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 the operational functionality 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 (CFG) that has been validated within the context of the most frequently expressed goals in over 100 Internet privacy policy documents. The CFG 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 these 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 also 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 the extraction of goals from data sources by the application of goal-based requirements analysis methods [10]. The extracted goals are expressed in structured natural language [9]. Goals are organized according to goal class (privacy protection or vulnerability [10]) as well as 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] developed at North Carolina State University. 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.

Our work to date has enabled us to develop 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.

The remainder of this paper is organized as follows. Section 2 introduces the 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 proposed approach compares to relevant work in requirements engineering.

2. From policy goals to semantic models

Developing automated policy analysis techniques relies on the fundamental ability to compare policy statements. Because natural language statements in general are intractable for our purposes, we compare policies using policy goals that offer more concise and consistent representations of information. In this approach (see Figure 1), goals that were previously (a) mined from privacy policy documents [9] and stored in the PGMT repository are now (b) re-stated to form restricted natural language statements (RNLS) that are then (c) parameterized to derive semantic models.

![Figure 1: Semantic Parameterization Process.](image)

Goals are semantically difficult to compare due to the variety of ways the same goal can be re-stated in natural language. However, they are well suited for the semantic parameterization process 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 goal repository 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:

$$G_{161}: \text{COLLECT information from non-affiliates.}$$

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 $G_{359}$:

$$G_{359}: \text{RECOMMEND customer limit providing access ID and password to when customer’s browser indicates an encrypted connection}$$

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 described 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. Our semantic models maintain these and other important relationships to ensure information is consistently encoded and comparable across multiple RNLSs.

### 2.2. Building semantic models

Semantic parameterization is the process of identifying RNLSs and expressing them as comparable semantic models. Semantic models describe the relationships between information necessary to map restricted natural language statements into a consistent, machine-readable format. Semantic models are built from formal *components* that each describes exactly one activity through a set of unique *parameters*. The parameters are second-order semantic relationships that assume values from natural language or other components. As depicted in Figure 2, every component includes at least the following parameters: an *actor*, *action* and *object*. The actor in an activity has a general capability to perform the action with respect to the object.

![Figure 2: The minimal model component.](image)

Every semantic model has only one *primary component* that describes the primary activity. However, semantic models may have *auxiliary components* assigned as values to a parameter for building relationships between activities (see Figure 3).

![Figure 3: Semantic model with a purpose.](image)

The significance of distinguishing primary from auxiliary components will become clearer as we proceed through the following example.

Semantic models are instantiated by assigning words from a RNLS with a single part of speech to the values of specific component parameters. Consider RNLS #5:

**RNLS #5**: The provider may share information.
In RNLS #5 the values for the primary actor, action and object are “provider,” “share” and “information,” respectively.

In order to maintain an atomic, meaning-preserving correspondence, parameter values never combine two or more parts of speech. For example, these assignments correspond to a noun, verb and noun in the RNLS, respectively. Modals such as “will” or “may” are subsumed by parameters not represented in the models figures (see Figures 2, 3, 6, 7, and 8.) The parameterization process systematically accounts for other parts-of-speech including adjectives, articles, parameterization process systematically accounts for other parts-of-speech including adjectives, articles, determiners, possessive qualifiers, conjunctions, etc.

Semantic models are formally defined using a modeling notation with only two asymmetric relations. Parameters in the model are defined using the associative relation α over a component and a parameter. Values are assigned to a parameter using the instance relation δ over a parameter and a value. The solid directed arrows in Figure 2 represent instance relations from the parameters to the values. Components, parameters, and values are represented in the notation using unique predicates with subscripts to distinguish parameters between different components.

Continuing with RNLS #5, we derive the following associative relations α(activity1, actor1), α(activity, action1), and α(activity, object1), as well as the instance relations δ(actor1, provider), δ(action1, share), and δ(object1, information). In all examples, the activity predicates are handles for the components represented by the shaded box in the model figures. Using only the associative and instance relations, the parameterization process is complete if and only if every word in an RNLS is assigned to or subsumed by one parameter. Completeness of the process guarantees that each semantic model maintains a natural language correspondence that enables reconstructing natural language statements from the instantiated model.

In addition to words from an RNLS, a parameter value may also be another component. In this case, the auxiliary component is an extension to the semantic model and describes a separate but related activity through the nested component. We extend RNLS #5 by including the purpose “to market services” stated in RNLS #6:

**RNLS #6:** The provider will share information to market services.

We complete the second parameterization by defining an auxiliary component using the formal parameter α(activity, purpose) and the assigned value δ(purpose, activity). 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 component for “to market services”, includes the associative relations α(activity2, action2), α(activity2, object2) and instance relations δ(action2, market), and δ(object2, services). Unless the purpose explicitly states a different actor, the actor for the auxiliary component is assumed to be the same actor as the primary component. Therefore, the relations α(activity2, actor) and δ(actor2, provider) are also implied by this purpose parameter. Figure 3 shows the purpose parameter with an auxiliary component as the assigned value.

Using these models, we are able to compare RNLS(s) by holding select parameter values constant and querying the values of remaining parameters across a set of instantiated models. Consider RNLS #6 and RNLS #7, below:

**RNLS #7:** The provider may contact the customer to market services.

We can build a query to ask the question, “What activity can the provider perform to market services?” The query will constrain the parameters α(activity1, actor1), α(activity2, action2), and α(activity2, object2) using the values δ(actor2, provider), δ(action2, market), and δ(object2, services), respectively. The query parameters α(activity1, action1), and α(activity1, object1) will then acquire the values ⟨share, information⟩ and ⟨contact, 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 (CFG), included in Appendix A. The CFG extends the modeling notation to account for conjunctions, disjunctions and negations, and provides capabilities for analysis through queries. A static interpreter was developed to validate the CFG and automate queries.

In the CFG, 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, the CFG includes special operators to describe conjunctions and disjunctions while defining special interpretations that are handled by the static interpreter. Conjunctions are handled by interpreting the instance 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 \( v_1, v_2 \) for a parameter \( p \) in Figure 4 includes cloning the model instance \( I \) for each value \( v_1, v_2, \ldots, v_n \) 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 clones of the model.

Figure 4: Interpreting a disjunction of values.

Queries are expressed in the CFG as semantic models with the addition of special query variables (written \(?\) 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 \( \text{wh} \)-queries (i.e., who, what, when, where, why); 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 \( \text{wh} \)-queries allow special variables as parameter values. Consider RNLS #8 below:

**RNLS #8: The provider may share information with third-parties.**

The Boolean query for this statement would simply ask the question, “\( \text{May the provider share information with third-parties?} \)” However, one possible \( \text{wh} \)-query, “\( \text{With whom may the provider share information?} \)” applied to RNLS #8 would yield the response third-parties. The \( \text{wh} \)-queries are useful for abstracting a class of semantic models that all share the same grammatical structure but differ in the values the model parameters may assume.

Queries may be 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 “\( \text{May the provider share information?} \)” would also match RNLS #8 despite information regarding “\( \text{with third-parties} \)” represented in the model and missing from the query. Query subsumption is a powerful property that enables iteratively refining the detail of information in a semantic model by comparing queries themselves.

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, “\( \text{What type of information is shared and with whom?} \)” 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 primary 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 statement.

<table>
<thead>
<tr>
<th>ID</th>
<th>Object</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>155</td>
<td>transaction info</td>
<td>Subsidiary</td>
</tr>
<tr>
<td>155</td>
<td>experience info</td>
<td>Subsidiary</td>
</tr>
<tr>
<td>822</td>
<td>PII</td>
<td>Affiliate</td>
</tr>
<tr>
<td>822</td>
<td>PII</td>
<td>service-provider</td>
</tr>
<tr>
<td>954</td>
<td>Information</td>
<td>third-party</td>
</tr>
<tr>
<td>954</td>
<td>Statistics</td>
<td>third-party</td>
</tr>
<tr>
<td>156</td>
<td>transaction info</td>
<td>affiliate</td>
</tr>
<tr>
<td>170</td>
<td>PII</td>
<td>subsidiary</td>
</tr>
</tbody>
</table>

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 \( \text{wh} \)-questions.
3.2. Natural language correspondence

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

Due to the subsumptive nature of queries, 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 model. We determine if the class of 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 model. We determine which query elicits more information for any given model.

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 the CFG:

$$m: \text{activity} \{ \text{actor} = \text{provider}, \text{action} = \text{share}, \text{object} = \text{information}, \text{target} = \text{affiliates} \}$$

$$q_1: \text{activity} \{ \text{actor} = \text{?actor}, \text{action} = \text{share}, \text{object} = \text{?object}, \text{target} = \text{?target} \}$$

$$s_1: \text{The ?actor may share ?object with ?target.}$$

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 $s_3$:

$$q_2: \text{?thing \{ attribute = ?attribute \}}$$

$$q_3: \text{?thing \{ property = ?owner \}}$$

$$s_2: \text{?attribute, and ?attribute, ?thing}$$

$$s_3: \text{?owner’s ?thing}$$

These sub-queries require special handlers to correctly format their output statements. For example, in the case of query $q_2$, it is not uncommon to have multiple attributes or adjectives describing a particular entity. The proper 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. Validation

As previously mentioned, 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 somewhat 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 a primary component is assigned a value of an auxiliary component 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, we show how a parameter value is shared by two components and therefore distinguished simultaneously by two separate activities as in section 4.2. In the third case, we show how two models may have a semantic correspondence where the meaning of one model is equivalent to another model 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 emphasis.

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:

$G_{102}$: NOTIFY customer of changes to privacy policy.

We use the parameterization process to decompose $G_{102}$ into RNLS #9 and #10.

**RNLS #9:** The provider will change the privacy policy.

**RNLS #10:** The provider will notify the customer that (RNLS #9).

Recognizing the transitive verb “notify” in RNLS #10 we derive the parameters $\alpha(\text{activity}_1, \text{object}_1)$ and assign it the value $\delta(\text{object}_1, \text{activity}_2)$ derived from RNLS #9. We add a new parameter, $\alpha(\text{activity}_1, \text{target}_1)$, to account for the customer who is the recipient of the notification.

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, separate activities. Including references to other activities constrains the scope of the primary 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.

$G_{779}$: COLLECT information provided by customer.

We use the parameterization process to decompose $G_{779}$ into RNLS #11 and #12.

**RNLS #11:** The customer will provide information.

**RNLS #12:** The provider will collect information.

The relations $\alpha(\text{activity}_1, \text{object}_1)$ and $\delta(\text{object}_1, \text{information})$ from RNLS #11 are aligned with the relations $\alpha(\text{activity}_2, \text{object}_2)$ and $\delta(\text{object}_2, \text{information})$ from RNLS #12. The resulting model is illustrated in Figure 6.

![Figure 6: Object value shared by two activities.](image)

From the subset of 100 goals, twelve goals were identified that generated the above semantic model. Most of the activities referred to by the auxiliary components for these models (object value shared by two activities) were in the past tense, although, a few activities were in the present-continuous tense.

4.3. Reflexivity of purpose and instruments

Semantic models may exhibit semantic reflexivity where the structural representation of information between two models is different while the meaning remains the same. In this case, the choice to use a particular model corresponds with the desire to emphasize different information represented by the models. For example, two models, one with the action parameter value “use” and a purpose parameter and another model with an instrument parameter are
reflexive. Consider goals $G_{291a}$ and $G_{291b}$, both express rights where the main actor is the provider:

$G_{291a}$: USE cookies to collect information.
$G_{291b}$: COLLECT information using cookies.

We use the parameterization process to decompose $G_{291a}$ into RNLS #13 and $G_{291b}$ into RNLS #14.

**RNLS #13:** The provider may use cookies to collect information.

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

Recognizing the verb “use” in RNLS #13 combined with the purpose “to collect information,” we can establish the reflexivity between these two statements by mapping the parameter values $\delta_{1}(\text{object}_1, \text{cookies})$, $\delta_{2}(\text{action}_2, \text{collect})$ and $\delta_{3}(\text{object}_2, \text{information})$ from RNLS #13 to $\delta_{4}(\text{instrument}_1, \text{cookies})$, $\delta_{5}(\text{action}_1, \text{collect})$ and $\delta_{6}(\text{object}_1, \text{information})$ from RNLS #14. The corresponding models appear in Figures 7 and 8.

![Figure 7: Model with action “use” and purpose.](image)

![Figure 8: Semantic model with instrument.](image)

**Table 2: Queries to identify model differences**

<table>
<thead>
<tr>
<th>ID</th>
<th>Expression</th>
<th>Freq.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>goal { right = ?x }</td>
<td>40.63%</td>
</tr>
<tr>
<td>2</td>
<td>goal { obligation = ?x }</td>
<td>50.78%</td>
</tr>
<tr>
<td>3</td>
<td>goal { !responsibility = ?x }</td>
<td>8.59%</td>
</tr>
<tr>
<td>4</td>
<td>goal { ?x = activity { actor = ?a action = ?b object = ?c } }</td>
<td>92.19%</td>
</tr>
<tr>
<td>5</td>
<td>goal { ?x = activity { target = ?y } }</td>
<td>26.56%</td>
</tr>
<tr>
<td>6</td>
<td>goal { ?x = activity { instrument = ?y } }</td>
<td>20.31%</td>
</tr>
<tr>
<td>7</td>
<td>goal { ?x = activity { purpose = ?y } }</td>
<td>12.50%</td>
</tr>
<tr>
<td>8</td>
<td>goal { ?x = activity { object = activity } }</td>
<td>17.19%</td>
</tr>
<tr>
<td>9</td>
<td>goal { ?x = activity { object = object : activity { object = ?y } } }</td>
<td>11.72%</td>
</tr>
</tbody>
</table>

The first three queries fully partition the goal subset since each goal only describes a right, obligation, or disclaimer of responsibility. In general, organizing goals has until now emphasized goal hierarchies constructed by the analyst. 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 for the purpose of formal reasoning in support of tasks such as conflict identification and goal management [14]. Unlike our semantic models, KAOS does not provide support for semantically comparing goals nor for generating natural language statements from formal goals.

The i* framework has been applied to security and privacy requirements analysis to analyze vulnerabilities [15]. 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 have further used 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 [16]. 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 [17]. The goal-oriented policies are derived from requirements specifications and defined using templates with attributes including 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 also 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 [18]. We believe that rights, permissions and obligations can be incorporated into the goal-based framework 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 set of goals (which 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 occurred in goals that were not completely parameterized. In each of the model instances with shared objects, the action value of the additional component was a past-tense verb unlike the action value of the primary component. For example, “information provided by the customer” uses the past-tense verb “provided.” In addition, other components were related to primary components using temporal conditions. These conditions most often coincided with the conjunction “unless” and 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 relate the primary activity conditionally with the completion
of a separate activity. Temporal relations were also identified from the adverbs “annually,” “monthly,” “periodically,” and “repeatedly.” Adverbs can easily be treated as attributes to actions in much the same way as 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 and the proposed template method is useful for generating natural language policy statements and ultimately requirements.

Acknowledgements

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

References


Appendix A

The following CFG represented by EBNF notation is modeled on the original syntax for the ANTLR parser generator toolset developed by Terence Parr [19].

```plaintext
<start> ::= (term)+
<term> ::= (IDENT|VAR) <dblock>?, <ablock>?
<ablock> ::= LCURL <stmt>+ RCURL
<dblock> ::= LBRKT <set>+ RBRKT
<ref> ::= NEGATE? <abs>
<abs> ::= <term> (ABS <abs>)?
<set> ::= <item> (OR | AND) <set>?
<iitem> ::= LPAREN <set> RPAREN | <ref>
<stmt> ::= <ref> (OP <value>|<item>)?
<op> ::= EQ | LE | LT | GT | GE
<value> ::= NUMBER | STRING
```