Formal Analysis of Privacy Requirements Specifications for Multi-Tier Applications

Travis D. Breaux **Ashwini Rao**

Jan 28, 2013

CMU-ISR-13-101

Institute for Software Research
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, 15213

Abstract

Companies require data from multiple sources to develop new information systems, such as social networking, e-commerce and location-based services. Systems rely on complex, multi-stakeholder data supply-chains to deliver value. These data supply-chains have complex privacy requirements: privacy policies affecting multiple stakeholders (e.g. user, developer, company, government) regulate the collection, use and sharing of data over multiple jurisdictions (e.g. California, United States, Europe). Increasingly, regulators expect companies to ensure consistency between company privacy policies and company data practices. To address this problem, we propose a methodology to map policy requirements in natural language to a formal representation in Description Logic. Using the formal representation, we reason about conflicting requirements within a single policy and among multiple policies in a data supply chain. Further, we enable tracing data flows within the supply-chain. We derive our methodology from an exploratory case study of Facebook platform policy. We demonstrate the feasibility of our approach in an evaluation involving Facebook, Zynga and AOL-Advertising policies. Our results identify three conflicts that exist between Facebook and Zynga policies, and one conflict within the AOL Advertising policy.

Authors thank Dave Gordon, Hanan Hibshi and Darya Kurilova for their early feedback, and the Requirements Engineering Lab at Carnegie Mellon University.

Keywords: privacy; requirements; standardization; description logic; formal analysis

1 Introduction

Increasingly, web and mobile information systems are leveraging user data collected from multiple sources without a clear understanding of data provenance or the privacy requirements that should follow this data. These emerging systems are based on multi-tier platforms in which the "tiers" may be owned and operated by different parties, such as cellular and wireless network providers, mobile and desktop operating system manufacturers, and mobile or web application developers. In addition, user services developed on these tiers are abstracted into platforms to be extensible by other developers, such as Google Maps, Facebook and LinkedIn. Application marketplaces, such as Amazon Appstore, Google Play and iTunes, have emerged to provide small developers increased access to customers, thus lowering the barrier to entry and increasing the risk of misusing personal information by inexperienced developers or small companies. Thus, platform and application developers bear increased, shared responsibility to protect user data as they integrate into these multi-tier ecosystems.

In Canada, Europe and the United States, privacy policies have served as contracts between users and their service providers and, in the U.S., these policies are often the sole means to enforce accountability [9]. In particular, Google has been found to re-purpose user data across their services in ways that violated earlier versions of their privacy policy [11], and Facebook's third-party apps were found to transfer Facebook user data to advertisers in violation of Facebook's Platform Policies [20]. The challenge for these companies is ensuring that software developer intentions at different tiers are consistent with privacy requirements across the entire ecosystem. To this end, we conducted a case study to formalize a subset of privacy-relevant requirements from these policies. We believe such formalism could be used to verify that privacy requirements are consistent across this ecosystem: "app" developers could express their intentions, formally, and then check whether these intentions conflict with the requirements of third parties. Furthermore, platform developers could verify that their platform policy requirements are consistent with app developer requirements.

Contributions: Our main contributions are as follows: (1) we systematically identify a subset of privacy-relevant requirements from privacy policies using a case study method; (2) we formalize data requirements subset in a privacy requirements specification language expressed using Description Logic (DL); the language supports modeling actors, data and data use purpose hierarchies within data requirements; (3) we model requirements conflict checking using DL concept satisfiability, while ensuring decidability and computational bounds; and (4) we model tracing of data flows within a privacy policy.

The remainder of the paper is organized as follows: in Section 2, we introduce a running example based on our case study; in Section 3, we introduce our formal language that we derived from our exploratory case study; in Section 4, we report our method for deriving the language; in Section 5, we report our extended case study findings to evaluate the language across three privacy-related policies; in Section 6, we consider threats to validity; in Section 7, we review related work; and in Section 8, we conclude with discussion and summary.

2 Running Example

We illustrate the problem and motivate our approach using a running example: in Figure 1, we present privacy policy excerpts from the Facebook Platform Policy that governs Zynga, the company that produces the depicted Farmville game. The colored arrows trace from the visual elements that the user sees in their web browser on the right-hand side to governing policy excerpts on the left-hand side. The black dotted lines along the left-hand side show how data flows across these application layers. Zynga has a third-party relationship with Advertising.com, a subsidiary of AOL Advertising that serves the online ad, "Buying Razors Sucks" in this game. Zynga also produces a version of this game for the Android and iPhone mobile devices, which would be available through the Google Play and iTunes marketplaces, which have their own platform developer policies that are not depicted, here.

Figure 1. Privacy policy excerpts and data flows mapped to web content that the user sees in their browser

As the platform provider, Facebook manages basic user account information, including user IDs, friend lists, and other data that may be made available to Zynga under the Facebook's platform policy. The Facebook policy excerpt in Figure 1 prohibits the developer (Zynga) from transferring any data to advertisers, regardless of whether users consent to the transfer. Zynga's privacy policy also prohibits such transfers, unless the user consents (an apparent conflict). Furthermore, AOL Advertising (the advertiser) retains the right to use collected information to better target advertising to users across multiple platforms, for which Farmville is just one example. Because this ad is placed by Zynga, AOL Advertising is a third-party advertiser and Facebook expects Zynga to ensure that AOL adheres to the rules governing access to Facebook's user data. At the time of this writing, Farmville was the top Facebook App with over 41.8 million active users per month and Facebook reports over 9 million apps exist for their platform, in general. Thus, this simple scenario has many potential variations.

In Figure 2, we illustrate a data supply chain between a user, Facebook, Zynga and AOL. The arrows denote data flows among the four actors, and the policies regulate these flows. Under the Facebook privacy policy, Facebook is permitted to collect and use the user's age and gender. Facebook may transfer that information to its developers' apps, such as Farmville developed by Zynga. However, the Facebook platform policy prohibits Zynga from transfering any Facebook user information, including aggregate data, to an advertiser, such as AOL. For a user, it is clear that she has privacy policy agreements with Facebook and Zynga, because these are first-party services. However, it's unlikely the user is aware of AOL's privacy agreement or that data flows to AOL. To identify the advertiser supplying the ad in Figure 1, "buying razors sucks," we had to collect TCP/IP network traffic using a traffic analyzer (Wireshark). The network traffic revealed the domain r1.ace.advertising.com as the server serving the ad into Farmville. Upon visiting the r1.ace.advertising.com website, the link to their privacy policy at http://www.advertising.com/privacy_policy.php contains an error message. Scrolling to the bottom of the webpage, the user can then click a "privacy" hyperlink to visit AOL's privacy policy that describes Advertising.com's privacy practices at http://advertising.aol.com/privacy.

This example illustrates how different parties reuse content from other parties to build more complex systems, and how developers need tools to ensure consistency between privacy requirements across different parties. However, at present, policies expressed in natural language remain disconnected and hence software can freely deviate from the coordination required and expected across these different parties. To address this problem we propose to develop a formal language as an interlingua to describe requirements that map natural language policy to formal statements that can eventually be traced to software.

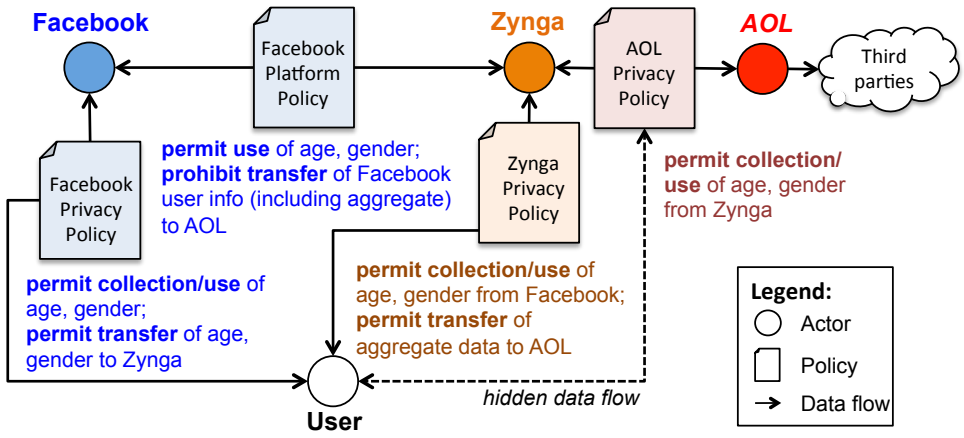

Figure 2. Example data supply chain through Facebook, Zynga and AOL Advertising

3 Approach

We aim to improve privacy by introducing a privacy requirements specification that serves to align multi-party expectations across multi-tier applications. This specification would express a critical subset of policy statements in a formalism that we can check for requirements conflicts. This includes conflicts within a party's specification, and conflicts between two or more specifications of different parties. We base our approach on semantic parameterization, wherein natural language requirements phrases are mapped to actions and roles in Description Logic (DL) [8]. This format was validated using 100 privacy policy goals [6] and over 300 data requirements governing health information [7]. We now introduce DL, followed by our precise definition of the privacy requirements specification.

3.1. Introduction to Description Logic

Description Logic (DL) is a subset of first-order logic for expressing knowledge. A DL knowledge base *KB* is comprised of intensional knowledge, which consists of concepts and roles (terminology) in the TBox, and extensional knowledge, which consists of properties, objects and individuals (assertions) in the ABox [4]. In this paper, we use the DL family $\mathcal{ALC}$, which includes logical constructors for union, intersection, negation, and full existential qualifiers over roles. The reasoning tasks of concept satisfiability, concept subsumption and ABox consistency in $\mathcal{ALC}$ are PSPACE-complete [4].

Reasoning in DL begins with an interpretation $\mathfrak{I}$ that consists of a nonempty set $\Delta^{\mathfrak{I}}$, called the *domain of interpretation*, and the interpretation function $\cdot^{\mathfrak{I}}$ that maps concepts and roles to subsets as follows: every atomic concept C is assigned a subset $C^{\mathfrak{I}} \subseteq \Delta^{\mathfrak{I}}$ and every role R is assigned the subset $R^{\mathfrak{I}} \subseteq \Delta^{\mathfrak{I}} \times \Delta^{\mathfrak{I}}$. For each $(a, b) \in R^{\mathfrak{I}}$, b is called the filler. Description Logic defines two special concepts: $\top$ (top) with the interpretation $\top^{\mathfrak{I}} = \Delta^{\mathfrak{I}}$ and $\bot$ (bottom) with interpretation $\bot^{\mathfrak{I}} = \emptyset$. In addition to constructors for union, intersection and negation, DL provides a constructor to constrain role values, written *R.C*, which means the filler for the role *R* belongs to the concept *C*. The interpretation function is extended to concept definitions in the DL family $\mathcal{ALC}$ as follows, where *C* and *D* are concepts, *R* is a role in the TBox and *a* and *b* are individuals in the ABox:

$$(\neg C)^{\mathfrak{I}} = \Delta^{\mathfrak{I}} \setminus C^{\mathfrak{I}}$$
$$(C \sqcap D)^{\mathfrak{I}} = C^{\mathfrak{I}} \cap D^{\mathfrak{I}}$$
$$(C \sqcup D)^{\mathfrak{I}} = C^{\mathfrak{I}} \cup D^{\mathfrak{I}}$$
$$(\forall R.C)^{\mathfrak{I}} = \{a \in \Delta^{\mathfrak{I}} \mid \forall b.(a, b) \in R^{\mathfrak{I}} \rightarrow b \in C^{\mathfrak{I}}\}$$
$$(\exists R.C)^{\mathfrak{I}} = \{a \in \Delta^{\mathfrak{I}} \mid \exists b.(a, b) \in R^{\mathfrak{I}} \wedge b \in C^{\mathfrak{I}}\}$$

Description Logic includes axioms for subsumption, disjointness and equivalence with respect to a TBox. Subsumption is used to describe individuals using generalities, and we say a concept *C* subsumes a concept *D*, written $T \vDash D \sqsubseteq C$, if $D^{\mathfrak{I}} \subseteq C^{\mathfrak{I}}$ for all interpretations $\mathfrak{I}$ that satisfy the TBox *T*. The concept *C* is disjoint from a concept *D*, written $T \vDash D \sqcap$

$C \rightarrow \perp$, if $D^{\mathfrak{I}} \cap C^{\mathfrak{I}} = \emptyset$ for all interpretations $\mathfrak{I}$ that satisfy the TBox T. The concept C is equivalent to a concept D, written $T \vDash C \equiv D$, if $C^{\mathfrak{I}} = D^{\mathfrak{I}}$ for all interpretations $\mathfrak{I}$ that satisfy the TBox T.

3.2. Privacy Requirements Specifications

We define a privacy requirements specification to be a DL knowledgebase *KB*. The universe of discourse consists of concepts in the TBox *T*, including the set *Req* of data requirements, the set *Actor* of actors with whom data is shared, the set *Action* of actions that are performed on the data, the set *Datum* of data elements on which actions are performed, and the set *Purpose* of purposes for which data may be acted upon. The following definitions precisely define the specification. The concepts for actor, datum and purpose can be organized into a hierarchy using DL subsumption. Figure 3 illustrates three hierarchies from our case study for datum, purposes and actors: inner bullets indicate when a concept is subsumed by the outer bullet concept (e.g., *information* subsumes *public-information* under Datum).

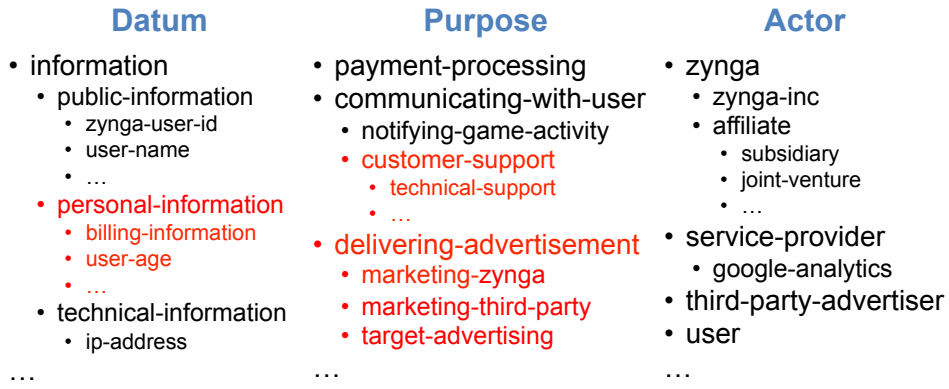

Figure 3. Example datum, purpose and actor hierarchy from Zynga privacy policy expressable in Description Logic; inner bullet concepts are subsumed by (contained within) outer-bullet concepts; red text denotes branches that were inferred to structure orphaned concepts

Definition 1. Each action concept $a \in Action$ has assigned roles that relate the action to actors, data elements and purposes. We begin with three default actions: *COLLECT*, which describes any act by a first party to access, collect, obtain, receive or acquire data from another party; *TRANSFER*, which describes any act by a first party to transfer, move, send or relocate data to another party; and *USE*, which describes any act by a first party to use data in any way for their own purpose. In the future, we may extend these actions, e.g., with aggregation, analysis, storage, and so on, as needed. Actions are further described by DL roles in the set of *Roles* as follows:

- *hasObject.Datum* denotes a binary relationship between an action and the data element on which the action is performed;
- *hasSource.Actor* denotes a binary relationship between an action and the source actor from whom the data was collected;
- *hasPurpose.Purpose* denotes a binary relationship between an action and the purpose for which the action is performed; and
- *hasTarget.Actor* denotes a binary relationship between a *TRANSFER* and the target actor to whom data was transferred

Each action has role *hasObject*, *hasSource* and *hasPurpose*, but only the *TRANSFER* action has the role *hasTarget*. The *hasObject* and *hasSource* roles are to trace data elements from any action back to the original source from which that data was collected, as we discuss in Section 3.2.2.

Definition 2. A *requirement* is a DL equivalence axiom $r \in Req$ that is comprised of the DL intersection of an action concept $a \in Action$ and a role expression that consists of the DL intersection of roles $\exists R_1 \sqcap \ldots \exists R_n \in Roles$. Consider requirement p_5 for $ip\_address \in Datum$, and $delivering\_ad \in Purpose$ in the TBox *T*, such that it is true that:

$$(1) \quad T \vDash p_5 \equiv COLLECT \sqcap \exists hasObject.ip\_address \sqcap \exists hasSource.Actor \sqcap \exists hasPurpose.delivering\_ad$$

Figure 4 illustrates two requirements wherein concepts in the Actor, Datum and Purpose hierarchies (circles) are linked to each requirement via roles (colored arrows): p_5 describes the act to collect IP addresses from anyone for a range of advertising-related purposes; and r_7 describes the collecting IP addresses from advertisers for any purpose.

In addition, each requirement is contained within exactly one modality, which is a concept in the TBox T as follows: *Permission* contains all actions that an actor is permitted to perform; *Obligation* contains all actions that an actor is required to perform; and *Prohibition* contains all actions that an actor is prohibited from performing. Consistent with the axioms of Deontic Logic [13], it is true that $T \vDash Obligation \sqsubseteq Permission$, wherein each required action is necessarily permitted. Thus, if our collection requirement p_5 is required such that $T \vDash p_5 \sqsubseteq Obligation$, then it is also true that $T \vDash p_5 \sqsubseteq Permission$. Using this formulation, we can compare the interpretations of two requirements based on the role fillers to precisely infer any conflicts, a topic considered next in Section 3.2.1.

1. Requirements Conflicts

Our formalism enables conflict detection between what is permitted and what is prohibited. A conflict in predicate logic is expressed as $Permission(x) \wedge Prohibition(x) \leftrightarrow Conflict(x)$, in which x is a DL individual in the ABox A. To implement these techniques, we compute an extension of the TBox that itemizes individual interpretations of the actors, data and purposes.

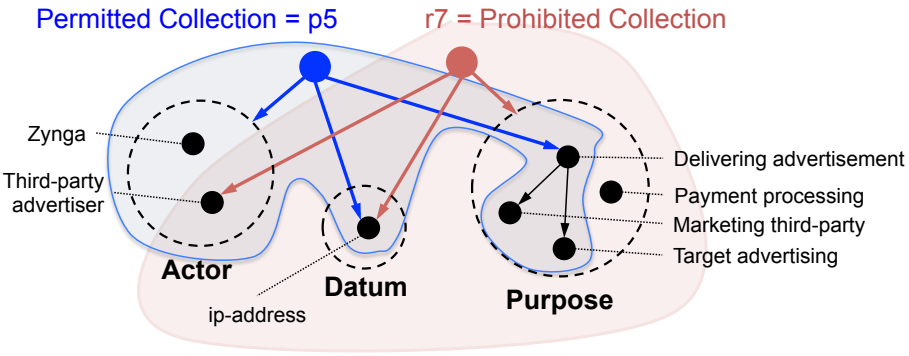

Figure 4. Diagram to illustrate itemized interpretations wherein permssion p5 and prohibition r7 are conflicting but do not subsume one another

The itemized interpretations allow us to identify conflicts within the intersection of complex descriptions that cannot be identified using DL intersection, alone. In Figure 4, the requirement p_5 is a permission, whereas the requirement r_7 is a prohibition. We cannot infer a direct subsumption relationship between these two requirements, because each requirement contains an interpretation that exists outside the other (e.g., Zynga is a permitted source for collecting IP addresses, and payment processing is a prohibited purpose). However, there is a conflict between these two requirements: it is both permitted and prohibited for a third-party to collect IP addresses for advertising-related purposes. To detect these conflicts, we define an extended specification $KB^E = T^E \cup A^E$ that consists of an extended TBox $T^E = T \cup E$ containing the original terminology T and axioms $e \in E$ that itemize interpretations for requirements $r \in T$, such that $T^E \vDash e \sqsubseteq r$. The ABox A^E contains individuals assigned to these interpretations.

Definition 3. The *extension* is a set of axioms E that itemize the interpretations for each requirement. An itemized interpretation of an arbitrary description X is written $(X)^{\mathfrak{I}} = (C)^{\mathfrak{I}} \setminus (D)^{\mathfrak{I}}$ for a concept C that subsumes a concept D. By itemizing interpretations in a requirement's role fillers, we can precisely realize a specific conflicting interpretation across a permission and a prohibition.

For each requirement written in the form $r \equiv a \sqcap \exists R_1.F_1 \sqcap \exists R_2.F_2 \sqcap \ldots \sqcap \exists R_n.F_n$ in the TBox T, such that $a \in \{COLLECT, TRANSFER, USE\}$ and $R_1 \ldots R_n \in Roles$, we derive an itemized interpretation e in the TBox T^E that is written in the form $e \equiv a \sqcap \exists R_1.H_1 \sqcap \exists R_2.H_2 \sqcap \ldots \sqcap \exists R_n.H_n$ by replacing each role filler F_i with a new role filler H_i, which is computed to exclude all sub-concepts $G_j \sqsubset F_i$ in the TBox T as follows: $(H_i)^{\mathfrak{I}} = (F_i)^{\mathfrak{I}} \setminus \bigcup (G_j)^{\mathfrak{I}} \mid (G_j)^{\mathfrak{I}} \subset (F_i)^{\mathfrak{I}}$ for an interpretation $\mathfrak{I}$ that satisfies the TBox T. To realize the itemized interpretation and later report the conflict to an analyst, we assign a unique individual x to the assertion $e(x) \in A^E$.

Definition 4. A *conflict* is an interpretation that is both permitted and required and that satisfies the TBox T^E, such that it is true that $T^E \vDash Conflict \equiv Permission \sqcap Prohibition$. For an individual x in the extended ABox A^E, each conflict is realized with respect to two or more conflicting requirements $r_i, r_j \in Req$, such that it is true that $A^E \vDash$

$r_i(x) \wedge r_j(x) \wedge Conflict(x)$ for $i \neq j$ and an interpretation $\mathfrak{I}$ that satisfies the ABox A^E. If there exists no individual $x \in A^E$ such that $A^E \vDash Conflict(x)$, then a privacy specification *KB* is *conflict-free*.

2. Tracing Data Flows Within a Single Specification

Conflict-free privacy requirements specifications describe permitted collections, transfers and uses of personal information. Using these specifications, we can trace any data element from collection requirements to requirements that permit the use or transfer of that data. This is important because organizations often need to ensure that policies covering collected data are implemented across their organization. Moreover, the actions to use and transfer data may be performed by separate information systems from those where the data is collected, and thus we can use these specifications to discover which systems data is required or permitted to flow to. To trace data across a specification, we introduce the following definitions.

Definition 5. A *trace* is a subset of requirements pairs $(r_s, r_t) \in Req \times Req$ that map from a permitted source action r_s to a permitted target action r_t for an interpretation $\mathfrak{I}$ that satisfies the TBox T. For example, we can trace permitted data collections (source action) to permitted data uses and data transfers (target actions) when the role values for the source actor, datum and purpose entail a shared interpretation. For each requirement written in the form $r_i \equiv a \sqcap \exists R_{i,1}.F_{i,1} \sqcap \exists R_{i,2}.F_{i,2} \sqcap \ldots \sqcap \exists R_{i,n}.F_{i,n}$ in the TBox T, such that $a \in \{COLLECT, TRANSFER, USE\}$ and $R_{i,1} \ldots R_{i,n} \in Roles$, we compare role fillers $F_{i,1} \ldots F_{i,n}$ between the source and target permissions to yield one of four exclusive *Modes* as follows:

- *U: Underflow*, occurs when the data target subsumes the source, if and only if, $T \vDash F_{s,j} \sqsubseteq F_{t,j}$
- *O: Overflow*, occurs when the data source subsumes the target, if and only if, $T \vDash F_{t,j} \sqsubseteq F_{s,j}$
- *E: Exact flow*, occurs when the data source and target are equivalent, if and only if, $T \vDash F_{s,j} \equiv F_{t,j}$
- *N: No flow, otherwise*

Figure 5 presents an example data flow trace from our case study. The collection requirements AOL-16 and AOL-14 trace to the transfer requirement AOL-48. The transfer requirement does not specify a purpose, which we interpret to mean "any purpose." Thus, the collection purposes "business purposes" and "contacting you to discuss our products and services" are more specific than the transfer purpose "any purpose," which the red links illustrate as underflows. The data elements in AOL-16 are similarly more specific than the transfer data elements.

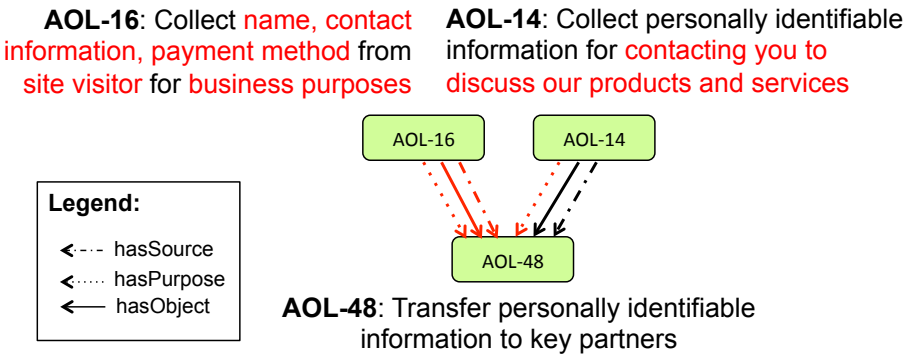

Figure 5. Example data flow trace: red lines represent underflows and black lines represent exact flows.

Below, the collection requirement p_1 in formula (3) encodes part of AOL-16 in Figure 5, and p_2 in formula (4) encodes the corresponding transfer requirement for AOL-48. In formula (2), we declare contact information to be subsumed by personally identifiable information (PII), such that it is true that:

$$(2) \quad T \vDash contact\_info \sqsubseteq PII$$

$$(3) \quad T \vDash p_1 \equiv COLLECT \sqcap \exists hasObject.contact\_info \sqcap \exists hasSource.site\_visitor \sqcap \exists hasPurpose.business\_purposes$$

$$(4) \quad T \vDash p_2 \equiv TRANSFER \sqcap \exists hasObject.PII \sqcap \exists hasSource.Actor \sqcap \exists hasTarget.key\_partners \sqcap \exists hasPurpose.Purpose$$

Based on the subsumption axiom entailed in formula (2), we can map the trace $(p_1, p_2) \longrightarrow (U, U, U)$ onto the three *Modes* for the roles *hasObject*, *hasSource* and *hasPurpose*, respectively. In general, tracing data flows allows an analyst to visualize dependencies between collection, use and transfer requirements. In this paper, we only formalize traces within a single policy. In future work, we will present tracing data flows across multiple policies in a data supply chain. This cross-policy tracing extends our notion of a trace, but requires a shared lexicon or dictionary to unify terminology across two or more policies. In our evaluation, we present select findings from cross-policy tracing.

4 Exploratory Case Study

We conducted a formative, exploratory case study using the Facebook Platform Policy by systematically identifying relevant policy statements that we could formalize into the privacy requirements specification language. We mapped each statement into one of the two categories: *policy statements* and *data requirements*. The *policy statements* category includes statements that describe an action outside the scope of the application such as "You must not violate any law or the rights of any individual or entity." It also includes *non-data requirements* that describe requirements of the app not directly dealing with the handling of data, for example, "You will include your privacy policy URL in the App Dashboard." The *data requirements* category includes statements that describe specific actions performed on data such as "You must not include functionality that proxies, requests or collects Facebook usernames or passwords." After the formative study, we developed our formal language to express privacy requirements and further validated this language in summative study on the two additional policies from Zynga and AOL using this same process. We were particularly interested in boundary cases that describe the limitations of our proposed language.

Figure 6 presents an example data requirement from the Zynga privacy policy (Z-92, where Z is for Zynga). The number, Z-92, within parenthesis denotes that it is statement number 92 within the policy. In step 1, we identify the action using phrase heuristics (e.g., "provide" indicates a TRANSFER action), the modality permission from the modal keyword "will," the datum "information," the target to whom the data is transferred "third party companies" and the purpose "to perform services on our behalf…" Purposes and other values may appear in comma-separate lists, which we interpret as disjunctions. In Figure 6, this purpose includes examples, which we separately translate into a purpose hierarchy similar to that shown in Figure 3. While this policy statement refers to "your information," it is unclear where this information was collected. User data can be collected from the user, data brokers or advertisers.

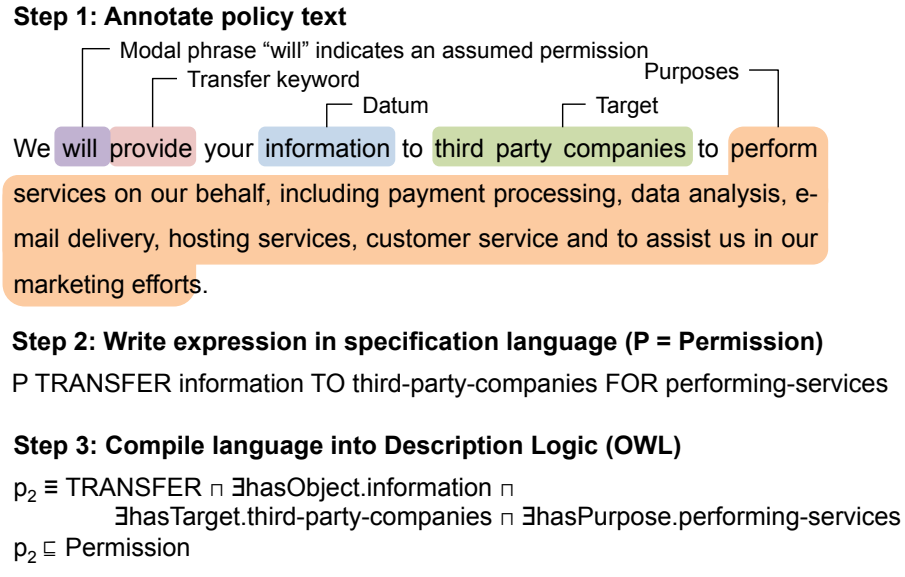

Figure 6. Steps to map data requirement from natural language to DL; step 1 shows data requirement in Zynga privacy policy; step 2 shows requirement expressed in language syntax; step 3 shows satement expressed in DL semantics

After we identify the values to assign to the roles, in step 2 we write these values into a privacy requirements specification language that uses an SQL-like syntax and our DL semantics described in Section 3. The letter "P" indicates that this is a permission, followed by the action verb, the object, and keywords to indicate the source ("FROM"), target ("TO") and the purpose ("FOR"). Once translated into the language, we use a tool to parse the language and generate OWL DL that we reason over using open source DL theorem provers (e.g., HermiT and Fact++).

During the case study, we traced all the keywords to indicate when an action was a collection, use or transfer; these appear in Table II. Among the keywords, many overlap across actions (e.g., access, use, share) while others are more exclusive (e.g., collect, disclose, transfer). The reason for this ambiguity is due to policies that include multiple viewpoints: a policy may describe access to a user's data by the app, which is a collection, or it may describe a third-party's access, which assumes a transfer. In these cases, the analyst should identify the viewpoint to correctly formalize the policy statement and consider reviewing their formalization for keywords that are known to be ambiguous.

TABLE I. PHRASE HEURISTICS USED TO INDICATE WHEN A STATEMENT WAS A COLLECTION, USE OR TRANSFER REQUIREMENT

DL Action	Action keywords
COLLECT	Access, assign, collect, collected, collection, collects, give you, import, keep, observes, provide, receive, record, request, share, use
USE	Access, accessed, communicate, delivering, include, matches, send, use, used, uses, using, utilized
TRANSFER	Access, disclose, disclosed, disclosure, give, in partnership with, include, make public, on behalf of, provide, see, share, shared, transfer, use, used with, utilized by

5 Extended Evaluation

We evaluated our approach by extending our exploratory case study, and implementing a tool-based performance simulation. As a problem domain, we chose the Facebook Platform as our starting point, because Facebook has received significant attention from privacy advocates and Facebook apps are frequently available on mobile device platforms, which provides a second context to study this problem in future work. From here, we chose the Farmville application, which at the time of our study, was the most used Facebook app with over 40.8 million active users per month. We analyzed the following three policies:

- Facebook Platform Policy, last revised 12 Dec 2012, which governs app developer practices in Facebook
- Zynga Privacy Policy, last updated 30 Sep 2011, which governs the user's privacy while they play Farmville and use other Zynga applications
- AOL Advertising, last updated 4 May 2011, which governs advertising distributed through Farmville and other websites and applications

In Table II, we illustrate the scope of this evaluation, including the total number of statements in the policies (S), the number of data requirements (D), which we break-down into the number of permissions (P), obligations (O), and prohibitions (R), including which among these requirements concern collection (C), use (U) and transfer (T) of data. Between 32-55% of these policies described data requirements with generally few obligations. The Zynga and AOL policies describe their own practices and focus more on permissible data practices, whereas the Facebook policy describes developer practices and focuses more on prohibitions. We now discuss findings from our formal analysis that includes conflicts and opportunities to extend our approach, or limitations of the current work.

TABLE II. NUMBER OF TYPES OF STATEMENTS FORMALIZED

Policy	S	D	Modality			Action		
			P	O	R	C	U	T
Facebook	105	39	15	4	25	6	15	14
Zynga	195	64	58	1	8	22	8	15
AOL	74	41	43	0	4	12	15	10

5.1. Example conflicts identified using the language

We found conflicts between Facebook and Zynga, and one conflict within the AOL policy, which we now discuss.

1. Conflicts between Facebook and Zynga

The Facebook Platform policy governs the data practices of Farmville, which is also governed by the developer Zynga's privacy policy. To conduct this conflict analysis, we performed an ontological alignment between terms in both policies that we formalized in DL using equivalence and subsumption. Using our formalization, we detected a conflict between these policies regarding the sharing of aggregate or anonymous data. Facebook requirement FB-43 prohibits a developer from transferring any user data obtained from Facebook to an ad network, whereas Zynga requirement Z-107 permits sharing aggregate data received from any source with anyone:

```
FB-43:  R TRANSFER user-data FROM facebook TO ad-network FOR anything

Z-107:  P TRANSFER aggregate-information,anonymous-information FROM anyone TO anyone
```

The Zynga permission is inferred from an exclusion, which states "Our collection, use, and disclosure of anonymous or aggregated information are not subject to any of the restrictions in this Privacy Policy." The Zynga definition of aggregate-information means non-personally identifiable information, which may include Facebook user data, such as gender, Zip code and birthdate, which are often viewed as not individually identifiable despite evidence to the contrary [21]. Under Facebook, the concept user-data is defined to include aggregate and anonymous data as follows: "By any data we mean all data obtained through the use of the Facebook Platform (API, Social Plugins, etc.), including aggregate, anonymous or derivative data," which we encoded in the datum concept hierarchy.

The second conflict appears where Zynga permits the transfer of unique user IDs to third party advertisers that advertise on Zynga Offer Wall. The purposes for sharing user IDs are crediting user accounts and preventing fraud. However, this sharing violates Facebook requirement FB-43, above. The Zynga requirement Z-113 describes the permission involved in this conflict: the Zynga *user-id*, which Zynga defines as either a unique Zynga user ID or the social networking service user ID, can thus be a data element within the Facebook *user-data*, which includes the Facebook user ID.

```
Z-113:   P TRANSFER unique-id,user-id TO offer-wall-provider FOR crediting-user-account,preventing-fraud
```

Finally, the Facebook and Zynga policies conflict on sharing data for the purposes of merger and acquisition by a third-party. In case of merger or acquisition, Facebook allows a developer to continue using the data within the app, but prohibits the transferring of data outside the app. Zynga does not put restrictions on data transfer, including personal data, for the purpose of merger of acquisition. The Facebook statement "If you are acquired by or merge with a third party, you can continue to use user data within your application, but you cannot transfer data outside your application" (FB-50) and the Zynga statement "In the event that Zynga undergoes a business transition, such as a merger, acquisition… We may transfer all of your information, including personal information, to the successor organization in such transition" (Z-115) map to these two requirements (information includes user data):

```
FB-50:  R TRANSFER user-data FROM facebook TO third-party FOR merger,acquisition

Z-115:  P TRANSFER information FOR merger,acquisition
```

2. Conflict within AOL Advertising

The AOL privacy policy contains an apparent conflict regarding collection and use of personally identifiable information. Unlike the Facebook and Zynga policies, the AOL policy describes data practices from multiple stakeholder viewpoints, simultaneously, including that of their affiliate Advertising.com. The conflict appears from the AOL Advertising viewpoint in a statement, "Personal information such as name, address and phone number is never accessed for [targeted advertising]" (AOL-27). The policy also states, "Advertisers utilizing Advertising.com Sponsored Listings technology may provide personally-identifiable information to Advertising.com Sponsored Listings, which may then be combined with information about purchasing patterns of Advertising.com Sponsored Listings' products and services, ... and all other information provided by the advertiser" (AOL-46). In addition, the following statement declares that this information may be used for targeted advertising: "this information is used to improve the applications provided to advertisers, improve the relevancy of ad serving and any other use deemed helpful to Advertising.com Sponsored Listings" (AOL-47). Note that the advertiser may be collecting the personally identifiable information from the user. The conflicting statements are:

```
AOL-27: R USE personally-identifiable-information FROM registration-environment FOR target-ads-that-are most-appropriate-
        for-site-visitor
```

```
AOL-46: P COLLECT personally-identifiable-information FROM anyone FOR improving-the-applications-provided-to-
        advertisers, improving-the-relevancy-of-ad-serving, anything
```

5.2. Opportunities for extending the language

Among the data requirements that we identified, we were unable to formalize requirements that describe actions outside the scope of collection, use and transfer as defined in Definition 1. The un-encoded requirements include how data is merged and stored and the policy implications of consent. We now discuss these three categories of requirement.

1. Merging data from different sources

The three policies in our study contain 12 requirements that describe how data is linked, combined or aggregated from multiple sources. For example, the Zynga privacy policy states "some of the cookies [that] the service places on your computer are *linked* to your user ID number(s)" (Z-57) and "[information from other sources] will be *combined* with other information we collect" (Z-83), and "additionally, we may keep statistics regarding toolbar use on an *aggregated* basis" (Z-62). In each of these three examples, data is linked, combined or aggregated with different implications. Linking data enables companies to derive inferences from correlations (i.e., statistical analyses) and to re-identify otherwise anonymized data. Combining data with other data raises the question: what purpose governs the combined data, and how long should the combined data be retained (the minimum or maximum period of the previously separate data sets?) Finally, aggregate data decreases the level of detail that an organization has on users. For example, knowing how many users are aged between 21 and 25 years old is different than knowing the specific birth dates of each user. Thus, aggregation requirements may indicate improved user privacy, but they also limit the types of linking and combining that can occur later, if needed.

2. Storing and deleting information

We observed 15 data storage requirements and 8 data deletion requirements in our study. The act of storing, retaining, and deleting data has temporal implications: once data is stored, it exists to be acted upon for the duration of storage; when data is deleted, it is no longer available for use, transfer, etc. For example, the AOL Advertising privacy policy states that, "log files, including detailed clickstream data used to create behavioral segments, are retained… for no longer than 2 years" (AOL-31). While DL is suited for reasoning about subsumption, different temporal logics exist to reasoning about time. We are looking into extensions to DL for temporal reasoning [17] that can be used to express these remaining privacy requirements.

3. Managing the implications of consent

In our analysis, 14 consent requirements were observed that require an organization to permit or prohibit a data action unless a user provides consent to perform that action. We observed two different approaches: opt-in requirements default to data user prohibitions in our language, but can be flipped to permissions when a user provides their consent; opt-out requirements default to data user permissions, but can be flipped to prohibitions when a user chooses to revoke consent. For example, the Facebook Platform Policy contains the opt-in statement, "for all other data obtained through the use of the Facebook API, you must obtain explicit consent from the user who provided the data to us before using it for any purpose other than displaying it back to the user on your application" (FB-42). In contrast, the Zynga Privacy Policy contains the opt-out statement, "when we offer [user] profiles, we will also offer functionality that allows you to opt-out of public indexing of your public profile information" (Z-30). Because opt-in and opt-out statements can change the interpretation of how data may be used and transferred based on the choices of the user, these statements can introduce conflicts into a previously conflict-free policy after the user has made their choice. We plan to further explore how to reason about consent in future work.

5.3. Challenges due to formats and writing styles

We observe different formats and phrasing that affect our approach, which we now discuss.

Embedded policies: A policy may contain hyperlinks to other policies. For completeness, it is important to analyze these links to assess whether the linked content contains relevant data requirements. The additional data requirements may reveal further inconsistent statements within a policy or across multiple policies. In our case study, the Facebook, Zynga and AOL Advertising policies each had 19, 16 and five links, respectively. The links serve different purposes, including linking to policies on special topics such as advertising policies (Facebook) or user rights and responsibilities (Zynga). These special topic policies were hosted by the same company and include additional data requirements, sometimes from a different stakeholder viewpoint. In addition, policies may link to third-party policies, such as

conduit.com (Zynga), or to additional data definitions or specific examples of data requirements (Facebook). Other links, such as "contact us" (AOL) and "change email preferences" (Zynga), do not lead to additional data requirements. Due to the large number of links that may arise across multiple websites, this problem suggests a need for additional automation using natural language processing techniques to identify relevant policies.

Separate collection, use and sharing sections: A policy may describe data collection, purpose for collection, and data sharing requirements in different sections. At the surface, this format makes extracting formal specifications easier, because each statement is relatively independent. However, the format can de-couple the collection requirements from use and transfer requirements through the use of ambiguity (e.g., using different terms or omitting sources, targets and purposes). The Zynga Privacy Policy separately describes the information types collected (see "Information We Collect") from the purposes for use (see "How We Use the Information We Collect"). This separation yields a many-to-many mapping between information types and purposes, because the analyst must reasonably assume that any data type maps to any purpose. In Figure 7, we present the data flow tracing for the hasObject role: the Zynga policy shows numerous requirements (nodes) with multiple cross-traces among collections to transfers due to the many-to-many mapping. Contrast the Zynga policy with the AOL Advertising policy, in which requirements have an observably smaller valiancy or edge count. Many-to-many tracing is likely an indicator of a less privacy protective policy, because it affords companies more opportunities to use data in difficult to comprehend or unforeseeable ways.

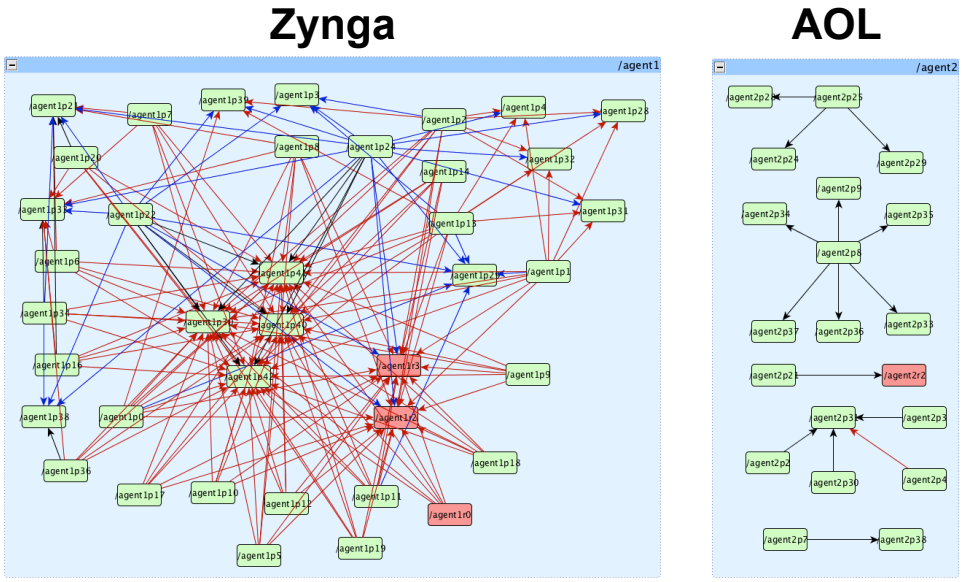

Figure 7. Data flow traces inferred from the Zynga policy (left) and AOL policy (right): arrows point from collections to transfers, red lines show underflows, blue lines show overflows and black lines show exact flows (see Definition 5). The Zynga policy defines broad transfer rights as seen by the three nodes with multiple incoming arrows.

Ambiguous and vague terms: Policies may contain vague or ambiguously worded purposes. For example, the Zynga privacy policy contains a statement, "in some cases, we will associate this information with your user ID number for our internal use" (Z-74). The purpose, "internal use" is vague, and an analyst can interpret this to mean any action performed by the actor, excluding perhaps transfers. Other examples include "operate our business" (AOL-51) and "data analysis" (Z-92). Further, policies may not define data items precisely. For example, the Zynga Privacy Policy describes "personal information," but does not define what this category includes, whereas other policies will refine this term into sub-categories. In such cases, the analyst may need to infer their own subsumption relationships that do not map to specific phrases or statements within the original policy to test for potential conflicts.

Multi-stakeholder viewpoints: A single policy can assign data requirements to multiple stakeholder viewpoints. For example, AOL Advertising describes data practices for sites operated by AOL Advertising, affiliates and subsidiaries as "AOL Advertising Sites" and on sites operated by publishers that participate in the AOL advertising network as "Network Participant Sites." Our approach encodes policies in the first-person viewpoint of a single stakeholder, thus policies such as AOL's Advertising policy can be decomposed into separate policies. In future work, we plan to study ways to analyze data requirements across multiple policies.

5.4. Simulation Results

We conducted a performance simulation to evaluate the computational practicality of using our language to reason about data requirements. While we reduce conflict detection to DL satisfiability, which is PSPACE-complete for a-cyclic TBoxes and the DL family $\mathcal{ALC}$ in which we express our language, we recognize that this bound does ensure that our language is practical for requirements specifications of reasonable size. Therefore, we implemented a prototype parser and compiler for our language using three popular theorem provers: the Pellet OWL2 Reasoner v2.3.0 developed by Clark and Parsia; the Fact++ Reasoner v1.5.2 developed by Dmitry Tsarkov and Ian Horrocks, and the HermIT Reasoner v1.3.4 by the Knowledge Representation and Reasoning Group at the University of Oxford.

We generated 32 privacy requirements specifications with actor, datum and purpose hierarchies comprised of binary trees with 2^3 concepts; this yields specifications with up to 1280 itemized interpretations. We conducted several preliminary runs and determined that concept tree height had no effect on performance. Of the three reasoners, the Pellet Reasoner did not respond within 30 minutes when realizing a policy of only four requirements. Thus, we only discuss results from the Fact++ and HermiT reasoners.

Figure 8 presents the performance time of the Fact++ and HermiT reasoners with respect to the specification size: the 32 runs are sorted along the x-axis from the fewest to the most requirements (from 3 to 72); the y-axis describes the response time in tenths of a second (red) and number of requirements (blue). As the number of requirements increases to 73, we see the Fact++ reasoner response time remains constant, whereas the HermiT response times appear to increase slightly (Pearson's R = 0.533). To understand this increase, we present Figure 9 that compares the Fact++ and HermiT reasoners by number of conflicts: the 32 runs are sorted along the x-axis from fewest to the most requirements (from 3 to 73); the y-axis describes the response time in tenths of a second (red) and the number of conflicts (blue).

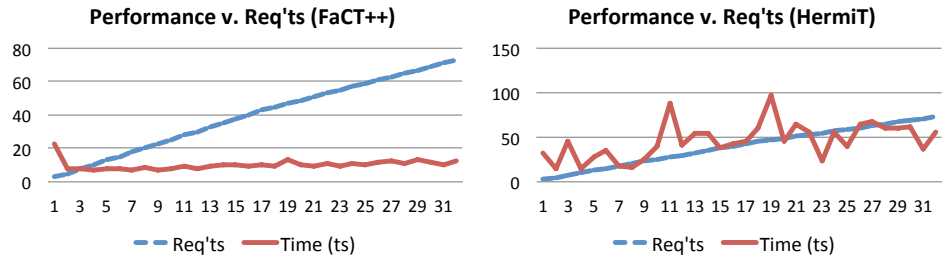

Figure 8. Performance time of Fact++ and HermiT reasoners on privacy requirements specifications with respect to number of requirements

Figure 9 shows, and we confirmed, that the response time of the HermiT reasoner is linear in the number of conflicts (Pearson's R = 0.966). The performance of a thereom prover depends on what type of inferences that prover is optimized to perform: Pellet produces a non-deterministic choice when handling general concept inclusion (GCI) axioms [16], which we rely on in our formalism; however, Fact++ and HermiT are not limited in this way. From this simulation, we believe the language is computationally practical for policies within the order of 100 requirements; however, we need to do more work on usable interfaces to the logic.

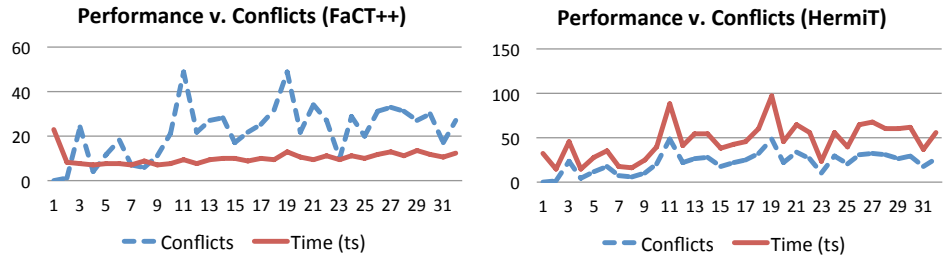

Figure 9. Performance time of Fact++ and HermiT reasoners on privacy requirements specifications with respect to number of conflicts

We now discuss four sets of observations enabled by the RSL: (A) how definitions and exemptions can be used to shape conditionality, or when and how businesses must implement the law; (B) the existence of a narrative and conditional surface structure evidenced by the typed relations; (C) three prominent regulatory structures made salient by the RSL-generated model; and (D) metrics for comparing regulatory specification writing styles.

6 Threats to Validity

Here we discuss the generalizability of our mapping methodology. To address construct validity, we maintained a project workbook that contains mappings of natural language statements to our language syntax and notes about shortfalls and boundary cases in our interpretation. We report on several of these shortfalls in Section 5.2. as limitations of our approach. While mapping statements to our formalism, we often required context outside a given statement to identify the action, source, target and purposes. To compare formalized statements from two policies, we also need to align the lexicons, which requires us to assume answers to such questions as, "is customer service equivalent to customer support, or does prevent crime include the concept of preventing fraud? We documented these assumptions in separate files to allow us to revise our findings as new information became available. In our case study, we found that a given purpose might be described using different descriptions. We plan to conduct human subject studies to understand the limitations of this lexical alignment. If disagreement exists, then our approach may be used to show analysts the consequences of two separate interpretations.

7 Related Work

We now discuss related work in requirements engineering (RE) and formal methods. In RE, Antón et al. analyzed over 40 privacy policies using goal mining, which is a method to extract goals from texts [1, 2]. Results include a clear need to standardize privacy policies and evidence to support a frame-based representation consisting of actors, actions, and constraints. Breaux et al. later extended this representation with notions of rights, obligations and permissions in a case study [6] and then formalized this extension in Description Logic [8]. Since, Young introduced a method for mining commitments, privileges and rights from privacy policies to identify requirements [24]. Commitments describe pledges that one actor will perform an action and these commitments are frequently found throughout privacy policies. Wan and Singh formalized commitments in an agent-based system, but had not applied this formalism to privacy policy [23]. In this paper, we describe a method to formalize specific data-related commitments, privileges and rights in privacy policies to logically reason about potential conflicts.

Formal and semi-methods have long been applied to privacy policy and privacy law as an application area. Early work on semi-formal privacy policy languages includes the Platform for Privacy Preferences (P3P), a website XML-based policy language aimed to align web browser user privacy preferences with website practices [10]. While P3P has experienced wide spread adoption, the P3P is a declarative language and website operators often make mistakes in how they configure these policies [15]. The EPAL is another declarative language that can be used to express data policies with constraints on purpose [19]. Unlike declarative languages, languages with a formal semantics can be used to reason about specification errors and inform website operators and other parties who depend on these policies about why a policy is erroneous, e.g., by presenting analysts with conflicting policies for resolution.

Several researchers have since formalized privacy-relevant regulations, including the HIPAA Privacy Rule [5, 18] and the Privacy Act [12]. Barth et al. encoded regulations as messages passed between actors using norms (e.g., permitted and prohibited actions), which is similar to Aucher et al. [3]. May encoded privacy regulations in Promela and used the Spin model checker to identify potential conflicts [18]. These prior approaches are limited in that they cannot express the hierarchical nature of actor roles, data composition, and purposes needed to describe privacy policies. Alternatively, others have used the Web Ontology Language (OWL) to formalize policies using permissions, obligations and prohibitions and to address this issue of concept hierarchies [14, 22]. The full OWL, which these prior approaches each use to express their formalization, is known to be undecidable. Work by Uszok et al., however, use algorithms to identify conflicts as opposed to theorem proving; an approach that may be decidable, but which is difficult to reproduce and generalize as the algorithms are not explicitly published. In this paper, we extend this prior work by reducing conflict detection to DL satisfiability, which is known to be PSPACE-complete for the $\mathcal{ALC}$ family of DL, and we believe our conflict detection technique is generalizable to a larger class of requirements than those found in privacy policies.

8 Discussion and Conclusions

In this paper, we presented a formal language to encode data requirements from natural language privacy policies so that an analyst can reason about these policies by checking for conflicts and tracing permissible and prohibited data flows within the policies. We applied the language to real-world policies from Facebook, Zynga and AOL Advertising in a case study. The study demonstrates how to identify conflicts, which an analyst can then resolve by modifying their

policy and/or their privacy practices. We also discuss limitations of the data requirements specification language and opportunities for improving the language. Finally, we conducted a simulation to demonstrate the computational complexity of identifying conflicts in policies of similar size. As software increasingly leverages platforms and third-party services, we believe developers need lightweight formalisms and tools such as this to check their intentions against policies in the larger ecosystem. This is especially true as developers work with compositions of services in which they are not aware of all the third parties in their data flow. In future work, we plan to consider multi-stakeholder interactions across more complex service compositions.

Acknowledgment

We thank Dave Gordon, Hanan Hibshi and Darya Kurilova for their early feedback, and the Requirements Engineering Lab at Carnegie Mellon University.

References

[1] A.I. Antón, J.B. Earp, Q. He, W. Stufflebeam, D. Bolchini, C. Jensen, "Financial privacy policies and the need for standardization," *IEEE Sec. & Priv.*, 2(2):36-45, 2004.

[2] A.I. Antón, J.B. Earp, "A requirements taxonomy for reducing web site privacy vulnerabilities," Req'ts Engr. J., 9(3):169-185, 2004.

[3] G. Aucher, G. Boella, L. van der Torre. "Privacy policies with modal logic: a dynamic turn," LNCS 6181: 196-213, 2010.

[4] F. Baader, D. Calvenese, D. McGuiness (eds.), The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, 2003.

[5] A. Barth, A. Datta, J.C. Mitchell, H. Nissenbaum, "Privacy and Contextual Integrity: Framework and Applications," IEEE Symp. on Sec. & Priv., 2006, pp. 184-198.

[6] T.D. Breaux, A.I. Antón, "Analyzing Goal Semantics for Rights, Permissions, and Obligations." IEEE Req'ts. Engr. Conf., Paris, France, pp. 177-186, 2005.

[7] T.D. Breaux, A.I Antón. "Analyzing regulatory rules for privacy and security requirements." IEEE Trans. Soft. Engr., Special Issue on Soft. Engr. for Secure Sys., 34(1):5-20, 2008.

[8] T.D. Breaux, A.I. Antón, J. Doyle, "Semantic parameterization: a conceptual modeling process for domain descriptions." ACM Trans. Soft. Engr. Method., 18(2): Article 5, 2009.

[9] T.D. Breaux, D.L. Baumer, "Legally 'Reasonable' Security Requirements: A 10-year FTC Retrospective." Computers & Security, 30(4):178-193. 2011.

[10] L. Cranor et al., "Platform for Privacy Preferences (P3P) Specification," W3C Working Group Note, 2006,

[11] C.B. Farrell. "FTC Charges Deceptive Privacy Practices in Google's Rollout of Its Buzz Social Network," U.S. Federal Trade Comission News Release, March 30, 2011.

[12] C. Hanson, T. Berners-Lee, L. Kagal, G.J. Sussman, D. Weitzner, "Data-purpose algebra: modeling data usage policies." 8th IEEE Work. Pol. Dist. Sys. & Nets., 2007, pp. 173-177.

[13] J.F. Horty. "Deontic logic as founded in non-monotonic logic." Annals of Math. & Art. Intel., 9: 69-91, 1993.

[14] M. Kahmer, M. Gilliot, G. Muller, "Automating Privacy Compliance with ExPDT." 10th IEEE Conf. E-Com. Tech., pp. 87-94, 2008

[15] P.G. Leon, L.F. Cranor, A.M. McDonald, R. McGuire, "Token attempt: the misrepresentation of website privacy policies through the misuse of p3p compact policy tokens," 9th Workshop on Priv. Elec. Soc., pp. 93-104, 2010.

[16] Lin, H. T., Sirin, E. (2008). Pellint - A Performance Lint Tool for Pellet. International Workshop on OWL: Experiences and Directions (OWL-ED 2008).

[17] C. Lutz, F. Wolter, M. Zakharyashev, "Temporal description logics: A survey." 15th IEEE Int'l Symp. Temp. Rep. & Reas., pp. 3-14, 2008

[18] M.J. May, Privacy APIs: Formal Models for Analyzing Legal and Privacy Requirements, Ph.D. Thesis, U. of Pennsylvania, 2008.

[19] C. Powers, M. Schunter, "Enterprise Policy Authorization Language," Version 1.2, W3C Member Submission, Nov. 2003.

[20] E. Steel, G.A. Fowler. "Facebook in privacy breach." Wall Street Journal, October 17, 2010.

[21] Sweeney, Latanya. "k-anonymity: a model for protecting privacy." Int'l J. of Uncertainty, Fuzziness and Knowledge-Based Sys., 10(5): 557-570, 2002.

[22] A. Uszok, J.M. Bradshaw, J. Lott, M. Breedy, L. Bunch. "New Developments in Ontology-Based Policy Management: Increasing the Practicality and Comprehensiveness of KAoS." IEEE Work. on Pol. Dist. Sys. & Nets., pp. 145-152, 2008.

[23] F. Wan, M.P. Singh. "Formalizing and achieving multiparty agreements via commitments." Auto. Agents & Multi-Agent Sys., pp. 770–777, 2005.

[24] J. Young. "Commitment analysis to operationalize software requirements from privacy policies." Req'ts Engr. J., 16:33-46, 2011.