Feature grammar systems. Incremental maintenance of indexes to digital media warehouses

Windhouwer, M.A.

Citation for published version (APA):
Chapter 7

Case Studies

De oplosmythe (1)
Met een computer is elk probleem op te lossen.

De oplosmythe (2)
Met een computer is mijn probleem op te lossen.

De oplosmythe (3)
Met een computer is een of ander probleem op te lossen.

Joachim Graf - De computerwetten van Murphy

Throughout the development of the Acoi system and its major component, feature grammars, several case studies were conducted to assess its practical impact. In the next sections these case studies will be described together with the lessons learned. However, the Acoi system itself is also a case study in software engineering of a system based upon feature grammar systems. This chapter starts with a section describing the successive versions of the system. The case studies can then be related to the version being used, and to the changes they inspired.

7.1 The Acoi Implementation

The Acoi system is developed by the database research group of the Dutch Centre for Mathematics and Computer Science (CWI) from 1997 to 2003. This section contains a brief history of the system development so the case studies, which will follow later on in this chapter, can be placed in the right context.

7.1.1 Acoi Prehistory

Before Acoi became a project subsidized by several national projects (AMIS, DMW and Waterland) the database research group already had laid some foundations. In
the late eighties the Grammatical Database Model was investigated in two unpublished manuscripts [Ker89, KS89] and a prototype implementation developed by a master student. This model is based on this quadruple: \((L, F, G, T)\). \(L\) is a language described by an unambiguous CF grammar, \(F\) is a set of transducers defined for \(L\) which can produce a new sentence from an existing sentence, \(G\) is a set of guardians defined for \(L\) which determine if a new sentence is valid and, finally, \(T\) is a collection of built-in and user defined types. The transducers and guardians from this model closely resemble the black- and whitebox feature detectors from the feature grammar systems. However, the design of this model was never completely finished.

\textit{Acoi} stands for \textit{Amsterdam Catalog of Images} and the system's first aim was to build and maintain a collection of images for research purposes. Feature detectors were identified as a basic building block for a database-based content-based image query system. The Acoi image algebra [NK98, Nes00] was a result of these efforts.

Both lines of (past) research flowed naturally into the Acoi project.

### 7.1.2 The Acoi Project

In the fall of 1997 Acoi became a CWI project for providing database support for the management of multimedia features. This internal project was mainly funded by the national DMW project, which will be described in some more detail in Section 7.3. Note that although not all the components were implemented in the various versions of the Acoi system, the general architecture shown in Figure 1.4 has been clear from the start. The exact details still needed to be filled in, which is the focus of this thesis.

### 7.1.3 Acoi 1998

The research started with the construction of a web robot in Java to gather images from the World Wide Web. This robot interacted with and helped debugging the Monet database server using the ODMG interface, which was under development at that moment. Concurrent to this robot a first version of a feature grammar based toolset was implemented [KNW98a]. This implementation would read in a specific feature grammar and generate the \(C\) source code for a grammar specific recursive descent parser.

### 7.1.4 Acoi 2000

The first rewrite, which was mainly targeted at a cleanup of the code base, of the Acoi system was still based on parser source code generation. The parse trees constructed by this generated FDE could first be dumped as a set of \(Mx\) macro calls [KSvdBB96]. The expansion of these macro calls would lead to a MIL script to insert the parse tree into the Monet database. Later on, this propriety setup was replaced by the combination of XML and XSLT (see Chapter 5).

The image robot was also rewritten into a feature grammar with an accompanying set of detectors. This provided the first experience with the general system
Section 7.1: The Acoi Implementation

architecture. Parts of the previous implementation were not reused as mainly string handling, needed for the parsing of the HTML pages, is far too expensive in Java. Also controlling timeouts on HTTP connections turned out to be cumbersome. The implementation of HTML related detectors was therefore done in Tcl and later on in C [Vei03, W3C02a].

Based on the performance characteristics of the system during several case studies the internals of the system underwent several optimizations. First of all the token pool was hierarchically organized so larger portions were shared by different recursive descent levels in the parsing process. This allowed descending and ascending, i.e. backtracking, to be cheap and to cut away extensive copying of tokens.

The next optimization concerned the binding of detector parameters. In this binding process the internal tree had to be traversed. This tree could be traversed by a path expression language loosely based on XPath 1.0. However, when this tree grew big these traversals would visit too many nodes. By allowing additional hints in the path expression, i.e. indicating forward or backward traversal, these superfluous node visits could be prevented. For example:

```
| *detector shotlist(ancestor::video/child-forward::filename); |
```

would ensure that traversal of the child axis would start at the first child of video. This in contrast to the default traversal strategy which implemented a backward depth-first search.

Feature grammars were seen as CF grammars with an limited amount of context-sensitivity [SWK99], i.e. not yet as a specific instantiation of CD grammar systems. Ambiguity was only allowed in a limited fashion: all alternatives should consume exactly the same tokens.

Around the FDE an extensible set of scripts in various languages, e.g. Tcl, MIL and XSLT, was build to implement the WWW search engine (see Figure 7.1). These scripts contained various hooks to insert knowledge not explicit in the feature grammar (at that time). This setup will be discussed in some more depth in Section 7.2.

This version of Acoi has been used extensively for case studies and has been described in [SWK99, WSK99, dVWAK00, NWH+01, WSvZ+03].

7.1.5 Acoi 2002

To accommodate the expected FDS implementation, the FDE was rewritten into an interpreter. As indicated in Chapter 4 an interpreter handles a changing grammar more easily and spares the FDS the hassle to manage a recompilation of a grammar specific FDE.

The feature grammar language, and also its parser, was redesigned to cope with modules, whitebox detectors, classifiers and plugins. The support for these features was added to the generic FDE. Detectors and plugins became dynamic loadable libraries. Furthermore, the implementation made use of more XML standards: DOM
Figure 7.1: The WWW multimedia search engine architecture.

for the internal parse tree format and subsets of XPath for whitebox detectors and detector parameters [Vei03]. Feature grammar checks were added to warn for various (possible) weaknesses in the grammar. This included the semantic checks described in Section 4.3.1.

A first implementation of the FDS was able to construct and visualize the dependency graph. Also experiments were conducted with various implementations of XML diff algorithms.

The robot was once more reimplemented. In this case the monolithic grammar was cut up into several media type related feature grammar modules. Lessons on the patterns embedded in these modules will be described in the upcoming Section 7.2.

7.1.6 Acoi Future

The current implementation of Acoi still lacks some of the key aspects of the theory described in this thesis. The main absence is a complete FDS to replace the scripts which make up the WWW robot. The current FDS implementation lacks interaction with one or more FDEs and an interface to a tree diff algorithm to find some internal changes.

As described in Chapter 4 parse forests can be stored in one XML document by adding a scope and context attribute. At this point this level of ambiguity is not supported by the FDE. The FDE allows multiple alternatives to be true, but they all should describe the same subsentence. However, the scope/context scheme has been prototyped using a set of XML documents and XSLT templates.
Likewise, the view of feature grammars as feature grammar systems is not completely reflected in the implementation. This view makes a cleaner separation between subsentences as produced by different detector functions and thus belonging to different grammar components. This asks for these sentences to be only within the scope of their component and thus prevents the need for hierarchical sharing of tokens.

Once Monet completely supports an XML/XQuery front-end (see Section 5.1) the specific XSLT scripts can be deleted as XML documents can then be stored natively. This would also accommodate a closer, but still standardized, binding between the FDE and the database. Also allowing a clean addition of support for both memoization and references.

7.2 The WWW Multimedia Search Engine

One of the first targets for Acoi was the construction of an image index for the Dutch AMIS project. The size of the index aimed for was 1,000,000 images. To find these images the HTML pages containing their URLs had to be parsed and interpreted, so soon the index was extended with a full text indexing facility. As this case study played a major role over the years [KNW98b, WSK99, WSK00, BWvZ+01] annotation extraction algorithms were added for other multimedia types. In the upcoming sections the feature grammars and system architecture involved will be discussed in more detail.

7.2.1 The Feature Grammars

Moving away from the first monolithic feature grammar, the current set of feature grammar modules are very similar to the running examples in the previous chapters and showcased in Appendix B. The major decision points used in the grammars are based on the MIME type of the multimedia object under inspection. This MIME type is retrieved by the generic WebHeader detector which knows the HTTP protocol to retrieve this information. Using the primary and secondary MIME type, whitebox detectors in the feature grammar steer the FDE to the set of multimedia type specific detectors, e.g. language detection for HTML pages and face detection for images. Detectors for a specific multimedia type are grouped into one feature grammar module. The Acoi module combines all the modules into one grammar, which is used by the FDE to harvest links from the web. The complete set of detectors is listed in Table 7.1.

As all multimedia web objects are related through HTML pages it is possible to traverse these anchors and access a specific context of an object. Using this navigational information this, typical, query can be answered: show me a web page about "Chet Baker" containing a portrait. This query combines key words from the HTML page ("Chet Baker") with a high-level concept (portrait) extracted from the image object.
<table>
<thead>
<tr>
<th>Multimedia type</th>
<th>Detectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic web objects</td>
<td>HTTP header information&lt;br&gt;allowance by the robot exclusion protocol</td>
</tr>
<tr>
<td>Text files</td>
<td>DRUID language classification</td>
</tr>
<tr>
<td>HTML pages</td>
<td>title, anchors and text extraction&lt;br&gt;WordNet synsets [CSL01]</td>
</tr>
<tr>
<td>images</td>
<td>global color features&lt;br&gt;graphic/photo classification [ASF97]&lt;br&gt;skin coverage [GAS00]&lt;br&gt;face detection [LH96]&lt;br&gt;portrait classification&lt;br&gt;thumbnail creation</td>
</tr>
<tr>
<td>MP3 audio files</td>
<td>ID3 tag extraction</td>
</tr>
<tr>
<td>MIDI audio files</td>
<td>MIDI fields</td>
</tr>
<tr>
<td>MPEG video files</td>
<td>animated video icon</td>
</tr>
</tbody>
</table>

Table 7.1: The WWW multimedia search engine detector set.

7.2.2 The System Architecture

The system architecture (shown in Figure 7.1) uses a set of shell, Tcl, MIL and XSLT scripts to explore the World Wide Web. The various system components provide hooks to plugin feature grammar specific scripts. These hooks are mainly used to implement knowledge about references, as those are not supported by the generic tools in Acoi 2002. The FDS as described in Chapter 6 exploits the explicit knowledge of multiple start symbols and references and can take over the role of these scripts in a future version of Acoi.

A small walk through will clarify the role of the various system parts. The user, i.e. the librarian, starts the database server and provides an initial set of candidate URLs. The user also starts one or more robots for corresponding Internet domains and/or multimedia types. Each robot contacts the database server for a subset of the candidates and starts a number of FDEs to index these. The FDE returns the location of the XML document containing the parse forest to the robot, which in its turn tells it to the database server. The database server contacts the Tomcat servlet engine [Pro03] to retrieve the XML document and transform it into a MIL script. The servlet engine provides several advantages: (1) it limits the startup time of the Java-based XSLT processor\(^1\), (2) it provides the possibility to run each robot on a different remote machine and thus makes the architecture more scalable.

---

\(^1\)XT [Cla99] is, although written in Java, one of fastest XSLT processors [Dat01]. However, the startup time of the Java interpreter is still significant, but can be reduced to a one time affair by embedding XT in a servlet [Tch01]. The alternative to link in a XSLT processor with the FDE has been proved to be still slower than this XT setup when the parse forest grows into a XML document of several hundred megabytes.
The extensibility of the system was tested several times by the addition of new detectors. A detector basically means the addition of a new branch to a parse forest. For this the FDE supports incremental parses of existing parse trees. This parse tree is loaded from the database and a special command line option tells the FDE for which symbol to start (and stop) the detection process. The FDE will thus rebuild the internal parse tree using the retrieved tokens and will start detection when the new symbol is encountered. When the FDE returns to the symbol in the post traversal the detection is stopped again. In this way the extended parse tree (or forest) is build and sent to the database server. The database server will replace the old parse tree by this new one.

Using this architecture the robot harvested (within 2 weeks in 2000) links to 4,300,000 web objects from which it entirely indexed about 2,000,000. The index contained 750,000 images from which about 10% were classified as photographs. The major bottleneck of the system was the MIL parser in the Monet database server. This parser is non-reentrant and thus protected by a lock. Concurrent parse forest insertion scripts spent their time mainly waiting to obtain this lock.

The scalability of the index has been extended on the level of the Monet database server by passing the terms on to a specialized full text indexing service [Blo02]. This service uses several machines for horizontal fragments of the term index. A term is assigned to a fragment based on the \textit{tf-idf} ranking model [BYRN99]. This integration has been described in [BWvZ+01] and [WSvZ+01].

This WWW indexing engine is different from traditional search engines at several points. Traditional search engines are mostly based on information retrieval (IR) theory and use technology common in this line of research [BYRN99], \textit{e.g.} inverted files instead of a database system. These IR indexes are build from scratch with each new web crawl. In the feature grammar case updates to the index happen incrementally, \textit{i.e.} queries (readers) and crawling (writers) find place on the same database using concurrent transactions. Furthermore, due to the feature grammar the index is easily extended with new multimedia types and new features.

### 7.2.3 Lessons Learned

The WWW multimedia search engine has been incrementally developed during the various version of Acoi. The extended language features – modules and references – were mainly inspired by this case study. Modules make it possible to easily reuse well defined parts of the feature grammar in a different context, \textit{e.g.} the other case studies. The fact that the anchors between HTML pages form a graph complicated the annotation extraction from the start. Its possible to keep this knowledge implicit by embedding it in the detector implementations. However, making it explicit gives the Acoi tools the opportunity to handle referential integrity and to offer support for complicated recursive structures, \textit{i.e.} deadlock detection. Ambiguity did not play a major role within this feature grammar system. Decision points are deterministic and mostly handled by whitebox detectors.
### Chapter 7: Case Studies

7.3 The Australian Open Search Engine

Research on the Acoi system has been mainly carried out within the *Digital Media Warehouse (DMW)* project. This project's aim was to advance content-based retrieval techniques in large multimedia databases. To achieve this goal, the project was split up into sub-projects for three levels (see Figure 7.2):

1. The conceptual level focuses on querying semi-structured data;
2. The logical level focuses on steering multimedia annotation extraction;
3. The physical level focuses on the storage of semi-structured data.

The logical level directly interacts with a collection of content analysis algorithms, also part of the research portfolio of the project.

Feature grammar systems and the accompanying Acoi system implement the logical level. The physical level was implemented by Monet XML (see Section 5.1.1). Both the content analysis algorithms and the conceptual level were developed at the University of Twente. Before describing the Australian Open case study in more detail, the research of these project members is shortly described\(^2\).

---

\(^2\)Parts of the subsequent sections are written by the co-authors of DMW related papers.
7.3.1 The Webspace Method

The conceptual level focuses on limited domains of the Internet, i.e. intranets and large web-sites. The content provided on such domains is often highly related and structured. This aspect makes it feasible to determine a set of concepts, which adequately describe the content of the document collection at a semantic level.

The Webspace Method [vZ02] offers a methodology to model and search such a document collection, called a webspace. The Webspace Method defines concepts in a webspace schema using an object-oriented data model. This collection is stored as XML documents in the XML storage level of the global system architecture, see Figure 7.2. A strong correlation between the persistent documents is achieved, since the structure of each XML document describes (a part of) the webspace schema in turn. Actually each document contains a materialized view over the webspace schema; it contains both content and schematic information.

The webspace schema is used to formulate queries over the entire document collection. Novel within the scope of search engines and query formulation over document collections is that it allows an user to integrate information stored in different documents in a single query. Traditional search engines (e.g. AltaVista) are only capable to query the content of a single document at a time. Furthermore, using the Webspace Method specific conceptual information can be fetched as the result of a query, rather than a bunch of relevant document URLs.
Figure 7.4: A fragment of the webspac e schema for the Australian Open website.

### 7.3.2 COBRA

In line with Chapter 1 the COBRA video modeling framework [Pet03] recognizes four layers (see Figure 7.3): the raw data, the feature, the object, and the event layer. The object and event layers consist of entities characterized by prominent spatial and temporal dimensions respectively. In [Pet03] several instantiations of this model are constructed for different domains and using different machine learning techniques. As will be shown in the upcoming section, feature grammar systems provide a way to build domain specific instantiations of the COBRA model.

### 7.3.3 The Australian Open DMW Demonstrator

To demonstrate the power of the DMW system the Australian Open demonstrator was build [BWvZ+01, WSvZ+01, PWvZ+02, WSvZ+03, WvZ03]. The Australian Open\(^3\) is a grand slam tennis tournament on a yearly basis. The demonstrator is based on the tournament of 2001.

The conceptual elements available in the structure of the website were modeled in a webspac e schema. A fragment of this schema is shown in Figure 7.4. Using a set of special purpose feature grammars the HTML pages from the original website were transformed into the base XML documents of the webspac e. These documents contain an instantiation of part of the schema (see the areas in Figure 7.4). As the website did not contain any video fragments from the matches, some matches were recorded and digitized. Then the index database for the base data, including the multimedia content, was build. For this the webspac e tools extracted meta-data from the base documents, and triggered the FDE when a multimedia object was found. The FDE would then steer the video annotation extraction process. This process worked along the lines

\(^3\)www.ausopen.org
Section 7.3: The Australian Open Search Engine

shown in Figure 7.5 and captured in the feature grammar in Appendix B.11. This feature grammar is a domain specific instantiation of the COBRA model. Notice that this grammar reuses feature grammar modules developed for the WWW multimedia search engine.

Finally a special purpose query interface was built. The formulation of a query in this GUI can be divided into three steps. During the first step, the query skeleton is constructed, using the visualization of the conceptual schema. Secondly, the constraints of the query are formulated, using the attributes of classes used in the query skeleton. In the last step, the user has to define the structure of the result of the query, which is generated as a materialized view on the conceptual schema.

Before continuing with the individual steps of the query formulation process, the queries presented below are used to illustrate the power of the search engine with respect to query formulation. The queries express the typical information need of an expert user querying the Australian Open document collection. It also shows, how after each query, the information need of the user can be refined, resulting in a more complex query over a document collection.

Q1. ‘Search for left-handed female players, who have played a match in one of the
Constructing the query skeleton. The first step of the query formulation process involves the construction of the query skeleton. This skeleton is created, using a visualization of the webspace schema. This visualization consists of a simplified class diagram, and only contains the classes and associations between the classes, as defined in the webspace schema. The user simply composes the query skeleton, based on his information need, by selecting classes and related associations from the visualization. The (single) graph that is created represents the query skeleton.

In Figure 7.6.a a fragment taken from the GUI of the webspace search engine is presented, which shows the query skeleton (depicted in black-filled text boxes), that is used for the query formulation of the three example queries.

Formulating the constraints. In the second step of the query formulation process, the constraints of the query are defined. In Figure 7.6.b another fragment
of the GUI of the search engine is presented, showing the interface that is used for this purpose. For each class contained in the query skeleton a tab is activated, which allows a user to formulate the conceptual constraints of the query. As shown in the figure, a row is created for each attribute. Each row contains two check boxes, the name of the attribute, and either a text field or a button.

The first checkbox is used to indicate whether the attribute is used as a constraint of the query. The second checkbox indicates whether the results of the query should show the corresponding attribute. If the type of the attribute is a BasicType, a text field is available that allows the user to specify the value of the constraint, if the first checkbox is checked. If the attribute is of type WebClass, a button is available, which, if pressed, activates the interface that is used to query that particular multimedia object.

Figure 7.6.c shows the interface that is used to formulate queries over Hyper-text-objects, i.e. define content-based constraints. The figure shows both a low-level and advanced interface to the underlying feature grammar system. In the low-level interface projection and selection criteria can be filled in (see Section 5.2.5). The advanced interface is similar to the interfaces offered by the well-known search engines such as Google and Alta-Vista. The main part of the query-interface allows a user to formulate one or more terms, which are used to find relevant text-objects. The interface also allows the user to perform a case-sensitive search, and to select the language of the Hyper-text-object in which the user is interested.

Figure 7.6.b shows the attributes of class Player. The constraints with respect to the player, specified in the first two example queries, are transposed in the selections depicted in the figure. Two constraints are formulated. The constraint that the user is only interested in female players is defined by selecting the constraint checkbox in front of gender, and by specifying the conceptual term 'female'. The second constraint refers to the second example query, where an extra condition with regard to the player's history is formulated. Again, the corresponding checkbox is activated, and the interface of Figure 7.6.c is started, and completed. In this case, the query-terms 'winner' and 'champion' are used to find the relevant Hyper-text-objects that are associated with the player's history.

3. **Defining the resulting view.** The second column of checkboxes is used to specify which attributes will be shown in the resulting views defined by the query. The XML document that is generated by the webspace search engine contains a (ranked) list of views on the webspace that is being queried. Besides selecting the attributes and the classes that will be shown as the result of the query, the user also has to determine which class is used as the root of the resulting view. In Figure 7.7 a screenshot of the result for the third query is shown. It includes a link to a tennis scene of the match played by Monica Seles
Figure 7.7: The result of example query 3.

in the quarter final round. The tennis scene shows a video-fragment in which Monica Seles plays near the net.

This DMW architecture consisting of webspaces, feature grammars, an instantiation of the COBRA video model and efficient XML storage resulted in a search engine which allows a combination of conceptual and content-based multimedia search [WvZ03], thus giving the user the power to post very specific queries to the database.

7.3.4 Lessons Learned

The feature grammar system for the Australian Open case study is a direct extension to the set of feature grammars for the WWW multimedia search engine. The extension did inspire two language features: constants within the production rules and plugins. The Segment detector not only detects scenes within the video but also their type. To prevent superfluous type detectors, this type was encoded as a string and matched by a string constant in the various alternatives. A remote procedure call (RPC) detector was developed which was quite generic and thus easily converted into the template-like approach of a plugin.

The main focus of this case study was the embedding of the Acoi system within the larger DMW application. In the WWW search engine the feature grammar is the main schema, but in this case the grammar is connected to the conceptual webspace schema.
Both the webspace tools and Acoi could have steered the meta-data and multimedia extraction process. The choice was made for a top-down implementation, i.e. the conceptual level triggers the logical level.

Looking back, the embedding of Acoi within the current DMW system could have been more tight. On the one hand by a tighter coupling with the webspace schema and thus with the conceptual data, allowing more specific semantic contexts for detectors. This coupling could be created by translating the concepts and their attributes into elementary feature grammar trees. Also on the side of the multimedia content analysis the current integration is shallow, i.e. the detector granularity is very coarse. By splitting the implementation of the Segment and Tennis detectors into smaller detectors, decision points can be made explicit and thus become manageable by the FDE and FDS. Drawbacks of a finer detector granularity is the possible many conversions needed from token to actual values, i.e. strings to integers or floats, and opening and closing of the media object. The latter drawback may be circumvented by adding a generic caching interface to the Acoi toolset.

### 7.4 Rijksmuseum Presentation Generation

As stated in Chapter 1 museums are digitizing their collection and making them available to the public. The Rijksmuseum, situated in Amsterdam, did the same. High resolution scans of photos made of paintings, statues etc. and the existing annotation database were made available to researchers in the Token2000 project. The Rijksmuseum Presentation Generation project developed an architecture for using (automatic extracted) annotation information for the automatic generation of user specific hypermedia presentations [NWP+01, NWH+01]4.

The architecture is shown in Figure 7.8 and consists of three major units:

- the style repository, which embodies style schemata, style grammars and rulebases for different presentation styles;
- the data repository, containing the images and related meta-data, and the retrieval engine;
- the presentation environment, including a presentation generator and a hypermedia browser.

In the next sections these units will be introduced and the way in which they interact with each other will be described.

#### 7.4.1 The Style Repository

The aim of the style repository is twofold. On the one hand it provides a collection of representations describing styles in fine art, such as clair-obscur, impressionism or

---

4 Parts of the subsequent sections are written by the co-authors of these papers.
Tell me all about clair obscurs

Query Generalisation

Linguistic Analysis

---

Data repository

Style repository

---

Query result

9 Images
3 Texts
4 Titles

6 Portraits
2 on Portraits
3 on Portraits

3 Landscapes
1 on Landscapes
1 on Landscape

---

Presentation environment

Linear
Topic Separated
6 Portraits
3 Landscapes

Spatial constraints
Optimisation

1 2 3 4 5 6 7 8
1 2 4 5 6

1 5 2 4 9 7 8

Go Back
or make
new query

---

Figure 7.8: Hypermedia presentation generation framework.

cubism, in a structured way. These collections are designed to improve the retrieval of images or other meta-data in the data repository (see the data repository section below). On the other hand it provides a presentation rule-base in which rhetorical structures describe how retrieved material can be presented.

The collection of representations of fine art styles provides for each style mainly text-based schemata. A schema holds information about the definition, the main period, the inventor of this style, other artists using or improving it, etc.. The advantage of these style ontologies is that they allow an enlargement of the search-space, if the style plays a prominent part in the query. This is state-of-the-art technology within the retrieval community [SR00]. The development of text-based ontologies is still a mainly manual task, which is today quite well supported [GEF+99].

This information, while important, is insufficient, low level feature descriptions are also required. Rather than a random choice of features as a style description, features that represent the intrinsic characteristics of a particular style need to be col-
lected. In *clair-obscur* images, for example, a clear distinction of light and dark areas can be found. Usually there is one dominant light source, predominantly filled with high luminance colors, alongside dark areas with a high proportion of brown colors which can be blended with other objects [Arn74]. Thus, a collection of features such as color, shapes, brightness, either in the form of their extraction algorithms or as threshold values for a particular style, facilitate the automatic identification of relevant material. Such a collection can naturally be represented by a feature grammar.

Note that the development of feature-based representations also requires human effort, in particular by specialized experts who have an understanding of the compositional structures of an image [Pei60]. The collection of these features is, on the other hand, not too difficult, since tools for this particular task do exist [FCP00].

The third representation form in the style repository is of a different sort. Here rules are collected which describe rhetorical presentation structures, as addressed in the Rhetorical Structure Theory [MMT89] or Cognitive Coherence Relations [KD96], which might vary between general and specialized levels. If, for example, the presentation environment is educationally oriented, it can build presentations on a larger level of the form: Introduction Topic; Introduction Subtopic 1; Details Subtopic 1; Introduction Subtopic 2; Details Subtopic 2; Introduction Next Topic. A more detailed level specifies what an introduction means, e.g. show a definition of the topic in combination with a visual example of the topic. Another detailed description might be concerned with the sort of interaction, such as a linear presentation in the form of a slide show, or a more interactive way in the form of additional buttons for individual traversal. The combination of these rules form themselves schemata, which can then be connected to relevant styles. The design of these schemata and the connection to particular style representations again requires human effort, such as that of a graphical designer of a museum Web environment. Development environments which support such tasks are described in [ACC+99, NL00]. Once these presentation rules are in place, a system can react to the particular needs of a user.

### 7.4.2 The Data Repository

The repository, as shown in Figure 7.8, stores annotation schemata in the form of XML-based documents and media-based data, such as images in various formats (pic, gif, tiff, etc.). The repository itself can be realized using federated database technology.

The annotation documents are created by experts, using ontology-based environments for task-specific controlled vocabulary/subject indexing schemata for in-depth semantic-based indexing of various media [DAR03]. Note that annotation schemata are different from the style representations. Annotation schemata provide information about one particular image or artist. For example, they capture information about the title of an image, its painter, production date, a list of exhibitions where it was presented, reviews, and so forth. The annotation process follows a strata-oriented approach, which allows a fine-granulated space-oriented description of media con-
tent, where particular areas within an image can be especially annotated. The connection can be based on linking mechanisms as described in XML path and pointer [W3C01d, W3C02b] or MPEG-4 [ISO02].

As visualized in Figure 7.8, the annotations will hardly ever be completed. Most of the time only the most basic data will be available. Thus, even if the potential search space can be enlarged, as described earlier, there may still be a very limited information space to apply the query to.

Imagine that a user would like to know everything about Rembrandt and the different styles he painted his images in. With the textual representation the system might be able to find images by Rembrandt in the database. However, if these images have no further annotation attached than "Artist = Rembrandt" it would not be able to classify the retrieval results according to the query. Having access to the style specific representation of intrinsic features, the image can now be analyzed during the retrieval process and decide based on the results which of the relevant styles is the most appropriate for this particular image. As a side effect, the feature information gained can be used as additional annotation for the image, not only in classifying it as a particular style, but also providing several different representations of it, which can be, for example, useful for presentational purposes. An image in the style of clair-obscure can be additionally represented in a grid that contains the light and dark areas. This grid can form the basis for a presentation of images, where the style of the images and the style of the presentation correspond. The FDE steers this automatic annotation extraction process based on the feature grammar from the style repository.

As the main goal of the suggested framework is to facilitate the automatic generation of user-centered multimedia presentations, the result space will not only contain the retrieved data, associated meta-data, and the relations between these different units but also information required for their presentation. Moreover, it also returns physical information about the retrieved data, i.e. image size and image file type.

7.4.3 The Presentation Environment

The presentation environment, as displayed in Figure 7.8, is basically a constraint-based planning system, which uses the definitions provided in the style representation schemata and the presentation styles [vOGC+01]. Since the system can access descriptions based on spatial, gradient and color features, the presentation generator is in the position to analyze the retrieved material based on the relevant presentation design, according to design issues such as graphic direction, scale, volume, depth, shapes (i.e. physical manipulation of the material for better integration into the presentation), temporal synchronization (interactive or linear presentation), etc., and provides a format that a hypermedia browser can interpret, e.g. SMIL or MPEG-4 [ISO02, W3C98, W3C01c].
7.4.4 A Style Feature Grammar

Feature grammars play a role in both the style and data repositories. In both cases they extract the low-level feature descriptions, which can be used for the selection of relevant material and for the layout of the presentation. But in the latter case the features are also mapped to high-level concepts. These concepts correspond to manual annotations, and can thus replace these when they are not available.

This section will shortly describe the algorithms involved in detecting low-level, presentation oriented, features and high-level, manual annotation replacement, concepts. The grammar itself is available in Appendix B.13.

Since the design aims at an approach that is data-driven and can therefore operate unsupervised, it is important to incorporate adaptive decision-making algorithms. For instance, in the case of the clair-obscure style a vague high-level description of the style could be "a brightly lit object or person surrounded by a dark background". To translate this vague conceptual description into an operational low-level feature-
extractor, precise values need to be assigned to fuzzy concepts such as bright and dark. However, these cannot be fixed in advance because these values depend on the context, dark and bright being defined relative to the rest of the painting.

The proposed approach is data-driven in that it inspects the data in search for natural thresholds, i.e. thresholds that are dictated by the structure apparent in the data [PF00]. To be more precise, assume a numerical image-feature \( x \) that can be computed at each of the \( n \) pixel in the image (e.g. hue, or brightness, see Figure 7.9.a). This gives rise to a numerical dataset \( x_1, x_2, \ldots, x_n \). The histogram gives an idea of how these values are distributed over the image. If, in terms of this feature, the image has a clear structure then a multi-modal histogram is expected, with peaks over the most-frequently occurring feature values.

For instance, in the case of clair-obscur, computing the brightness histogram (at least) two peaks are expected: one peak at low values created by the pixels in the dark regions, and one at high values corresponding to bright pixels, see Figure 7.9.b.

Locating the grey-value at the minimum in between these peaks determines a threshold that can be used to separate the bright from the dark regions in the image. This seemingly simple task is complicated by the fact that a data-histogram almost never has a clear-cut unimodal or multi modal structure, but exhibits many local maxima and minima due to statistical fluctuations. The challenge therefore is to devise a mathematically sound methodology that allows us to construct a smoothed version of the histogram, suppressing the spurious local extrema that unduly complicate the histogram structure.

For this the empirical distribution function \( F_n(x) \) is introduced, which for each feature-value \( x \) determines the fraction of observations \( x_i \) that are smaller than \( x \). The reason for switching to the empirical distribution is that it allows to compute the precise probability that the given sample is drawn from a theoretically proposed distribution \( F(x) \). The idea is simple: search for the smoothest distribution \( F \) that is compatible with the data, i.e. such that there is a high probability that the sample \( x_1, \ldots, x_n \) has been obtained by sampling from \( F \).

In mathematical parlance this amounts to solving the following constrained optimization problem: given \( F_n(x) \) find \( F(x) \) that minimizes the functional

\[
\Psi(F) = \int (F''(x))^2 \, dx \text{ subject to } \sup_x |F_n(x) - F(x)| \leq \epsilon.
\]

(The value for \( \epsilon \) is fixed in advance by specifying an acceptable level of statistical risk). This optimization problem can be solved using standard spline-fitting routines. Once the shape of the smoothest compatible distribution \( F \) is determined, its inflection points can be used to determine the genuine local minima in the histogram, thus yielding natural thresholds for the image-segmentation extractor.

The lowest of these thresholds is then used to segment the image into dark and light areas, see Figure 7.9.c. And as a next step information about the areas is localized by overlaying the image with a grid, see Figure 7.9.d.
In the feature grammar of Appendix B.13 these steps are distributed over several detectors and their dependencies are described. For example, the \textit{light\_segment} detector calculates the brightness value of each pixel, this set of values is then taken as input by the \textit{histo\_segment} detector to determine the segmentation thresholds.

The grammar defines several other (global) features. For example, the \textit{co\_corr} detector computes the normalized correlation between the color histogram of the painting and two average normalized histograms, respectively for \textit{clair\_obscur} and non-\textit{clair\_obscur} paintings (see for a similar approach [ASF97]).

<table>
<thead>
<tr>
<th>classified as</th>
<th>clair_obscur</th>
<th>cubism</th>
<th>impressionism</th>
<th>unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>18 clair_obscur paintings</td>
<td>17</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>25 cubist paintings</td>
<td>6</td>
<td>21</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>56 impressionist paintings</td>
<td>1</td>
<td>5</td>
<td>31</td>
<td>20</td>
</tr>
<tr>
<td>83 unclassified paintings</td>
<td>70</td>
<td>1</td>
<td>26</td>
<td>-</td>
</tr>
<tr>
<td>182 paintings</td>
<td>94</td>
<td>27</td>
<td>60</td>
<td>22</td>
</tr>
</tbody>
</table>

Figure 7.10: Results for the style feature grammar

All these features form the input for the final step: determining if the painting is in the \textit{clair\_obscur} or one of the other styles. For this step style specific decision trees are derived using C4.5 [Qui93], resulting in the detectors \textit{clair\_obscur}, \textit{cubism} and \textit{impressionism}. The performance of the feature grammar in annotating paintings with the various styles is shown in Figure 7.10. The last column corresponds with the event that there is no matching style found, \textit{i.e.} the validity of the \textit{style} rule is optional.

Notice also that more than one of the alternative \textit{style} rules can be valid, which means that a painting can be annotated with multiple styles, \textit{i.e.} ambiguous views on the same painting. So the support for ambiguity by the Acol system comes in to play here. When a painting matches more styles multiple parse trees describe this one image. Each alternative, rooted by a detector, also contains a confidence value. In this case this confidence value is based on the support of the decision rule.

Furthermore, this grammar is mainly constructed to recognize paintings in the \textit{clair\_obscur} style. More features may be needed to fine tune the decision rules for the other styles, \textit{e.g.} \textit{impressionism}. The use of a feature grammar is well suited for this evolutionary approach as it supports incremental maintenance of the annotations.

### 7.4.5 Generating the Presentation

Part of the prototype implementation for the Rijksmuseum case study is a generation engine that is able to transform a high-level description of a presentation [RBvO+00] in the concrete final-form encoding that is readily playable on the end-user’s system. In this system the final encoding form is SMIL [W3C98].
Figure 7.11: Ordered retrieval result before optimization.

The presentation generation engine of the system is a constraint-based planning system. The constraint system is used for solving the design-based constraints, such as:

- the overall presentation dynamics (e.g. linear or interactive) and the resulting subdivision of information blocks;
- organizing material for each information block, e.g. number of elements on a page and their spatial outline based on the actual size of each information unit;
- optimization of ordered material based on additional style criteria, such as color or brightness distribution, in particular to emphasize a particular style.

The use of the annotations produced by the FDE are mainly of interest in the last bullet. The inner details of the other parts of the system itself, especially the transformation of the presentation structures generated by the constraint engine into a SMIL presentation, have been discussed in [vOGC+01].

It is assumed that the system constructs a linear presentation for educational reasons and creates topic blocks to present the material. Finally, based on spatial constraints, it calculates how many images for each topic block can be presented on a page.

At this stage the generator tries to arrange the images in each topic block in such a way that the style criteria for clair-obscur are fulfilled. A decision rule could look
Section 7.4: Rijksmuseum Presentation Generation

as follows:

```plaintext
style_order(Image_Style, List_of_Images, Images_Per_Page, Presentation_List):
    gradient-match(Image_Style, List_of_Images, Result_List),
    border-match(Result_List, Images_Per_Page, Presentation_List).
```

With this rule the system analyzes not the image itself but rather its grid abstraction, as shown in Figure 7.9.d. The system tries, for a particular style (Image_Style), to order the images of one topic (List_of_Images) based on the pattern provided by those cells of the grid that represent light values. The analysis of these patterns is based on graphical shapes, such as triangle or rectangles. The direction of the light is derived from a number of criteria, such as solidness of a pattern (main light center), position in the grid (at the border indicates that the light source is outside the image), and the direction of the dissolve of this shape (direction of light beam). For Figure 7.9.d the result is that the light source is outside the image, that light is coming from the left side and dissolves towards the right side in a rectangular way. The `Result_List` groups images in lists, where light follows similar directions, such as left, up-left, up, up-right, right, down-right, down, down-left, circular.

Once that is done, the system tries to align the images based on similar border pattern. Take once again Figure 7.9.d as the example, the system would try to find an image which shares a similar distribution of light and dark cells (up or down by one grid cell) but only on its right side. The combination of images is performed on the previous calculated maximum size of images per page.

Figure 7.11 shows the random image sequence. While the final presentation in Figure 7.12 uses the optimized order. This presentation is based on the rhetorics of an educational-oriented presentation, which requires introductions of topics and subtopics.

The temporal duration for every single page is calculated by the number of presented objects and their graphical complexity, the number of words for text elements, or temporal presentation qualifiers such as fade-in or out times for media units. The last screen offers choices for the next step.

### 7.4.6 Lessons Learned

The Rijksmuseum feature grammar has a much finer detector granularity than the previous grammars. It also called for the reuse of various detectors within a different context. The same feature detectors are used to determine global and local, i.e. per grid cell, features.

The use of decision rules for the various art styles inspired the addition of classifiers as a special instance of detector plugins. Furthermore, ambiguity plays a major role in this grammar as various styles may match concurrently. This triggered the addition of support for detector confidences and parse forests.
7.5 Discussion

The case studies in this chapter showed the viability of the Acoi system and its formal basis feature grammar systems. Due to its focus on limited context-sensitivity basic building blocks for an multimedia annotation system can be constructed. Actual applications can then be build by putting these blocks, i.e. grammar components, together using the dependencies.

Next to being a description of the actual annotations, feature grammar systems can also be used in a more traditional way: to describe a workflow and its associated meta-data. This was done in a recent Waterland related case study. The Acoi system offers in this case the advantage that new workflow actions can be easily plugged in. Furthermore, the whole infrastructure, e.g. plugins, can be used to easily automate these actions.

Creating a semi-automatic feature grammar is one of the future research topics. Manual detectors may provide new annotations or validate automatically extracted annotations. A case study may provide insights in how conveniently the current implementation supports this mixed type of annotation extraction.

Comparison of the Acoi system with peer systems, and thus evaluation of implementation issues, e.g. performance, remains future work. As discussed in Sections 2.3 and 4.4 the explicit description and usage of context knowledge is unique to feature grammar systems, but can also be used to generate specifications, although probably more verbose and fragmented, for these peer systems. Applying such a translation of one or several of the case studies may enable comparable runs of these systems, and thus provide (more) insight into their specific strengths and weaknesses.