Ontological Issues in the Representation and Presentation of Mathematical Concepts

Armin Fiedler[1], **Andreas Franke**[1], **Helmut Horacek**[1], **Markus Moschner**[1], **Martin Pollet**[1] and **Volker Sorge**[2]

Abstract. Major purposes underlying the functionality of formal systems include reasoning services and presentation facilities, prominently their systematic coordination. An important role in this coordination is played by the ontology and its underlying organization principles exhibited in the systems' knowledge bases. We address this issue by presenting specific components of our proof development system ΩMEGA, which combines reasoning facilities for handling mathematical proofs with presentation capabilities in natural language. These components are the mathematical knowledge base of ΩMEGA and the linguistic knowledge base of the attached proof explanation system *P.rex*. We feature their ontological principles, modeling coverage, and their interoperability. Interfacing reasoning and presentation skills is crucial for increasing the quality in illustrating the results of formal inference systems.

1 INTRODUCTION

Major purposes underlying the functionality of formal systems include reasoning services and presentation facilities, in particular their systematic coordination. An important role in this coordination is played by the ontology and its underlying organization principles exhibited in the systems' knowledge bases, prominently the interoperability across several knowledge bases.

We address this issue by a case study, presenting specific components of our proof development system ΩMEGA [14]. ΩMEGA is a mathematical assistant tool that supports proof development at a user-friendly level of abstraction. The system combines interactive and automated proof construction in mathematical domains. Figure 1 illustrates the architecture of ΩMEGA: several independent modules are connected via the mathematical software bus MATHWEB-SB [5]. An important benefit is that MATHWEB modules can be distributed over the Internet and are accessible by other distant research groups as well. The core of ΩMEGA is the proof plan data structure $\mathcal{PDS}$ [3], the proof planner MULTI [11], the suggestion mechanism Ω-ANTS [2], and a hierarchy of mathematical theories, represented by a mathematical data base, which constitutes the basic mathematical ontology of the system.

Various heterogeneous external reasoning systems with complementary capabilities are integrated into ΩMEGA (see the right side of Figure 1). ΩMEGA interfaces systems such as computer algebra systems (CASs), higher- (HO) and first-order (FO) automated theorem proving systems (ATPs), constraint solvers (CSs), and model generators (MGs). Their use is twofold: they may provide a solution to a subproblem, or they may give hints for the control of the proof search. These systems are either connected directly or indirectly via proof transformation modules, in order to orchestrate their use, and to integrate their results. The output of the incorporated reasoning systems is translated and inserted as sub-proofs in ΩMEGA's central proof plan data structure, which maintains proof plans simultaneously at different levels of abstraction. This is beneficial for interfacing systems that operate at divergent levels of abstraction as well as for a human oriented display and inspection of a partial proof. From an ontological perspective, the mathematical theories in ΩMEGA provide some sort of normative ontology for mathematical concepts, to which concepts of external systems must be related.

User interaction is supported by the graphical user interface $\mathcal{LOUI}$ [15] and the proof explainer *P.rex* [4] (see the left side of Figure 1). The latter is a particular feature of ΩMEGA, since it provides proof explanations in natural language, in interactive and adaptive form. In order to be able to describe mathematical concepts in context, *P.rex* needs its own ontology, organized on the demands of natural language presentations.

In the following, we address the two major subsystems involved in ontological issues, the knowledge bases of ΩMEGA and *P.rex*, in dedicated sections. Thereby, we feature their divergent ontological principles and complementary modeling coverage. Next, we address their interoperability. Finally, we list case studies carried out with ΩMEGA and we sketch relations to other approaches to ontology.

2 THE REPRESENTATION COMPONENT

The organization of ΩMEGA's knowledge base is motivated by the following observations. The statement of a mathematical theorem can depend on the availability of a possibly large set of definitions of mathematical concepts, which in turn, may themselves depend on other concepts. Moreover, previously proven theorems or lemmas may be reused within the context of a proof. Going beyond pure representation purposes, a formal reasoning system needs access to other forms of knowledge, including inference steps, such as tactics, and information about control knowledge for automated reasoners

[1] Computer Science Department, Saarland University, P.O.Box 151150, D-66041 Saarbrücken, Germany
[2] School of Computer Science, University of Birmingham, Birmingham B15 2TT, UK

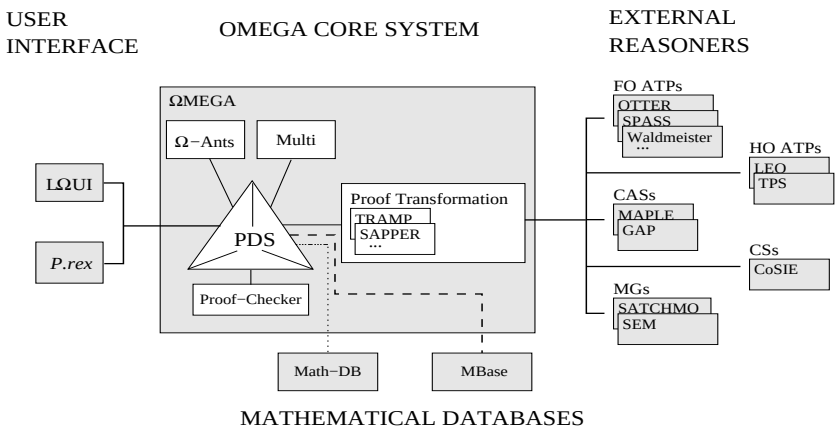

Figure 1. The architecture of the ΩMEGA proof assistant. Thin lines denote internal interfaces, thick lines denote communication via MATHWEB-SB. ΩMEGA has access to its local mathematical database (thin dotted line) and MBASE (thick dashed line).

and about human-oriented presentation knowledge. To support reasoning in such an environment, the main requirement is an efficient access to definitions and axioms, thereby avoiding redundancy in storing that information to ease maintenance. For meeting this demand, we have decided to use an inheritance network. It provides most effective percolation of information to build the core semantics of each mathematical object. In contrast, it ignores their meaning and interrelations other than componential ones, since this knowledge is irrelevant for the intended use.

In order to define a mathematical concept, which is a structured object with several properties, a number of options typically exist, depending on the subfield in question. The prototypical alternative lies between the use of strictly *minimal* properties in logical terms and *commonly used* ones. The latter are stronger than the former ones and might contain redundancies, but they may be better known in the field or preferred for some social reasons. Hence, it is logically inferable that these properties hold, once it is known that the minimal properties hold. Consequently, it may be possible to define a mathematical concept in several equivalent ways within the same theory, each of which may prove useful for reasoning purposes.

Let us consider a well-known example for which a number of alternative, yet equivalent, definitions exist, the notion of a *group*. A group is usually defined as a nonempty set together with an operation, where the operation is closed and associative for the elements of this set. Moreover, there exists a unit element, and for each elements of this set an inverse element. While this definition is the most common one, it is not minimal: for instance we can replace the existence of the unit element or of inverse elements with respective weaker properties that only a left unit element or left inverse elements exist. Still we would arrive at a definition of a group, equivalent to the first. Moreover, we can replace the postulation of the unit element and the inverses entirely by the single axiom: *For all* $a, b \in G$ *exist* $x, y \in G$ *such that* $a \circ x = b$ *and* $y \circ a = b$.

We can also define the notion of a group via larger, less concrete algebraic structures. Hereby we have several possible ways to arrive at a definition as is outlined in Figure 2.

Let G be a nonempty set and let $\circ$ be a binary operation on G. The following assertions are equivalent:

- $(G, \circ)$ is a group.
- $(G, \circ)$ is a monoid and every element of $a \in G$ has an inverse.
- $(G, \circ)$ is a loop and $\circ$ is associative.
- $(G, \circ)$ is both a quasi-group and a semi-group.

The three variants except to the first one, each correspond to one path leading to *Group* in Figure 2, where the link coming from *Loop* adds the property of associativity, and the link coming from *Monoid* adds the property of the existence of inverses. In order to establish these equivalent representations within the system in a justified manner, the equivalences entailed must be proved.

To support this task, we are currently replacing the knowledge base by the system MBASE [6], which is a distributed mathematical knowledge base designed to make storage and access through queries efficient. The specification of correspondences on the ontological level is to be carried out interactively and supported by theorem proving devices.

In order to store and manipulate these kinds of information, MBASE distinguishes several categories of information objects, on which the structure of the underlying data base model is grounded:

- Definitions for associating meanings to symbols in terms of already defined ones.
- Assertions, which are logical sentences, including axioms, theorems, conjectures and lemmas, distinguished according to pragmatic or proof-theoretic status. item Proofs, encapsulating the actual proof objects in various formats, including formal, informal, system-dependent proof scripts, and even natural language formats.
- Examples, due to their importance in mathematical practice.
- Theories, which allow grouping of mathematical objects and knowledge and the introduction of inheritance between theories.
- Inference steps, in form of system-dependent programming code, as basic calculus rules and compound steps.
- Human-oriented (technical) knowledge, such as names for theorems, and specifications for notation and symbol handling.

The data base model contains some relations between objects of these kinds onto which inheritance is made. These include

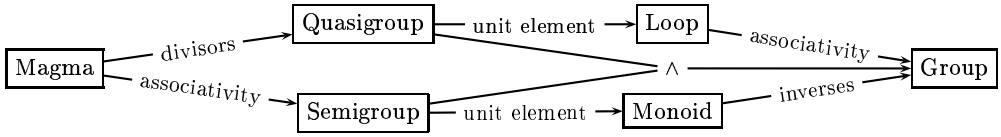

Figure 2. Constructing groups via more general algebras

- Definition-entailment, to relate defined symbols to others and, ultimately, to symbols marked as primitive.
- Depends-on/Local-in, which specify dependency and locality information for primary knowledge items. The relation makes explicit the use of symbols in a definition or assertion, and the application of lemmas in a proof.
- Theory-inheritance, which specifies the organization of mathematical subdomains and the associated inheritance of their properties.

The purpose of the categories described before is to ensure correctness in the process of building mathematics on computers and at the same time ease this process. The introduced entities should help to construct proofs but at the same time avoid additional efforts in their administration. Definitions as explicit entities allow for the abbreviation of terms and formulas. At the same time it has to be ensured that the definition functionality will not introduce inconsistencies. The explicit hierarchy of theories allows for grouping of mathematical knowledge and helps the user to keep an overview. On the other hand one has to decide on how to structure the formalizations so that it will be useful for the proof construction in theories that inherit these theories. As described for the equivalent definitions of the concept *Group* there seems to be no best formalization that covers every aspect and the rigidity of the theory hierarchy has to be equalized by mechanisms as the development graph [9], which allows us to manage a later change of underlying formalizations.

The given categorization is not specific to ΩMEGA but holds as well for other proof development systems which maintain a library of mathematical knowledge. The systems can differ in the following aspects:

- finer classification of objects, e.g. recursive definitions in COQ [16],
- the language used for terms and formulas, e.g typed lambda calculus in ΩMEGA or set theory in MIZAR [13],
- the available functionality, e.g. theory interpretations as mechanism for the reuse of theorems in IMPS [10],
- the basic logical calculus and the formalism to build more complex inference steps.

3 THE PRESENTATION COMPONENT

P.rex is a proof explainer attached to ΩMEGA, which is responsible for expressing a mathematical proof in terms of natural language text, interleaved with formulas. A particular capability of *P.rex* is to explain a proof step at different levels of abstraction, initially at the most abstract level that the user is assumed to know. The system reacts flexibly to questions and requests. While the explanation is in progress, the user can interrupt *P.rex* anytime, if the current explanation is not satisfactory. *P.rex* analyzes the user's interaction and enters into a clarification dialog when needed to identify the reason why the explanation was not satisfactory and re-plans a better explanation, for example, by switching to another level of abstraction. During the presentation process, *P.rex* constructs a *discourse structure tree* to organize the utterances to be produced. Each utterance is represented by a tree, called *Text Structure*, that encodes its linguistic specification (cf. [8] for details).

The specifications in a proof at some level of abstraction, which *P.rex* is supposed to convert into a natural language presentation, contain references to mathematical concepts. For presentation purposes, these are organized in terms of *semantic categories*. The main role of the *semantic categories* is to provide vocabularies, which specify type restrictions for nodes of the Text Structure. They define how separate Text Structures can be combined, and ensure that the planner only build expressible Text Structures. For instance, if tree A should be expanded at node n by tree B, the resulting type of B must be compatible to the type restriction attached to n. Following Panaget [12], however, we split the type restrictions into two orthogonal dimensions: the ideational dimension in terms of the *Upper Model* [1], and the *hierarchy of textual semantic categories* to be discussed below. As in other conceptual hierarchies, the relation between the top level node and its immediate successors is realized as a specialization, rather than as an instantiation. Since the meaning of this relation is never communicated, this inaccuracy is tolerable.

Technically speaking, the Text Structure in *P.rex* is a tree recursively composed of kernel subtrees or composite subtrees: An atomic *kernel subtree* has a head at the root and arguments as children, representing basically a predicate/argument structure. *Composite subtrees* can be divided into two subtypes: the first has a special *matrix* child and zero or more *adjunct* children and represents linguistic hypotaxis, the second has two or more *coordinated* children and stands for parataxis.

Each node is typed both in terms of the Upper Model and the hierarchy of textual semantic categories. The Upper Model is a domain-independent property inheritance network of concepts that are hierarchically organized according to how they can be linguistically expressed. Figure 3 shows a fragment of the Upper Model in *P.rex*. For every domain of application, domain-specific concepts must be identified and placed as an extension of the Upper Model. In the domain of mathematics, most domain-specific concepts (formally: one-place predicates) are placed under the concept *non-conscious thing*, except to some real-world concepts, which appear in mathematical subtheories for puzzles. Moreover, domain properties appear under *discrete place relations*. The organization of these items is driven by specialization relations and by type restrictions of the fillers of relations. Note that these principles are complementary to those of the mathematical knowledge base, which leads to differently structured (typically flatter) hierarchies.

The hierarchy of textual semantic categories is also a domain-independent property inheritance network. The con-

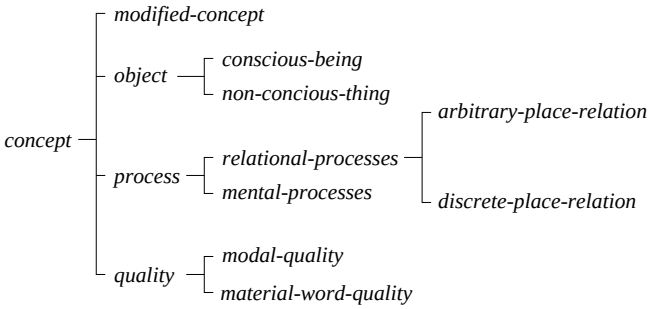

Figure 3. A Fragment of the Upper Model in *P.rex*

cepts are organized in a hierarchy based on their textual realization. For example, the concept *clause-modifier-rankingI* is realized as an adverb, *clause-modifier-rankingII* as a prepositional phrase, and *clause-modifier-embedded* as an adverbial clause. Figure 4 shows a fragment of the hierarchy of textual semantic categories.

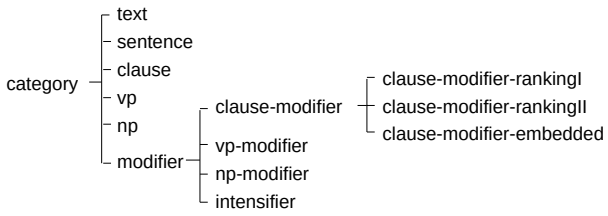

Figure 4. A Fragment of the Hierarchy of Textual Semantic Categories in *P.rex*

The equivalences in defining groups or similar mathematical objects, as exposed in the previous section, are not reflected by the linguistic knowledge bases. Therefore, building an alternative representation, which may be motivated by various presentation goals, has to be carried out on the basis of the proof representation prior to verbalization, which is a matter of future work. In contrast, *P.rex* is able to produce different phrasings for some mathematical properties, such as associativity, the adjective 'associative' or the noun 'associativity', enabled through the interplay of the Upper Model, the lexicon, and the textual semantic categories. This distinction is not reflected by the mathematical knowledge base.

4 INTEROPERABILITY

When comparing the organization principles of MBASE and the Upper Model of *P.rex*, it is interesting to see that the reasoning purposes motivating the organization in MBASE do not require specialization in their hierarchical structuring, which is quite common for other ontologies. In the Upper Model, however, the way these specializations are set up is oriented on presentation purposes rather than on mathematical properties. Conversely, the precise logical definitions percolated through the inheritance network expressing the semantics of the mathematical concepts and relations are not accessible to the Upper Model.

Through these definitions extending the Upper Model, the mathematical concepts are in some sense re-represented, with links to their counterparts in the mathematical knowledge base. The associated maintenance effort is unavoidable, since for each mathematical notion newly modeled in the mathematical theory part, the corresponding counterpart on the linguistic side must be appropriately integrated in the linguistic knowledge bases. Only for those cases, where a mathematical concept requires no linguistic knowledge other than reference by its name, we use a short-cut for handling interoperability between mathematical and linguistic knowledge bases. Mathematical concepts of this kind are mapped onto a catch-all concept in the linguistic knowledge base, and reference by name to this concept is done by passing through its name from the mathematical knowledge base. The main advantage of this realization lies in some degree of independency, that is, extending the mathematical knowledge base and re-running the system without adapting the linguistic knowledge base accordingly is possible, if the resulting (in some cases, temporary) limitation in the presentation quality is acceptable. Maintaining this short-cut will be impossible when the extensions described below will be addressed.

Apart from this basic integration, purposes of presentation impose some additional demands that cannot be met adequately by the inheritance network and by the Upper Model in its present state. Those which can be met easier by system extensions are the following:

- Items preferred for referring expression need to be marked. For example, the terms *Group* and *Semigroup* are more common than *Monoid*, which identifies a similar algebraic structure. The more common terms should be preferred in descriptions even though this may require extended descriptions.
- Conceptual equivalences must be made explicit to avoid redundancy in presentations. As demonstrated in Section 2, for example, various possibilities to define a group in algebraic terms, and switching between definitions might easily result in a strange rephrasing of an obvious equivalence. For reasoning purposes, the equivalence is established by an inference step or, in more complicated cases, by a subproof.
- Additional, highly special conceptual definitions that have no relevance for reasoning purposes may improve the presentation capabilities significantly. Typically, axioms that are expressed as compound or nested rules are good sources for building such definitions, which relate to subexpressions that appear in that axiom and are likely to be described as intermediate results in proofs.

The techniques described so far mainly support the coordination of static knowledge originating from heterogeneous sources for problem solving purposes. For presentation purposes, there are two fundamental shortcomings of the present representation, which severely limit presentation capabilities:

- The connection between mathematical objects and Upper Model objects is too simple, since it is restricted to one-to-one correspondences. This is meaningfully applicable to domain terms, but not to some relations and inference rules.
- The dynamic knowledge (e.g., proofs, examples) is represented too coarse-grained and on a superficial level only, which limits variations for presenting it.

These shortcomings are as fundamental as they are deliberate, since the complexity of the design and development tasks for MBASE is high enough. For future extensions, these are good candidates for linking more linguistically oriented tools.

5 CASE STUDIES AND AVAILABILITY

The ΩMEGA system is available on the Internet at `http://www.ags.uni-sb.de/~omega`. It has been evaluated in multiple case studies, which have been carried out for varying purposes motivated by theorem-proving issues; as a consequence, ΩMEGA's knowledge bases contain representation fragments of a variety of subdomains, including:

ε-δ-proofs: Limit theorems and assertions about functions and sequences and continuity of functions.

Automatic classification of residue-class structures: Proofs of properties of residue classes and about isomorphisms between residue class structures.

Interactive proof planning: A combination of Ω-ANTS and MULTI was used to support the interactive exploration of residue classes and to prove theorems about properties of group homomorphisms in student exercises.

Set theory: Ω-ANTS has been applied with ND rules, an automated theorem prover, and a model generator to prove or disprove set equations.

Group theory: Equivalence of different group definitions, uniqueness of the unit element and inverse element.

The software components referred to in this paper are further described at the following websites:

- ΩMEGA: `http://www.ags.uni-sb.de/~omega/soft/omega`
- *P.rex*: `http://www.ags.uni-sb.de/~prex`
- MBASE: `http://www.mathweb.org/mbase`

6 CONCLUSION AND DISCUSSION

In comparison to other domains, high quality representations are required, so that they can only be produced in a handcrafted manner. Apart from that quality aspect, building ontologies for the domain of mathematics can in some sense be considered less difficult than a similar task for another domain:

- Vagueness does not play a role at all.
- There is a high degree of agreement about the domain concepts; discrepancies merely concern alternative representation variants, formats, and conventions.
- Although the domain as a whole is very large, it can be reasonably well broken down into subdomains of manageable size.

In some sense, the domain is atypical, since the domain objects are completely artificial from the reasoning perspective, with the exception of puzzle subdomains. From the presentation perspective, however, the representation purposes are not much different from other domains. They do differ in terms of richness and degrees of variety, which is more limited in our domain in comparison to narratives. Consequently, our Upper Model is merely a copy of the one used by Penman [1], oriented on the purpose of expressibility. The main emphasis in our approach is establishing a basic interoperability.

Altogether, we have shown that representation and presentation purposes in the domain of mathematical can be met by distributed knowledge bases, with different organization principles, complementary coverage, and minimal linking, which reduces the maintenance effort. Not surprisingly, the present, simple design, imposes limitations on the overall functionality. In general, shortcomings in the presentation part raise demands that can be met best by extensions in the mathematical representation component or by some module that manipulates these representations. We have identified several of these shortcomings, which we will address in future work. When doing this, we expect aspects of formal ontologies, such as identity and unity [7], to become relevant for our presentation part as well.

REFERENCES

[1] J. A. Bateman, R. T. Kasper, J. D. Moore, and R. A. Whitney, 'A general organization of knowledge for natural language processing: the Penman upper model', Technical report, University of Southern California, Information Science Institute, (1990).

[2] C. Benzmüller and V. Sorge, 'Ω-ANTS – An open approach at combining Interactive and Automated Theorem Proving', in *Proc. of Calculemus-2000*, eds., M. Kerber and M. Kohlhase. AK Peters, (2001).

[3] L. Cheikhrouhou and V. Sorge, '$\mathcal{PDS}$ — A Three-Dimensional Data Structure for Proof Plans', in *Proc. of ACIDCA'2000*, (2000).

[4] A. Fiedler, 'Dialog-driven adaptation of explanations of proofs', in *Proc. of the 17th International Joint Conference on Artificial Intelligence (IJCAI)*, ed., B. Nebel, pp. 1295–1300, Seattle, WA, (2001). Morgan Kaufmann.

[5] A. Franke and M. Kohlhase, 'System description: MATHWEB, an agent-based communication layer for distributed automated theorem proving', in *Proc. of CADE-16*, ed., H. Ganzinger, number 1632 in LNAI. Springer, (1999).

[6] A. Franke and M. Kohlhase, 'System description: MBASE, an open mathematical knowledge base', in *Proc. of CADE-17*, ed., D. McAllester, number 1831 in LNAI. Springer, (2000).

[7] N. Guarino and C. Welty, 'Identity, unity, and individuality: Towards a formal toolkit for ontological analysis', in *Proc. of ECAI-2000*. IOS Press, (2000).

[8] X. Huang and A. Fiedler, 'Proof verbalization as an application of NLG', in *Proc. of the 15th International Joint Conference on Artificial Intelligence (IJCAI)*, ed., M.E. Pollack, pp. 965–970, Nagoya, Japan, (1997). Morgan Kaufmann.

[9] D. Hutter, 'Management of change in structured verification', in *Proceedings Automated Software Engineering (ASE-2000)*. IEEE Press, (2000).

[10] The IMPS online theory library. Internet interface at `http://imps.mcmaster.ca/theories/theory-library.html`.

[11] E. Melis and A. Meier, 'Proof Planning with Multiple Strategies', in *Proc. of CL-2000*, volume 1861 of *LNAI*. Springer, (2000).

[12] F. Panaget, 'Using a textual representational level component in the context of discourse or dialogue generation', in *Proc. of the 7th International Workshop on Natural Language Generation*, pp. 127–136, Kennebunkport, ME, (1994). INLG.

[13] Piotr Rudnicki, 'An overview of the mizar project', in *Proceedingsof the 1992 Workshop on Types and Proofs as Program s*, pp. 311–332, (1992).

[14] J. Siekmann, C. Benzmüller, V. Brezhnev, L. Cheikhrouhou, A. Fiedler, A. Franke, H. Horacek, M. Kohlhase, A. Meier, E. Melis, M. Moschner, I. Normann, M. Pollet, V. Sorge, C. Ullrich, C.-P. Wirth, and J. Zimmer, 'Proof development with ΩMEGA', in *Proc. of CADE-18*, Copenhagen, Denmark, (2002). forthcoming.

[15] J. Siekmann, S. Hess, C. Benzmüller, L. Cheikhrouhou, A. Fiedler, H. Horacek, M. Kohlhase, K. Konrad, A. Meier, E. Melis, M. Pollet, and V. Sorge, '$\mathcal{LOUI}$: $\mathcal{L}$ovely ΩMEGA $\mathcal{U}$ser $\mathcal{I}$nterface', *Formal Aspects of Computing*, **11**, 326–342, (1999).

[16] Coq Development Team, *The Coq Proof Assistant Reference Manual*, INRIA. see `http://coq.inria.fr/doc/main.html`.