Project-Team LIS

Logical Information Systems

Rennes

Activity Report

2012

1 Team

Head of the team
Olivier Ridoux, Professor, Université Rennes 1

Administrative assistant
Élodie Lequoc

Université Rennes 1 personnel
Yves Bekkers, Professor
Annie Foret, Assistant Professor, HDR
Sébastien Ferré, Assistant Professor

Insa personnel
Mireille Ducassé, Professor
Peggy Cellier, Assistant Professor
Benjamin Sigonneau, Research Engineer (part time: 70%)

Visitors Sebastian Rudolph, visiting professor from Karlsruhe Institute of Technology (KIT), March 2012
Preeti Dahiya, visiting undergraduate student from B.K. Birla Inst. of Engineering and Technology (India), September-October 2012

PhD students
Mouhamadou Ba, ARED/Insa grant, since October 2012
Soda Marème Cissé, MENRT/Rennes 1 grant, since October 2012
Alice Hermann, *MENRT*/Insa grant, since October 2009 (PhD defended on 17th December)

Associate members
Erwan Quesseveur, Assistant Professor, Université Rennes 2, Associate Member
François Le Prince, President of ALKANTE and Associate Professor, Université Rennes 2, Associate Member

2 Overall Objectives

2.1 Overview

The LIS team aims at developing *formal* methods for handling complex data sets in a *flexible* and *precise* way. "Flexible" means that the content determines the shape of the container. Very often, it is the opposite that is observed; e.g., the tree-like shape of a hierarchical file system enforces the tree-like shape of software packages. "Precise" means that any subset of the data set can be easily characterized. Again, it is the opposite that is often observed; e.g., in a hierarchical file system only sub-trees can be easily characterized. More and more information is available on the Web, and more and more information can be stored on a single machine. However, whereas the related low-level technology is developing, and performance is increasing, little is done for organizing the ever-growing amount of information. Therefore, the LIS team addresses the issues of organizing and querying information in general. The solutions are to be both formal and practical. Operational issues such as index technologies are important, but we are convinced that their scope is too limited to solve the crucial issues.

At a formal level, *queries* and *answers* are two key notions. It is nowadays standard to consider queries as logical formulas and answers as special models of queries. Computing the preferred model of a query in some context is conceptually easy, and it warrants flexibility. However, the opposite is not that easy in general; given a subset of the data, how can we compute a query of which it is a model? Given two different subsets of the data, how can we compute a query that explains the difference? Knowing this would warrant precision. The LIS team proved that *formal concept analysis* (FCA [GW99]) is a powerful framework for analyzing $\langle query, answer \rangle$ pairs. *Formal concepts* formalize the association between a query and its answers. Formal concepts are structured into a lattice which provides navigation links between concepts.

However, standard FCA cannot deal with queries considered as logical formulas (recall that this is the key for flexibility). Therefore a variant of FCA for logical description has been developed [5] altogether with the generic notion of *Logical information system* (LIS) that provided a reconstruction of all information system operations based on logical concept analysis. In particular, some data-mining operations are native in LIS [5, 3].

The mottoes of the LIS research are:

1. *Never impose* a priori *a structure on information.* E.g., do not use hierarchical structures. Imposing *a priori* a structure causes the *tyranny of the dominant decomposition*[TOHS99]. For instance, the usual class-based organisation of source code makes highly visible the connections between methods of the same class, but masks the possible connections between methods in different classes.

   Instead, consider pieces of information as a bulk. Structure should emerge *a posteriori* from the contents or the point of view. As a consequence, updating the contents may change the structure: we accept it.

[GW99] B. Ganter, R. Wille, *Formal Concept Analysis — Mathematical Foundations*, Springer, 1999.

[TOHS99] P. Tarr, H. Ossher, W. Harrison, S. Sutton, "N Degrees of Separation: Multi-Dimentional Separation of Concerns", *in: ICSE*, IEEE Computer Society, p. 107–119, 1999.

2. *Consider every possible rational classification, and permit changes at any time.* Here, rational means that what makes a piece of information belong or not to a class depends on the very piece of information, not on other pieces. The concept lattice induced by FCA is precisely a means to grasp all possible rational classifications.

3. *Rare events are as important as the frequent ones.* One cannot say *a priori* if a piece of information is interesting because it presents a frequent pattern, or because it presents a rare pattern.

   So, rare events must not be masked by statistical artefacts. Statistics is not forbidden, but it is only a complement of a symbolic logic approach.

4. *Queries should be possible answers.*

   In usual information systems (say relational databases or Web browsers) there is a strict dichotomy between queries (they are intensional expressions), and answers (they are strictly extensional expressions, i.e. sets of things). We contend that a good answer must be a mix of extensional and intensional answers. E.g. the good answer to "I would like to buy a book" is seldom the whole catalog of the bookshop; it is more relevant to answer such a query with other queries, like "Is this for a child" or "Do you prefer novels or documents".

   Note that hierarchical file systems already do that. Queries (i.e., *filepaths*) yield answers that contain other queries (i.e., *sub-directories*). One of the LIS achievements is a formalization of this behaviour that does not rely on an *a priori* hierarchical structure.

Our research is intended to be *vertical* in the sense that all aspects of information systems are of interest: design, implementation, and applications.

On the implementation side, the LIS team develops systems that present the LIS abstraction either at the file system level [8] or at the user level [6].

On the application side, the LIS team explores the application of LIS to *Geographical information systems* (GIS). The intuition here is that the traditional layered organization of information in GIS suffers a rigid structure of thematic layers. Moreover, GIS applications usually cope with highly heterogeneous information and large amount of data; this makes them an interesting challenge for LIS. The team also works on a data-mining interpretation of bug tracking. In this case, the intuition is that pieces of information relevant to software engineering, e.g. programs, specifications or tests, can be explored very systematically by a LIS. More generally, applications to software engineering are important for the team. A recent trend of application is the assistance to group decision and negotiation. In particular LIS provide new technological support to *social choice*. For example, in committee decision making, navigating with LIS in the facts recorded in a context allows decision makers to treat all candidates in a fair way [2].

2.2 Key Issues

In its current state, LIS studies the following key issues:

- The LIS formalism is generic w.r.t. the logic used for describing pieces of information.

  What are the appropriate logics for the application fields that we have chosen? (GIS and error localization) Do we need a brand new logic for every application, or is there something that different applications can share?

- Genericity of LIS w.r.t. logic opens the door for creating ad hoc logics for describing pieces of information of an application. We already have proposed the framework of *logic functors* for helping a user build safely ad hoc logics. Logic functors are certified logic components that can be composed to form certified implementations of a logic.

  What are the useful logic functors? How can we be sure that a toolbox of logic functors is complete for a given purpose?

  Can the idea of certified composition be applied to another domain? Given a domain *foo*, *foo* functors would be certified *foo* components that can be assembled to form certified implementations of *foo* systems.

  Is it possible to certify other properties than meta-logical properties? E.g. is it possible to characterize complexity, or other non-functional properties like security?

- A family of non-commutative logics has developed over the years in the domain of computational linguistics, e.g. Lambek logic, pregroups. As for LIS, a great amount of creativity is expected for extending this family with ad hoc logics that would tackle fine-grained linguistics phenomena.

  Is it possible to build up an implementation of these logics using logic functors?

  Some LIS applications deal with objects that are sequential by nature (say, texts).

  Can these non-commutative logics primarily developed for computational linguistics help in LIS applications?

- Hierarchical file systems have a preferred metaphor which is the tree.

  What is the proper metaphor for LIS?

  The tree is also the graphical metaphor of hierarchical file systems.

  What is the graphical metaphor for LIS?

  Knowing this is crucial for the acceptance of LIS in end-user applications.

- Geographical information systems also suffer the *tyranny of the dominant decomposition*. Here, the dominant decomposition is in rigid thematic layers that inherit from plastic sheets of ancient map design. These layers are omnipresent in the design and interface of GIS applications.

  How can LIS abstract these layers, and still display layers when needed?

  Mining geographical information is difficult because of the layers and because it must cope with complex spatial relations.

  What is the proper modeling of these relations that will permit efficient LIS operations, including data-mining?

- Up to now, mining execution traces for bug tracking has used poor trace representations and ad hoc algorithms.

  How can the theoretical and practical framework of LIS help benefit from the wide range of information of program development environments?

- The file system implementation of LIS can handle around 1 million elementary pieces of information, which corresponds approximately to a full homedir with 10 to 20 thousands files. This is rather small compared to relational database capabilities, but already large compared to other approaches based on formal concept analysis.

  How can it handle more? Can we reach 100 million in the next few years?

3 Scientific Foundations

3.1 Logics for Information Systems

Keywords: Syntax, interpretation, semantics, subsumption.

Glossary :
Syntax Definition of the well-formed statements of a language. Statements are finite.
Interpretation Complete description of a world. Interpretations can be arbitrary mathematical constructs, and so can be infinite. Interpretations are models of statements, namely the worlds in which the statement is true. Statements are features of interpretations, namely the statements that are true in the world.
Semantics A binary relation between syntactic statements and interpretations.
Subsumption A relation which states that a property is more specific than another property.

Logic is the core of Logical Information Systems. However, this does not say everything because every particular usage of logic is also a point of view on logic. For instance, logic in Logic Programming is not the same as in Description Logics. This section describes the point of view on logic from information systems.

Logic is a wide domain that is concerned with formal representation and reasoning. The point of view on logic in logical information systems can be characterized by two things. Firstly, we are interested in the individual description of objects (e.g., files, pictures, program functions or methods), so that we need to represent concrete domains and data structures. This entails two levels of statements: (1) statements about objects, and (2) statements about the world (e.g., ontologies and *subsumption*). Subsumption helps to decide when an object is an answer to a query. Secondly, we need automated reasoning facilities as the subsumption must be decided between any object and a query in information retrieval. This forces us to only consider decidable logics, unless consistency or completeness are weakened.

Properties of a Logic A characteristic of logic is the ability to derive new statements from known statements. Such a derivation is valid w.r.t. semantics only if every model of the known statements is also a model of the new statements. This ability opens the room for *reasoning*, i.e. the production of valid statements by working at the syntactical level only. Reasoning is

formalized by *inference systems* (e.g., axioms and rules). An inference system is *consistent* if it produces only valid statements; it is *complete* if it produces all valid statements. Reasoning is *decidable* if a consistent and complete inference system can be realized by an algorithm.

Examples of Logics for Information Systems Proposition logic is a possible logic for an information system, but it needs a lot of encoding for handling structured information. Instead, non-standard logics have been defined for some structured domains.

A large family of logics that comes into our scope is the family of Description Logics (DL) [Bra79,CLN98], which have been widely studied, implemented, and applied in knowledge and information management. Moreover, their semantic structure is especially well-suited to be used in a LIS. The semantics of proposition logic is often exposed in terms of truth values and truth tables. To the contrary, the semantics of description logic is defined in terms of sets of objects that are close to answers to a query. DL are, therefore, of a special interest for the LIS team.

Another family of interest is *categorial grammars*. Many substructural logics come into this scope, among which non-commutative linear logic or Lambek Calculus[Lam58] that handle various concatenation principles (or ordered conjunction) in categorial grammars where logic is used both for attaching formulas to objects and for parsing seen as deduction.

At an empirical level, the categorial approach comes very close to the LIS approach. Categorial grammars correspond to LIS contents, because they both attach formulas to objects, and sentence types correspond to queries. The difference is that the answer to a LIS query is an unordered set, whereas a sentence generated by a categorial grammar is an ordered sequence. We expect a cross-fertilization of both theories in the future, especially in the LIS applications where the objects are naturally ordered.

3.2 Concept Analysis

Keywords: Objects, descriptors, context, instance, property, extension, intension, concept.

Glossary :

Objects A set of distinguished individuals.

Descriptors A set of distinguished properties.

Context A set of objects associated with descriptors.

Instance An object is an *instance* of a descriptor if it is associated with it in a given context.

Property A descriptor is a *property* of an object if it is associated with it in a given context.

[Bra79] R. J. Brachman, "On the Epistemological Status of Semantic Nets", *in: Associative Networks: Representation of Knowledge and Use of Knowledge by Examples*, N. V. Findler (editor), Academic Press, New York, 1979.

[CLN98] D. Calvanese, M. Lenzerini, D. Nardi, "Description Logics for Conceptual Data Modeling", *in: Logics for Databases and Information Systems*, J. Chomicki, G. Saake (editors), Kluwer, p. 229–263, 1998.

[Lam58] J. Lambek, "The Mathematics of Sentence Structure", *American Mathematical Monthly 65*, 1958, p. 154–170.

Extension The *extension* of a collection of descriptors is the set of their common instances. Extent is a synonym.

Intension The *intension* of a collection of objects is the set of their common properties. Intent is a synonym.

Concept Given a context, and extensions and intensions taken from it, a *concept* is a pair (E, I) of an extension E and an intention I that are mutually complete; i.e., I is the intention of the extension, and E is the extension of the intention.

Formal Concept Analysis Formal Concept Analysis (FCA) is part of the mathematical branch of applied lattice theory [Bir40,DP90]. It can be seen as a reformulation by Wille of Galois lattices [BM70] that emphasizes lattices as conceptual hierarchies [Wil82]. The mathematical foundations of FCA have been extensively studied by Ganter and Wille [GW99].

FCA mainly aims at the automatic construction of *concepts* and their classification according to a generalization ordering, given a flat representation of data. The adjective *formal* means that concepts are given a mathematical definition, which reflects the usual philosophical meaning of a "concept". The basic notions of FCA are those of *formal context*, and *formal concept*.

A *formal context* is a binary relation between a set of objects, and a set of attributes. Through this relation attributes can be seen as properties of objects, and reciprocally, objects can be seen as instances of attributes. This is a very general settings that applies to various domains such as data analysis, information retrieval, data-mining or machine learning. In all these domains, the objects of interest are described by sets of attributes, and the objective is to relate in some way sets of objects and sets of attributes. In information retrieval a set of attributes is a query, whose answers is a set of objects. In machine learning a set of objects is a set of positive examples, whose characterization is a set of attributes.

A *formal concept* is the association of a set of objects, the *extent*, and a set of attributes, the *intent*. This comes close to the classical definition of concept in philosophy, but in FCA the relationship between extent and intent is formally defined. The extent must be the set of instances shared by all attributes of the intent; and the intent must be the set of properties shared by all objects in the extent.

The fundamental theorem of FCA says that the set of all concepts forms a complete lattice when they are ordered according to the set inclusion on extents (or intents). This is called the *concept lattice*, and it can be computed automatically from the formal context. The concept lattice is the structure that is implicit in any formal context. It contains all the information contained in the formal context; the latter can be rebuilt from the former. In data analysis, the concept lattice permits a flexible classification of data (where a concept is a class), because

[Bir40] G. Birkhoff, *Lattice Theory*, American Mathematical Society, 1940.

[DP90] B. A. Davey, H. A. Priestley, *Introduction to Lattices and Order*, Cambridge University Press, 1990.

[BM70] M. Barbut, B. Monjardet, *Ordre et classification — Algèbre et combinatoire (2 tomes)*, Hachette, Paris, 1970.

[Wil82] R. Wille, *Ordered Sets*, Reidel, 1982, ch. Restructuring lattice theory: an approach based on hierarchies of concepts, p. 445–470.

[GW99] B. Ganter, R. Wille, *Formal Concept Analysis — Mathematical Foundations*, Springer, 1999.

concepts are not organized as a strict hierarchy. In information retrieval and data-mining it is used as a search space for answers.

Logical Concept Analysis In Formal Concept Analysis (FCA) object properties are restricted to Boolean attributes. In many applications there is a need for richer properties, where properties are not independent. For instance, if a book has been published in 2000, it can be given the property `year = 2000`, and has then the implicit properties `year in 1990..2000` and `year in 2000..2010`. This means that properties are statements about objects that can be subject to reasoning, exactly like logical statements. Other examples of useful properties are strings and string patterns, spatial descriptions for locating objects, or patterns over the programming type of functions and methods.

FCA has been extended by other authors to handle multi-valued contexts [GW99], but this extension takes the form of a preprocessing stage that results in a standard formal context, and forgets all logical relations between properties. Moreover it is limited in practice to valued attributes with finite domains of attributes. In 2000 we proposed a logical generalization of FCA, named Logical Concept Analysis (LCA) [5], that is the abstraction of FCA w.r.t. object descriptions and concept intents. This makes LCA an abstract component, and makes FCA the composition of LCA with a logic component. LCA makes the theory of concept analysis easily reusable in various applications.

For good composability of LCA and logics, they must agree on the specification of logics. What LCA needs from a logic is:

- a language of formulas (or statements), L, for the representation of object descriptions and concept intents,
- a procedure, $\sqsubseteq$, for deciding the subsumption between 2 formulas; $\sqsubseteq$ means "is subsumed by", "is more specific than", "entails",
- a procedure, $\sqcup$, for computing the least common subsumer of 2 formulas; it is a kind of logical disjunction,
- a formula, $\bot$, that is the most specific according to subsumption (logical contradiction).

This specification provides everything required to extend fundamental results of FCA to LCA (formal context, extent, intent, concept, complete lattice of concepts). For information retrieval and the expression of queries, it is useful to add, to this specification, operations such as logical conjunction, and logical tautology (the most general formula).

Any formal context defines a logic whose subsumption relation is isomorphic to the concept lattice that is derived from the formal context. An interesting result is that the *contextualized logic* (the logic defined by the logical context) is a refinement or extension of the logic used by LCA. Everything true in the logic is also true in the contextualized logic (because it is *eternal truth*); and everything true only in the contextualized logic says something that is true in the context, but not in general (because it is *instant truth*). Thus, contextualized logic forms the basis for data-mining and machine learning tasks, whose aim is to discover outstanding regularities in a given context [5, 3].

[GW99] B. Ganter, R. Wille, *Formal Concept Analysis — Mathematical Foundations*, Springer, 1999.

3.3 Logical Querying, Navigation, and Data-mining

Keywords: Querying, navigation, data-mining.

Glossary :

Querying The process that takes a query (e.g., a logical formula), and returns the collection of objects that satisfy the query (e.g., the extent of the query).

Navigation The process of moving from place to place, where each place indicates objects they contain (i.e. *local objects*) and other places where it is possible to move (i.e. *neighbouring places*).

Data-mining The process of extracting outstanding regularities from data (e.g., a context) hoping to discover new and useful knowledge.

In most information systems, querying and navigation are two disconnected means for information retrieval. With querying, users formulate queries which belong to more or less complex querying languages, from simple words as in Google to highly structured languages like SQL. The system returns a set of answers to the query. This permits expressive search criteria over large amounts of data, but lacks interactivity because the dialogue is only one-way. If the answers are not satisfying, users have to imagine new queries and formulate them, which requires *a priori* knowledge of both querying language and data. With navigation, users move from place to place following links. The most common systems are folder hierarchies (e.g., file systems, bookmarks, emails), and hypertext. As opposed to querying, navigation provides interactivity by making suggestions at each step, but offers limited expressivity because navigation structures are rigid. In a hierarchy, selection criteria are presented in a fixed order. For instance, if pictures are classified first by date, then by type, one cannot easily find all landscape pictures.

The need for combining querying and navigation has already been recognized. Most proposals, however, are unsatisfying. Indeed, either querying and navigation cannot be mixed freely in a same search, or consistency of querying is not maintained. An example of the former is SFS [GJSO91], once a querying step is done, there is no more navigation. An example of the latter is HAC [GM99], some query answers may not satisfy the query. A proposal based on FCA has not these drawbacks [GMA93], and we have generalized it to work within LCA, which allows us to use logical formulas for object description and queries [5]. Logic brings expressivity in querying, and concept analysis brings the concept lattice as a navigation structure (i.e., navigation places are formal concepts). The advantages of this navigation structure is that (1) it is automatically derived from data, the logical context (see motto 1), (2) it is complete as navigation alone makes it possible to reach any object (see motto 3), and (3) it is flexible because selection criteria can be chosen in any order, thus allowing user to express

[GJSO91] D. K. Gifford, P. Jouvelot, M. A. Sheldon, J. W. J. O'Toole, "Semantic file systems", *in : ACM Symp. Operating Systems Principles*, ACM SIGOPS, p. 16–25, 1991.

[GM99] B. Gopal, U. Manber, "Integrating Content-Based Access Mechanisms with Hierarchical File Systems", *in : third symposium on Operating Systems Design and Implementation*, USENIX Association, p. 265–278, 1999.

[GMA93] R. Godin, R. Missaoui, A. April, "Experimental Comparison of Navigation in a Galois Lattice with Conventional Information Retrieval Methods", *International Journal of Man-Machine Studies 38*, 5, 1993, p. 747–767.

their preferences (see motto 2). Querying and navigation can be freely mixed (see motto 4) in a same search because every logical formula points to a formal concept, and every formal concept is labelled by a logical formula. Put concretely, this means that a user can at each step of his search: either modify by hand the current query and reach a new place, or follow a suggested link that will modify the current query and reach a new place.

The critical operation is the computation of navigation links, which correspond to edges in the concept lattice. Indeed, the worst-case time complexity for computing the concept lattice is exponential in the number of objects, which makes it intractable in most interesting cases. We demonstrated both in theory and practice that this computation is not necessary. A key feature of LIS is that its semantics is expressed in terms of LCA, though it is not required to actually build the concept lattice. This is opposed to most (all?) previous proposals for using LCA in information retrieval.

The concept lattice upon which our navigation is based is also a rich structure for data-mining and machine learning [Kuz04]. Here again, we have combined existing techniques with logic [5, 3], and applied them to the automatic classification of emails [FR02], and the prediction of the function of proteins from their sequence [3].

3.4 Genericity and Components

Keywords: Abstraction, reusability, composability, component.

Glossary :

Abstraction a mechanism and practice to reduce and factor out details so that one can focus on few concepts at a time.

Reusability the likelihood a segment of structured code can be used again to add new functionalities with slight or no modification. Reusable code reduces implementation time, it increases the likelihood that prior testing and use has eliminated bugs and it localizes code modifications when a change in implementation is required.

Composability a system design principle that deals with the inter-relationships of components. A highly composable system provides recombinant components that can be selected and assembled in various combinations to satisfy specific user requirements.

Component a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties.

The application scope of Logical Information Systems is very large, and we do not expect that one design (e.g., one logic) will fit all possible applications. That is why we emphasize genericity, and we use the plural in "logical information systems". The need for genericity is not limited to theoretical results and design, but extends to the concrete implementation of LIS.

[Kuz04] S. O. Kuznetsov, "Machine Learning and Formal Concept Analysis.", *in : Int. Conf. Formal Concept Analysis*, P. W. Eklund (editor), *LNCS 2961*, Springer, p. 287–312, 2004.

[FR02] S. Ferré, O. Ridoux, "The Use of Associative Concepts in the Incremental Building of a Logical Context", *in : Int. Conf. Conceptual Structures*, G. A. U. Priss, D. Corbett (editor), *LNCS 2393*, Springer, p. 299–313, 2002.

Genericity requires programming facilities for *abstraction*, *composability*, and *reusability* of *software components*.

In LIS, abstraction is of the upper importance in the design of logical concept analysis; LCA is an abstraction of FCA. It is also at the heart of the *logic functor* framework and of its implementation; a logic functor is an abstraction of a logic (see Section 3.5). Reusability and composability are the expected outcomes of this framework. It is expected to make things easier to the designer of a LIS application. Composability is also at the heart of the very notion of formal context, and thus at the heart of concept analysis. Indeed, the flat structure of formal concepts makes it trivial to extend a context or merge two contexts, and the burden of giving a structure to the context is left to the construction of the concept lattice.

A generic implementation of LIS can be seen as a central component that is parameterized by several application-dependent components: at least a logic, and a transducer for importing data. These parameter components can be linked at compilation time (plugins). The central component as well as parameter components can themselves be the result of the composition of smaller components.

3.5 Logic Functors

Keywords: Logics, genericity, composability.

The genericity w.r.t. logic implies that for every new application a logic has to be found for describing objects in a logic context. Either a suitable logic is already known, or it must be created. Creating a logic requires designing a syntax, a semantics, algorithms for subsumption and other procedures, and proving that these algorithms are correct w.r.t. semantics. This definitely requires logic expertise and programming skills, especially for the subsumption procedure that is a theorem prover for which consistency and completeness must be proven. However, application developers and logic experts are likely to be different persons in most cases. Moreover, creating new logics from scratch for each application is unsatisfying w.r.t. reusability as these logics certainly share common parts. For instance, many applications need propositional reasoning, only changing the notion of what is a propositional variable.

We introduced high-level logic components, named *logic functors* [4], in order to make the creation of a new logic the mere composition of abstract and reusable components. All logics share a common specification that contains all useful procedures (e.g., subsumption); logic functors are functions from logics to logics, implemented as parameterized modules. Some logic functors take no parameter, and provide stand-alone but reusable logics: this is the case of concrete domains such as integers or strings. Other logic functors take one or several logics as parameters. For instance the functor `Prop(X)` is propositional logic abstracted over its atoms. This makes it possible to replace atoms in propositional logic by the formulas of another logic (e.g., valued attributes, terms from a taxonomy).

Logics are built by applying logic functors to sub-logics, which can themselves be defined as a composition of logic functors. For instance, the propositional logic where atoms are replaced by integer-valued attributes (and allowing for integer intervals) can be defined by the expression

```
L = Prop(Set(Prod(Atom,Interval(Int)))).
```

This results in a concrete software component `L` that is fully equipped with implementations of the logic specification procedures. This component can then be composed itself with LCA or a LIS system.

3.6 Categorial Grammars

Keywords: Categorial grammar, identification in the limit.

Categorial grammars are used for natural language modeling and processing; they mainly handle syntactic aspects, but Lambek variants also have a close link with semantics and Lambda-calculus. Formally, a *categorial grammar* is a structure $G = (\Sigma, I, S)$ where: Σ is a finite alphabet (the words in the sentences); I is a function that maps a finite set of types to each element of Σ (the possible categories of each word, a lexicon); S is the *main type* associated to correct sentences. A *k-valued categorial grammar* is a categorial grammar where, for every word $a \in \Sigma$, $I(a)$ has at most k elements. A *rigid categorial grammar* is a 1-valued categorial grammar. Rigidity is a useful constraint to get learnable subclasses of grammars (and related algorithms).

Each variant of categorial grammar formalism is also determined by a derivability relation on types $\vdash$ (which can be seen as a subcase of *linear logic* deduction in the case of Lambek grammars). Given a categorial grammar $G = \langle \Sigma, I, S \rangle$, a sentence w on the alphabet Σ belongs to the language of G whenever the words in w can be assigned by I a sequence of types that derive (according to $\vdash$) the distinguished type S.

A simplified example is $G_1 = (\Sigma_1, I_1, S)$ with $\Sigma_1 = \{John, Mary, likes\}$ $I_1 = \{John \mapsto \{N\}$, $Mary \mapsto \{N\}$, $likes \mapsto \{N \setminus (S/N)\}\}$ the sentence "John likes Mary" belongs to the language of G_1 because $N, N\backslash(S/N), N \vdash S$ due to successive applications of the two elimination rules : $X, X \setminus Y \vdash Y$ and $Y, Y/X \vdash Y$. Type constructors / and $\setminus$ can be seen as oriented logic implications, the elimination rules are analogues of the "Modus Ponens" logic rule. An interesting issue is how the underlying rules or logics may compose (this is the design of logic functors) to deal with more fine-grained linguistic phenomenon.

Since they are lexicalized, such grammar formalisms seem well-adapted to automatic acquisition or completion perspectives. Such studies are performed in particular in Gold's paradigm.

Identification in the limit in the model of Gold consists in defining an algorithm on a finite set of (possibly structured) sentences that converges to obtain a grammar in the class that generates the examples. Let $\mathcal{G}$ be a class of grammars that we wish to learn from positive examples; let $\mathcal{L}(G)$ denote the language associated with a grammar G; a *learning algorithm* is a function ϕ from finite sets of (structured) strings to $\mathcal{G}$, such that for any $G \in \mathcal{G}$ and $\langle e_i \rangle_{i \in \mathbb{N}}$ any enumeration of $\mathcal{L}(G)$, there exists a grammar $G' \in \mathcal{G}$ such that $\mathcal{L}(G') = \mathcal{L}(G)$ and $n_0 \in \mathbb{N}$ such that $\forall n > n_0$ $\phi(\{e_0, \ldots, e_n\}) = G'$.

Categorial grammars have close connections with logic. We therefore consider different ways of connecting computational linguistic data and LIS, such as implementing parameterized pregroups as a logic functor [7].

4 Application Domains

4.1 Geographical Information Systems

Participants: Soda Marème Cissé, Olivier Ridoux, Sébastien Ferré, Erwan Quesseveur, François, Le Prince.

Geographical Information Systems (GIS) is an important, fast developing domain of Information technology, and it is almost absent from INRIA projects. It is especially important for local communities (e.g. region and city councils).

Geographical information systems [LT92] handle information that are localized in space (*geolocalized*). GIS form an area which incorporates various technologies such as web, databases, or imaging. One characteristic of GIS is their organization as *layers*. This is inherited from the plastic sheets that where used until recently for drawing maps. A layer represents the road system, another the fluvial system, another the relief, etc. This is another instance of the tyranny of the dominant decomposition, and is not satisfactory: to which layer belong bridges, into which layer can we represent a multimodal network? Moreover, mining GIS is known to be difficult for the same reason; the layer structure makes inter layer relationships difficult to discover.

The first advantage of applying LIS to GIS is to allow cross-layer navigation. Another advantage is to permit a logical handling of scales. In current GIS systems, scales are treated as different layers, and it is difficult to keep the consistency between all layers that describe the same object. Another advantage that we have observed in a preliminary work is that LIS helps cleaning a data-base. This was not expected, and opens an interesting research area. Another characteristic of GIS is an intensive usage of topological relations (touches, overlaps, etc) and geographical relations (North, upstream, etc). Logic offers a rich language for expressing these relations and combining them.

4.2 Group Decision and Negotiation

Participants: Peggy Cellier, Mireille Ducassé, Sébastien Ferré.

Group decision and negotiation focuses on complex and self-organizing processes that constitute multiparticipant, multicriteria, ill-structured, dynamic, and often evolutionary problems. Group decision and negotiation refers to the whole process or flow of activities relevant to reaching a group decision, and not merely to the final choice - aspects of the process in group decision and negotiation include scanning, communication and information sharing, problem definition (representation) and evolution, alternative generation, and social-emotional interaction. Group decision support systems (GDSS) and negotiation support systems (GDNSS) are amongst the major approaches to address the problems.

In the current thread of research, we are showing that Logical Information Systems provide an innovative technological support for most of the above mentioned aspects of GDSS. In particular, the navigation and filtering capabilities of LIS help detect inconsistencies and

[LT92] R. Laurini, D. Thompson, *Fundamentals of Spatial Information Systems*, Elsevier, Academic Press Limited, 1992.

missing knowledge during meetings. The updating capabilities of LIS enable participants to add objects, features and links between them on the fly. As a result the group has a more complete and relevant set of information. Furthermore, the compact views provided by LIS and the OLAP features help participants embrace the whole required knowledge. The group can therefore build a shared understanding of the relevant information previously distributed amongst the participants. Lastly, the navigation and filtering capabilities of LIS are relevant to quickly converge on a reduced number of targets. A future trend of research will be to investigate how LIS can also support negotiation.

5 Software

5.1 Camelis and Sewelis

Participants: Sébastien Ferré.

Camelis is a stand-alone application that allows to store, retrieve and update objects through a graphical interface. Its main purpose is to experiment with the LIS paradigm. In particular, it has been very useful for refining the query-answer principle in special circumstances (e.g. when there are many answers, or when there are few answers). It is currently used as a personal storage device for handling photos, music, bibliographical references, etc, up to tens of thousands of objects. It implements as closely as possible the LIS paradigm. It is generic w.r.t. logics, and is compatible with our library of logic functors, LogFun (see Section 5.2). It is available on Linux and Windows, and comes with a user manual.

An important extension, Sewelis, has been developped to browse RDF(S) graphs, a Semantic Web standard. It uses a query language whose expressivity is similar to SPARQL, the reference query language of the Semantic Web. The LIS navigation has been proved safe (i.e., does not lead to dead-ends), and complete (i.e., can reach all conjunctive queries), so that users can perform complex searches easily and safely [6]. Sewelis also supports the guided creation and update of objects, according to the UTILIS approach (see Section 6.1).

5.2 LogFun

Participants: Sébastien Ferré.

The formal definition of a LIS is generic with respect to the logic used for object descriptions and for queries. The counterpart is that it is up to the user to design and implement a logic solver to plug in a LIS. This is too demanding on the average user, and we have developed a framework of *logic functors* that permits to build *certified* logic solvers (see Section 3.5).

LogFun is a library of *logic functors* and a *logic composer*. A user defines a logic using the logic functors, and produces a certified software implementation of the logic (i.e., parser, printer, prover) by applying the logic composer to the definition. For instance, using a functor *Interval* for reasoning on intervals (e.g. $x \in [2,5] \implies x \in [0,10]$), and a functor *Prop* for propositional reasoning (e.g. $a \wedge b \implies a$), a user can define logic *Prop(Interval)*. In this logic, a theorem like $x \in [2,5] \vee x \in [7,9] \implies x \in [0,10]$ can be proven. Note that $[2,5] \cup [7,9]$ *is not* an interval, so that *Prop(Interval)* is an actual extension over *Interval*.

What the logic composer does when building logic *Prop(Interval)* is to compose the solver of *Interval* and the generic solver of *Prop*, and build a solver for *Prop(Interval)*. It also type-checks *Prop(Interval)* to produce its certificate using the certificates of *Interval* and *Prop*. In this example, the certificate says that *Prop(Interval)* is complete: everything that could be deduced from the meaning of *Prop(Interval)* can be proved by its solver. In other circumstances, the certificate indicates that the logic defined by the user is incomplete, w.r.t. the semantics and solvers that come with the functors. In this case, the certificate also indicates what hypotheses are missing for completeness; this may help users to define a more complete variant of their logic.

Logic functors offer basic bricks and a building rule to safely design new logics. For instance, in a recent application of LIS to geographical information system, a basic reasoning capability on locations was needed. The designer of the application, not a LIS or LogFun author, could build a relevant ad hoc logic safely and rapidly.

5.3 Abilis

Participants: Benjamin Sigonneau, Pierre Allard, Mireille Ducassé.

Abilis provides the LIS functionalities as a Web application. The advantage of a Web application is that users do not have to bother about set up, and we hope that this will foster the diffusion of logical information systems. We plan to publish a number of datasets, for example the collection of LIS publications, and to allow people to create their own datasets. Each dataset is defined by a logical context, and a set of users. Abilis is developed in the Ocsigen framework, based on a thin client – thick server architecture. Abilis is based on Camelis, which provides the LIS API, and where the graphical user interface is replaced by a XHTML Web interface.

Abilis improves Camelis on two aspects: multi-user access and visualization. Multi-user access is crucial in a Web application, and requires the management of users and access rights. Anonymous users can browse a context, while advanced users can also update, create and delete contexts. The development of multi-user access is led by Benjamin Sigonneau.

The work on visualization is part of the PhD of Pierre Allard. The main idea is to allow users to create complex views of the extension (the query answers), reusing ideas and concepts from OLAP. A key characteristic of LIS, the consistency between the query, the navigation tree, and extension views, is retained. Users can partition the extension by selecting dimensions, project it by selecting a measure, and aggregate the results (e.g., count, sum). The resulting OLAP cubes can be displayed as tables, charts, or on a map. Thanks to the map representation, most functionalities of GEOLIS are now available in Abilis.

The work on Abilis this year can be broadly summarized by the following points:

- adding features that were missing from Camelis;
- adding new unique features not found in Camelis, in particular features linked to note-taking;
- redesigning part of the interface internals to match some of web applications current best practices;

- improving the visualization part;
- improving geographic operations;
- improving performances.

5.4 Portalis

Participants: Yves Bekkers, Benjamin Sigonneau.

The aim of Portalis is to produce *on the shelve software bricks* to construct web services built on top of our logical information systems. The purpose of this sub-project is to facilitate scientific popularization and industrial transfer of tools produced by the LIS team. The logical information systems are built in OCaml, a very powerful functional language but which is not commonly well accepted in the industrial community.

A first software brick is terminated. It implements an HTTP server which consists in a wrapping of Camelis via a dedicated XML HTTP protocol built on top of Ocaml. Each Camelis function is implemented as an XML HTTP request. XML is used to serialize parameter values and function results. The Camelis server can be set to listen to client's XML requests. For each request, the server marshals the entry parameter values into Ocaml values. Then, it calls the corresponding Camelis function (in Ocaml). When it receives the result, the Camelis server serializes this result into an XML representation which is sent back to the client as an HTTP answer. This brick is presently under test.

A second brick implements a Java layer which is the counter part of the first brick. It is meant to easily build HTTP clients. It provides a Java function for each Camelis XML HTTP request. The function serializes in XML the client's parameter values, sends then through the WEB as an HTTP Request. Wait for the the response to come. When the response is received, it is marshaled into a java value, given back to then client. This brick is also under test.

Two other bricks are under development, they are meant to easily build the administrative part of Portalis portal.

5.5 Typed grammars

Participants: Denis Béchet [LINA-Nantes], Annie Foret [contact point], Sébastien Ferré.

A Pregroup ToolBox is under development on the gforge Inria as a collaborative work with LINA. It includes a generic pregroup parser (LINA) and grammar lexicon definitions and manipulation tools based on XML. An interface with Camelis has been developped (from Camelis to the Pregroup XML format, and the other way round). It has been used to define and experiment grammar prototypes for different natural languages.

Connexions are made possible between the Pregroup parser and Camelis, as shown as part of a demonstration session at LACL2012 [21] (available at `http://lacl.gforge.inria.fr/lacl-2012/demo-proceedings.pdf`).

5.6 SQUALL: a Semantic Query and Update High-Level Language

Participants: Sébastien Ferré.

SQUALL (Semantic Query and Update High-Level Language) is a controlled natural language (CNL) for querying and updating RDF graphs [14]. The main advantage of CNLs is to reconcile the high-level and natural syntax of natural languages, and the precision and lack of ambiguity of formal languages. SQUALL has a strong adequacy with RDF, and covers all constructs of SPARQL, and many of SPARQL 1.1. Its syntax completely abstracts from low-level notions such as bindings and relational algebra. It features disjunction, negation, quantifiers, built-in predicates, aggregations with grouping, and n-ary relations through reification.

SQUALL is available as a Web application at `http://lisfs2008.irisa.fr/ocsigen/squall/` under two forms: one that translates SQUALL sentences to SPARQL, and another one that directly return query answers given a SPARQL endpoint.

5.7 PEW: Possible World Explorer

Participants: Sébastien Ferré, Sebastian Rudolph.

The Possible World Explorer (PEW) reuses the query-based faceted search principles of Sewelis for exploring the possible worlds of an OWL ontology. Users are guided in the incremental construction of class expressions, such that only satisfiable classes are reachable. All classes made of qualified existential restrictions, nominals, intersections, unions, and atomic negations are reachable.

PEW not only supports the exploration of an ontology's possible worlds, but also supports its completion by the addition of axioms [12]. When a class is found satisfiable, and this contradicts domain knowledge (e.g., a man that is not a person), the undesirable possible worlds can be excluded ("pew pew!") by asserting an axiom saying that this class is unsatisfiable (e.g., every man is a person). This could be made a game, where the player would strive to exclude as many undesirable worlds as possible. The benefits are to complete the ontology with more knowledge, and therefore to improve its deduction power.

In addition to asserting negative axioms (about things that should not exist), PEW also allows for the definition of named classes (OWL equivalent class axioms), and for the creation of named individuals as instances of class expressions (OWL class assertion axioms).

6 New Results

6.1 Guided creation and update of objects in RDF(S) bases

Participants: Alice Hermann, Mireille Ducassé, Sébastien Ferré.

With existing tools, when creating a new object in the Semantic Web, users benefit neither from existing objects and their properties, nor from the already known properties of the new object. We have proposed UTILIS [17, 16, 15, 2], an interactive process to help users add new objects. While creating a new object, relaxation rules are applied to its current description to find similar objects, whose properties serve as suggestions to expand the description. A user

study conducted on a group of master students shows that students, even the ones disconcerted by the unconventional interface, used UTILIS suggestions. In most cases, they could find the searched element in the first three sets of properties of similar objects. Moreover, with UTILIS users did not create any duplicate whereas with the other tool used in the study more than half of them did. Another user study was conducted with a visiting student from India, Preeti Dahiya. She intensely used UTILIS in SEWELIS for the annotation of the 573 episodes the Hindi serial "Sasural Genda Phool". After a first phase for learning the tool and defining a schema for describing episodes and related people, Preeti produced around 18,000 triples and 8,000 objects in about 2-3 weeks. The time for describing an episode went decreasing from 3 hours for the 1st episode to less than 10 minutes after 200 episodes and a bug fix.

6.2 Reconciling faceted search and query languages for the Semantic Web

Participants: Sébastien Ferré, Alice Hermann.

Faceted search and querying are two well-known paradigms to search the Semantic Web. Querying languages, such as SPARQL, offer expressive means for searching RDF datasets, but they are difficult to use. Query assistants help users to write well-formed queries, but they do not prevent empty results. Faceted search supports exploratory search, i.e., guided navigation that returns rich feedbacks to users, and prevents them to fall in dead-ends (empty results). However, faceted search systems do not offer the same expressiveness as query languages. We introduce *Query-based Faceted Search* (QFS), the combination of an expressive query language and faceted search, to reconcile the two paradigms. We formalize the navigation of faceted search as a navigation graph, where navigation places are queries, and navigation links are query transformations. We prove that this navigation graph is *safe* (no dead-end), and *complete* (every query that is not a dead-end can be reached by navigation). The LISQL query language generalizes existing semantic faceted search systems, and covers most features of SPARQL. A prototype, SEWELIS, has been implemented, and a usability evaluation demonstrated that QFS retains the ease-of-use of faceted search, and enables users to build complex queries with little training [6] .

6.3 Logical Navigation for Fair and Fast Convergence in Multicriteria Group Decision Making

Participants: Peggy Cellier, Mireille Ducassé, Sébastien Ferré.

Group work represents a large amount of time in professional life while many people feel that much of that time is wasted. This amount of time is even going to increase because problems are becoming more complex and are meant to be solved in a distributed way. Each involved person has a local and partial view of the problem, no one embraces the whole required knowledge. Building up shared knowledge in order to gather relevant distributed knowledge of a problem is therefore a crucial issue.

This thread of research consists in applying Logical Information Systems to Support Group Decision. Information overload is a key issue in group decision. In order to minimize the effects

of information overload, people tend to employ conscious or even unconscious heuristics [VC10]. The "gaze heuristics", reported in [MGG10], is exemplary. In order to catch a ball high up in the air, a player fixates it, starts running and keeps the angle of gaze constant. He does not beforehand calculate a complex differential equation but he will be at the proper place to catch the ball. Another of the heuristics described in [MGG10], called "take-the-best", has been shown useful to face multicriteria decisions while reducing information overload: when making decisions people often take criteria in a predefined order, the first criterion which discriminates the alternatives at stake is used to make the decision. In order to rationalize group work, Briggs and de Vreede have proposed collaboration design patterns, called thinkLets [BdV09]. We have designed the LogicalMulticriteriaSort thinkLet that can be seen as a generalization of the "take-the-best" heuristics. It also proposes to consider criteria one at the time but once a criterion has been found discriminating it is kept in a record, and the process is iterated. The thinkLet is supported by a group decision support system, based on Logical Information Systems. It firstly enables participants to easily navigate in the data. It secondly gives an instantaneous feedback of each micro decision and it thirdly builds a shared knowledge by keeping tracks of all of the decisions taken so far. It thus provides support for the gaze heuristic with no cognitive overload. The LogicalMulticriteriaSort thinkLet has been tested on the debriefing of an academic year validation jury whose results had been controversial. The test case participants were positive about the process and the main result was that they all agreed to use the LogicalMulticriteriaSort thinkLet and the supporting tool for the forthcoming jury at the same level [10].

6.4 Exploratory Enrichment of Ontologies with Negative Constraints

Participants: Sébastien Ferré, Sebastian Rudolph.

With the persistent deployment of ontological specifications in practice and the increasing size of the deployed ontologies, methodologies for ontology engineering are becoming more and more important. However, the specification of negative constraints is often neglected by the human expert, whereas they are crucial for increasing an ontology's deductive potential.

We have proposed a novel, arguably cognitively advantageous methodology for identifying and adding missing negative constraints to an existing ontology. To this end, a domain expert navigates through the space of satisfiable class expressions with the aim of finding absurd ones, which then can be forbidden by adding a respective constraint to the ontology.

We have given the formal foundations of our approach, provided an implementation, called Possible World Explorer (PEW) and illustrated its usability by describing prototypical navigation paths using the example of the well-known pizza ontology [12].

[VC10] D. Vogel, J. Coombes, "The Effect Of Structure On Convergence Activities Using Group Support Systems", *in: Handbook of Group Decision and Negotiation*, D. M. Kilgour and C. Eden (editors), *Advances in Group Decision and Negotiation*, *4*, Springer Netherlands, 2010, ch. 17, p. 301–311.

[MGG10] J. N. Marewski, W. Gaissmaier, G. Gigerenzer, "Good judgments do not require complex cognition", *Cognitive Processing 11*, 2, 2010, p. 103–121.

[BdV09] R. Briggs, G.-J. de Vreede, *ThinkLets: Building Blocks for Concerted Collaboration*, Center for Collaboration Science, University of Nebraska at Omaha, USA, 2009.

6.5 Extension of the LISQL query language for the representation and exploration of mathematical expressions in RDF

Participants: Sébastien Ferré.

Mathematical expressions account for a large part of human knowledge. We have proposed an RDF representation of them in order to integrate them to other kinds ofr knowledge in the Semantic Web. We have also extended the LISQL description and query language so as to reconcile non-ambiguous representations, expressive queries, and natural and concise notations. For example, the query `int(...?X ^ 2...,?X)` retrieves all integrals in x whose body contains the sub-expression x^2. The above features allows for the use of SEWELIS for the guided representation and exploration of mathematical expressions. The guiding frees users from mastering the syntax of LISQL, and the dataset vocabulary, while guaranteeing well-formed expressions and non-empty query results [13].

6.6 SQUALL: a Controlled Natural Language for Querying and Updating RDF Graphs

Participants: Sébastien Ferré.

Formal languages play a central role in the Semantic Web. An important aspect regarding their design is syntax as it plays a crucial role in the wide acceptance of the Semantic Web approach. The main advantage of controlled natural languages (CNL) is to reconcile the high-level and natural syntax of natural languages, and the precision and lack of ambiguity of formal languages. In the context of the Semantic Web and Linked Open Data, CNL could not only allow more people to contribute by abstracting from the low-level details, but also make experienced people more productive, and make the produced documents easier to share and maintain. SQUALL is a controlled natural language for querying and updating RDF graphs. It has a strong adequacy with RDF, an expressiveness close to SPARQL 1.1, and a CNL syntax that completely abstracts from low-level notions such as bindings and relational algebra. We have formally defined the syntax and semantics of SQUALL as a Montague grammar, and its translation to SPARQL. It features disjunction, negation, quantifiers, built-in predicates, aggregations with grouping, and n-ary relations through reification [14].

6.7 Type-logical Grammar formalisms

Participants: Annie Foret.

HDR . We have made a synthesis of our work "on some classes of type-logical grammars that model syntax", in a habilitation document, defended in 2012 (5 july) [1].

These frameworks are used in computational linguistics to model syntax using a strongly lexicalized style, properties (logical types) beeing attached directly to words. Type-logical grammars are also connected with logic, especially linear logic. For instance, parsing a sentence is similar to searching a proof, and semantics can be transparently mapped onto structure in the style of the Curry-Howard isomorphism.

A significant part of this work is connected to automatic acquisition (learning, grammatical inference) issues, and possibilities to help in the design of valuable grammars.

In more details, among the formal and computational issues underlying these systems and some specific subclasses, we have been involved in these questions: (i) relations between classes (or subclasses) of grammars, language hierarchies, language decidability questions; (ii) relations among types assigned to words, given a grammar system based on a type calculus; (iii) learnability from strings of subclasses of grammars (given that the whole class is usually not learnable from "raw texts" as positive examples); (iv) structures and learnability for subclasses of grammars (subclasses unlearnable from strings, may become learnable from adequate structures -or properly annotated texts-); (v) other type systems (adding type constructors and their rules); (vi) exploiting logical views (composing calculus and browsing, maintaining a categorial grammar)

A lot of this work has been performed in collaboration, in particular with the Gracq group (ARC Inria "acquisition de grammaires catégorielles"), then with the TALN team at LINA and with the LIS team at Irisa; on the computational side, we have also taken part in the design of prototypes, in collaboration with these two teams; small demos are possible.

Categorial grammar hierarchies. The notion of k-valued categorial grammars in which every word is associated to at most k types is often used in the field of lexicalized grammars as a fruitful constraint for obtaining interesting properties like the existence of learning algorithms. This constraint is reasonable only when the classes of k-valued grammars correspond to a real hierarchy of generated languages. Such a hierarchy has been established earlier for the classical categorial grammars.

In a recent paper [5] with collegues in LINA (Denis Béchet, Alexandre Dikovsky), the hierarchy by the k-valued constraint is established in the class of categorial grammars extended with iterated types adapted to express the so called projective dependency structures.

6.8 Categorial Grammars as LIS

Participants: Annie Foret, Sébastien Ferré.

Categorial grammars already have close connections with logic. We consider more generally different ways of connecting computational linguistic data and LIS.

In this perspective, a theoretical study has been proposed for a version of a library of logic functors (LogFun) dedicated to the logic of pregroup: this is detailed in [7].

The connexions between Categorial Grammars and LIS have been further developped, on the pratical side, including a prototype demonstration at LACL 2012 [21]. We had explored before different perspectives on how categorial grammars can be considered as Logical Information Systems (LIS) where objects are organized and queried by logical properties both theoretically, and practically. We had also considered LIS for the development of pregroup grammars. The demonstration proposes to illustrate these points with the CAMELIS tool that is an implementation of Logical Information Systems (LIS) and that has been developped at Irisa-Rennes. CAMELIS may give another view on linguistic data, and provide an easy help to browse, to update, to create and to maintain or test such data.

6.9 GEOLIS: a Logical Information System to Organize and Search Geo-Located Data

Participants: Olivier Bedel, Sébastien Ferré, Olivier Ridoux.

GEOLIS is an exploration tool for geolocalized data. It applies to GML files, and is used through a Web interface that contains a query box, a dynamic map displaying the query results, and a dynamic index reflecting the distribution of those results. The main specificity of GEOLIS is to guide the user step by step in the construction of complex queries, while ensuring that the result set is never empty. This navigation enables the user to perform both retrieval search and exploratory search. GEOLIS is mainly targeted at end-users, but it offers several extension points that allows its specialization to different applications. *This work has been published as a chapter of a book on "Innovative Software Development in GIS" [4, 3].*

6.10 Cubes of Concepts: Multi-dimensional Exploration of Multi-valued Contexts

Participants: Pierre Allard, Sébastien Ferré, Olivier Ridoux.

A number of information systems offer a limited exploration in that users can only navigate from one object to another object, e.g. navigating from folder to folder in file systems, or from page to page on the Web. An advantage of conceptual information systems is to provide navigation from concept to concept, and therefore from set of objects to set of objects. The main contribution of this paper is to push the exploration capability one step further, by providing navigation from set of concepts to set of concepts. Those sets of concepts are structured along a number of dimensions, thus forming a cube of concepts. We have described a number of representations of concepts, such as sets of objects, multisets of values, and aggregated values. We have applied our approach to multi-valued contexts, which stand at an intermediate position between many-valued contexts and logical contexts. Users can navigate from one cube of concepts to another. This navigation includes and extends both conceptual navigation and OLAP operations on cubes [11].

6.11 Sequential Pattern Mining to Discover Relations between Genes and Rare Diseases

Participants: Peggy Cellier, Nicolas Béchet [University of Caen], Thierry Charnois [University of Caen], Bruno Crémilleux [University of Caen].

Orphanet provides an international web-based knowledge portal for rare diseases including a collection of review articles. However, reviews and literature monitoring are manual. Thus, new documentation about a rare disease is a time-consuming process and automatically discovering knowledge from a large collection of texts is a crucial issue. This context represents a strong motivation to address the problem of extracting gene rare diseases relationships from texts. We tackle this issue with a cross-fertilization of information extraction and data mining techniques (sequential pattern mining under constraints) [8, 9]. Experiments show the interest of the method for the documentation of rare diseases.

6.12 Sequential Data Mining Techniques to Identify Linguistic Patterns

Participants: Peggy Cellier, Nicolas Béchet [University of Caen], Thierry Charnois [University of Caen], Bruno Crémilleux [University of Caen], Solen Quiniou [University of Nantes].

We have presented two methods based on data mining techniques to automatically discover linguistic patterns.

The first one approach allows to automatically discover linguistic patterns matching appositive qualifying phrases [7]. We have developped an algorithm mining sequential patterns made of itemsets with gap and linguistic constraints. The itemsets allow several kinds of information to be associated with one term. The advantage is the extraction of linguistic patterns with more expressiveness than the usual sequential patterns. In addition, the constraints enable to automatically prune irrelevant patterns. In order to manage the set of generated patterns, we propose a solution based on a partial ordering. A human user can thus easily validate them as relevant linguistic patterns. We illustrate the efficiency of our approach over two corpora coming from a newspaper.

We have also studied the use of data mining techniques for stylistic analysis, from a linguistic point of view, by considering emerging sequential patterns [20, 18]. First, we show that mining sequential patterns of words with gap constraints gives new relevant linguistic patterns with respect to patterns built on n-grams. Then, we investigate how sequential patterns of itemsets can provide more generic linguistic patterns. We validate our approach from a linguistic point of view by conducting experiments on three corpora of various types of French texts (Poetry, Letters, and Fictions). By considering more particularly poetic texts, we show that characteristic linguistic patterns can be identified using data mining techniques. We also discuss how to improve our proposed approach so that it can be used more efficiently for linguistic analyses.

6.13 Graph Mining Under Linguistic Constraints to Explore Large Texts

Participants: Peggy Cellier, Thierry Charnois [University of Caen], Dominique Legallois [University of Caen], Solen Quiniou [University of Nantes].

We have proposed an approach to explore large texts by highlighting coherent sub-parts [19]. The exploration method relies on a graph representation of the text according to the Hoey linguistic model which allows the selection and the binding of sentences in the graph. Our contribution relates to using graph mining techniques under constraints to extract relevant subparts of the text (i.e., collections of homogeneous sentence sub-networks). We have conducted some experiments on two large English texts to show the interest of the proposed approach.

7 Contracts and Grants with Industry

7.1 Portalis: funding for maturation (FEDER Région Bretagne)

Participants: Yves Bekkers, Benjamin Sigonneau, Sébastien Ferré.

As part of the implementation of the pole "Regional Competitiveness and Employment (2007-2013)" in Brittany, we obtained a funding from FEDER and Région Bretagne for an engineer for a year to participate in CAMELIS transfer to industry. Benjamin Sigonneau has been appointed on this founding. It started in October 2012.

Presently, we have discussions with Mediadone, a company specializing in processing, indexing and image enhancement. Mediadone provides tools for interactive and enriched WebTV. The company is interested in using Portalis bricks to build an intelligent navigation tool based on the use of Camelis. Mediadone has already acquired a license for the commercial exploitation of CAMELIS.

8 Other Grants and Activities

8.1 International Collaborations

- The LIS team received the visit of Sebastian Rudolph during March 2012, as a Rennes 1 invited professor. Sebastian is an associate professor from the Karlsruhe Institute of Technology (KIT), working on ontological-based reasoning and formal concept analysis. His visit led to the publication of a paper at EKAW'12 [12] on the combination of query-based faceted search and ontological reasoning for the "exploratory enrichment of ontologies with negative constraints". Sebastian also gave a conference on the Semantic Web in front of students and academic staff.

- Sébastien Ferré is a member of the management committee of the COST action MUMIA (IC1002 - Multilingual and multifaceted interactice information access). MUMIA aims to coordinate collaboration between the following disciplines: machine translation, information retrieval, and faceted search. The objectives of the action is to foster research and development for next generation search technologies. The domain of patent search has been selected as a common use case, as it provides highly sophisticated and information intensive search tasks that have significant economic ramifications.

8.2 National Collaborations

- Annie Foret is an external collaborator of LINA (research lab. Nantes), in TALN team (Natural Language Processing), and member of "Agence Universitaire de la Francophonie" (AUF) , LTT network on "Lexicologie, terminologie et traduction". Annie Foret is member of ATALA (Association pour le Traitement automatique des Langues), and of SIF (Société Informatique de France).

- Sébastien Ferré has been invited to talk about Query-based Faceted Search and SEWELIS at the thematic working days (October 15-17) of CrEDIBLE (Data Federation and Distributed Knowledge in Biomedial Imagery), a CNRS MASTODONS project.

9 Dissemination

9.1 Scientific Responsabilities

- Mireille Ducassé has served in the program committee of CLA 2012 (Concept Lattices and their Applications).

- Annie Foret has been a program committee member of the *Formal Grammar* 2012 International Conference. She has been a program committee of the *Logical Aspects of Computational Linguistics* (LACL) 2012 International Conference. She belongs to the program committee of next *Mathematics of Linguistics* (MOL) International Conference, and next *Formal Grammar* international conference. Annie Foret has been thesis reviewer and jury member for a PHD in Nantes in 2012, by Ramadan Alfared on the "acquisition de grammaire catégorielle de dépendances de grande envergure".

- Sébastien Ferré and Peggy Cellier are members of the organization committee of EGC 2014 to be held in Rennes in January 2014. The presidents of the committee are Arnaud Martin and René Quiniou.

- Sébastien Ferré is a member of the Editorial Board of the International Conference on Formal Concept Analysis (ICFCA), and a member of the program committee of the workshop FCA4AI, co-located with ECAI. He also served as an external reviewer for the journals *Journal of Data Semantics* (JoDS), *Journal of Knowledge-based Systems* (KNOSYS), *Technique et Science Informatique* (TSI), and the conference *Information Retrieval Facility Conference* (IRFC). Sébastien Ferré served as an examiner in the PhD defense committee (Rennes 1 Lannion) of Nouredine Tamani on "Personalized querying of information systems dedicated to transport: a fuzzy bipolar approach". Sébastien is also a supervisor of the PhDs of Alice Hermann and Mouhamadou Ba. He is a member of the committee of the DKM scientific department (Data and Knowledge Management) at IRISA.

- Peggy Cellier has served in the program committee of SEKE 2012 (Software Engineering ans Knowledge Engineering) and ICCS 2013 (Int. Conf. on Conceptual Structures). She has also served as an external reviewer for ICDM 2012 (Int. Conf. on Data Mining)

- Olivier Ridoux served in Mehdi Khiari PhD defense committee at University of Caen, "Découverte de motifs n-aires utilisant la programmation par contraintes".

9.2 Involvement in the Scientific Community

- Peggy Cellier has taken part in the GDR I3 seminar ("Journée sur la fouille de données"), which groups together the French data mining community.

- Sébastien Ferré has been invited to give a conference on Semantic Web at the general assembly of GRANIT, on November 29. GRANIT is a Breton regional network in the IT sector. On this occasion, contacts were made with two SME in Rennes using Semantic Web technologies: Semsoft (François Paulus) and OGBD (Romain Bovyn).

- Annie Foret has been elected as a member of the scientific committee of ISTIC-Rennes1. She is a member of the committee "Développement Durable (Sustainable development)"
- Olivier Ridoux organized with Marie-Odile Cordier and David Gross-Amblard (Data and Knowledge Management department at IRISA) a Turing 100th Birthday. The event was targeted at non-specialists and designed as an initiation to the problematics in which Alan Turing was so successful. The event was organized with the help of IRISA, Service Commun de Documentation (University Library, http://www-scd.univ-rennes1.fr/), and Espace Ferrié (Telecommunication Museum, http://www.espaceferrie.fr/). It was organized as a parcours of several workshops, each dedicated to a facet of Turing's heritage: Turing's Test using chatbots, cryptanalysis, a demo of Enigma (a simulator as well as a real one), Turing's machine (a real unplugged wooden device), a bibliographical exposition, and movie projections (including Spielberg's A.I. Artificial Intelligence).
- Alice Hermann presented her PhD results at the "Atelier RàPC", an annual meeting for the community of case-based reasoning.

9.3 Teaching

- Olivier Ridoux teaches data-bases, algorithmics, the theory of formal languages and compilation in the engineering school. He is also in charge of the innovation training in the school. He also teaches logic and constraint programming at the Master level, and an introduction to the principles of IT systems at the Bachelor level.

  He has been the chair of two recruitment committees ("comité de sélection") at University of Rennes 1.

- Mireille Ducassé is the director of international relations of the INSA of Rennes since december 2010. As such, she is a member of the direction of the Insa of Rennes.

  She has been a member of a recruitment committee ("comité de sélection") in computer science at the university of Rennes 1.

  At Insa, she is responsible of three courses: *Formal Methods for Software Engineering* (with the "B formal method") and *Constraint Programming* at Master 1 level, as well as *Participatory Design* based on the work of Wendy Mackay from the "In Situ" project of Inria Futurs, at Master 2 level. She is also involved in the lab work of the Logic Programming course.

- Sébastien Ferré teaches symbolic data mining and compilation at the master level. He also teaches formal methods for programming and software engineering at the license level. He is vice-director of the MIAGE at ISTIC.
- Annie Foret teaches university courses including formal logic, functional programming, and databases. She is in charge of Master1 Internship (for a section in computer science at ISTIC).
- Peggy Cellier is a member of the "Conseil de composante IRISA/INSA". She has been a member of two recruitment committees ("comités de sélection") in computer science at

the University of Rennes 1 and INSA of Rennes. At Insa, she teached database at licence level; symbolic data mining and formal methods for software engineering at Master level. With a colleague, she organized "Journée portes ouvertes de l'INSA de Rennes", an event to let the public visit INSA of Rennes.

- Yves Bekkers teaches programming languages: functional programming, logic programming (Prolog, lambdaProlog), artificial intelligence, relational databases, XML, new technologies for programming distributed applications on the Web. After being a specialist of logic programming, his current principal interest is teaching distributed application design using tools based on model technologies (Object programming, XML, SGBDR) which allows building applications from one need to the other using numerous bridges allowing passing automatically from one model to another.
- Alice Hermann teaches programming in Java at Licence 1 level of INSA and database at Licence 2 level of INSA.

10 Bibliography

Major publications by the team in recent years

[1] D. Béchet, A. Foret, "A Pregroup Toolbox for Parsing and Building Grammars of Natural Languages", *Linguistic Analysis Journal 36*, 2010.

[2] M. Ducassé, S. Ferré, "Fair(er) and (almost) serene committee meetings with Logical and Formal Concept Analysis", *in: Proceedings of the International Conference on Conceptual Structures*, P. Eklund, O. Haemmerlé (editors), Springer-Verlag, July 2008. Lecture Notes in Artificial Intelligence 5113.

[3] S. Ferré, R. D. King, "A dichotomic search algorithm for mining and learning in domain-specific logics", *Fundamenta Informaticae – Special Issue on Advances in Mining Graphs, Trees and Sequences 66*, 1-2, 2005, p. 1–32.

[4] S. Ferré, O. Ridoux, "A Framework for Developing Embeddable Customized Logics", *in: Int. Work. Logic-based Program Synthesis and Transformation*, A. Pettorossi (editor), *LNCS 2372*, Springer, p. 191–215, 2002.

[5] S. Ferré, O. Ridoux, "An Introduction to Logical Information Systems", *Information Processing & Management 40*, 3, 2004, p. 383–419.

[6] S. Ferré, "Camelis: a logical information system to organize and browse a collection of documents", *Int. J. General Systems 38*, 4, 2009.

[7] A. Foret, "A modular and parameterized presentation of pregroup calculus", *Information and Computation Journal 208*, 5, may 2010, p. 395–604.

[8] Y. Padioleau, O. Ridoux, "A Logic File System", *in: Usenix Annual Technical Conference*, 2003.

[9] B. Sigonneau, O. Ridoux, "Indexation multiple et automatisée de composants logiciels", *Technique et Science Informatiques 25*, 1, 2006.

Doctoral dissertations and "Habilitation" theses

[1] A. Foret, *On some classes of type-logical grammars that model syntax*, PhD Thesis, Matisse, Univ. Rennes 1, 2012, Habilitation à Diriger des Recherches (HDR), defended on July 5th.

[2] A. Hermann, *Création et mise à jour d'objets dans une base de connaissances*, PhD Thesis, Thèse de l'INSA Rennes - École doctorale MATISSE, 17 décembre 2012, supervised by M. Ducassé and S. Ferré.

Articles in referred journals and book chapters

[3] O. Bedel, S. Ferré, O. Ridoux, *Développements logiciels en géomatique – innovations et mutualisations (ed., B. Bucher and F. Le Ber)*, *Information géographique et Aménagement du Territoire*, Hermes/Lavoisier, 2012, ch. GEOLIS : un système d'information logique pour l'organisation et la recherche de données géolocalisées, p. 149–180.

[4] O. Bedel, S. Ferré, O. Ridoux, *Innovative Software Development in GIS (ed., B. Bucher and F. Le Ber)*, *Geographical Information Systems Series*, Wiley, 2012, ch. GEOLIS: a Logical Information System to Organize and Search Geo-Located Data, p. 151–188.

[5] D. Béchet, A. Dikovsky, A. Foret, "Categorial grammars with iterated types form a strict hierarchy of k-valued languages", *Theor. Comput. Sci. 450*, 2012, p. 22–30.

[6] S. Ferré, A. Hermann, "Reconciling faceted search and query languages for the Semantic Web", *Int. J. Metadata, Semantics and Ontologies 7*, 1, 2012, p. 37–54.

Publications in Conferences and Workshops

[7] N. Béchet, P. Cellier, T. Charnois, B. Crémilleux, "Discovering Linguistic Patterns Using Sequence Mining", *in : Int. Conf. on Computational Linguistics and Intelligent Text Processing (CICLing)*, A. F. Gelbukh (editor), *LNCS, 7181*, Springer, p. 154–165, 2012.

[8] N. Béchet, P. Cellier, T. Charnois, B. Crémilleux, "Fouille de motifs séquentiels pour la découverte de relations entre gènes et maladies rares", *in : Journées francophones d'ingénierie des connaissances*, S. Szulman, J. Charlet (editors), INSERM UMPC, p. 149–164, 2012, `http://ics.upmc.fr/`.

[9] N. Béchet, P. Cellier, T. Charnois, B. Crémilleux, "Sequential Pattern Mining to Discover Relations between Genes and Rare Diseases", *in : IEEE Int. Symp. on Computer-Based Medical Systems (CBMS)*, p. 1–6, 2012.

[10] M. Ducassé, P. Cellier, "The LogicalMulticriteriaSort ThinkLet: Logical Navigation for Fair and Fast Convergence in Multicriteria Group Decision Making", *in : Proceedings of the Group Decision and Negotiation Conference*, A. T. de Almeida, D. C. Morais, S. de França Dantas Daher (editors), Federal University of Pernambuco, p. 87–96, May 2012. ISBN 978-85-415-0036-4.

[11] S. Ferré, P. Allard, O. Ridoux, "Cubes of Concepts: Multi-dimensional Exploration of Multi-valued Contexts", *in : Int. Conf. Formal Concept Analysis*, F. Domenach, D. I. Ignatov, J. Poelmans (editors), *LNCS 7278*, Springer, p. 112–127, 2012.

[12] S. Ferré, S. Rudolph, "Advocatus Diaboli - Exploratory Enrichment of Ontologies with Negative Constraints", *in : Int. Conf. Knowledge Engineering and Knowledge Management (EKAW)*, A. ten Teije et al. (editor), *LNAI 7603*, Springer, p. 42–56, 2012.

[13] S. Ferré, "Extension du langage de requêtes LISQL pour la représentation et l'exploration d'expressions mathématiques en RDF", *in : Journées francophones d'ingénierie des connaissances*, S. Szulman, J. Charlet (editors), INSERM UMPC, p. 285–300, 2012, `http://ics.upmc.fr/`.

[14] S. Ferré, "SQUALL: a Controlled Natural Language for Querying and Updating RDF Graphs", *in : Controlled Natural Languages*, T. Kuhn, N. Fuchs (editors), *LNCS 7427*, Springer, p. 11–25, 2012.

[15] A. Hermann, S. Ferré, M. Ducassé, "Aide à la création d'objets dans une base RDF(S) avec des règles de relaxation", *in : Journées francophones d'ingénierie des connaissances*, S. Szulman, J. Charlet (editors), INSERM UMPC, p. 301–316, 2012, `http://ics.upmc.fr/`.

[16] A. Hermann, S. Ferré, M. Ducassé, "Guided Semantic Annotation of Comic Panels with Sewelis", *in : Knowledge Engineering and Knowledge Management (EKAW)*, A. ten Teije et al. (editor), *LNCS 7603*, Springer, p. 430–433, 2012.

[17] A. Hermann, S. Ferré, M. Ducassé, "An Interactive Guidance Process Supporting Consistent Updates of RDFS Graphs", *in : Int. Conf. Knowledge Engineering and Knowledge Management (EKAW)*, A. ten Teije et al. (editor), *LNAI 7603*, Springer, p. 185–199, 2012.

[18] S. Quiniou, P. Cellier, T. Charnois, D. Legallois, "Fouille de données pour la stylistique : cas des motifs séquentiels émergents", *in : Actes des Journées Internationales d'Analyse Statistique des Données Textuelles*, p. 821–833, Liège, Belgique, 2012, `http://hal.archives-ouvertes.fr/hal-00675586`.

[19] S. Quiniou, P. Cellier, T. Charnois, D. Legallois, "Fouille de graphes sous contraintes linguistiques pour l'exploration de grands textes", *in : Actes de la Conférence sur le Traitement Automatique des Langues Naturelles*, p. 253–266, Grenoble, France, 2012, `http://hal.archives-ouvertes.fr/hal-00702606`.

[20] S. Quiniou, P. Cellier, T. Charnois, D. Legallois, "What About Sequential Data Mining Techniques to Identify Linguistic Patterns for Stylistics?", *in : Int. Conf. on Computational Linguistics and Intelligent Text Processing (CICLing)*, A. F. Gelbukh (editor), *LNCS, 7181*, Springer, p. 166–177, New Delhi, Inde, March 2012.

Miscellaneous

[21] A. Foret, S. Ferré, "On Categorial Grammars and Logical Information Systems : using CAMELIS with linguistic data", presented at the demo session of LACL'12, 2012.