UMR IRISA

Project-Team LIS

Logical Information Systems

Rennes

Activity Report

2009

1 Team

Head of the team

Olivier Ridoux, Professor, Université Rennes 1

Administrative assistant

Lydie Mabil, INRIA, until June 2009
Marie-Noëlle Georgeault, INRIA, since July 2009

Université Rennes 1 personnel

Annie Foret, Assistant Professor
Sébastien Ferré, Assistant Professor
Peggy Cellier, ATER until August 2009

Insa personnel

Mireille Ducassé, Professor
Véronique Abily, Research Engineer (part time: 20%)

PhD students

Olivier Bedel, Région Bretagne grant, until January 2009
Pierre Allard, Région Bretagne grant, since October 2008
Alice Hermann, *MENRT* grant, since October 2009

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.

4. *Queries should be possible answers.*

[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.

> 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 [7] or at the user level [4].

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: 13th ACM Symposium on 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: 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 $\Sigma_1 = \{John, Mary, likes\}$ $I_1 = \{John \mapsto \{N\},\ Mary \mapsto \{N\},\ likes \mapsto \{N \backslash (S/N)\}\}$ 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 : $X, X \backslash Y \vdash Y$ and $Y, Y/X \vdash Y$. Type constructors / and \ 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 $\forall n > n_0\ \phi(\{e_0, \ldots, e_n\}) = G'$.

4 Application Domains

4.1 Geographical Information Systems

Participants: Olivier Bedel, 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 [7]. 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) [2].

5.2 GEOLIS

Participants: Olivier Bedel.

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

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 [18].

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 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 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 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 GEOLIS: a Logical Information System for Organizing and Searching Geographical Data

Participants: Olivier Bedel.

The following is a summary of the PhD thesis of Olivier Bedel [2], supervised by Olivier Ridoux and Sébastien Ferré.

Today, the thematic layer is still the prevailling structure in geomatics for handling geographical information. However, the layer model is rigid: it implies partitionning geographical data in predefined categories and using the same description schema for all elements of a layer. Futhermore, Geographical Information Systems (GIS) rely exclusively on querying for geographical information retrieval. Using Logical Information Systems (LIS) paradigm for information management and retrieval, we propose a more flexible organisation of vectorial geographical data at a thiner level since it is centered on the geographical object. Our data model allows to consider every collections of geographical objects that share a common description. Geographical objects descriptions mix spatial and non-spatial properties that are handled by specialized logics. Especially, a spatial logic has been designed to test the inclusion of the different kinds of geometrical description (i.e. polygon, line and point) and to reason on derived properties such as the area or the length. Our navigation model allows to freely combine querying and navigation on geographical data. More particularly, the navigation model relies on three different views over the geographical data: 1) the current selection is described intentionnaly by the current query, 2) its extension is represented graphically on the geographical map, and 3) the navigation tree gathers the properties describing objects of the current selection. These properties also serve as navigation links to refine or generalize the current query. The data and the navigation models have been implemented in the GEOLIS prototype, which has been used to lead experiments on a real dataset.

6.2 Dynamic Taxonomies and Faceted Search: Theory, Practice, and Experience

Participants: Sébastien Ferré.

The following is the backcover summary of a book on dynamic taxonomies and faceted search, edited by G.M. Sacco and Y. Tzitzikas. We contributed into 5 chapters [7, 11, 8, 10, 9] of this book on the contribution of logical information systems and logic functors.

"Current access paradigms for the Web, i.e., direct access via search engines or database queries and navigational access via static taxonomies, have recently been criticized because they are too rigid or simplistic to effectively cope with a large number of practical search applications. A third paradigm, dynamic taxonomies and faceted search, focuses on user-centered conceptual exploration, which is far more frequent in search tasks than retrieval using exact specification, and has rapidly become pervasive in modern Web data retrieval, especially in critical applications such as product selection for e-commerce. It is a heavily interdisciplinary area, where data modeling, human factors, logic, inference, and efficient implementations must be dealt with holistically.

Sacco, Tzitzikas, and their contributors provide a coherent roadmap to dynamic taxonomies and faceted search. The individual chapters, written by experts in each relevant field and carefully integrated by the editors, detail aspects like modeling, schema design, system implementation, search performance, and user interaction. The basic concepts of each area are introduced, and advanced topics and recent research are highlighted. An additional chapter is completely devoted to current and emerging application areas, including e-commerce, multimedia, multidimensional file systems, and geographical information systems.

The presentation targets advanced undergraduates, graduate students and researchers from different areas from computer science to library and information science as well as advanced practitioners. Given that research results are currently scattered among very different publications, this volume will allow researchers to get a coherent and comprehensive picture of the state of the art."

6.3 Organizing and browsing a collection of documents with Camelis

Participants: Sébastien Ferré.

Since the arrival of digital cameras, many people are faced with the challenge of organizing and browsing the overwhelming flood of photos their life produces. The same is true for all sorts of documents, e.g. emails, audio files. Existing systems either let users fill query boxes without any assistance, or drive them through rigid navigation structures (e.g., hierarchies); or they do not let users put annotations on their documents, even when this would support the organization and retrieval of any documents on customized criteria. We present [4] a tool, Camelis, that offers users with an organization that is dynamically computed from documents and their annotations. Camelis is designed along the lines of Logical Information Systems (LIS), which are founded on logical concept analysis. Hence, (1) an expressive language can be used to describe photos and query the collection, (2) manual and automatic annotations can be smoothly integrated, and (3) expressive querying and flexible navigation can be mixed in a same search and in any order. This presentation is illustrated on a real collection of more than 5,000 photos.

In the domain of Dynamic Taxonomies and faceted search, we show that Camelis extends the navigational capabilities. In addition to the zoom-in navigation mode (e.g., replacing Europe by France in the query), we present other navigation modes for less directed and more exploratory browsing of a document collection. The presented navigation modes are zoom-out (e.g., replacing France by Europe), shift (e.g., replacing France by Spain), pivot (e.g., switching location and time), and querying by examples. These modes all correspond to query transformations, and make use of boolean operators. The current focus of the search is always clearly specified by a query, and complex boolean queries can be constructed through navigation only, i.e. by successive selection of navigation links.

6.4 Efficient Browsing and Update of Complex Data Based on the Decomposition of Contexts

Participants: Sébastien Ferré.

Formal concept analysis is recognized as a good paradigm for browsing data sets. Besides browsing, update and complex data are other important aspects of information systems. To have an efficient implementation of concept-based information systems is difficult because of the diversity of complex data and the computation of conceptual structures, but essential for the scalability to real-world applications. We decompose contexts into simpler and specialized components: logical context functors [17]. We demonstrate this allows for scalable implementations, updatable ontologies, and richer navigation structures, while retaining genericity.

6.5 Data Mining for Fault Localization

Participants: Peggy Cellier, Mireille Ducassé, Sébastien Ferré, Olivier Ridoux.

Most dynamic fault localization methods aim at totally ordering program elements from highly suspicious to innocent. This ignores the structure of the program and creates clusters of program elements where the relations between the elements are lost. We have proposed a data mining process that computes program element clusters and that also displays dependencies between program elements. Experimentations show that our process gives a comparable number of lines to analyze than the best related methods while providing a richer environment for the analysis. We have also shown that the method scales up by tuning the statistical indicators of the data mining process [14].

6.6 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 [15]. 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.

6.7 Alert Correlation in Intrusion Detection

Participants: Mireille Ducassé.

Managing and supervising security in large networks has become a challenging task, as new threats and flaws are being discovered on a daily basis. This requires an in depth and up-to-date knowledge of the context in which security-related events occur. Several tools have been proposed to support security operators in this task, each of which focuses on some specific aspects of the monitoring. Many alarm fusion and correlation approaches have also been investigated. However, most of these approaches suffer from two major drawbacks. First, they only take advantage of the information found in alerts, which is not sufficient to achieve the goals of alert correlation, that is to say to reduce the overall amount of alerts, while enhancing their semantics. Second, these techniques have been designed on an ad hoc basis and lack a shared data model that would allow them to reason about events in a cooperative way. We have proposed a federative data model for security systems to query and assert knowledge about security incidents and the context in which they occur. This model constitutes a consistent and formal ground to represent information that is required to reason about complementary evidences, in order to confirm or invalidate alerts raised by intrusion detection systems [6].

6.8 Optional and iterated types for Pregroup Grammars

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 [12, 3, 13]. The use of Camelis in this context is also presented in these papers.

Our interest is also in learning categorial grammars [2]. 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 [5] (to appear in a journal).

6.9 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 NXP Software

Participants: Sébastien Ferré.

A collaboration contract has been signed between Sébastien Ferré (LIS team), Laurent Amsaleg and Patrick Gros (TEXMEX team) on one hand, and the company NXP Software on the other hand. LIS and TEXMEX have agreed to provide advices on the feasibility of the portability of multimedia data indexation and exploration technologies on present or future mobile devices.

7.2 ECOMER

Participants: Mireille Ducassé, Sébastien Ferré.

At this date (19/11/2009), this contract is still awaiting final approval.

The program ECOMER is a grouping of several projects submitted to the call for projects « Réduction de la consommation d'énergie à la pêche » (energy consumption savings in fishery). It is a collaborative project between industrial partners and academic partners. Its objectives are:

- to provide fishers with hardware and software tools that allow them to better control their oil consumption,
- 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 role of the LIS team will be to provide a user interface 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

- The team LIS collaborates with the Belgian company *Mission Critical IT* on the use of logical information systems in ontology-driven engineering. Mission Critical IT has submitted a European Project, ODESSC (Ontology-Driven Engineering for Software Services on the Cloud), in which the team LIS is a partner.

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 are organized with communications by local actors or by other specialists on this issue.
- 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".
- Pierre Allard, Alice Hermann, and Sébastien Ferré have took part in the "treillis clermontois", the annual seminar of the French FCA community.

9 Dissemination

9.1 Involvement in the Scientific Community

- Olivier Ridoux has served in several doctoral and habilitation committees: Gilles Trédan « Structures et systèmes répartis », Pierre Crégut « Contribution à la vérification des logiciels » and Thomas Genet « Analyse d'atteignabilité en réécriture pour la vérification de programmes ». He also served in several recruitment committees in University of Rennes 1 (President of 3 commitees), University of Rennes 2 (1 committee) and University of Nantes (1 committee).
- Mireille Ducassé has served in the program committee of ICLP 2009 (International Conference on Logic Programming), Pasadena, USA. At that conference, she has been invited to give a tutorial on "(C)LP Tracing and Debugging" [16]. She has been in one PhD committee : Grégoire Jacob, Université de Rennes 1. 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.
- Sébastien Ferré has been one of the two Program Chairs for the International Conference on Formal Concept Analysis (ICFCA), which took place in May in Darmstadt, Germany [1]. He has been named as a Program Chair for the International Conference on Conceptual Structures (ICCS), to be held in July 2010 in Malaysia.

  He has served as an external reviewer for the journals: The Computer Journal (CompJ), AMAI (Annals of Mathematics and Artificial Intelligence), and Information Sciences (INS). He has participated in the PhD defense committee of Olivier Bedel, as a co-director, and he is also a member of the PhD committee of Nicolas Lebreton.

  Sébastien Ferré has also served in the recruitment committee ("comité de sélection") for the recruitement of two assistant professors at the National Institute of Applied Sciences (INSA).
- Annie Foret has been a program committee member of the *Formal Grammar* 2009 International Conference.

9.2 Teaching

- Olivier Ridoux was the head of IFSIC (Institut de Formation Supérieure en Informatique et Communication - the Computing Science department at University of Rennes 1) until February 2009, and then 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 is the head of the computer science department of the INSA of Rennes. She is also an elected member of the board of directors ("Conseil d'administration") of the Insa of Rennes. She has been a member of two recruitment committees ("comités de sélection") in computer science at the Insa of Rennes.

  At Insa, she teaches compilation and formal methods for software engineering (with the "B formal method") at Master 1 level of Insa. She leads an exercise of participatory design based on the work of Wendy Mackay from the "In Situ" project of Inria Futurs. She contributes to a course on risk analysis at Licence 2 level. She has presented the participatory design course at a French pedagogical event [19].

- 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. Until August, he has been in delegation at the CNRS.

- Annie Foret teaches university courses including formal logic, functional programming, and databases.

- Peggy Cellier has been ATER (Attachée Temporaire à l'Enseignement et la Recherche) at University of Rennes 1. She teaches algorithmics of graphs at Master 1 level of DIIC (Diplôme d'Ingénieur de l'IFISIC), object-oriented modelling (UML, JUnit) at Master 1 level of IFSIC, and an introduction to computability and complexity at the Licence level.

- Pierre Allard teaches datamining and programming in C at the INSA of Rennes.

- Olivier Ridoux and Sébastien Ferré have animated one day during the "Journées pédagogiques de l'IFSIC" (pedagogical days of IFISC) in July, on Indexation and Logical Information Systems.

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] Y. Padioleau, O. Ridoux, “A Logic File System”, *in: Usenix Annual Technical Conference*, 2003.

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

Books and Monographs

[1] S. Ferré, S. Rudolph (editors), *Formal Concept Analysis, 7th International Conference, ICFCA 2009, Darmstadt, Germany, May 21-24, 2009, Proceedings*, *LNCS 5548*, Springer, 2009.

Doctoral dissertations and “Habilitation” theses

[2] O. Bedel, *GEOLIS : Un Système d information logique pour l organisation et la recherche de données géolocalisées*, PhD Thesis, Thèse de l’université de Rennes 1, 22 janvier 2009, coencadrée par Olivier Ridoux et Sébastien Ferré.

Articles in referred journals and book chapters

[3] D. Béchet, A. Foret, “A Pregroup Toolbox for Parsing and Building Grammars of Natural Languages”, *Linguistic Analysis Journal, Vol. 36*, to appear.

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

[5] A. Foret, “A modular and parameterized presentation of pregroup calculus”, *Information and Computation Journal*, to appear.

[6] B. Morin, L. Mé, H. Debar, M. Ducassé, “A Logic-based model to support alert correlation in intrusion detection”, *Information Fusion 10*, 4, October 2009, p. 285–299.

[7] G. M. Sacco, S. Ferré, Y. Tzitzikas, *Dynamic Taxonomies and Faceted Search: Theory, Practice, and Experience*, *The Information Retrieval Series*, *25*, Springer, 2009, ch. 3 - Comparison with Other Techniques.

[8] G. M. SACCO, S. FERRÉ, *Dynamic Taxonomies and Faceted Search: Theory, Practice, and Experience*, *The Information Retrieval Series*, *25*, Springer, 2009, ch. 5 - Extensions to the Model.

[9] G. M. SACCO, S. FERRÉ, *Dynamic Taxonomies and Faceted Search: Theory, Practice, and Experience*, *The Information Retrieval Series*, *25*, Springer, 2009, ch. 9 - Applications and Experiences.

[10] G. M. SACCO, Y. TZITZIKAS, S. FERRÉ, *Dynamic Taxonomies and Faceted Search: Theory, Practice, and Experience*, *The Information Retrieval Series*, *25*, Springer, 2009, ch. 8 - System Implementation.

[11] M. STEFANER, S. FERRÉ, S. PERUGINI, J. KOREN, Y. ZHANG, *Dynamic Taxonomies and Faceted Search: Theory, Practice, and Experience*, *The Information Retrieval Series*, *25*, Springer, 2009, ch. 4 - User Interface Design.

Publications in Conferences and Workshops

[12] D. BÉCHET, A. FORET, "(PPQ) : a pregroup parser using majority composition", *in: Proceedings of Parsing with Categorial Grammars, ESSLLI workshop, 20-24 July, 2009 in Bordeaux, France*, 2009.

[13] D. BÉCHET, A. FORET, "Une boîte à outils pour développer et utiliser les grammaires de prégroupe", *in: journée ATALA : Quels analyseurs syntaxiques pour le français ?*, 2009.

[14] P. CELLIER, M. DUCASSÉ, S. FERRÉ, O. RIDOUX, "DeLLIS: A Data Mining Process for Fault Localization", *in: Int. Conf. Software Engineering (SEKE)*, Knowledge Systems Institute Graduate School, p. 432–437, 2009.

[15] M. DUCASSÉ, S. FERRÉ, "Aide à la décision multicritère : cohérence et équité grâce à l'analyse de concepts", *in: Modèles et Apprentissage en Sciences Humaines et Sociales*, Juin 2009.

[16] M. DUCASSÉ, "(C)LP Tracing and Debugging", *in: 25th International Conference on Logic Programming, ICLP 2009*, P. M. Hill, D. S. Warren (editors), *Lecture Notes in Computer Science*, *5649*, Springer, p. 38, July 2009.

[17] S. FERRÉ, "Efficient Browsing and Update of Complex Data Based on the Decomposition of Contexts", *in: Int. Conf. Conceptual Structures*, S. Rudolph, F. Dau, S. O. Kuznetsov (editors), *LNCS 5662*, Springer, p. 159–172, 2009.

Internal Reports

[18] S. FERRÉ, "Navigating the Semantic Web with Logical Information Systems", *internal publication number 1934*, Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA), August 2009, `http://www.irisa.fr/centredoc/publis/PI/2009/navigating-the-semantic-web-with-logical-information-systems`.

Miscellaneous

[19] M. DUCASSÉ, "Cours de conception participative dans la formation d'ingénieur en informatique", Poster aux Rencontres "Pédagogie et Formations d'Ingénieurs", Mars 2009, INSA de Toulouse.