# Lexical Semantics Domain Model for Information Extraction

**Patricia Lutsky**

Arbortext, Inc.
1000 Victors Way
Ann Arbor, MI 48108
plutsky@arbortext.com

**Abstract**

The domain of operating system reference manuals uses linguistic constructs that are difficult to process since many terms are similar and most concepts are abstract software engineering constructs. The SIFT system for automatic test generation from these documents uses a natural-language-based formalism for software domain models. The formalism is based on the generative lexicon framework (Pustejovsky 1995). Examples show how this model is used for information extraction from texts in the software engineering domain.

## 1 Introduction

SIFT, which stands for "specification information from texts," is a natural-language-based information extraction tool that can extract test-related information from the free-text portion of semi-formatted software engineering documents. This information is then used to generate tests for the domain. It uses simple techniques for information extraction yet has been shown to improve tests for the domain of OpenVMS operating system regression testing (Lutsky 2000b)(Jones 1984) and in improving the documentation of an XML editor (Lutsky 2000a). For the operating system documents, required input formats and data ranges, prerequisites, and parameter interdependencies described in the reference manual can be mechanically transformed into tests for the routines. A domain model based on the linguistic constructs of the sublanguage of the domain is used in SIFT. Given the nature of software engineering, where most concepts are abstract data structures, this model is important to understanding the semantics of the texts of the domain.

The text inside the free-text sections of software reference manuals uses a restricted sublanguage (Kittredge and Lehrberger 1982), so simple domain-specific parsing techniques can be effective. A sublanguage is a semantically constrained version of a natural language spoken by a particular group of people. A sublanguage is not a subset of a natural language, but rather has its own grammar that reflects the way the group of people communicates. For reference manuals, linguistic constructs such as puns and fanciful metaphors will not be used. The tone is simple and uniform. Specific concepts tend to be described the same way throughout a document. However, processing of these texts can be difficult because software concepts are almost all abstract, taking place inside the workings of a computer. The lack of physical entities means that verbs such as "specify," "use," "has," or "is" are prevalent and these require domain knowledge to understand their meaning in this sublanguage.

### 1.1 The Domain Model

The domain model is built using the Generative Lexicon framework (Pustejovsky 1995) for representing semantic information. Although it was developed for computational linguistics applications, the Generative Lexicon formalism was used because it is a systematic and complete way to represent semantics of objects and actions. It includes multiple inheritance hierarchies, elegant handling of events, and coercion operators to handle semantic phenomena such as metonymy (when a subpart of an event or entity represents the event or entity in the sentence) (Lapata et al. 2003). With this domain model, the linguistic constructs are the core modeling medium for the model.

In the generative lexicon, each lexical item is described in terms of its argument structure, qualia structure, inheritance structure, and event structure. The qualia has four fields: *constitutive* (what the entity consists of), *agentive* (how the entity can be created), *telic* (what operations can be done to the entity), and *formal* (what type of entity it is). The fields of the qualia describe both the structure of the entity and the operations in which it can be involved. This integration of objects and actions allows domain concepts to be modeled concurrently with the operations that work on them, similar to object-oriented modeling techniques.

Inheritance can be identified along one of the perspectives (i.e. telic inheritance or formal inheritance). Type coercion mechanisms for polymorphic composition of lexical items have been developed. The generative lexicon also contains an event structure that represents how different operations can be combined and how operations must be sequenced.

As Grishman (2001) points out, there are similarities in work that was done to automatically produce sublanguage models (Grishman et al. 1986) and that being done to automatically generate information extraction patterns (Yangarber et al. 2000). Although this domain model was not generated automatically, the lexical semantics

orientation could be incorporated into automatic discovery methods.

## 2 Domain Model Specifics

The intangible nature of operating system components is unusual for domain modeling. There are few physical objects, and these (such as disk drives) are tangential to the domain. The model must focus on the conceptual level of the entities, rather than their physical characteristics. For example, the following are example domain model entries for two OpenVMS (Digital 1988) operating system concepts:

Access mode:

| | |
|---|---|
| Constitutive | value |
| Formal | inherits from enumerated list |
| Telic | specify access for an entity |
| Agentive | agent specifies value |

Logical name:

| | |
|---|---|
| Constitutive | name, access mode, equivalence string |
| Formal | software entity |
| Telic | translate to a file or device |
| Agentive | $CRELNM system service, DEFINE acl command |

The constitutive and formal fields describe the entity itself: what the sub-parts are and what type it is. Access modes are enumerated lists that have a value and logical names are software entities that have three fields: name, access mode, and equivalence string.

The telic and agentive fields concern the actions in which the entity takes part: what operations it does and how it can be created. Access modes are used to specify an access level and logical names are used to translate to a file or device. Access modes are created when a value is specified for them, and logical names are created either by a call to a system routine or by a DCL command.

## 3 Use of XML tagging

XML (WWWC 1998) tags in the subject documents provide domain-specific information for generated section headings and this context information is used in information extraction. The following tags for descriptions of arguments to an OpenVMS operating system routine include the name, logical type (vms-usage), data type, access method, and parameter passing mechanism.

```
<ARGITEM
  name="pidadr"
  vms-usage ="process_id"
  type="longword (unsigned)"
  access ="write only"
  mechanism ="by reference"/>

<ARGITEM
  name ="image"
  vms-usage ="logical_name"
  type="character-coded text string"
  access="read only"
  mechanism ="by descriptor"/>
```

For example, if the sentence

> All undefined bits in the longword must be 0.

occurs in the description of the PIDADR argument that is headed by the above PIDADR ARGITEM, the SIFT document parser would know from the

> type="longword (unsigned)"

information in the ARGITEM tag that the type of the argument is longword (unsigned). Then, when the parser identifies the referent of the phrase "the longword," the heading correctly directs the semantic processing to choose the argument that is being described.

## 4 Use of Lexical Semantics

For operating system documents, specific facts that can be used to test the software were extracted. These facts are conveyed either with generic sentences as in

> The maximum length of the table name is 31 characters.

modal sentences as in

> All unused bits in the longword must be 0.

or conditional sentences as in

> If the value of buffer length is too small, the service truncates the data.

The prevalence of these three types of sentences is due partially to the nature of the information being conveyed and partially to style preferences. For instance, modal verbs are often encouraged in technical writing as a way to avoid using the passive voice.

Sentences are first translated into an annotated syntax tree, and that tree is passed to the semantic processor that uses the domain model entries to determine the meaning of the sentence. As the following two examples show, lexical coercion and inheritance are often needed to access the appropriate semantics for lexical items in this domain.

### 4.1 Example Generic sentence

One of the sentences that conveys an argument restriction is:

> This argument is required.

The sentence is identified as a generic because of the bare "is" as the main verb. Since "this argument" is the subject of the sentence, the xml tag from the section header

indicates that the argument being described is the TABNAM argument of the $CRELNM system service. There is a domain model entry for TABNAM that can be used for further processing. The object clause is then evaluated. Since it is just the descriptor "required", the qualia roles for TABNAM are searched for a REQUIRED boolean in the constitutive role. This is not found, so the entry is checked for inheritance in the constitutive role. There is none, so coercion of the entry to its formal role is tried and the formal role is scanned for the REQUIRED attribute. The formal role is ARGUMENT, so the constitutive role of the ARGUMENT domain model entry is scanned for a REQUIRED attribute. It is found and it is a boolean value, so this is the boolean that the sentence says is true.

### 4.2 Example conditional sentence

An example sentence that contains a conditional testable fact is:

> If you omit this argument, the access mode of the caller is associated with the logical name.

While "you" refers directly to the reader and the referent of "this argument" comes from the context of the sentence as identified in the XML markup, "the caller" and "the logical name" require domain knowledge.

To process this sentence, the system starts with the verb "omit" which in the sublanguage requires a human subject and a value argument. The human subject comes from "you" and the value argument from coercing "this argument" to the value of the current argument, ACMODE, using the XML context information and the constitutive role of ACMODE.

Then, "associate" requires two value arguments. The first is the value of the access mode of the caller and the second is the value of the access mode of the logical name. To locate the first argument, we must know from the sublanguage domain model that "caller" refers to the process making the $CRELNM call from the formal role of "caller." Further, the constitutive role of "process" contains an access mode. To locate the second argument, we must know from the context that the routine being described is $CRELNM, the result of which will be the creation of a new logical name. Then, we use the constitutive role of logical name to know that it also has an access mode. The value of this access mode is the second argument to "associate." The processing uses the telic role of "caller," to call the routine, and the telic role of "$CRELNM," to create a logical name. It also uses the constitutive role of "process," " ACMODE," and "logical name," and the formal role of "caller."

## 5 Conclusions

This paper showed examples where the generative lexicon-based domain model was useful for information extraction tasks in the domain of automatic test generation from software system documents. This technique would be useful for other domains where a model of the sublanguage is needed in order to extract facts from texts. Building the complete domain model is an expensive undertaking; perhaps adaptive text extraction discovery methods could be used to seed a lexical semantics model automatically.

## References


Digital Equipment Corporation. 1988. *OpenVMS System Services Reference Manual Version 5.0.*

Grishman, Ralph. 2001. Adaptive Information Extraction and Sublanguage Analysis. *IJCAI-2001 Workshop on Adaptive Text Extraction and Mining.*

Grishman, R., Hirschman, L., & Nhan, N. 1986. Discovery procedures for sublanguage selectional patterns: Initial experiments. *Computational Linguistics*. *12*. 205-215.

Jones, Larry. 1984. Regression testing of VMS. *Proceedings of the Digital Equipment Computer Users Society.* 611-618.

Kittredge, R., & Lehrberger, J. (Eds.). 1982. *Sublanguage: Studies of language in restricted semantic domains*. New York:Walter de Gruyter.

Lapata, Mirella, Keller, Frank, and Scheepers, Christoph. 2003. Intra-sentential context effects on the interpretation of logical metonymy. *Cognitive Science*. *27*. 649-668.

Pustejovsky, James. 1995. *The Generative Lexicon*. MIT Press.

Lutsky, Patricia. 2000a. Information Extraction for Validation of Software Documentation. *Proceedings of the 13th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems.* 583-590.

Lutsky, Patricia, 2000b. Information extraction from documents for automating software testing. *Artificial Intelligence in Engineering. 14.* 63-69.

World Wide Web Consortium 1998. *Extensible Markup Language (XML) 1.0. W3C Recommendation 10-February-1998*. http://www.w3c.org/TR/REC-xml.

Yangarber, Roman, Grishman, Ralph, Tapanainen, Huttunen, Silja, 2000. Automatic Acquistion of Domain Knowledge for Information Extraction. *Proc. of the 18th International Conference on Computational Linguistics (COLING 2000)* 940-946.