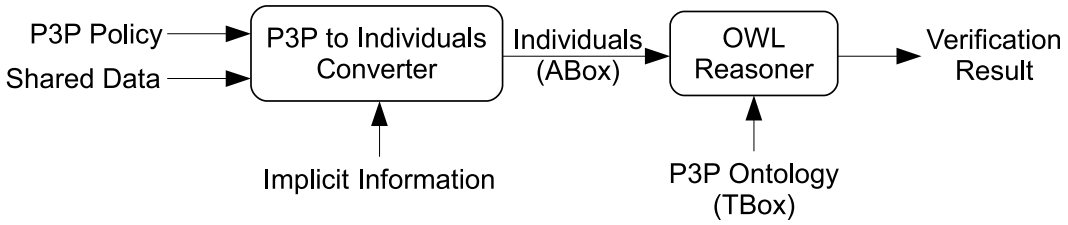


Figure 5.8.: P3P Policy Verification Tool Model

TBox Development

The P3P ontology (TBox) is developed according to data–purpose centric paradigm using Protégé 4.1[1]. We have implemented it as described in the previous section. InvalidPolicy classes are created to capture policies being inconsistent in terms of compatibility (as in Fig. 5.6) and validity (as in Fig. 5.5). During this TBox creation, we always check whether all the defined concepts are satisfiable and the expected subsumption relationships hold.

ABox Conversion

In the second step, individuals are created from a given P3P policy together with *SharedData* (if present), and implicit information. Since the individuals' structure must conform to the P3P ontology structure that is data–purpose centric, we first extract the necessary information from the P3P document and restructure it to the desired format.

[1]The syntax used in Protégé-OWL is Manchester OWL Syntax (http://www.w3.org/TR/owl2-manchester-syntax/)

The P3P specification allows websites to describe their own defined data, hence, there can be data in a given P3P policy that are not in the P3P base data. We do not consider these data and thus do not extract them from the P3P document. The *Purpose* value `other-purpose` describes other usage of the collected data not covered by the other *Purpose* values. We extract this value when there is only one `other-purpose` value in a policy since two `other-purpose` values may have different meaning.

The information about which data item is shared data is also extracted from the *SharedData* file and is added by class assertion. The implicit information consists of default values, non-category, number of purpose and recipient values, and data hierarchy.

Our ABox converter is implemented in such a way that it can convert the given P3P policy for both the original version and the extended version as we proposed. If the *SharedData* file is present, policy incompatibility can also be verified at the same time; otherwise only policy validity can be checked. An example of individuals converted from `walmart.com` is shown in Fig. 5.7.

Querying Verification Result

The derived individuals (ABox) from the second step are evaluated against the P3P ontology (TBox) defined in the first step by an OWL reasoner. Since we employ OWL API, there are four reasoners that provide implementations of the OWL API `OWLReasoner` interface; FaCT++ [FaC], HermiT [SMH08, Herb], Pellet [Pel] and RacerPro [Rac]. From these reasoners, we choose HermiT (HermiT1.3.5) since FaCT++ does not support datatype `unsignedLong` whereas Pellet and RacerPro cannot perform reasoning successfully since they require more memory than our machine can provide.

If the P3P ontology is consistent, we further check the validity and compatibility of the P3P policy. If the policy is invalid or incompatible, we retrieve the reasons by checking which InvalidPolicy or IncompatiblePolicy class the policy individual is inferred to.

5.4.2. Result and Analysis

We devide our experiment to verify P3P policies in two cases, compatibility verification and validity verification. In the first case, we illustrate an example using Walmart policy; while in the second case, we validate 500 P3P policies acquired from actual websites. The time consumption for ABox conversion and for reasoning is also measured for both cases and analysed in the latter case.

P3P Policy Compatibility Verification

We show the compatibility verification case using Walmart policy shown in Listing 5.1. This policy is added with our proposed extensions in Section 5.1. In this scenario Walmart is trying to combine its service with another service to become a composite service in which the *shared data* are *#user.name* as input data and *#user.home-info* as output data.

Listing 5.1: Extended P3P Policy of Walmart.com with Proposed Extensions Elements

```
Policy{Entity(#business.name):walmart.com,...,
        Statement1{Purpose:(current,contact[opt-in]),
                   Recipient:(ours,
                              ours-list:(urn:service:id:Walmart)),
                   Retention:(indefinitely),
```

```
                Data:(#user.login,#user.home-info)}
    Statement2{Purpose:(current,develop[opt-in],contact[opt-in]),
                Recipient:(ours,
                           ours-list:(urn:service:id:Walmart)),
                Retention:(stated-purpose,
                           current-duration:(PT10M),
                           develop-duration:(P3DT10H30M),
                           contact-duration:(P2Y4M)),
                Data:(#user.name,#user.login,#user.home-info)}
    Statement3{Purpose:(current,develop,admin),
                Recipient:(ours,delivery,
                           ours-list:(urn:service:id:Walmart),
                           delivery-list:(DeliveryConcepts)),
                Retention:(stated-purpose,
                           current-duration:(PT10M),
                           develop-duration:(P3DT10H30M),
                           admin-duration:(P6M)),
                Data:(#user.name,#user.home-info,#thirdparty.name,
                      #thirdparty.home-info,#dynamic.interactionrecord,
                      #dynamic.cookies
                         {Categories:(uniqueId,computer,navigation,
                                      state,purchase)})}
    Statement4{Purpose:(current,develop,admin,contact[opt-in]),
                Recipient:(ours,delivery,
                           ours-list:(urn:service:id:Walmart),
                           delivery-list:(DeliveryConcepts)),
                Retention:(stated-purpose,
                           current-duration:(PT10M),
                           develop-duration:(P3DT10H30M),
                           admin-duration:(P6M),
                           contact-duration:(P2Y4M)),
                Data:(#user.name,#user.home-info,#user.login,
                      #thirdparty.name,#thirdparty.home-info,
                      #dynamic.interactionrecord,#dynamic.clickstream,
                      #dynamic.http,#dynamic.searchtext,
                      #dynamic.cookies
                         {Categories:(uniqueId,computer,navigation,
                                      state,purchase)})}
    Statement5{Purpose:(current,develop,admin),
                Recipient:(ours,
                           ours-list:(urn:service:id:Walmart)),
                Retention:(indefinitely),
                Data:(#dynamic.interactionrecord,#dynamic.clickstream,
                     #dynamic.http,#dynamic.searchtext,
                     #dynamic.cookies
                         {Categories:(uniqueId,computer)})}
    Statement6{Purpose:(current),
                Recipient:(prospective-composite-service),
                Retention:(stated-purpose,
                           current-duration:(PT10M)),
                Data:(#user.home-info)}
}
```

Given the extended Walmart policy and the shared data, the tool verifies that the Walmart policy is compatible for the composite service but is an invalid policy since it is inferred as an instance of InvalidPolicy–Purpose–DataCat1, InvalidPolicy–MultipleRetention, InvalidPolicy–MultiplePurposeOpt and InvalidPolicy–Recipient–DataCat classes as shown in Fig. 5.9. In other words, the policy from Listing 5.1 does not comply with four designed constraints. More precisely, the first case is due to the incompatibility between the purpose Contact and the data item such as Login, which does not belong to Online nor Physical category. The second case stems from the fact that the Login–Current collection practice has multiple *Retention* values, viz. Indefinitely and StatedPurpose. The third case is, for in-

P3P_POLICY filename : C : \MyProgram\runtes_policies\Walmartp3ppolicy_CSTest.xml

ABox Converter finished in : 1212milliseconds

P3P_POLICY : INVALID
Reason :
InvalidPolicy–Purpose–DataCat1
"Invalid if the purpose is Contact but the data category is neither Physical nor Online."
InvalidPolicy–MultipleRetention
"Invalid due to multiple retention values per one data–purpose collection."
InvalidPolicy–MultiplePurposeOpt
"Invalid due to multiple data usage optionality values at the purpose level."
InvalidPolicy–Recipient–DataCat
"Invalid if the collected data are non–physical for the delivery recipient."

P3P_POLICY : COMPATIBLE

Reasoning finished in : 3714 milliseconds

Figure 5.9.: Verification Result of Extended Walmart Policy

stance, because the purpose Develop has multiple optionality values, i.e. Always and Opt-In; whereas, the last case is because of the incompatibility between the recipient Delivery and the data item such as HTTP, which does not belong to Physical category. The extended Walmart policy takes 1212 and 3714 milliseconds for ABox conversion and reasoning tasks respectively.

P3P Policy Validity Verification

In addition to the compatibility verification, we have verified 500 actual P3P policies obtained from websites to analyze their validities. Fig. 5.10 illustrates the number of policies having certain elements

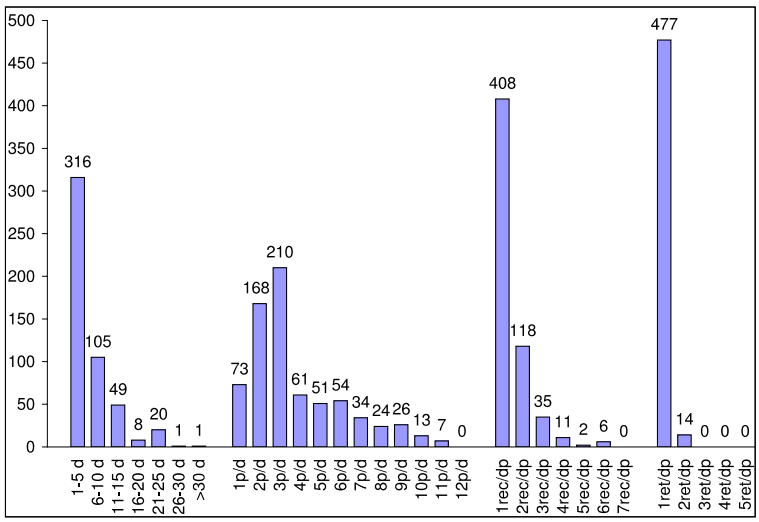


Figure 5.10.: Numbers of P3P Policies in Certain Numbers of Elements

where the first group shows the number of policies that contain data in certain numbers; the second,

third and fourth group show the number of policies that contain numbers of purposes per a data item, numbers of recipients per a data–pupose pair, and numbers of retentions per a data–purpose pair respectively. The majority of policies contain one to five data items, three purposes per data item, one recipient per data–purpose pair, and one retention per data–purpose pair.

Table 5.1.: Overall Validity Verification Results of 500 P3P Policies

	ABox Conversion	Policies without Base Data	OWL Reasoning		
			Inconsistent	Consistent	
				Valid	Invalid
Number Of P3P Policies	500	22	9	215	254

From 500 policies, the overall results in number of policies is shown in Table 5.1. Only 469 of them are actually successfully verified. This is because 22 policies do not contain base data as defined in the P3P specification. Nine policies are inconsistent with the defined P3P ontology (TBox) where six of them contain purpose `current` with optionality value other than `always` and the remaining three policies contain recipient `ours` with optionality value other than `always`. With the 469 policies being successful verified, there are 215 (45.8%) valid policies and 254 (54.2%) invalid policies.

The reasons of policies invalidity are presented in Fig. 5.11. From the 254 invalid policies, the first,

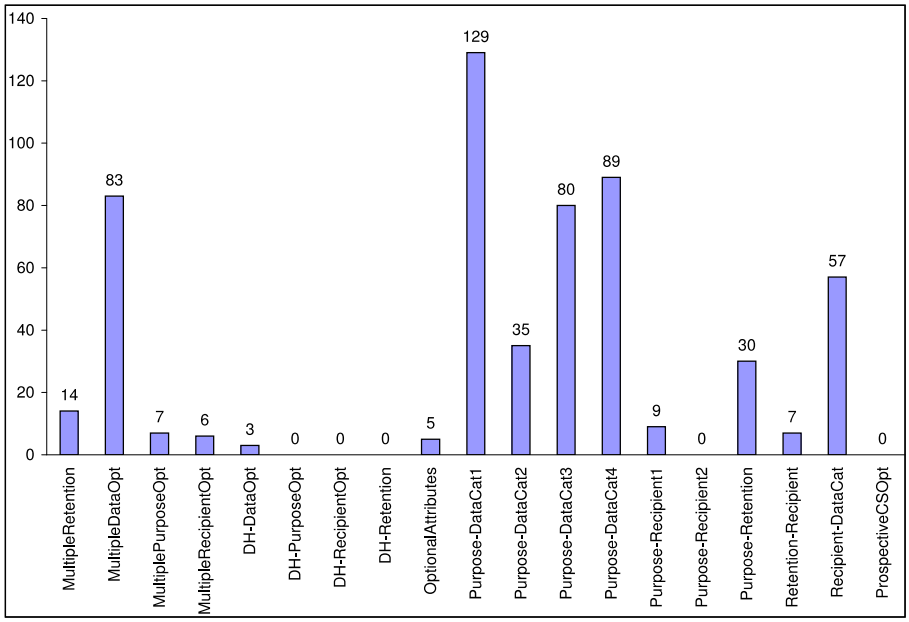


Figure 5.11.: Numbers of P3P Policies of Each Invalid Type

second and fourth most popular causes are from conflict between *Purpose* and *Data Categories* values case 1, 4 and 3 (β_{10}, β_{13} and β_{12}) respectively. The third most popular cause is from having multiple optionality values for a data item (β_2). None of the policies are invalid due to the three data hierarchy

constraints for the *Required* value of *Purpose* and *Recipient* elements, and for the *Retention* value (β_6–β_8). No policy is invalid due to conflict between *Purpose* and *Recipient* values case 2 (β_{15}) and Prospective-composite-service optionality (β_{19})) because these constraints are applicable only with the extended version of P3P policy as we proposed. Thus, none of the actual P3P policies can be inferred as a member of these InvalidPolicy classes. When considering the number of invalid types per P3P policy shown in Fig. 5.12, the majority (44%) of the 254 invalid policies has one invalid type and the maximum invalid types per policy is seven.

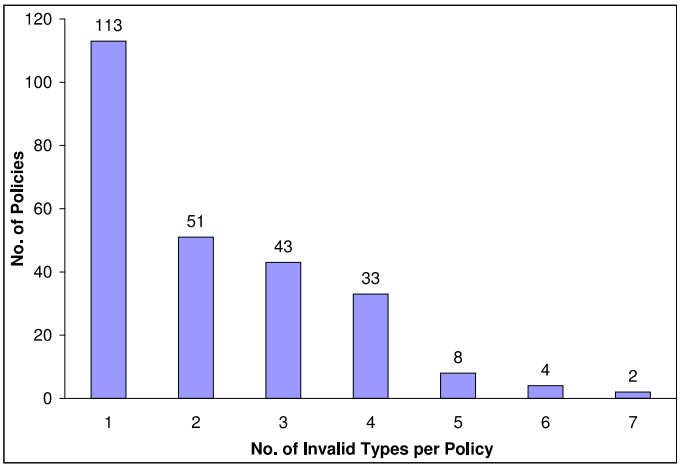


Figure 5.12.: Numbers of Invalid Types per a P3P Policy

The time consumption for ABox conversion and for reasoning is measured separately below. The test machine has a 3.16 GHz Intel Duo processor, 4 GB of RAM, running 64-bit Windows 7. The Java Runtime Environment used in the test environment was 64-bit Sun Java 1.7.

ABox Conversion Time: From all 500 policies, the time for ABox conversion of most policies is acceptable and not much different (the majority of the policies takes between 1100 - 1300 ms) as depicted in Fig. 5.13. As expected the conversion time corresponds to the number of data–purpose pairs. Fig. 5.14 shows the ABox conversion time of the first twenty policies with their number of data–purpose pairs. When considering at the same number of data–purpose pairs, policies with a higher number of *Recipient* values or *Retention* values have higher conversion time.

Reasoning Time: The time spent for the reasoning task by the HermiT reasoner is shown in Fig. 5.15. In general, the reasoning time for most of the P3P policies is acceptable. The reasoning task for most of the policies (401 from 469 policies, 86%) can be done within 5 seconds.

In order to investigate the impact of policy elements to reasoning time, we create three series of P3P policies test cases; Ideal1, Ideal2, and Ideal3. The first case, Ideal1, is created to observe the impact of data and purpose. It is a series of P3P policies where the numbers of data and purposes are varied while the number of retentions and recipients are fixed to 1. The number of purposes is varied from 1 to 12

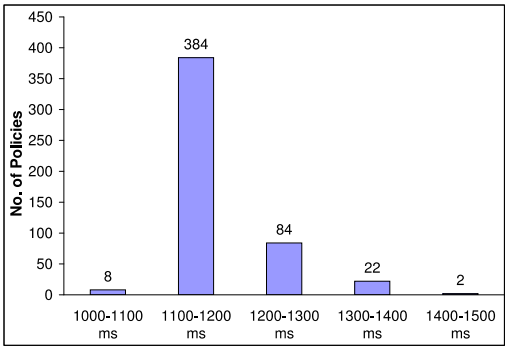


Figure 5.13.: ABox Conversion Time of 500 P3P Policies

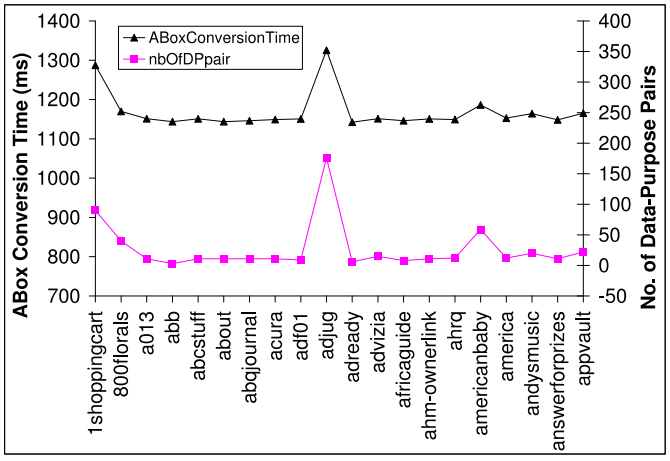


Figure 5.14.: ABox Conversion Time and Numbers of Data–Purpose Pairs of the First 20 P3P Policies

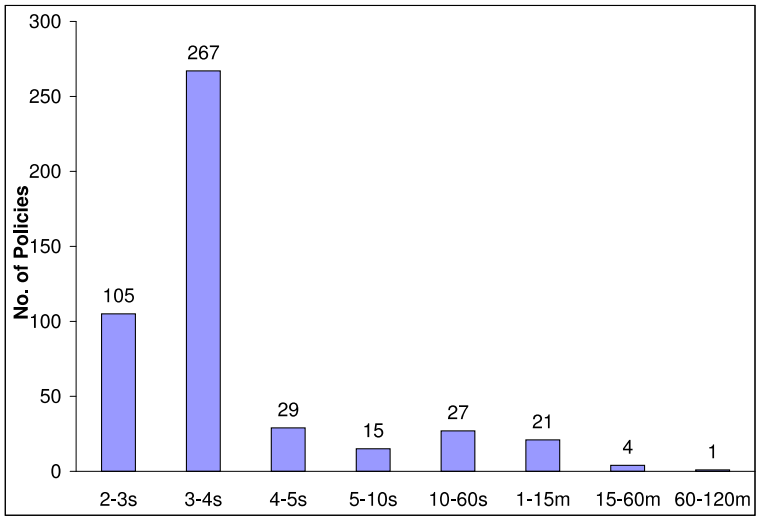
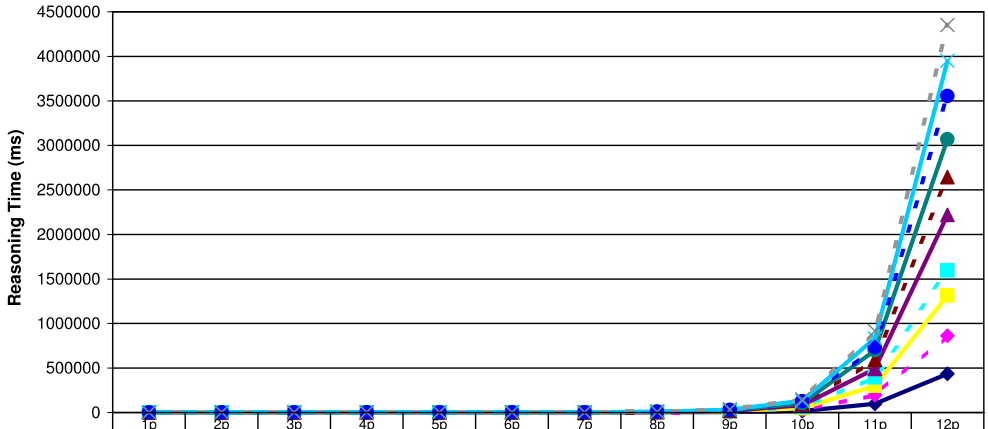


Figure 5.15.: Reasoning Time of P3P Policies with Successful Verification

since the maximum number of purpose value is 12. There is no data category specified in the policies which means the default categories are used. Ideal2 and Ideal3 are used to observe the influence of the number of retentions and recipients on the reasoning time by extending policies in the Ideal1 with maximum number of retention (i.e., 5) and recipient (i.e., 7) values respectively.

The test result of the Ideal1 case is shown in Fig. 5.16. When the number of purposes for a data item is from 8 on, the reasoning time increases rapidly. This exponential increase is in our expectation since it is normal for description logic with general TBox to have computational complexity as ExpTime [BHS08]. Considering the same number of purposes (from 8 on), a higher number of data causes higher reasoning time. However, if we compare between the number of purposes and data, the former has greater effect on the reasoning time than the latter. For instance, the reasoner uses 49640 ms for the policy containing three data and ten purposes while it uses only 4243 ms for the policy with ten data and three purposes. This is because in our proposed ontology model for P3P, purpose has more



	1p	2p	3p	4p	5p	6p	7p	8p	9p	10p	11p	12p
1d	3074	3307	3183	3058	3229	3994	3354	3666	6302	18049	99044	435834
2d	3151	3151	3167	3557	3011	2902	3651	4789	9282	31340	195812	862916
3d	3136	3651	2902	3151	3213	3526	3322	5335	13432	49640	299848	1315332
4d	3198	3323	2979	3323	3042	3417	3588	6178	15897	60668	396210	1595524
5d	2824	3416	2933	3479	3729	3011	4103	6100	19328	81526	490246	2220180
6d	3027	3244	2964	3603	2995	3260	4259	7223	22308	97345	591849	2642754
7d	3385	3635	3229	3057	3915	3416	4134	7847	27768	105440	697836	3069337
8d	2730	2917	3244	3494	3447	3213	4571	9032	30311	128466	736228	3553234
9d	2511	3463	3183	3152	3603	3479	4103	9219	32261	134410	833119	3950348
10d	3120	3214	4243	3432	3151	3635	4899	10280	36816	149822	917328	4352750

Figure 5.16.: Reasoning Time of P3P Policies in Ideal1 Case

relationships and constraints with other concepts than data. Therefore, the number of purposes has more influence on the reasoning time than the number of data.

The test results of Ideal2 and Ideal3 cases are compared with Ideal1 in logarithmic scale in Fig. 5.17. The reasoning time of Ideal2 (with 5 retentions) has similar characteristic as Ideal1, that is the reasoning

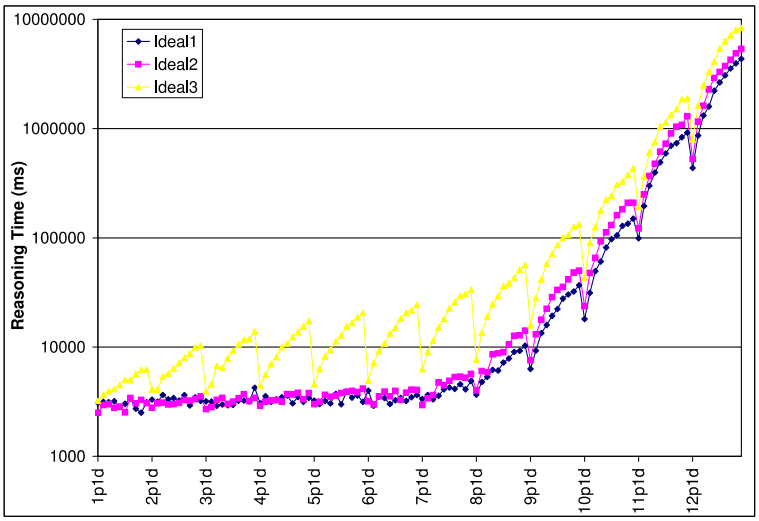


Figure 5.17.: Comparison of Reasoning Time of P3P Policies between Ideal1, Ideal2, and Ideal3

time of Ideal2 starts to increase dramatically when the number of purposes is from 8 on where Ideal2 has higher reasoning time than Ideal1 from approximately 9% to 56%. This means that the number of retentions can increase the reasoning time significantly when the number of purposes is high (from 8 on). On the contrary, the reasoning time of the Ideal3 case (with 7 recipients) compared with Ideal1

is always higher (approximately 5% to 646%). This increase in time is higher when the number of purposes is higher until the number of purposes is 8 where the increase in time starts to gradually reduce. In other words, the number of recipients always increase the reasoning time especially when the number of purposes is 7.

5.5. Combining Mechanism

Obtaining privacy policies of composite services brings benefit to the users such that they do not have to check their preferences several time with all member services. To preserve privacy, the resulting privacy policies must be equal or more restrictive than the individual ones. This section presents a mechanism for combining P3P policies of the member services of a composite service to obtain the privacy policy of the composite service.

Following the data–purpose centric model, we design the combining mechanism to determine values of P3P elements and attributes (i.e. *Recipient* and *Retention* elements and *Required* attribute) for each data–purpose pair. For the *Data Categories* element and data collection optionality attribute (*Optional* attribute), the combined value determination of these two entities is considered per data item since they are directly related to the data. The main idea to determine the values of these elements and attributes are twofold a) for elements/attributes that must have only one value, the most restrictive value is chosen (except *Retention* value) b) for the elements/attributes that can have multiple values, all the values are combined using the union operation. Note that we assume that the privacy policies in this step are valid policies.

5.5.1. Data

All data items from all member services are added together to become the data of the composite service's privacy policy. For two data where one is an ancestor of the other, if the data collection optioanality value of the ancestor is equal or more restrictive, the descendant one is deleted. This is because the ancestor data already covers its descendant. The same data items from all services are taken into account for their optionality values and categories.

Data Collection Optionality

The data collection optionality describes whether the data are required to be collected by the services or not. There can be multiple values for this optionality attribute for the same data from all member services. Since each data item must have only one optionality value, we determine the data collection optionality (*Optional* value) for a data item of a composite service to be the most restrictive value, where the value `no` is more restrictive than `yes`.

Data Categories

A data item can belong to several categories. Therefore, we determine the *Data Categories* value of a data item of the composite service to be the union of all *Data Categories* values defined in the privacy policies of all service members.

5.5.2. Purpose

A data item can have multiple *Purpose* values. With the same data, all different *Purpose* values are added together as to be the *Purpose* values of the composite service's privacy policy. Each *Purpose* value can have only one purpose optionality (*Required* attribute) value. We determine the purpose optionality (*Optional* value) for the same data–purpose of a composite service to be the most restrictive value, where the value `always` is more restrictive than `opt-out` and `opt-out` is more restrictive than `opt-in`.

5.5.3. Retention

As described in Section 5.1.3, it is not trivial to determine *Retention* values for a composite service because some values are incomparable. The proposed retention duration can help in determining the *Retention* value.

Since there can be only one *Retention* value for a data–purpose pair, we determine the *Retention* value of a data–purpose pair of the composite service to be the least restrictive one (the longest retention), where the value `indefinitely` is less restrictive than `business-practices`, `legal-requirement` and `stated-purpose`; and `business- practices`, `legal-requirement` and `stated-purpose` are less restrictive than `no-retention`.

Due to the fact that `business-practices`, `legal-requirement` and `stated-purpose` are incomparable on their retention lenght, we employ the retention duration by considering the longest storing period and its *Retention* value to become the retention duration and *Retention* value of the composite service respectively. For example, let the purpose for collecting a particular data be `admin` and the *Retention* value of the first service be `stated-purpose` with a retention duration of three months, while the second service collects the same data for the same purpose for six months with the *Retention* value `business-practices`. The *Retention* value and its duration of the composite service for this data-purpose pair is `business-practices` with six months duration due to the longer time duration.

In case when the retention duration is equal, we determine the *Retention* value of the composite service to be the one that has the least restrictive meaning where the `business-practices` has the least unambiguous meaning on what business practices are and the `legal-requirement` has the most restrictive meaning that the service must retain the data to comply with the law.

5.5.4. Recipient

Determining the *Recipient* value for composite services is the most complicated case. Though a data–purpose pair can have several *Recipient* values, obtaining the value for the composite service is not possible by just adding all values from all member services. This is due to the change of scope of the values (as described in Section 5.1.2). Therefore, we use the proposed explicit-recipient list to facilitate determining the *Recipient* value of the composite service. The following steps describe how to determine *Recipient* value, its optionality, and its explicit-recipients.

1. *Add all Recipient values and their explicit-recipients:* With the same data–purpose pair, all different *Recipient* values from all services are added together as to be the *Recipient* values of the

composite service's privacy policy. For a given *Recipient* value, the actual recipients associated to the value from all services are added. Since each *Recipient* value can have only one optionality value, we determine its optionality value to be the most restrictive one (the value `always` is more restrictive than `opt-out` and `opt-out` is more restrictive than `opt-in`).

2. *Check explicit-recipients that are redundant with member services, i.e. for* Recipient *value* `ours`: Since the scope of *Recipient* values from each member service of the composite service may be different, there might be some explicit-recipients in the lists that are redundant. This overlapping must be eliminated. We thus check whether there are any explicit-recipients in same-list, delivery-list, other-recipient-list and unrelated-list that are the member of the composite service. If there are, those explicit-recipients must be removed from their current lists and must be added in ours-list. In case `ours` is not one of the *Recipient* value, it must be added.

3. *Check explicit-recipients that have the same privacy practices as the composite service, i.e. for* Recipient *value* `same`: We check this case in two steps. Firstly we check if there are any explicit-recipients in delivery-list, other-recipient-list and unrelated-list that are redundant with explicit-recipients in the same-list. If there are, we remove those recipients from their current lists. Secondly we also have to check whether the remaining recipients in delivery-list and other-recipient-list have the same data practices as the composite service or not by verifying their privacy policies against the privacy policies of the composite service which will be explained later in *Privacy Practices Checking* section. It is not necessary to check services in the unrelated-list because this list contains services whose privacy policies are unknown. If any of the recipients in delivery-list and other-recipient-list conform to the composite service's privacy policies, they are moved to the same-list. We do not have to check redundancy of the explicit-recipients between delivery-list, other-recipient-list and unrelated-list since from their definitions they do not overlap with each other.

4. *Revise* Recipient *value* `prospective-composite-service`: Since the scope of the composite service is different from the single service, we remove the *Recipient* value `prospective-composite-service` that is associated with the data that are not the output of the composite service. These data can be derived from, e.g. WS-BPEL.

5. *Revise* Recipient *values*: If any explicit-recipient list is empty, we delete its corresponding *Recipient* value. The remaining *Recipient* values and their associated explicit-recipients, thus, are the *Recipient* values and explicit-recipients of the composite service's privacy policies.

Privacy Practices Checking

Regarding the checking of the same privacy practices in step 3, by now we already obtained the values of *Data*, *Purpose* and *Retention* elements, and their attributes. The verification between the privacy policies of the remaining services in delivery-list and other-recipient-list and the privacy practices of the composite service is done by checking with the obtained values and with restricted *Recipient*

values. For a data–purpose pair, if the *Data Categories* values are equal or super set of, the data optionality (*Optional* attribute) value and purpose optionality (*Required* attribute) value are equal or more restrictive than, and the *Retention* value is equal or less restrictive than those of the remaining explicit-recipients in delivery-list and other-recipient-list; and if the *Recipient* values of the explicit-recipients in these lists contain only `ours` or `same` or both, this implies that their privacy policies conform to the composite service.

5.5.5. Example

In this section, we create a scenario to show the proposed combining mechanism to obtain privacy policy of a potential composite service named NearestAirportFinder. The NearestAirportFinder service may consist of CurrentCityLocator and AirportFinder services. Given a user's mobile number, the CurrentCityLocator returns user's current city name which will be sent to the AirportFinder service. In return, the AirportFinder replies with a list of nearest airports to the user's current city. Fig. 5.18 and 5.19 depict P3P policies of the CurrentCityLocator and AirportFinder services respectively. Note that these policies are already extended by some extensions as we proposed in Section 5.1.

Policy { Entity (#business.name): currentcitylocator.com,. . . ,
Statement1 { Purpose: (admin, current, develop),
Recipient: (ours,
ours-list: (CurrentCityLocator)),
Retention: (stated-purpose,
admin-duration: (P6M),
current-duration: (PT5M),
develop-duration: (P6M)),
Data: (#dynamic-clickstream, #user.home-info.telecom.mobile) }
Statement2 { Purpose: (current),
Recipient: (ours, prospective-composite-service
ours-list: (CurrentCityLocator)),
Retention: (stated-purpose,
current-duration: (PT5M)),
Data: (#dynamic.miscdata
{Categories: (location)}) } }

Figure 5.18.: CurrentCityLocator's P3P Policy

Policy { Entity (#business.name): airportfinder.com,. . . ,
Statement1 { Purpose: (current),
Recipient: (ours,
ours-list: (AirportFinder)),
Retention: (stated-purpose,
current-duration: (PT3M)),
Data: (#dynamic.miscdata
{Categories: (location)}) } }

Figure 5.19.: AirportFinder's P3P Policy

First we have to map the P3P policies of both services to be data–purpose base as shown in Table

5.2 and 5.3. It is clearer now that there is a data–purpose pair *#dynamic.miscdata–current* from both services. Therefore the values of *Retention* and *Recipient* of this pair must be combined.

Table 5.2.: Data Optionality, Data Categories and Retention Values for Each Data–Purpose Pair of CurrentCityLocator and AirportFinder Services

Services	**Data–Purpose**	**Data Opt.**	**Data Cat.**	**Retention**
CurrentCity-Locator	#dynamic.clickstream-admin	no	navigation, computer	stated-purpose (P6M)
	#dynamic.clickstream-current	no	navigation, computer	stated-purpose (PT5M)
	#dynamic.clickstream-develop	no	navigation, computer	stated-purpose (P6M)
	#user.home-info.telecom.mobile-admin	no	physical	stated-purpose (P6M)
	#user.home-info.telecom.mobile-current	no	physical	stated-purpose (PT5M)
	#user.home-info.telecom.mobile-develop	no	physical	stated-purpose (P6M)
	#dynamic.miscdata-current	no	location	stated-purpose (PT5M)
AirportFinder	#dynamic.miscdata-current	no	location	stated-purpose (PT3M)

Table 5.3.: Purpose Optionality, Recipient and Recipient Optionality Values for Each Data–Purpose Pair of CurrentCityLocator and AirportFinder Services

Services	**Data–Purpose**	**Purpose Opt.**	**Recipient**	**Recipient Opt.**
CurrentCity-Locator	#dynamic.clickstream-admin	always	ours {CurrentCityLocator}	always
	#dynamic.clickstream-current	always	ours {CurrentCityLocator}	always
	#dynamic.clickstream-develop	always	ours {CurrentCityLocator}	always
	#user.home-info.telecom.mobile-admin	always	ours {CurrentCityLocator}	always
	#user.home-info.telecom.mobile-current	always	ours {CurrentCityLocator}	always
	#user.home-info.telecom.mobile-develop	always	ours {CurrentCityLocator}	always
	#dynamic.miscdata-current	always	ours {CurrentCityLocator}, prospective-composite-service	always
AirportFinder	#dynamic.miscdata-current	always	ours {AirportFinder}	always

According to our combining mechanism that chooses *Retention* value with the longest retention time and union *Recipient* value, the privacy policy of the NearestAirportFinder service is determined as in Table 5.4 and 5.5. This means that the location data (*#dynamic.miscdata*) collected for usage purpose `current` will be retained for five months and it will be distributed to CurrentCityLocator and AirportFinder services. The value `prospective-composite-service` is removed since the location data is not the output data of the NearestAirportFinder service.

Table 5.4.: Data Optionality, Data Categories and Retention Values for Each Data–Purpose Pair of Composite Service NearestAirportFinder

Services	Data–Purpose	Data Opt.	Data Cat.	Retention
NearestAirport-Finder	#dynamic.clickstream-admin	no	navigation, computer	stated-purpose (P6M)
	#dynamic.clickstream-current	no	navigation, computer	stated-purpose (PT3M)
	#dynamic.clickstream-develop	no	navigation, computer	stated-purpose (P6M)
	#user.home-info.telecom.mobile-admin	no	physical	stated-purpose (P6M)
	#user.home-info.telecom.mobile-current	no	physical	stated-purpose (PT3M)
	#user.home-info.telecom.mobile-develop	no	physical	stated-purpose (P6M)
	#dynamic.misc-current	no	location	stated-purpose (PT5M)

Table 5.5.: Purpose Optionality, Recipient and Recipient Optionality Values for Each Data–Purpose Pair of Composite Service NearestAirportFinder

Services	Data–Purpose	Purpose Opt.	Recipient	Recipient Opt.
NearestAirport-Finder	#dynamic.clickstream-admin	always	ours {CurrentCityLocator}	always
	#dynamic.clickstream-current	always	ours {CurrentCityLocator}	always
	#dynamic.clickstream-develop	always	ours {CurrentCityLocator}	always
	#user.home-info.telecom.mobile-admin	always	ours {CurrentCityLocator}	always
	#user.home-info.telecom.mobile-current	always	ours {CurrentCityLocator}	always
	#user.home-info.telecom.mobile-develop	always	ours {CurrentCityLocator}	always
	#dynamic.miscdata-current	always	ours {CurrentCityLocator, AirportFinder}	always

5.6. Discussions

The P3P specification provides a means for services to declare new data elements by publishing their own data schemata. To be able to use our P3P ontology in case additional schemata are included, the new data elements and their hierarchy must be added in the ontology according to the new data schema.

Since the values of *Recipient* elements of existing P3P policies, i.e. without the explicit-recipient list, are defined as coarse-grained and in the single service paradigm, the P3P policies of a service provider will not change often. On the other hand, P3P policies with explicit-recipient lists which

could be altered dynamically may have to be updated more frequently. Though it might be inconvenient especially for the users who normally have to approve the new list every time it changes, knowing the list helps the users to protect their data disclosure to unwanted recipients both when they first check whether to use this service or not and later when the list changes. A way to provide more convenience to the users for the list's update is to allow the users to specify a recipient black list. When the list changes the service providers can obtain the users' decision automatically whether they agree with the new list or not without contacting the users again.

Regarding the ontology, an alternative way to converting a P3P policy to Individuals is to create additional classes that have the same structure as the base ontology. Once the reasoner is run, if these classes are inferred as subclasses of the IncompatiblePolicy and/or InvalidPolicy, then that policy is, therefore, incompatible and invalid respectively.

Ontology could be used in the *Privacy Practices Checking* step to determine the *Recipient* value in the combining mechanism. Since this step aims to check whether a data recipient has the same privacy practices as a composite service, a possible way is to infer a subsumption relation between their policies. If the privacy policy of the recipient is a subclass of the privacy policy of the composite service, we can conclude that this recipient has the same privacy practices as a composite service. In order for the reasoner to infer a subsumption relation, additional relations between values of P3P elements and attributes must be defined on which value is more restrictive than others.

In our ontology, we added a number of *Purpose* and *Recipient* values using the maxCardinality constructor. In fact, exactCardinality could be used since it represents the exact number of *Purpose* and *Recipient* values as we want. However, exactCardinality is equal to the minCardinality and maxCardinality constructors which means when exactCardinality is used, the reasoner has to do reasoning on both minCardinality and maxCardinality. In our application, employing maxCardinality instead of exactCardinality does not make any differences on the reasoning result. Therefore, we decided to employ maxCardinality for better performance.

5.7. Related Work

Yu et al. [YLA04] proposed a formal semantics for P3P. It interprets P3P policy in a data–centric view. According to this view, it defines five relations among elements in a P3P statement i.e. between a) data and purpose, b) data and recipient, c) data and retention, d) data and data collection, and e) data and data category. The semantics of a P3P policy is considered as a database where each relation is represented in a table. This work also defines constraints to check semantic consistency in three types i.e. data–centric, data hierarchy and semantic vocabulary constraints. Compared with this work, we have the same constraints for the data hierarchy. Our multiple statement constraints are comparable with the data–centric constraints except that ours are data–purpose base. The semantic vocabulary constraints are comparable with our pre-defined vocabulary constraints but the former does not identify all potential conflicts between *Purpose*, *Data Categories*, *Recipient* and *Retention* elements while we do. In addition, we also define constraints on combination of optionality values (Optional Attributes constraint). Since this work only concerns the original P3P policy, it does not define constraints related

to the composite service paradigm.

A work on formalizing P3P in an ontology [Hog04] was proposed as a W3C working group note. This and our work share the ideas of modeling most P3P entities as concepts (classes of individuals), of doing away with the notion of *Statement* by flattening them, of modeling nested data structures by an aggregation role instead of the subclass relation, and modeling data categories as superclasses. The modeling choice made in [Hog04] could be said to correspond to Paradigm♯1 in which each policy statement is flattened to a few ReifiedStatement objects, each describing a collection practice of a data item. This class is referred to in [Hog04] as CollectionPractice and related from Policy via a role describes. A subtle difference however remains in the choice between OWL quantifications. We reckon that a sensible policy should describe at least one collection practice of a data item, so someValueFrom ($\exists$) is chosen instead of the allValueFrom ($\forall$) constructor. In addition, we use roles subDataStructureOf and hasSubDataStructure in place of may-include-members-of to describe data structure defined in the base data schema. The fact that a super-data structure *may or may not* include a sub-data structure is straightforwardly modeled in our ontology using a number restriction, such as

$$\mathsf{User} \sqsubseteq (\mathsf{hasSubDataStructure}\ \mathbf{max}\ 1\ \mathsf{Login}).$$

Hogben [Hog05] proposed a way to represent P3P-based data schema in the Semantic Web. This work is similar to ours that the data items are modeled as classes. The data are interrelated via role type with existential quantification which is different from ours that we use roles subDataStructureOf and hasSubDataStructure with cardinality restriction as described above. .

Li and Benbernou [LB05] proposed to model a P3P policy and user preference (APPEL) as Description Logic classes, and matching between the two can be resorted to checking subsumption. Their ontology model of P3P policy is comparable to Paradigm♯1 in Fig. 5.1. Constraints within a policy structure were not considered in that work.

Berendt et al. [BPT08] proposed a privacy-protecting business-analytics service for online data controllers. This service checks whether certain collected data can be processed by data controllers without violating privacy restrictions where possible inferred data, privacy legislation and data controllers' privacy policies (P3P policies) are taken into account. The work proposed a formal model for the P3P statements which is similar to our P3P model, i.e. data–purpose base, but it omits retention element because it assumes that the data are present when the analysis is executed. While this service aims to verify whether a data analysis is allowed by checking conflicts between self-imposed privacy policies (P3P policies) and data usage, and between P3P policies and privacy legislation, our work verifies internal conflicts of a P3P policy and conflicts between P3P policies of data controllers.

Since our fundamental view on privacy policy is data–purpose centric, it is important that the ontology also reflects this. This makes our proposed Paradigm♯3 and the ontology (extension of Fig. 5.4) of fundamental difference compared to all the mentioned related works except from [BPT08]. Not only does this design make clear the semantics of the P3P, it also fully utilizes the facilities of OWL and its reasoning tools for knowledge retrieval and constraint checking.

5.8. Conclusion

We have extended a number of elements in P3P policy so that it can be applied with the composite service paradigm. To prevent certain semantic inconsistencies of the P3P policy, OWL ontology is employed. We have described three ontology models for the policy and implemented the most suitable one, i.e. the data–purpose centric view. Several constraints required to prevent certain semantic inconsistencies both for internal conflict of a P3P policy and for compatibility check of policies of member services in a composite service have been identified and formalized in an OWL ontology. We have proposed to capture constraint violation in OWL classes instead of to translate the constraint violation to a logical inconsistency. These OWL classes are defined with full definitions sufficient to detect semantic problems in P3P policy. A P3P verification tool has been developed. We have illustrated one case for compatibility verification of extended P3P policy and verified 500 P3P policies for their validities. The validity verification results show that according to our proposed contraints, the number of valid policies is approximately equal to the number of invalid policies. The most popular cause of invalidity is from when a policy containing *Purpose* value `contact` but its associated *Data* element does not belongs to one of `physical` and `online` data categories. We also have proposed a combining mechanism to obtain privacy policy of composite services.

6. XACML Policy Administration and Enforcement in Composite Service Environment

In Section 3.2.6 we have stated the problems that may occur in access control systems using XACML in a composite service scenario. These problems can cause burdens that may arise in policy administration and policy enforcement.

In this chapter we explain our solutions on how XACML policy can be expressed to ease policy administration and at the same time can improve the efficiency of XACML decision evaluation in Section 6.1. The prototype implementation of our solution is presented in Section 6.2. Existing work related to our solutions is reviewed in Section 6.3. Finally the chapter is concluded in the last section.

6.1. A Mechanism for XACML Policy Administration and Efficient XACML Decision Evaluation

A way to make policy administration easier is to specify individualized policies from a general policy for a meaningful part of data and additional information that tells which data items of the meaningful part are allowed to be revealed. For example, meaningful parts of a presence document can be classified in five parts that are an entire presence document, location data, service data, person data and device data. The additional information to tell which nodes in the requested part are permitted could be processed before (pre-processing) or after (post-processing) the authorization decision for an access is made. For the pre-processing case, the additional information can be used to modify the general policy. Moreover, data requesters have to be familiar with the policies so that they know what to request. For example if the Multiple Resource profile is employed to create a single request for multiple resources such as location data which consists of thirteen nodes, the path representing the root node of the data ("presence/tuple/status/geopriv") must be specified in a single request. Since an XACML PDP can evaluate one resource per time, thirteen individual requests created from the single request shall be evaluated where each individual request contains one node in the location data. However, more additional processing must be done that maps the single request for multiple resources to individual requests.

We therefore propose to use a post-processing concept as our solution. This means the additional information describing which nodes in the required part are permitted is employed to filter the requested data after the authorization decision is made. We propose an entity called Transformer Engine (TE) to do this filtering task. Based on the fact that hierarchical resources are frequently controlled under the

same restrictions [OAS05c], our mechanism (with TE) can let the PDP evaluate the request against an XACML policy for only the root node of the requested part/document. If the root node is prohibited, all of its sub-nodes are also prohibited. If the root node is permitted, all of its sub-nodes are checked later by the TE in such a way that the unauthorized nodes are filtered out before sending the document to the requester. Note that we aim for our solution to adhere to the XACML specification as much as possible.

In order for the TE to recognize which parts of the requested data should be filtered out, we employ *Transformations* from the Common Policy framework. Since the XACML specification allows operations that must be fulfilled in conjunction with the decision result to be defined in *Obligation* elements, we thus specify Transformations in an *Obligation*. In other words, this information is conveyed to the TE via an XACML *Obligation*. Note that in general, there can be several *Obligation* elements in a policy. The Policy Enforcement Point (PEP), which is the entity to enforce the policy according to the decision result including *Obligation*, must know where each *Obligation* should be sent to. Compared with the pre-processing scheme, our scheme does not require processes for policy modification and request mapping but rather filtering the requested data instead. This scheme can also improve the efficiency of PDP evaluation when the policy applicable with the request contains conditions that employ XPath Expression since our scheme evaluates the request for multiple resources only one time. Our solution can be applied to the access control system employing XACML without changes.

We describe how to express an appropriate XACML policy according to our proposed scheme in Section 6.1.1. Where the Transformer Engine is located in the access control system and how it interacts with other components in the system are elaborated in Section 6.1.2. The mapping of Transformations to an XACML *Obligation* will be described in Section 6.1.3.

6.1.1. Policy Expression

In order to express an XACML policy on who (subject) is allowed or forbidden to access (action) some meaningful data parts (resource), one can start with a policy where the subject and action are specified in its *Target* element and resource is specified in its *Rule* element. The allowed data parts are specified in an allowed rule that has the *Effect* value `Permit` and the forbidden data parts are specified in a prohibited rule that has the *Effect* value `Deny`. If the policy expresses explicit permissions and implicit prohibitions, the rule combining algorithm Permit-overrides can be employed and a default deny rule shall be used in a replace of prohibited rule. The information (Transformation) on how the allowed data parts shall be filtered is contained in the *Obligation* element and its *FulfillOn* attribute must be `Permit`.

Since each allowed data part can have one Transformation and all *Obligation* elements corresponding to the policy evaluation result (in this case, `Permit`) must be contained in the response message to be further processed by the TE, there can be some *Obligation* elements that contain Transformations that are redundant which makes the TE filter the required data part several times. This situation can occur when a permitted data part is a part of other permitted parts. For instance, in a presence document location data is a part of service data.

A way to avoid redundant filtering is to use the same Transformation for all allowed data parts. Therefore, we propose to use a Transformation that is specified for the entire document so that it can cover all accessible parts. Though, one might think that using this Transformation potentially reveals data not requested, it is actually not possible since the original request that the PEP forwards to the service according to positive result states what part of data the requester wants. The service hence knows which part of the document it must retrieve. Therefore, there is no such case that data beyond the request will be sent out.

From this conclusion, we define the XACML policy shown in Fig. 6.1 employing a meaningful data

```
Policy [RuleCombiningAlg: Permit-overrides]
    { Target { dataowner: ... ,
               subject-id (requester): ... ,
               document: ... },
     Rule1 [id: AllowedPart, Effect: Permit]
        { Target { resource-id: AllowedDataPart } },
     Rule2 [id: ProhibitedPart, Effect: Deny],
     Obligations
            { Obligation { ... Transformations ... } }
     }
```

Figure 6.1.: An XACML Policy for Our Proposed Scheme

part that a particular data owner can control access to his resource for a particular requester, using the combining algorithm `Permit-overrides` and consisting of two rules and an *Obligation*. The first rule is applicable to the request requiring data parts that are allowed to be accessed. Therefore, all permitted data parts are specified in the *Target* element of this rule and the *Effect* value of the rule is `Permit`. The last rule is a default rule with `Deny` result used for the prohibited data part and for handling other cases. The *Obligation* in the policy containing Transformations is derived as described in Section 6.1.3. The *Obligation* will be sent to the PEP when the decision result from the PDP is `Permit`.

6.1.2. Transformer Engine

The Transformer Engine (TE) is an entity that implements Transformations carried out by a special XACML *Obligation* received from the PEP. Since its main functionality is to filter out the prohibited part of the outgoing data, we thus place it in the access control architecture as depicted in Fig. 6.2 where its interactions with other entities are also shown.

Once there is a request for a hierarchical or sub-hierarchical document (1), the PEP creates an XACML request and sends it to the PDP for evaluation (2). By verifying the request with the proper XACML policy (elaborated in Section 6.1.1), i.e. containing *Transformations* in an *Obligation*, the PDP evaluates this request only one time for the root node of the requested document. If the decision is `Deny`, the PDP sends the negative result to the PEP. If the decision is `Permit`, the PDP sends this result to the PEP together with the *Obligations* containing the *Transformations* (3). The PEP forwards the acquired *Obligations* to the Transformer Engine (4) and the user's request to the service (5). According to

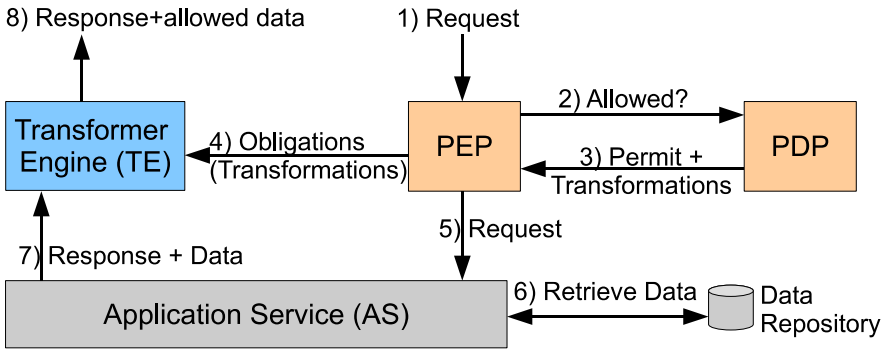


Figure 6.2.: Transformer Engine in Access Control Architecture

the incoming request (5), the service retrieves the requested part of the data (i.e. the entire hierarchical or entire sub-hierarchical document) (6) from the data repository. After the response message from the service is created and sent out with the requested data attached (7), the Transformer Engine intercepts the outgoing message and verifies the data items. If there are any data items that are not specified in the *Transformations*, the Transformer Engine will cut those parts out. The output data, thus, contain only the information that the data owner is willing to expose (8).

6.1.3. Mapping Transformations to XACML Obligations

As described in Section 3.2, an *Obligation* consists of an *ObligationId* attribute, a *FulfillOn* attribute and *AttributeAssignment* elements. An *AttributeAssignment* is composed of an *AttributeId* attribute, a *DataType* attribute and a value. In order to map a Transformation to an XACML *Obligation*, we define an *Obligation* with *ObligationId* `data-transform` to convey a Transformation from the PDP to the PEP. The value of the *FulfillOn* attribute of the defined *Obligation* must be `Permit`. This is because with a `Deny` decision, no data can be disclosed.

For the *AttributeAssignment* elements, since the Transformation carries application-specific information, we have to consider its mapping in detail for each application data. To better illustrate the idea, we utilize the presence document described in Section 4.2 as our showcase. Consequently, the Transformations employed for the mapping is obtained from the Presence Authorization Rules [Ros07] and Geolocation Policy [STC+11] which are authorization policies that use the Common Policy framework to govern access to the presence documents and location information as elaborated in Sections 4.3.2 and 4.3.3 respectively.

As described in Section 4.3.2, the Presence Authorization Rules define two types of Transformation, i.e. permissions for access to data component elements and permissions for access to presence attributes. We map these permissions to be the value of the *AttributeId* attribute of the *AttributeAssignment* element. For the data type value of the *AttributeAssignment* derived from the first type of Transformations, we use `http://www.w3.org/2001/XMLSchema#string`. The data type value of the latter is as specified in the Presence Authorization Rules. Except the *provide-user-input* per-

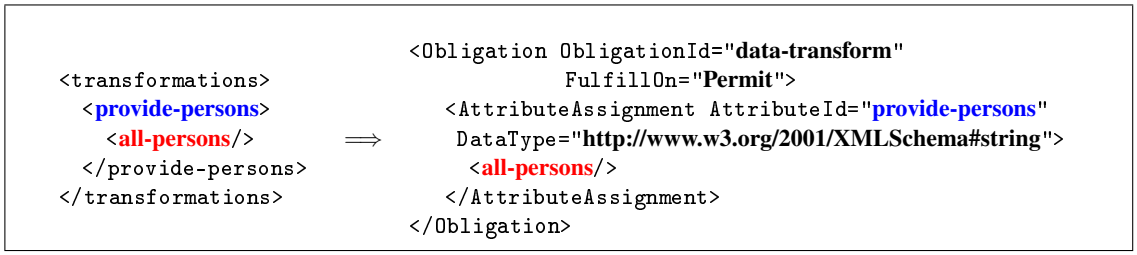


Figure 6.3.: Mapping Transformations to XACML Obligations

mission whose data type value is `http://www.w3.org/2001/XMLSchema#integer`, the remaining permissions have `http://www.w3.org/2001/XMLSchema#boolean` as their data type value. For the value of the *AttributeAssignment* element, we map all of the sub-hierarchy of the permissions to be the value of the corresponding *AttributeAssignment*. Fig. 6.3 illustrates how to map Transformation to *Obligation*.

Considering the Transformation of location data described in Section 4.3.3, we map *provide-location*, *provide-civic* and *provide-geo* permissions to be the value of the *AttributeId* attribute. The values of the *AttributeAssignment* element derived from the first and the last permissions are obtained from the values of the *profile* and *radius* attributes respectively while the value of the *AttributeAssignment* element derived from the second permission is the value of the *provide-civic* element itself. For the data type value of the *AttributeAssignment* element, we use `http://www.w3.org/2001/XMLSchema#string` for the first two permissions and `http://www.w3.org/2001/XMLSchema#integer` for the last permission.

Given the example of Transformations in Fig. 4.2, the XACML Obligations mapped from it are depicted in Listing 6.1.

Listing 6.1: XACML Obligations Derived from a Transformation for a Presence Document

```
<Obligations>
  <Obligation ObligationId="data-transform" FulfillOn="Permit">
    <AttributeAssignment AttributeId="provide-services"
      DataType="http://www.w3.org/2001/XMLSchema#string">
        <service-uri-scheme>sip</service-uri-scheme>
        <service-uri-scheme>mailto</service-uri-scheme>
    </AttributeAssignment>
    <AttributeAssignment AttributeId="provide-persons"
      DataType="http://www.w3.org/2001/XMLSchema#string">
        <all-persons/>
    </AttributeAssignment>
    <AttributeAssignment AttributeId="provide-activities"
      DataType="http://www.w3.org/2001/XMLSchema#boolean">
        true
    </AttributeAssignment>
    <AttributeAssignment AttributeId="provide-sphere"
      DataType="http://www.w3.org/2001/XMLSchema#boolean">
        true
    </AttributeAssignment>
    <AttributeAssignment AttributeId="provide-class"
      DataType="http://www.w3.org/2001/XMLSchema#boolean">
        true
    </AttributeAssignment>
```

```
      <AttributeAssignment AttributeId="provide-place-type"
        DataType="http://www.w3.org/2001/XMLSchema#boolean">
          true
      </AttributeAssignment>
      <AttributeAssignment AttributeId="provide-privacy"
        DataType="http://www.w3.org/2001/XMLSchema#boolean">
          true
      </AttributeAssignment>
      <AttributeAssignment AttributeId="provide-status-icon"
        DataType="http://www.w3.org/2001/XMLSchema#boolean">
          true
      </AttributeAssignment>
      <AttributeAssignment AttributeId="provide-note"
        DataType="http://www.w3.org/2001/XMLSchema#boolean">
          true
      </AttributeAssignment>
      <AttributeAssignment AttributeId="provide-location"
        DataType="http://www.w3.org/2001/XMLSchema#string">
          civic-transformation
      </AttributeAssignment>
      <AttributeAssignment AttributeId="provide-civic"
        DataType="http://www.w3.org/2001/XMLSchema#string">
          building
      </AttributeAssignment>
    </Obligation>
  </Obligations>
```

This *Obligation* is contained in an XACML policy stored at a policy repository where the PDP can retrieve it for its evaluation. With the obtained decision result from the PDP, if there is an *Obligation* where its *ObligationId* value is `data-transform`, the PEP will forward the *Obligation* to the Transformer Engine. According to the *AttributeAssignments* elements in the *Obligation*, the Transformer Engine knows which part of the outgoing data must be removed.

6.2. A Prototype Implementation

The aim of this prototype is to provide an implementation according to our proposed solution and to compare the computational time between our mechanism and the normal XACML PDP evaluation process. For measuring the computational time of both cases when the required part of the requested XML document has n nodes, our proposed mechanism requires time for one PDP evaluation plus one Transformer Engine process while the normal XACML PDP has to evaluate n individual requests where each individual request is specified for each node in the required data. We employ the Multiple Resource profile for the latter case. The PDP engine we employ in this test is from the widely used Sun XACML implementation [Sun].

We test our implementation according to the same scenario as described in Section 3.2.6 where a composite service consists of PresenceInfo and RestaurantFinder services. This composite service enables the users to obtain the list of restaurants nearby their location. Anna who is a Bob's friend wants to invite him for a dinner near his current location. She uses this composite service to retrieve a list of Bob's nearest restaurants.

6.2.1. Sun XACML Implementation

We modify the Sun XACML implementation in two points. First we update to use a standard Java API (JAXP) for evaluation of XPath Expressions instead of the original design that uses an external library. Second, to implement the Multiple Resource profile, we add some code to the Sun XACML engine to map the single XACML request for multiple resources to individual requests before PDP evaluation.

6.2.2. Transformer Engine

We have implemented the Transformer Engine (TE) in the Java programming language. From our scenario, we choose to focus only on the access control system of the PresenceInfo service. Thus, the implementation of the Transformer Engine in this prototype handles only the Transformations of presence information. The program consists of four classes; PresDoc_filter, CollectedTimeInMilliSec, Obligation_Interpretation, and DataFiltering. The class diagram of the TE is shown in Fig. 6.4. First, the class PresDoc_filter receives two input files (SAX parser is used to process the files), which are *Obligation* from the PEP and requested data from the service, and uses CollectedTimeInMilliSec class to measure the computational time of the program. The permissions in the Obligation file are extracted by the Obligation_Interpretation class. Then the requested data are filtered according to the obtained permissions by the DataFiltering class. Finally the actual permitted data are written to an output file.

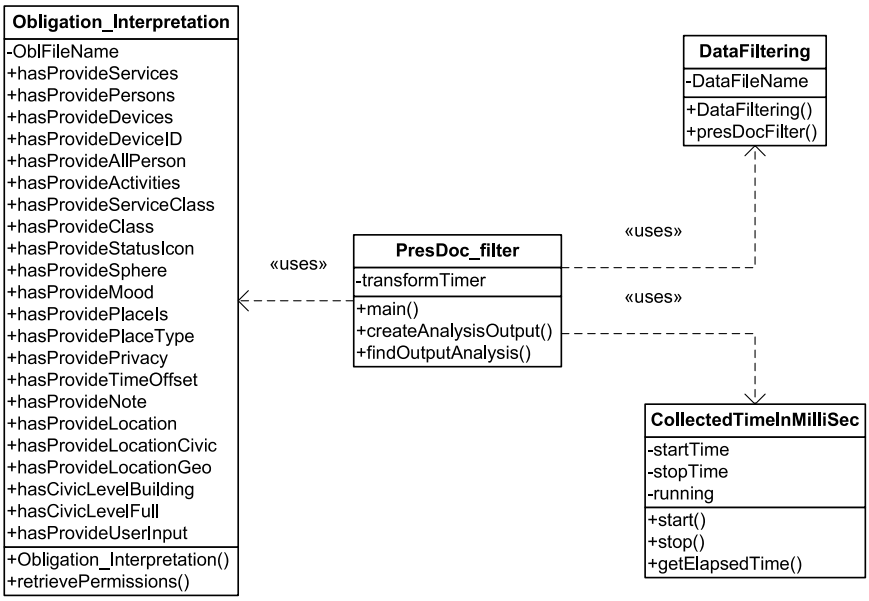


Figure 6.4.: Class Diagram of Transformer Engine

6.2.3. XACML Requests and XACML Policies Test Case

XACML Requests

When Anna uses the composite service, the RestaurantFinder service sends a request on behalf of Anna to the PresenceInfo service asking for Bob's location data. The request for our mechanism where the

requested resource is specified according to meaningful data part is given in Fig. 6.5 while the request for normal XACML evaluation using Multiple Resource profile is shown in Fig. 6.6.

```
Request
    { Subject { dataowner: Bob,
                subject-id: RestaurantFinder,
                on-behalf-of: Anna },
      Resource { ResourceContent: ... actual presence document ...,
                 Document: PresenceDocument,
                 resource-id: LocationData },
      Action { action-id: read }
    }
```

Figure 6.5.: An XACML Request for Our Proposed Scheme

```
Request
    { Subject { dataowner: Bob,
                subject-id: RestaurantFinder,
                on-behalf-of: Anna },
      Resource { ResourceContent: ... actual presence document ...,
                 Document: PresenceDocument,
                 resource-id: presence/tuple/status/geopriv,
                 profile: multiple:scope:xml,
                 scope: Descendants },
      Action { action-id: read }
    }
```

Figure 6.6.: An XACML Request for Normal XACML Scheme Using Multiple Resource Profile

Due to the Multiple Resource profile, the request also contains a *scope* Attribute with value `Descendants` and *profile* Attribute in the *Resource* element to indicate that all sub-nodes of the node element *geopriv* are also requested. We map this request to individual requests where each individual request expresses each node in the location data part. The individual requests are the same as the original request except that they do not contain *scope* and *profile* Attributes in the *Resource* element and the value of the *resource-id* Attribute is changed to each node in the location data part. Note that the *ResourceContent* element containing the actual resource (Bob's presence document as given in Listing 4.1) is required for the policy with XPath Expression. It is retrieved by the Context Handler as additional information from the original request. The real XACML requests employed for our mechanism in Fig. 6.5 and for the normal XACML scheme in Fig. 6.6 are illustrated in Listing E.1 and E.3 in Appendix E.

XACML Policies

Bob allows RestaurantFinder on behalf of Anna to access his location data. XACML policies expressing this permission employed for our mechanism and for the normal XACML scheme are given in Fig. 6.7 and 6.8 respectively.

```
Policy [RuleCombiningAlg: Permit-overrides]
    { Target { dataowner: Bob,
               subject-id: RestaurantFinder,
               on-behalf-of: Anna,
               document: PresenceDocument,
               action-id: read },
      Rule [id: AllowedPart, Effect: Permit]
        { Target { resource-id: LocationData } },
      Rule [id: ProhibitedPart, Effect: Deny],
      Obligations
            { Obligation [id: data-transform, FulfillOn: Permit]
                  { ... the same Obligation as Listing 6.1 ... } }
    }
```

Figure 6.7.: A Simple XACML Policy for Our Proposed Scheme

```
Policy [RuleCombiningAlg: Permit-overrides]
    { Target { dataowner: Bob,
               subject-id: RestaurantFinder,
               on-behalf-of: Anna,
               document: PresenceDocument,
               action-id: read },
      Rule [id: AllowedPart, Effect: Permit]
        { Target { (resource-id: presence/tuple/status/geopriv) or
                   (resource-id: presence/tuple/status/geopriv/location-info) or
                                ...
                   (resource-id: presence/tuple/status/geopriv/usage-rules/retransmission-allowed) or
                   (resource-id: presence/tuple/status/geopriv/usage-rules/retention-expiry)  } },
      Rule [id: ProhibitedPart, Effect: Deny],
    }
```

Figure 6.8.: A Simple XACML Policy for Normal XACML Scheme

Compared with the policy for our mechanism, the policy for the normal XACML scheme is the same except that it must specify all paths of allowed nodes of the location data in the *Target* element of the first rule and it does not contain XACML *Obligations*. The actual XACML policy for our mechanism and the full list of allowed node paths of location data are given in Listing E.3 and E.4 in Appendix E respectively.

6.2.4. Evaluation Results

Results from the Proposed Solution

Applying our mechanism, the request shown in Fig. 6.5 is evaluated by the Policy Decision Point (PDP) only once instead of evaluating several individual requests. The result of this evaluation for the policy in Fig. 6.7 is contained in the XACML response illustrated in Fig. 6.9. According to Bob's policy, the authorization decision is `Permit` and hence an *Obligation* having an *Effect* value corresponding to the decision is also provided in the response message. The PDP sends the response to the Policy

```
<Response>
  <Result ResourceID="LocationData">
    <Decision>Permit</Decision>
    <Status>
      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
    </Status>
    <Obligations>
      <Obligation ObligationId="data-transform" FulfillOn="Permit">
              . . . the same Obligation as Listing 6.1 . . .
      </Obligation>
    </Obligations>
  </Result>
</Response>
```

Figure 6.9.: An XACML Response of RestaurantFinder Service's Request on Behalf of Anna for Bob's Location Information

```
<geopriv>
  <location-info>
    <civicAddress>
      <country>US</country>
      <A1>New York</A1>
      <A3>New York</A3>
      <A6>Broadway</A6>
      <HNO>123</HNO>
      <PC>10027-0401</PC>
    </civicAddress>
  </location-info>
  <usage-rules>
    <retransmission-allowed>yes</retransmission-allowed>
    <retention-expiry>2013-06-23T04:57:29Z</retention-expiry>
  </usage-rules>
</geopriv>
```

Figure 6.10.: Bob's Location Information after Filtering by Transformer Engine

Enforcement Point (PEP).

To enforce the policy for a positive result, the PEP forwards the request to the PresenceInfo service to retrieve location data and sends the *Obligation* with *ObligationId* `data-transform` to the Transformer Engine (TE). The obtained location information is filtered by the TE in accordance with the *Obligation*. The actual location data (output data from the TE) to be sent out as RestaurantFinder's request is depicted in Fig. 6.10. Since Bob allows RestaurantFinder service on behalf of Anna to read his location information as civic location in "building" level where the *LOC* (additional location information) data is prohibited, the *LOC* element and its value are cut out.

Results from the Normal XACML Evaluation

The response message from the PDP after evaluating all individual requests against their policy is shown in Listing 6.2. This response contains all evaluation results of each node in the location data. We can notice that the evaluation result of the prohibited data i.e. "LOC" node (represented by "ResourceID="presence/tuple/status/geopriv/location-info/civicAddress/LOC"" in the last result) is

```
Deny.
```

Listing 6.2: An XACML Response of RestaurantFinder Service's Request for Bob's Location Information

```
<Response>
  <Result
   ResourceID="presence/tuple/status/geopriv/location-info/civicAddress/A6">
    <Decision>Permit</Decision>
    <Status>
      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
    </Status>
  </Result>
  <Result
   ResourceID="presence/tuple/status/geopriv/usage-rules/retention-expiry">
    <Decision>Permit</Decision>
    <Status>
      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
    </Status>
  </Result>
  <Result
   ResourceID="presence/tuple/status/geopriv/location-info/civicAddress/A3">
    <Decision>Permit</Decision>
    <Status>
      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
    </Status>
  </Result>
  <Result
   ResourceID="presence/tuple/status/geopriv/location-info/civicAddress/PC">
    <Decision>Permit</Decision>
    <Status>
      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
    </Status>
  </Result>
  <Result ResourceID="presence/tuple/status/geopriv/location-info/civicAddress/
country">
    <Decision>Permit</Decision>
    <Status>
      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
    </Status>
  </Result>
  <Result ResourceID="presence/tuple/status/geopriv/location-info">
    <Decision>Permit</Decision>
    <Status>
      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
    </Status>
  </Result>
  <Result
   ResourceID="presence/tuple/status/geopriv/location-info/civicAddress">
    <Decision>Permit</Decision>
    <Status>
      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
    </Status>
  </Result>
  <Result ResourceID="presence/tuple/status/geopriv">
    <Decision>Permit</Decision>
    <Status>
      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
    </Status>
  </Result>
  <Result ResourceID="presence/tuple/status/geopriv/usage-rules">
    <Decision>Permit</Decision>
    <Status>
      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
    </Status>
  </Result>
```

```
  <Result
   ResourceID="presence/tuple/status/geopriv/location-info/civicAddress/A1">
    <Decision>Permit</Decision>
    <Status>
      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
    </Status>
  </Result>
  <Result ResourceID="presence/tuple/status/geopriv/usage-rules/
retransmission-allowed">
    <Decision>Permit</Decision>
    <Status>
      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
    </Status>
  </Result>
  <Result
   ResourceID="presence/tuple/status/geopriv/location-info/civicAddress/HNO">
    <Decision>Permit</Decision>
    <Status>
      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
    </Status>
  </Result>
  <Result
   ResourceID="presence/tuple/status/geopriv/location-info/civicAddress/LOC">
    <Decision>Deny</Decision>
    <Status>
      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
    </Status>
  </Result>
</Response>
```

After evaluation of all nodes of the required data, the PEP forwards the results to the service application to obtain the data where only the allowed parts are retrieved. The output location data that will be sent to the RestaurantFinder service is the same as in the Fig. 6.10.

6.2.5. Performance Evaluation

To investigate the performance of PDP evaluation, we test our mechanism and the normal XACML scheme with two parameters that are the number of nodes of the requested data and the number of XPath Expressions to represent complex policies. The former is relevant since how many times the normal XACML scheme evaluates requests depends on it. In addition to the request for location data (containing 13 nodes), we thus test the requests for data parts that have more nodes, i.e. service data (21 nodes) and presence document (39 nodes). The latter is to show how our mechanism can improve the performance of the XACML evaluation process when policies become more complex. Besides the simple XACML policies (without XPath Expression) in Fig. 6.7 and 6.8, we create two other types of policies containing one XPath Expression and two XPath Expressions. With XPath Expression, a more complex policy can be specified. For example, in our scenario Bob does not want to be disturbed by anyone when he is sad and/or when he is at home. The two policies extend the simple policies with the condition that Bob will allow the RestaurantFinder service to access his location data on behalf of Anna when he is not in a sad mood and when he is neither in a sad mood nor at home respectively. The actual *Condition* elements expressing one and two XPath Expressions are given in Listing E.5 and E.6 in Appendix E.

We use Windows7 64-bit with Intel Core 2 Duo CPU 3.16 GHz, 4 GB of RAM. The overall performance of the XACML engine can be considered in two phases; a) loading time of policy and b)

request evaluation against the loaded policy. Fig. 6.11 and 6.12 illustrate loading time of policies used

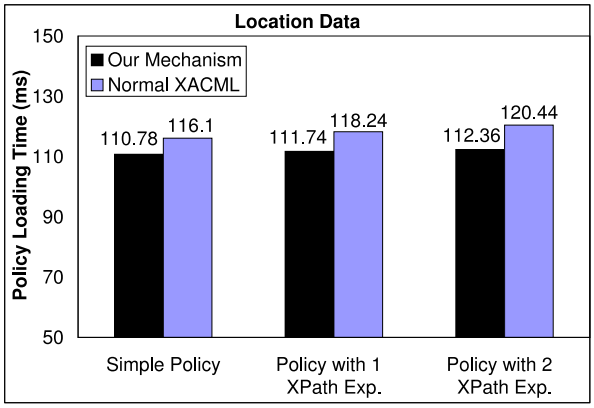


Figure 6.11.: Loading Time Comparison of Policies for Location Data between Our Mechanism and Normal XACML

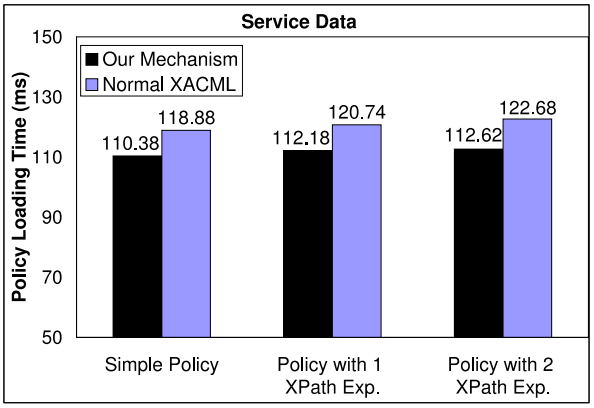


Figure 6.12.: Loading Time Comparison of Policy for Service Data between Our Mechanism and Normal XACML

in our mechanism and normal XACML scheme, that govern access to location data and service data respectively. The result shows that in all three policy cases, policies used by normal XACML take a bit more time for loading than the policies used by our mechanism. This is because the policies for the former are larger than the policies for the latter. In addition, this increase in time in case of the service data is higher than the location data. This is because the service data has more permitted nodes. Thus, the policy for the service data contains more nodes in the first rule which makes it bigger than the policy for the location data. Though the difference of the loading time is quite small, our scheme could bring more benefits when the number of permitted nodes of the requested data, the number of policies to be loaded and the frequency of policy loading are high.

Fig. 6.13 and 6.14 depict the evaluation time comparison for location data and service data be-

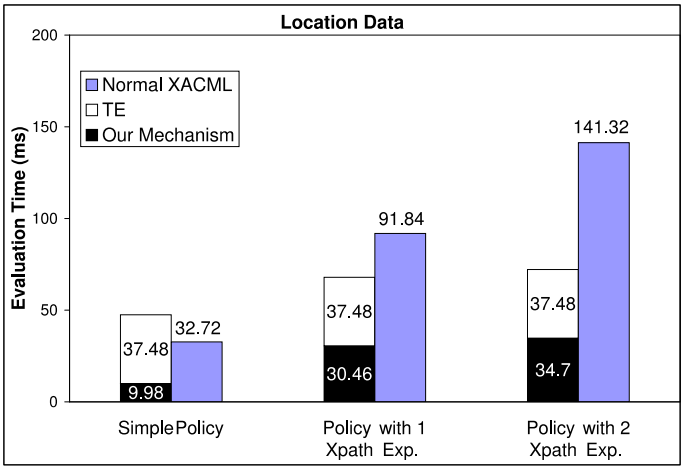


Figure 6.13.: Evaluation Time Comparison for Location Data Access between Our Mechanism and Normal XACML

tween our mechanism and normal XACML respectively. For the location data, consider only the PDP

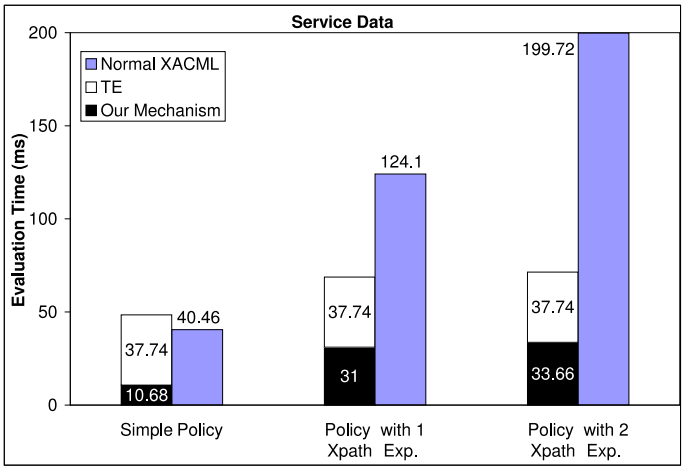


Figure 6.14.: Evaluation Time Comparison for Service Data between Our Mechanism and Normal XACML

evaluation time for simple policy, our mechanism (9.98 ms) takes less time than the normal XACML (32.72 ms) by 69%. However the overall evaluation time of our mechanism must include the time of Transformer Engine (TE) process (37.48 ms) which makes our mechanism bring overhead of 45% to the normal XACML evaluation for the simple policy case. On the contrary, our mechanism can bring saving of 26% and 49% when the policy contains conditions using one and two XPath Expressions respectively. For the service data, the result is similar to the case of location data that our mechanism brings overhead of 20% to the normal XACML evaluation for the simple policy case but brings savings of 45% and 64% to the normal XACML evaluation for one and two XPath Expressions respectively.

Comparing the result of XPath Expressions between location data and service data, the result depicts that the increase in time of the normal XACML evaluation for service data from our mechanism is higher than the increase in time of the evaluation for location data. This is because the service data contains more nodes than the location data thus the PDP has to evaluate more times for the service data than the location data. Thus, in case of complex policies using XPath Expressions, the more nodes the requested part contains, the more benefits our mechanism can provide.

We also measure the evaluation time of the simple policy case for presence document. The result is shown in Fig. 6.15 in comparison with the location and service data. Though our mechanism takes more time than normal XACML in the location data and service data cases, the additional time our mechanism introduces for service data is less than the additional time for location data. This is because service data has more nodes than location data. Moreover, in case of presence documents where the number of nodes is even higher, our mechanism takes less time than normal XACML. Thus we can conclude for simple policy case in the same direction as complex policy that our mechanism can bring an advantage when the number of nodes is high.

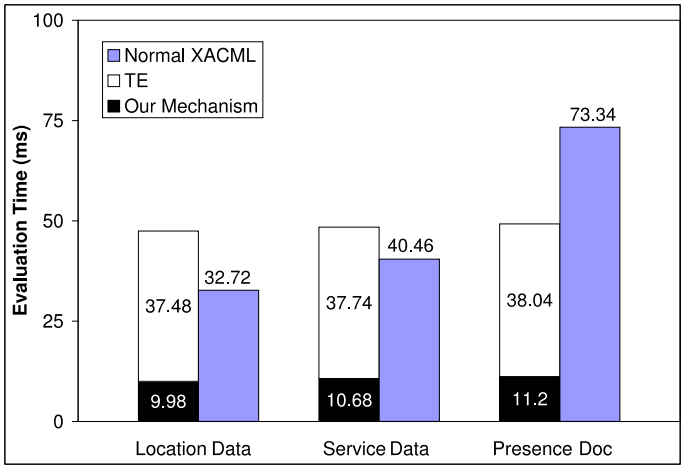


Figure 6.15.: Evaluation Time Comparison of Simple Policy between Our Mechanism and Normal XACML for Location Data, Service Data and Presence Document

6.3. Related Work

Hsieh et. al [HFE+09] proposed a way of providing fine-grained access control for digital content by extending XACML policy to contain actual data where different data parts are embedded in different rules. The PDP has to evaluate all rules to find all permitted data parts which will be listed in the response message. The PEP concatenates all data parts and sends them to the requester. Similar to our work, this work can administrate access control policy to govern data at the level of data parts. Unlike this work, our work separates policy from the actual data and we do not have to extend the XACML policy.

Rota et. al [RSR10] proposed fine-grained access control for XML documents by extending XACML with ontologies. Data parts are described as domain concepts in an ontology. XACML policies express which concepts are permitted or prohibited. The association of concepts and their XML elements are described in another ontology. New XACML functions were designed to enable querying on the knowledge base. According to the querying result, the PDP provides authorization decision to an XML filter that creates an XML view consistent with the decision before sending it to the requester. Although, this work allows an access control policy to be expressed at the level of data parts like our work, it is different from us in that the PDP still has to evaluate each node in the requested part one by one.

Herrmann [Her10, Her11] proposed access control systems using (Geo)XACML and their administration for spatial data infrastructures by introducing an administrative model of access control policies that is suitable in large service-oriented architecture. This model presents layers of access control systems to administrate access control policies of previous layers. Guidelines are defined to ease administration of complex access control policies. Though the aim of this work is similar to ours, that is to improve complex access control policy administration, but its solution is different by providing additional access control systems.

6.4. Conclusion

We have proposed a way to ease policy administration of access control system in composite services environment by introducing meaningful data part. Our mechanism is based on post-processing concept which enables the PDP to evaluate the request only once. We have developed an implementation prototype and measured the computational time used in our mechanism to compare with normal XACML scheme. The implementation results show that though our mechanism can bring more computational overhead when the number of nodes is low, it can reduce the overhead when the number of nodes is high.

7. Conclusion

Online services in the beginning were single where each service developed its own technology to serve its customers. To provide services to users, personal data of the users are collected which urges privacy laws for the online sector. In general, the laws require the data controller (service) to provide sufficient protection to personal data and to inform the users about its privacy practices for the data it wants to collect. Though the online services, e.g. websites, normally announce their privacy policies on what they intend to do with the data in natural language, this information is normally long and sometimes hard to understand. Most naive users then ignore this information which could result in privacy breaches. W3C, thus, provided a technical approach called P3P which enables websites to express their privacy practices in a standard machine-readable format. These privacy practices will be compared with users' preferences by P3P user agents. The result of this comparison assists the users in their decision to use the websites. Therefore, with P3P users do not have to read the websites' privacy policies by themselves and the matching between the privacy policies and the users' preferences can be done automatically. When the users use the service and their data has already been collected by the service, an important mechanism to protect users' privacy called XACML access control is employed to govern access to their personal data.

The P3P policy language itself has some problems that its flexible syntax and some combinations of its standardized vocabulary may introduce semantic inconsistencies. Despite of these problems, the P3P specification does not define any means to solve them. Yu et al. [YLA04] proposed to mitigate these problems by introducing a formal semantics for P3P using relational tables. Based on a data–centric model, this work defined some constraints to check semantic inconsistencies. However, the constraints for conflicts occurring from some vocabulary were not identified in detail.

To survive in the highly competitive market, online services have been adapted such that more advanced services (composite services) can be composed of multiple existing services with less developing time. With this change, questions arise on P3P how to deal with conflicts from multiple existing policies that may cause failure of composite services at runtime and how to build policies for a service combination from existing policies since these policies may address the same data item. In addition, in case the users are already customers of an existing service in a composite service, there is also the question on how the users can give consent in this situation.

From these challenges, we have introduced an ontology for P3P using OWL which adds a formal semantics to P3P instead of only a taxonomy as in its original. Our P3P ontology is constructed based on a data–purpose centric model. We have defined constraints so that the potential semantic inconsistencies can be verified. The data–purpose centric model provides more fine-grained interpretation of P3P than the data–centric model from Yu et al. which in turn allows us to also verify the relations

between purpose and recipient, and purpose and retention components. Our P3P ontology is designed in such a way that we can tell the reason for the semantic inconsistencies. Instead of specifying logical constraints in the ontology, special classes are created to capture those constraints violations. Since a P3P policy retrieved from each website is unique, we view each P3P policy as an individual of the P3P ontology. Thus, the reasoning task *instance checking* is used in our application to verify a P3P policy and the reasons of conflict are known by querying which special classes the P3P policy individual is an instance of. We have crawled various websites and retrieved five hundreds P3P policies. These policies were verified with our P3P verification tool where we found that more than half have semantic inconsistencies and the most frequent cause of conflict is from unconformity between *Purpose* and *Data Categories*. Besides, we have extended the P3P ontology with additional constraints such that the conflicts between privacy policies of service members can also be verified with the same tool. We have proposed a combining mechanism to resolve the privacy policies of composite services and extended some elements of the P3P policy language that support our combining mechanism. This extension allows more concrete information to be expressed. When an active service, in which user data are already collected and stored, cooperates with new services, users can give their consent to those services through specifying access control policies.

In the XACML access control system, a service providing a resource specifies XACML policies customized by the users who are the resource owners to control access to their resource. The XACML access control model was designed to evaluate policies against a given request for a single resource at a time. If the resource is not structured, the requester can obtain either the entire resource when it is allowed or nothing when it is prohibited. If the resource is structured, the users can specify their policies to control access to each element in the resource. This fine-grained access control though bringing better privacy protection, also introduces more policy administration complexity since the policies must be specified for each element instead of an entire resource, and more calculation expenses since each element is evaluated by the XACML engine one by one. In a composite service, some member services may need different parts of a resource. In order to provide resource in parts, fine-grained access control must be employed. Based on these circumstances, it is thus a challenge for XACML when fine-grained access control is required on how the sophisticated policies can be built easier and is there a way to improve the efficiency of the XACML decision evaluation algorithm for hierarchical resources?

We, therefore, try to find a better way that can provide fine-grained access control of the resource with more efficient performance and less complex policy administration. Our idea is to make access control as coarse-grained as possible to reduce the complexity of the policy administration. Instead of specifying policies for each element, we have proposed to specify the policies for meaningful parts. To provide fine-grained access control of each part, we add detailed information of allowed or forbidden elements in that part which is specified in an XACML Obligation. This Obligation will be processed by the Transformer Engine to filter out the forbidden elements in the requested part before the response is passed to the requester. This scheme aids policy administration and reduces XACML evaluation process to only one time since only meaningful parts are specified and evaluated by the XACML

engine instead of all elements in the requested part. We have implemented a prototype and tested it with XML documents. Though our scheme adds a calculation overhead by the Transformer Engine, it can bring benefit to the performance of the access control system when the number of elements is high. Moreover, it can bring large advantages when XPath Expressions are employed.

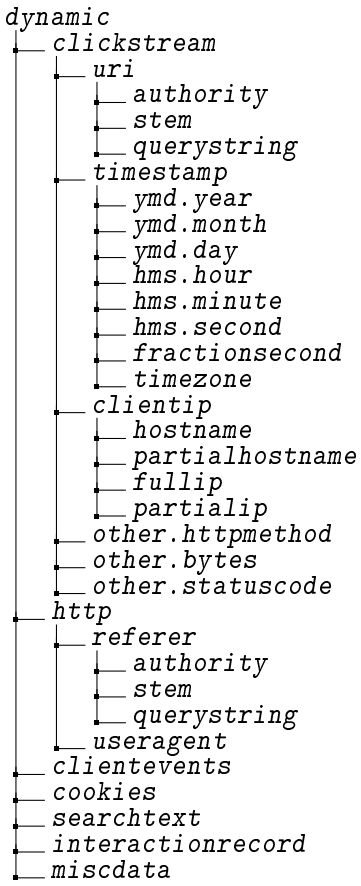
In this research, we have developed concepts to deal with various issues when setting privacy policies for composite services as explained above. Our proposed solutions are based on industrial standards. The idea for specifying XACML policies could be employed not only in composite service environment but also in general access control system protecting hierarchical resources. With on-the-fly service creation, policies have to be formulated and customized on the fly. This implies that user interaction is always essential and customization is required more often. Whether this is realistic is still an open issue which can be determined by real world practices in the future. Though in fact, there are more unsolved problems in privacy aspect such as to understand what users actually want to protect their privacy when interacting with online services, at least our solutions have improved privacy protection for users via a technical point of view.

Future Work

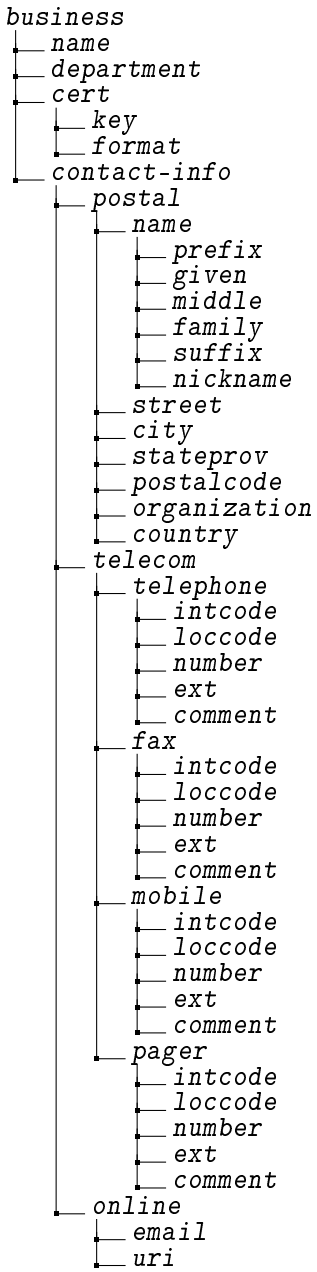
Besides privacy policy from service providers' side, privacy preference from users' side is an interesting area to research on. There are several open issues regarding to it. First, there is no standard available yet for the user preference formats. Though APPEL was proposed as a user preference language for P3P policy, it was not accepted as a standard by W3C due to shortcomings from its fundamental design, e.g. it allows only for negative rules [LYA03, AKSX03b, APP]. Second, end-users without technical background are not keen to define their privacy preferences which are normally quite complex to them. One reason is because they do not know the impacts when their data are released to the services. Thus, how to make the ordinary users realize the potential danger of data disclosure is a challenge. Last, though some users are willing to define their privacy preferences, it is still another challenge on how well the users can understand the actual meaning of, e.g. terms and ideas used in a user interface such that they can set their privacy preferences correctly.

A. P3P Base Data Structure

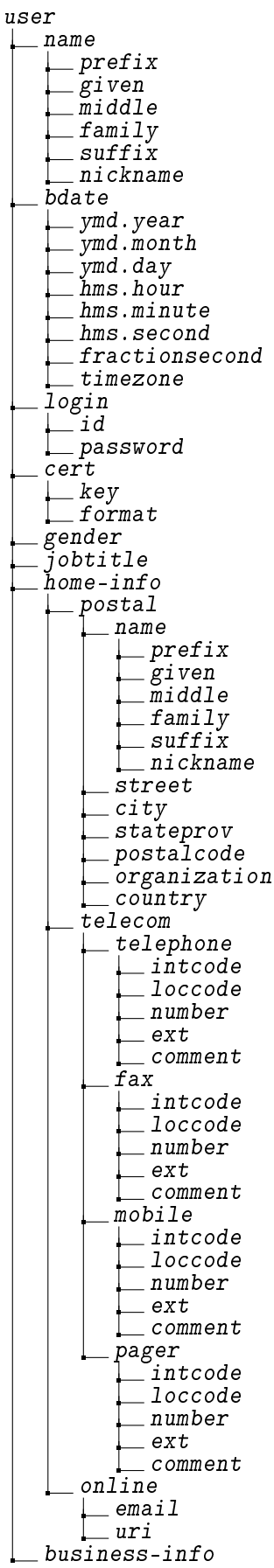
Dynamic Data



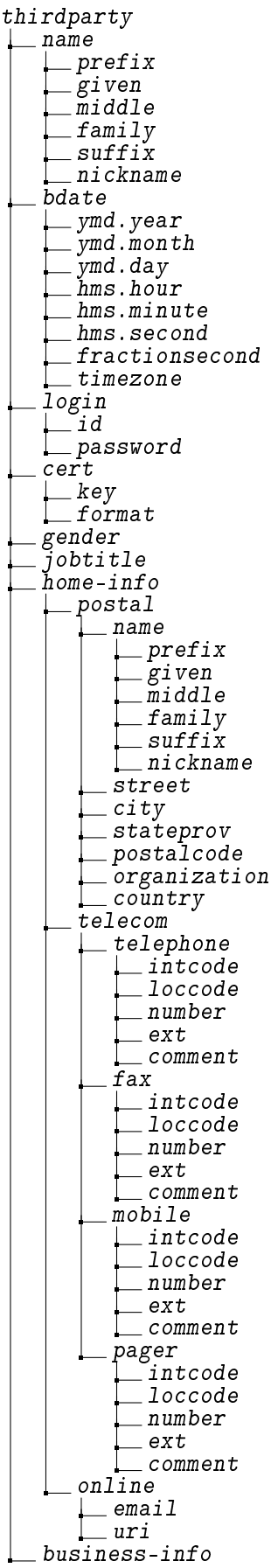
Business Data

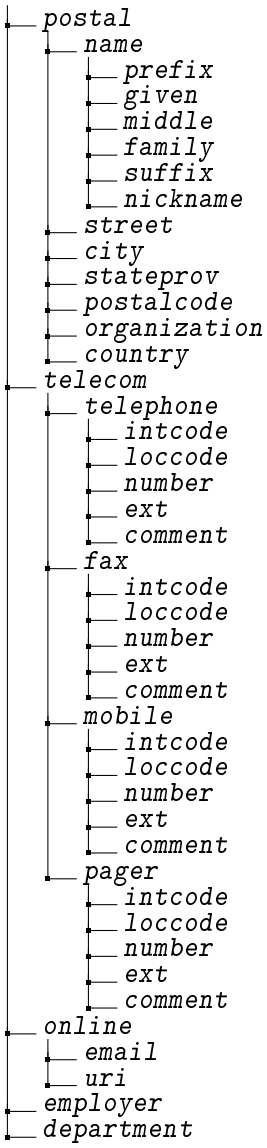


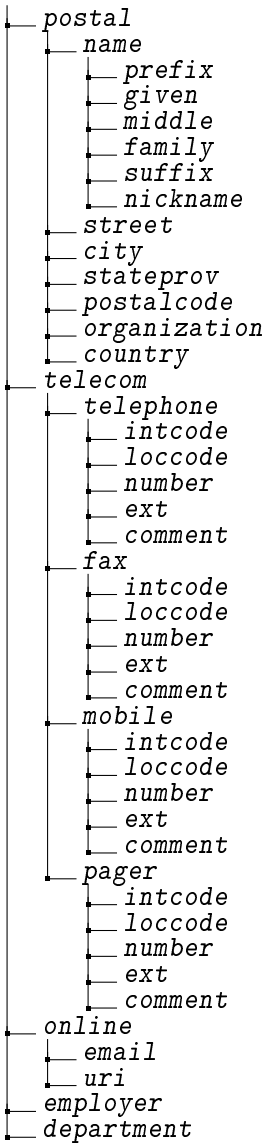
User Data



Third Party Data




postal
name
prefix
given
middle
family
suffix
nickname
street
city
stateprov
postalcode
organization
country
telecom
telephone
intcode
loccode
number
ext
comment
fax
intcode
loccode
number
ext
comment
mobile
intcode
loccode
number
ext
comment
pager
intcode
loccode
number
ext
comment
online
email
uri
employer
department


postal
name
prefix
given
middle
family
suffix
nickname
street
city
stateprov
postalcode
organization
country
telecom
telephone
intcode
loccode
number
ext
comment
fax
intcode
loccode
number
ext
comment
mobile
intcode
loccode
number
ext
comment
pager
intcode
loccode
number
ext
comment
online
email
uri
employer
department

B. Allowed Data Categories

The following table shows data and their allowed categories.

Type	**Allowed Categories**
Dynamic	Variable
User	Physical Contact Information, Demographic and Socioeconomic Data, Unique Identifiers, Online Contact Information
Third-party	Physical Contact Information, Demographic and Socioeconomic Data, Unique Identifiers, Online Contact Information
Business	
orgname	Physical Contact Information, Demographic and Socioeconomic Data
name	Physical Contact Information, Demographic and Socioeconomic Data
bdate	Demographic and Socioeconomic Data
login	Unique Identifiers
cert	Unique Identifiers
gender	Demographic and Socioeconomic Data
employer	Demographic and Socioeconomic Data
department	Demographic and Socioeconomic Data
jobtitle	Demographic and Socioeconomic Data
home-info	Physical Contact Information, Online Contact Information, Demographic and Socioeconomic Data
contact-info	Physical Contact Information, Online Contact Information, Demographic and Socioeconomic Data
business-info	Physical Contact Information, Online Contact Information, Demographic and Socioeconomic Data
clickstream	Navigation and Click-stream Data, Computer Information
http	Navigation and Click-stream Data, Computer Information
clientevents	Navigation and Click-stream Data
cookies	Variable
miscdata	Variable
searchtext	Interactive Data
interactionrecord	Interactive Data

Type	Allowed Categories
ymd.year	Variable
ymd.month	Variable
ymd.day	Variable
hms.hour	Variable
hms.minute	Variable
hms.second	Variable
fractionsecond	Variable
timezone	Variable
prefix	Demographic and Socioeconomic Data
given	Physical Contact Information
family	Physical Contact Information
middle	Physical Contact Information
suffix	Demographic and Socioeconomic Data
nickname	Demographic and Socioeconomic Data
id	Unique Identifiers
password	Unique Identifiers
key	Unique Identifiers
format	Unique Identifiers
postal	Physical Contact Information, Demographic and Socioeconomic Data
telecom	Physical Contact Information
online	Online Contact Information
intcode	Physical Contact Information
loccode	Physical Contact Information
number	Physical Contact Information
ext	Physical Contact Information
comment	Physical Contact Information
street	Physical Contact Information
city	Demographic and Socioeconomic Data
stateprov	Demographic and Socioeconomic Data
postalcode	Demographic and Socioeconomic Data
country	Demographic and Socioeconomic Data
organization	Demographic and Socioeconomic Data
telephone	Physical Contact Information
fax	Physical Contact Information
mobile	Physical Contact Information
pager	Physical Contact Information

Type	Allowed Categories
email	Online Contact Information
uri	Online Contact Information
authority	Variable
stem	Variable
querystring	Variable
hostname	Computer Information
partialhostname	Demographic
fullip	Computer Information
partialip	Demographic
uri	Navigation and click-stream data
timestamp	Navigation and click-stream data
clientip	Computer Information, Demographic and Socioeconomic Data
other.httpmethod	Navigation and click-stream data
other.bytes	Navigation and click-stream data
other.statuscode	Navigation and click-stream data
referer	Navigation and click-stream data
useragent	Computer Information

C. XML Schema Definition of Our Purposed P3P Extension Elements

XML Schema Definition for P3P1.2 Extensions.

```
<xs:schema
  targetNamespace="http://www.w3.org/2011/P3Pv12"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:p3p12="http://www.w3.org/2011/P3Pv12">

<!-- *********** P3P 1.2 Explicit Recipient List Element ************ -->
<xs:element name="explicit-recipient-list" type="p3p12:ExplicitRecipientListType"/>

<xs:complexType name="ExplicitRecipientListType"/>
  <xs:choice minOccurs="1" maxOccurs="unbounded">
    <xs:element name="ours-list" type="p3p12:ExplicitRecipientType"/>
    <xs:element name="same-list" type="p3p12:ExplicitRecipientType"/>
    <xs:element name="delivery-list" type="p3p12:ExplicitRecipientType"/>
    <xs:element name="other-recipient-list" type="p3p12:ExplicitRecipientType"/>
    <xs:element name="unrelated-list" type="p3p12:ExplicitRecipientType"/>
  </xs:choice>
</xs:complexType>

<xs:complexType name="ExplicitRecipientType"/>
  <xs:sequence maxOccurs="unbounded">
    <xs:element name="recipient-id" type="xs:anyURI"/>
  </xs:sequence>
</xs:complexType>

<!-- *********** P3P 1.2 Retention Duration Element ************ -->
<xs:element name="retentionDuration" type="durationType"/>

<xs:complexType name="durationType"/>
  <xs:choice>
    <xs:element name="stated-purpose-duration" type="p3p12:stated-purpose-durationType"/>
    <xs:element name="legal-requirement-duration" type=" xs:duration"/>
    <xs:element name="business-practices-duration" type=" xs:duration"/>
  </xs:choice>
</xs:complexType>

<xs:complexType name="stated-purpose-durationType"/>
  <xs:choice minOccurs="1" maxOccurs="unbounded">
    <xs:element name="current-duration" type="xs:duration"/>
    <xs:element name="admin-duration" type="xs:duration"/>
    <xs:element name="develop-duration" type="xs:duration"/>
    <xs:element name="tailoring-duration" type="xs:duration"/>
    <xs:element name="pseudo-analysis-duration" type="xs:duration"/>
    <xs:element name="pseudo-decision-duration" type="xs:duration"/>
    <xs:element name="individual-analysis-duration" type="xs:duration"/>
    <xs:element name="individual-decision-duration" type="xs:duration"/>
    <xs:element name="contact-duration" type="xs:duration"/>
```

```
    <xs:element name="historical-duration" type="xs:duration"/>
    <xs:element name="telemarketing-duration" type="xs:duration"/>
  </xs:choice>
</xs:complexType>
</xs:schema>
```

XML Schema Definition of Shared Data.

```
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:p3p="http://www.w3.org/2002/01/P3Pv1"
  targetNamespace="http://www.w3.org/2011/P3Pv12/SharedData">

<import
    namespace="http://www.w3.org/2002/01/P3Pv1"
    schemaLocation="http://www.w3.org/2002/01/P3Pv1.xsd"/>

<xs:element name="SharedData">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="InputSharedData" type="p3p:data-group-type"/>
      <xs:element name="OutputSharedData" type="p3p:data-group-type"/>
     </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

D. Invalid Policy Class Definitions

This section shows the full description of three invalid policy classes, i.e. InvalidPolicy–DH–PurposeOpt (β_6), InvalidPolicy–DH–RecipientOpt (β_7) and InvalidPolicy–DH–Retention (β_8) respectively.

β_6 InvalidPolicy–DH–PurposeOpt $\equiv$
Policy $\sqcap$ ($\exists$collects.(CollectedData $\sqcap$
(((($\exists$hasPart.(Admin $\sqcap$ ($\exists$optionality.OptIn))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Admin $\sqcap$ ($\exists$optionality.(Always $\sqcup$ OptOut)))))) $\sqcup$
(($\exists$hasPart.(Admin $\sqcap$ ($\exists$optionality.OptOut))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Admin $\sqcap$ ($\exists$optionality.Always)))))) $\sqcup$
((($\exists$hasPart.(Develop $\sqcap$ ($\exists$optionality.OptIn))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Develop $\sqcap$ ($\exists$optionality.(Always $\sqcup$ OptOut)))))) $\sqcup$
(($\exists$hasPart.(Develop $\sqcap$ ($\exists$optionality.OptOut))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Develop $\sqcap$ ($\exists$optionality.Always)))))) $\sqcup$
((($\exists$hasPart.(Tailoring $\sqcap$ ($\exists$optionality.OptIn))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Tailoring $\sqcap$ ($\exists$optionality.(Always $\sqcup$ OptOut)))))) $\sqcup$
(($\exists$hasPart.(Tailoring $\sqcap$ ($\exists$optionality.OptOut))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Tailoring $\sqcap$ ($\exists$optionality.Always)))))) $\sqcup$
((($\exists$hasPart.(PseudoAnalysis $\sqcap$ ($\exists$optionality.OptIn))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(PseudoAnalysis $\sqcap$ ($\exists$optionality.(Always $\sqcup$ OptOut)))))) $\sqcup$
(($\exists$hasPart.(PseudoAnalysis $\sqcap$ ($\exists$optionality.OptOut))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(PseudoAnalysis $\sqcap$ ($\exists$optionality.Always)))))) $\sqcup$
((($\exists$hasPart.(PseudoDecision $\sqcap$ ($\exists$optionality.OptIn))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(PseudoDecision $\sqcap$ ($\exists$optionality.(Always $\sqcup$ OptOut)))))) $\sqcup$
(($\exists$hasPart.(PseudoDecision $\sqcap$ ($\exists$optionality.OptOut))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(PseudoDecision $\sqcap$ ($\exists$optionality.Always)))))) $\sqcup$
((($\exists$hasPart.(IndividualAnalysis $\sqcap$ ($\exists$optionality.OptIn))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(IndividualAnalysis $\sqcap$ ($\exists$optionality.(Always $\sqcup$ OptOut)))))) $\sqcup$
(($\exists$hasPart.(IndividualAnalysis $\sqcap$ ($\exists$optionality.OptOut))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(IndividualAnalysis $\sqcap$ ($\exists$optionality.Always)))))) $\sqcup$
((($\exists$hasPart.(IndividualDecision $\sqcap$ ($\exists$optionality.OptIn))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(IndividualDecision $\sqcap$ ($\exists$optionality.(Always $\sqcup$ OptOut)))))) $\sqcup$
(($\exists$hasPart.(IndividualDecision $\sqcap$ ($\exists$optionality.OptOut))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(IndividualDecision $\sqcap$ ($\exists$optionality.Always)))))) $\sqcup$
((($\exists$hasPart.(Contact $\sqcap$ ($\exists$optionality.OptIn))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Contact $\sqcap$ ($\exists$optionality.(Always $\sqcup$ OptOut)))))) $\sqcup$
(($\exists$hasPart.(Contact $\sqcap$ ($\exists$optionality.OptOut))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Contact $\sqcap$ ($\exists$optionality.Always)))))) $\sqcup$
((($\exists$hasPart.(Historical $\sqcap$ ($\exists$optionality.OptIn))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Historical $\sqcap$ ($\exists$optionality.(Always $\sqcup$ OptOut)))))) $\sqcup$
(($\exists$hasPart.(Historical $\sqcap$ ($\exists$optionality.OptOut))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Historical $\sqcap$ ($\exists$optionality.Always)))))) $\sqcup$
((($\exists$hasPart.(Telemarketing $\sqcap$ ($\exists$optionality.OptIn))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Telemarketing $\sqcap$ ($\exists$optionality.(Always $\sqcup$ OptOut)))))) $\sqcup$
(($\exists$hasPart.(Telemarketing $\sqcap$ ($\exists$optionality.OptOut))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Telemarketing $\sqcap$ ($\exists$optionality.Always)))))) $\sqcup$
((($\exists$hasPart.(OtherPurpose $\sqcap$ ($\exists$optionality.OptIn))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(OtherPurpose $\sqcap$ ($\exists$optionality.(Always $\sqcup$ OptOut)))))) $\sqcup$
(($\exists$hasPart.(OtherPurpose $\sqcap$ ($\exists$optionality.OptOut))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(OtherPurpose $\sqcap$ ($\exists$optionality.Always))))))))

β_7 InvalidPolicy–DH–RecipientOpt $\equiv$ Policy $\sqcap$ ($\exists$collects.(CollectedData $\sqcap$
(((($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.(Same $\sqcap$ ($\exists$optionality.OptIn))))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.(Same $\sqcap$ ($\exists$optionality.(Always $\sqcup$ OptOut)))))))) $\sqcup$
(($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.(Same $\sqcap$ ($\exists$optionality.OptOut))))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.(Same $\sqcap$ ($\exists$optionality.Always)))))))) $\sqcup$
((($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.(Delivery $\sqcap$ ($\exists$optionality.OptIn))))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.(Delivery $\sqcap$ ($\exists$optionality.(Always $\sqcup$ OptOut)))))))) $\sqcup$
(($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.(Delivery $\sqcap$ ($\exists$optionality.OptOut))))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.(Delivery $\sqcap$ ($\exists$optionality.Always)))))))) $\sqcup$
((($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.(OtherRecipient $\sqcap$ ($\exists$optionality.OptIn))))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.(OtherRecipient $\sqcap$
($\exists$optionality.(Always $\sqcup$ OptOut)))))))) $\sqcup$
(($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.(OtherRecipient $\sqcap$ ($\exists$optionality.OptOut))))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.(OtherRecipient $\sqcap$ ($\exists$optionality.Always)))))))) $\sqcup$
((($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.(Unrelated $\sqcap$ ($\exists$optionality.OptIn))))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.(Unrelated $\sqcap$
($\exists$optionality.(Always $\sqcup$ OptOut)))))))) $\sqcup$
(($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.(Unrelated $\sqcap$ ($\exists$optionality.OptOut))))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.(Unrelated $\sqcap$ ($\exists$optionality.Always)))))))) $\sqcup$
((($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.(Public $\sqcap$ ($\exists$optionality.OptIn))))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.(Public $\sqcap$ ($\exists$optionality.(Always $\sqcup$ OptOut)))))))) $\sqcup$
(($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.(Public $\sqcap$ ($\exists$optionality.OptOut))))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.(Public $\sqcap$ ($\exists$optionality.Always)))))))) $\sqcup$
((($\exists$hasPart.(Current $\sqcap$ ($\exists$hasPart.(Same $\sqcap$ ($\exists$optionality.OptIn))))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Current $\sqcap$ ($\exists$hasPart.(Same $\sqcap$ ($\exists$optionality.(Always $\sqcup$ OptOut)))))))) $\sqcup$
(($\exists$hasPart.(Current $\sqcap$ ($\exists$hasPart.(Same $\sqcap$ ($\exists$optionality.OptOut))))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Current $\sqcap$ ($\exists$hasPart.(Same $\sqcap$ ($\exists$optionality.Always)))))))) $\sqcup$
((($\exists$hasPart.(Current $\sqcap$ ($\exists$hasPart.(Delivery $\sqcap$ ($\exists$optionality.OptIn))))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Current $\sqcap$ ($\exists$hasPart.(Delivery $\sqcap$
($\exists$optionality.(Always $\sqcup$ OptOut)))))))) $\sqcup$
(($\exists$hasPart.(Current $\sqcap$ ($\exists$hasPart.(Delivery $\sqcap$ ($\exists$optionality.OptOut))))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Current $\sqcap$ ($\exists$hasPart.(Delivery $\sqcap$ ($\exists$optionality.Always)))))))) $\sqcup$
((($\exists$hasPart.(Current $\sqcap$ ($\exists$hasPart.(OtherRecipient $\sqcap$ ($\exists$optionality.OptIn))))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Current $\sqcap$ ($\exists$hasPart.(OtherRecipient $\sqcap$
($\exists$optionality.(Always $\sqcup$ OptOut)))))))) $\sqcup$
(($\exists$hasPart.(Current $\sqcap$ ($\exists$hasPart.(OtherRecipient $\sqcap$ ($\exists$optionality.OptOut))))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Current $\sqcap$ ($\exists$hasPart.(OtherRecipient $\sqcap$ ($\exists$optionality.Always)))))))) $\sqcup$
((($\exists$hasPart.(Current $\sqcap$ ($\exists$hasPart.(Unrelated $\sqcap$ ($\exists$optionality.OptIn))))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Current $\sqcap$ ($\exists$hasPart.(Unrelated $\sqcap$
($\exists$optionality.(Always $\sqcup$ OptOut)))))))) $\sqcup$
(($\exists$hasPart.(Current $\sqcap$ ($\exists$hasPart.(Unrelated $\sqcap$ ($\exists$optionality.OptOut))))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Current $\sqcap$ ($\exists$hasPart.(Unrelated $\sqcap$ ($\exists$optionality.Always)))))))) $\sqcup$
((($\exists$hasPart.(Current $\sqcap$ ($\exists$hasPart.(Public $\sqcap$ ($\exists$optionality.OptIn))))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Current $\sqcap$ ($\exists$hasPart.(Public $\sqcap$ ($\exists$optionality.(Always $\sqcup$ OptOut)))))))) $\sqcup$
(($\exists$hasPart.(Current $\sqcap$ ($\exists$hasPart.(Public $\sqcap$ ($\exists$optionality.OptOut))))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Current $\sqcap$ ($\exists$hasPart.(Public $\sqcap$ ($\exists$optionality.Always)))))))) $\sqcup$
((($\exists$hasPart.(Develop $\sqcap$ ($\exists$hasPart.(Same $\sqcap$ ($\exists$optionality.OptIn))))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Develop $\sqcap$ ($\exists$hasPart.(Same $\sqcap$ ($\exists$optionality.(Always $\sqcup$ OptOut)))))))) $\sqcup$
(($\exists$hasPart.(Develop $\sqcap$ ($\exists$hasPart.(Same $\sqcap$ ($\exists$optionality.OptOut))))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Develop $\sqcap$ ($\exists$hasPart.(Same $\sqcap$ ($\exists$optionality.Always)))))))) $\sqcup$
((($\exists$hasPart.(Develop $\sqcap$ ($\exists$hasPart.(Delivery $\sqcap$ ($\exists$optionality.OptIn))))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Develop $\sqcap$ ($\exists$hasPart.(Delivery $\sqcap$
($\exists$optionality.(Always $\sqcup$ OptOut)))))))) $\sqcup$
(($\exists$hasPart.(Develop $\sqcap$ ($\exists$hasPart.(Delivery $\sqcap$ ($\exists$optionality.OptOut))))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Develop $\sqcap$ ($\exists$hasPart.(Delivery $\sqcap$ ($\exists$optionality.Always)))))))) $\sqcup$
((($\exists$hasPart.(Develop $\sqcap$ ($\exists$hasPart.(OtherRecipient $\sqcap$ ($\exists$optionality.OptIn))))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Develop $\sqcap$ ($\exists$hasPart.(OtherRecipient $\sqcap$
($\exists$optionality.(Always $\sqcup$ OptOut)))))))) $\sqcup$
(($\exists$hasPart.(Develop $\sqcap$ ($\exists$hasPart.(OtherRecipient $\sqcap$ ($\exists$optionality.OptOut))))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Develop $\sqcap$ ($\exists$hasPart.(OtherRecipient $\sqcap$ ($\exists$optionality.Always)))))))) $\sqcup$

(((∃hasPart.(Develop ⊓ (∃hasPart.(Unrelated ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Develop ⊓ (∃hasPart.(Unrelated ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(Develop ⊓ (∃hasPart.(Unrelated ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Develop ⊓ (∃hasPart.(Unrelated ⊓ (∃optionality.Always)))))))) ⊔
(((∃hasPart.(Develop ⊓ (∃hasPart.(Public ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Develop ⊓ (∃hasPart.(Public ⊓ (∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(Develop ⊓ (∃hasPart.(Public ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Develop ⊓ (∃hasPart.(Public ⊓ (∃optionality.Always)))))))) ⊔
(((∃hasPart.(Tailoring ⊓ (∃hasPart.(Same ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Tailoring ⊓ (∃hasPart.(Same ⊓ (∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(Tailoring ⊓ (∃hasPart.(Same ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Tailoring ⊓ (∃hasPart.(Same ⊓ (∃optionality.Always)))))))) ⊔
(((∃hasPart.(Tailoring ⊓ (∃hasPart.(Delivery ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Tailoring ⊓ (∃hasPart.(Delivery ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(Tailoring ⊓ (∃hasPart.(Delivery ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Tailoring ⊓ (∃hasPart.(Delivery ⊓ (∃optionality.Always)))))))) ⊔
(((∃hasPart.(Tailoring ⊓ (∃hasPart.(OtherRecipient ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Tailoring ⊓ (∃hasPart.(OtherRecipient ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(Tailoring ⊓ (∃hasPart.(OtherRecipient ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Tailoring ⊓ (∃hasPart.(OtherRecipient ⊓ (∃optionality.Always)))))))) ⊔
(((∃hasPart.(Tailoring ⊓ (∃hasPart.(Unrelated ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Tailoring ⊓ (∃hasPart.(Unrelated ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(Tailoring ⊓ (∃hasPart.(Unrelated ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Tailoring ⊓ (∃hasPart.(Unrelated ⊓ (∃optionality.Always)))))))) ⊔
(((∃hasPart.(Tailoring ⊓ (∃hasPart.(Public ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Tailoring ⊓ (∃hasPart.(Public ⊓ (∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(Tailoring ⊓ (∃hasPart.(Public ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Tailoring ⊓ (∃hasPart.(Public ⊓ (∃optionality.Always)))))))) ⊔
(((∃hasPart.(PseudoAnalysis ⊓ (∃hasPart.(Same ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(PseudoAnalysis ⊓ (∃hasPart.(Same ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(PseudoAnalysis ⊓ (∃hasPart.(Same ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(PseudoAnalysis ⊓ (∃hasPart.(Same ⊓ (∃optionality.Always)))))))) ⊔
(((∃hasPart.(PseudoAnalysis ⊓ (∃hasPart.(Delivery ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(PseudoAnalysis ⊓ (∃hasPart.(Delivery ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(PseudoAnalysis ⊓ (∃hasPart.(Delivery ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(PseudoAnalysis ⊓ (∃hasPart.(Delivery ⊓ (∃optionality.Always)))))))) ⊔
(((∃hasPart.(PseudoAnalysis ⊓ (∃hasPart.(OtherRecipient ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(PseudoAnalysis ⊓ (∃hasPart.(OtherRecipient ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(PseudoAnalysis ⊓ (∃hasPart.(OtherRecipient ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(PseudoAnalysis ⊓ (∃hasPart.(OtherRecipient ⊓
(∃optionality.Always)))))))) ⊔
(((∃hasPart.(PseudoAnalysis ⊓ (∃hasPart.(Unrelated ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(PseudoAnalysis ⊓ (∃hasPart.(Unrelated ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(PseudoAnalysis ⊓ (∃hasPart.(Unrelated ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(PseudoAnalysis ⊓ (∃hasPart.(Unrelated ⊓ (∃optionality.Always)))))))) ⊔
(((∃hasPart.(PseudoAnalysis ⊓ (∃hasPart.(Public ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(PseudoAnalysis ⊓ (∃hasPart.(Public ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(PseudoAnalysis ⊓ (∃hasPart.(Public ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(PseudoAnalysis ⊓ (∃hasPart.(Public ⊓ (∃optionality.Always)))))))) ⊔

(((∃hasPart.(PseudoDecision ⊓ (∃hasPart.(Same ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(PseudoDecision ⊓ (∃hasPart.(Same ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(PseudoDecision ⊓ (∃hasPart.(Same ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(PseudoDecision ⊓ (∃hasPart.(Same ⊓ (∃optionality.Always))))))) ⊔
(((∃hasPart.(PseudoDecision ⊓ (∃hasPart.(Delivery ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(PseudoDecision ⊓ (∃hasPart.(Delivery ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(PseudoDecision ⊓ (∃hasPart.(Delivery ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(PseudoDecision ⊓ (∃hasPart.(Delivery ⊓ (∃optionality.Always))))))) ⊔
(((∃hasPart.(PseudoDecision ⊓ (∃hasPart.(OtherRecipient ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(PseudoDecision ⊓ (∃hasPart.(OtherRecipient ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(PseudoDecision ⊓ (∃hasPart.(OtherRecipient ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(PseudoDecision ⊓ (∃hasPart.(OtherRecipient ⊓
(∃optionality.Always))))))) ⊔
(((∃hasPart.(PseudoDecision ⊓ (∃hasPart.(Unrelated ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(PseudoDecision ⊓ (∃hasPart.(Unrelated ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(PseudoDecision ⊓ (∃hasPart.(Unrelated ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(PseudoDecision ⊓ (∃hasPart.(Unrelated ⊓ (∃optionality.Always))))))) ⊔
(((∃hasPart.(PseudoDecision ⊓ (∃hasPart.(Public ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(PseudoDecision ⊓ (∃hasPart.(Public ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(PseudoDecision ⊓ (∃hasPart.(Public ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(PseudoDecision ⊓ (∃hasPart.(Public ⊓ (∃optionality.Always))))))) ⊔
(((∃hasPart.(IndividualAnalysis ⊓ (∃hasPart.(Same ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(IndividualAnalysis ⊓ (∃hasPart.(Same ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(IndividualAnalysis ⊓ (∃hasPart.(Same ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(IndividualAnalysis ⊓ (∃hasPart.(Same ⊓ (∃optionality.Always))))))) ⊔
(((∃hasPart.(IndividualAnalysis ⊓ (∃hasPart.(Delivery ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(IndividualAnalysis ⊓ (∃hasPart.(Delivery ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(IndividualAnalysis ⊓ (∃hasPart.(Delivery ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(IndividualAnalysis ⊓ (∃hasPart.(Delivery ⊓ (∃optionality.Always))))))) ⊔
(((∃hasPart.(IndividualAnalysis ⊓ (∃hasPart.(OtherRecipient ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(IndividualAnalysis ⊓ (∃hasPart.(OtherRecipient ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(IndividualAnalysis ⊓ (∃hasPart.(OtherRecipient ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(IndividualAnalysis ⊓ (∃hasPart.(OtherRecipient ⊓
(∃optionality.Always))))))) ⊔
(((∃hasPart.(IndividualAnalysis ⊓ (∃hasPart.(Unrelated ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(IndividualAnalysis ⊓ (∃hasPart.(Unrelated ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(IndividualAnalysis ⊓ (∃hasPart.(Unrelated ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(IndividualAnalysis ⊓ (∃hasPart.(Unrelated ⊓
(∃optionality.Always))))))) ⊔
(((∃hasPart.(IndividualAnalysis ⊓ (∃hasPart.(Public ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(IndividualAnalysis ⊓ (∃hasPart.(Public ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(IndividualAnalysis ⊓ (∃hasPart.(Public ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(IndividualAnalysis ⊓ (∃hasPart.(Public ⊓ (∃optionality.Always))))))) ⊔
(((∃hasPart.(IndividualDecision ⊓ (∃hasPart.(Same ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(IndividualDecision ⊓ (∃hasPart.(Same ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(IndividualDecision ⊓ (∃hasPart.(Same ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(IndividualDecision ⊓ (∃hasPart.(Same ⊓ (∃optionality.Always))))))) ⊔

(((∃hasPart.(IndividualDecision ⊓ (∃hasPart.(Delivery ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(IndividualDecision ⊓ (∃hasPart.(Delivery ⊓
(∃optionality.(Always ⊔ OptOut))))))) ⊔
((∃hasPart.(IndividualDecision ⊓ (∃hasPart.(Delivery ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(IndividualDecision ⊓ (∃hasPart.(Delivery ⊓ (∃optionality.Always))))))) ⊔
(((∃hasPart.(IndividualDecision ⊓ (∃hasPart.(OtherRecipient ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(IndividualDecision ⊓ (∃hasPart.(OtherRecipient ⊓
(∃optionality.(Always ⊔ OptOut))))))) ⊔
((∃hasPart.(IndividualDecision ⊓ (∃hasPart.(OtherRecipient ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(IndividualDecision ⊓ (∃hasPart.(OtherRecipient ⊓
(∃optionality.Always))))))) ⊔
(((∃hasPart.(IndividualDecision ⊓ (∃hasPart.(Unrelated ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(IndividualDecision ⊓ (∃hasPart.(Unrelated ⊓
(∃optionality.(Always ⊔ OptOut))))))) ⊔
((∃hasPart.(IndividualDecision ⊓ (∃hasPart.(Unrelated ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(IndividualDecision ⊓ (∃hasPart.(Unrelated ⊓
(∃optionality.Always))))))) ⊔
(((∃hasPart.(IndividualDecision ⊓ (∃hasPart.(Public ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(IndividualDecision ⊓ (∃hasPart.(Public ⊓
(∃optionality.(Always ⊔ OptOut))))))) ⊔
((∃hasPart.(IndividualDecision ⊓ (∃hasPart.(Public ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(IndividualDecision ⊓ (∃hasPart.(Public ⊓ (∃optionality.Always))))))) ⊔
(((∃hasPart.(Contact ⊓ (∃hasPart.(Same ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Contact ⊓ (∃hasPart.(Same ⊓ (∃optionality.(Always ⊔ OptOut))))))) ⊔
((∃hasPart.(Contact ⊓ (∃hasPart.(Same ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Contact ⊓ (∃hasPart.(Same ⊓ (∃optionality.Always))))))) ⊔
(((∃hasPart.(Contact ⊓ (∃hasPart.(Delivery ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Contact ⊓ (∃hasPart.(Delivery ⊓
(∃optionality.(Always ⊔ OptOut))))))) ⊔
((∃hasPart.(Contact ⊓ (∃hasPart.(Delivery ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Contact ⊓ (∃hasPart.(Delivery ⊓ (∃optionality.Always))))))) ⊔
(((∃hasPart.(Contact ⊓ (∃hasPart.(OtherRecipient ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Contact ⊓ (∃hasPart.(OtherRecipient ⊓
(∃optionality.(Always ⊔ OptOut))))))) ⊔
((∃hasPart.(Contact ⊓ (∃hasPart.(OtherRecipient ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Contact ⊓ (∃hasPart.(OtherRecipient ⊓ (∃optionality.Always))))))) ⊔
(((∃hasPart.(Contact ⊓ (∃hasPart.(Unrelated ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Contact ⊓ (∃hasPart.(Unrelated ⊓
(∃optionality.(Always ⊔ OptOut))))))) ⊔
((∃hasPart.(Contact ⊓ (∃hasPart.(Unrelated ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Contact ⊓ (∃hasPart.(Unrelated ⊓ (∃optionality.Always))))))) ⊔
(((∃hasPart.(Contact ⊓ (∃hasPart.(Public ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Contact ⊓ (∃hasPart.(Public ⊓ (∃optionality.(Always ⊔ OptOut))))))) ⊔
((∃hasPart.(Contact ⊓ (∃hasPart.(Public ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Contact ⊓ (∃hasPart.(Public ⊓ (∃optionality.Always))))))) ⊔
(((∃hasPart.(Historical ⊓ (∃hasPart.(Same ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Historical ⊓ (∃hasPart.(Same ⊓ (∃optionality.(Always ⊔ OptOut))))))) ⊔
((∃hasPart.(Historical ⊓ (∃hasPart.(Same ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Historical ⊓ (∃hasPart.(Same ⊓ (∃optionality.Always))))))) ⊔
(((∃hasPart.(Historical ⊓ (∃hasPart.(Delivery ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Historical ⊓ (∃hasPart.(Delivery ⊓
(∃optionality.(Always ⊔ OptOut))))))) ⊔
((∃hasPart.(Historical ⊓ (∃hasPart.(Delivery ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Historical ⊓ (∃hasPart.(Delivery ⊓ (∃optionality.Always))))))) ⊔
(((∃hasPart.(Historical ⊓ (∃hasPart.(OtherRecipient ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Historical ⊓ (∃hasPart.(OtherRecipient ⊓
(∃optionality.(Always ⊔ OptOut))))))) ⊔
((∃hasPart.(Historical ⊓ (∃hasPart.(OtherRecipient ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Historical ⊓ (∃hasPart.(OtherRecipient ⊓ (∃optionality.Always))))))) ⊔

(((∃hasPart.(Historical ⊓ (∃hasPart.(Unrelated ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Historical ⊓ (∃hasPart.(Unrelated ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(Historical ⊓ (∃hasPart.(Unrelated ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Historical ⊓ (∃hasPart.(Unrelated ⊓ (∃optionality.Always))))))) ⊔
(((∃hasPart.(Historical ⊓ (∃hasPart.(Public ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Historical ⊓ (∃hasPart.(Public ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(Historical ⊓ (∃hasPart.(Public ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Historical ⊓ (∃hasPart.(Public ⊓ (∃optionality.Always))))))) ⊔
(((∃hasPart.(Telemarketing ⊓ (∃hasPart.(Same ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Telemarketing ⊓ (∃hasPart.(Same ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(Telemarketing ⊓ (∃hasPart.(Same ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Telemarketing ⊓ (∃hasPart.(Same ⊓ (∃optionality.Always))))))) ⊔
(((∃hasPart.(Telemarketing ⊓ (∃hasPart.(Delivery ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Telemarketing ⊓ (∃hasPart.(Delivery ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(Telemarketing ⊓ (∃hasPart.(Delivery ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Telemarketing ⊓ (∃hasPart.(Delivery ⊓ (∃optionality.Always))))))) ⊔
(((∃hasPart.(Telemarketing ⊓ (∃hasPart.(OtherRecipient ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Telemarketing ⊓ (∃hasPart.(OtherRecipient ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(Telemarketing ⊓ (∃hasPart.(OtherRecipient ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Telemarketing ⊓ (∃hasPart.(OtherRecipient ⊓
(∃optionality.Always))))))) ⊔
(((∃hasPart.(Telemarketing ⊓ (∃hasPart.(Unrelated ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Telemarketing ⊓ (∃hasPart.(Unrelated ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(Telemarketing ⊓ (∃hasPart.(Unrelated ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Telemarketing ⊓ (∃hasPart.(Unrelated ⊓ (∃optionality.Always))))))) ⊔
(((∃hasPart.(Telemarketing ⊓ (∃hasPart.(Public ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Telemarketing ⊓ (∃hasPart.(Public ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(Telemarketing ⊓ (∃hasPart.(Public ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(Telemarketing ⊓ (∃hasPart.(Public ⊓ (∃optionality.Always))))))) ⊔
(((∃hasPart.(OtherPurpose ⊓ (∃hasPart.(Same ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(OtherPurpose ⊓ (∃hasPart.(Same ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(OtherPurpose ⊓ (∃hasPart.(Same ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(OtherPurpose ⊓ (∃hasPart.(Same ⊓ (∃optionality.Always))))))) ⊔
(((∃hasPart.(OtherPurpose ⊓ (∃hasPart.(Delivery ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(OtherPurpose ⊓ (∃hasPart.(Delivery ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(OtherPurpose ⊓ (∃hasPart.(Delivery ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(OtherPurpose ⊓ (∃hasPart.(Delivery ⊓ (∃optionality.Always))))))) ⊔
(((∃hasPart.(OtherPurpose ⊓ (∃hasPart.(OtherRecipient ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(OtherPurpose ⊓ (∃hasPart.(OtherRecipient ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(OtherPurpose ⊓ (∃hasPart.(OtherRecipient ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(OtherPurpose ⊓ (∃hasPart.(OtherRecipient ⊓
(∃optionality.Always))))))) ⊔
(((∃hasPart.(OtherPurpose ⊓ (∃hasPart.(Unrelated ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(OtherPurpose ⊓ (∃hasPart.(Unrelated ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(OtherPurpose ⊓ (∃hasPart.(Unrelated ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(OtherPurpose ⊓ (∃hasPart.(Unrelated ⊓ (∃optionality.Always))))))) ⊔
(((∃hasPart.(OtherPurpose ⊓ (∃hasPart.(Public ⊓ (∃optionality.OptIn))))) ⊓
(∃subDataStructureOf.(∃hasPart.(OtherPurpose ⊓ (∃hasPart.(Public ⊓
(∃optionality.(Always ⊔ OptOut)))))))) ⊔
((∃hasPart.(OtherPurpose ⊓ (∃hasPart.(Public ⊓ (∃optionality.OptOut))))) ⊓
(∃subDataStructureOf.(∃hasPart.(OtherPurpose ⊓ (∃hasPart.(Public ⊓ (∃optionality.Always)))))))))

β_8 InvalidPolicy–DH–Retention $\equiv$
Policy $\sqcap$ ($\exists$collects.(CollectedData $\sqcap$
(((($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.NoRetention))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.(StatedPurpose $\sqcup$ BusinessPractices $\sqcup$
LegalRequirement $\sqcup$ Indefinitely)))))) $\sqcup$
(($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.(StatedPurpose $\sqcup$ BusinessPractices $\sqcup$ LegalRequirement)))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Admin $\sqcap$ ($\exists$hasPart.Indefinitely)))))) $\sqcup$
((($\exists$hasPart.(Current $\sqcap$ ($\exists$hasPart.NoRetention))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Current $\sqcap$ ($\exists$hasPart.(StatedPurpose $\sqcup$ BusinessPractices $\sqcup$
LegalRequirement $\sqcup$ Indefinitely)))))) $\sqcup$
(($\exists$hasPart.(Current $\sqcap$ ($\exists$hasPart.(StatedPurpose $\sqcup$ BusinessPractices $\sqcup$ LegalRequirement)))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Current $\sqcap$ ($\exists$hasPart.Indefinitely)))))) $\sqcup$
((($\exists$hasPart.(Tailoring $\sqcap$ ($\exists$hasPart.NoRetention))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Tailoring $\sqcap$ ($\exists$hasPart.(StatedPurpose $\sqcup$ BusinessPractices $\sqcup$
LegalRequirement $\sqcup$ Indefinitely)))))) $\sqcup$
(($\exists$hasPart.(Tailoring $\sqcap$ ($\exists$hasPart.(StatedPurpose $\sqcup$ BusinessPractices $\sqcup$ LegalRequirement)))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Tailoring $\sqcap$ ($\exists$hasPart.Indefinitely)))))) $\sqcup$
((($\exists$hasPart.(Develop $\sqcap$ ($\exists$hasPart.NoRetention))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Develop $\sqcap$ ($\exists$hasPart.(StatedPurpose $\sqcup$ BusinessPractices $\sqcup$
LegalRequirement $\sqcup$ Indefinitely)))))) $\sqcup$
(($\exists$hasPart.(Develop $\sqcap$ ($\exists$hasPart.(StatedPurpose $\sqcup$ BusinessPractices $\sqcup$ LegalRequirement)))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Develop $\sqcap$ ($\exists$hasPart.Indefinitely)))))) $\sqcup$
((($\exists$hasPart.(PseudoAnalysis $\sqcap$ ($\exists$hasPart.NoRetention))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(PseudoAnalysis $\sqcap$ ($\exists$hasPart.(StatedPurpose $\sqcup$
BusinessPractices $\sqcup$ LegalRequirement $\sqcup$ Indefinitely)))))) $\sqcup$
(($\exists$hasPart.(PseudoAnalysis $\sqcap$ ($\exists$hasPart.(StatedPurpose $\sqcup$ BusinessPractices $\sqcup$
LegalRequirement)))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(PseudoAnalysis $\sqcap$ ($\exists$hasPart.Indefinitely)))))) $\sqcup$
((($\exists$hasPart.(PseudoDecision $\sqcap$ ($\exists$hasPart.NoRetention))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(PseudoDecision $\sqcap$ ($\exists$hasPart.(StatedPurpose $\sqcup$
BusinessPractices $\sqcup$ LegalRequirement $\sqcup$ Indefinitely)))))) $\sqcup$
(($\exists$hasPart.(PseudoDecision $\sqcap$ ($\exists$hasPart.(StatedPurpose $\sqcup$ BusinessPractices $\sqcup$
LegalRequirement)))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(PseudoDecision $\sqcap$ ($\exists$hasPart.Indefinitely)))))) $\sqcup$
((($\exists$hasPart.(IndividualAnalysis $\sqcap$ ($\exists$hasPart.NoRetention))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(IndividualAnalysis $\sqcap$ ($\exists$hasPart.(StatedPurpose $\sqcup$
BusinessPractices $\sqcup$ LegalRequirement $\sqcup$ Indefinitely)))))) $\sqcup$
(($\exists$hasPart.(IndividualAnalysis $\sqcap$ ($\exists$hasPart.(StatedPurpose $\sqcup$ BusinessPractices $\sqcup$
LegalRequirement)))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(IndividualAnalysis $\sqcap$ ($\exists$hasPart.Indefinitely)))))) $\sqcup$
((($\exists$hasPart.(IndividualDecision $\sqcap$ ($\exists$hasPart.NoRetention))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(IndividualDecision $\sqcap$ ($\exists$hasPart.(StatedPurpose $\sqcup$
BusinessPractices $\sqcup$ LegalRequirement $\sqcup$ Indefinitely)))))) $\sqcup$
(($\exists$hasPart.(IndividualDecision $\sqcap$ ($\exists$hasPart.(StatedPurpose $\sqcup$ BusinessPractices $\sqcup$
LegalRequirement)))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(IndividualDecision $\sqcap$ ($\exists$hasPart.Indefinitely)))))) $\sqcup$
((($\exists$hasPart.(Contact $\sqcap$ ($\exists$hasPart.NoRetention))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Contact $\sqcap$ ($\exists$hasPart.(StatedPurpose $\sqcup$ BusinessPractices $\sqcup$
LegalRequirement $\sqcup$ Indefinitely)))))) $\sqcup$
(($\exists$hasPart.(Contact $\sqcap$ ($\exists$hasPart.(StatedPurpose $\sqcup$ BusinessPractices $\sqcup$ LegalRequirement)))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Contact $\sqcap$ ($\exists$hasPart.Indefinitely)))))) $\sqcup$
((($\exists$hasPart.(Historical $\sqcap$ ($\exists$hasPart.NoRetention))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Historical $\sqcap$ ($\exists$hasPart.(StatedPurpose $\sqcup$ BusinessPractices $\sqcup$
LegalRequirement $\sqcup$ Indefinitely)))))) $\sqcup$
(($\exists$hasPart.(Historical $\sqcap$ ($\exists$hasPart.(StatedPurpose $\sqcup$ BusinessPractices $\sqcup$ LegalRequirement)))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Historical $\sqcap$ ($\exists$hasPart.Indefinitely)))))) $\sqcup$
((($\exists$hasPart.(Telemarketing $\sqcap$ ($\exists$hasPart.NoRetention))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Telemarketing $\sqcap$ ($\exists$hasPart.(StatedPurpose $\sqcup$
BusinessPractices $\sqcup$ LegalRequirement $\sqcup$ Indefinitely)))))) $\sqcup$
(($\exists$hasPart.(Telemarketing $\sqcap$ ($\exists$hasPart.(StatedPurpose $\sqcup$ BusinessPractices $\sqcup$
LegalRequirement)))) $\sqcap$
($\exists$subDataStructureOf.($\exists$hasPart.(Telemarketing $\sqcap$ ($\exists$hasPart.Indefinitely)))))

$$
\begin{aligned}
(((&\exists \mathsf{hasPart}.(\mathsf{OtherPurpose} \sqcap (\exists \mathsf{hasPart}.\mathsf{NoRetention}))) \sqcap \\
&(\exists \mathsf{subDataStructureOf}.(\exists \mathsf{hasPart}.(\mathsf{OtherPurpose} \sqcap (\exists \mathsf{hasPart}.(\mathsf{StatedPurpose} \sqcup \\
&\qquad \mathsf{BusinessPractices} \sqcup \mathsf{LegalRequirement} \sqcup \mathsf{Indefinitely}))))) \sqcup \\
&((\exists \mathsf{hasPart}.(\mathsf{OtherPurpose} \sqcap (\exists \mathsf{hasPart}.(\mathsf{StatedPurpose} \sqcup \mathsf{BusinessPractices} \sqcup \\
&\qquad \mathsf{LegalRequirement})))) \sqcap \\
&(\exists \mathsf{subDataStructureOf}.(\exists \mathsf{hasPart}.(\mathsf{OtherPurpose} \sqcap (\exists \mathsf{hasPart}.\mathsf{Indefinitely}))))))))
\end{aligned}
$$

E. XACML Requests and Policies Test Case

Listing E.1: An XACML Request of RestaurantFinder Service on Behalf of Anna for Bob's Location Information

```
<Request>
 <Subject>
  <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:dataowner"
   DataType="http://www.w3.org/2001/XMLSchema#string">
    <AttributeValue>Bob</AttributeValue>
  </Attribute>
  <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
   DataType="http://www.w3.org/2001/XMLSchema#string">
    <AttributeValue>RestaurantFinder</AttributeValue>
  </Attribute>
  <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:on-behalf-of"
   DataType="http://www.w3.org/2001/XMLSchema#string">
    <AttributeValue>Anna</AttributeValue>
  </Attribute>
 </Subject>
 <Resource>
  <ResourceContent>
   <presence entity="pres:bob@example.com">
    <tuple id="eg92n8">
     <status>
      <basic>open</basic>
      <geopriv>
       <location-info>
        <civicAddress>
         <country>US</country>
         <A1>New York</A1>
         <A3>New York</A3>
         <A6>Broadway</A6>
         <HNO>123</HNO>
         <LOC>Suite 75</LOC>
         <PC>10027-0401</PC>
        </civicAddress>
       </location-info>
       <usage-rules>
        <retransmission-allowed>yes</retransmission-allowed>
        <retention-expiry>2013-06-23T04:57:29Z</retention-expiry>
       </usage-rules>
      </geopriv>
     </status>
     <class>email</class>
     <service-class><electronic/></service-class>
     <status-icon>http://example.com/mail.gif</status-icon>
     <contact>mailto:bob@example.com</contact>
    </tuple>
    <person id="p1">
     <activities>
       <note>Far away</note>
       <away/>
     </activities>
     <class>calendar</class>
     <mood><angry/></mood>
     <place-is><audio><noisy/></audio></place-is>
     <place-type><residence/></place-type>
```

```
      <privacy><unknown/></privacy>
      <sphere>bowling league</sphere>
      <status-icon>http://example.com/play.gif</status-icon>
      <time-offset>-240</time-offset>
      <note xml:lang="en">Scoring 120</note>
      <timestamp>2010-02-13T16:09:44+05:00</timestamp>
     </person>
     <device id="pc147">
      <user-input>idle</user-input>
      <deviceID>mac:8asd7d7d70</deviceID>
     </device>
    </presence>
   </ResourceContent>
   <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:document"
    DataType="http://www.w3.org/2001/XMLSchema#string">
     <AttributeValue>PresenceDocument</AttributeValue>
   </Attribute>
   <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
    DataType="http://www.w3.org/2001/XMLSchema#string">
     <AttributeValue>LocationData</AttributeValue>
   </Attribute>
  </Resource>
  <Action>
   <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
    DataType="http://www.w3.org/2001/XMLSchema#string">
     <AttributeValue>read</AttributeValue>
   </Attribute>
  </Action>
</Request>
```

Listing E.2: An XACML Request Based on Multiple Resource Profile from RestaurantFinder Service on Behalf of Anna for Bob's Location Data

```
<Request>
 <Subject>
  <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:dataowner"
   DataType="http://www.w3.org/2001/XMLSchema#string">
    <AttributeValue>Bob</AttributeValue>
  </Attribute>
  <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
   DataType="http://www.w3.org/2001/XMLSchema#string">
    <AttributeValue>RestaurantFinder</AttributeValue>
  </Attribute>
  <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:on-behalf-of"
   DataType="http://www.w3.org/2001/XMLSchema#string">
    <AttributeValue>Anna</AttributeValue>
  </Attribute>
 </Subject>
 <Resource>
  <ResourceContent>
   <presence entity="pres:bob@example.com">
    <tuple id="eg92n8">
     <status>
      <basic>open</basic>
      <geopriv>
       <location-info>
        <civicAddress>
         <country>US</country>
         <A1>New York</A1>
         <A3>New York</A3>
         <A6>Broadway</A6>
         <HNO>123</HNO>
         <LOC>Suite 75</LOC>
         <PC>10027-0401</PC>
        </civicAddress>
```

```
      </location-info>
      <usage-rules>
       <retransmission-allowed>yes</retransmission-allowed>
       <retention-expiry>2013-06-23T04:57:29Z</retention-expiry>
      </usage-rules>
     </geopriv>
    </status>
    <class>email</class>
    <service-class><electronic/></service-class>
    <status-icon>http://example.com/mail.gif</status-icon>
    <contact>mailto:bob@example.com</contact>
   </tuple>
   <person id="p1">
    <activities>
      <note>Far away</note>
      <away/>
    </activities>
    <class>calendar</class>
    <mood><angry/></mood>
    <place-is><audio><noisy/></audio></place-is>
    <place-type><residence/></place-type>
    <privacy><unknown/></privacy>
    <sphere>bowling league</sphere>
    <status-icon>http://example.com/play.gif</status-icon>
    <time-offset>-240</time-offset>
    <note xml:lang="en">Scoring 120</note>
    <timestamp>2010-02-13T16:09:44+05:00</timestamp>
   </person>
   <device id="pc147">
    <user-input>idle</user-input>
    <deviceID>mac:8asd7d7d70</deviceID>
   </device>
  </presence>
 </ResourceContent>
 <Attribute AttributeId="urn:oasis:names:tc:xacml:2.0:resource:profile"
  DataType="http://www.w3.org/2001/XMLSchema#string">
   <AttributeValue>
     urn:oasis:names:tc:xacml:2.0:profile:multiple:scope:xml
   </AttributeValue>
 </Attribute>
 <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:document"
  DataType="http://www.w3.org/2001/XMLSchema#string">
   <AttributeValue>PresenceDocument</AttributeValue>
 </Attribute>
 <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
  DataType="http://www.w3.org/2001/XMLSchema#string">
   <AttributeValue>presence/tuple/status/geopriv</AttributeValue>
 </Attribute>
 <Attribute AttributeId="urn:oasis:names:tc:xacml:2.0:resource:scope"
  DataType="http://www.w3.org/2001/XMLSchema#string">
   <AttributeValue>Descendants</AttributeValue>
 </Attribute>
</Resource>
<Action>
 <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
  DataType="http://www.w3.org/2001/XMLSchema#string">
   <AttributeValue>read</AttributeValue>
 </Attribute>
</Action>
```

Listing E.3: A Simple XACML Policy of Bob Governing His Presence Information Access for RestaurantFinder Service on Behalf of Anna

```
<Policy PolicyId="BobPresenceDocument"
 RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:
```

```
permit-overrides">
 <Target>
  <Subjects>
   <Subject>
    <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
        Bob
     </AttributeValue>
     <SubjectAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:1.0:subject:dataowner"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </SubjectMatch>
    <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
        RestaurantFinder
     </AttributeValue>
     <SubjectAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </SubjectMatch>
    <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
        Anna
     </AttributeValue>
     <SubjectAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:1.0:subject:on-behalf-of"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </SubjectMatch>
   </Subject>
  </Subjects>
  <Resources>
   <Resource>
    <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
        PresenceDocument
     </AttributeValue>
     <ResourceAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:1.0:resource:document"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </ResourceMatch>
   </Resource>
  </Resources>
  <Actions>
   <Action>
    <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
        read
     </AttributeValue>
     <ActionAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </ActionMatch>
   </Action>
  </Actions>
 </Target>
 <Rule RuleId="AllowedPart" Effect="Permit">
  <Target>
   <Subjects><AnySubject/></Subjects>
   <Resources>
    <Resource>
     <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
string-equal">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
         LocationData
      </AttributeValue>
      <ResourceAttributeDesignator
```

```
        AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
        DataType="http://www.w3.org/2001/XMLSchema#string"/>
      </ResourceMatch>
     </Resource>
    </Resources>
    <Actions><AnyAction/></Actions>
   </Target>
  <Rule RuleId="ProhibitedPart" Effect="Deny"/>
  <Obligations>
   <Obligation ObligationId="data-transform" FulfillOn="Permit">
       . . . the same Obligation as Listing 6.1 . . .
   </Obligation>
  </Obligations>
</Policy>
```

Listing E.4: All Allowed Node Paths of Location Data Residing in the Target of the First Rule of XACML Policy for Normal XACML Scheme

```
<Resources>
 <Resource>
  <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
     presence/tuple/status/geopriv
   </AttributeValue>
   <ResourceAttributeDesignator
    AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
    DataType="http://www.w3.org/2001/XMLSchema#string"/>
  </ResourceMatch>
 </Resource>
 <Resource>
  <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
     presence/tuple/status/geopriv/location-info
   </AttributeValue>
   <ResourceAttributeDesignator
    AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
    DataType="http://www.w3.org/2001/XMLSchema#string"/>
  </ResourceMatch>
 </Resource>
 <Resource>
  <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
     presence/tuple/status/geopriv/location-info/civicAddress
   </AttributeValue>
   <ResourceAttributeDesignator
    AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
    DataType="http://www.w3.org/2001/XMLSchema#string"/>
  </ResourceMatch>
 </Resource>
 <Resource>
  <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
     presence/tuple/status/geopriv/location-info/civicAddress/country
   </AttributeValue>
   <ResourceAttributeDesignator
    AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
    DataType="http://www.w3.org/2001/XMLSchema#string"/>
  </ResourceMatch>
 </Resource>
 <Resource>
  <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
     presence/tuple/status/geopriv/location-info/civicAddress/A1
   </AttributeValue>
   <ResourceAttributeDesignator
```

```
    AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
    DataType="http://www.w3.org/2001/XMLSchema#string"/>
  </ResourceMatch>
 </Resource>
 <Resource>
  <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
     presence/tuple/status/geopriv/location-info/civicAddress/A2
   </AttributeValue>
   <ResourceAttributeDesignator
    AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
    DataType="http://www.w3.org/2001/XMLSchema#string"/>
  </ResourceMatch>
 </Resource>
 <Resource>
  <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
     presence/tuple/status/geopriv/location-info/civicAddress/A3
   </AttributeValue>
   <ResourceAttributeDesignator
    AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
    DataType="http://www.w3.org/2001/XMLSchema#string"/>
  </ResourceMatch>
 </Resource>
 <Resource>
  <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
     presence/tuple/status/geopriv/location-info/civicAddress/A4
   </AttributeValue>
   <ResourceAttributeDesignator
    AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
    DataType="http://www.w3.org/2001/XMLSchema#string"/>
  </ResourceMatch>
 </Resource>
 <Resource>
  <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
     presence/tuple/status/geopriv/location-info/civicAddress/A5
   </AttributeValue>
   <ResourceAttributeDesignator
    AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
    DataType="http://www.w3.org/2001/XMLSchema#string"/>
  </ResourceMatch>
 </Resource>
 <Resource>
  <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
     presence/tuple/status/geopriv/location-info/civicAddress/A6
   </AttributeValue>
   <ResourceAttributeDesignator
    AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
    DataType="http://www.w3.org/2001/XMLSchema#string"/>
  </ResourceMatch>
 </Resource>
 <Resource>
  <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
     presence/tuple/status/geopriv/location-info/civicAddress/RD
   </AttributeValue>
   <ResourceAttributeDesignator
    AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
    DataType="http://www.w3.org/2001/XMLSchema#string"/>
  </ResourceMatch>
 </Resource>
 <Resource>
  <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
```

```
    presence/tuple/status/geopriv/location-info/civicAddress/RDBR
   </AttributeValue>
   <ResourceAttributeDesignator
    AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
    DataType="http://www.w3.org/2001/XMLSchema#string"/>
  </ResourceMatch>
 </Resource>
 <Resource>
  <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
    presence/tuple/status/geopriv/location-info/civicAddress/LMK
   </AttributeValue>
   <ResourceAttributeDesignator
    AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
    DataType="http://www.w3.org/2001/XMLSchema#string"/>
  </ResourceMatch>
 </Resource>
 <Resource>
  <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
    presence/tuple/status/geopriv/location-info/civicAddress/STS
   </AttributeValue>
   <ResourceAttributeDesignator
    AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
    DataType="http://www.w3.org/2001/XMLSchema#string"/>
  </ResourceMatch>
 </Resource>
 <Resource>
  <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
    presence/tuple/status/geopriv/location-info/civicAddress/HNO
   </AttributeValue>
   <ResourceAttributeDesignator
    AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
    DataType="http://www.w3.org/2001/XMLSchema#string"/>
  </ResourceMatch>
 </Resource>
 <Resource>
  <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
    presence/tuple/status/geopriv/location-info/civicAddress/PC
   </AttributeValue>
   <ResourceAttributeDesignator
    AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
    DataType="http://www.w3.org/2001/XMLSchema#string"/>
  </ResourceMatch>
 </Resource>
 <Resource>
  <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
    presence/tuple/status/geopriv/usage-rules
   </AttributeValue>
   <ResourceAttributeDesignator
    AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
    DataType="http://www.w3.org/2001/XMLSchema#string"/>
  </ResourceMatch>
 </Resource>
 <Resource>
  <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
    presence/tuple/status/geopriv/usage-rules/retransmission-allowed
   </AttributeValue>
   <ResourceAttributeDesignator
    AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
    DataType="http://www.w3.org/2001/XMLSchema#string"/>
  </ResourceMatch>
 </Resource>
```

```
 <Resource>
  <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
     presence/tuple/status/geopriv/usage-rules/retention-expiry
   </AttributeValue>
   <ResourceAttributeDesignator
    AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
    DataType="http://www.w3.org/2001/XMLSchema#string"/>
  </ResourceMatch>
 </Resource>
</Resources>
```

Listing E.5: Condition Using One XPath Expression

```
     <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:not">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-is-in">
       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
         sad
       </AttributeValue>
       <AttributeSelector
         RequestContextPath="//ResourceContent/presence/person/mood/node()"
         DataType="http://www.w3.org/2001/XMLSchema#string"/>
      </Apply>
     </Condition>
```

Listing E.6: Condition Using Two XPath Expressions

```
    <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:not">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-is-in">
       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
         sad
       </AttributeValue>
       <AttributeSelector
        RequestContextPath="//ResourceContent/presence/person/mood/node()"
        DataType="http://www.w3.org/2001/XMLSchema#string"/>
      </Apply>
     </Apply>
     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:not">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-is-in">
       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
         home
       </AttributeValue>
       <AttributeSelector
        RequestContextPath="//ResourceContent/presence/person/sphere/text()"
        DataType="http://www.w3.org/2001/XMLSchema#string"/>
      </Apply>
     </Apply>
    </Condition>
```

List of Abbreviations

AAS	Authentication and Authorization Support
ABox	Assertion Box
APPEL	A P3P Preference Exchange Language
BPEL	Business Process Execution Language
CS	Composite Service
DL	Description Logic
EC	European Community
EPAL	Enterprise Privacy Authorization Language
HIPAA	Health Insurance Portability and Accountability Act
IETF	Internet Engineering Task Force
MAC Address	Media Access Control Address
OASIS	Organization for the Advancement of Structured Information Standards
OECD	Organisation for Economic Co-operation and Development
OWL	Web Ontology Language
OWL-S	Semantic Markup for Web Services
P3P	Platform for Privacy Preferences
PAP	Policy Administration Point
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PEP-SC	Policy Enforcement Point in Service Component
PIP	Policy Information Point
PRIME	Privacy and Identity Management for Europe
PrimeLife	Privacy and Identity Management in Europe for Life
RFC	Request For Comments
S4P	SecPAL for Privacy
SAC	Service Access Control
SecPAL	Security Policy Authorization Language
SIP	Session Initiation Protocol
SPICE	Service Platform for Innovative Communication Environment
TBox	Terminology Box
TE	Transformer Engine
UDDI	Universal Description, Discovery, and Integration
W3C	World Wide Web Consortium

WSDL	Web Services Description Language
XACML	eXtensible Access Control Markup Language

List of Figures

Listings

List of Tables

Bibliography

[95/95] DIRECTIVE 95/46/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. *Official Journal of the European Communities*, November 1995.

[ABFG04] D. Austin, A. Barbir, C. Ferris, and S. Garg. Web Services Architecture Requirements. *W3C Working Group Note*, February 2004.

[ABV+09] C. A. Ardagna, L. Bussard, S. Vimercati, G. Neven, S. Paraboschi, E. Pedrini, F. Preiss, D. Raggett, P. Samarati, S. Trabelsi, and M. Verdicchio. PrimeLife Policy Language. *Position Paper W3C Workshop on Access Control Application Scenarios*, November 2009.

[ACKM04] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services: Concepts, Architectures and Applications*. Springer, 2004.

[AHK+03] P. Ashley, S. Hada, G. Karjoth, C. Powers, and M. Schunter. Enterprise Privacy Authorization Language (EPAL 1.2). *IBM*, November 2003. http://www.w3.org/Submission/2003/SUBM-EPAL-20031110/.

[AKSX02] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic databases. In *Proceedings of the 28th international conference on Very Large Data Bases*, VLDB '02, pages 143–154. VLDB Endowment, 2002.

[AKSX03a] R. Agrawal, J. Kierman, R. Srikant, and Y. Xu. Implementing P3P Using Database Technology. *In Proc. of the 19th Int'l Conference on Data Engineering*, 2003.

[AKSX03b] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. An XPath-based preference language for P3P. In *Proceedings of the 12th international conference on World Wide Web*, WWW '03, pages 629–639, New York, NY, USA, 2003. ACM.

[And06] A. Anderson. A comparison of two privacy policy languages: EPAL and XACML. In *Proceedings of the 3rd ACM workshop on Secure web services*, SWS '06, pages 53–60, New York, NY, USA, 2006. ACM.

[APP] Minutes of the P3P 2.0 Workshop. Last update November 2003. June 2013 http://www.w3.org/2003/p3p-ws/minutes.html.

[BCC+04] T. Bellwood, S. Capell, L. Clement, J. Colgrave, M. Dovey, D. Feygin, A. Hately, R. Kochman, P. Macias, M. Novotny, M. Paolucci, C. Riegen, T. Rogers, K. Sycara,

P. Wenzel, and Z. Wu. UDDI Version 3.0.2. *UDDI Spec Technical Committee Draft*, October 2004.

[BCFM00] E. Bertino, S. Castano, E. Ferrari, and M. Mesiti. Specifying and enforcing access control policies for XML document sources. *World Wide Web*, 3:139–151, June 2000.

[BCK03] S. Byers, L. Cranor, and D. Kormann. Automated analysis of P3P-enabled Web sites. In *Proceedings of the 5th international conference on Electronic commerce*, ICEC '03, pages 326–338, New York, NY, USA, 2003. ACM.

[BCKM04] S. Byers, L. F. Cranor, D. Kormann, and P. McDaniel. Searching for Privacy: Design and Implementaiton of a P3P-Enabled Search Engine. *In 2004 Workshop on Privacy Enhancing Technologies (PET2004)*, pages 314–328, 2004.

[BCM+07] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, second edition, 2007.

[BDS06] Bundesdatenschutzgesetz (BDSG), Federal Data Protection Act as of 15 November 2006. *The Federal Commissioner for Data Protection and Freedom of Information*, November 2006.

[BFG10] M. Y. Becker, C. Fournet, and A. D. Gordon. SecPAL: Design and Semantics of a Decentralized Authorization Language. *in Journal of Computer Security (JCS)*, pages 597–643, 2010.

[BG04] D. Brickley and R. V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. *World Wide Web Consortium (W3C) Recommendation*, February 2004.

[BHH+04] S. Bechhofer, F. Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider, and L. Stein. OWL Web Ontology Language reference. W3C Recommendation, 10 February 2004. http://www.w3.org/TR/owl-ref/.

[BHS08] F. Baader, I. Horrocks, and U. Sattler. *Handbook of Knowledge Representation*. Elsevier B.V., 2008.

[BMB10] M. Y. Becker, A. Malkis, and L. Bussard. S4P: A Generic Language for Specifying Privacy Preferences and Policies. *MSR-TR-2010-32*, April 2010.

[BNP10] L. Bussard, G. Neven, and F. Preiss. Downstream Usage Control. *IEEE Policy Workshop 2010*, July 2010.

[BPM04] P. V. Biron, K. Permanente, and A. Malhotra. XML Schema Part 2: Datatypes Second Edition. *W3C Recommendation*, October 2004.

[BPT08] B. Berendt, S. Preibusch, and M. Teltzrow. A Privacy-Protecting Business-Analytics Service for On-Line Transactions. *Int. J. Electron. Commerce*, 12(3):115–150, April 2008.

[BRDM07] P. Beatty, I. Reay, S. Dick, and J. Miller. P3P Adoption on E-Commerce Web sites: A Survey and Analysis. *IEEE Internet Computing*, 11(2):65–71, March 2007.

[CBK03] L. Cranor, S. Byers, and D. Kormann. An Analysis of P3P Deployment on Commercial, Government, and Children's Web Sites as of May 2003. *Technical Report, Federal Trade Commission*, May 2003.

[CCLF07] S. Chang, A. Chebotko, S. Lu, and F. Fotouhi. Graph Matching Based Authorization Model for Efficient Secure XML Querying. In *Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops - Volume 02*, AINAW '07, pages 473–478, Washington, DC, USA, 2007. IEEE Computer Society.

[CDE+06] L. Cranor, B. Dobbs, S. Egelman, G. Hogben, J. Hamphrey, M. Langheinrich, M. Marchiori, M. Presler-Marshall, J. Reagle, M. Schunter, D. A. Stampley, and R. Wenning. *The Platform for Privacy Preference 1.1 (P3P1.1) Specification*. W3C Working Group Note 13 November 2006, November 2006.

[CDL+04] S. Cox, P. Daisy, R. Lake, C. Portele, and A. Whiteside. OpenGIS Geography Markup Language (GML) Implementation Specification. *OGC 03-105r1, version: 3.1.1*, February 2004.

[CES+08] L. Cranor, S. Egelman, S. Sheng, A. McDonald, and A. Chowdhury. P3P deployment on websites. *Electron. Commer. Rec. Appl.*, 7(3):274–293, November 2008.

[CHM+07] R. Chinnici, H. Haas, J. Moreau, D. Orchard, and S. Weerawarana. Web Services Description Language (WSDL) Version 2.0 Part 2: Adjuncts. *W3C Recommendation*, June 2007.

[Cla98] R. Clarke. Platform for Privacy Preferences: A Critique. July 1998. http://www.rogerclarke.com/DV/P3PCrit.html.

[CLM02a] L. Cranor, M. Langheinrich, and M. Marchiori. A P3P Preference Exchange Language 1.0 (APPEL 1.0). *W3C Working Draft*, April 2002.

[CLM+02b] L. Cranor, M. Langheinrich, M. Marchiori, M. Presler-Marshall, and J. Reagle. *The Platform for Privacy Preference 1.0 (P3P1.0) Specification*. W3C Recommendation, April 2002.

[CMN09] J. Camenisch, S. Mödersheim, and G. Neven. Credential-Based Access Control Extensions to XACML. *Position Paper W3C Workshop on Access Control Application Scenarios*, November 2009.

[CMRW07] R. Chinnici, J. Moreau, A. Ryman, and S. Weerawarana. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. *W3C Recommendation*, June 2007.

[Com08] OASIS XACML Technical Committee. XACML References and Products, Version 1.84. February 2008. http://www.oasis-open.org/committees/download.php/27298/xacmlRefs-V1-84-1.htm#Products.

[Coy99] K. Coyle. P3P: Pretty Poor Privacy? A Social Analysis of the Platform for Privacy Preferences (P3P). June 1999. http://www.kcoyle.net/p3p.html.

[Coy00] K. Coyle. A Response to "P3P and Privacy: An Update for the Privacy Community" by the Center for Democracy and Technology. May 2000. http://www.kcoyle.net/response.html.

[Cra03] L. Cranor. P3P 1.1 User Agent Guidelines. *P3P User Agent Task Force Report 23*, May 2003.

[DDFP04] E. Damiani, S. De Capitani di Vimercati, C. Fugazza, and P.Samarati. Semantics-aware Privacy and Access Control: Motivation and Preliminary Results. In *1st Italian Semantic Web Workshop*, Ancona, Italy, December 2004.

[Dep02] Department of Health and Human Services. 45 CFR Parts 160 and 164 Standards for Privacy of Individually Identifiable Health Information; Final Rule. *Office of the Secretary*, August 2002.

[Dir95] DIRECTIVE 95/46/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. *Official Journal of the European Communities*, November 1995.

[Dir02] DIRECTIVE 2002/58/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector (Directive on privacy and electronic communications). *Official Journal of the European Communities*, July 2002.

[Dir06] DIRECTIVE 2006/24/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 15 March 2006 on the retention of data generated or processed in connection with the provision of publicly available electronic communications services or of public communications networks and amending Directive 2002/58/EC. *Official Journal of the European Communities*, April 2006.

[DPA98] Data Protection Act 1998. *Office of Public Sector Information*, 1998.

[DRS00] M. Day, J. Rosenberg, and H. Sugano. A Model for Presence and Instant Messaging. *RFC 2778*, February 2000.

[DVPS00] E. Damiani, S. Vimercati, S. Paraboschi, and P. Samarati. Securing XML Documents. *Lecture Notes in Computer Science*, 1777, 2000.

[EJ00] EPIC and Junkbusters. Pretty Poor Privacy: An Assessment of P3P and Internet Privacy. June 2000. http://epic.org/reports/prettypoorprivacy.html.

[Erl06] T. Erl. *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice-Hall, 2006.

[ESO] The Enterprise Sign On Engine (ESOE), Last update June 2013. http://esoeproject.org/.

[FaC] FaCT++. June 2013 http://code.google.com/p/factplusplus/.

[FCG04] W. Fan, C. Chan, and M. Garofalakis. Secure XML querying with security views. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, SIGMOD '04, pages 587–598, New York, NY, USA, 2004. ACM.

[GHM+07a] M. Gudgin, M. Hadley, N. Mendelsohn, J. Moreau, H. Nielsen, A. Karmarkar, and Y. Lafon. SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). *W3C Recommendation*, April 2007.

[GHM+07b] M. Gudgin, M. Hadley, N. Mendelsohn, J. Moreau, H. Nielsen, A. Karmarkar, and Y. Lafon. SOAP Version 1.2 Part 2: Adjuncts (Second Edition). *W3C Recommendation*, April 2007.

[GHM+08] B. Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, and U. Sattler. OWL 2: The next step for OWL. *J. of Web Semantics*, 6(4):309–322, November 2008.

[GR00] R. Grimm and A. Rossnagel. P3P and the Privacy Legislation in Germany: Can P3P help to protect privacy worldwide? August 2000.

[Gro99] W3C Working Group. Removing Data Transfer from P3P. September 1999. http://www.w3.org/P3P/data-transfer.html.

[HERa] HERAS. Last update January 2013. June 2013 http://www.herasaf.org/.

[Herb] HermiT. June 2013 http://www.hermit-reasoner.com.

[Her10] J. Herrmann. Access control systems for spatial data infrastructures and their administration. In *Proceedings of the 1st International Conference and Exhibition on Computing for Geospatial Research and Application*, COM.Geo '10, pages 47:1–47:2, New York, NY, USA, 2010. ACM.

[Her11] J. Herrmann. Administration of (Geo)XACML policies for spatial data infrastructures. In *Proceedings of the 4th ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS*, SPRINGL '11, pages 53–59, New York, NY, USA, 2011. ACM.

[HFE+09] G. Hsieh, K. Foster, G. Emamali, G. Patrick, and L. Marvel. Using XACML for Embedded and Fine-Grained Access Control Policy. In *Availability, Reliability and Security, 2009. ARES '09. International Conference on*, pages 462–468, March 2009.

[Hog02] G. Hogben. A Technical Analysis of Problems with P3P v1.0 and Possible Solutions. *Position Paper for "Future of P3P" workshop*, November 2002. http://www.w3.org/2002/p3p-ws/pp/jrc.html.

[Hog03] G. Hogben. Suggestions for Long Term Changes to P3P. *Long Term Future of P3P workshop*, June 2003.

[Hog04] G. Hogben. P3P Using the Semantic Web (Web Ontology, RDF Policy and RDQL Rules). *W3C Working Group Note 3 September 2004*, September 2004. http://www.w3.org/P3P/2004/040920_p3p-sw.html.

[Hog05] G. Hogben. Describing the P3P base data schema using OWL. In *WWW2005, Workshop on Policy Management for the Web*, 2005.

[JSJP07] C. Jensen, C. Sarkar, C. Jensen, and C. Potts. Tracking website data-collection and privacy practices with the iWatch web crawler. In *Proceedings of the 3rd symposium on Usable privacy and security*, SOUPS '07, pages 29–40, New York, NY, USA, 2007. ACM.

[KA08] A. Khurat and J. Abendroth. A Mechanism for Requesting Hierarchical documetns in XACML. In *Proceedings of the 2008 IEEE International Conference on Wireless & Mobile Computing, Networking & Communication*, WIMOB '08, pages 202–207, Washington, DC, USA, 2008. IEEE Computer Society.

[KC04] G. Klyne and J. J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. *World Wide Web Consortium (W3C) Recommendation*, February 2004.

[KGA10] A. Khurat, D. Gollmann, and J. Abendroth. A formal P3P semantics for composite services. In *Proceedings of the 7th VLDB conference on Secure data management*, SDM'10, pages 113–131, Berlin, Heidelberg, 2010. Springer-Verlag.

[KH00] M. Kudo and S. Hada. XML document security based on provisional authorization. In *Proceedings of the 7th ACM conference on Computer and communications security*, CCS '00, pages 87–96, New York, NY, USA, 2000. ACM.

[KS11] A. Khurat and B. Suntisrivaraporn. An Ontological Approach to Verifying P3P Policies. In Joaquim Filipe and Jan L. G. Dietz, editors, *KEOD*, pages 349–353. SciTePress, 2011.

[KSHW03] G. Karjoth, M. Schunter, E. V. Herreweghen, and M. Waidner. Amending P3P for Clearer Privacy Promises. *In Proceedings of the 14th international Workshop on Database and Expert Systems Applications, IEEE Computer Society*, September 2003.

[law98] Opinion 1/1998 Platform for Privacy Preferences (P3P) and the Open Profiling Standard(OPS). *Working Party on the Protection of Individuals with Regard to the Processing of Personal Data*, June 1998.

[LB05] Y. H. Li and S. Benbernou. Representing and Reasoning About Privacy Abstractions. In *WISE*, pages 390–403, 2005.

[LHSL07] X. Liu, Y. Hui, W. Sun, and H. Liang. Towards Service Composition Based on Mashup. *Services, 2007 IEEE Congress on*, pages 332–339, July 2007.

[LLLL04] B. Luo, D. Lee, W. Lee, and P. Liu. QFilter: Fine-Grained Run-Time XML Access Control via NFA-based Query Rewriting. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, CIKM '04, pages 543–552, New York, NY, USA, 2004. ACM.

[LS98] K. Lee and G. Speyer. White Paper: Platform for Privacy Preferences Project (P3P) & Citibank. *Citibank Advanced Development Group*, October 1998. http://dollar.ecom.cmu.edu/p3pcritique/.

[LY09] D. Lee and T. Yu. XML Access Control. In *Encyclopedia of Database Systems*, pages 3573–3576. 2009.

[LYA03] N. Li, T. Yu, and Antón. A Semantics-Based Approach to Privacy Languages. *Technical Report TR2003-28, CERIAS*, November 2003.

[MB06] E. A. Marks and M. Bell. *Service-Oriented Architecture: A Planning and Implementation Guide for Business and Technology*. Wiley, 2006.

[MPSP09] B. Motik, P. F. Patel-Schneider, and B. Parsia. OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax. *W3C Recommendation*, October 2009.

[MTKH03] M. Murata, A. Tozawa, M. Kudo, and S. Hada. XML Access Control Using Static Analysis. In *Proceedings of the 10th ACM conference on Computer and communications security*, CCS '03, pages 73–84, New York, NY, USA, 2003. ACM.

[Nes08] T. Nestler. Towards a mashup-driven end-user programming of SOA-based applications. *iiWAS '08: Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services*, pages 551–554, 2008.

[New02] E. Newcomer. *Understanding web services: XML, WSDL, SOAP, and UDDI*. Addison-Wesley, 2002.

[NFPS09] T. Nestler, M. Feldmann, A. Preußner, and A. Schill. Service Composition at the Presentation Layer using Web Service Annotations. *Proceedings of the First International Workshop on Lightweight Integration on the Web, ICWE*, 2009.

[OAS05a] OASIS. Core and hierarchical role based access control (RBAC) profile of XACML v2.0. *OASIS Standard*, February 2005.

[OAS05b] OASIS. eXtensible Access Control Markup Language (XACML) Version 2.0. *OASIS Standard*, February 2005.

[OAS05c] OASIS. Hierarchical resource profile of XACML v2.0. *OASIS Standard*, February 2005.

[OAS05d] OASIS. Multiple resource profile of XACML v2.0. *OASIS Standard*, February 2005.

[OAS05e] OASIS. SAML 2.0 profile of XACML v2.0. *OASIS Standard*, February 2005.

[OAS07] OASIS. Web Services Business Process Execution Language Version 2.0. *OASIS Standard*, April 2007.

[OEC80] OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data. *Organisation for Economic Co-operation and Development (OECD)*, September 1980.

[O'R05] T. O'Reilly. What is web 2.0: Design patterns and business models for the next generation of software. 2005. http://oreilly.com/web2/archive/what-is-web-20.html.

[owl] The OWL API. June 2013 http://owlapi.sourceforge.net/.

[P3Pa] P3P Builder. June 2013 http://www.p3pbuilder.com/.

[P3Pb] IBM P3P Policy Editor. http://www.alphaworks.ibm.com/tech/p3peditor.

[P3Pc] P3P Implementation Guide. June 2013 http://p3ptoolbox.org/guide/section2.shtml.

[P3Pd] P3PToolbox Introduction. June 2013 http://p3ptoolbox.org/guide/introduction.shtml.

[P3Pe] JRC Policy Workbench. Last update April 2013. June 2013 http://sourceforge.net/projects/jrc-policy-api/.

[Par10] PrimeLife Partners. Second research report on next generation policies. *PrimeLife Deliverable D5.2.2*, February 2010.

[Pel] Pellet. June 2013 http://clarkparsia.com/pellet.

[Pet05] J. Peterson. A Presence-based GEOPRIV Location Object Format. *RFC 4119*, December 2005.

[Pre06] S. Preibusch. Privacy Negotiations with P3P. *W3C Workshop on Languages for Privacy Policy Negotiation and Semantics-Driven Enforcement*, October 2006.

[PRIa] Privacy and Identity Management for Europe (PRIME). June 2013 https://www.prime-project.eu.

[Prib] Privacy and Identity Management in Europe for Life (PrimeLife). June 2013 http://www.primelife.eu.

[Pric] Privacy Bird. June 2013 http://www.privacybird.org/.

[Prid] Privacy Finder. http://www.privacyfinder.org/.

[Pri74] THE PRIVACY ACT OF 1974. December 1974.

[Pri86] ELECTRONIC COMMUNICATIONS PRIVACY ACT OF 1986. October 1986.

[Pri00] ELECTRONIC COMMUNICATIONS PRIVACY ACT OF 2000, H.R.5018. July 2000.

[Rac] RacerPro. Last update May 2013. June 2013 http://www.racer-systems.com/.

[RAM] Research Activity flow and Middleware Priorities (RAMP). December 2007 http://www.ramp.org.au.

[RBDM07] I. Reay, P. Beatty, S. Dick, and J. Miller. A Survey and Analysis of the P3P Protocol's Agents, Adoption, Maintenance, and Future. *IEEE Trans. Dependable Secur. Comput.*, 4(2):151–164, April 2007.

[Ros06] J. Rosenberg. A Data Model for Presence. *RFC 4479*, July 2006.

[Ros07] J. Rosenberg. Presence Authorization Rules. *RFC 5025*, December 2007.

[RSR10] A. Rota, S. Short, and M. Rahaman. XML secure views using semantic access control. In *Proceedings of the 2010 EDBT/ICDT Workshops*, EDBT '10, pages 5:1–5:10, New York, NY, USA, 2010. ACM.

[SF02] A. Stoica and C. Farkas. Secure XML Views. In *DBSec*, pages 133–146, 2002.

[SFK+04] H. Sugano, S. Fujimoto, G. Klyne, A. Bateman, W. Carr, and J. Peterson. Presence Information Data Format (PIDF). *RFC 3863*, August 2004.

[SHW02] M. Schunter, E. V. Herreweghen, and M. Waidner. Expressive Privacy Promises-How to Improve the Platform for Privacy Preferences (P3P). *Position paper for W3C Workshop on the Future of P3P*, September 2002.

[SMH08] R. Shearer, B. Motik, and I. Horrocks. HermiT: A Highly-Efficient OWL Reasoner. *5th OWL Experienced and Directions Workshop*, 2008.

[SPI] Service Platform for Innovative Communication Environment (SPICE). September 2008 http://www.ist-spice.org.

[SS09] S. Staab and R. Studer. *Handbook on Ontologies*. Springer, 2nd edition, 2009.

[STC+11] H. Schulzrinne, H. Tschofenig, J. Cuellar, J. Polk, J. Morris, and M. Thomson. Geolocation Policy: A Document Format for Expressing Privacy Preferences for Location Information, draft-ietf-geopriv-policy-25. *IETF Draft, Work in Progress*, October 2011.

[STM+07] H. Schulzrinne, H. Tschofenig, J. Morris, J. Cuellar, J. Polk, and J. Rosenberg. Common Policy: A Document Format for Expressing Privacy Preferences. *RFC 4745*, February 2007.

[Sun] Sun's XACML Implementation. Last update June 2006. June 2013 http://sunxacml.sourceforge.net.

[Thi00] R. Thibadeau. A Critique of P3P: Privacy on the Web. August 2000. http://dollar.ecom.cmu.edu/p3pcritique/.

[TKG04] Telecommunications Act (TKG) of 22 June 2004. *The Federal Commissioner for Data Protection and Freedom of Information*, June 2004.

[TMG07] Telemediengesetz(TMG). *The Federal Commissioner for Data Protection and Freedom of Information*, February 2007.

[TNBN10] S. Trabelsi, A. Njeh, L. Bussard, and G. Neven. PPL Engine: A Symmetric Architecture for Privacy Policy Handling. *Position Paper W3C Workshop on Privacy and Data Usage Control*, October 2010.

[TW08] M. Thomson and J. Winterbottom. Revised Civic Location Format for Presence Information Data Format Location Object (PIDF-LO). *RFC 5139*, February 2008.

[VOH+07] A. Vedamuthu, D. Orchard, F. Hirsch, M. Hondo, P. Yendluri, T. Boubez, and Ü. Yalçinalp. Web Services Policy 1.5 - Framework. *W3C Recommendation*, September 2007.

[W3C05] W3C. Web Services Choreography Description Language Version 1.0. *W3C Candidate Recommendation*, November 2005.

[Whi09] J. Whittaker. Producing for web 2.0: A student guide. *Routledge*, 2009.

[WTT09] J. Winterbottom, M. Thomson, and H. Tschofenig. GEOPRIV Presence Information Data Format Location Object (PIDF-LO) Usage Clarification, Considertions, and Recommendations. *RFC 5491*, March 2009.

[YLA04] T. Yu, N. Li, and A. Antón. A Formal Semantics for P3P. In *ACM Workshop on Secure Web Services*, October 2004.

[YSLJ02] T. Yu, D. Srivastava, L. Lakshmanan, and H. Jagadish. Compressed Accessibility Map: Efficient Access Control for XML. In *Proceedings of the 28th international conference on Very Large Data Bases*, VLDB '02, pages 478–489. VLDB Endowment, 2002.

[ZZSZ05] H. Zhang, N. Zhang, Kenneth Salem, and D. Zhuo. Compact Access Control Labeling for Efficient Secure XML Query Evaluation. *International Conference on Data Engineering Workshops*, April 2005.

Publications

- A. Khurat and B. Suntisrivaraporn, An Ontological Approach to Verifying P3P Policies, International Conference on Knowledge Engineering and Ontology Development, KEOD 2011.
- A. Khurat, D. Gollmann, and J. Abendroth, A Formal P3P Semantics for Composite Services, SDM 2010, Springer-Verlag, LNCS 6358, pp. 113-131, 2010.
- A. Khurat and J. Abendroth, A Mechanism for Requesting Hierarchical Documents in XACML, 4th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, October 2008.