Analyzing Goal Semantics for Rights, Permissions, and Obligations

Travis D. Breaux and Annie I. Antón
Department of Computer Science
North Carolina State University
{tdbreaux, aianton}@eos.ncsu.edu

Abstract

Software requirements, rights, permissions, obligations, and operations of policy enforcing systems are often misaligned. Our goal is to develop tools and techniques that help requirements engineers and policy makers bring policies and system requirements into better alignment. Goals from requirements engineering are useful for distilling natural language policy statements into structured descriptions of these interactions; however, they are limited in that they are not easy to compare with one another despite sharing common semantic features. In this paper, we describe a process called semantic parameterization that we use to derive semantic models from goals mined from privacy policy documents. We present example semantic models that enable comparing policy statements and present a template method for generating natural language policy statements (and ultimately requirements) from unique semantic models. The semantic models are described by a context-free grammar called KTL that has been validated within the context of the most frequently expressed goals in over 100 Internet privacy policy documents. KTL is supported by a policy analysis tool that supports queries and policy statement generation.

1. Introduction

Internet privacy policies inform consumers about how organizations collect and use their personal information. These policies theoretically serve as a basis for consumer browsing and transaction decisions. Each policy differs greatly because of the lack of standardization across different industries and organizations. This lack of standardization for expressing organizational privacy practices presents a daunting learning curve for consumers who wish to compare different organizations' policies before deciding to whom they will entrust their personal information. Because both software requirements specifications and privacy policy documents establish, at a minimum, a semi-formal contract, it is important that the statements expressed in both artifacts be accessible through natural language regardless of the stakeholders' technical expertise.

Policies and requirements are similar in that they express what *must* or *ought* to be done [1], but they also differ significantly. Policies have broader scope than a system's requirements because they govern multiple systems and the activities of users. Just as a law must survive constitutional challenge, a specified system should be demonstrably policy-compliant [1]. Policies are also more open-ended than requirements and subsequently more open to interpretation. Thus, while policy and requirements are sets of rules, they are not equally subject to formal specification and analysis. Alignment of policies and requirements is therefore not a matter of mere logical consistency. The inherent ambiguity in policies makes them especially vulnerable to potential misinterpretation as well as inconsistent enforcement, making it difficult to properly operationalize policies into software requirements.

Goal analysis [2, 3, 4] offers methodical and systematic approaches both for formulating policy goals and guaranteeing that a system's requirements are in compliance with these policies [5]. A teleological model consists of a hierarchy of goals, in which some goals are sub-goals of higher-level goals [2, 6, 7]. High-level goals represent business objectives or high-level mandates. Lower-level refinements consist of achievement goals that are associated with the performance of tasks either by the system or its users. Goal-driven approaches address why systems are specified and implemented as they are, expressing the rationale and justification for specific features.

The GBRAM (Goal-Based Requirements Analysis Method) [6, 7, 8] is a straightforward methodical approach to identify system and enterprise goals and requirements. It is useful for identifying and refining the goals that software systems must achieve, managing trade-offs among goals, and converting them into operational requirements. The method has been successfully applied to analyze systems for various organizations [6, 7, 8] and goals have proven useful for analyzing and refining privacy policies [9, 10].

Natural language privacy policy statements can be systematically analyzed using the GBRAM and a content analysis technique called goal-mining. *Goal-mining* refers to extracting goals from data sources by applying goal-based requirements analysis methods [10]. The extracted goals are expressed in structured

Proceedings of the 2005 13th IEEE International Conference on Requirements Engineering (RE'05)
0-7695-2425-7/05 $20.00 © 2005 IEEE

natural language [9]. Goals are organized according to goal class (privacy protection or vulnerability [10]) and according to keyword and subject (e.g. browsing patterns, personalization, cookies, etc.). These goals are documented in a Web-based Privacy Goal Management Tool (PGMT) [9]. To date, the tool contains over 1,200 goal statements extracted from over 100 Internet privacy policy documents. The following are example goals as expressed in the PGMT:

G_{642}: SHARE customer information with subsidiaries to recommend services to customer

G_{867}: USE customer email address for marketing and promotional purposes

G_{1166}: SHARE customer information with third-parties to perform marketing services on our behalf

Researchers have acknowledged the need for methods to analyze and refine policy specifications [11]. Bandara et al. note the need to derive enforceable policies from high-level goals. Their approach relies on Event Calculus and abductive reasoning to derive the operations that together satisfy system policy goals. Our approach seeks to develop rich models that enable specification and enforcement of specific rights, permissions and obligations established by organizations, individuals, or a combination of parties involved in deciding how information is used.

We have developed a preliminary framework for specifying and analyzing privacy policies [12], but given the informal nature of structured natural language goal statements, we need a "language":

- in which rights are made concrete and applicable to descriptions of states and actions (the matter of system specifications);
- that forces analysts to think about relational commitments and communication among parties (the matter of information systems); and
- that defines right as relationships between parties so that we can analyze the rights as well as their conditions of legitimacy and relativity to various circumstances [13].

For example, we can ask whether a company is entitled to disclose certain information according to its published policy in the United States and/or Europe given different International privacy laws. To this end, we seek ways to represent the rights, permissions, obligations and other relationships relevant to privacy policies so that they may be compared and systematically analyzed. Ultimately, this will enable companies and government agencies to automatically monitor and audit policy enforcement.

This paper is organized as follows: section 2 introduces relevant background, terminology and the semantic parameterization process. Section 3 discusses queries and their role in an application: the natural language correspondence between semantic models and policy statements. Section 4 provides validation and observations from example semantic models developed using the proposed process. Section 5 explains how our proposed approach compares to relevant work in requirements engineering. Finally, Section 6 summarizes our findings and plans for future work.

2. From policy goals to semantic models

Semantic models are structural representations of meaning that are sufficiently unique to distinguish different concepts from one another. We seek a modeling formalism in the spirit of Minsky's frames or Schank's scripts [14, 15]. Whereas frames and scripts refer to the full scope of natural language, our work has the restricted scope of requirements and machine-readable policies. For our work, *semantic models* consist of three formal relations defined over natural language words: a unary relation that identifies the main idea, an associative, binary relation that relates a concept to a set of unique conceptual relations and a declarative, binary relation that relates a conceptual relation to a set of concepts. The range of acceptable words for each concept and conceptual relation share a single part of speech. The full definition of semantic models is presented with an example in section 2.2.

Our approach is motivated by grounded theory, where the discovery of theory that was systematically obtained is valid for that dataset [16]. We develop our semantic models using a process called semantic parameterization that includes identifying restricted natural language statements (RNLSs) and expressing them as comparable semantic models. In this approach (see Figure 1), goals that were previously (a) mined from privacy policy documents [9] and stored in the PGMT are now (b) re-stated to form RNLSs that are then (c) parameterized to derive semantic models.

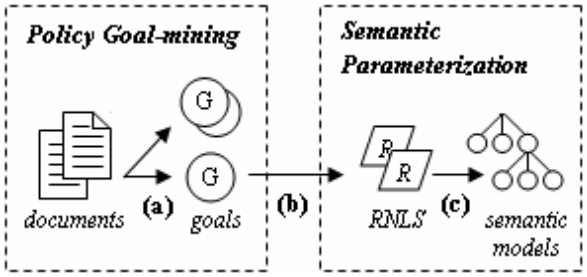

Figure 1: Semantic Parameterization Process.

Developing automated policy analysis techniques relies on the minimal capability to compare policy statements. Because natural language statements are intractable for our purposes, we initially compared policies using policy goals that offer more concise and

consistent representations of information. Goals are semantically difficult to compare due to the various ways the same goal can be re-stated in natural language. However, they are well suited for semantic parameterization because their structure often satisfies the requirements for RNLS. Furthermore, the parameterization process yields semantic models that are easily comparable using automated techniques.

2.1. Restating goals into RNLS

The PGMT contains over 1,200 privacy goals extracted from over 100 website privacy policy documents. The goals are expressed in structured natural language typical of requirements specifications that describe an event and allow nested activities that identify the *actors*, *actions* and *objects*. Each goal begins with a specific keyword, a verb that describes the primary action performed by the actor. Consider goal G_{161} from the PGMT repository:

```
G161: COLLECT information from non-
affiliates.
```

In this goal, the action is "collect" (a verb) and the object is "information" (a noun). The actor is specified as the "provider" in the PGMT. Depending on the action, other parts of speech will consistently follow the action-object pair. For example, in this goal a noun "non-affiliates" follows the preposition "from". The verb "collect" suggests the pairing of this preposition with some noun; however, it is not required by the verb. In other words, "from non-affiliates" could have been omitted in the goal statement, but this would have generalized the statement's meaning.

RNLS, like goals, have exactly one primary actor, action and at least one object. Unlike goals that may describe nested activities in the full scope of natural language, each RNLS is consistently composed from a single activity with nested activities identified as separate RNLSs. Consider goal G_{359}:

```
G359: RECOMMEND customer limit providing
access ID and password to when customer's
browser indicates an encrypted connection
```

Re-stating G_{359} as RNLSs requires decomposing the goal into discrete but related activities. Each activity described by an RNLS has exactly one actor and action, and must exhaustively describe the essential information in the original goal. In the decomposition, the modal "may" distinguishes rights and "will" distinguishes obligations. The following RNLSs correspond to goal G_{359}:

RNLS #1: *The customer's browser indicates an encrypted connection.*

RNLS #2: *The customer will provide access ID and password.*

RNLS #3: *The customer will limit (RNLS #2) to when (RNLS #1).*

RNLS #4: *The provider will recommend (RNLS #3) to the customer.*

The above decomposition demonstrates the re-statement of goals into RNLS(s) with respect to two common cases: transitive verbs and objects described by other activities. In RNLS #4, note the parenthetical reference to the activity in RNLS #3: this reference is characteristic of transitive verbs like "recommend" that describe another activity. In RNLS #3, we observe the conditioning of the activity expressed in RNLS #2 on the event of the activity expressed in RNLS #1. Semantic models maintain these and other important relationships to ensure information is consistently encoded and comparable across multiple RNLSs.

2.2. Building semantic models from RNLS

Semantic models describe the relationships between concepts necessary to map RNLSs into a consistent, machine-readable format. Semantic models are formally defined using a modeling notation with only a unary relation and two asymmetric, binary relations. Each model has only one unary *root relation* σ that defines the root concept or main idea. The root concept is represented by the shaded box in the model figures (see Figures 2, 3, 6, 7, 8). Parameters in the model are defined using the *associative relation* α over a concept and a parameter name. Values are assigned to a parameter using the *declarative relation* δ over a parameter and a concept. The solid directed arrows in the model figures represent declarative relations from the parameters to their assigned concepts. Different concept and parameter labels are distinguished in the notation using subscripts.

RNLS are parameterized by assigning words from the RNLS with a single part of speech to specific parameters and values. Consider RNLS #5:

RNLS #5: *The provider may share information.*

From RNLS #5, the values for the *actor*, *action* and *object* parameters are "provider," "share" and "information" (see Figure 2).

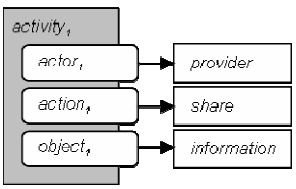

Figure 2: The activity model instance.

Parameter values never combine two or more parts of speech in order to ensure models representing similar concepts are comparable. For example, from RNLS #5

the assignments correspond to a noun, verb and noun in the semantic model. Modals such as "will" or "may" are subsumed by the parameters *right* and *obligation* not represented in the model figures. The parameterization process systematically accounts for other parts-of-speech including adjectives, articles, determiners, possessive qualifiers and conjunctions.

Continuing with RNLS #5, we derive the root relation σ ($activity_1$), associative relations $\alpha(activity_1, actor_1)$, $\alpha(activity_1, action_1)$, and $\alpha(activity_1, object_1)$, as well as the declarative relations $\delta(actor_1, provider)$, $\delta(action_1, share)$, and $\delta(object_1, information)$. Using only these three relations, the parameterization process is complete if and only if every word in an RNLS is assigned to or subsumed by one parameter or value. Completeness of the process guarantees that each semantic model maintains a natural language correspondence that enables reconstructing natural language statements from the instantiated model.

It is common for semantic models to have parameter values that are concepts with other parameters. In some instances, these concepts are also activities. We define the purpose "to market services" in RNLS #6 and restate RNLS #5 as RNLS #7 including this purpose:

RNLS #6: *The provider* ***may*** *market services.*

RNLS #7: *The provider may share information to (RNLS #6).*

We complete the second parameterization by defining the associative relation $\alpha(activity_1, purpose_1)$ and declarative relation $\delta(purpose_1, activity_2)$ in addition to other parameters and values (see Figure 3).

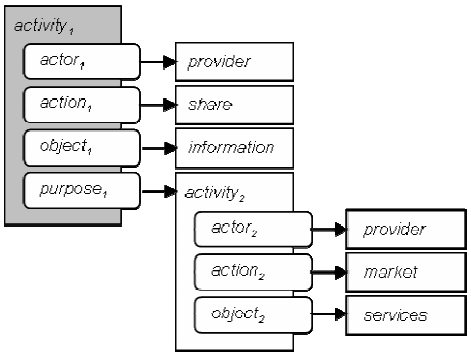

Figure 3: Semantic model with a purpose.

In the case of the *purpose* parameter, the preposition "to" will always be subsumed by this parameter; and in general, prepositions are normally subsumed by parameters. The new concept for "to market services", includes the associative relations $\alpha(activity_2, action_2)$, $\alpha(activity_2, object_2)$ and declarative relations $\delta(action_2, market)$, and $\delta(object_2, services)$. Unless the purpose explicitly states a different actor, the actor for the new activity is assumed to be the same actor as the first activity. Therefore, the relations $\alpha(activity_2, actor_2)$ and $\delta(actor_2, provider)$ are also implied by the purpose parameter.

Using these models, we can compare RNLSs by holding select parameter values constant and querying the remaining parameters' values across a set of instantiated models. Consider RNLSs #6, 7 and 8, below:

RNLS #8: *The provider may contact the customer to (RNLS #6).*

We can build a query to ask the question, "What activity can the provider perform to market services?" The query will constrain the parameters $\alpha(activity_1, actor_1)$, $\alpha(activity_2, action_2)$, and $\alpha(activity_2, object_2)$ using the values $\delta(actor_1, provider)$, $\delta(action_2, market)$, and $\delta(object_2, services)$, respectively. The query parameters $\alpha(activity_1, action_1)$, and $\alpha(activity_1, object_1)$ will then acquire the values ®*share*, *information*⌊ and ®*ontact*, *customer*⌊ from both parameterizations, respectively. These result sets are indeed the answers to our query.

2.3. Formalization in a context-free grammar

To ensure correctness of the semantic models throughout the parameterization process, the models are formally expressed in a context-free grammar called the Knowledge Transformation Language (KTL) pronounced 'kettle'. Future iterations of KTL are being developed and this discussion refers to the first iteration of the grammar KTL–1 included in Appendix A. KTL extends the formal notation to account for conjunctions, disjunctions and negations, and provides capabilities for analysis through queries. A static interpreter was developed to validate KTL and automate queries.

In KTL, an RNLS with logical conjunctions and disjunctions that are attributed to a single parameter value require special treatment. For example, the objects of an activity in an RNLS might be "employees or contractors." In this case, the restricted statement is divided into two statements, one whose object is "employees" and another whose object is "contractors". Such disjunctions are encountered with actions, objects and purposes, and each is handled in the same fashion.

To limit the burden placed on the user, KTL includes operators to describe conjunctions and disjunctions while defining special interpretations that are handled by the static interpreter. Conjunctions are handled by interpreting the declarative relation δ as a set relation with a new conjunction operator. In contrast, disjunctions describe different interpretations of a semantic model for each value. For example, the interpretation of disjunctions of values v_1, v_2 for a parameter p in Figure 4 includes cloning the model instance I for each value v_1, v_2, … , v_n in a disjunction

and assigning each distinct value v_i to the same corresponding parameter p in one of the cloned models I_i. For n separate disjunctions there are 2^n total model clones.

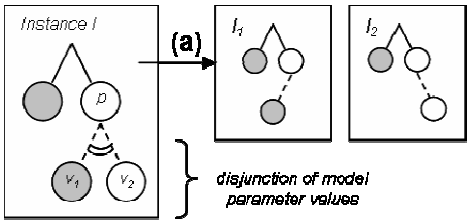

Figure 4: Interpreting a disjunction of values.

Queries are expressed in KTL as semantic models with the addition of special query variables (written *?name* for the *name* of the variable) that may substitute for parameters and values. When a query matches a model instance, the corresponding parameter or value is stored by variable name. The variable names are used in the tool to access the results from queries.

3. Queries in semantic analysis

Queries play an important role in analyzing semantic models because they enable the comparison of information across models. Queries may be designed to ask several degrees of open-ended questions by utilizing the structure of semantic models. Furthermore, queries are necessary to build more advanced applications such as the template method for maintaining natural language correspondences between semantic models and policy statements.

There are two types of queries: Boolean queries or multi-variable *wh*-queries (i.e., who, what, when, where, why and how); the latter are used to obtain answers in the form of model parameters or parameter values. All queries have an underlying semantic model, although, the *wh*-queries allow special variables in place of parameters and values in these models. Consider RNLS #9 below:

> ***RNLS #9:*** *The provider may share information with third-parties.*

The Boolean query for this statement would simply ask the question, "*May the provider share information with third-parties?*" However, one possible *wh*-query, "*With whom may the provider share information?*" applied to RNLS #9 would yield the response "third-parties." The *wh*-queries are useful for abstracting a class of semantic models that all share the same grammatical structure but differ by specific parameters and values.

Queries are subsumptive, meaning they will match models that at minimum describe the information specified in the query. Model information beyond the scope of the query is ignored and does not cause the query to fail. For example, the Boolean query "*May the provider share information?*" would also match the model for RNLS #9 despite the information "with third-parties" being represented in the model but missing from the query. Finally, queries can be partially ordered (by comparison) and used to iteratively refine the detail of information in a semantic model.

3.1. Asking targeted, open-ended questions

In order to illustrate the impact of queries on the parameterized goal subset, we present the results from an example query that asks the open-ended question, *"What type of information is shared and with whom?"* The answer includes the goal ID for each matching parameterized goal. In this query, we restrict the action to "share" and the object of the main activity to types of information. The goal ID, actor, and target parameters are allowed to range over any possible value. In this example, the "target" is the recipient of the action "share." Each row in Table 1 represents a result from the query over the 100 most frequently occurring goals in our set. The repetition of the goal IDs among responses is characteristic of model cloning resulting from disjunctions in the goal statements.

Table 1: Results from Qualitative Analysis

ID	*Object*	*Target*
155	transaction information	subsidiary
155	experience information	subsidiary
822	PII†	affiliate
822	PII	service-provider
954	Information	third-party
954	Statistics	third-party
156	transaction information	affiliate
156	experience information	affiliate
170	PII	subsidiary

The query results demonstrate the comparability of instantiated models and corresponding RNLS(s). The ability to formulate such queries is prerequisite to tasks such as automated conflict identification and requirement categorization by purpose. Queries can play a significant role in goal refinement and requirements specification by iteratively eliciting statements that answer important *wh*-questions.

† Personally-Identifiable Information (PII)

3.2. Natural language correspondence

Semantic models may be mapped to natural language statements using a template method based on queries. Each template consists of two parts: a query and an output statement. The query establishes a class of models that match a given template. The output statement is used to generate the natural language statement to express the model's meaning in human-readable form.

Query subsumption dictates that multiple templates may match a single semantic model. We resolve this conflict by selecting the template whose query matches the most information for any given model. We determine which query elicits more information by establishing a partial-order relation based on the comparability of queries. For example, given two queries q_1 and q_2, we determine if the class of models M_1 defined by q_1 are a subset of the class M_2 defined by q_2 by applying the query q_1 to the model underlying q_2. If q_1 matches q_2 and q_2 does not match q_1 then q_1 subsumes q_2 and furthermore q_2 elicits more information than q_1.

The output statements for the two types of queries Boolean or *wh*-queries are recognizably different. If the query is Boolean, then the output statement exactly describes the meaning of the matching models and therefore contains no variable names. In a *wh*-query, the output statement may contain variable names matching those used in the query. For example, consider the model m and template $T = \{ q_1, s_1 \}$ in KTL:

m:
```
activity {
    actor    = provider
    action   = share
    object   = information
    target   = affiliates
}
```
q₁:
```
activity {
    actor    = ?actor
    action   = share
    object   = ?object
    target   = ?target
}
```
s₁: The `?actor` may share `?object` with `?target`.

The parameter value "share" in q_1 works like a constraint that must match within a given model, while variables may assume any possible value. Therefore, if we apply q_1 to m, then the variables ®`?actor`, `?object`, `?target`¯ would acquire the values ®*provider*, *information*, *affiliates*¯, respectively. Matching q_1 from the template T would cause the variable data acquired from m to be used to populate the variables in s_1. The final statement generated from this template would be: "The provider may share information with affiliates."

In addition, sub-queries may be used to generate custom output for query responses that represent special cases. Consider the following queries q_2 and q_3 and corresponding output statements s_2 and s_3:

q₂: `?thing { attribute = ?attribute }`

q₃: `?thing [ property : ?owner ]`

s₂: `?attribute`$_1$ and `?attribute`$_2$ `?thing`

s₃: `?owner's ?thing`

These sub-queries require special handlers to correctly format the output statements. For example, in query q_2, it is not uncommon to have multiple attributes describing a particular entity. The output separates the first $n - 1$ adjectives by commas and the last two adjectives by the conjunction "and." Similarly, the proper output for q_3 must determine if the value for the variable `?owner` ends with the character 's' to know how to output the correct possessive form. Without these special handlers, the output statement may be vague because the presence of adjectives and/or the possessive form are used to disambiguate information.

4. Results

For this investigation, the semantic parameterization process was applied to the 100 most frequent goals in the Privacy Goal Management Tool (PGMT). These goals were restated in a two-stage process to form proper RNLS(s). In the first pass, the semantic models were derived from the goals only when an obvious combination of parameters in the model notation was identified for a complete parameterization. In the second pass, the goals that were not previously parameterized were re-stated using observations from the first pass to produce a complete parameterization. Although this process is partially subjective, the first pass produced general models that were re-used during the second pass to simplify more elaborate models. In general, identifying the atomic activities and making explicit the implied actors and objects is all that is required to restate goals into proper RNLS(s) and build a complete semantic model. The two-pass procedure made it possible to consistently parameterize the entire goal set; this required less than eight person-hours.

Applying the semantic parameterization process to the policy goal subset produced valuable insights into the semantic relationships within privacy policies. These insights are exemplified via three distinct cases. In the first case, a parameter of an activity is assigned a value of another activity as in section 4.1. Recall, this type of assignment was first introduced in Section 2.2 and shown in Figure 3. In the second case, a parameter

value is shared and thus distinguished by two different activities as in section 4.2. In the third case, two models may have a semantic correspondence where the meaning of one model is equivalent to another despite significant structural differences as in section 4.3. In these three cases, the only formal relations mentioned are those that characterize the point of interest.

4.1. Objects as other activities

Transitive verbs that describe how an actor may affect another activity can be captured by a unique semantic model. In some situations, these models may describe how actors delegate permissions and obligations to other parties. In other situations, these models may describe notifications and warnings that actors provide to other parties. In the semantic model of Figure 5, we examine the situation where an actor provides notification to another party. Consider goal G_{102}, an obligation where the main actor is the provider:

```
G102: NOTIFY customer of changes to privacy
policy.
```

We use the parameterization process to decompose G_{102} into RNLS #10 and #11.

> ***RNLS #10:*** *The provider changes the privacy policy.*
>
> ***RNLS #11:*** *The provider will notify the customer that (RNLS #10).*

Recognizing the transitive verb "notify" in RNLS #11 we derive the parameter $\alpha(activity_1, object_1)$ and assign it the value $\delta(object_1, activity_2)$ derived from RNLS #10. We add a parameter, $\alpha(activity_1, target_1)$, to account for the customer who is the recipient of the notification.

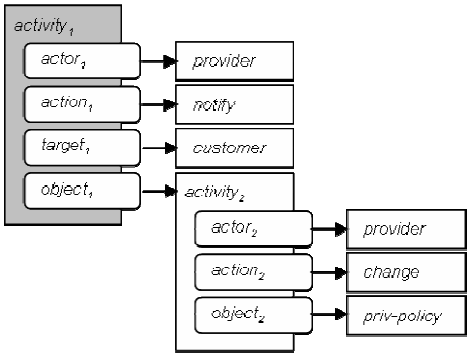

Figure 5: Object value is an activity.

In general, the model in Figure 5 covers situations where a transitive verb directly affects another activity. In addition to "notify", other transitive verbs identified in the goal subset during this process include "allow," "deny," "restrict," "limit," and "recommend." From the entire goal subset, this case occurred in 17 goals.

4.2. Objects shared by two activities

Entities may be described by multiple activities. References to other activities constrain the scope of the main activity to only those objects that have been affected by the other activities. Consider goal G_{779}, a right where the actor is the provider.

```
G779: COLLECT information provided by
customer.
```

We use the parameterization process to decompose G_{779} into RNLS #12 and #13.

> ***RNLS #12:*** *The customer will provide information.*
>
> ***RNLS #13:*** *The provider will collect information.*

The relations $\alpha(activity_1, object_1)$ and $\delta(object_1, information)$ from RNLS #12 are aligned with the relations $\alpha(activity_2, object_2)$ and $\delta(object_2, information)$ from RNLS #13 (see Figure 6).

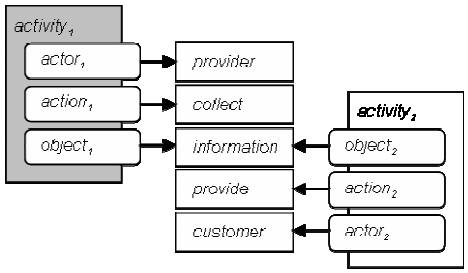

Figure 6: Object value shared by two activities.

Twelve goals were identified that generated the above semantic model. Most actions (verbs) referred to by the second activity for these models (object value shared by two activities) were in the past tense, although, a few were in the present-continuous tense.

4.3. Reflexivity of purpose and instruments

Semantic models are reflexive when the structural representation of information in two models is different yet the meaning remains the same. The choice to use one model over the other is determined by the desire to emphasize different information. For example, a model with the action parameter value "use" is reflexive with a separate model whose instrument parameter value matches the first model's object parameter value. Consider goals G_{291a} and G_{291b}, both express rights where the main actor is the provider:

```
G291a: USE cookies to collect information.
G291b: COLLECT information using cookies.
```

We use the parameterization process to decompose G_{291a} into RNLS #14 and G_{291b} into RNLS #15.

RNLS #14: *The provider may use cookies to collect information.*

RNLS #15: *The provider may collect information using cookies.*

Recognizing the verb "use" in RNLS #14 combined with the purpose "to collect information," we can establish the reflexivity between these two statements by mapping the parameter values $\delta(object_1, cookies)$, $\delta(action_2, collect)$ and $\delta(object_2, information)$ from RNLS #14 to $\delta(instrument_1, cookies)$, $\delta(action_1, collect)$ and $\delta(object_1, information)$ from RNLS #15. The corresponding models appear in Figures 7 and 8.

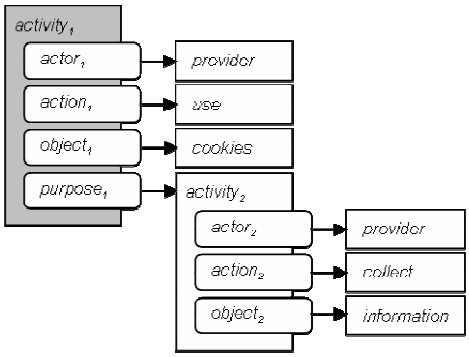

Figure 7: Model with action "use" and purpose.

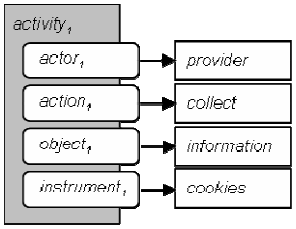

Figure 8: Semantic model with instrument.

It is important to recognize reflexive cases because it is common to use different models to emphasize different information. However, to guarantee the correct comparability of models without ambiguity, these cases must be formally identified and treated as equivalent. In all, 26 goals conform to this type of reflexivity.

4.4. Range of possible models

The above examples provide a glimpse of the range of possible models. While these examples idealize the separation of these cases, it is not uncommon for models to combine multiple cases. We performed queries over the parameterized goals to identify individual cases. Table 2 lists the queries encoded using KTL (***Expression***) and the frequency (***Freq.***) of responses among the entire goal subset.

Table 2: Queries to identify model differences

ID	Expression	Freq.
1	`goal [ right ] = ?x`	40.63%
2	`goal [ obligation ] = ?x`	50.78%
3	`goal [ !responsibility ] = ?x`	8.59%
4	`goal [ ?x ] = activity { actor = ?a action = ?b object = ?c }`	92.19%
5	`goal [ ?x ] = activity { target = ?y }`	26.56%
6	`goal [ ?x ] = activity { instrument = ?y }`	20.31%
7	`goal [ ?x ] = activity { purpose = ?y }`	12.50%
8	`goal [ ?x ] = activity { object = activity }`	17.19%
9	`goal [ ?x ] = activity { object = object : activity { object = ?y } }`	11.72%

The first three queries fully partition the goal subset since each goal only describes a right, obligation, or disclaimer of responsibility. Until now, organizing goals has emphasized hierarchies. Using semantic models and queries, goals may be dynamically categorized based on rich semantic structures such as rights, obligations, purpose, instruments, and relationships sharing the elements of multiple activities, allowing requirements engineers to view goals from different viewpoints.

5. Related Work

This section distinguishes our work from two prominent modeling frameworks in requirements engineering and provides an overview of two related approaches that transform requirements artifacts such as goals into conceptual models or semantic graphs.

The KAOS framework enables specifying system requirements in formal goal and agent models [2]. These models include a meta-level description that provides definitions for concepts, relationships and attributes relevant to goals and agents. For example, the agent model includes relations that identify agent responsibilities as states or activities represented by single predicates. The model semantics are influenced by first-order, temporal logic to support formal reasoning in conflict identification and goal management [17]. Unlike our semantic models, KAOS does not enable semantically comparing goals.

The *i** framework has been applied to security and privacy requirements analysis to analyze vulnerabilities [18]. The models in *i** represent dependencies between agents, goals, resources, and tasks using a high-level system of nodes and relations. For example, a healthcare system may depend on the activity "Perform Insurance Transaction" that in turn depends on the agent "Insurance Agent". Transforming the agent into an attacker, the dependency between agent and activity becomes a vulnerability. In contrast to *i**, our semantic models express general dependencies as specific references to the objects, instruments, and purposes expressed in natural language requirements. We further use these references in automated queries to identify privacy vulnerabilities across multiple activities with a high degree of detail as seen in the results in Table 1.

Delugach presents an algorithm for converting requirements specifications encoded in Entity-Relationships (ER), data flow, or state transition diagrams into Conceptual Graphs (CGs) with temporal extensions [19]. Our parameterization process begins with natural language goals, not structured specifications. Furthermore, Delugach does not impose strict modeling guidelines on CGs to ensure separate graphs are comparable. For example, two nodes in a CG labeled "withPurpose" and "hasPurpose" may be synonymous to the reader; however the relationship between "has" and "with" in this case is lost in the node labeling strategy. Alternatively, the relationships "with" and "has" could have been specified using individual arcs. Unlike general CGs, our semantic models enforce specific guidelines that ensure parameter values are limited to single parts of speech that represent atomic concepts. Relationships like "with" and "has" are consistently subsumed by the same parameters, ensuring relevant information remains comparable.

Koch et al. describe a framework that combines semantic graphs with goal-oriented policies [20]. The goal-oriented policies are derived from requirements specifications and defined using templates with the attributes *subject*, *action*, *target object*, and *modality* that determine authorization or obligation policies. The templates are populated using natural language requirements that describe discrete activities. Our semantic models are more expressive than the goal templates given their ability to represent purpose (see Figure 3) and actors and objects distinguished by separate activities (see Figure 6). Unlike Koch et al., our approach has been validated using an extensive repository of privacy goals.

6. Discussion and future work

In requirements engineering, the goals for a system are the customer's stated or inferred goals; and in organizational policies, the goals are those of the organization. In many situations, however, the reality and determinative role of goals has been questioned [21]. We believe that while rights, permissions and obligations are not traditionally distinguished in the goal-based framework they can be incorporated simply by admitting multiple sets of goals and indexing each set with the stakeholder that wishes to achieve them. We leave it to the politics of the situation to determine which goal set (or stakeholder) will ultimately prevail.

To this end, this paper presents a generalizable process for developing semantic models from goals that supports analysis through queries and natural language correspondences using a template method. Furthermore, these models and queries aid in the identification of conflicts, redundancies, and responsibilities of actors. We foresee semantic models playing a role in requirements engineering, but we must still address the limitations of our approach. For example, using the semantic parameterization process, we were able to completely parameterize 88 of the 100 privacy goals. The remaining 12 goals were not completely parameterized due to limitations in the context-free grammar for temporal relations.

Temporal relations were required to completely parameterize goals. In many model instances with shared objects, the action value of the second activity was a past-tense verb unlike the action value of the first activity. For example, "information provided by the customer" uses the past-tense verb "provided." In addition, different activities were related using temporal conditions. These conditions include the conjunction "unless" or the preposition "upon" (preconditions). For example, a customer right may be withheld "unless the customer initiates the transaction" or a provider obligation must be fulfilled "upon customer notification." Each of these examples relates the main activity conditionally with the completion of a separate activity. Temporal relations were also identified from the adverbs "annually," "monthly," "periodically," and "repeatedly." We treat such adverbs as attributes to actions in much the same way adjectives are handled for actors and objects.

We recognize that the RNLS restatement process, whether applied to goals or directly to policy statements, may change the meaning from what was intended in the original policy documents. For this reason, we foresee the RNLS(s) and semantic models playing a direct role in the authorship process; when policy authors need to specify policy semantics.

Goal semantics can help requirements engineers and policy makers bring policies and system requirements into better alignment. The semantic models allow analysts to compare goals and policy statements using queries to search policy or requirements documents and determine if systems comply with specific stakeholder

needs. The template method for generating natural language policy statements and ultimately requirements provides non-technical analysts the ability to validate the correctness of semantic models.

Acknowledgements

This work was supported by NSF Cyber Trust Grant #0430166. The authors thank Carolyn Brodie, Calvin Powers and NCSU ThePrivacyPlace.org reading group for their generous and helpful comments.

References

[1] Antón, A. I., Earp, J. B., Potts, C., Alspaugh, T. *5th IEEE Int'l Sym. On Requirements Engineering (RE'01)*, Aug. 2001, pp. 138–145.

[2] Dardenne, A., van Lamsweerde, A., Fickas, S. F., "Goal-Directed Requirements Acquisition", *Science of Computer Programming*, vol. 20, 1993, pp. 3 – 50.

[3] Rolland, C. Souveyet, C. and Achour, C.B.. Guiding Goal Modeling Using Scenarios, *IEEE Transactions on Software Engineering*, 24(12), pp. 1055-1071, December 1998.

[4] van Lamsweerde, A., Darimont, R. and Massonet, P.. Goal-Directed Elaboration of Requirements for a Meeting Scheduler: Problems and Lessons Learnt, *2nd Int'l Symp. on Req'ts Eng. (RE'95),* York, UK, pp. 194-203, March 1995.

[5] Antón, A.I. and Earp, J.B. Strategies for Developing Policies and Requirements for Secure Electronic Commerce Systems. In *Recent Advances in Secure and Private E-Commerce*, Kluwer Academic Publishers, 2001.

[6] Antón, A., I. Goal Identification and Refinement in the Specification of Software-based Information Systems, PhD Thesis, Georgia Tech, Atlant, Georgia, June 1997.

[7] Antón, A.I. and Potts, C. The Use of Goals to Surface Requirements for Evolving Systems, *Int'l Con'f. on Software Eng. (ICSE '98),* Kyoto, Japan, pp. 157-166, 19-25 April 1998.

[8] Antón, A.I. Goal-Based Requirements Analysis, *International Conference on Requirements Engineering (ICRE `96),* Colorado Springs, Colorado, USA, pp. 136-144, April 1996.

[9] Antón, A. I., Earp, J. B., Bolchini, D., He, Q., Jensen, C., and Stufflebeam, W. "The Lack of Clarity in Financial Privacy Policies and the Need for Standardization," *IEEE Security & Privacy*, 2(2), pp. 36-45, 2004.

[10] Antón, A. I., Earp, J. B., "A Requirements Taxonomy for Reducing Website Privacy Vulnerabilities." *Requirements Engineering Journal,* 9(3), pp.169 – 185, 2004.

[11] Bandara, A. K., Lupu, E. C., Moffett, J., Russo, A., "A Goal-based Approach to Policy Refinement." *5th IEEE Workshop on Policies for Distributed Systems and Networks (POLICY'04)*, London, June 2004, pp. 229–239.

[12] Antón, A. I. Bertino, E., Li, N., Yu, T.. "A Roadmap for Comprehensive Online Privacy Policy Management," Purdue University CERIAS Technical Report #TR 2004-47, 2004.

[13] Antón, A. I., Potts, C. "Encoding Rights, Permissions and Obligations: Privacy Policy Specification and Compliance" NSF ITR-0325269, Sept. 2003.

[14] Minksy, M. "A Framework for Representing Knowledge." In P. Winston (ed.) The Psychology of Computer Vision, McGraw-Hill, 1975, pp. 211 – 277.

[15] Schank, R. C., Abelson, R. P. Scripts, Plans, Goals and Understanding: An Inquiry into Human Knowledge Structures. Lawrence Erlbaum Assoc., Hillsdale, New Jersey, 1977.

[16] Glaser, B. G., Strauss, A. L. The Discovery of Grounded Theory. Aldine de Gruyter, Hawthorne, New York, 1967.

[17] van Lamsweerde, A., Darimont, R., Letier, E. "Managing Conflicts in Goal-driven Requirements Engineering." *IEEE Transactions on Software Engineering (TSE),* 24(11) pp. 908 – 926, 1998.

[18] Liu, L., Yu, E., Mylopoulos, J. "Security and Privacy Requirements Analysis in a Social Setting." *11th IEEE Int'l Conf. on Req'ts Eng. (RE'03)*, pp. 151 – 161, Sept. 2003.

[19] Delugach, H. S., "Specifying Multiple-Viewed Software Requirements with Conceptual Graphs." *Journal of Systems and Software*, vol. 19, pp. 207 – 224, 1992.

[20] Koch, T., Krell, C., Kraemer, B., "Policy Definition Language for Automated Management of Distributed Systems." *2nd IEEE Int'l Workshop on Sys. Mgmt. (SMW'96)*, p. 55, June 1996.

[21] G. Morgan, *Images of Organization*: Sage Publications, 1986.

[22] Parr, T. J. "ANTLR: a predicated-LL(k) parser generator." *Journal of Software Practice and Experience,* 25(7), pp. 789 – 810, July 1995,

Appendix A

Following is the context-free grammar KTL–1 represented using EBNF notation modeled on the original syntax for the ANTLR parser generator toolset developed by Terence Parr [22].

```
<start> ::= (<term>)+
<term>  ::= (IDENT|VAR) <dblock>? <ablock>?
<ablock>::= LCURL (<stmt>)+ RCURL
<dblock>::= LBRKT (<set>)+ RBRKT
<ref>   ::= <num>? NEGATE? <abs>
<abs>   ::= <term> (ABS <abs>)?
<item>  ::= <set> ((OR | AND) <item>)?
<set>   ::= LPAR <item> RPAR | <ref>
<stmt>  ::= (IDENT|VAR) EQ <set>
<num>   ::= NUMBER ((GT | LT) NUMBER)? HASH
```