Chapter 1

Excellence

This document is under construction

Bugs may kill. Would you fly in an autonomous plane fully driven by a piece of software that has not been formally verified? Or formally verified using a technology that Europe does not master? Today, the trust in critical systems relies on formal methods, in particular formal proofs, that guaranty the safety of the people using transportation systems—autonomous car, metros, trains, planes, etc.—, health systems—robotic surgery, etc.—, energy provided by nuclear plants, etc. The crucial role of formal proof is highlighted by several successes, like the correctness proof of the automatic Paris metro line 14 [**?**] or the detect and avoid system for unmanned aircraft system developed by Nasa [**?**].

Unfortunately, the development of formal methods is slowed down by the multiplicity of proof systems and the lack of a common theory used by these systems. This restricts interoperability, sustainability, certification, etc. each small community being centered around one theory and one system, and often re-doing work done elsewhere. For instance, the metro line 14 has been proved correct in ATELIER B, while the Nasa detect and avoid system for unmanned aircraft system has been proved correct in PVS.

There are around twenty major proof systems in the world (Figure 1.1) and making these systems interoperable would avoid work duplication, reduce the development time, and allow independent certification. After three decades dedicated to the developement of these systems, allowing such a cooperation between these systems is the next step in the developement of the formal proof technology.

ABELLA	ACL2
AGDA	HOL LIGHT
ATELIER B	IMPS
COQ	LEAN
FOCALIZE	NUPRL
HOL4	PVS
ISABELLE	TLA+
MATITA	
MINLOG	
MIZAR	
RODIN	

Figure 1.1: Some major proof systems. The European ones are in the first column. Those addressed in the project are underlined

1.1 Objectives

To foster the interoperability of proof systems and the sustainability and the certification of formal proofs, we propose to build an on line encyclopedia of formal proofs called LOGIPEDIA, that indicates which proof can be used in which system and, when it is the case, provides a version of the proof in the theory of this system. Such a project will increase networking activities between its members and also between the academic and industrial users of this encyclopedia. This encyclopedia will be accessible through a web browser from every country in Europe and beyond. This project will also trigger joint research activities between its users and between its developers.

LOGIPEDIA aims at including, in twenty years, all formal proofs developed at that time. To measure the level of integration of the library of a proof system in LOGIPEDIA, we propose to define a metric: *the* LOGIPEDIA *readiness levels*, that countsg six levels.

In the next four years, we propose to focus on the theory and library of 15 systems (Figure 1.1), and increase their LOGIPEDIA readiness levels, bringing 10 of them to level 5 or 6.

Such a project can only have a worldwide ambition. However, as a majority of proof systems are developed in Europe, Europe can take the lead on this project, so that the economic spinoffs from the project benefit the active participants mainly based in Europe. That is why the consortium gathers most of the European actors active on formal proof systems, and also proposes to develop links with non

European partners.

1.2 Relation to the work program

This section must explain how the proposal addresses the specific challenge and scope of that topic, as set out in the work programme. Page 53-55 of the call.

Research e-infrastructures for formal proofs Proofs systems and automated theorem provers are research infrastructure, as they allow computer scientists, logicians, and mathematicians to build and study formal proofs, just like particle accelerators allow physicists to build and study particles. Each proof system comes with its own library, and these libraries are also research infrastructures.

Currently these infrastructures are small, distributed and disconnected. Our project aims at building a large, worldwide infrastructure from the small ones by integrating them in an encyclopedia.

The integration effort is substantial because the proofs in the various libraries are expressed in different theories. But this integration effort is doable and contributes to the challenge to bring European researchers and engineers an effective and convenient access to the best research infrastructures in order to conduct research for the advancement of knowledge and technology.

A strating community This project has never been supported under FP7 or Horizon 2020 calls. It mobilize xxx research groups in Europe, that is almost all groups working on formal proof technology, as well as two clubs of academic and industrial users.

Compliance policy with EU regulations A data management plan will be provided according to Inria's compliance policy with EU regulations. We also engage ourselves to make the evolution of the logipedia e-infrastructure compliant with the European charter for access to research infrastructures.

Networking activities, transnational access, joint research activities This project fosters networking activities and joint research activities that currently do exist in Europe, but need to be strengthened, through a joint project.

The transnational access is more or less obvious as the encyclopedia is on-line and accessible through a web browser.

Towards standardization This project opens the door for a possible standardization of proof languages, although such an effort would be premature today.

Inter-disciplinarity This project that includes mathematicians, logicians, and computer scientists is clearly inter-disciplinary.

1.3 Concept and methodology

State of the art. Formal proofs and systems manipulating such proofs, have become central tools both in safety and security, as shown by several major successes: the correctness proof of the Paris metro line 14, the Nasa detect and avoid system for unmanned aircraft system proved correct in PVS, the proved operating system seL4 [**?**], the proved C compiler CompCert [**?**]. In the realm of pure mathematics also, formal proofs have permited to convice ourselves of the correctness of several complex proofs, such as those of the Feit-Thompson theorem [**?**], Hales' theorem [**?**], etc. The development of these formal proofs has led to the construction of huge libraries, totalizing millions of hours of work, that are a significant part of mankind's mathematical assets.

Software development has always been accompanied with the definition of standards, that make systems interoperable and data sustainable. For example, web browsers are interoperable and websites are sustainable because they all comply with the html standard. The area of formal proofs is however an exception. So, while we had in the past (informal) proofs of Pythagoras' theorem or Fermat's little theorem, we now have (formal) proofs of these theorems in COQ, MATITA, or HOL LIGHT, etc. This lack of standards is the major weakness of this area, as it jeopardizes the usability and the sustainability of these libraries. Indeed, each library is specific to a proof system, most of the time even to some version of this system. A library developed in one system cannot, in general, be used in another and when the system is no more maintained, the library may disappear. Being a major weakness, standard design is also a major challenge.

On more philosophical grounds, this lack of a standard jeopardizes the universality of logical truth, just like, in the 19th century, non-Euclidean geometries did, as some statements could be true in one geometry, but not in others. This lack of universality severely limits the spreading of formal proofs in non-specialist communities. For instance, while logic is taught to undergrad students, it is difficult to teach them formal proofs, as this requires the choice of a specific language, theory and system that orients the course according to these language, theory, and

system. The same could be said for the use of formal proofs in industry or by working mathematicians. So another goal of this project is to make formal proofs accessible to a much larger community, as standards often do.

At the beginning of the 20th century, a remediation has been found to the crisis of non-Euclidean geometries: the definition of the various geometries in predicate logic [**?**] permitted to restore the universality of mathematical truth, allowing to determine which axiom was used in which proof and which theorem held in which geometry.

In the same way, we defend the thesis that the interoperability of proof systems can only be achieved if we are able to express the theories implemented in these proof systems in a common logical framework.

In 1928, predicate logic, the first logical framework in the history of logic, was a huge success, since three important theories used ay that time: geometry, arithmetic and set theory have been expressed in it. But it also has limitations, explaining that another of the major theories used at that time: Russell's type theory (The Principia Mathematica) has not been expressed in it. Then, several other versions of type theory, Church's type theory [**?**], Martin-Löf's type Theory [**?**], and the Calculus of constructions [**?**], that are the theories implemented in most of the current proof systems, have also been defined as autonomous theories, and not in predicate logic.

This failure has led, in the field of proof systems, to abandon predicate logic, and even the concept of logical framework: the theories implemented in COQ, MATITA, HOL LIGHT, etc. are often defined as autonomous systems, and not in a logical framework.

However, a stream of research has attempted to understand the limitations of predicate logic and to propose other logical frameworks repairing them. The most prominent limitations of predicate logic are the lack of function symbols binding variables, the lack of a syntax for proof terms, the lack of a notion of computation, the lack of a notion of cut for axiomatic theories, and the impossibility to express constructive proofs. These limitations have led to the development of other logical frameworks: λ-Prolog [**?**, **?**], Isabelle [**?**], the $\lambda\Pi$-calculus [**?**], also called the "Edinburgh logical framework", Deduction modulo theory [**?**, **?**], Pure Type Systems [**?**, **?**], and ecumenical logics [**?**, **?**, **?**]. The $\lambda\Pi$-calculus modulo theory [**?**], implemented in the system DEDUKTI [**?**], is a synthesis of these frameworks. It not only allows the expression of geometry, arithmetic and set theory, but also that of Russell's type theory, Church's type theory, Martin-Löf's type theory, and the Calculus of constructions.

In the years 2010-2015, it was shown that theories implemented in HOL

LIGHT [**?**], MATITA [**?**], and FOCALIZE [**?**] could be expressed in DEDUKTI, and that the libraries of these systems could be translated to DEDUKTI, as well as the proofs produced with the automated theorem proving systems IPROVER [**?**] and ZENON [**?**], and by the SMT solver ARCHSAT [**?**]. This lead, in particular, to express, in DEDUKTI, ATELIER B proofs produced by ZENON.

Just like for the case of non-Euclidean geometries, it is not sufficient to express the theories implemented in MATITA, HOL LIGHT, etc. in a common logical framework and to translate the proofs originally defined in these systems to DEDUKTI, we must also analyze each proof to understand which feature of which theory it uses and in which theory it can be used, a domain traditionally called "reverse mathematics" [**?**, **?**, **?**].

In the years 2015-2020, we started to focus on the translation of proofs from one library to another [**?**, **?**]. This led us to propose an on-line system-independent encyclopedia of formal proofs LOGIPEDIA (`http://logipedia.science`) in which each proof is labeled with the axioms and computation rules it uses, which helps to determine the systems in which it can be used. In particular, we have showed that the arithmetic library of MATITA could be translated into five other, significantly different, systems: HOL LIGHT, ISABELLE, PVS, COQ, and LEAN, preserving the readability of the statements of the theorems.

These successes have convinced us that we are now ready to scale up and develop this encyclopedia, with the objective to include in twenty years all the formal proofs developed at that time, and in four years a significant part of it.

Concept The main notions articulated in this project are thus those of

- logical framework,
- theory,
- instrumentation,
- reverse mathematics,
- concept alignmenent,
- and structuration.

Indeed, suppose we start with a proof developed in a system X. We must first express, in the logical framework DEDUKTI, the theory $D[X]$ implemented in the

system X. Then, we must instrument the system X so that the proof can be exported from it, as a piece of data, expressed as a proof in $D[X]$ and included in LOGIPEDIA. Then, we can analyze this proof in order to understand which features of $D[X]$ it is uses and thus in which alternative theories it can be expressed. Finally, we must align its concepts with the definitions already pressent in LOGIPEDIA and decide where it fits in the general structure of the encyclopedia.

This required to include in the project computer scientists,gg logicians, and mathematicians who have developed knowldge on one theory or one system, others who have developed knowldge on logical frameworks, other who have developed knowldge on reverse mathematics and concept alignment.

Such an infrastructure is new in the European Strategy on Research Infrastructures. We can even say that the idea to structure a research effort around an infrastructure, such as *Software heritage*, is relatively new in computer science and mathematics.

Depending on the system X, we consider more of less research needs to be done, for instrance to desing the theory $D[X]$. Among the 15 systems we focus on, we plan to reach, for 10 of them, the level 5 or 6. For the others, more research needs to be done and we plan to reach level 3 only.

Methodology The methodology is the same for integrating any library to LOGIPEDIA, but due to a difference of readiness of the various systems we focus on, our priorities are different.

1. We already know to express most of the theories of ATELIER B, COQ, FOCALIZE, HOL LIGHT, HOL4, ISABELLE, MATITA, and RODIN in DEDUKTI. We propose to instrument these systems so that they can produce DEDUKTI proofs, that we can include in LOGIPEDIA.

2. For other theories, such as those of ABELLA, AGDA, LEAN, MINLOG, MIZAR, PVS, TLA+, the work is in progress, or not yet started. So we must first understand how they can be expressed in DEDUKTI.

3. Besides the standard libraries of these systems, large libraries have been developed: the Isabelle Archive for formal Proofs, Flyspeck, MathComp, CompCert... We want to including some of these libraries in LOGIPEDIA.

4. Besides proof systems, we also want to include proofs coming from automated theorem provers, SAT solvers, SMT solvers, and model checkers. So

we must instrument some of these systems so that they can produce DEDUKTI proofs, that we can include in LOGIPEDIA.

5. We want to develop algorithms to analyze which symbol, axiom, rewrite rule is used in each proof, and consequently in which system each proof can be used.

6. Each library imported in LOGIPEDIA will come with its own definition of natural numbers, real numbers, etc. We want to develop "concept alignments algorithms" to transport theorems from one structure to another isomorphic one.

7. Besides data, we propose to include in LOGIPEDIA, metadata and an inner structure.

Readiness of the project This idea of building such a standard for proofs has already been investigated in the past, such as in the Qed manifesto [**?**], but have produced limited results.

Our thesis is that, since the Qed project, the situation has radically changed, because after thirty years of research, we have an empirical evidence that most of the formal proofs developed in one of these systems can also be developed in another, we understand the relationship between the theories implemented in these systems much better, we have developed several logical frameworks, extending predicate logic, in which these theories can be expressed, and we have developed reverse mathematics algorithms to analyze which axioms and rules are used in each proofs and algorithms, such as constructivization algorithms, to translate proofs from one theory to another.

1.4 Ambition

This section must explain how the proposal addresses the specific ambitions (i), (ii), and (iii) on page 54 of the call, that are described in more details in appendix D. page 107 of the call.

1.4.1 Progress beyond current achievements

Networking activities This project will include networking activities, to foster a culture of cooperation between scientific communities, that are today often too

centered around one system and one theory. Instead, such a project will incent them to build this encyclopedia together and to always wonder if the proofs they develop are specific to one system or are universal. This will also foster a culture of cooperation between scientific communities and two communities of users: teachers and industrial partners. This is why the project includes two "clubs of users", one of teachers and one of companies using formal methods.

Besides these two clubs, the research directions will be discussed every year by the advisory board that will gather LOGIPEDIA contributors and users.

The development and maintenance of LOGIPEDIA will eventually lead to the discussion of standards for proof languages, even if such an effort is premature today.

We plan to organize a yearly workshop of LOGIPEDIA developers and users, continuing the effort started on the January 2019 meeting.

As explained before, the development of a system-independent and theory-independent encyclopedia of proofs will trigger new ways to teach formal proofs to new a audience.

Transnational access Our encyclopedia being on line, it will of course be accessible from every country in Europe and beyond.

Joint research activities The project includes two types of joint research activities. First, it will as any infrastructure, it will allow joint research projects between the users of this infrastructure that will be able to develop new proofs together using different systems.

Second, as any infrastructure, LOGIPEDIA raises new research problems. Some of them have already been solved in the past and require to be implemented jointly in a first version of the encyclopedia. Others are newer and will trigger new cooperation between the teams of the project.

1.4.2 Innovation potential

Formal methods are at the turning point. Several academic and industrial successes have proved the readiness of the technology, but this technology takes to much time to be generalized, except in the transportation industry.

Analyzing this phenomenon, it is clear that the redundancy of the efforts to develop proof systems and the lack of common theories, benchmarks, and standards for these systems is a limiting factor.

This effort to integrate the scientific and technological effort around formal proofs in Europe is a way to address this issue so that the economic spinoffs from the project benefits the European industry.

Chapter 2

Impact

2.1 Expected Impact

This section must explain how the proposal addresses the expected impact on page 56 of the call.

The proposal is in coherence with the goals of the call INFRAIA-02-2020: Integrating activities for starting communities (listed on page 56 of the part 4 of Horizon 2020 Work Programme 2018-2020). It also clearly refers to one of the mentioned cross-cutting activities: open science.

Wider and more efficient access — New research infrastructure In a shared public encyclopedia providing virtual access, each user, in research, industry, and eucation, can find the formal proofs she needs in the logic she wants, regardless the logic and system this proof has been developed in.

Synergy — Open research data. Instead of having a scattered community, each group developing a library for its own logic and its own system, researchers and engineers will be able to work together on common developments, reusing proofs developed in other systems and in other communities.

Partnership with industry Several European and non European companies are member of the project, as contributors or as members of the future LOGIPEDIA user's group.

Education — Closer interaction between a larger number of researchers. Education to formal methods in computer science and to formal proofs in mathematics always hits the same obstacle: the need to choose a specific theory or system, in contradiction with the claimed universality of logical truth. Education to formal methods and formal proofs will gain in universality once it will be demonstrated that this choice amounts to include, or not, a few axioms and reduction rules. We also defend that this renewal of logic education at university level and before is of prime importance in our "post-truth era".

Better management of the continuous flow of data. A shared encyclopedia allows a better sustainability of the formal proofs developed over time. Too many formal proofs developed in the past are not available any more.

As discussed above, the shift from informal, pencil and paper, proofs to formal computerized proof is a major improvement on the never ending quest for logical rigor, with a strong impact both on mathematics, where much more complex proofs can be built, and computer science, where safety and security can be dramatically improved with the use of formal methods. But this major step forward also has a negative side effect: we have moved from a time where we had (informal) proofs of Pythagoras' theorem or Fermat's little theorem, to a time where we have (formal) proofs in Coq, in Matita, in HOL Light, in PVS, etc. of these theorems, jeopardizing the universality of mathematical truth.

We see this loss of universality of mathematical truth as the main obstacle to the diffusion of the notion of formal proof, in the communities of mathematicians and computer scientists, but also engineers and students. Our long-term goal is to resurrect the universality of mathematical truth in order to build a strong formal proof community including specialists and non-specialists such as working mathematicians, engineers and students.

This requires to express the theories implemented in these systems in a common logical framework, each with a finite number of axioms and reduction rules, in order to be able to say, not that a proof is expressed in one system or in another, but to say which axioms and reduction rules it uses, as we have been used to since the development of non-Euclidean geometries.

Having a standard for expressing theories and proofs and resurrecting this way the universality of mathematical truth will also make proof systems interoperable and will allow the construction of an on-line system-independent encyclopedia. More importantly, this will suppress one of the main obstacles to the diffusion

of formal proofs in mathematics, computer science, industry, and education, just like the development of the html standard induced a renewal of document sharing in general and the definition of predicate logic induced a renewal of logic in the 1930's.

Innovation Formal methods are now an important part of some advanced industrial projects. For instance, mastering formal methods is key to give Europe a competitive advantage in conquering the market of autonomous cars, trains, planes, and drones. But this penetration of formal methods in industry hits the same obstacle that researchers often promote one method, theory or system, while their industrial partners are in search of universality. We expect to make formal proofs more accessible to industry by avoiding each project to redevelop elementary proofs, but instead benefit of the formalization work shared with other communities.

Key exploitable results. - Logipedia in itself from TRL 3 to TLR 4
- Representation of theories implemented in various systems
- Rechecking formal proofs for higher Evaluation Assurance Level

2.2 Measures to maximise impact

Chapter 3

Implementation

3.1 Work plan - Work packages, deliverables

Where should this sentence go?: The Logipedia kick-off meeting http://deducteam.gforge.inria.fr/seminars/190 in January 2019, brought together 38 researchers from Austria, Czech Republic, France, Italy, the Netherlands, and Poland. Since then, colleagues from Belgium, Germany, Serbia, Sweden, and the United Kingdom have manifested interest. Some of these researchers are ready to contribute to Logipedia, that currently contains a few hundred lemmas, aiming at having in twenty years all the formal proofs then developed, in a single encyclopedia.

Currently, we know how to express the theories of HOL LIGHT, MATITA, and FOCALIZE in DEDUKTI and recheck proofs developed in these systems. In the next five years, we plan to address the theories of ABELLA, AGDA, ATELIER B, COQ, HOL4, ISABELLE, LEAN, MINLOG, MIZAR, PVS, RODIN, and TLA+. Other systems, ACL2, IMPS, and NUPRL are kept for later, except if some other groups join the project.

We propose to introduce a metric to measure the progress of the integration of a library in Logipedia (Figure 3.1)

The various systems addressed in the project are currently at this level

Level 1: The theory implemented in the system X has been defined in the lambda-Pi-calculus modulo theory and in Dedukti.

Level 2: The system X has been instrumented to export of proof that can be checked in Dedukti.

Level 3: A significant part of the library of the system X has been exported and checked in Dedukti.

Level 4: A tool has been defined to analyze the Dedukti X proofs, detect those that can be expressed in a theory weaker than that of the system X and translate those proofs in a weaker logic.

Level 5: These proofs have been made available in Logipedia.

Level 6: All proofs of the system X have been exported, translated and made available in Logipedia.

Figure 3.1: The Logipedia readiness levels

Matita:	level 5
FoCaliZe:	level 3
HOL Light:	level 3
Coq:	level 3
Agda:	level 2
Atelier B:	level 2
Isabelle:	level 2
HOL4:	level 1
Minlog:	level 1
Rodin:	level 1
Lean:	level 0 to 1
PVS:	level 0 to 1
Abella:	level 0
Mizar:	level 0
TLA+:	level 0

3.1.1 Goals

Specific, Measurable, Achievable, Relevant, Time-Bound

Matita:	from level 5 to level 6
HOL Light:	from level 3 to level 5
FoCaliZe:	from level 3 to level 5
Coq:	from level 3 to level 5
Agda:	from level 2 to level 3
Atelier B:	from level 2 to level 5
Isabelle:	from level 2 to level 5
HOL4:	from level 1 to level 5
Minlog:	from level 1 to level 5
Rodin:	from level 1 to level 5
Lean:	from level 0 to 1 to level 3
PVS:	from level 0 to 1 to level 2
Abella:	from level 0 to level 2
Mizar:	from level 0 to level 5
TLA+:	from level 0 to level 2

Beyond our main focus on interactive systems, we also plan to integrate some proofs coming from automated theorem provers, SMT solvers, and model checkers, when these proofs have a reasonable size. We already have experience with Zenon, iProver, and Archsat, but we also plan to go in this direction, in cooperation with our colleagues working on LFSC [Stump09].

Finally, we must also structure this encyclopedia: some of the libraries we start with already have a structure (modules, qualified names, etc.) that it is important to preserve.

But, in addition, each library contains a definition of natural numbers, real numbers, etc. and, most importantly, logical connectors, that must be aligned. All these objectives contribute to building a new formal proof community, focused on the values of knowledge exchange and sustainability.

3.1.2 Work packages and tasks

Questions :

- merge WP1 and WP4. Pros: ATP/SMT produce proofs that are not fundamentally different from ITPs, cons: WP1 is already very big

- add a management WP

- add a dissemination WP

- Change the order : WP6 and WP7 before WP1.

WP 1: Instrument proof systems to produce Dedukti proofs

coordinators: Frédéric Blanqui and Jesper Cockx

David Deharbe, Tobias Nipkow, Guillaume Genestier, Jesper Cockx, Guillaume Burel, Filip Marić, Makarius Wenzel, Helmut Schwichtenberg, Nicolas Magaud, Gaspard Férey, Ulf Norell

We know how to express in Dedukti the theories implemented in Matita, HOL Light, FoCaliZe, Coq, Agda, Lean, Minlog, Isabelle, HOL4, Atelier B, and Rodin. The systems Matita, HOL Light, FoCaliZe, and Coq, already have been instrumented to export proofs that can be checked in Dedukti. Our first work package is to do the same thing for Coq, Agda, Lean, Minlog, Isabelle, HOL4, Atelier B, and Rodin. Three methods have to be used here: some of the systems (Automath style), such as Coq, Agda, Lean, and Minlog already have proof-terms that can be output, thus the main task is to translate these proofs into the Dedukti format. Others (LCF style), such as Isabelle and HOL4, have an inference kernel that can be instrumented, and often already has, the main task here is to transform the internal proof-object into an external proof-term. Others, such as Atelier B and Rodin are slightly more difficult to address. For those, we need to use the water ford method: extract an incomplete trace (a sequence of lemmas) and fill the gap using automated theorem proving, as experimented with Atelier B and Zenon.

Task 1.1: instrument Agda [Göteborg, Delft]

Agda is a popular dependently typed programming language / proof assistant based on Martin-Löf's intuitionistic type theory. Its theory is similar to Coq and Lean, but is more focused on interactive development and direct manipulation of proof terms (in contrast to using a tactic language to generate the proof terms). Agda has a sizable standard library (available at https://github.com/agda/agda-stdlib) that consists of both utilities for programming and mathematical proofs.

In the summer of 2019, Guillaume Genestier worked together with Jesper Cockx on the implementation of an experimental translator from Agda to Dedukti during a research visit at Chalmers University in Sweden. This translator is still work in progress, but it is already able to translate 142 modules of the Agda standard library to a form that can be checked in Dedukti. This exploratory work uncovered several challenges and opportunities for further work, which are outlined below.

(1) To support the construction of proof terms, Agda provides powerful features such as dependent pattern and copattern matching, eta equality for functions and record types, and definitional proof irrelevance. The first one – dependent pat-

tern matching – can be translated directly to rewrite rules in Dedukti. However, the two latter features – eta equality and irrelevance – rely on Agda's type-directed conversion algorithm, while Dedukti's conversion is untyped. Hence in order to translate Agda proofs to Dedukti these features need to be encoded.

One particular concern with the encoding of eta-equality is that in general it requires storing of additional type information in the proof terms. It can hence lead to a large blow-up in the size of those proof terms, and thus greatly increase the cost of typechecking. The same problem also occurs in other parts of Agda; for example constructors of parametrized datatypes do not store the values of the parameters, but they need to be reconstructed in the translation to Dedukti. We plan to investigate two possible approaches to this problem: either we can try to find a better encoding which reduces the size of the type annotation, or alternatively we can extend the Dedukti language with type-directed conversion rules to render the type annotations unneccessary.

(2) Another unique feature of Agda is the support for first-class universe level polymorphism. In particular, Agda has a built-in type of levels that has complex structure of (in)equality between levels. Compared to universe polymorphism in Coq, an additional challenge is that levels in Agda can contain arbitrary terms as subexpressions. Our plan is to define a sound and complete embedding of Agda's level type in Dedukti, based on the existing 1work on encoding AC (associative-commutative) theories. This would both serve as a stress test of how well Dedukti can handle complex equational theories, and improve our understanding of type theories with first-class universe level polymorphism, which would be useful for the implementation of Agda.

(3) In contrast to Coq and Lean, Agda does not have a well-defined core language to which proofs are elaborated. Instead, definitions are translated to an internal representation that is relatively close to the user input. This provides a challenge when translating Agda proofs to Dedukti: each feature in Agda's internal syntax needs to have its own translation. As part of this project, we will hence investigate possible designs for a core language for Agda. Having such a core language would have several benefits: it would deepen our understanding of the Agda language, it would increase the trustworthiness of Agda proofs, and it would make it much easier to export Agda terms to other languages (such as Dedukti in the context of this project).

(4) Agda provides an experimental option for extending the language with user-defined rewrite rules, which are very similar to the rewrite rules provided by Dedukti. Because of this similarity, we expect it to be straightforward to translate rewrite rules from Agda to Dedukti. However, by comparing the two implementa-

tions we hope to gain new insights and find opportunities for improvement on both sides. The interest of some of these features goes beyond just the Agda language. In particular, Lean also supports definitional proof irrelevance, as does Coq with the recent addition of the SProp universe. Hence we plan to collaborate with the teams working on those languages to improve the support for these features where there is overlap.

One research engineer at Chalmers, and one PhD student or postdoc at TU Delft.

Question: does this task belong to WP1 or WP2?

Task 1.2: instrument Lean [?]

Task 1.3: instrument Minlog [LMU München]

The Minlog system <http://minlog-system.de> implements a theory of computable functionals (TCF, cf. [SchwichtenbergWainer12], Ch.7).). It is a form of higher order arithmetic where partial functionals are first-class citizens.

The intended model of TCF is the Scott-Ershov model C of partial continuous functionals [Ershov77]. Computable functionals are defined by so-called computation rules, a form of (possibly non-terminating) defining equations understood as left-to-right conversion rules. An important example is the corecursion operator, which is needed to define functions operating for instance on streams of signed digits (a convenient format to represent real numbers). The logical framework allows to declare a proven equality as a rewrite rule. Now it is tempting to identify two terms or formulas when they have the same normal form w.r.t. rewriting (including of course beta-conversion); this is often called deduction modulo rewriting. However, in a setup like TCF where non-termination is allowed we cannot use normal forms, but we can consider two terms or formulas as identical when they have a common reduct. This drastically simplifies proofs involving real number arithmetic.

Another central feature of TCF (and hence the Minlog system) is that it internalizes a proof-theoretic realizability interpretation (in the form of Kreisel's so-called modified realizability, with realizers of higher type). More precisely, for every (co)inductive predicate we have another one with one argument more, denoting a realizer. It is important that this realizers is expressed in the term language of TCF (an extension of Gödel's system T). Since a realizer can be seen as a program representing the computational content of a constructive existence proof (expressing that a certain specification has a solution), we now can reason

about such programs in a formal way, inside TCF. In fact, given a proof M in TCF of a specification all x ex y A(x,y) we can extract a term (program) t_M and automatically generate a new proof of all x A(t_M(x)). In other words, for programs generated in this way from existence proofs, formal verification is automatic. Note that for a proof involving coinduction the extracted term contains the (non terminating) corecursion operator. Of course, for efficient evaluation in a second step we want to translate our extracted term into an efficient (functional) programming language like Haskell.

Other aspects of Minlog are more common. It is a proof system in Gentzen-style natural deduction and therefore based on proof terms (the so-called Curry-Howard correspondence). We distinguish between (co)inductive predicates with and without computational content; in fact, computational content only arises from (co)inductive predicates marked as computationally relevant (c.r.). In particular, both universal und existential quantifiers do not influence computational content: one needs to relativize them to c.r. predicates (like totality) to make them computationally relevant.

A central application area of Minlog is to formalize Bishop-style [Bishop67] constructive analysis and extract interesting algorithms from proofs. An example is the Intermediate Value Theorem treated in [LindstroemPalmgrenSegerbergStoltenberg08]. More recently we have extracted algorithms operating on (both signed digit and Gray-coded) stream-represented real numbers from proofs which never mention streams. They come in by relativizing real number quantifiers to appropriate coinductive predicates [Berger09].

The work planned to be done by the Minlog group is to extend this kind of work further into constructive analysis, e.g. Euler's existence proof of solutions of ODEs. We would like to learn from the experience of other proof assistant developers working on related matters, both conceptually and by sharing libraries. It would be very helpful if this can be done in a unified setting where different approaches can be compared.

Here we need to go a little futher and propose to express Minlog proofs in Dedukti.

One Postdoc and one Phd position

Task 1.4: instrument Isabelle [TU München]

Isabelle as a logical framework [**?**] is an intermediate between Type-Theory provers (like Coq or Agda) and classic LCF-style systems (like HOL Light or HOL4). The inference kernel can already output proofs as λ-terms on request, but

this has so far been only used for small examples [**?**]. The challenge is to make Isabelle proof terms work routinely for reasonably big entries from The Archive of Formal Proofs [**?**]. Preliminary work by Wenzel (2019) has demonstrated the feasibility for relatively small parts of Isabelle/HOL: some orders of magnitude in scalability are still missing.

This work package will revisit important aspects of the Isabelle/HOL logic implementation on top of the Isabelle/Pure framework, such as normalization of proofs, efficient type-class reasoning, special representation of derived rules and definition principles (datatypes, recursion, induction). The volume of proof term output may be reduced further, by taking more structure of the target language (Dedukti) into account and omitting certain low-level reasoning of HOL (e.g. for inductive types). [Subcontracted to Makarius Wenzel, Augsburg, Germany.]

The underlying Isabelle/ML implementation platform (on top of Poly/ML) will be revisited as well, to improve monitoring of memory usage, and to double the standard heap size from 16 GB to 32 GB (without suffering from the full overhead of the 64 bit addressing). [Subcontracted to David Matthews, Edinburgh, UK.]

Task 1.5: instrument HOL4 [Göteborg]

Task 1.6: instrument Atelier-B [Southhampton, Toulouse, Clearsy].

Task 1.7: instrument Rodin [Southhampton, Toulouse, Clearsy].

Task 1.8: integrate the translator to Matita in Matita itself and export the full Matita library [Bologna]

WP2: Defining theories in Dedukti

Coordinators: Cezary Kaliszyk and Stephan Merz

David Deharbe, Stephan Merz, Guillaume Genestier, Guillaume Burel, Yamine Ait Ameur, Jean-Paul Bodeveix, Mamoun Filali, Arthur Charguéraud, Gaspard Férey

For other theories, such as Abella, PVS, Mizar and TLA+, we have not yet investigated the possibility to express them in Dedukti.

Task 2.1: express the theory of PVS in Dedukti and instrument the system [Saclay]

Task 2.2: express the theory of Mizar in Dedukti and instrument the system [Innsbruck, Bialystok].

Task 2.3: express the theory of TLA+ in Dedukti and instrument the system [Nancy, Liège].

Task 2.4: express the theory of Abella in Dedukti and instrument the system [Saclay]

The usual approach to capturing either Peano and Heyting arithmetics is to use various axioms (and an axiom scheme for induction) on top of classical and intuitionistic first-order logic. Indeed, this is the approach used in the Dedukti proof checker.

A different approach to encoding arithmetic has been developed over the past 20-30 years, starting with papers by Schroeder-Heister and Girard in the early 1990s and extended in a series of papers by Baelde, Gacek, McDowell, M, Momigliano, Nadathur, and Tiu. In this new setting, first-order logic is extended by considering both equality and the least fixed point operator as *logical connectives*: these logical connectives are not available directly in Dedukti.

This new foundations for arithmetic has been implemented in two systems: the automated Bedwyr prover and the interactive Abella prover. While neither Bedwyr nor Abella are as popular as many of the theorem provers that are covered by this proposal, there are two important reasons to consider incorporating them into the Logipedia effort.

First, the Bedwyr prover is capable of constructing proofs for the kind of queries that are part of emphmodel checkers. This class of provers has not yet been incorporated into Dedukti. The proof-theoretic work behind model checking in Bedwyr should provide some of the insights needed for allowing Dedukti to proof check the results of model checkers.

Second, Bedwyr and Abella provide for direct and elegant support of meta-level reasoning. Given that the foundations for Bedwyr and Abella have been given using Gentzen's sequent calculus, it was possible to enrich their foundations to allow for the treatment of binding structures within terms. As a result, it is possible to reason directly on terms representing λ-terms and π-calculus expressions. In particular, the Abella prover has probably the most natural and compact

formal treatment of the π-calculus and its meta-theory when compared to all other attempts in any other theorem provers. More generally, the Abella prover should be able to treat the meta-theory of programming and specification languages as well as various logics and their proofs. While these tasks are not the typical tasks considered by the majority of theorem provers within the scope of this proposal, meta-theory results do play an important role at times: in fact, the ultimate questions as to whether or not a proof checkers (such as that used by Dedukti) is correct or not will involve meta-theoretic questions.

We propose to work on the general problem of exporting proofs from Abella to Dedukti. (Since all proofs that are constructed automatically via Bedwyr can also be constructed manually within Abella, we shall limit our discussion below to Abella only.) The proposed work will serve not only to answer the question of how to relate these two different foundations for arithmetic but also to allow Abella's particular style of proofs to find applications in the wider world of formalized proofs.

The general problem described above has the following constituent parts.

(1) Proofs involving searching finite structures. Proofs built for model checking problems over finite structures have two different kinds of phases. To illustrate, consider trying to find a specific node within a binary tree. If such a node exists, then the proof essentially encodes the path to the node in the tree. If, however, no such node exists, then the proof of that negative fact is essentially a computation that exhaustively explores the tree. Using the Dedukti terminology: in the first case, the proof involves several deduction steps, while in the second case, the proof involves a pure computation. When dealing with model checking problems such as simulation (in concurrency theory) and winning strategies (in game theory), proofs will involve alternating phases involving either deduction or computation. Since the notion of computation in Abella-style proofs involves backtracking search, that style computation will be quite different from Dedukti's notion of computation as confluent rewriting.

(2) Extending model checking problems to the general case of infinite structures and the associated inductive reasoning methods. Although the formal basis of Abella uses least and greatest fixed-point combinators and explicit (co-)invariants, the Abella implementation of (co-)induction is based on cyclic reasoning using size-annotated relations. It is known, in principle, how to convert cyclic proofs using annotations to proofs with explicit invariants, but an invariant extraction procedure that works in all cases is still missing. Once such invariants are available, incorporating them into Dedukti should be straightforward in association with part (1).

(3) Binding structures. Abella, as well as several other computational logic systems (λProlog, Isabelle/Pure, Twelf, Beluga, etc) make use of the so-called λ-*tree syntax* (a form of *higher-order abstract syntax*, HOAS) approach to represent bindings. This approach is further enriched in Abella with the ∇-quantifier that allows inductive and co-inductive properties to be defined based on the *structure* of λ-terms. We propose to examine encodings of $lambda$-tree syntax in Dedukti. The best approach probably involves extending the underlying theory of Dedukti with a quantifier similar to Abella's ∇-quantifier.

(4) Reflective treatment of unification. One of the features of Abella's style of proofs is the use of left-introduction rules for equality that exhaustively examine complete sets of unifiers for λ-terms. This is implemented in terms of a unification engine that is currently a trusted black box, which complicates any proposal for exporting proofs to different implementations of unification or equality. In Dedukti the unification procedure can be recast as a rewrite system, but it is unclear how to derive reflective properties based on the unifiability of terms.

Task 2.5: expressing HoTT [Saclay, Leeds]

WP3 : Libraries

Coordinators: Georges Gonthier and Tobias Nipkow

David Deharbe, Nicola Gambino, Tobias Nipkow, Makarius Wenzel, Julien Narboux, Gaspard Férey, François Thiré

Translating the standard libraries of the systems is part of the WP1.

Task 3.1: MathComp [Sophia, Saclay, Paris]

Task 3.2: the Mizar library [Innsbruck, Bialystok]

Task 3.3: Isabelle Archive of Formal Proofs [TU München, Saclay]

Task 3.4: the GeoCoq library [Sophia, Strasbourg, Belgrade]

Task 3.5: the Flyspeck library [?]

Task 3.6: the NASA PVS library [?]

Task 3.7: the seL4 library [?]

Task 3.8: the CompCert library [?]

WP4: ATP, SAT, SMT, Model checkers

Coordinators: Pascal Fontaine and Chantal Keller

David Deharbe, Cezary Kaliszyk, Pascal Fontaine, Dale Miller, Stephan Merz, Josef Urban, Martin Suda, Guillaume Burel, Filip Marić, Chantal Keller, Julien Narboux, Thibault Gauthier

[Nancy, Liège]

The importance of proofs in automated theorem provers, satisfiability modulo theories solvers, propositional satisfiability solvers and model checkers is increasingly recognized. While for the propositional case, the community agrees on a well defined proof format, the situation is not clear for the other kind of automated reasoners. There is no clear format for SMT, and the TSTP format for automated theorem provers fixes a syntactic template for proofs rather than providing an unambiguous framework to express proofs semantically.

Some preliminary works predating this proposal clearly establish that Dedukti can accommodate proofs in Satisfiability Modulo Theories, automated theorem provers, and SMT. In this work package, we will build on those preliminary work and provide a set of conduits from the established formats used in automated tools. For the tools that do not have yet an established format, we will make a selection of tools (Zipperposition and E for automated theorem provers, CVC4 and veriT for SMT, ??? for model checking) and provide a conduits for those tools. These conduits and the techniques used in the embedded translation will be properly documented, to ease integration of further tools of the kind. If a standardized proof format appears for some kind of tools, the conduits will be updated to adopt the new standard.

In this work package, we also plan to integrate in Logipedia some well-chosen proofs coming from automated tools. Well-chosen proofs will have to be representative of typical applications of the tools, and be of reasonable size. They will serve as examples to the community, to illustrate the potentials of Dedukti and Logipedia.

Create the infrastructure to enable the long term goal: be able to split a large proof obligation into smaller parts and distribute to the appropriate automatic engines, that would all produce proofs, glued together in a single large proof for the original proof obligation.

Automatic Tools Exporting Proofs

Logipedia as a Source of Challenges for Automatic Reasoners

–> Translation to TPTP, SMT-LIB, DIMACS

A language for Communication between Automatic Reasoners

WP5 : Reverse mathematics

coordinators : Nicola Gambino and Julien Narboux

Nicola Gambino, Michael Rathjen, Guillaume Genestier, Julien Narboux, François Thiré

[Saclay, Leeds]

Task 5.42: Ecumenical Dedukti [Grienenberger, Dowek]

We plan to define in DEDUKTI both constructive and classical connectives and quantifiers following [**?**, **?**, **?**], so that both constructive and classical proofs can be expressed in DEDUKTI.

We plan to develop constructivization algorithms to transform proofs expressed in this theory, into its constructive fragment.

Task 5.43: A universal type theory [Grienenberger, Dowek]

A type theory that contains both a dependent and non dependent arrow contains two fragments that correspond to Simple type theory and to the Calculus of constructions. Such a theory can express proofs developed in HOL Light, Isabelle, HOL4, Coq, Matita...

We plan to develop algorithms to transform proofs expressed in this theory, into its Simple type theory fragment.

WP6: Concept alignment

Coordinators: Thibault Gauthier and Dale Miller

Florian Rabe, Cezary Kaliszyk, Dale Miller, Josef Urban, Filip Marić, Yamine Ait Ameur, Jean-Paul Bodeveix, Mamoun Filali, Chanta Keller, Julien Narboux, Nicola Magaud, Arthur Charguéraud, François Thiré

Construct tools and proofs to analyze these proofs and align concepts, that is unify concepts such as connectives and quantifiers, the concept of natural number,

etc. and theorems that occur in several libraries. [Paris, Saclay, Innsbruck, Prague, Strasbourg, Belgrade]

WP7: Structuration of the theories

Coordinators: Florian Rabe and Burkhart Wolff

David Deharbe, Nicola Gambino, Florian Rabe, Michael Rathjen, Dale Miller [Erlangen, Saclay]

1. concrete/surface syntaxes

1. Central Library Backend Systems

2. Cross-System Front-Ends/Portals (Logipedia, ...)

3. Semantic Middleware-based System Interoperability

Erlangen: two postdocs over the course of the project

Since proof-objects for substantial theory developments tend to be very large (the representation of current POs for the Isabelle/AFP can easily reach several TB although using techniques for compression), A technical pre-requisite for interchangeability, connectivity and advanced search consists in a structured, typed format for meta-data together with a flexible mechanism of their validation. Technically, this kind of meta-data has the form of a function annoconst : arg1 -> ... -> argn -> proof-term -> proof-term where annoconst is a constant symbol which represents an identity in the proof-term (so, any import function of a specific system can actually ignore it), and where the argi represent terms with meta-information such as, eg., “this proof-term represents a free data-type construction of the form ...”, or “this part of the proof is a derivation of a free data-type of the following form ...”, “this lifting over assumptions represents in Isabelle a Locale-instantiation”, “this part of a theory development is connected to ... ”, “this theorem belongs to the sub-class of XXX ... theorems”, etcpp. For arguments of annotations, validation-functions can be defined that may check that the argument terms satisfy a certain property wrt. to the proof-term and the current logical context. Dedukti will provide a framework that allows for each proof-system (Coq, HOL4, Isabelle...) to declare meta-data together with validations and thus communicate tool-specific knowledge to other systems. This framework can be seen as a particular form of an ontology definition language.

WP8: Indexing and browsing [?] Construct tools to index and browse this encyclopedia, that is find the theorem one needs, either by looking for it with its name, with its statement, or with symbols occurring in it.

WP8: Dissemination, communication, and exploitation

WP9: Management

Deliverables

Milestones

3.2 Management structure, milestones and procedures

Is it the right place to speak about the assessment / self-evaluation of the project

The construction of such a system-independent online encyclopedia of formal proofs requires a cooperation between teams that have experience in 1. the development of proof systems, 2. interoperability between proof systems (point-to-point and larger scale), including concept alignment, that is the identification of isomorphic structures and notions, such as Cauchy and Dedekind real numbers, 3. the development and use of logical frameworks, 4. the development of large libraries of proofs.

We have identified 19 groups in 10 European countries ready to contribute to this effort : Belgrade, Bialystok, Bologna, Cambridge, Clearsy, Erlangen, Göteborg, Innsbruck, Liège, TU München, LMU München, Nancy, Paris, Prague, Saclay, Sophia-Antipolis, Strasbourg and Toulouse.

These include researchers working on the development of proof systems (such as Agda, Coq, Isabelle, and Mizar) researchers working on interoperability (for instance, between Hol Light and Coq or between proof systems and automated theorem provers: hammers), researchers working on concept alignment (in particular using machine learning to detect similarities), researchers working on databases of theorems (such as the MMT library), researchers working on the development of Dedukti and Logipedia, researchers working on large libraries (such as the Isabelle Archive of Formal Proofs, MathComp, and GeoCoq).

We also are in contact with non-European colleagues at SRI, Nasa Langley Research Center, Amazon WS, the University of Iowa, and Data61 working on HOL Light, PVS, LFSC, and seL4.

The proposal will focus on

1. Hiring doctoral students, post-doctoral researchers, and engineers to solve the above mentioned problems.

2. Organizing a yearly meeting to present the state of the art of the development of the encyclopedia.

3. Organizing smaller meetings where four to eight researchers work on a specific theory or library.

4. Building a permanent advisory board where industrial partners, and international academic partners (including non-European ones) will discuss the future of the encyclopedia. This board will include among others June Andronick (Data61, Kensington NSW), Denis Cousineau (Mitsubishi Electric), Natarajan Shankar (SRI), Aaron Stump (Iowa), Laurent Voisin (Systerel).

3.2.1 Risks and opportunities technical, organizational, external, management

1. Risks

- more difficult that expected (probability: low, severity medium). Mitigation: at least some systems

- too many specific proofs (probability: low (empirical evidence in informal maths), severity medium). Mitigation: do not translate these proofs and start understanding why they need strong axioms, but translate the rest (basic maths).

- too high complexity (time and memory), difficulty to scale up (probability medium, severity medium). Mitigation: lower the objectives (basic maths, use more powerful machines, wait for Moore's law to help you).

- one partner leaves (probability low) or does not deliver (probability medium). Mitigation: downsize the project.

- difficulty to find people (doctoral students, post-docs) in some countries. Mitigation use the size of the network to find more peoples in others.

- Logipedia splits into several libraries: face the risk (we have avoided to have a classical and a constructive logipedia, a predicative one and a non-predicative one, the diversity of theories expressed in logipedia permits to make the probability very low).

- a beautiful encyclopedia, but nobody cares (probability low). Mitigation: improve the interface, the communication, make it more completed

2. Opportunities

- More people want to join: model checking, sat solvers...

- math teachers want to use it for teaching

3.3 Consortium as a whole

3.4 Resources to be committed

Chapter 4

Members of the consortium

4.1 Participants (applicants)

Univerzitet u Beogradu (Belgrade): Predrag Janičić, Filip Marić, Vesna Marinković, Danijela Simić, Sana Stojanović-Đurđević.

Uniwersytet w Białymstoku (Bialystok): Karol Pąk.

Alma Mater Studiorum – Università di Bologna: Claudio Sacerdoti.

Clearsy: David Deharbe, Thierry Lecomte, Ronan Saillard.

Delft: Jesper Cockx.

Friedrich-Alexander-Universität Erlangen-Nürnberg: Michael Kohlhase, Dennis Müller, Florian Rabe.

Göteborg University - Chalmers Tekniska Högskola: Andreas Abel, Magnus Myreen, Ulf Norell.

Universität Innsbruck: Joshua Chen, Cezary Kaliszyk, Miroslav Olšák, Stanisław Purgał.

Leeds: Nicola Gambino, Michael Rathjen, Paul Shafer.

Université de Liège: Pascal Fontaine.

Technische Universität München: Tobias Nipkow, Makarius Wenzel (subcontractor)

LMU München: Helmut Schwichtenberg. Kenji Miyamoto

Inria Nancy – Grand Est: Stephan Merz.

Inria Paris: Émilio Gallego, Hugo Herbelin, Théo Zimmerman.

České vysoké učení technické v Praze (Prague): Thibault Gauthier, Martin Suda, Josef Urban.

Inria Saclay – Île de France: Bruno Barras, Frédéric Blanqui, Valentin Blot,

Guillaume Burel, Kaustuv Chaudhuri, Gilles Dowek, Catherine Dubois, Georges Gonthier, Olivier Hermant, Jean-Pierre Jouannaud, Chantal Keller, Dale Miller, Pierre-Yves Strub, Burkhart Wolff.

Inria Sophia Antipolis - Méditerranée: Yves Bertot, Pierre Boutry, Cyril Cohen.

University of Southampton: Michael Butler, Thai Son Hoang, Andrew Sogokon.

Université de Strasbourg: Arthur Charguéraud, Nicolas Magaud, Julien Narboux, Pascal Schreck.

National Polytechnique Institute of Toulouse: Yamine Ait Ameur, Jean-Paul Bodeveix, Mamoun Filali.

4.2 Third parties involved in the project (including use of third party resources)

Chapter 5

Ethics and Security

5.1 Ethics

5.2 Security