Requirements Eng
DOI 10.1007/s00766-013-0190-7

RE2013

Eddy, a formal language for specifying and analyzing data flow specifications for conflicting privacy requirements

Travis D. Breaux · Hanan Hibshi · Ashwini Rao

Received: 25 August 2013 / Accepted: 26 October 2013

Abstract Increasingly, companies use multi-source data to operate new information systems, such as social networking, e-commerce, and location-based services. These systems leverage complex, multi-stakeholder data supply chains in which each stakeholder (e.g., users, developers, companies, and government) must manage privacy and security requirements that cover their practices. US regulator and European regulator expect companies to ensure consistency between their privacy policies and their data practices, including restrictions on what data may be collected, how it may be used, to whom it may be transferred, and for what purposes. To help developers check consistency, we identified a strict subset of commonly found privacy requirements and we developed a methodology to map these requirements from natural language text to a formal language in description logic, called Eddy. Using this language, developers can detect conflicting privacy requirements within a policy and enable the tracing of data flows within these policies. We derived our methodology from an exploratory case study of the Facebook platform policy and an extended case study using privacy policies from Zynga and AOL Advertising. In this paper, we report results from multiple analysts in a literal replication study, which includes a refined methodology and set of heuristics that we used to extract privacy requirements from policy texts. In addition to providing the method, we report results from performing automated conflict detection within the Facebook, Zynga, and AOL privacy specifications, and results from a computer simulation that demonstrates the scalability of our formal language toolset to specifications of reasonable size.

Keywords Privacy · Requirements · Standardization · Description logic · Formal analysis

T. D. Breaux (✉) · H. Hibshi · A. Rao
Institute for Software Research, Carnegie Mellon University, Pittsburgh, PA, USA
e-mail: breaux@cs.cmu.edu

H. Hibshi
e-mail: hhibshi@cmu.edu

A. Rao
e-mail: arao@cmu.edu

1 Introduction

Emerging Web and mobile information systems leverage user data that are collected from multiple sources without a clear understanding of data provenance or the privacy requirements that should follow these data. These systems are increasingly based on multi-tier platforms in which each "tier" may be owned and operated by a different party, 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 and the Facebook and LinkedIn social networking platforms. Application marketplaces, such as Amazon Appstore, Google Play, and iTunes, have also 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. Therefore, platform and application developers bear increased, shared responsibility to protect user data as they integrate their services into multi-tier ecosystems.

In Canada, Europe, and the United States, privacy policies, also called privacy notices, have served as contracts between users and their service providers and, in the United

States, these policies are often the sole means to enforce accountability [12]. In particular, Google has been found to repurpose user data across their services in ways that violated earlier versions of their privacy policy [19], and Facebook's third-party apps were found to transfer Facebook user data to advertisers in violation of Facebook's platform policies [34]. In response to privacy policy violations, the US Federal Trade Commission (FTC) has brought regulatory enforcement actions against companies whose business practices were inconsistent with their privacy policies [12]. In addition, multiple US laws, such as regulations implementing the Fair Credit Reporting Act (FCRA), the Gramm–Leach–Bliley Act (GLBA), the Children Online Privacy Protection Act (COPPA), and the Health Information Portability and Accountability Act (HIPAA), require companies to inform their customers about their privacy practices, which is generally done through privacy notices. In 2013, the California Attorney General Kamela Harris forged an agreement with major mobile app platforms, including Google, Apple, Microsoft, and others, to require app developers to post privacy policies for their apps in mobile app marketplaces. Given the pressure to post privacy policies and the pressure to keep policies honest, companies need tools to align their policies and practices. In this respect, we believe that developers need tools to specify their data flows at a requirements and architectural level of abstraction (i.e., denoting the actors, information types, and general business practices involved) and that privacy policies only present a subset of this view to the general public. Certain practices and actors will be intentionally kept from public view as these entities sometimes provide the developer or company a strategic advantage in the marketplace.

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 series of studies to formalize a subset of privacy-relevant requirements from privacy policies. We show that such a formalism can be used to verify that privacy requirements are consistent across this ecosystem: "app" developers can express their intentions, formally, and then check whether these intentions conflict with the requirements of third parties. Furthermore, platform developers can verify that their platform policy requirements are consistent with app developer requirements.

1.1 Contributions

Our main contributions are as follows: (1) we systematically identify a subset of privacy-relevant requirements from privacy policies using a case study research method; (2) we formalize the data requirements subset in a privacy requirements specification language, called Eddy,[1] that is 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 subsumption, while ensuring decidability and reasonable bounds on computational complexity; (4) we model tracing of data flows within a single privacy policy. These contributions were first demonstrated in a formative study [13] and have since been further validated through a literal replication study with multiple analysts and additional results that are reported, herein.

In case study research, a literal replication study serves to evaluate whether an emerging theoretical framework yields the same results in the same context [39]. While empirical case studies emphasize repeatability, the numerous factors and assumptions that underlie the case study context can still lead to variability in the results. The logic of a literal replication is analogous to repeating experiments: the case study investigator may alter one or two conditions to determine whether these conditions affect the outcome. In addition, each time, a case study design is executed and new lessons emerge about the ability of existing theory to explain or predict the results. We conducted our replication by changing two conditions: first, the coders in our replication changed from the third author to the first and second author; second, the coders began with an enhanced coding methodology that was developed in the exploratory study. The first change aims to evaluate the impact of different coders on the methodology (the first contribution of our work), and the second change aims to evaluate the further generalizability of the formal language (the second contribution of our work).

The remainder of the paper is organized as follows: in Sect. 2, we introduce a running example based on our formative case study; in Sect. 3, we introduce the Eddy language that we derived from this study; in Sect. 4, we describe our coding method for translating natural language text policies into the language; in Sect. 5, we report our case study replication findings to evaluate the language across three privacy-related policies; in Sect. 6, we report the results of a simulation to evaluate the runtime performance of conflict detection using our language and open-source theorem provers; in Sect. 7, we consider threats to validity; in Sect. 8, we review related work, including other approaches to model privacy requirements; and in Sect. 9, we conclude with discussion and summary.

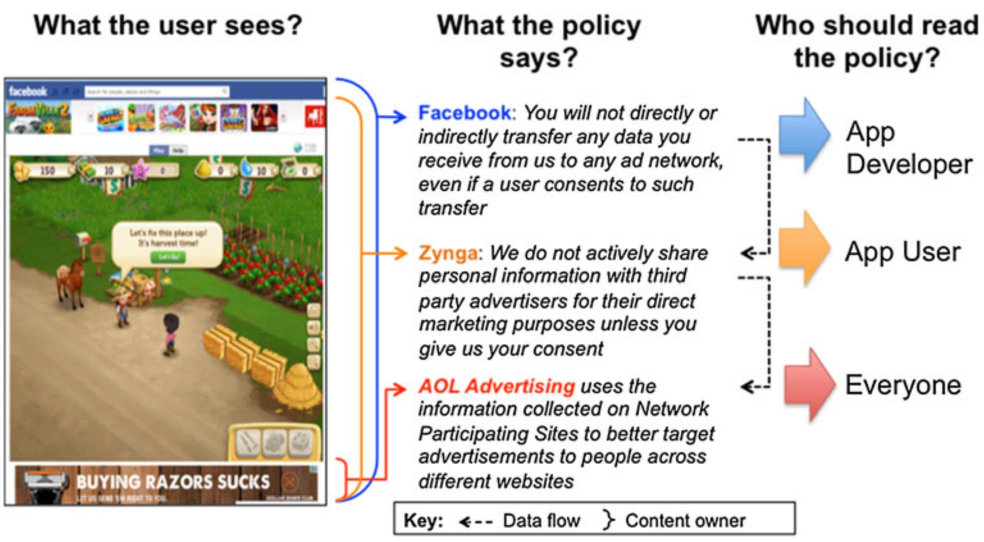
[1] For the purposes of comparison to other approaches, we call our privacy requirements specification language "Eddy" for the circular movement of water that runs counter to the main current, a connotation that appears appropriate when describing data flows in multi-tier systems where the flow of data may run counter to conflicting privacy requirements.

2 Running example

We illustrate the problem and motivate our approach using a running example: in Fig. 1, we present privacy policy excerpts from the Facebook platform policy that governs Zynga, the company that produces the depicted Farmville game. The solid colored arrows trace from the visual elements that the user sees in their Web browser on the left-hand side to governing policy excerpts in the middle; on the right-hand side, we summarize the target audience who should read these policies (e.g., developers, users, or everyone). The dotted black lines along the right-hand side show how data flow 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 that have their own platform developer policies that are not depicted here.

As the platform provider, Facebook manages basic user account information, including user IDs, friend lists, and other data that are made available to Zynga under the Facebook platform policy. The Facebook policy excerpt in Fig. 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 AOL Advertising is a third-party advertiser placing ads through the social plugin Zynga, the platform provider 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.

We illustrate a data supply chain between a user, Facebook, Zynga, and AOL that is shown in Fig. 2. The arrows denote data flows among the four actors, and the paper icons represent policies that regulate these data 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 developer’s app, such as Zynga’s Farmville app. However, the Facebook platform policy prohibits Zynga from transferring 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 that she uses directly. However, it is unlikely that the user is aware of AOL’s privacy agreement or of data flows to AOL, which we represent as a hidden data flow in Fig. 2. To identify the advertiser supplying the ad in Fig. 1, “Buying Razors Sucks,” we had to collect TCP/IP network traffic data using the Wireshark traffic analyzer. 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* Web site, however, the link (http://www.advertising.com/privacy_policy.php) to their privacy policy reported an error message instead of presenting the privacy policy. 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. We assume that most users either would not be able to, or would not bother to, go through this level of effort to find the advertiser’s name and privacy policy.



Facebook: *You will not directly or indirectly transfer any data you receive from us to any ad network, even if a user consents to such transfer*

Zynga: *We do not actively share personal information with third party advertisers for their direct marketing purposes unless you give us your consent*

AOL Advertising uses the information collected on Network Participating Sites to better target advertisements to people across different websites

Fig. 1 Privacy policy excerpts and idealized data flows mapped to Web content that the user sees in their browser; these policies affect different categories of stakeholder

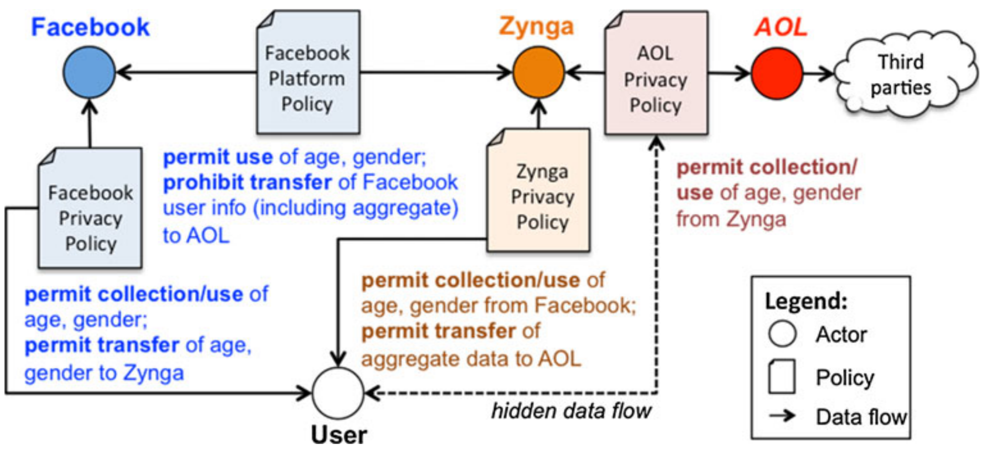

Facebook
Zynga
AOL
Facebook Platform Policy
AOL Privacy Policy
Third parties
Facebook Privacy Policy
Zynga Privacy Policy
permit use of age, gender; prohibit transfer of Facebook user info (including aggregate) to AOL
permit collection/ use of age, gender from Zynga
permit collection/use of age, gender; permit transfer of age, gender to Zynga
permit collection/use of age, gender from Facebook; permit transfer of aggregate data to AOL
hidden data flow
User
Legend:
Actor
Policy
Data flow

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

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. These parties have different stakes in the data flow: Zynga and Facebook have a different first-party relationship with the user, and Zynga relies primarily on Facebook for a large portion of their user base; thus, Zynga aims to respect Facebook's platform policy, while sharing sufficient data with advertisers to generate needed revenue; meanwhile, AOL does not have a first-party relationship with the user and thus has less incentive to restrict their practices. Moreover, AOL is one of many advertisers and Zynga may be willing to change advertisers if AOL's practices are unacceptable. That said, policies expressed in natural language presently remain disconnected and hence, software can freely deviate from the coordination required and expected across these different parties. To address this problem, we propose the Eddy language as an interlingua to describe requirements that map natural language policy statements to formal logical statements that can be checked for consistency and eventually traced to other software artifacts.

3 Approach

Our approach is to improve privacy by introducing a privacy requirements specification into the regular software development process that serves to align multi-party expectations across multi-tier applications. This specification would serve as a high-level design document that can be referenced by both privacy law experts as well as software developers, and we envision that some portion of these specifications could be used to generate privacy policies for users. The specification would express a critical subset of policy statements in a formalism that developers can formally check for requirements conflicts. This includes conflicts within a single party's privacy 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 DL [10]. The DL is suitable for expressing and reasoning over ambiguity that frequently appears in natural language requirements. Whereas other requirements specifications may be expressed in linear temporal logic to reason over constraints on events, timings, and concurrency, we use the DL to check whether terms correctly contain, exclude, or equate to other terms in a specification. For analyzing privacy requirements, this allows us to answer questions concerning how information is shared and used based on how different types of information have been defined. Semantic parameterization was validated using 100 privacy policy goals [8] and over 300 data requirements governing personal health information [9]. For a detailed analysis of the advantages of our approach, see Sect. 8.5 for a discussion of related work and further justification for using DL. We now introduce DL, followed by our precise definition of the privacy requirements specification.

3.1 Introduction to description logic

Description logic 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 [6]. In this paper, we use the DL family 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 ALC are PSPACE-complete [6].

Reasoning in DL begins with an interpretation $\mathfrak{I}$ that consists of a non-empty set $\Delta^{\mathfrak{I}}$, called the *domain of* *interpretation*, and the interpretation function $.^{\mathfrak{I}}$ 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 role $(a, b) \in R^{\mathfrak{I}}$, b is called the filler of that role. Description logic defines two special concepts: $\top$ (top) with the interpretation $\top^{\mathfrak{I}} = \Delta^{\mathfrak{I}}$ and $\bot$ (bottom) with interpretation $\bot^{\mathfrak{I}} = \varnothing$. In addition to constructors for union, intersection, and negation, DL provides a constructor to constrain role values, written *R.C*, which means that the filler for the role *R* belongs to the concept *C*. The interpretation function is extended to concept definitions in the DL family 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:

$$(\neg C)^{\mathfrak{I}} = \Delta^{\mathfrak{I}} \backslash C^{\mathfrak{I}}$$

$$(C \sqcap D)^{\mathfrak{I}} = C^{\mathfrak{I}} \cap D^{\mathfrak{I}}$$

$$(C \sqcup D)^{\mathfrak{I}} = C^{\mathfrak{I}} \cup D^{\mathfrak{I}}$$

$$(\forall R.C)^{\mathfrak{I}} = \{a \in \Delta^{\mathfrak{I}} | \forall b.(a, b) \in R^{\mathfrak{I}} \rightarrow b \in C^{\mathfrak{I}}\}$$

$$(\exists R.C)^{\mathfrak{I}} = \{\ a \in \Delta^{\mathfrak{I}} | \exists b.(\mathrm{a},\ \mathrm{b}) \in R^{\mathfrak{I}} \wedge b \in C^{\mathfrak{I}}\}$$

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 $T \vDash D \sqcap C \rightarrow \bot$, if $D^{\mathfrak{I}} \cup C^{\mathfrak{I}} = \varnothing$ for all interpretations $\mathfrak{I}$ that satisfy the TBox *T*. Finally, 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*.

Software engineers and database enthusiasts may see a natural alignment between concepts and individuals and classes and instances, e.g., in object-oriented modeling; however, this alignment can be misleading. While it is true that we can query a DL *KB* to ask whether an individual is an instance of a concept, the ABox represents many different interpretations and not a finite model. In our application of DL, we frequently define named concepts for actors, such as Zynga and Facebook. This allows us to reason about data practices solely in the TBox, and we reserve the ABox for high-precision conflict detection, which we discuss in more detail in Sect. 2.

3.2 Privacy requirements specifications

We define a privacy requirements specification in the Eddy language 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 among whom data are 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, purpose, and actor: inner bullets indicate when a concept is subsumed by the outer bullet concept (e.g., *information* subsumes *public-information* under the Datum hierarchy).

Definition 1 Each action concept $a \in Action$ has assigned roles that relate the action to actors, data elements, and purposes. We begin with four default actions: *COLLECT*, which describes any act by a party to access, collect, obtain, receive, or acquire data from another party; *USE*, which describes any act by a party to use data in any way for their own purpose; *RETAIN*, which describes any act by a party to retain, store, or preserve data; and *TRANSFER*, which describes any act by a party to transfer, move, send, or relocate data to another party. In the future, one can extend these actions, e.g., with aggregation, analysis, 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 were 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* action and the target actor to whom data were transferred

Each action definition is expressed using the roles *hasObject*, *hasSource*, and *hasPurpose*, but only the *TRANSFER* action is expressed using 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 were collected, as we discuss further with respect to data flow tracing in Sect. 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

Datum

- information
  - public-information
    - zynga-user-id
    - user-name
    - …
  - *personal-information*
    - *billing-information*
    - *user-age*
    - *…*
  - technical-information
    - ip-address

…

Purpose

- payment-processing
- communicating-with-user
  - notifying-game-activity
  - *customer-support*
    - *technical-support*
    - *…*
- *delivering-advertisement*
  - *marketing-zynga*
  - *marketing-third-party*
  - *target-advertising*

…

Actor

- zynga
  - zynga-inc
  - affiliate
    - subsidiary
    - joint-venture
    - …
- service-provider
  - google-analytics
- third-party-advertiser
- user

…

Fig. 3 Example datum, purpose, and actor hierarchy from Zynga privacy policy expressible in description logic; *inner bullet* concepts are subsumed by (contained within) *outer bullet* concepts; *italicized red text* denotes branches that were inferred to include otherwise orphaned concepts; *black sub-bullets* were inferred using refinement heuristics that we describe in Sect. 4 (color figure online)

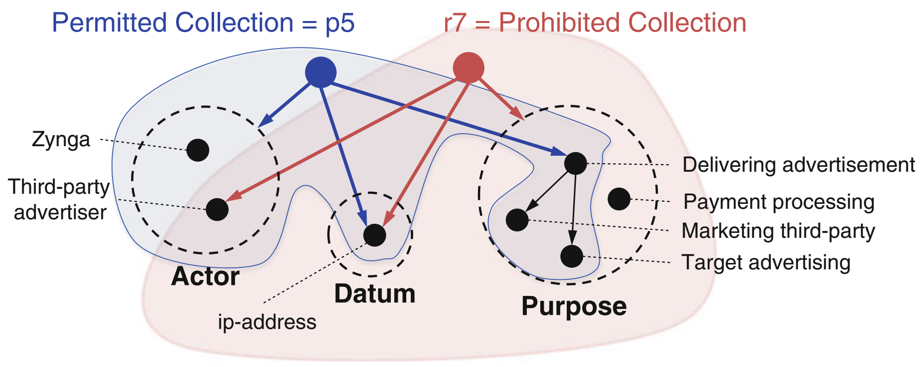


Fig. 4 Diagram to illustrate itemized interpretations wherein permission p_5 and prohibition r_7 are conflicting but do not subsume one another; p_5 permits collection of IP address from anyone for marketing purposes, whereas r_7 prohibits collection of IP address from advertisers for any purpose

$$\begin{aligned} T \vDash p_5 \equiv\ & \mathit{COLLECT} \sqcap \exists \mathit{hasObject.ip\_address} \sqcap \\ & \exists \mathit{hasSource.Actor} \sqcap \\ & \exists \mathit{hasPurpose.delivering\_ad} \end{aligned} \tag{1}$$

In formula (1), the source filler of the collection is a topmost concept "Actor;" this convention is used to mean "anyone" or "any actor" and serves to complete the requirement specification in the presence of an ambiguity or unstated role. 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 an act to collect IP addresses from anyone for a range of advertising-related purposes; and r_7 describes an act to collect 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. We adapted the axioms of Deontic Logic to our approach, wherein a required action is necessarily permitted [21]; hence, it is true that $T \vDash \mathit{Obligation} \sqsubseteq \mathit{Permission}$, in which each required action is necessarily permitted. Thus, if our collection requirement p_5 is required such that $T \vDash p_5 \sqsubseteq \mathit{Obligation}$, then it is also true that $T \vDash p_5 \sqsubseteq \mathit{Permission}$. This formalization can be extended using a concept called exclusions, wherein a specification excludes certain permissions from the specification and then the reasoner can detect situations where an obligation exists without a corresponding permission. For example, an exclusion of permission is expressed in the TBox such that it is true that $T \vDash \mathit{Exclusion\,Of\,Permission} \equiv \mathit{Modality} \sqcap \neg \mathit{Permission}$. Using our requirements formalization, we can compare the interpretations of two requirements based on the role fillers to precisely infer any conflicts, a topic considered next in Sect. 3.2.1.

3.2.1 Detecting requirements conflicts

Our formalism enables conflict detection between what is permitted and what is prohibited. A conflict in predicate logic is expressed as $\mathit{Permission}(x) \wedge \mathit{Prohibition}(x) \leftrightarrow \mathit{Conflict}(x)$, in which x is a DL individual in the ABox. A second approach is to express $\mathit{Permission} \sqcap \mathit{Prohibition} \rightarrow \bot$ in the TBox, which means there is no interpretation that can be satisfied by an individual, that is, both permitted and prohibited. This second approach yields an inconsistent TBox, which we can detect in reasonable time. However, to support analysts who want to de-conflict their privacy specifications, we need to precisely identify these conflicts and present them to the analyst. Thus, we adopted the first approach by computing an extension of the TBox that itemizes individual interpretations of the actors, data, and purposes. By itemize, we mean every interpretation is assigned to an individual that no two individuals are equivalent and no individual is unnamed.

The itemized interpretations allow us to identify conflicts within the intersection of complex descriptions that cannot be identified using DL subsumption or intersection, alone. In Fig. 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 advertiser 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 the interpretations of 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 $(X)^{\mathfrak{I}} = (C)^{\mathfrak{I}} \backslash (D)^{\mathfrak{I}}$ 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 shared by a permission and 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 $a \in \{COLLECT, USE, RETAIN, TRANSFER\}$, 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 subconcepts $G_j \sqsubset F_i$ in the TBox T as follows: $(H_i)^{\mathfrak{I}} = (F_i)^{\mathfrak{I}} \backslash \cup (G_j)^{\mathfrak{I}} | (G_j)^{\mathfrak{I}} \subset (F_i)^{\mathfrak{I}}$ 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 $T^E \vDash Conflict \equiv Permission \sqcap Prohibition$. 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*.

3.2.2 *Tracing data flows within a single specification*

Conflict-free privacy requirements specifications describe permitted collections, uses, retentions, and transfers of personal information. Using these specifications, we can trace any data element from collection requirements to requirements that permit the use, retention, 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, retain, and transfer data may be performed by separate information systems from those where the data are originally collected, and thus, we can use the formal specifications to discover to which systems the data are required or permitted to flow. To trace data within a specification, we introduce the following definitions.

Definition 5 A *trace* is a subset of requirements pairs $(r_s, r) \in Req \times Req$ that maps 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 $a \in \{COLLECT, USE, RETAIN, TRANSFER\}$ 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 *Flow Modes* as follows:

- *U: Underflow* occurs when the data source is subsumed by the target, if and only if, $T \vDash F_{s,j} \sqsubseteq F_{t,j}$
- *O: Overflow* occurs when the data target is subsumed by the source, 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 nodes represent requirements and the edges represent the traced roles for *hasPurpose* (dotted), *hasSource* (dashed), and *hasObject* (solid). In Fig. 5, the red text corresponds to those roles for which an underflow was detected and the black text corresponds to exact flows, e.g., both AOL-14 and AOL-48 refer to personally identifiable information; and implied actors or purposes appear in parentheses. The collection requirements such as AOL-16 and AOL-14 trace to the transfer requirement AOL-48. The transfer requirement does not specify a purpose, which we consistently 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 of name, contact information, and payment method in AOL-16 are similarly more specific than the transfer data element of personally identifiable information, which leads to another underflow. Finally, because AOL-48 does not restrict the data source, we infer the source to be "anyone," which includes site visitors in AOL-16.

Below, the collection requirement p_1 in formula (3) encodes part of AOL-16 in Fig. 5, and p_2 in formula (4) encodes the corresponding transfer requirement for AOL-48, wherein the implied source and purpose are mapped to the general concept for actor and purpose, respectively. In formula (2), contact information is subsumed by personally identifiable information (PII); thus, it is true that:

$$T \vDash contact\_info \sqsubseteq PII \tag{2}$$

$$T \vDash p_1 \equiv COLLECT \sqcap \exists hasObject.contact\_info \sqcap \exists hasSource.site\_visitor \sqcap \exists hasPurpose.business\_purposes \tag{3}$$

$$T \vDash p_2 \equiv TRANSFER \sqcap \exists hasObject.PII \sqcap \exists hasSource.Actor \sqcap \exists hasTarget.key\_partners \sqcap \exists hasPurpose.Purpose \tag{4}$$

Corresponding transfer requirements are detected automatically using DL queries constructed from the formalized collection requirement. Based on the subsumption axiom entailed in formula (2), we can map the trace $(p_1, p_2) \rightarrow (U, U, U)$ to the three *Flow Modes* for the roles *hasObject*, *hasSource*, and *hasPurpose*,

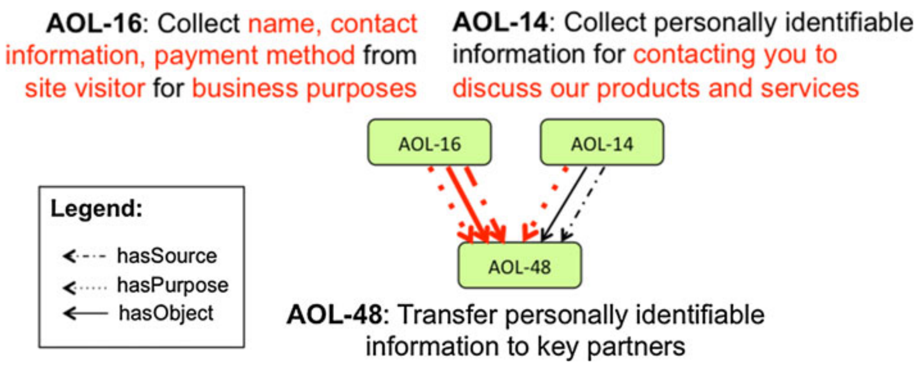

AOL-16: Collect name, contact information, payment method from site visitor for business purposes
AOL-14: Collect personally identifiable information for contacting you to discuss our products and services
AOL-16
AOL-14
AOL-48
Legend:
hasSource
hasPurpose
hasObject
AOL-48: Transfer personally identifiable information to key partners

Fig. 5 Example data flow trace: *thick red lines* represent underflows and *thinner black lines* represent exact flows; *each line* corresponds to a different role; implied actors and purposes appear in *parentheses* (color figure online)

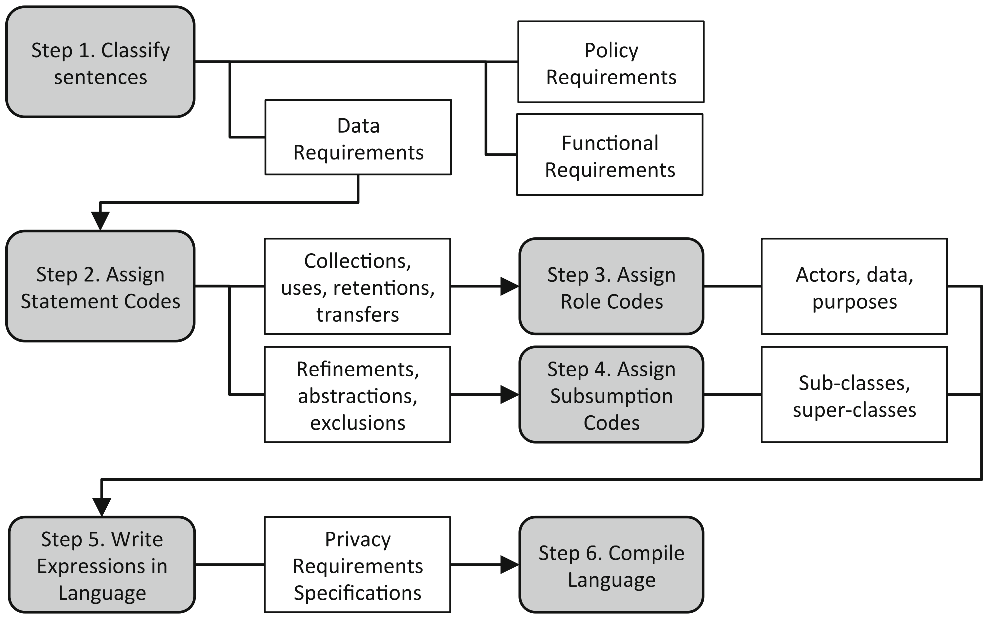

Step 1. Classify sentences
Policy Requirements
Data Requirements
Functional Requirements
Step 2. Assign Statement Codes
Collections, uses, retentions, transfers
Step 3. Assign Role Codes
Actors, data, purposes
Refinements, abstractions, exclusions
Step 4. Assign Subsumption Codes
Sub-classes, super-classes
Step 5. Write Expressions in Language
Privacy Requirements Specifications
Step 6. Compile Language

Fig. 6 Overview of the coding methodology that consists of six steps: (1) identifying data requirements, (2) coding data requirements by action keyword, (3) coding phrases by action-specific roles, (4) coding refinement statements to infer subsumption relationships, (5) mapping coded phrases into description logic formula, and (6) using a tool to compile the language and automatically check for conflicts

respectively. In general, tracing data flows allow an analyst to visualize dependencies between collection, use, retention, and transfer requirements. In this paper, we only formalize traces within a single policy and we discuss our case study results on tracing in Sect. 5.4.2. 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 manually performing cross-policy tracing.

4 Coding method to extract data requirements

Requirements analysts can extract privacy specifications by coding privacy policies using a six-step process (see Fig. 6). In step 1, the analyst systematically classifies each sentence as one of three types: *policy statements* 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;" *non-data requirements* describe the app, but are not primarily concerned with handling data, for example, "You will include your privacy policy URL in the App Dashboard;" finally, *data requirements* describe actions performed on data, such as "You must not include functionality that proxies, requests or collects Facebook usernames or passwords". In the remainder of this discussion, we employ a running example that assumes the analyst only considers data requirements for formalization using our approach.

In step 2, the analyst reviews each text-based data requirement and assigns one of the following four codes:

Collect	any act by a first party to access, collect, obtain, receive, or acquire data from another party;
Use	any act by a first party to use data in any way for their own purpose;
Retain	any act by a first part to retain data for a particular period of time or in a particular location; and
Transfer	any act by a first party to transfer, move, send, or relocate data to another party

The above list is not complete with respect to the types of data requirements that were described in the policies studied. For example, the policies also describe privacy notice practices and opportunities for consumers to control access to their information through consent and privacy settings. However, for the purposes of this study—to trace simple data flows within a single policy—the four actions codes above were deemed minimal.

In step 3, the analyst parameterizes the statement into six possible roles and their role values for each of the coded actions as follows: the first five roles apply to all actions, whereas the last role applies uniquely to transfers:

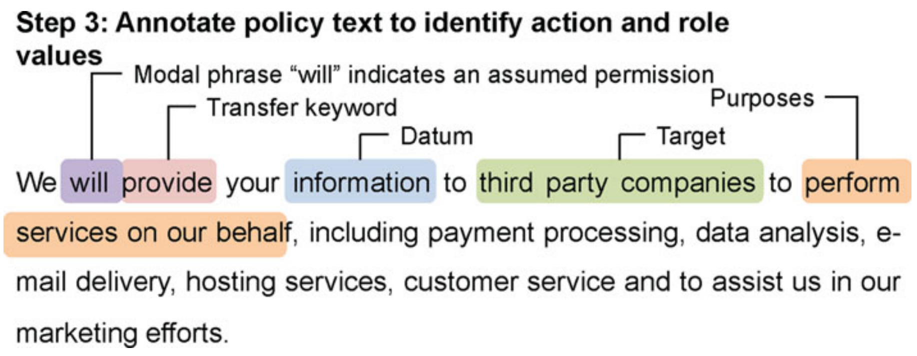

Step 3: Annotate policy text to identify action and role values
Modal phrase "will" indicates an assumed permission
Transfer keyword
Datum
Target
Purposes
We will provide your information to third party companies to perform services on our behalf, including payment processing, data analysis, e-mail delivery, hosting services, customer service and to assist us in our marketing efforts.

Fig. 7 Coded policy statement with the action, modality, and roles for the datum, target, and purpose of the transfer

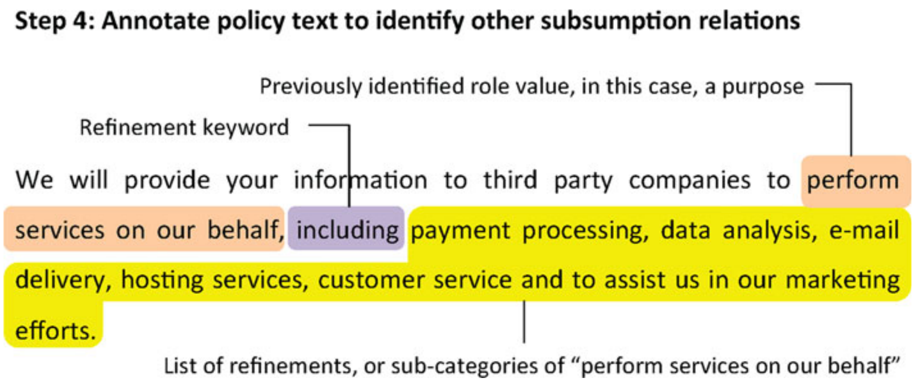
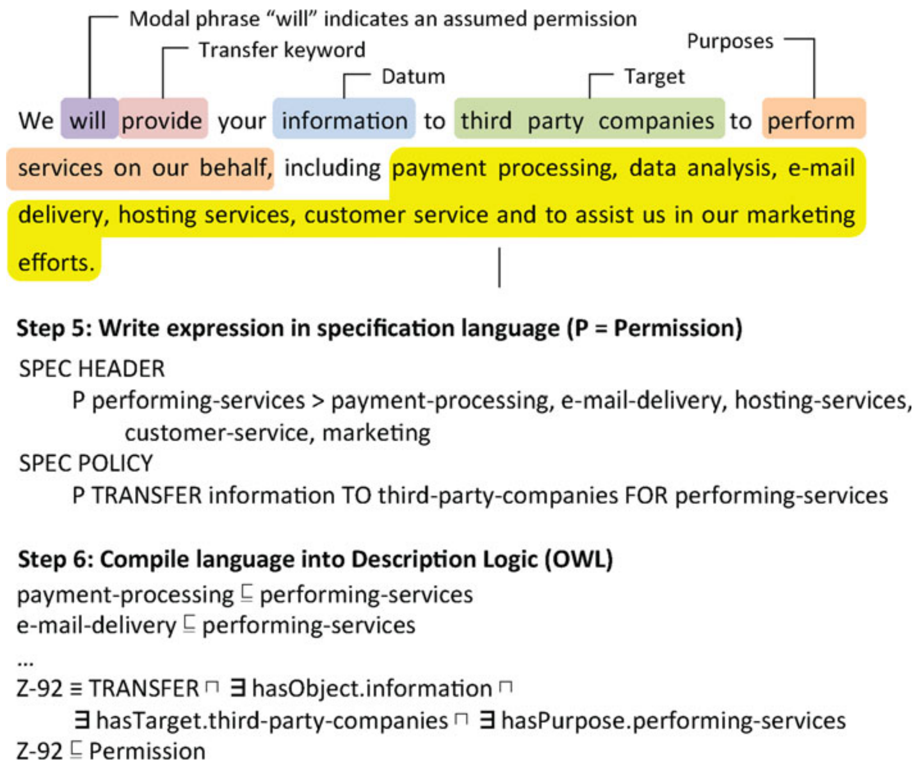

Step 4: Annotate policy text to identify other subsumption relations
Previously identified role value, in this case, a purpose
Refinement keyword
We will provide your information to third party companies to perform services on our behalf, including payment processing, data analysis, e-mail delivery, hosting services, customer service and to assist us in our marketing efforts.
List of refinements, or sub-categories of "perform services on our behalf"

Fig. 8 Steps to code subsumption relationships, in this case a role refinement that describes the kinds of services performed on Zynga's behalf

Modality	whether the action is a permission, obligation, or prohibition;
Subject	the actor who performs the action on the datum;
Datum	the information on which the action is performed;
Purpose	the purpose for which the action is performed; this role applies to any action, but is especially necessary to describe an information use
Source	the source from which the information is collected;
Target	for transfer actions, the recipient to whom the information is transferred

Consider an example data requirement that has been coded by the analyst in Fig. 7; this requirement appears in the Zynga privacy policy. The identifier Z-92 indicates this is the 92nd statement in Zynga policy. In step 2, the analyst identifies the action using phrase heuristics (verbs) that indicate which action should be assigned (e.g., "provide" indicates a *TRANSFER* action), the modality of permission is inferred from the modal keyword "will," the datum from "information," the target to whom the data are transferred from "third-party companies" and the purpose from "to perform services on our behalf…." In general, unless the statement is an express obligation or prohibition, we generally assume the statement refers to a permission. Role values may appear in comma-separate lists, which we interpret as logically disjoint. While this policy statement refers to "your information," it is unclear *from where* this information was collected. User data can be collected from the user, data brokers, or advertisers to name a few possibilities.

In step 4, the analyst uses three additional codes to extract subsumption relationships: *role abstractions*, which consist of one or more concepts that are more generic than a given role value (e.g., "information" is more generic than "a person's name"); *role refinements*, which consist of one or more concepts that are more specific than a given role value (e.g., "browser type" and "screen resolution" are specific kinds of "technical information"); and *role exclusions*, which consist of one or more concepts that are excluded from a given role value (e.g., "IP addresses" are excluded from what a company might consider "personal information"). Role abstractions, refinements, and exclusions are often used by policy writers to illustrate by example the types of information that are acted upon. In addition, requirements may tailored to particular types of information (e.g., permissions to share a user's birthdate are particularly sensitive because this information can be used to infer a person's age, versus a user's browser type, which is generally non-identifiable information because a browser type is one of a handful of common types used by large numbers of users). Entire sentences may be used to elaborate role values, or they may appear as clauses in the same sentence that describes a data practice. For example,



We will provide your information to third party companies to perform services on our behalf, including payment processing, data analysis, e-mail delivery, hosting services, customer service and to assist us in our marketing efforts.

Step 5: Write expression in specification language (P = Permission)

```
SPEC HEADER
    P performing-services > payment-processing, e-mail-delivery, hosting-services,
        customer-service, marketing
SPEC POLICY
    P TRANSFER information TO third-party-companies FOR performing-services
```

Step 6: Compile language into Description Logic (OWL)

payment-processing $\sqsubseteq$ performing-services
e-mail-delivery $\sqsubseteq$ performing-services
…
Z-92 $\equiv$ TRANSFER $\sqcap$ $\exists$ hasObject.information $\sqcap$
    $\exists$ hasTarget.third-party-companies $\sqcap$ $\exists$ hasPurpose.performing-services
Z-92 $\sqsubseteq$ Permission

Fig. 9 Steps to map data requirement from natural language to DL show the data requirement from the Zynga privacy policy, previously annotated; step 5 shows the requirement expressed in the Eddy language syntax, including the specification header consisting of definitions for kinds of purpose and the policy body consisting of data requirements; step 6 shows the header and policy body compiled as statements expressed in DL semantics

Fig. 8 illustrates an example of role refinement, in which the keyword "including" indicates that the purpose "perform services" will be refined by examples, such as payment processing, data analytics, and so on. In general, the phrase heuristics for identifying refinements were "including," "such as," and "for example," and abstractions were identified when the last item in a list was prefixed by the adjective "other," such as "other personal information" or "other marketing purposes".

Steps 3 and 4 are performed in parallel because policy sentences intermix requirement statements with phrases that indicate subsumption relationships, so the analyst may switch from coding requirements to refinements as needed.

Finally, in step 5 in Fig. 9, the analyst transfers the encoded values into the Eddy language that employs an SQL-like syntax and our DL semantics described in Sect. 3. The complete grammar for the Eddy language is included in the Appendix. Figure 9 illustrates this mapping beginning with the original annotated statement, the expression in the language syntax, and the corresponding DL formula that is generated by our compiler (step 6). Specifications in the language have two parts: the specification header ("SPEC HEADER"), which consists of definitions for terminology derived from role refinements, abstractions and exclusions, and the specification body ("SPEC POLICY"), which consists of the data requirements that describe the privacy policy. In the header, each statement begins with a letter that indicates the role filler category (e.g., "P" for purpose, "A" for actor, and "D" for datum), followed by a list of concepts separated by an operator (e.g., ">" for subsumes, "<" for is subsumed by, "=" for equivalent, and "\" for disjoint) that maps to a DL axiom type. In step 6, the definition for the purpose "performing-services" is separated into individual subsumption axioms for each kind of purpose by the compiler.

The specification body, or policy body, consists of data requirements. Data requirements begin with a letter that indicates the modality (e.g., "P" indicates permission, "O" indicates obligation, and "R" indicates prohibition or refrainment), followed by the action verb, the datum, and keywords to indicate the roles followed by the role filler concepts in Fig. 9: source ("FROM"), target ("TO"), and the purpose ("FOR"). When the Eddy expression is compiled in step 6, the resulting DL expression consists of two axioms: an equivalence axiom linking the requirement identifier to a conjunction of the action and role descriptions, and a subsumption axiom that indicates this requirement is a subclass of the specified modality, in this case a permission. Once the analyst translates the text into the Eddy 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++). We discuss our performance evaluation of these theorem provers in Sect. 6.

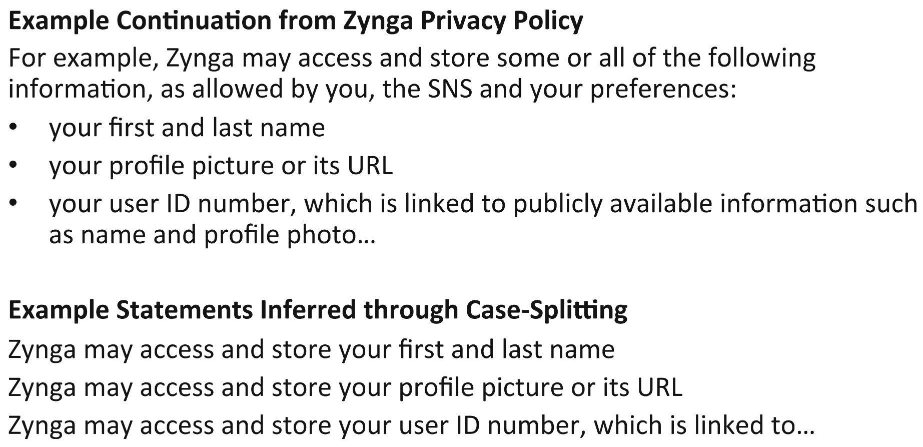
We now discuss the results of our case study replication to evaluate the language against three privacy policies that share a common data flow.

5 Case study results

The Eddy language was developed in a series of studies: a formative, exploratory case study to discover the language semantics using the Facebook platform policy; an extended case study aimed at further evaluating the efficacy of the language to generalize to other policy types, specifically privacy policies for Zynga and AOL Advertising; and finally, a replication of the extended case study. The formative and extended case studies were previously reported [13]. Herein, we report new results of the replication study. The three policies were originally selected because they represent a multi-tier data flow: Facebook provides the social networking platform that allows third-party app developers to reach Facebook users; third-party app developers, such as Zynga, provide new functionality, called social plugins, that enriches the platform with new user experiences; and AOL provides ads that, when clicked by users, generate revenue for Zynga. The replication study re-examined all three policies with two analysts (the first and second authors). The three policies that we analyzed are as follows:

- Facebook platform policy, last revised April 9, 2013, which governs app developer practices in Facebook (an earlier version of this policy was used in the formative and extended case study [13]).
- Zynga privacy policy, last updated September 30, 2011, which governs the user's privacy while they play Farmville and use other Zynga applications
- AOL Advertising, last updated May 4, 2011, which governs advertising distributed through Farmville and other Web sites and applications

The units of analysis consist of complete English sentences within the policy text. Sentences are typically indicated by punctuation, such as periods, but may consist of leading phrases followed by a colon and a bulleted list of examples, which we call *continuations*. Figure 10 presents an example continuation from the Zynga privacy policy: the continuation consists of a permission to access and store information, which is itemized in sub-bullets. Because these actions on information are independent, i.e., the Zynga may use the first and last name separately from the profile picture, the analyst extracts separate data requirements for each datum. This separation is called *case*



Example Continuation from Zynga Privacy Policy

For example, Zynga may access and store some or all of the following information, as allowed by you, the SNS and your preferences:

- your first and last name
- your profile picture or its URL
- your user ID number, which is linked to publicly available information such as name and profile photo…

Example Statements Inferred through Case-Splitting

Zynga may access and store your first and last name

Zynga may access and store your profile picture or its URL

Zynga may access and store your user ID number, which is linked to…

Fig. 10 Example from the AOL advertising privacy policy that illustrates two types of sentences that appeared in the data set: sentences with a continuation comprised of a *bulleted list*, and standard sentences

Table 1 Number of types of statements formalized in the replication study

Policy	Stmt.	Data	Modality			Action			
			P	O	R	C	U	R	T
Facebook	131	54	23	1	30	12	21	4	17
Zynga	190	76	73	0	3	36	7	13	20
AOL	75	35	32	0	3	14	11	2	8

Table 2 Phrase heuristics used to indicate when a statement was interpreted to be a collection, use, retention, or transfer requirement in the specification language

Specification	Equivalent	Disjoint	Subsumption
Facebook	1	0	10
Zynga	2	0	34
Zynga ∪ Facebook	3	0	21
AOL	4	0	12

splitting and has previously been described in the analysis of continuations in regulatory documents [11].

Two analysts (the first and second authors) replicated the formative case study originally conducted by Breaux and Rao [13] using the three privacy policies from Facebook, Zynga, and AOL. On average, the first analyst expended 1.09 min per statement extracting requirements, whereas the second analyst expended 2.21 min per statement. We attribute the larger time expenditure of the second analyst to the fact that the second analyst was learning the extraction method for the first time.

Table 1 presents the replication study results, including the total number of statements in the policies (Stmt.), the number of data requirements (data) that were formalized, which we break down into the number of permissions (P), obligations (O), and prohibitions (R), including which among these requirements concern the collection (C), use (U), retention (R), and transfer (T) of data. Between 30 and 51 % of statements in these policies described data requirements with generally few to no obligations. The Zynga and AOL policies describe their own data practices and focus more on permissible practices, whereas the Facebook policy describes the practices of developers (in this case, Zynga) and focuses slightly more on prohibitions (what developers must not do). In Table 1, the Zynga policy has relatively fewer descriptions of what data are used than compared to collections and transfers. As we show later in Sect. 5.4.2, the Zynga policy is written in a particular style that exhibits multiple underflows (instances where narrowly defined collections feed into only a few broadly defined uses) that may increase developer flexibility under this policy.

In Step 5 of our formalization of the three policies, we made inferences to align terminology in the same specification. These inferences are due to ambiguities in the policy text, such as using different terms that seemingly refer to the same concept (e.g., “data broker” and “information broker”). We formalized these inferences in the specification header and present the resulting frequencies in Table 2: the number of equivalent, disjoint, and subsumption relationships. Equivalent terms are close synonyms, whereas subsumption relationships denote terms that are exemplified by other terms (e.g., “device id” includes “MAC address”) and composite information types that include various parts (e.g., “name” includes “first name” and “last name”). In Sect. 5.1, we discuss our conflict analysis results from comparing the Zynga and Facebook specifications and in that process, we aligned the terminologies from the two specifications to yield an additional number of terminological inferences, shown in Table 2 as Zynga ∪ Facebook. With respect to conflict detection, inferences made to disambiguate the policies are often necessary to reveal conflicts. Thus, it is important that the policy owners participate in the disambiguation process to ensure that potential conflicts detected by this analysis are indeed real conflicts.

During the replication study, we traced all keywords and phrases that indicated when an action was to be classified as a collection, use, retention, or transfer; these appear in Table 3, which shows the total number of different verbs coded for each category and examples of the most common verbs. Among these keywords, a few words and phrases are not exclusively used to signal only one data practice (e.g., the verbs "access," "use, and" "share"), while other words were discovered to be more exclusive (e.g., the verbs "collect," "disclose, and" "transfer"). For example, consider the following three statements from the Zynga and AOL privacy policies:

1. *Collection*: "[We] may *access* and store some or all of the following information, as allowed by you, the SNS, and your preferences" (Zynga)
2. *Usage*: "Personal information such as name, address, and phone number is never *accessed* for this purpose" (AOL)
3. *Transfer*: "In that case, the acquiring (or merging) company will have *access* to your information" (AOL)

In each of these three statements, the main verb is "access" and statement's context determines how to disambiguate the meaning and categorize the data practice. The first statement describes a collection, which the analyst infers from the source from which the data are collected (e.g., the SNS). The second statement describes a usage, because this statement appears in a section describing AOL's uses of personal information. Finally, the third statement describes a transfer, because a third party (the acquiring company) is accessing the personal information. Recording which verbs correspond to which data actions in the formalism enables the analyst to identify verbs that are more or less ambiguous and to check for coding for errors (e.g., to check the statement context of each "access" and check which actor is initiating the action).

Table 3 Phrase heuristics used to indicate when a statement was interpreted to be a collection, use, retention, or transfer requirement in the specification language

DL action	Different phrases	Most common action keywords and phrases
Collect	31	Access, asking, assign, collect, collected, collection, collects, create, enter, gathered, import, obtained, observes, organizes, provide, provided, providing, receive, receives, request, requests, share, to know, understand
Use	12	Accessed, include, integrates, monitored, processed, processing, see, use, used, uses, utilized
Retain	13	Cache, delete, erase, keep, removed, retain, retained, retains, retention, store, stored, stores, storing
Transfer	23	Communicate, disclose, disclosed, disclosure, is visible to, provide, receive, reveal, see, sell, send, share, shared, shares, transfer, transferred, updated, use, utilized, view, will be public

When designing our language, we preferred to use the verbs that were exclusively associated with the intended action keywords in our language syntax (e.g., the verb "collect" was exclusively used for collections, whereas "access" was not). One reason for why some verbs are non-exclusive and thus ambiguous is that policies can shift among multiple viewpoints in the text. In our coding of privacy policies, we intentionally shift the policy viewpoints to the application point of view to align the privacy policies with a key goal of our specification language, which is to describe a single developer's viewpoint, as opposed to describing the world of cross-application data flows in one specification. We believe this viewpoint is natural for developers and shifts the burden of aligning requirements to the system interfaces, which is consistent with the approach used in software architecture. To reconcile these viewpoints in privacy policies, the analyst should identify the viewpoint used by the policy and re-topicalize relevant statements to yield the correct formalization from the viewpoint of the application. By identifying which keywords and phrases are ambiguous, the analyst can further conduct a second pass over coded statements that use those ambiguous verbs to check that errors were not made during formalization. We discuss specific cases that can result in inter-rater disagreement and missing codes in Sect. 5.1, below.

We noted several differences in the results from our exploratory study by Breaux and Rao [13] and the replication study reported in this paper. Notably, the replication study was conducted on a more recent version of the Facebook platform policy: the new policy includes seven new guidelines concerning how to integrate advertising in Facebook apps; however, these new statements did not add to the total number of data requirements formalized. In addition, the replication study surfaced more permissions and fewer refrainments in the Zynga policy than the exploratory study. The decrease in refrainments and increase in permissions were partly due to a change in assumptions underlying how the analysts interprets consumer consent (e.g., the ability to opt-in, opt-out, or control data collections using the Facebook platform privacy settings). In the exploratory study, multiple collection, use, and transfer statements were conditioned upon the receipt by Zynga of consent from the user to conduct these activities. In that first study, we assumed that opting-in

Table 4 For the 2-rater replication study, the number of missing codes, disagreements, and inter-rater reliability computed using Cohen's Kappa and Krippendorf's Alpha; and the Alpha computed for all three raters

Policy	Missing codes	Disagreements	Cohen's kappa	Krippendorf's alpha	
				Replication (2 raters)	All 3 raters
Facebook	5	3/54	0.884	0.941	0.937
Zynga	13	4/76	0.919	0.922	0.906
AOL	5	5/35	0.800	0.828	0.849

yields a default modality of refrainment, i.e., the formalization expresses the pre-consent view of the world in which Zynga is prohibited from acting until the user opts in. In the replication study, we assumed the post-consent view of the world, i.e., all consents have been received to maximize information sharing. Requirements analysts should consider both worlds in their data flow designs. The remaining increase in the number of collections, uses, and transfers are due to an increased use of case splitting to separate a single natural language sentence into multiple statements and to separately code single sentences into multiple formal statements in the language.

We now discuss findings from our formal analysis that includes sources of analyst disagreement during the statement classification steps, as well as, policy conflicts found with the formalism and opportunities to extend our approach, which includes the limitations of our current work.

5.1 Sources of disagreement and inter-rater reliability

During steps 3 and 4 of the process that consists of annotating the policy text, two coders (the first and second authors) identified sentences and phrases that map to collection, use, retention, and transfer actions. We measured inter-rater reliability for classifying only the action verbs using two statistics that account for chance agreement: Cohen's Kappa [16], which is predominantly used by social scientists, and Krippendorf's Alpha [24], which is predominantly used by content analysts. Cohen's Kappa statistic is more popular among social scientists and is routinely used in experimental measures in which two coders map units into exclusive categories, i.e., no unit can be mapped to two or more categories by the same coder and every unit must be mapped to a category by both coders. Krippendorf's Alpha statistic supports any number of coders and allows for omissions or units in which one coder does not assign the unit to a category. In Table 4, we report both statistics: *Missing Codes* refers to the number of units that were coded by only one coder, which were not used to compute Cohen's Kappa but were factored into Krippendorf's Alpha; and *Disagreements* reports the number of units where the coders disagreed out of the total number of units.

We did not evaluate whether missing codes were overlooked or were intentionally excluded by one of the analysts. In general, we aim to achieve a Kappa and Alpha statistic above 0.6, which we infer to mean a moderate agreement above chance, and 0.8, which we infer to mean a high agreement above what is expected by chance alone. Krippendorf's Alpha appears larger than Cohen's Kappa, because Alpha accounts for missing codes, wherein a missing code is not necessarily the characteristic of disagreement. We also aligned and computed Krippendorf's Alpha for the combined results of the exploratory and replication study to yield an Alpha measure between 0.849 and 0.937 across all three raters, which we infer to be a high level of agreement. Finally, we inspected the sentences where the coders disagreed to characterize the source of disagreement. We identified two broadly construed heuristics to explain the different sources of missing codes and disagreement, which we now discuss: multiple actor viewpoints and implied data practices.

5.1.1 Multiple actor viewpoints

Among the disagreements, we discovered that most disagreements, including five AOL, four Zynga, and one Facebook disagreements, were due to differences in determining which actor viewpoint should be used to assign the code. For example, consider the following statement that was separately assigned the codes *Transfer* and *Collect* by two different raters:

> Network Participants Sites may share certain non-personally identifiable information about their users (such as age, gender, and zip code) with AOL Advertising

The verb "share" indicates a *transfer* from network participant sites, which is one actor, to AOL Advertising, which is a second actor. Conversely, an analyst could view this verb as indicating a *collection* by re-topicalizing the statement in their memory, e.g., "AOL Advertising collects certain non-personally identifiable information from Network Participant Sites." Recall that each privacy requirements specification is written from one actor's viewpoint, typically the viewpoint of the developer. This design choice for our approach reflects that analysts and developers likely know how they plan to use personal information that they need, but generally have limited knowledge about how others use this information. Thus, when privacy policy authors write about third-party practices, analysts who use our approach must re-topicalize those statements to reflect the viewpoint of the target system. In this case, we believe improving our heuristics with this critical guidance helps analysts to avoid these disagreements or to promptly reconcile these differences to reach agreement.

In addition to disagreements, when a statement contains multiple viewpoints, analysts may not always map the other viewpoint to that of the target system, which can lead to missing codes that the other analyst overlooks because they strictly focus only on the dominant viewpoint. In the statement below, two data flows are described: the first flow as the user ("you") provides payment information to a third-party processor, and second flow as Zynga receives that payment information from the third-party processor, described in reverse order here:

> Zynga may also receive the billing and payment information that you provide when your purchase is processed by another party, such as Facebook (for Facebook Credits) or Apple (for purchases on iOS devices)

In this case, both analysts assigned the *Collect* code for the second flow (the receipt by Zynga), and one analyst also assigned a transfer code for the first flow (the provision by the user). Both analysts ultimately agree with these assignments, noting that the first flow describes the user's viewpoint (a transfer of data from the user to the third party), and thus these missing codes can be justified using the viewpoint heuristic. In practice, developers may wish to maintain partial specifications of third-party practices to account for these different viewpoints. In future work, we aim to verify these partial specifications as a part of cross-flow data requirements validation. For this study, however, we only observed four missing codes in Zynga's policy and two missing codes in Facebook's policy that resulted from a second action or viewpoint being overlooked by an analyst. For policies that describe data practices of platform users, the analyst may expect to see a larger number of multi-viewpoint statements, as these policies tend to describe the provider's platform as well as the developer's application.

5.1.2 Implied collections, uses, and transfers

Policy statements may describe implied data practices. This includes statements about privacy notices and privacy choices. Statements about privacy notices describe acts to inform users about data practices and may include reference to those practices that could be considered for coding. For example, the following AOL statement contains verbs typically associated with these actions, such as the verb "inform" to indicate a notice, and the verb "share" to indicate a transfer:

> AOL Advertising requires Network Participant Sites to inform their users about the information they share with third party networks such as AOL Advertising

For the purpose of our case study, however, we chose to exclude these statements from the specification-writing Step 5 (above), since these statements do not substantially describe a new collection, use or transfer.

Alternatively, policy statements describe privacy choices that affect data practices. This includes statements about a users right to opt-out or opt-into a data practice, or their ability to configure their privacy settings to control how data are used. For example, the following AOL statement describes the implication of opting-out of a particular usage and the purpose for which the data would otherwise be used, if the user chooses not to opt-out:

> "If you opt-out, you will continue to receive ads from AOL Advertising; however, it will prevent us from delivering ads tailored to your preferences and usage and may prevent us from controlling the frequency with which you may view particular advertisements."

In general, when specific purposes were described, we chose to formalize these uses in Step 5 in our process, assuming the user had opted-in or configured their privacy settings to maximize the use and sharing of their information.

Other implied data collections and transfers also include account registration, which requires collecting personal information, and making personal information public or visible to others. These statements account for three missing codes from the Zynga policy. In some cases, the indicative verbs are near synonyms of collection and transfer, such as "importing" or "exporting" data. In other cases, however, the action indirectly implies these activities, such as "registering" for an account, in which the registration process necessarily implies the collection of an e-mail address, person's name, birthdate, and other personally identifiable information. The extent to which these implications are made explicit within the policy, or to which the analyst uses other sources of information (e.g., Web pages from the registration process), will determine whether the analyst can justify coding these complex actions in accordance with those implicated data practices. Otherwise, these complex actions will more likely be missed in the case of vague or ambiguous privacy policies.

Finally, privacy policies occasionally include co-reference, including anaphora that refers to previously described practices. For example, the Zynga policy describes several data collections, after which the policy refers to the "information collected" in the context of additional rights and obligations. This partial phrase coding resulted in three missing codes from Zynga's policy and two from Facebook's policy. If the code mapped to a previously coded practice, then the coded practice was omitted to avoid duplication; otherwise, the code was preserved. For example, in Zynga, the statement "If you have provided your e-mail address to Zynga, We'll use it to respond to…" describes an act to collect the user's e-mail address and subsequently the purpose for which it will be used, which results in a *Collect* and *Use* code both being preserved in the formalization.

Finally, implied actions frequently result from axioms of logical implication: for any act to use, retain, or transfer personal information, we may assume that this information was previously collected. On occasion, the analyst may perform this inference and produce a code; however, we recommend strictly adhering to the text when extracting formal specifications of data flows and later using the axioms of inference to infer these implied collections. The formally inferred implications can then be used to verify that the policy states all relevant or implied collections, or to infer ambiguities in the policy when the collection was not described in the policy text.

5.2 Conflicts identified using the Eddy language

We found multiple conflicts within the Facebook, Zynga, and AOL policies using our DL formalization and our automated toolset. Table 5 presents an overview of our automated conflict detection results: we checked for conflicts within each specification and between the Zynga and Facebook specifications, because Facebook's policy describes permitted and prohibited practices covering Zynga. In Table 5, we report the total number of conflicts between unique pairs of permitted and prohibited actions, the time in seconds to compute the total number of conflicts, and the number of conflicting interpretations out of the total interpretations analyzed. When we combined the Zynga and Facebook specifications to detect conflicts between these two policies, we only retained the permissions from Zynga and prohibitions from Facebook; thus, we only report the 8 new conflicts discovered between the two policies, excluding the 19 and 10 other conflicts found within each specification, respectively.

Table 5 Results from performing automated conflict identification using DL formalization of privacy requirements specifications

Specification	Number of conflicts found	Compute time conflicts (s)	Proportion of conflicting interpretations
Facebook	10	166	33/9,116
Zynga	19	1,138	1,001/2,584
Zynga ∪ Facebook	8	3,689	164/133,558
AOL	0	30	0/1,827

The automated conflict analysis procedure is based on a MapReduce-style algorithm [17] that decomposes the inference task into independent units, each containing a subset of the total interpretations that potentially conflict. These units are then distributed to available processes running on a multicore processor, or potentially to processes on other computers in a cluster, and the final results are later combined into a composite report. For our study, we used a single 2.66 GHz Intel Core i7 MacBook Pro with 8 GB 1,067 MHz DDR3 memory. As we discuss in Sect. 6, the automated conflict analysis has runtime complexity that appears linear in the number of conflicts and polynomial in the size of the interpretation. To reduce runtime, we only compute interpretations for prohibitions, because the number of prohibitions is generally less than the number of permissions. For the Zynga and Facebook analysis, we compared the permitted actions in the Zynga specification with the prohibited actions in the Facebook specification. We envision several optimizations that can further reduce this runtime.

In this study, we only examined conflicts between actors who share data directly with each other. The conflicts reported here were first found in the exploratory study and confirmed in the replication study using our toolset. In future work, we plan to examine conflicts across transitive data flows.

5.2.1 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 (see Table 2). Using our formalization and reasoner, 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 [35]. 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 the 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, which also depends on inferring that 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 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
```

Transferring the information to the successor organization without restriction puts the information at risk of repurposing and onward transfer to third parties.

5.2.2 *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 in the Eddy language are expressed as follows, where AOL-27 describes the prohibition and AOL-46 describes the permission:

```
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
```

While this conflict was identified manually in our study, it raises an important concern for analysts: do data collections imply uses or transfers? In future work, we prose to study this question and the underlying ambiguity by examining alternative interpretations to identify potential conflicts.

5.3 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, retention, and transfer as defined in Definition 1. The un-encoded requirements include how data are merged and the policy implications of consent. With respect to retention, there are additional constraints that we aim to formalize in future work, such as the physical location and duration of retention as well as the implication of not storing data on other data practices. We now discuss these three categories of requirement.

5.3.1 Merging data from different sources

The three policies in our first study yielded 12 requirements that describe how data are 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 are linked, combined, or aggregated with different implications. Linking data enable 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 required by the business in developing new services.

5.3.2 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 are stored, it exists to be acted upon for the duration of storage; when data are 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 concept subsumption (e.g., is one type of information subsumed by another type), different temporal logics exist to reasoning about time. We are looking into extensions to DL for temporal reasoning [28] that can be used to express these remaining privacy requirements.

5.3.3 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. In our case studies, we explored the impact of coding the pre-consent and post-consent view of data flow, in which certain practices default to prohibited and change to permitted and vice versa. We plan to further explore how to reason about consent in future work.

5.4 Challenges due to formats and writing styles

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

5.4.1 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 the third-party conduit.com linked to by Zynga, or to additional data definitions or specific examples of data requirements as exhibited in Facebook's platform policy. Other links, such as "contact us" in AOL's policy and "change e-mail preferences" in Zynga's policy, do not lead to additional data requirements. Due to the large number of links that may arise across multiple Web sites, this problem suggests a need for additional automation using natural language processing techniques to identify relevant policies.

5.4.2 Separate collection, use, and sharing sections

A policy may describe data collection, the purposes for collection, and data sharing requirements in different sections. At first glance, this format makes extracting formal specifications easier, because each statement is relatively independent. However, the format can decouple the collection requirements from use and transfer requirements through the use of ambiguity (e.g., using different terms or omitting sources, targets, and purposes needed to realize a complete end-to-end data flow). For example, the Zynga privacy policy describes the information types collected (see "Information We Collect") separately from the purposes for using information (see "How We Use the Information We Collect"). This separation yields a logically inferred many-to-many mapping between information types and purposes, because the analyst must reasonably assume that any data type maps to any purpose within the entailment of subsumption. In Fig. 11, we present the data flow tracing from the first study 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 data requirements have an observably smaller valiancy or edge count. Each node is labeled with the agent number and a separate requirement number, e.g., agent1p2 is permission number two for agent one, who is Zynga in this figure. The green nodes represent permissions, and the red nodes represent prohibitions. 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 ways or ways that are unforeseeable by examining any one statement in the policy.

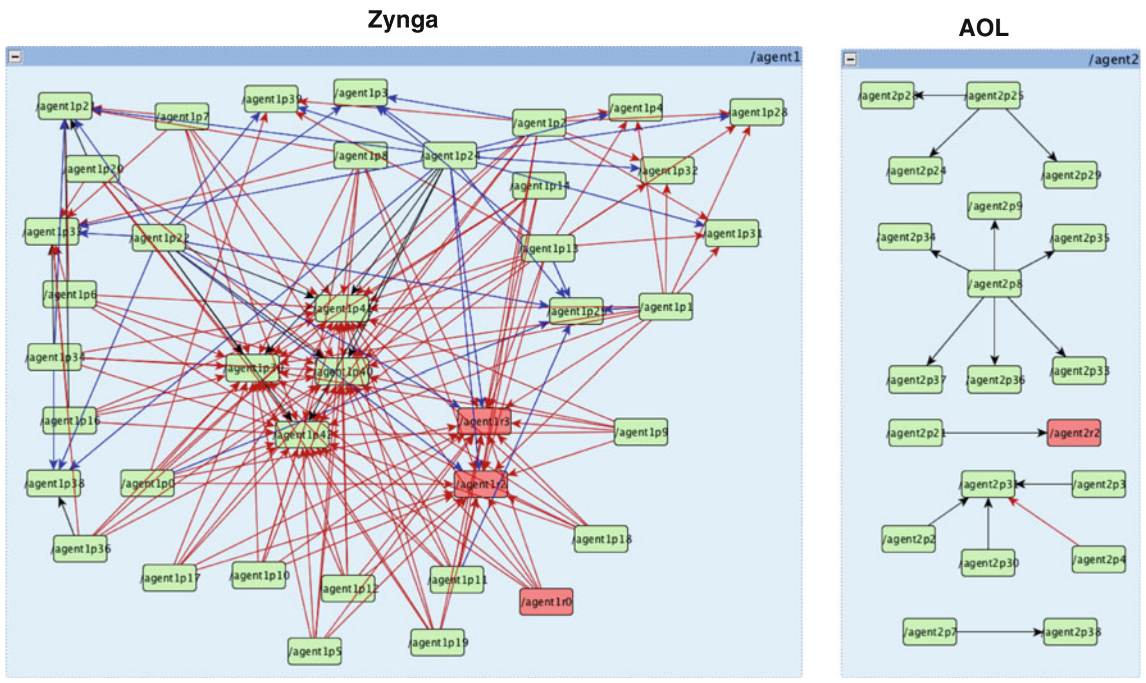


Fig. 11 Data flow traces inferred from the Zynga policy (*left*) and AOL policy (*right*): *arrows* point from collections to transfers, *red lines* show underflows, *blue lines* show overflows and *black lines* show exact flows (see Definition 5), *red nodes* show prohibitions and *green nodes* show permissions. The Zynga policy defines broad transfer rights as seen by the four central *green nodes* with multiple incoming *arrows* (color figure online)

5.4.3 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.

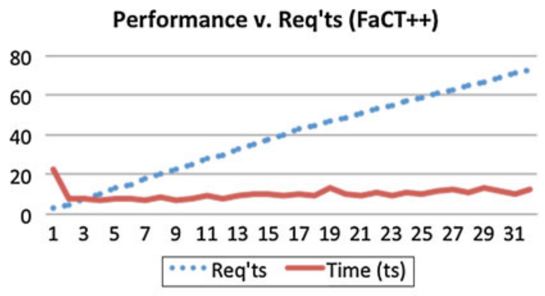
6 Simulation results

We conducted a performance simulation to evaluate the computational practicality of using our language to reason about data requirements. While we discovered several conflicts in our case study, the case study was chosen because it demonstrates a common case at the time of this study (i.e., many users interact with the Farmville game and Facebook). Our simulation aims to examine a larger population of specifications than we examined in our case study with respect to two variables that are specific to our approach: the number of conflicts and the number of requirements. In our simulation, we hold the number of concepts and individuals constant; however, these two other variables would likely affect the performance of any DL theorem prover. Furthermore, while we reduce conflict detection to DL satisfiability, which is PSPACE-complete for a cyclic TBoxes and the DL family ALC in which we express our language, we recognize that this bound does not ensure that our language is practical for reasonable size specifications. Therefore, we implemented a prototype parser and compiler for our language using three popular theorem provers: the Pellet OWL2 Reasoner version 2.3.0 from Clark and Parsia; the Fact++ Reasoner version 1.5.2 from Tsarkov and Horrocks, and the HermiT Reasoner version 1.3.4 by the Knowledge Representation and Reasoning Group at the University of Oxford.

We randomly generated 32 privacy requirements specifications with actor, datum, and purpose hierarchies comprised of binary trees with 23 concepts; this yields specifications with up to 1,280 itemized interpretations. Each specification had a variable number of requirements from three to 73 and conflicts from zero to 50. We conducted several preliminary runs and determined that concept tree height had no observable effect on performance at this scale. We ran each generated specification separately, only one time, and avoided resource consumption by processes external to the simulation. Of the three reasoners, the Pellet Reasoner did not respond within 30 min when realizing a policy of only four requirements. Thus, we only discuss results from the Fact++ and HermiT reasoners.

Figure 12 presents the performance time of the Fact++ and HermiT reasoners with respect to the specification size or number of requirements: the 32 runs are sorted along the *x*-axis from the fewest to the most requirements (from 3 to 73); the *y*-axis describes the response time in tenths of a second (solid red) and number of requirements (dotted blue). As the requirements increase to 73, we see that Fact++ response time remains constant, whereas the HermiT response times appear to increase slightly (Pearson's $R = 0.533$). To understand this increase, we present Fig. 13 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 (solid red) and the number of conflicts (dotted blue).

Figure 13 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 theorem 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 [26], which we rely on in our



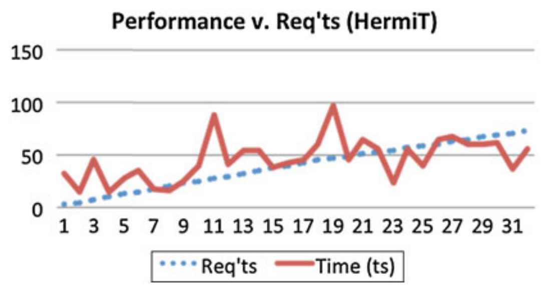


Fig. 12 Performance time of Fact++ and HermiT reasoners on privacy requirements specifications with respect to the number of requirements

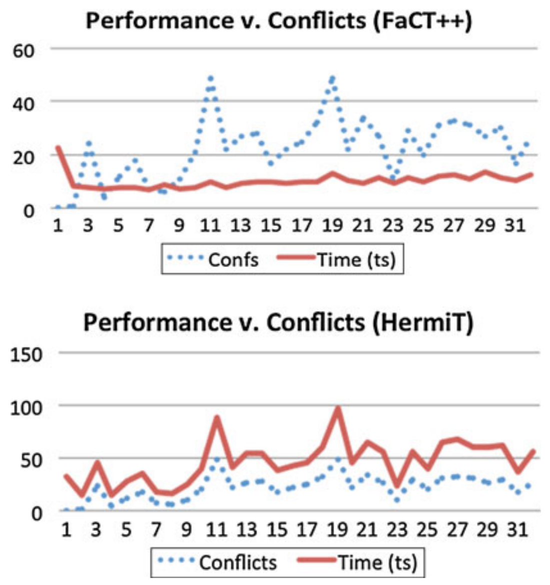


Fig. 13 Performance time of Fact++ and HermiT reasoners on privacy requirements specifications with respect to *number* of conflicts

formalism; however, Fact++ and HermiT are not limited in this way. From this simulation, we believe the Eddy language is computationally practical for policies within the order of 100 requirements, which includes the specifications in our case study; however, we see opportunities to further optimize our approach to analysis and we need to do more work on usable interfaces to the logic.

7 Threats to validity

Here, we discuss the generalizability of our mapping methodology. To address construct validity, we maintained a project workbook that contains the source text and annotations and mappings of natural language statements to our Eddy language syntax and notes about shortfalls and boundary cases in our interpretation. The source text and annotations for the replication study were collected using the TAMS Analyzer, a free tool for coding multiple data formats. The shortfalls were reported in Sect. 5.3 as limitations of our approach.

Construct validity reflects whether the construct we propose to measure is indeed what we measured [39]. While mapping statements to our formalism, we use heuristics to infer that a particular statement corresponds to an action in our formalism. These heuristics may require additional context outside a given statement to identify the action, source, target, and purposes. As discussed in Sect. 4, we need to resolve ambiguity in the phrase-to-action mappings; for example, does the word "access" indicate a collection, use, or transfer? Furthermore, as discussed in Sect. 5.4, we found that given purposes might be described using different grammatical styles. Lastly, we had to infer subsumption relationships between extracted terms to build our hierarchies for datum, purpose, and action when they were not explicitly stated. To address this threat to validity, we aim to further study how analysts identify this context, what is the variability among analysts and what demographic factors of analyst expertise correlate with better performance for resolving ambiguities and achieving agreement.

Internal validity is the extent to which observed causal relationships exist within the data and, particularly, whether the investigator's inferences about the data are valid [39]. A concern related to documenting analyst interpretations arises when we align the policy lexicons to compare formalized statements from two different policies and infer the presence of conflicts. This alignment 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. We plan to conduct human subject studies and expert surveys 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. Input from expert surveys and interviews, e.g., with legal scholars and privacy officers, can help us understand the feasibility of resolving different interpretations. We plan to study the effect of user workload and human resource requirements on the usability of our mapping methodology. In addition to estimating the time required for mapping, these studies will also evaluate human effort required to deal with the challenges posed by our methodology, for example, resolving ambiguities and inferring subsumption hierarchies.

External validity is the extent to which our approach generalizes [39]. We observed multiple styles of policy construction, as shown in Fig. 11, wherein policies may describe their data practices at varying levels of detail. These styles and others we have yet to encounter may limit the efficacy of analysis techniques, particularly is the policy writers take an adversarial perspective to conceal, obfuscate, or thwart interpretation. Furthermore, there are data practice descriptions in privacy policies that we are not presently accounting for, such as user consent, data storage and deletion, and aggregation statements. Therefore, we plan to conduct additional studies of more policies to evaluate the generalizability of the language and to extend the language to account for these other practices.

Table 6 Comparison of languages used to express different parts of privacy policies: generic languages may be used to express any rule-based statement, whereas others are tailored to a domain; most languages express permissions, obligations, and refrainments (prohibitions), for two languages obligations only follow from certain permissions ([O]) and a few languages express sanctions (+S) and exclusions (+E)

Language	Domain	Prescriptions	Semantics
P3P	Web privacy	–	Declarative
EPAL	Access control	P[O]R	Declarative
XACML	Access control	P[O]R	Declarative
UCON_{ABC}	Usage control	POR	Declarative
AIR	Web rules	–	RDF
Rei	Generic	POR	OWL-full
KAoS policy	Generic	POR + EO	OWL-full
Ponder	Networks	POR + S	Event calculus
ExPDT	Data privacy	POR + S	OWL-DL
Eddy	Data privacy	POR + E	OWL-DL

8 Related work

We now discuss related work in requirements engineering (RE) and formal methods. In RE, Antón et al. [3, 4] analyzed over 40 privacy policies using goal mining, which is a method to extract goals from texts. 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. [8] later extended this representation with notions of rights, obligations and permissions in a case study, and then formalized this extension in DL [10]. More recently, Young introduced a method for mining commitments, privileges, and rights from privacy policies to identify requirements [41]. 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 [38]. 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-formal methods have long been applied to privacy policy and privacy law as an application area. Privacy policy frequently refers to (1) consumer privacy policies, which may be electronic [15] and which have been shown to be misaligned with business practices and software systems [12]; (2) enterprise privacy policies, which govern an organization's internal business practices in relation to privacy; and (3) access control policies that implement a subset of enterprise privacy policy governing access to personal information. We now discuss these three categories of privacy policy, distinguishing access control policies separately as authorization languages, semantic Web frameworks, and formal privacy models and frameworks. While these methods and languages are not intended to specify requirements, they effectively do express requirements-related phenomena from high-level goals to functional specifications that relate directly to our approach. We summarize these related languages in comparison with our approach in Table 6.

8.1 The platform for privacy preferences

The platform for privacy preferences (P3P) is a declarative XML-based language for describing the privacy practices of Web sites to consumers and allowing consumers to state their preferences [15]. P3P policies were not originally intended to govern the exchange of data between intermediaries and third parties, such as telecommunication providers and law enforcement. Instead, P3P policies describe the content of Web site directories, including images and online forms, and focus on actions specific to a Web site and the user agent (Web browsers) to check whether a policy conforms to a user's stated preferences. Separate P3P policies can be created for separate data collectors, such as a third-party ad network that integrates targeted advertising content into Web pages. In describing data collection requirements, P3P policies distinguish between multiple roles, including what data are collected, for whom and for what purpose. Retention requirements in P3P are categorical (no retention, stated purpose, legal requirement, business practices, and indefinite). In P3P, data and purposes may be expressed in classification hierarchies. While P3P lacks a formal semantics [40], which may have led to language misuse [25], we believe the data requirement descriptions are sufficiently expressive to be coordinated with enterprise policies expressed in our data model and that they could be used to generate statements in the Eddy language.

8.2 General authorization languages

General authorization languages include declarative access control languages, which support stating privacy properties over data. The platform for enterprise privacy practices (E-P3P) and Enterprise Policy Authorization Language (EPAL) were developed by IBM to write enterprise privacy policies that govern data handling for internal business practices [2, 32]. E-P3P and EPAL both express policies as allow/deny rules over a set of terms, including data categories, data users, and purposes, each organized into classification hierarchies.

The eXtensible Access Control Markup Language (XACML) is an open standard to express general access control rules [29]. The differences between EPAL and XACML are subtle [1]: both use international standards for their policy enforcement architecture; both support hierarchical data categories, data users, and data purposes with minor discrepancies; both include obligations, which are post-conditions that must be fulfilled once an authorization is granted. When there exists documented limitations in one or the other, there are obvious work-around solutions [1].

Usage Control with Authorizations, oBligations, and Conditions ($\mathrm{UCON}_{\mathrm{ABC}}$) is a model for expressing usage rules for general objects, which Park and Sandhu differentiate from other access control languages in terms of obligations [33], which are concurrent or post-conditions on authorizations similar to EPAL and XACML. Envisioned applications for $\mathrm{UCON}_{\mathrm{ABC}}$ include privacy, intellectual property rights, and access control.

Obligations in E-P3P, EPAL, XACML, and $\mathrm{UCON}_{\mathrm{ABC}}$ are functional externalities to the language: they describe actions to be taken by the environment within which the interpreter or model executes its rules. Breaux and Antón found that many pre- and post-conditions in data access requirements are not decidable by software because they require human input and, once externally decided, can even be operationalized by software design across multiple instances of the same transaction [9]. Reuse across transactions precludes the need to express and evaluate these conditions in access control rules during runtime and rather requires they exist as first-class software requirements to be implemented by designers. In contrast, the notion of obligation may also be used to distinguish between discretionary requirements (permitted actions) and mandatory requirements (or obligatory actions) [11]. In Eddy, we employ obligations for this purpose to distinguish when a data action, such as a transfer to law enforcement, may be required.

The existence of arbitrary attributes in E-P3P, EPAL, XACML, and $\mathrm{UCON}_{\mathrm{ABC}}$ is evidence of under-specification in the languages and model, which provides robustness to express unforeseen situations for solving new problems; however, this bears the disadvantage that the language constructs can be misappropriated in unforeseen ways. Moreover, E-P3P, EPAL, and XACML are declarative languages, which lack a formal semantics with a complexity bound for evaluating policies. Functions for comparing these languages are delegated to abstract architectures. In our approach, we show that policy compliance can be reduced to DL subsumption, which is PSpace-complete, when reasoning over hierarchies of actors, data, and purposes.

Finally, the Ponder language, introduced by Lupu et al. [27], was developed for the specification of network management and security policies for distributed systems. Ponder supports expressions for permissions, obligations, and delegation of rights; however, it lacks explicit support for expressing role values, such as purposes, that are required to reason about repurposing of data.

8.3 Semantic Web policy frameworks

The Web Ontology Language (OWL) and its predecessors, DAML and DAML + OIL, are used to express classification hierarchies with data type constraints for the Semantic Web. The Semantic Web is a vision that seeks data interoperability by standardizing data vocabularies. Select variants of OWL have a formal semantics in a decidable family of DL, which is a fragment of first-order logic [6].

KAoS is a framework that contains an OWL policy ontology, which consists of authorizations (rights and prohibitions) and obligations, including exclusions (*is not required to*) [37]. Similar to the KAoS policy ontology, the OWL policy language Rei provides classes for expressing rights, prohibitions, and obligations. Neither KAoS nor the Rei ontology languages have axiomatic encodings for detecting conflicts between rights and prohibitions, nor for inferring rights from obligations. Rather, to resolve modality conflicts, KAoS relies on a special priority-based algorithm [14] and Rei employs an RDF-S policy engine [36]. ExPDT, another OWL-based policy language [23], focuses instead on conflict resolution via runtime monitoring. Our approach extends this prior work with a formalization and evaluation of conflict detection in the subset of OWL that maps to DL.

8.4 Privacy models and frameworks

Privacy models and frameworks tend to focus more specifically on data transfer and the purpose for which data are used. This includes contextual integrity, proposed by Helen Nissenbaum [31] and formalized by Barth et al. [7], the PrivacyLFP, which is based on contextual integrity [18], privacy APIs by May et al. [30], and the data purpose algebra proposed by Hanson et al. [19].

Contextual integrity is a conceptual framework for expressing personal information transmission rules as positive and negative norms [7], which loosely correspond to conditional rights and prohibitions, respectively.

Transmission rules are written over the domain of actors and attributes, which are mapped to role hierarchies and data, respectively. Attributes can be used to express norms about data sets with implications on the data elements contained in those sets. The norms are used to express transmissions of attributes in messages between actors using linear temporal logic (LTL), which allows the construction of data flow traces over time and checking whether the next instance of a message will violate a standing privacy norm. Aucher et al. [5] employ a related approach that checks transmission rules using a modal logic.

The PrivacyLFP (least fixed point) logic introduced by DeYoung et al. [18] was applied to the HIPAA Privacy Rule and Gramm-Leach-Bliley Act (GLBA), which governs the privacy of personal financial information. This formalization is based on contextual integrity [7] and introduces the least fixed point, which is used to combine norms: a transmission is *lawful* if it satisfies at least one positive norm and all of the laws negative norms. Exceptions in PrivacyLFP are alternate courses of action, similar to the priority hierarchies described by Breaux and Antón [9]. While these formalizations of contextual integrity address a related problem, how to reason over permitted and prohibited information flows to detect potential conflicts, they do not account for data or purpose hierarchies, which introduce conflicts due to partially shared interpretations as we discuss in Sect. 3.2.1. In addition, these prior findings are largely theoretical without a step-wise method to map text policy statements to their formalizations.

May et al. [30] introduce privacy APIs, which is a logical framework that includes a language to express permissions using *commands*, which upon execution modify a knowledge state. The command language includes actions such as transfer, data creation, and rights establishment, among others. Policies expressed using the framework are formalized in the Promela model-checking language, which is based on LTL and can be checked against invariants using the model checker SPIN.

Finally, Hanson et al. [20] introduce their vision for data purpose algebra to calculate the set of restrictions under which data can be used. This envisioned calculation restricts uses to the purposes for which data are collected, which is needed to comply with the US Privacy Act.

8.5 Summary differences

While substantial prior work exists in the area of expressing privacy policy, there are stark differences to distinguish these approaches. Table 6 presents a side-by-side comparison of the different languages, including the *domain* of focus (e.g., Web privacy governing the sharing of consumer information with Web sites vs. access control within an application or operating system); the *prescriptions*, such as (P)ermissions, (O)bligations, (R)efrainments or prohibitions, (E)xclusions, and (S)anctions; and the *semantics*, whether the semantics are declarative (e.g., a documented syntax, but no formal semantics) or based on an established formalism (RDF, OWL, etc.) If there exists a formal semantics, then we can more concretely compare the implications of expressions in the language and consider the computational complexity of reasoning under an increasingly large rule base.

P3P is a unique policy language for expressing consumer-facing privacy preferences [15]; this category is generally unmatched by other approaches, but also not intended to address the enterprise data flow requirements we describe in this paper. There are few intrinsic differences between the general-purpose authorization languages (E-P3P, EPAL, XACML, Ponder) and semantic Web policy frameworks (KAoS, Rei). Notably, both EPAL and XACML provide a mechanism for encoding follow-on obligations that are incurred by exercising a permission (follow-on obligations are indicated by an [O] in Table 6). While the semantic Web policy frameworks are based on the Web Ontology Language (OWL), the fundamental benefits of using DL, which is not supported by the full OWL language, are never discussed and the potential benefits of the Semantic Web appear to be assumed by this prior work [22, 37]. That said, we know that OWL-full and RDF are undecidable, which may incur limits in the ability to scale applications of these languages. Finally, ExPDT is a unique case in that it provides the bounded reasoning of DL with the general expressiveness of Rei and KAoS. In ExPDT, conflicts are reduced to DL satisfaction, which means that if a policy contains a conflict, that policy is not satisfiable. In our approach, we reduce conflict detection to DL subsumption, which allows an analyst to precisely identify conflicting interpretations between two requirements and remove these conflicts from their specification; an outcome not achievable in ExPDT as presently formulated.

The privacy models and frameworks also provide significant contributions in policy analysis. Each approach provides formal definitions for compliance and directly addresses relevant privacy principles in the specification of data flow requirements. In contextual integrity, *weak compliance* means an actor satisfies a policy with each action, locally, whereas *strong compliance* means that, with each action, actors can also satisfy any future obligations entailed by these actions. Alternatively, PrivacyLFP defines *lawful* transmissions as those that satisfy at least one positive norm and all negative norms, thus formally incorporating the notion of exceptions. While privacy APIs does not define compliance *per se*, it does employ a model-checking algorithm to find a model that satisfies a policy. Finally, the data purpose algebra provides a vision that we believe can be formalized by our DL axiomatization and can thus be used to verify policy compliance with the purpose specification principle. The first three approaches concern policies expressed in LTL, which intrinsically focus on event traces as the domain of discourse.

Our approach complements this prior work with several new contributions, including a detailed analysis of the method to map privacy-related policies to our formalization, including verb heuristics to support this mapping, and a DL axiomatization framework that is shown to reduce conflict detection to DL subsumption, which is PSPACE-complete for the DL family ALC [6]. The framework was designed for use by developers in isolation with limited knowledge about third-party practices. Finally, the language is supported by a tool set that has been validating in a simulation to assess performance across policies of reasonable size.

9 Discussion and conclusions

In this paper, we presented the Eddy language that requirements analysts and system designers can use to formally specify their application's data flow requirements. These formal specifications enable checking for conflicts and tracing permissible, required and prohibited data flows within the specification. As software increasingly leverages platforms and third-party services, we believe developers need lightweight formalisms and tools such as Eddy to check their design intent against developer guidelines and privacy policies in the larger ecosystem. This is especially true as developers work with the compositions of services in which they have no access to the internals of third-party code and processes and they are unaware of all the third parties in their data flow. To extend the language, we are presently studying multi-stakeholder interactions across more complex service compositions, including an extension to our formalism to verify privacy properties across third-party data flows.

As our first demonstration, we applied the formal language Eddy to real-world policies from Facebook, Zynga, and AOL Advertising in an extended case study and a replication study. These studies demonstrate how to identify conflicts, which an analyst can then resolve by modifying their policy and/or their privacy practices. Conflicts are difficult to detect in data flow requirements because the cross section between what is permitted and prohibited appears in the intersection of multiple classification hierarchies, including hierarchies of actor roles, information types, and purposes. Description logic provides a basis for formally making these comparisons and detecting conflicting interpretations. Our formalization provides a means to itemize these interpretations, which then allows analysts to take additional steps to reconcile the conflict.

We also discuss limitations of the Eddy language and opportunities for improving the language. In addition to reasoning about data aggregation, we foresee opportunities to extend the language to include formal expressions of user consent. By documenting the permitted and prohibited privacy requirements that are electable by users, including their default status (elected or not elected), analysts can reason about how consent affects and shapes data use and transfer. Similarly, data retention requirements may be used by analysts to explore how data minimization reduces the risk of data breaches. Data minimization must be appropriately scoped to avoid disrupting services or inhibiting an organization's ability to respond to legal requests. Together, these envisioned extensions could further help engineers design their systems to more effectively accommodate privacy and we believe the Eddy language described in this paper provides a foundation for these extensions.

Finally, we conducted a simulation to demonstrate the computational complexity of identifying conflicts in specifications of reasonable size (above what we observed in the wild). By selecting DLs, we chose a logic that could accommodate the kinds of inferences needed to check for conflicts between actor roles, information types, and purposes. This particular family of DL is known to be PSPACE-complete; however, theorem prover performance is affected by how each prover's Tableau algorithm is designed. For example, our performance evaluation found significant differences between Pellet, HermiT and Fact++ in which Pellet was unusable due to the kinds of DL expressions that we use in our approach. In the future, we plan to investigate new optimizations that can reduce the resource requirements to analyze specifications larger than 100 privacy requirements.

Acknowledgments We thank Dave Gordon and Darya Kurilova for their earlier feedback and the Requirements Engineering Lab at Carnegie Mellon University. This work was supported by NSF Award #1330596 and ONR Award #N00244-12-1-0014.

Appendix

The context-free grammar for the privacy requirements specification language, called Eddy, is presented here in the Extended Backus-Naur Form (EBNF):

```
S            = HEADER newline POLICY newline
HEADER       = "SPEC HEADER" newline ONTOLOGY*
ONTOLOGY     = tab actor ONTO_OP actor newline
                 | tab datum ONTO_OP datum newline
                 | tab purpose ONTO_OP purpose newline
ONTO_OP      = "<" | ">" | "=" | "!="
POLICY       = "SPEC POLICY" newline REQT+
REQT         = tab MODALITY ACTION newline
MODALITY     = "R " | "O " | "P " | "E " | "ER " | "EO " | "EP "
ACTION       = "COLLECT " (OBJECT)? (SOURCE)? (PURPOSE)?
                 | "USE " (OBJECT)? (SOURCE)? (PURPOSE)?
                 | "TRANSFER " (OBJECT)? (SOURCE?) (TARGET)? (PURPOSE)?
OBJECT       = datum ("," datum)*
SOURCE       = " FROM " actor ("," actor)*
TARGET       = " TO " actor ("," actor)*
PURPOSE      = " FOR " purpose ("," purpose)*
```

References

1. Anderson A (2006) A comparison of two privacy policy languages: EPAL and XACML. ACM workshop on secure web services, pp 53–60
2. Ashley P, Hada S, Karjoth G, Schunter M (2002) E-P3P privacy policies and privacy authorization. In: Proceedings of the ACM workshop on privacy in the electronic society, pp 103–109
3. Antón AI, Earp JB, He Q, Stufflebeam W, Bolchini D, Jensen C (2004) Financial privacy policies and the need for standardization. IEEE Secur Priv 2(2):36–45
4. Antón AI, Earp JB (2004) A requirements taxonomy for reducing web site privacy vulnerabilities. Requir Eng J 9(3):169–185
5. Aucher G, Boella G, van der Torre L (2010) Privacy policies with modal logic: a dynamic turn. In: Lecture Notes on Computer Science, vol 6181, pp 196–213
6. Baader F, Calvenese D, McGuiness D (eds) (2003) The description logic handbook: theory, implementation and applications. Cambridge University Press, Cambridge
7. Barth A, Datta A, Mitchell JC, Nissenbaum H (2006) Privacy and contextual integrity: framework and applications. In: IEEE symposium on security and privacy, pp 184–198
8. Breaux TD, Antón AI (2005) Analyzing goal semantics for rights, permissions, and obligations. In: IEEE international requirements engineering conference, Paris, France, pp 177–186
9. Breaux TD, Antón AI (2008) Analyzing regulatory rules for privacy and security requirements. IEEE Trans Softw Eng 34(1):5–20
10. Breaux TD, Antón AI, Doyle J (2009) Semantic parameterization: a conceptual modeling process for domain descriptions. ACM Trans Softw Eng Method 18(2) (article 5)
11. Breaux TD, Vail MW, Antón AI (2006) Towards regulatory compliance: extracting rights and obligations to align requirements with regulations. In: IEEE requirements engineering conference, pp 49–58
12. Breaux TD, Baumer DL (2011) Legally 'reasonable' security requirements: a 10-year FTC retrospective. Comput Secur 30(4):178–193
13. Breaux TD, Rao A (2013) Formal analysis of privacy requirements specifications for multi-tier applications. In: IEEE 21st international requirements engineering conference (to appear)
14. Bradshaw J, Uszok A, Jeffers R, Suri N, Hayes P, Burstein M, Acquisti A, Benyo B, Breedy M, Carvalho M, Diller D, Johnson M, Kulkarni S, Lott J, Sierhuis M, van Hoof R (2003) Representation and reasoning for DAML-based policy and domain services in KAoS and Nomads. In: 2nd International joint conference on autonomous agents and multi agent systems
15. Cranor L et al (2006) Platform for privacy preferences (P3P) specification. W3C working group note
16. Cohen J (1968) Weighted kappa: nominal scale agreement with provision for scaled disagreement or partial credit. Psychol Bull 70(4):213–220
17. Dean J, Ghemawat S (2004) MapReduce: simplified data processing on large clusters. In: 6th Symposium on operating system design and implementation
18. DeYoung H, Garg D, Jia L, Kaynar D, Datta A (2010) Experiences in the logical specification of the HIPAA and GLBA privacy laws. In: ACM workshop on privacy in the electronic society, pp 73–82
19. Farrell CB (2011) FTC charges deceptive privacy practices in Google's rollout of its buzz social network. In: U.S. Federal Trade Commission News Release, March 30, 2011
20. Hanson C, Berners-Lee T, Kagal L, Sussman GJ, Weitzner D (2007) Data-purpose algebra: modeling data usage policies. In: 8th IEEE workshop on policies for distributed systems and networks, pp 173–177
21. Horty JF (1993) Deontic logic as founded in non-monotonic logic. Ann Math Artif Intell 9:69–91
22. Kagal L (2004) A policy-based approach to governing autonomous behavior in distributed environments. Ph.D. Thesis, University of Maryland, Baltimore County
23. Kahmer M, Gilliot M, Muller G (2008) Automating privacy compliance with ExPDT. In: 10th IEEE conference on e-commerce technology, pp 87–94
24. Krippendorff K (2004) Content analysis: an introduction to its methodology. Sage, Thousand Oaks
25. Leon PG, Cranor LF, McDonald AM, McGuire R (2010) Token attempt: the misrepresentation of website privacy policies through the misuse of p3p compact policy tokens. In: 9th Workshop on privacy in the electronic society, pp 93–104
26. Lin HT, Sirin E (2008) Pellint—a performance lint tool for pellet. In: International workshop on OWL: experiences and directions (OWL-ED 2008)
27. Lupu E, Sloman M, Dulay N, Damianou N (2000) Ponder: realizing enterprise viewpoint concepts. In: 4th International conference on enterprise distributed object computing, Japan, pp 66–75
28. Lutz C, Wolter F, Zakharyashev M (2008) Temporal description logics: a survey. In: 15th IEEE international symposium on temporal representation and reasoning, pp 3–14
29. Moses T (ed) (2005) eXtensible Access Control Markup Language (XACML), v.2.0, OASIS Standard
30. May MJ (2008) Privacy APIs: formal models for analyzing legal and privacy requirements. Ph.D. Thesis, University of Pennsylvania

31. Nissenbaum H (2004) Privacy as contextual integrity. Wash Law Rev 791:119–158
32. Powers C, Schunter M (2003) Enterprise policy authorization language, version 1.2. W3C Member Submission
33. Park J, Sandhu R (2004) The UCONABC usage control model. ACM Trans Inf Syst Secur 7(1):128–174
34. Steel E, Fowler GA (2010) Facebook in privacy breach. Wall Street J. http://online.wsj.com/news/articles/SB10001424052702304772804575558484075236968
35. Sweeney Latanya (2002) k-Anonymity: a model for protecting privacy. Int J Uncertain Fuzziness Knowl Based Syst 10(5):557–570
36. Tonti G, Bradshaw JM, Jeffers R, Montanari R, Suri N, Uszok A (2003) Semantic web languages for policy representation and reasoning: a comparison of KAoS, Rei, and Ponder. LNCS 2870:419–437
37. Uszok A, Bradshaw JM, Lott J, Breedy M, Bunch L (2008) New developments in ontology-based policy management: increasing the practicality and comprehensiveness of KAoS. In: IEEE workshop on policies for distributed systems and networks, pp 145–152
38. Wan F, Singh MP (2005) Formalizing and achieving multiparty agreements via commitments. In: 4th international joint conference on autonomous agents multiagent systems, pp. 770–777
39. Yin RK (2009) Case study research, 4th edn. In: Applied social research methods series, v.5. Sage Publications
40. Yu T, Li N, Antón AI (2004) A formal semantics for P3P. ACM workshop on secure web services, pp 1–8
41. Young J (2011) Commitment analysis to operationalize software requirements from privacy policies. Requir Eng J 16:33–46