Adaptation Guided Retrieval Based on Formal Concept Analysis*

Belén Díaz-Agudo, Pablo Gervás, and Pedro A. González-Calero

Dep. Sistemas Informáticos y Programación
Universidad Complutense de Madrid, Spain
{belend, pgervas, pedro}@sip.ucm.es

Abstract. In previous papers [5,4] we have proved the usefulness of Formal Concept Analysis (FCA) as an inductive technique that elicits knowledge embedded in a case library. The dependency knowledge implicitly contained in the case base is captured during the FCA process in the form of dependence rules among the attributes describing the cases. A substitution-based adaptation process is proposed that profits from these dependence rules since substituting an attribute may require to substitute dependant attributes. Dependence rules will guide an interactive query formulation process which favors retrieving cases where successful adaptations can be accomplished. In this paper we exemplify the use of FCA to help query formulation in an application to generate Spanish poetry versions of texts provided by the user.

1 Introduction

Structured domains are characterized by the fact that there is an intrinsic dependency between certain elements in the domain. Considering these dependencies leads to better performance of CBR systems and it is an important factor for determining the relevance of the cases stored in a case base [10,5]. Although dependency knowledge could be manually identified from a domain expert, we consider that the case library itself can be used as the dependency knowledge source as it contains useful knowledge beyond the individual specific pieces of problem solving experiences to be reused.

We use Formal Concept Analysis (FCA) [13,7] as an inductive technique that elicits knowledge embedded in a concrete case library. In previous papers [5,4] we have described the use of FCA to support CBR application designers in the task of discovering knowledge embedded in a case base. The application of FCA provides an internal view of the conceptual structure of the case base and it uncovers patterns of regularity among the cases. In [5] we showed how the dependency knowledge implicitly contained in the case base is captured during the FCA process in the form of dependence rules among the attributes describing the cases. These dependence rules are used to guide the process of formulating a CBR query. Moreover, the concept lattice that results from the FCA application can be used as a case organization structure: the formal concepts represent maximal groupings of cases with shared properties. Within this structure we

* Supported by the Spanish Committee of Science & Technology (TIC2002-01961)

K.D. Ashley and D.G. Bridge (Eds.): ICCBR 2003, LNAI 2689, pp. 131–145, 2003.

can access together all the cases sharing a set of properties with the given query, because they are grouped under the same concept. The extracted knowledge is dependent on the case library and it will be used to complete the knowledge already acquired by other techniques of domain modelling. In [4] we focused in classification based retrieval and the utility of Galois lattices as structures to classify and retrieve planning cases that are described by the goals satisfied by the solution and the precondition properties needed to apply the case solution. In this paper we review the use of FCA to extract dependency knowledge from a case base an exemplify it using an specific application to generate Spanish poetry versions of texts provided by the user [2]. In this application we define processes for adaptation guided retrieval [12] and substitution-based adaptation that can be enhanced by applying dependency knowledge to take into account correlated substitutions.

The poetry generation application is introduced in Section 2. Section 3 describes the basics of the Formal Concept Analysis technique and how we are applying it for knowledge elicitation on case bases. Section 4 describes how this knowledge can be used during the query formulation and retrieval tasks. Section 5 briefly describes how the rest of the CBR cycle works within this application. Finally the conclusions and limitations of our approach are discussed.

2 Poetry Generation with CBR

In [2] we chose poetry generation as an example of the use of the COLIBRI (Cases and Ontology Libraries Integration for Building Reasoning Infrastructures) system. COLIBRI assists during the design of knowledge intensive CBR (KI-CBR) systems that combine cases with various knowledge types and reasoning methods. It is based on CBROnto [3,6], an ontology that incorporates reusable CBR knowledge and serves as a domain-independent framework to develop CBR systems based on generic components like domain ontologies and Problem Solving Methods (PSMs).

Our approach to poetry generation with CBR uses an specific process that is conceptually based on a procedure universally employed when not-specially-talented individuals need to personalise a song, for instance, for a birthday, a wedding, or a particular event: pick a song that everybody knows and rewrite the lyrics to suit the situation under consideration. This particular approach to the problem of generating customised lyrics or poetry has the advantage of being easily adapted to a formal CBR architecture. No claims whatsoever regarding the general suitability of this approach for poetry composition in a broad sense should be read into this particular choice.

This paper does not aim to describe all the details application but to exemplify one of the CBROnto PSMs. Interested readers will find the details of the application in [2]. The following sections introduce the basic rules of Spanish poetry and how the required knowledge and the cases are represented in this application.

2.1 Basic Rules of Spanish Poetry

Formal poetry in Spanish is governed by a set of rules that determine a valid verse form and a valid strophic form. A given poem can be analysed by means of these rules in order to establish what strophic form is being used. Another set of rules is applied to analyse (or *scan*) a given verse to count its metrical syllables.

Given that words are divided into syllables and each word has a unique syllable that carries the prosodic stress, the constraints that the rules have to account for are the following:

Metric Syllable Count. Specific strophic forms require different number of syllables to a line. Metric syllables may be run together thereby shortening the syllable count of the line involved. When a word ends in a vowel and the following word starts with a vowel, the last syllable of the first word and the first syllable of the following word constitute a single syllable (*synaloepha*).

Word Rhyme. Each strophic form requires a different rhyming pattern.

Stanza or Strophic Form. For the purpose of this application only poems of the following regular strophic forms are considered: *cuarteto*, a stanza of four lines of 11 syllables where the two outer lines rhyme together and the two inner lines rhyme together; and *terceto*, a stanza of three lines of 11 syllables where the either the two outer lines rhyme together or the three lines have independent rhymes.

2.2 Poetry Domain Knowledge Ontology and Poem Case Base

The COLIBRI approach to building KI-CBR systems takes advantage of the explicit representation of domain knowledge. An initial sketch of an ontology about the domain of application has been developed for purposes of illustration, resulting in a knowledge base containing 86 concepts, 22 relations and 606 individuals [2].

Cases describe a solved problem of poem composition. We describe cases using the CBROnto case description language and domain knowledge terminology. We choose a case representation structure where both description and solution are linked to a correct poem.

Within our representation, a poem is a text made up of words, and built up as a series of stanzas, which are groups of a definite number of lines of a specific length in syllables, satisfying a certain rhyme pattern. Each word is represented as an individual which is an instance of the domain concept *Word* and is described in terms of the following attributes: the number of syllables that the word has, the position of the stressed syllable of the word counted from the beginning of the word, the rhyme of the word, whether the word begins or ends with a vowel, and the part-of-speech tags associated with that word. There are currently 4872 words in the vocabulary, of which 313 appear in the cases and the rest is additional vocabulary available for adaptation during the generation of new poems.

Part-of-speech Tags. In our representation each word of the available vocabulary has one Part Of Speech (POS) tag representing the syntactical category of this word. To this purpose we use the tags used in the CRATER project and its POS tagger [8] that extracts automatically the syntactical category associated to a given word. These tags are common and easy to understand so that not specific linguistic knowledge is required to use the system. What follows is an example of a poem of the case base and the POS tags associated to its words.

marchitara la rosa el viento helado
todo lo mudara la edad ligera
por no hacer mudanza en su costumbre[1]
VLFI3S ARTDFS NCFS ARTDMS NCMS ADJGMS
QUXMS ARTDNS VLFI3S ARTDFS NCFS ADJGFS
PREP NEG VLINF NCFS PREP PPOSPS NCFS

The poem uses a subset of the more than 300 POS tags identified in this project. In particular it uses the following POS tags:

- VLFI3S Lexical verb. Indicative future tense third person singular
- VLINF Lexical verb. Infinitive
- ARTDFS Feminine singular definite article
- ARTDMS Masculine singular definite article
- ADJGFS Feminine singular general positive adjective
- ADJGMS Masculine singular general positive adjective
- NCFS Feminine singular common noun
- NCMS Masculine singular common noun
- ARTDNS Neuter singular definite article
- PREP Preposition

For the sake of a comprehensive exposition we can not use the whole set of tags but we present a reduced example using the subset of tags appearing in the poem example.

3 Formal Concept Analysis

FCA [13,7] is a mathematical approach to data analysis based on the lattice theory of Garret Birkhoff [1]. It provides a way to identify groupings of objects with shared properties. FCA is especially well suited when we have to deal with a collection of items described by properties. This is a clear characteristic of the case libraries where there are cases described by features. A *formal context* is defined as a triple $\langle G, M, I\rangle$ where there are two sets G (of objects) and M (of attributes), and a binary (incidence) relation $I \subseteq G\text{x}M$, expressing which attributes describe each object (or which objects are described using an attribute), i.e., $(g, m) \in I$ if the object g carries the attribute m, or m is a descriptor of the object g. With a general perspective, a concept represents a group of objects and is described by using *attributes* (its intent) and *objects* (its extent). The extent covers all objects belonging to the concept while the intent comprises all attributes (properties) shared by all those objects. With $A \subseteq G$ and $B \subseteq M$ the following operator (*prime*) is defined as:

[1] The icy wind will cause the rose to wilt,// and all things will be changed by fickle time,// so as to never change its own routine.

$A\prime = \{m \in M \mid (\forall g \in A)(g,m) \in I\}$
$B\prime = \{g \in G \mid (\forall m \in B)(g,m) \in I\}$

A pair (A,B) where $A \subseteq G$ and $B \subseteq M$, is said to be a *formal concept* of the context $\langle G, M, I\rangle$ if $A' = B$ and $B' = A$. A and B are called the *extent* and the *intent* of the concept, respectively.

It can also be observed that, for a concept (A, B), $A'' = A$ and $B'' = B$, which means that all objects of the extent of a formal concept, have all the attributes of the intent of the concept, and that there is no other object in the set G having all the attributes of (the intent of) the concept.

The set of all the formal concepts of a context $\langle G, M, I\rangle$ is denoted by $\beta(G, M, I)$. The most important structure on $\beta(G, M, I)$ is given by the subconcept-superconcept order relation denoted by $\leq$ and defined as follows: $(A1, B1) \leq (A2, B2)$ if $A1 \subseteq A2$ (which is equivalent to $B2 \subseteq B1$ see [7]).

Basic Theorem for Concept Lattices. [13]
Let $\langle G, M, I\rangle$ be a context. Then $\langle \beta(G, M, I), \leq\rangle$ is a complete lattice, called the concept lattice of the context $\langle G, M, I\rangle$, for which infimum and supremum can be described as follows:

$$Inf\beta(G,M,I) = \left[\bigwedge_{\alpha}(A_\alpha, B_\alpha) = \bigcap_{\alpha} A_\alpha, \left(\bigcup_{\alpha} B_\alpha\right)''\right]$$

$$Sup\beta(G,M,I) = \left[\bigvee_{\alpha}(A_\alpha, B_\alpha) = \left(\bigcup_{\alpha} A_\alpha\right)'', \bigcap_{\alpha} B_\alpha\right]$$

Graphically, contexts are usually described by cross-tables while concept lattices are visualized by Hasse diagrams. The following sections illustrate how FCA is applied to our simple example, and how the dependency knowledge is extracted from the concept lattice interpretation.

3.1 FCA Application Example

We propose the application of FCA as an automatic technique to elicit the knowledge about attribute co-appearance implicit in a case library.

In the example presented in this paper we apply FCA to compute the concept lattice taking the POS tags of the words as the features under consideration. In the formal context (G, M, I), the set of objects (G) consists of all the cases in the case base[2], the set of attributes (M) consists of all the POS tags appearing in these cases[3], and the incidence relation (I) between G and M indicates that a poem has a word with a certain POS tag.

That way, the lattice captures formal concepts representing the co-appearance of POS tags that appear in the poems of the case base. We are using POS tags and not other characteristics of the words (or poems) because

[2] Case7 corresponds to the example case presented in Section 2.2
[3] In the example we are using the reduced set presented in Section 2.2

	ARTDFS	ARTDMS	ARTDNS	ADJGFS	ADJGMS	VLFI3S	VLINF	NCFS	NCMS	PREP
Caso1	☑	☑		☑				☑	☑	☑
Caso2	☑	☑		☑		☑	☑	☑		☑
Caso3			☑		☑		☑	☑	☑	☑
Caso4		☑			☑				☑	
Caso5	☑	☑			☑			☑	☑	☑
Caso6	☑	☑		☑	☑			☑	☑	
Caso7	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
Caso8		☑		☑	☑		☑	☑	☑	☑
Caso9					☑				☑	☑
Caso10								☑	☑	☑

Fig. 1. Case base formal context

of the kind of adaptation that we propose. The high level idea of the adaptation process is to substitute as many words from the poem with words from the query, without loosing the syntactic structure of the poem lines. We assume that the query is a meaningful sentence and, therefore, if we can accommodate those words into the poem in a similar order it is plausible to think that the new poem will reflect, to a certain extent, the original message in the query. In order to maintain the syntactic correctness of the poem, and taking into account that the system has no additional syntactic knowledge, we constrain substitutions to words with exactly the same POS tag.

So, POS tags of the words in the cases are interpreted as formal contexts, and represented by using the incidence tables in Fig. 1.

Besides the cross table representation, there is a graphical representation of formal contexts using Hasse diagrams. Fig. 2 shows Hasse diagrams of the concept lattice associated to the context in Fig. 1. Each node in the diagram represents a formal concept of the context, and the ascending paths of line segments represent the subconcept-superconcept relation. The lattice (Fig. 2) contains exactly the same information that the cross table (Fig. 1), so that the incidence relation I can always be reconstructed from the lattice.

In the Hasse Diagram, labels meaning attributes from the intent are marked by [ ] and labels meaning object from the extent are marked by {}. A lattice node is labelled with the attribute m∈M if it is the upper node having m in its intent; and a lattice node is labelled with the object g∈G if it is the lower node having g in its extent. Using this labelling, each label (attribute or object name) is used exactly once in the diagram. If a node C is labelled by the attribute $[m]$ and the object $\{g\}$ then all the concepts greater than C (above C in the graph) have the object g in their extents, and all the concepts smaller than C (below C in the graph) have the attribute m in their intents. In a Hasse diagram, the intent of a concept can be obtained as the union of the attributes in its label [ ] and attributes in the labels [ ] of the concepts above it in the lattice. Conversely, the extent of a concept, is obtained as the union of the objects in its label {} and objects in the labels {} of the concepts below it in the lattice.

To reconstruct a row of the original incidence relations in Fig. 1, look for the unique concept C whose label {} contains the object name heading the row and mark with a cross (in the row we are reconstructing) the column of each one of the attributes of the intent of C.

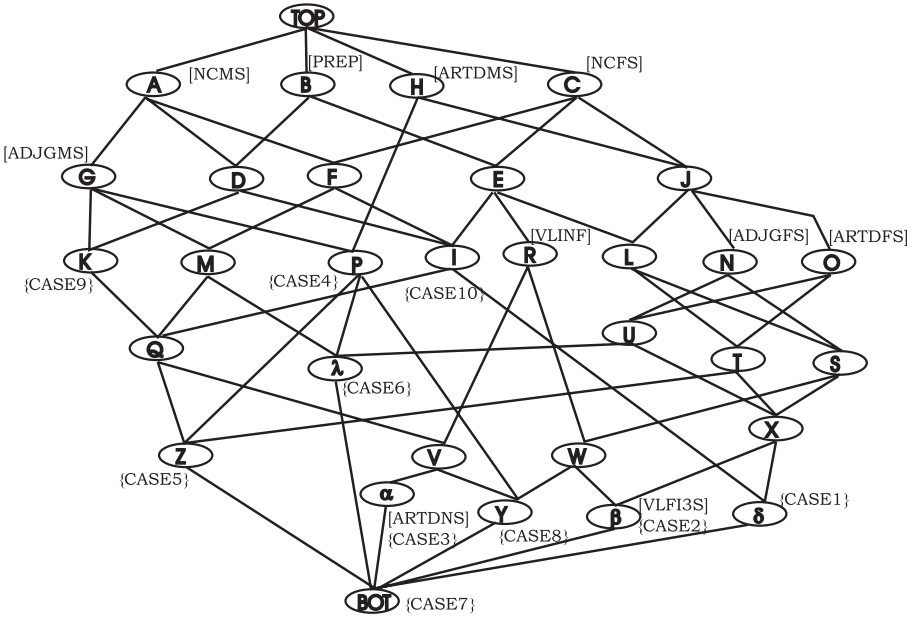

Fig. 2. FCA applied to the poem case base

For example, to reconstruct the cross table row of Case4, mark with a cross the columns corresponding to the intent of concept P in the lattice: [Adjgms, Artdms, Ncms]. Dually, to reconstruct a table column, look for the concept C which label [ ] contains the attribute name heading the column, and mark with a cross the row of each one of the objects of the extent of C. For example, to reconstruct the column named Vlinf in the cross table, mark the rows corresponding to the concept R extent: {Case2, Case3, Case7, Case8}. Figure 3 shows the complete extent and intent of the formal concepts in the diagram of Figure 2.

Besides the hierarchical conceptual clustering of the cases, the concept lattice provides a set of implications between attributes: *dependence rules* [5]. A dependence rule between two attribute sets (written $M1 \rightarrow M2$, where $M1, M2 \subseteq M$) means that any object having all attributes in $M1$ has also all attributes in $M2$. We can read the dependence rules in the graph as follows:

- Each line between nodes labelled with attributes means a dependence rule between the attributes from the lower node to the upper one.
- When there are several attributes in the same label it means that there is a co-appearance of all these attributes for all the cases in the sample.

We can read the following set of dependence rules in Fig. 2. Besides other rules can be obtained by transitivity:
{Adjgms → ncms ; Vlinf → ncfs ; Vlinf → prep; Adjgfs → ncfs ;
Artdfs → ncfs ; Adjgfs → artdms; Artdfs → artdms ; Vlfi3s → vlinf ;
Vlfi3s → adjgfs; Vlfi3s → artdfs ; Artdns → vlinf ; Artdns → Adjgms;
Artdns → ncms ; Artdns → ncfs ; Artdns → prep }

	EXTENT	**INTENT**
BOT	CASE7	ARTDFS ARTDMS ARTDNS ADJGFS ADJGMS VLFI3S VLINF NCFS NCMS PREP.
χ	CASE6,CASE7	ARTDFS ARTDMS ADJGFS ADJGMS NCFS NCMS
δ	CASE1,CASE7	ARTDFS ARTDMS ADJGFS NCFS NCMS PREP.
β	CASE2,CASE7	ARTDFS ARTDMS ADJGFS VLFI3S VLINF NCFS PREP
α	CASE3,CASE7	ARTDNS ADJGMS VLINF NCFS NCMS PREP.
Z	CASE5,CASE7	ARTDFS ARTDMS ADJGMS NCFS NCMS PREP.
Y	CASE7,CASE8	ARTDMS ADJGFS ADJGMS VLINF NCFS NCMS PREP
X	CASE1,CASE2,CASE7	ARTDFS ARTDMS ADJGFS NCFS PREP
W	CASE2,CASE7,CASE8	ARTDMS ADJGFS VLINF NCFS PREP
V	CASE3,CASE7,CASE8	ADJGMS VLINF NCFS NCMS PREP.
U	CASE6,CASE1,CASE2,CASE7	ARTDFS ARTDMS ADJGFS NCFS
T	CASE1,CASE2,CASE5,CASE7	ARTDFS ARTDMS NCFS PREP
S	CASE1,CASE2,CASE7,CASE8	ARTDMS ADJGFS NCFS
R	CASE2,CASE3,CASE7,CASE8	VLINF NCFS PREP
Q	CASE3,CASE5,CASE7,CASE8	ADJGMS NCFS NCMS PREP
P	CASE4,CASE6,CASE5,CASE7,CASE8	ARTDMS ADJGMS NCMS
O	CASE6,CASE1,CASE2,CASE5,CASE7	ARTDFS ARTDMS NCFS
N	CASE6,CASE1,CASE2,CASE7,CASE8	ARTDMS ADJGFS NCFS
M	CASE6,CASE3,CASE5,CASE7,CASE8	ADJGMS NCFS NCMS
L	CASE1,CASE2,CASE5,CASE7,CASE8	ARTDMS NCFS PREP
K	CASE3,CASE5,CASE7,CASE8,CASE9	ADJGMS NCMS PREP
J	CASE6,CASE1,CASE2,CASE5,CASE7,CASE8	ARTDMS NCFS
I	CASE1,CASE3,CASE5,CASE7,CASE8,CASE10	NCFS NCMS PREP
H	CASE4,CASE6,CASE1,CASE2,CASE5,CASE7,CASE8	ARTDMS
G	CASE4,CASE6,CASE3,CASE5,CASE7,CASE8,CASE9	ADJGMS NCMS
F	CASE6,CASE1,CASE3,CASE5,CASE7,CASE8,CASE10	NCFS NCMS
E	CASE1,CASE2,CASE3,CASE5,CASE7,CASE8,CASE10	NCFS PREP
D	CASE1,CASE3,CASE5,CASE7,CASE8,CASE9,CASE10	NCMS PREP
C	CASE6,CASE1,CASE2,CASE3,CASE5,CASE7,CASE8,CASE10	NCFS
B	CASE1,CASE2,CASE3,CASE5,CASE7,CASE8,CASE9,CASE10	PREP.
A	CASE4,CASE6,CASE1,CASE3,CASE5,CASE7,CASE8,CASE9,CASE10	NCMS
TOP	CASE1, CASE2, CASE3, CASE4, CASE5, CASE6, CASE7, CASE8, CASE9, CASE10, CASE11, CASE12	∅

Fig. 3. Formal Concepts in the Poetry Domain

We apply FCA as a complementary way of knowledge acquisition and although it is not general knowledge about the domain, but knowledge associated to this concrete case base, it suggests concepts and rules to be added to the general domain model.

4 Retrieval over the Formal Concept Lattice

We propose an organization structure using an inductive technique over the case base, that is guided by the domain knowledge. FCA application to a case library provides a conceptual hierarchy, because it extracts the formal concepts and the hierarchical relations among them, where related cases are clustered according to their shared properties. Concepts in the lattice represent maximal groupings of cases with shared properties, and for a given query, we can access all the cases that share properties with the query at the same time so that they are grouped under the same concept [5]. The order between concepts allows us to structure the library according to the attributes describing the cases. The lower

in the graph, the more characteristics can be attributed to the cases; i.e. the more general concepts are higher up than the more specific ones.

Classification based retrieval [11,4] over a concept taxonomy is typically implemented as a three step process: first, a query is represented as an individual and it is classified or recognized in its corresponding place in a certain hierarchy; second, a number of individuals are retrieved from the most specific concepts of which this individual is an instance, and, third, one (or more) of them is chosen by applying a selection function or by the user.

Classification based retrieval on the formal concept lattice makes use of the case library to guide the search. The properties of the lattice justify how this approach always finds the *best* case without travelling through all cases, but taking advantage of the formal concepts clustering the cases.

The query is given as a sequence of words that we want to inspire our poem. In the example, the cases with the largest number of POS tag in common with the query should be retrieved. This way, during adaptation it will be easy to substitute words in the retrieved poem with words from the query without loosing syntactic correctness. Furthermore, dependence rules are used to suggest additional words to be included in the query, not only for retrieval purpose, but also to improve the quality of the resulting poem by imposing correlated substitutions.

Query formulation guided by dependencies

As we have described, FCA applied on a case library allows us to capture its specific dependencies, i.e. it detects regularities satisfied by all the cases in the library. We propose an interactive query definition process where the user is guided towards the definition of "good" queries for this case base. During the query description process, the user provides certain descriptors (POS tags) while the system proposes others using the dependence rules captured during the FCA. This begins an interaction cycle where the system requests the assistance of the user using the knowledge extracted from the case library, so the cases themselves are guiding the query formulation process. That allows an exact matching process where the retrieved cases fulfills all the query POS tags when it is possible (but not its words).

The query completion mechanism applies the dependence rules extracted from the lattice to help the user to make a good query (according to this case base). For instance, rule {ADJGMS → NCMS} indicates that every case having a general adjective that is masculine and singular goes together with a common name that is also masculine and singular. Similarly, rule {ARTDFS → NCFS } indicates that feminine singular definite articles appear together with feminine singular common nouns.

This method uses the rules to complete the user's queries and carry out a process of exact retrieval, in the sense that the retrieved cases satisfy exactly the requirements of the query employed (considering as a requisite that the syntactical categories of the words in the query must be maintained). The lattice in the figure has been constructed using as attributes of the formal context (some of) the syntactic categories of the words that make up the poems in the

cases. This process implies that the query is interpreted as the succession of the syntactic categories of the given words.

Dependence rules provide patterns to complete the query with words that are similar to the ones in the cases (which may be used to substitute them during adaptation, given that they have the same syntactic categories). The correction characteristics are determined with respect to the dependence rules that are extracted from the casebase. When completing the query by using the dependence rules, intuitively, what we are doing is to guide the user towards more specific and defined concepts in the lattice. Note, this kind of retrieval over the lattice finds all the cases where all the query descriptors appear, i.e. the cases retrieved as similar are those with the greater number of properties (POS tags) shared with the query. From the adaptation point of view, dependence rules can be seen as imposing additional adaptations induced by the words already in the query. Whenever a word in the query leads to the substitution of a poem word, the system suggests that candidates substitutes should also be provided for words linked to the substituted one by dependence rules.

After that, the system explicitly constructs an individual with the query case description, classifies it in the lattice, and retrieves individuals placed near it. We use a Description Logic (DL) system, LOOM [9], whose recognition module automatically classifies the new individual in its corresponding place in the lattice (belonging to the extent of certain formal concept). All instances classified under the most specific concept the query instance belongs to, are retrieved as similar. Moreover, as this concept will be typically specialized by other most specific concepts, the goal of the query definition module is to guide the user from general concepts towards their subconcepts. The DLs reasoning mechanisms are useful to automatically organize the concept lattice and to keep the cases automatically organized under them. Besides, the instance recognition mechanism is used for the direct retrieval of the siblings of the query individual.

Example 1:
Let us assume that the user chooses"*descansara(VLFI3S)*" (will rest) as a reference word to inspire our poem. The system (using the dependence rules VLFI3S → ADJGFS; VLFI3S → ARTDFS; VLFI3S → VLINF) will suggest to the user that she complete her query with other words having the obtained POS tags: ARTDFS, VLINF, ADJGFS. The system could also offer a list of vocabulary words with a certain POS tag so that the user chooses between them. In particular, there is only one word "la"(the, feminine and singular) with the POS tag ARTDFS[4]. That way the system can directly include this word in the query, and suggest that the user include of other words using the dependence rules: {ARTDFS → ARTDMS ; ARTDFS → NCFS }. The process continues until the user decides to stop the query formulation process and begin the classification based retrieval of cases over the lattice.

Word: descansara(VLFI3S)
Rules: VLFI3S → ARTDFS; VLFI3S → ADJGFS; VLFI3S → VLINF
- VLFI3S → ARTDFS

  Word: la(ARTDFS) (unique choice)

[4] as with ARTDMS and ARTDNS

Rules: ARTDFS → ARTDMS ; ARTDFS → NCFS
- ARTDFS → ARTDMS
  Word: el(ARTDMS) (unique choice)
  Rules: ∅
- ARTDFS → NCFS
  Word: flor(NCFS) (multiple choices)
  Rules: ∅

– VLFI3S → ADJGFS
Word: helada(ADJGFS)(multiple choices)
Rules: ADJGFS → NCFS
- ADJGFS → NCFS
  Word: noche(NCFS)[5]
  Rules: ∅

– VLFI3S → VLINF
Word: cambiar(VLINF)
Rules: VLINF → PREP ; VLINF → NCFS
- VLINF → PREP
  Word: por(PREP). Do not change the original preposition.
  Rules: ∅
- VLINF → NCFS
  Word: mirada(NCFS)

The resultant query is: "*descansara(VLFI3S) la(ARTDFS) flor(NCFS) la(ARTDFS) noche(NCFS) helada(ADJGFS) mirada(NCFS) por(PREP) el (ARTDMS) cambiar(VLINF)*" (The flower will rest the frozen night glance for a change). This query is represented as an individual that is classified below concept β in the lattice, and CASE7 AND CASE2 are retrieved as they share all the query POS tags (consult β intent in Figure 3).

Example 2:
The user begins a query with the word "maravilloso(ADJGMS)" (marvellous). The system then detects a dependency among the POS tags ADJGMS and NCMS, meaning that in this case library there are no poems that have an ADJGMS and not a NCMS. So, the query formulation process will guide the user towards these concrete cases (instances of the concept G in the lattice). To get the query individual classified below concept G, the system requires a NCMS to participate in the query. The user can provide one word with this POS tag (for example, "dia" –day–). If she does not provide a word the system will use the word that belong to the original poem. As there are no more applicable rules the user could choose between beginning with the retrieval process or giving additional words. At this point the retrieval process would create an instance that would be automatically classified under the concept G. The cases resulting from the current retrieval query are those from the G extent, i.e CASE4,CASE6,CASE3,CASE5,CASE7,CASE8,CASE9 (see Figure 3). If the user wants to choose another query descriptor, she will be guided towards more specific concepts in the lattice[6], i.e. those classified below G. If the user decides finishing

[5] As word flor has already this POS tag it is not necessary to include a different word, but it will provide more vocabulary to make substitutions.

[6] Reducing the descriptor set when needed as it was described in [5]

the query, a selection method based on similarity computation (Section 7) is applied to filter one between the seven retrieved cases. Similarity computation will take into account the rest of the word features (as the number of syllables, rhyme, accent, and so on).

Suppose now that the user gives another word: "despertar(VLINF)". The system applies the rules VLINF → NCFS; VLINF → PREP and lets the user include two additional words: "flor(NCFS)" (flower) and "de(PREP)" (of).

The resulting query "*despertar flor de dia maravilloso*" is classified below the concept V in the lattice, and cases CASE3,CASE7,CASE8 are retrieved because they all have as common features the POS tags: ADJGMS, VLINF, NCFS, NCMS, PREP (see V intent in Figure 3).

A dependence rule detects a regularity satisfied by all the cases in the library, but we cannot assure its applicability for every possible case in the domain. In this example, there are some rules representing how to build correct phrases in Spanish, for example, that an adjective always accompanies a noun ADJGMS → NCMS ; ADJGFS → NCFS. These are general rules that will be useful in this domain for every case base. But in the general case, the rules[7] mean a co- apparition of a set of attributes for all the cases in this particular case base but not representing general domain rules, i.e., they are not general rules to build correct phrases in Spanish. For example, the rule ARTDFS → ARTDMS. In any case, the rules are useful to retrieve cases over this particular case base because they represent properties that are satisfied by all the cases in this case base.

The system proposes (and not imposes) certain POS tags using the dependence rules to maintain the semantic sense of certain groups of words. For example, the rule ADJGMS → NCMS suggests changing the noun that accompanies an adjective if the latter changes. If the user does not follow the system suggestion the system will use the original word in the query (or even could randomly select a word from the vocabulary).

Complementary methods of retrieval. As we have explained in previous papers, the advantage of COLIBRI is that it allows us to experiment with different methods that solve the CBR tasks. In the method presented in this paper a retrieval process is carried out guided by the adaptation, since we guide the user to build well-structured query, which will insure retrieval of a case that guarantees a possible substitution for all the given words during adaptation. This takes place under the assumption that only the syntactic categories, and not the rest of the properties of the words (number of syllables, begins or ends with a vowel, rhyme). As a result, the metric of the poem will be disrupted, and a stronger process of revision is required at a later stage. In last year's paper [2] a different method was presented, that retrieved the cases with highest number of syntactic categories in common with the query. This was guided by the same idea of guaranteeing the required possibilities for substitution during adaptation. After the initial filtering, a measure of the similarity value that took into account the rest of the attributes of the words (such as number of syllables, begins or ends with a vowel) was applied. In this way, subsequent revision was minimized.

[7] specially in a reduced example as the one used in the paper

5 Illustration of the Rest of the CBR Cycle

To give an idea about how the whole system works, in this section we briefly introduce the adaptation and revision processes.

Lets suppose we are using the query of Example 1 from Section 4: "*descansara(VLFI3S) la(ARTDFS) flor(NCFS) la(ARTDFS) noche(NCFS) helada(ADJGFS) mirada(NCFS) por(PREP) el (ARTDMS) cambiar(VLINF)*". And the selected case to be adapted is Case7 that is the one we used to get the example POS tag subset in Section 2.2:

marchitara la rosa el viento helado
todo lo mudara la edad ligera
por no hacer mudanza en su costumbre

The adaptation process consists of substitution certain words in the retrieved poem using the words given by the query (in italic):

descansara la noche el viento helado
todo lo pintara *la flor helada*
por no *cambiar mirada* en *su* primavera

During the proposed adaptation process, we only have considered the POS tags of the substituted words and not the rest of the word properties, like the number of syllables, if a word begins or ends with a vowel (to check the *synaloephas* in the lines), or the rhyme. That is why, even though the substitutions maintain the syntactical correctness of the sentences, the metric of the poem can be damaged and the revision process is in charged of fixing the metric of the poem. In particular the revision process fixes the number of syllables of the verses and their rhyme. The original poem in the example was (before adaptation) a *Terceto* with a stanza of three lines of 11 syllables where the three lines have independent rhymes.

After adaptation, the revision process is in charge of evaluating and, if needed, repairing the proposed poem. It consists of identifying one by one the problems we have to fix and repair them. As we have described in [2] we propose a classification based evaluation method that compares the classification between the adapted case and the retrieved case. The concepts under which the retrieved case is classified in the domain model are used as declarative descriptions of the properties that should be maintained by the adapted case after transformations. In the example the retrieved case is recognized as *Poem*, the stanza is recognized as a *Terceto* and each one of its poem lines are recognized as *Endecasilabo*s.

Before substitutions, the copy of the retrieved case that will be adapted has the same classification. If substitutions provokes a change in the concepts the system recognizes for a certain individual, the evaluation task classifies this individual below a concept representing a type of problem. The revision is implemented as the process of substituting words in the adapted poem so that the detected problems can be repaired. In the example we have reused a poem without rhyme, so revision is restricted to the number of syllables of the lines.

In the example, we have substituted a word by other of the same POS tag but possibly different number of syllables. The problem type called *Syllables-count-failure* involves the domain concept *Endecasilabo*. When an individual leaves this concept it is recognized as an instance of *Syllables-count-failure*. The problem

type has associated a repair strategy that tries to put the individual representing each line, back as an instance of the goal concept *Endecasilabo*.

The first and third lines have 12 syllables instead of 11. In the first line the words *helado* and *viento* are candidates to be substituted as they do not belong to the query. In order to find a replacement for this word, we search for a word with the same POS tag and one less syllable than the candidate one. For example, the word *triste (ADJGMS)* meets those requirements when substituting *helado*. The next loop tries to repair the number of syllables of the third poem line. The candidate to be substituted is the word *primavera (NCFS)* that is substituted by a word with the same POS tag and three syllables. For example, the word *juventud*. After these substitutions, the number of syllables are automatically recomputed and the individuals representing the lines are reclassified as instances of the concept *Endecasilabo*. The resulting poem is:

descansara la noche el viento triste
todo lo pintara *la flor helada*
por no *cambiar mirada* en *su* juventud[8]

6 Conclusions

FCA has been successfully used in many data analysis applications. We use it as a complementary technique to enrich the domain taxonomy, providing an alternative organization of the case library that facilitates a direct and guided access to the cases. FCA application provides an internal sight of the case base conceptual structure and allows finding patterns and regularities among the cases.

We have described the application of FCA to a CBR system to generate well-formed Spanish poetry versions of texts provided by the user. The knowledge acquired through FCA is used during the query formulation and retrieval tasks. Dependence rules provide patterns to complete the query with words that are similar to the ones in the cases (which may be used to substitute them during adaptation, given that they have the same syntactic categories). Furthermore, dependence rules require new words that impose further adaptations.

Although our approach have clear advantages we have illustrated in this paper, we have identified certain limitations of our proposal. For example, FCA does not allow to extract disjunctive rules like ARTDMS → NCMS OR VLINF, meaning that ARTDMS words always appear together with a NCMS or a VLINF. We think that kind of rules would be also very useful but FCA detects POS tags co-occurrences for *all* the cases in the casebase. That is also the main reason for other limitation: the strong noise sensibility of this technique. Besides, our proposal only consider if a POS tag appears or does not appear in a case, but it does not take into account the situations where several words with the same POS tag belong to the same poem. They are all considered as one POS tag occurrence.

[8] The night will ease the sorrowful wind,//and all things will be tinted by the frozen bloom// so as not to exchange a glance in its prime

Even considering these limitations we have illustrated the usefulness of the lattice structure obtained by FCA application, to organize the case base and support an adaptation guided retrieval process that takes into account the semantic meaning of the parts that compose the poems.

References

1. G. Birkhoff. Lattice theory, third editon. In *American Math. Society Coll. Publ. 25, Providence, R.I.* 1973.
2. B. Díaz-Agudo, P. Gervás, and P. A. González-Calero. Poetry generation in COLIBRI. In *Procs 6th European Conference on Case Based Reasoning(ECCBR'02). Springer LNAI 2416, pp.73–87.* 2002.
3. B. Díaz-Agudo and P. A. González-Calero. An architecture for knowledge intensive CBR systems. In E. Blanzieri and L. Portinale, editors, *Advances in Case-Based Reasoning – (EWCBR'00).* Springer-Verlag, Berlin Heidelberg New York, 2000.
4. B. Díaz-Agudo and P. A. González-Calero. Classification based retrieval using formal concept analysis. In *Procs. of the (ICCBR 2001).* Springer-Verlag, 2001.
5. B. Díaz-Agudo and P. A. González-Calero. Formal concept analysis as a support technique for CBR. In *Knowledge-Based Systems, 14 (3–4), June,.* Elsevier. (ISSN:0950–7051), pp. 163–172., 2001.
6. B. Díaz-Agudo and P. A. González-Calero. CBROnto: a task/method ontology for CBR. In S. Haller and G. Simmons, editors, *Procs. of the 15th International FLAIRS'02 Conference (Special Track on CBR*, pages 101–106. AAAI Press, 2002.
7. B. Ganter and R. Wille. Formal concept analysis. mathematical foundations. 1997.
8. F. S. León. Corpus resources and terminology extraction project (mlap-93/20), 1993.
9. R. Mac Gregor and R. Bates. The loom knowledge representation language. ISI Reprint Series ISI/RS-87-188, University of Southern California, 1987.
10. H. Muñoz-Avila and J. Hullen. Retrieving cases in structured domains by using goal dependencies. In *CBR Research and development(ICCBR'95).* Springer-Verlag, 1995.
11. A. Napoli, J. Lieber, and R. Courien. Classification-based problem solving in cbr. In I. Smith and B. Faltings, editors, *Advances in Case-Based Reasoning – (EWCBR'96).* Springer-Verlag, Berlin Heidelberg New York, 1996.
12. B. Smyth and M. T. Keane. Adaptation-guided retrieval: Questioning the similarity assumption in reasoning. *Artificial Intelligence*, 102(2):249–293, 1998.
13. R. Wille. Conceptual lattices and conceptual knowledge systems. In *Computers and Mathematics with Apps, 23 (6–9)*, pages 493–515. 1992.