Project-Team LIS

# *Logical Information Systems*

*Rennes*

*Activity Report*

*2010*

# 1 Team

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

**Administrative assistant**
Marie-Noëlle Georgeault, INRIA

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

**Insa personnel**
Mireille Ducassé, Professor
Peggy Cellier, Assistant Professor, since September 2010
Benjamin Sigonneau, Research Engineer (part time: 50%)
Véronique Abily, Research Engineer (part time: 20%)

**PhD students**
Pierre Allard, Région Bretagne grant, since October 2008
Alice Hermann, *MENRT* grant, since October 2009
Luigi Spagnolo, visiting student from Politecnico di Milano, since November 2010

**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 [6] 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 [6, 4].

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.

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.

[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.

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 [7].

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 *social choice*, e.g., committee decision making [3]. The idea is to register all the pros and cons of a set of candidates as a formal context and to explore their consequences.

## 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.

**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.

[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.

**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 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

[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.

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) [6], 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 [6, 4].

### 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).

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

**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 [6]. 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 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

[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.

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 [6, 4], and applied them to the automatic classification of emails [FR02], and the prediction of the function of proteins from their sequence [4].

### 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.

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

[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.

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* [5], 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: $\Sigma$ is a finite alphabet (the words in the sentences); $I$ is a function that maps a finite set of types to each element of $\Sigma$ (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 $\Sigma$ 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 $\phi$ 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 $\forall n > n_0$ $\phi(\{e_0, \ldots, e_n\}) = G'$.

# 4 Application Domains

## 4.1 Geographical Information Systems

**Participants**: Pierre Allard, 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

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

(*geolocalized*). GIS form a 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 (toochs, overlaps, etc) and geographical relations (North, upstream, etc). Logic offers a rich language for expressing these relations and combining them.

### 4.2 Mining Software Repositories

**Participants**: Peggy Cellier, Mireille Ducassé, Olivier Ridoux.

There exist numerous repositories related to the development of software: for example source versions generated by control systems, archived communications between project personnel, defect tracking systems, component libraries and execution traces. They are used to help manage the progress of software projects. Software practitioners and researchers are beginning to recognize the potential benefit of mining this information to support the maintenance of software systems, improve software design/reuse, and empirically validate novel ideas and techniques.

Logical information systems seem particularly adapted to mine these repositories. Indeed, the repositories contain heterogeneous and incomplete information. Their size is too large to be directly handled by human beings and it is still manageable by the current implementations of LIS. The LIS team currently focuses on component retrieval and execution traces cross-checking.

## 5 Software

### 5.1 LISFS

**Participants**: Yoann Padioleau, Olivier Ridoux [contact point].

The main objective of a LIS is not to go *faster*; it is to go *easier*. This must be evaluated by experimenting the use of LIS in various contexts. However, the price for an easier usage must not be too high compared to more classical storage means such as a file system. At the same time we want to promote a *file system level implementation of LIS* so that every application that uses the file system interface could use a LIS.

LISFS is a file system that implements LIS under the Linux file system interface [8]. It uses the whole usual file system technology (e.g. caching, journaling) to offer reactivity, safety, robustness, etc. At the same time, it is not intimately attached to Linux technology because it is programmed at the user level, and it uses the FuseFS bridge to redirect file system calls to the user level.

LISFS manages a set of objects described with attributes that may be valued. The attributes of an object form a logic conjunction, disjunction and negation can be expressed in queries. More sophisticated logical descriptions can be embedded in the attributed values. For instance, `title:contains "logic"` is an attribute whose value is `contains "logic"` and refers to a logic of string pattern matching. Objects can be created and deleted (e.g. shell commands `touch` $f$ and `rm` $f$); their attributes can be changed anytime (e.g. shell command `mv` $f1$ $f2$; and new attributes can be created anytime (e.g. shell command `mkdir` $a$).

LISFS response time grows with the number of objects, the number of attributes, and the complexity of their values. We have proved, under hypotheses which are met in practice, that LISFS response time to queries is linear with the number of objects. However, this is not enough in practice, and the ideal complexity is amortized constant time. This is what we try to achieve through the use of file system and database technologies like caches, indexes, and journals. In its current state LISFS can manage up to 1 000 000 objects×attributes with affordable response time for queries. However, the response time for updates is not yet as good as we wish, and this is a track for improvement in the future. Similarly, 100 000 objects is good enough for a personal computer, but is not enough for some professional usage; this is also something we wish to improve.

An important service of LISFS is to permit navigation, querying and updates inside files. This is the part-of-file service, PofFS [PR05]. The idea is that a file is considered as a composition of subparts; the subparts are to the file as files are to a mount point. This is a way to overcome artificial constraints that are often imposed by applications. For instance, it is often the case that methods of the same class must be textual neighbours in a source file. Sometimes what is desired is to see together all methods with the same role, say print. PofFS permits that. In fact, PofFS is just the right thing to do in many applications where the goal is not to find one answer but to display together all answers. This is the case of GIS applications, for instance.

In 2008, LISFS has been extended with relations for linking objects together. This has been applied to spatial relations between geographic features (distance and topology).

## 5.2 GEOLIS

**Participants**: Olivier Bedel, Pierre Allard [contact point].

GEOLIS is a prototype combining a Logical Information System (LIS) and webmapping tools for geographical data exploration. GEOLIS takes the form of a web application. Server-side, GEOLIS relies on LISFS to organize the data and on the webmapping engine MapServer to produce a map representation of data selection. Client-side, the GEOLIS user interface

[PR05] Y. Padioleau, O. Ridoux, "A Parts-of-File File System", *in: USENIX Annual Technical Conference, General Track (Short Paper)*, 2005, `http://www.usenix.org/events/usenix05/tech/general/padioleau.html`.

provides three components: 1) a query box similar to web search engines querying interfaces, 2) a map area, and 3) a navigation tree gathering navigation links. Navigation links can be followed to reduce (resp. enlarge) the current selection of data visible on the map by refining (resp. generalizing) the current query written in the query box. GEOLIS is not yet distributed, but online demos are available. A demo is a partial dataset concerning rodents distribution in Soudano-Sahelian Africa. It aims at helping geographs in the research of factors impacting rodents distribution.

### 5.3 Camelis and Camelis 2

**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 5.4). It is available on Linux and Windows, and comes with a user manual.

An important extension, Camelis2, 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 consistent (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 [11].

### 5.4 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. $x \in [2,5] \Longrightarrow x \in [0,10]$), and a functor *Prop* for propositional reasoning (e.g. $a \wedge b \Longrightarrow a$), a user can define logic *Prop(Interval)*. In this logic, a theorem like $x \in [2,5] \vee x \in [7,9] \Longrightarrow x \in [0,10]$ 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 typechecks *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 the user to define a more complete variant of its 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.

### 5.5 Abilis

**Participants**: Benjamin Sigonneau, Pierre Allard, Mireille Ducassé, Véronique Abily.

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 applications, for example the LIS publications, and to allow people to create their own applications Each application 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 and Véronique Abily. Abilis has been used by Mireille Ducassé for experiments in collaborative decision making.

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.

### 5.6 Typed grammars

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

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.

# 6 New Results

## 6.1 Multi-criteria decision support : consistency and fairness thanks to formal concept analysis

**Participants**: Mireille Ducassé, Sébastien Ferré.

In academia, many decisions are taken in committee, for example to hire people or to allocate resources. Genuine people often leave such meetings quite frustrated. Indeed, it is intrinsically hard to make multi-criteria decisions, selection criteria are hard to express and the global picture is too large for participants to embrace it fully. We describe a recruiting process where logical concept analysis and formal concept analysis are used to address the above problems [3]. We do not pretend to totally eliminate the arbitrary side of the decision. We claim, however, that, thanks to concept analysis, genuine people have the possibility to 1) be fair with the candidates, 2) make a decision adapted to the circumstances, 3) smoothly express the rationales of decisions, 4) be consistent in their judgements during the whole meeting, 5) vote (or be arbitrary) only when all possibilities for consensus have been exhausted, and 6) make sure that the result, in general a total order, is consistent with the partial orders resulting from the multiple criteria.

## 6.2 Type-logical Grammar formalisms and toolbox

**Participants**: Annie Foret.

Pregroup grammars are a context-free grammar formalism which may be used to describe the syntax of natural languages. However, this formalism is not able to easily define types corresponding to optional or iterated arguments like an optional complement of a verb or a sequence of its adverbial modifiers. A former paper [1] has introduced two constructions that make up for this deficiency where Gentzen-style rules are introduced to take care of two new operations, and an equivalent rewriting system. The extended pregroup calculus enjoys several properties shared with traditional dependency grammars, yet does not significantly expand the polynomial complexity of the syntactic analysis on the pregroup grammar. The basic formalism and this extension has been further studied, and explored in the pregroup toolbox under development with a team in Nantes [2]. The use of Camelis in this context is also presented in this work.

Our interest is also in learning categorial grammars [2]. We study in [7] the learnability problem in the family of Categorial Dependency Grammars (CDG). A class of categorial grammars defining unlimited dependency structures is shown incrementally learnable in the limit, under a reasonable condition on the iterant dependencies.

## 6.3 Pregroup Grammars as LIS

**Participants**: Annie Foret, Sébastien Ferré.

A theoretical study was recently proposed for a version of a library of logic functors (LogFun) dedicated to the logic of pregroup: this is detailed in [4].

In [12], we explore different perspectives on how categorial grammars can be considered as Logical Information Systems (LIS) both theoretically, and practically. Categorial grammars already have close connections with logic. We discuss the advantages of integrating both approaches. We consider more generally different ways of connecting computational linguistic data and LIS as an application of Formal Concept Analysis.

### 6.4 Conceptual Navigation in RDF Graphs with SPARQL-Like Queries

**Participants**: Sébastien Ferré, Alice Hermann.

Concept lattices have been successfully used for information retrieval and browsing. They offer the advantage of combining querying and navigation in a consistent way. Conceptual navigation is more flexible than hierarchical navigation, and easier to use than plain querying. It has already been applied to formal, logical, and relational contexts, but its application to the semantic web is a challenge because of inference mechanisms and expressive query languages such as SPARQL. We have extended conceptual navigation to the browsing of RDF graphs, where concepts are accessed through SPARQL-like queries. This extended conceptual navigation [11] is proved consistent w.r.t. the context (i.e., never leads to an empty result set), and complete w.r.t. the conjunctive fragment of the query language (i.e., every query can be reached by navigation only). Our query language has an expressivity similar to SPARQL, and has a more natural syntax close to description logics. We have implemented this extended conceptual navigation in Camelis 2, and extension of Camelis. Finally, Alice Hermann has performed a user evaluation that demonstrates the benefits and usability of this approach.

### 6.5 Discovering Functional Dependencies and Association Rules by Navigating in a Lattice of OLAP Views

**Participants**: Pierre Allard, Sébastien Ferré, Olivier Ridoux.

Discovering dependencies in data is a well-know problem in database theory. The most common rules are Functional Dependencies (FDs), Conditional Functional Dependencies (CFDs) and Association Rules (ARs). Many tools can display those rules as lists, but those lists are often too long for inspection by users. We propose a new way to display and navigate through those rules [6, 5]. Display is based on On-Line Analytical Processing (OLAP), presenting a set of rules as a cube, where dimensions correspond to the premises of rules. Cubes reflect the hierarchy that exists between FDs, CFDs and ARs. Navigation is based on a lattice, where nodes are OLAP views, and edges are OLAP navigation links, and guides users from cube to cube.

### 6.6 Sequential Patterns to Discover and Characterise Biological Relations

**Participants**: Peggy Cellier, Thierry Charnois [University of Caen], Marc Plantevit [University of Lyon], Bruno Crémilleux [University of Caen].

We have presented a method to automatically detect and characterise interactions between genes in biomedical literature [8, 9]. The approach is based on a combination of data mining

techniques: frequent sequential patterns filtered by linguistic constraints and recursive mining. Unlike most Natural Language Processing (NLP) approaches, our method does not use syntactic parsing to learn and apply linguistic rules. It does not require any resource except the training corpus to learn patterns.

The process is in two steps. First, frequent sequential patterns are extracted from the training corpus. Second, after validation of those patterns, they are applied on the application corpus to detect and characterise new interactions. An advantage of our method is that the interactions can be enhanced with modalities and biological information.

We use two corpora containing only sentences with gene interactions as training corpus. Another corpus from PubMed abstracts is used as application corpus. We conduct an evaluation that shows that the precision of our approach is good and the recall correct for both targets: interaction detection and interaction characterisation.

### 6.7 Sequential Pattern Mining of Itemsets to Extract Linguistic Patterns

**Participants**: Peggy Cellier, Thierry Charnois [University of Caen].

We have presented a method based on the extraction of itemset sequential patterns in order to automatically generate linguistic patterns. In addition, we have proposed to use the partial ordering between sequential patterns to enumerate and validate them. The method is used to extract linguistic patterns that represent qualifying phrases (e.g., ",connu pour sa cruauté,", "En bon père de famille,") [10].

### 6.8 Assessment of achievements

The results achieved by the LIS team must be compared with the key issues presented in the objective part. Not all key issues have deserved attention yet. However, a few of them have been sufficiently well explored to start and draw conclusions.

We have now gained sufficient experience to claim that even if every application uses a specific logic, it is not necessary to design every specific logic from scratch. Not only do we have proposed a toolbox of logic components for building logic tools (parser, prover, printer, etc), but we also have designed a theory of logic composition that allows to prove meta-properties about the prover thus obtained. This methodology has been successfully applied to build Description Logic style provers, topological property provers, time comparators, text comparators, etc. Moreover, we have design a non-commutative logic comment that allows to build variants of Lambek logics.

We have also progressed on the metaphor issue. We have observed that all applications we have designed require a three-components interface. One component exposes a query, another one exposes a navigation tree, and the last one exposes a set of application oriented entities represented using an application oriented interface. The second component provides at the same time a summary of the answers to the query, and a set of navigation links that lead to related queries. The third component can be a map display for GIS applications, a thumbnail array for a picture application, or an agenda for a personal organizer application. In all applications the three interface components must be tightly connected so that acting on

one component effects the two other components. The three components must always be kept coherent.

Finally, we have demonstrated that LIS principles can be successfully applied to the GIS domain, and to the software fault localization domain. In both cases, LIS principles have permitted new functionalities, and have opened new perspectives on possible developments. The most interesting result is that the same high-level LIS features give rise to different capabilities in different domains. This shows that more than a set of information management principles, LIS is also an application integration principle. This is due to the use of logic for describing data and to the lack of rigid data schematas.

# 7 Contracts and Grants with Industry

## 7.1 ECOMER

**Participants**: Mireille Ducassé, Sébastien Ferré, Benjamin Sigonneau.

The ECOMER project is a national project founded by the French "Ministère de l'Alimentation, de l'Agriculture et de la Pêche" and partially supported by the "Fonds Européen de la Pëche". The objectives are firstly to provide fishers with hardware and software tools that allow them to better control their oil consumption, and secondly to design a module for the training of fishers to the economical steering of ships. The motivation for this project comes from the fact that oil takes a heavier and heavier place in the budget of the fish industry.

The partners are: Chantier Naval Pierre Gléhen et Fils (coordinator), Marinelec Technologies, Peden Ingénierie, iXElek, Avel Vor Technologie, Coopérative Maritime Etaploise, Marelec, From Nord, INSA of Rennes, Comité Régional des Pêches Maritimes et des Elevages Marins de Bretagne, Comité Régional des Pêches Maritimes et des Elevages Marins de Langueddoc-Roussillon, Lycée professionnel maritime du Guilvinec, Lycée professionnel maritime de Boulogne/Le Portel, Lycée de la mer Paul Bousquet - Sète et Centre Européen de Formation Continue. The duration is December 2009 to May 2011. LIS budget share is 53 keuros.

The LIS team provides powerful user interaction mechanisms to browse streams of data about oil consumption. The objective is that fishermen get a detailed feedback about their consumption so that they can learn how to reduce it.

# 8 Other Grants and Activities

## 8.1 International Collaborations

- Sébastien Ferré has been nominated as 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. The kick-off meeting of the action happened on November 30th.

### 8.2 National Collaborations

- The LIS team has a contract with Région Bretagne in collaboration with the laboratory RESO of the University of Rennes 2, for the funding of O. Bedel's PhD (until september 2008), and P. Allard's PhD (since october 2008).
- The GeoTal project has been accepted by MSHB (Maison des sciences de l'homme de Bretagne) for 2009-2010, it regroups local actors from various fields, that are interested in Geographical Data and Natural Language. Meetings have been organized with communications by local actors or by other specialists on this issue. In 2010 we had sessions on "Logic and objects" (UBO and Sorbonne) and on "linguistic collocations" (UBS).
- 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".

## 9 Dissemination

### 9.1 Scientific Responsabilities

- Olivier Ridoux has served in several doctoral and habilitation committees. He also served in several recruitment committees.
- Mireille Ducassé has chaired four PhD committees : Florence Charreteur, University of Rennes 1, March 2010 ; Mathieu Petit, Ecole Navale de Brest, June 2010 ; Damien Cram, University of Lyon 1, September 2010 ; Nizar Kheir, ENSTB, November 2010. She is an elected member of the "Conseil National des Universités (CNU) 27e section", a national assessment committee for teaching and research staff. This amounts for more than 5 weeks of full time work per year. She is a member of SPECIF.
- Annie Foret has been a program committee member of the *Formal Grammar* 2010 International Conference. She has been elected as a member of the scientific committee of ISTIC-Rennes1.
- Sébastien Ferré has been one of the two Program Chairs for the International Conference on Conceptual Structures (ICCS), which took place in July in Kuching, Malaysia [1]. He 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 international conference on Concept Lattices and their Applications (CLA). He has also served as an external reviewer for the International Semantic Web Conference (ISWC).

  Sébastien has presented the scientific focus of the department of Data and Knowledge Management (DKM), on the occasion of the AERES evaluation of the IRISA laboratory. On the same occasion, he as made a demo of GEOLIS to members of the evaluation

committee. He is a supervisor of the PhDs of Pierre Allard and Alice Hermann, and he is also a member of the PhD committee of Nicolas Lebreton.

- Peggy Cellier have served as an external reviewer for a journal: Data & knowledge Engineering ; three international conferences: International Conference on Data Warehousing and Knowledge Discovery (DAWAK'10), International Symposium on Intelligent Data Analysis (IDA'10) and Symposium on Applied Computing (SAC'11) ; two national conferences: Conférence Internationale Francophone sur l'Extraction et la Gestion des Connaissances (EGC'10) and Conférence Francophone d'Apprentissage (CAp'10).

  She was involved in the ANR project BINGO2 about the extraction of gene interactions from biomedical texts.

  She also collaborates with the NOOPSIS company about the extraction of information about companies from press articles.

- Pierre Allard and Alice Hermann served as external reviewers for the French conference on "Langages et Modèles à Objets (LMO)".

## 9.2 Involvement in the Scientific Community

- In september, the LIS team received the visit of James H. Andrews, Associate Professor at the Department of Computer Science of the University of Western Ontario, Canada. The commun topic of interest is the use of data minig for fault localization.

- Peggy Cellier has been invited to the Lorentz Center Workshop about "Mining Patterns and Subgroups".

  She has took part in the GDR/GPL seminar, which groups together the french software engineering community[13].

- Sébastien Ferré has been invited to give a seminar in June in the team DoDoLa, at the University of Caen. He is member of the GDR/I3, and took part in "Les Assises du GDR I3" in July 2010, Strasbourg.

- Annie Foret is member of ATALA (Association pour le Traitement automatique des Langues), and of SPECIF (Société des Personnels Enseignants et Chercheurs en Informatique de France).

- Pierre Allard, Alice Hermann, Sébastien Ferré, Annie Foret and Peggy Cellier have took part in the "treillis camsiens", the annual seminar of the French FCA community.

- Pierre Allard took part in the "Journées Géomatiques de l'Ouest", an annual regional meeting on geographical information systems.

## 9.3 Teaching

- Olivier Ridoux has been the head of ESIR (École supérieure d'ingénieurs de Rennes).

Olivier Ridoux teaches compilation, logic and constraint programming, as well as software engineering at the Master level of IFSIC. He teaches an introduction to computability and complexity at the Licence level. He also teaches an introduction to the principles of IT systems at the Licence level.

- Mireille Ducassé has been the head of the computer science department of the INSA of Rennes until end of February 2010. She has been an elected member of the board of directors ("Conseil d'administration") of the Insa of Rennes until June 2010. She has been a member of three recruitment committees ("comités de sélection") in computer science at the IUT of Vannes, the ENSSAT of Lannion and Insa of Rennes. She is head of the international office of the INSA of Rennes since mid-December 2010.

  At Insa, she teached compilation and formal methods for software engineering (with the "B formal method") at Master 1 level of Insa. She led an exercise of participatory design based on the work of Wendy Mackay from the "In Situ" project of Inria Futurs. She set up a new course on constraint programming.

- 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).

- When Peggy Cellier was a post doctoral researcher at the University of Caen, she teached symbolic data mining at Master 2 level. Since september 2010, she is assistant professor at the INSA of Rennes. She teaches database and risk analysis at licence level ; object-oriented modelling and programming, symbolic data mining and formal methods for software engineering at Master level.

- Pierre Allard teaches initiation to functional programming in Scheme at INSA and ISTIC, at License 1 level.

- Alice Hermann teaches programming in Java at Licence 1 level of INSA and datamining at Master 1 level of INSA.

# 10 Bibliography

## Major publications by the team in recent years

[1] D. Bechet, A. Dikovsky, A. Foret, E. Garel, "Optional and Iterated Types for Pregroup Grammars", *in : Int. Conf. Language and Automata Theory and Applications (LATA)*, C. Martín-Vide, F. Otto, H. Fernau (editors), *LNCS 5196*, Springer, p. 88–100, 2008.

[2] D. Bechet, A. Foret, "k-Valued Non-Associative Lambek Grammars are learnable from Generalized Functor-Argument Structures", *Journal of Theoretical Computer Science 355*, 2, 2006.

[3] 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.

[4] 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.

[5] 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.

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

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

[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.

## Books and Monographs

[1] M. Croitoru, S. Ferré, D. Lukose (editors), *Conceptual Structures: From Information to Intelligence, 18th International Conference on Conceptual Structures, ICCS 2010, Kuching, Sarawak, Malaysia, July 26-30, 2010. Proceedings*, *LNCS 6208*, *6208*, Springer, 2010.

## Articles in referred journals and book chapters

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

[3] M. Ducassé, S. Ferré, "Aide à la décision multicritère : cohérence et équité grâce à l'analyse de concepts", *Revue internationale de systémique complexe et d'études relationnelles, Nouvelles Perspectives en Sciences Sociales 5*, 2, Mai 2010, p. 181–196.

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

## Publications in Conferences and Workshops

[5] P. Allard, S. Ferré, O. Ridoux, "Discovering Functional Dependencies and Association Rules by Navigating in a Lattice of OLAP Views", *in : Concept Lattices and Their Applications*, M. Kryszkiewicz, S. Obiedkov (editors), CEUR-WS, p. 199–210, 2010.

[6] P. Allard, S. Ferré, "Recherche de dépendances fonctionnelles et de règles d'association avec OLAP", *in : Extraction et Gestion des Connaissances*, S. B. Yahia, J.-M. Petit (editors), *Revue des Nouvelles Technologies de l'Information*, *RNTI-E-19*, Cépaduès-Éditions, p. 651–652, 2010.

[7] D. Bechet, A. Dikovsky, A. Foret, "Two models of learning iterated dependencies", *in : Formal Grammar*, *LNCS*, Springer, 2010.

[8] P. Cellier, T. Charnois, M. Plantevit, B. Crémilleux, "Recursive Sequence Mining to Discover Named Entity Relations", *in : Advances in Intelligent Data Analysis IX, 9th International Symposium (IDA)*, P. R. Cohen, N. M. Adams, M. R. Berthold (editors), *LNCS 6065*, Springer, p. 30–41, 2010.

[9] P. Cellier, T. Charnois, M. Plantevit, "Sequential Patterns to Discover and Characterise Biological Relations", *in : Computational Linguistics and Intelligent Text Processing (CICLing)*, A. F. Gelbukh (editor), *LNCS 6008*, Springer, p. 537–548, 2010.

[10] P. Cellier, T. Charnois, "Fouille de données séquentielle d'itemsets pour l'apprentissage de patrons linguistiques", *in : Traitement Automatique des Langues Naturelles (short paper)*, 2010.

[11] S. Ferré, "Conceptual Navigation in RDF Graphs with SPARQL-Like Queries", *in : Int. Conf. Formal Concept Analysis*, L. Kwuida, B. Sertkaya (editors), *LNCS 5986*, Springer, p. 193–208, 2010.

[12] A. Foret, S. Ferré, "On Categorial Grammars as Logical Information Systems", *in : Int. Conf. Formal Concept Analysis*, L. Kwuida, B. Sertkaya (editors), *LNCS 5986*, Springer, p. 225–240, 2010.

## Miscellaneous

[13] P. Cellier, M. Ducassé, S. Ferré, O. Ridoux, "Fouille de données pour la localisation de fautes dans les programmes", *in : Journées nationales du GDR GPL (GDR Génie de la Programmation et du Logiciel)*, 2010.