Integrating and opening research infrastructures of European interest

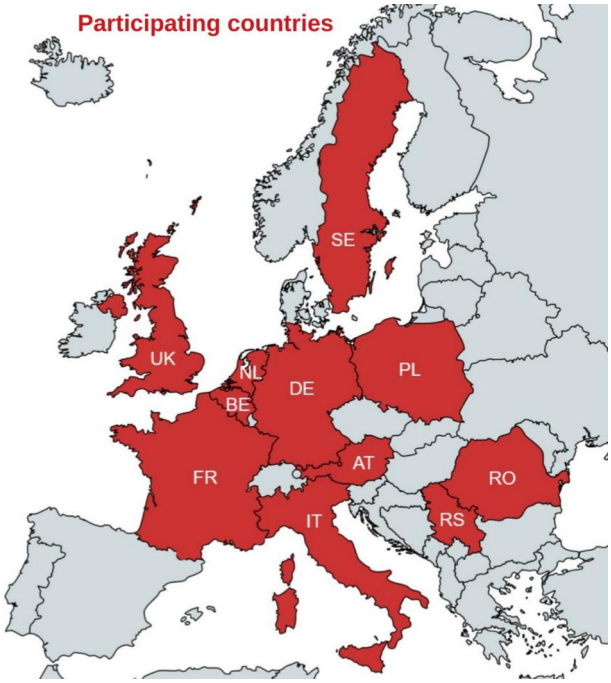
H2020-INFRAIA-2018-2020

Logipedia

Coordinator: Gilles Dowek (gilles.dowek@inria.fr)

#	Partner organisation name	Short name	Country	Type
1	Institut National de Recherche en Informatique et Automatique	Inria	FR	●
2	Université de Strasbourg	Unistra	FR	●
3	Institut National Polytechnique de Toulouse	INPT	FR	●
4	Universität Innsbruck	UIBK	AT	●
5	Université de Liège	ULiege	BE	●
6	Alma Mater Studiorum – Università di Bologna	Unibo	IT	●
7	Faculty of Mathematics, University of Belgrade	UBel	RS	●
8	Technische Universität München	TUM	DE	●
9	Technische Universiteit Delft	TUDelft	NL	●
10	Université Paris-Saclay	USaclay	FR	●
11	Friedrich-Alexander Universität Erlangen-Nürnberg	FAU	DE	●
12	University of Leeds	ULeeds	UK	●
13	Göteborgs Universitet	UGot	SE	●
14	Chalmers Tekniska Högskola	Chalmers	SE	●
15	Ludwig-Maximilians-Universität München	LMU	DE	●
16	Institut Mines-Télécom	IMT	FR	●
17	Uniwersytet w Białymstoku	UBia	PL	●
18	ClearSy	ClearSy	FR	⋆
19	OCamlPro	OCamlPro	FR	⋆
20	University of Birmingham	UoB	UK	●
21	Commissariat à l'Energie Atomique et aux Energies Alternatives	CEA	FR	⋆
22	Duale Hochschule Baden-Württemberg	DHBW	DE	●
23	Edukera	Edukera	FR	⋆
24	MED-EL Elektromedizinische Geraete GmbH	MED-EL	AT	⋆
25	Prove & Run	P&R	FR	⋆
26	Konrad-Zuse-Zentrum für Informationstechnik Berlin	ZIB	DE	●
27	Universitatea Alexandru Ioan Cuza din Iasi	UAIC	RO	●
28	Runtime Verification SRL	RV	RO	⋆

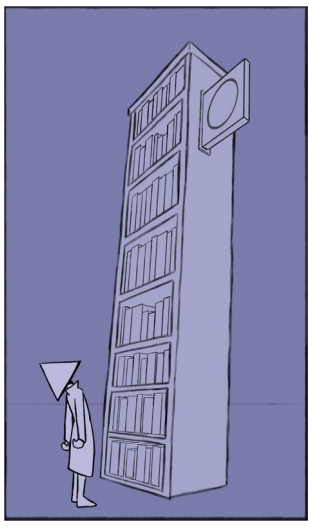
● Academic partner ⋆ Industrial partner



The club of industrial users[1]

[1]The letters of intent are at the end of the document (Section 6).

Contents



Section 1

Excellence

Digital infrastructures are the backbone of our societies, as we can observe in times of a crisis: teleworking, telemedecine, distance learning, etc. are the basis of their resilience. In such situations, as well as in more common ones, the solidity of our societies depends on the reliability of these infrastructures. A single bug in a critical system like a transportation system (autonomous car, subway, train, plane, etc.), a health care system (robotic surgery, ear implant, telemedecine, etc.), a nuclear plant, a financial application, an e-governance application, etc. can cause people to die and enterprises to go bankrupt.

If testing software makes it possible to reveal the presence of bugs, only formal verification can guarantee their absence. Thus, the trust in critical systems relies on formal verification, in particular computerised formal proofs, that guarantee the safety of people using them. This crucial role of computerised formal proof is highlighted by several successes, like the correctness proofs of the automatic Paris metro line 14[1], the detect-and-avoid system for unmanned aircraft system developed by NASA[2], the operating system seL4[3], and the C compiler CompCert[4]. It has been empirically observed that operating systems and compilers often have bugs, but proved systems and compilers, such as seL4 or CompCert, have none, or much fewer. This is why, at the highest Evaluation Assurance Levels (EAL) of the Common Criteria (CC) security evaluation international standard, in effect since 1999, certification processes require not only testing, but also computerised formal proofs.

Several companies in our consortium from sectors such as transportation, health care, and energy have expressed interest in using formal methods or scaling up their existing use of them. These companies are at the forefront of the adaptation of formal methods in the industry, and if they are successful many others will likely follow. Making formal methods more accessible to these companies and all other interested partners, via better theories, better tools and better-training of workers, would bring a significant competitive advantage to European companies and Europe as a whole.

Aside from their use in software verification, proofs have also always been the basis of mathematics, and hence of many mathematised sciences. In this area, the use of computerised formal proofs has made significant progress, as witnessed for example by the formalisation of the Feit-Thompson theorem[5]. Such a use of formal proofs in mathematics and in sciences is important, as the correctness of a pencil and paper proof is often the object of a controversy, as shown by the case of Hales' theorem (Kepler's conjecture) which is too long and too complex to be verified without a computer (and which has now

[1] P. Behm, P. Desforges, and J.-M. Meynadier. "MÉTÉOR : An Industrial Success in Formal Development". In: *B'98: Recent Advances in the Development and Use of the B Method, Second International B Conference*. Vol. 1393. LNCS. 1998

T. Lecomte et al. "Applying a Formal Method in Industry: A 25-Year Trajectory". In: *Formal Methods: Foundations and Applications - 20th Brazilian Symposium, SBMF 2017*. Vol. 10623. LNCS. 2017.

[2] C. Muñoz et al. "Unmanned aircraft systems in the national airspace system: a formal methods perspective". In: *SIGLOG News* 3.3 (2016), pp. 67–76.

[3] G. Klein et al. "seL4: formal verification of an OS kernel". In: *Proceedings of the 22nd ACM Symposium on Operating Systems Principles 2009, SOSP 2009*. 2009.

[4] X. Leroy. "Formal certification of a compiler back-end or: programming a compiler with a proof assistant". In: *Proceedings of the 33rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2006*. 2006.

[5] G. Gonthier et al. "A Machine-Checked Proof of the Odd Order Theorem". In: *Interactive Theorem Proving - 4th International Conference, ITP 2013*. Vol. 7998. LNCS. 2013.

been formally verified[6]) or with the more recent controversy on the *abc* conjecture[7]. Thus, formal proofs have a wide range of applications, from safety and security of software to mathematics.

Whether they are used for software verification or mathematics, formal proofs are developed using research infrastructures called "proof systems". These proof systems are pieces of software that allow computer scientists, mathematicians, engineers, and logicians to build and study formal proofs, just like particle accelerators allow physicists to build and study particles. Figure 1 lists the most prominent proof systems that are used today, both those developed in Europe and those from outside Europe.

Major proof systems

In Europe				Outside Europe	
Abella	Agda ⋆	Atelier B ⋆	Coq ⋆	Acl2	HOL Light ⋆
FoCaliZe ⋆	HOL4 ⋆	Isabelle ⋆	KProver ⋆	IMPS	Lean
Matita ⋆	Minlog ⋆	Mizar ⋆	ProB	LFSC ⋆	NuPRL
ProvenTools ⋆	Rodin ⋆	TLA$^+$ ⋆	Why3 ⋆	PVS ⋆	TSTP ⋆

Figure 1: Most of today's proof systems are developed at different places in Europe. Those addressed in the project are marked with a ⋆.

★ ★ ★

A lot of formal proofs developed for one critical system could be used in another. Unfortunately, the development of formal methods is slowed down by the large number of proof systems (and sometimes the large number of versions, over time, of one single system) and the lack of a common theory used by these systems. For instance, the Paris 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, the seL4 operating system has been proved correct in Isabelle/HOL[8], and the compiler CompCert has been proved correct in Coq. Some projects, such as the proof of Hales' theorem, have been started in different systems and required significant integration efforts for obtaining the overall result.

Because of the focus on this large variety of proof systems, the field is fragmented into a number of small communities, each of which is centred around one theory and one proof system. This proposal brings together partners from a large number of these different communities for the first time, and unite them in a new community centred around the development of a shared infrastructure for formal proofs. In fact, the enthusiasm for this proposal was so large that the number of formal participants had to be limited; partners that could not formally be part of the proposal joined the various clubs that we describe later. Figure 1 marks all the proof systems represented in the project with a ⋆. This clearly demonstrates the potential for this new community to further expand during and after the course of this project.

The fragmented nature of the current proof systems is a major bottleneck for the adaptation of formal systems by the industry. Several of the companies in our consortium have expressed interest in the use of formal methods, but are faced with the difficult task of choosing a proof system to work with. In general, a library developed in one system cannot be used in another, so a wrong choice may lead to a critical library being unavailable in that system. This forces the company to either implement the library again themselves or switch to a different proof system. Thus, interoperability (the possibility for one user to use a proof developed in another system), sustainability (the possibility to use a proof decades after it has

[6] T. Hales et al. "A formal proof of the Kepler conjecture". In: *Forum of Mathematics, Pi* 5 (2017), e2.

[7] D. Castelvecchi. "Mathematical proof that rocked number theory will be published". In: *Nature* 580.177 (2020). DOI: 10.1038/d41586-020-00998-2.

[8] L. C. Paulson. "Isabelle: The Next 700 Theorem Provers". In: *Logic and Computer Science*. Academic Press, 1990, pp. 361–386.

been developed), and cross-verification (the possibility, for higher assurance of its correctness, to verify a proof in a system different from the one in which it has been constructed) are restricted.

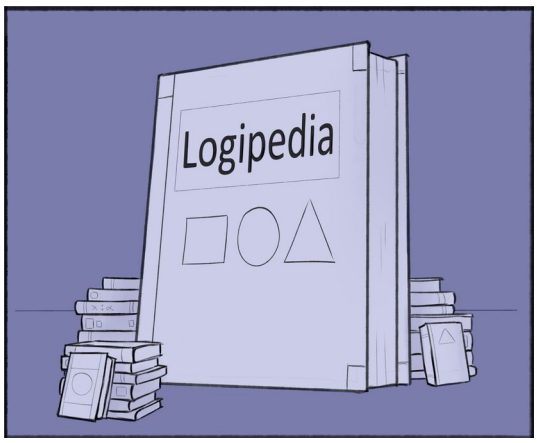
The fragmentation of these infrastructures also limits the dissemination of formal proofs in non-specialist communities. For instance, teaching formal proving to undergraduate students in a logic course is difficult, as it requires the choice of a specific language, a specific theory, and a specific system that restrict the scope of the course, instead of giving students the tools that are useful everywhere.

An important goal of this project is thus to provide free and virtual access to these critical libraries from all proof systems, which will greatly increase the adoption of formal verification in both industry and education.

★★★

Making proof systems interoperable would avoid duplication of work, reduce development time, enable cross-verification, and make formal proofs accessible to a much larger community.

After three decades dedicated to the development of these systems, enabling cooperation between systems is the next step in the development of the formal proof technology.



Artwork by Nadia Miller.

To bring together the fragmented field of formal methods and to provide everyone – no matter what proof system they use – with free and open access to libraries of formal proofs, we propose to develop an encyclopedia that will eventually contain all formal proofs in existence.

This encyclopedia will be called Logipedia.

Building such an encyclopedia will allow interoperability, sustainability, and cross-verification of all formal proofs in the encyclopedia.

Logipedia is thus a research infrastructure that integrates proof systems, through the sharing of data (formal proofs).

- Logipedia will bring together the communities centred around different proof systems, providing new opportunities for networking between these previously separate communities.
- Logipedia will provide a vast library of proofs developed in different proof systems and make them available to the public, greatly increasing the trans-national and virtual access to these valuable resources.
- Logipedia will allow researchers from different areas to share and discuss their work in a common language, opening up new avenues for joint research activities.

★★★

Such an infrastructure is, in many ways, new in the European Strategy on Research Infrastructures. In fact, the idea to structure a networking activity around the construction and the use of a large scale infrastructure is itself relatively new in computer science and mathematics, even though some efforts have been made in this direction with the *OpenDreamKit* and *Software Heritage*, for example.

So, we also aim at contributing to an evolution of the organisation of research in the computer science and mathematics in Europe.

History of the project

Convinced that a cloud of formal proofs could bring to the applications of formal proof technology the same boost that the cloud has brought to computing, and also that managing a large encyclopedia required some interdisciplinary effort, we developed a proof of concept containing a few hundreds lemmas expressed in the language of six systems and organised, in January 2019, a meeting to discuss the future of this project. This meeting brought together 38 researchers from Austria, the Czech Republic, France, Italy, the Netherlands, and Poland.

During this meeting, the idea of making this proof of concept a European infrastructure emerged. Since then, colleagues from Belgium, Germany, Romania, Serbia, Sweden, and the United Kingdom, from academia and industry, have expressed their interest in participating in this effort. These researchers and engineers are ready to contribute to develop this encyclopedia, aiming at sharing proofs, under a creative commons licence, making them searchable, accessible, interoperable, and reusable.

How do proofs contribute to safety and security of software?

Imagine the following casino game. At the beginning, a player is given eleven euros. At each round, she throws a six-sided dice. If the result is a six, then the game ends. If it is a five, a four, a three, or a two, she is given twice the amount of money she already has. If it is a one she loses two euros, if she has at least two. When the game ends, the player wins the money she got, except if she has zero, in which case she loses one million euros.

This game can be modelled by the following programme:

```
n = 11
stay = True;
while stay:
    roll = random.randint(1,6)
    if roll == 6:
        stay = False
    elif roll >= 2:
        n = n + 2 * n
    elif (n >= 2):
        n = n - 2
print(n)
```

To be on the safe side, the player wants to be sure, before starting playing, that she will never finish with zero. And indeed, at all times the content of the variable `n` is an odd number, thus it cannot be zero. **This property "nothing bad happens" is called the "safety" of this program.** This property is a consequence of two simple theorems of arithmetic:

$$\forall x\ (odd(x) \Rightarrow odd(x + 2 * x))$$

$$\forall x\ (odd(x) \Rightarrow odd(x - 2))$$

meaning that, for all x, if x is an odd number, then $x + 2 * x$ and $x - 2$ are odd numbers too. Hence, proving the safety of this programme amounts to prove these two theorems.

Let's now consider the case where there is a tiny bug in the programme. For instance the `2` has been replaced by a `3` in the instruction `n = n + 2 * n`. The programme is then unsafe as shown by the sequence $11, 9, 7, 5, 3, 1, 4, 2, 0$. Yet, testing this program will, most likely, not reveal this bug, since it only manifests very rarely. In contrast, attempting to prove the correctness of this program will reveal the bug as it is impossible to prove the proposition

$$\forall x\ (\mathit{odd}(x) \Rightarrow \mathit{odd}(x + 3 * x))$$

What is a formal proof? What is a proof system?

Since Antiquity, we have known that proofs, both purely mathematical ones, as in Euclid's elements or the recent proof of the Kepler's conjecture by Thomas Hales, and proofs used to establish the safety and security of software, can be built with a limited number of rules, for example

- From $A \Rightarrow B$ ("A implies B") and A, deduce B.
- From A, deduce $A \vee B$ ("A or B").
- etc.

Yet for most of mathematical history, proofs have been written in a pidgin of natural language and mathematical formulas. When proofs are very long (as it is often the case for the proofs used in safety and security, but also for some proofs in pure mathematics), mistakes are very difficult to detect. For instance, dozens of wrong proofs of the parallel postulate have been given through history, sometimes by the best of mathematicians such as Ptolemy, Proclus, al-Haytam, Tacket, Clairaut, Legendre, etc.

In the 1960s, Robin Milner and Nicolaas de Bruijn noticed that the correctness of a mathematical proof could be checked by a computer. This led to the development of the two first proof systems in history: Milner's LCF and de Bruijn's Automath.

For instance, from the axioms

$$\forall x\ (\mathit{philosopher}(x) \Rightarrow \mathit{human}(x))$$

$$\forall x\ (\mathit{human}(x) \Rightarrow \mathit{mortal}(x))$$

we can deduce

$$\forall x\ (\mathit{philosopher}(x) \Rightarrow \mathit{mortal}(x))$$

In the language implemented in Automath, this proof is written

$$\lambda x \lambda h\ (g\ x\ (f\ x\ h))$$

What is a theory?

Deduction rules such as "From $A \Rightarrow B$ and A, conclude B" are universal, but building proofs requires more rules, that are often specific to a domain of knowledge and are called "axioms". Examples are the axioms of geometry, the axioms of arithmetic, etc. These axioms constitute a theory.

At the beginning of the 20^{th} century, an axiomatic theory, *set theory*, has been proposed to express all mathematical proofs. In the first half of the 20^{th} century a few variants of set theory

have been proposed, as well as a few alternatives (such as *Simple Type Theory*). But because these theories had not been designed for being implemented on a computer, each proof system such as Coq, Isabelle/HOL, Mizar, Atelier B, etc. has implemented its own theory. **Thus, the rise of computer-checked formal proofs has led to a multiplication of alternative theories for mathematics.**

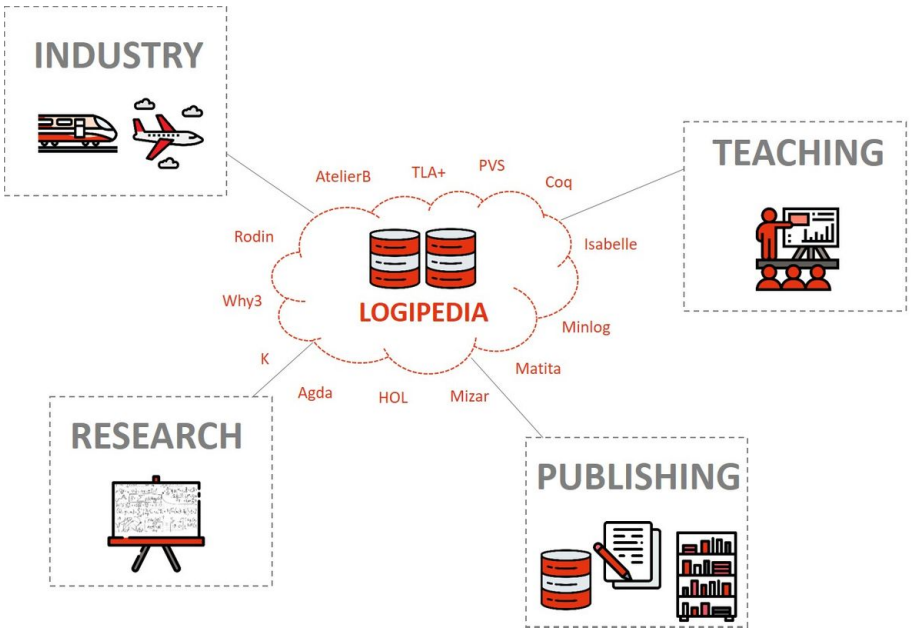
This is the major obstacle to interoperability between proof systems.

1.1 Objectives

To foster the interoperability of proof systems and the sustainability and the cross-verification of formal proofs, we propose to collect them in an online encyclopedia, called Logipedia. For each proof, Logipedia will indicate in which systems it can be used and, it will provide a version of this proof in the theory of these systems.

Such a project will not only foster the use of formal proofs in research in mathematics and computer science, but also in industry, by allowing cross-verification, sustainability, and interoperability of formal proofs, and in education, freeing the teaching of formal proof technology from being bound to one system.

As a majority of proof systems are developed in Europe, there is a unique opportunity for Europe to take the lead on such a project and prepare the grounds for the economic spinoffs from the project benefiting the European industry. That is why the consortium gathers most of the European actors active on formal proof systems, while also developing links with non-European scientists.



When defining our objectives, we need indicators to measure our performance. We can use the notion of a "Technology readiness level" (TRL) to assess the status of some of our objectives. But, to measure the level of integration in Logipedia of an existing proof system and of its associated proof libraries, we had to introduce another metric, that has been discussed with all the members of the consortium: the *Logipedia Integration Level* (LIL).

The Logipedia Integration Levels (LIL)

LIL	Description
0	No effort of integration has been made yet.
1	The theory implemented in the system has been defined in the logical framework Dedukti, the core language of Logipedia.
2	The system has been instrumented so that examples of proofs can be exported and checked in Dedukti.
3	25% of the library of the system has been exported and checked in Dedukti.
4	25% of the library of the system has been made available in Logipedia.
5	A tool has been built to analyse the Dedukti proofs translated from the system, detect those that can be expressed in a theory weaker than that of the system, and translate those proofs into a weaker theory.
6	All proofs of the system have been exported, translated, and made available in Logipedia.

The ultimate goal of Logipedia is to have all the formal proofs available to mankind in a single encyclopedia. A first proof of concept (available at http://logipedia.science) contains a few hundred lemmas, from the Matita library, expressed in the theory of six different systems: Matita, Coq, Lean, HOL Light, Isabelle/HOL, and PVS.

In the next four years, we plan to address the libraries developed in the proof systems Agda, Atelier B, Coq, FoCaLiZe, HOL Light, HOL4, Isabelle/HOL, K Prover, Matita, Minlog, Mizar, ProvenTools, PVS, Rodin, and TLA$^+$, as well as three formats used in automatic theorem proving: TSTP, LFSC, and Why3.

The above systems can be roughly divided into three groups.

- For those that currently have an integration level LIL 1 or higher (Matita, HOL Light, FoCaLize, Coq, Isabelle/HOL, Agda, Atelier B, Rodin, and HOL4) we have preliminary results, and we plan to bring them to a high level of integration.
- For those that have a current integration level of 0 (Mizar, TLA$^+$, PVS, Minlog, ProvenTools, and K Prover) we are only starting out and our goals within this project are generally less ambitious.
- Finally, TSTP, LFSC, and Why3 are formats that permit to exchange information with Automatic Theorem Provers (ATP).

These three groups of systems correspond to different objectives, but all three are key to the project. The first ones will constitute Logipedia in four years, the second prepare the long-term future of the infrastructure, and the third extend the frontier of Logipedia beyond proofs produced by humans.

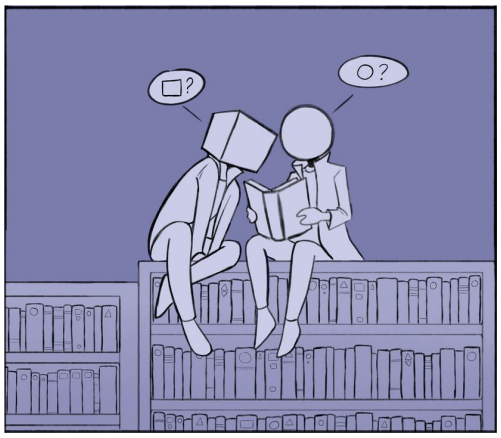
Very large proofs, requiring large libraries of auxiliary results, have been developed in some proof systems. These large libraries constitute an ambitious benchmark for the methods developed, and they contribute to populate Logipedia with a large number of elements. We decided to focus on five libraries, chosen for their relevance, coverage, and maturity: the Isabelle Archive of Formal Proofs (AFP), the Isabelle probability and analysis library, the Coq geometry library, the Flyspeck library, and the CakeML library. The targeted integration level for all these libraries is LIL 4. Other important libraries are left for the future.

The integration of systems of the first group, the integration of automated theorem proving systems, and the integration of these large libraries will provide the content of Logipedia, and collecting all these formal proofs in a single infrastructure is *per se* a networking activity.

If we now turn to the access to these formal proofs, our objective is to develop an infrastructure that makes them freely accessible through a web browser. So, by construction, the access will be trans-national and virtual. But beyond these two objectives of a trans-national and virtual access, most of

the effort will be made to make this encyclopedia accessible to a large community of specialists and non-specialists: researchers, engineers, teachers, students, etc. This requires us to develop an ergonomic web interface, a package distribution system, and a search engine for mathematical formulas. To make this infrastructure accessible, we also need to structure its content into libraries, books, chapters, etc., reaping the benefit of the structure (modules, qualified names, etc.) of some of the libraries we start with and to enrich the data with meta-data. For these two objectives the Logipedia integration levels are not relevant, and we use the common Technology Readiness Level (TRL) indicator.

As noted before, for some proof systems, the implemented theory has already been analysed and related to other systems, so that we can immediately start integrating proofs from these systems into Logipedia. For other systems (Mizar, TLA$^+$, PVS, Minlog, ProvenTools, and K Prover) as well as proposed theories without a mature system (such as homotopy type theory), more research work is needed. Understanding how the theories implemented in these systems can be expressed in Logipedia is one of our joint research objectives.



Finally, to export formal proofs to systems different from that in which they have been developed, we need to analyse which axioms they use and, when it is possible, transform them so that they do not use such an axiom any longer.

In addition, each system has its own definitions of various mathematical objects and structures that must be aligned: structural results proved for one definition of real numbers, for instance, will be transported to any isomorphic structure, regardless of the way it has been defined. This is another objective of this project. Its performance indicator is the number of proofs brought from level LIL 4 to level LIL 5.

Objective	**Activity to achieve this objective**	**Performance indicator**		
Integration of proof systems	Integrate the librairies of Matita, HOL Light, FoCaLiZe, Coq, Isabelle/HOL, Agda, Atelier B, Rodin, and HOL4.	System	current	targeted
		Matita	LIL 5	LIL 6
		HOL Light	LIL 3	LIL 5
		FoCaLiZe	LIL 3	LIL 5
		Coq	LIL 3	LIL 5
		Isabelle/HOL	LIL 2	LIL 5
		Agda	LIL 2	LIL 4
		Atelier B	LIL 1	LIL 5
		Rodin	LIL 1	LIL 3
		HOL4	LIL 1	LIL 5
Integration of automated theorem proving	Integrate proofs coming from automated theorem provers and SMT solvers.	Format	current	targeted
		TSTP	LIL 1	LIL 4
		LFSC	LIL 0	LIL 4
		Why3	LIL 0	LIL 4
Integration of big libraries	Integrate the Isabelle Archive of formal proofs, the Isabelle probability and analysis library, the Coq geometry library, the Flyspeck library, and the CakeML library.	Library	current	targeted
		AFP	LIL 2	LIL 4
		Probability / analysis	LIL 2	LIL 4
		GeoCoq	LIL 2	LIL 4
		Flyspeck	LIL 2	LIL 4
		CakeML	LIL 2	LIL 4

<table>
<tr><td>Development of the infrastructure</td><td>Make Logipedia freely accessible through a web browser and a package distribution system. Develop a search engine for mathematical formulas.</td><td colspan="3">TRL 4, as our target is beyond a proof of concept validated in an academic and industrial environment.</td></tr>
<tr><td>Structuring the encyclopedia</td><td>Structure the content of Logipedia into libraries, books, chapters, etc., reaping the benefit of the structure (modules, qualified names, etc.) of some of the libraries, and enrich the data with meta-data.</td><td colspan="3">TRL 4, as our target is beyond a proof of concept validated in an academic and industrial environment.</td></tr>
<tr><td>New theories, new systems</td><td>Express the theories of Mizar, TLA$^+$, PVS, Minlog, ProvenTools, K Prover, and Homotopy Type Theory in Dedukti.</td><td>System</td><td>current</td><td>targeted</td></tr>
<tr><td></td><td></td><td>Mizar</td><td>LIL 0</td><td>LIL 4</td></tr>
<tr><td></td><td></td><td>TLA$^+$</td><td>LIL 0</td><td>LIL 2</td></tr>
<tr><td></td><td></td><td>PVS</td><td>LIL 0</td><td>LIL 2</td></tr>
<tr><td></td><td></td><td>Minlog</td><td>LIL 0</td><td>LIL 4</td></tr>
<tr><td></td><td></td><td>ProvenTools</td><td>LIL 0</td><td>LIL 3</td></tr>
<tr><td></td><td></td><td>K Prover</td><td>LIL 0</td><td>LIL 3</td></tr>
<tr><td></td><td></td><td>Homotopy Type Theory</td><td>LIL 0</td><td>LIL 3</td></tr>
<tr><td>Proof engineering</td><td>Develop algorithms to analyse which axioms are used in a proof, eliminate them, and align concepts.</td><td colspan="3">Bringing 25% of the encyclopedia to LIL 5, that is making the proofs of a significant part of the library available in systems different from those in which they have been developed.</td></tr>
</table>

The seven objectives give a first idea of the shape of the project. They all contribute to build a new formal proof community, focused on the values of knowledge sharing, safety, security, privacy, open access, and education.

1.2 Relation to the work programme

Access to the best research e-infrastructures	Proof systems and automated theorem provers are research infrastructures. Each proof system comes with its own library, and these libraries are also part of the research infrastructure. Currently these infrastructures are small, distributed and disconnected. Logipedia aims at co-building a large, central European infrastructure from smaller ones by an integration effort. This integration effort is substantial but doable. It contributes to the challenge of bringing to European researchers and engineers effective and convenient access to the best research infrastructures, in order to foster further advances in knowledge and technology. This idea to structure a networking activity around the construction and the use of an infrastructure is relatively new in computer science and mathematics. The novelty is that Logipedia is not an infrastructure made of computers, but an infrastructure made of data and of algorithms manipulating these data.
A starting community	The objectives of Logipedia have never been supported under FP7 or Horizon 2020 calls.

	It mobilises 28 research groups (21 in academia and 7 in industry), which is almost all the groups working on formal proof technology in Europe. The success of the project requires such a large network. Indeed, according to Metcalfe's law, the effect of a network is proportional to the square of the number of connected users. We can postulate that the effect of an infrastructure, such as Logipedia, is proportional to the product of the number of proofs it contains and the number of its potential users, each of them being proportional to the number of theories it supports. So, such a project needs a critical mass to succeed. This is also why we have developed an ecosystem surrounding the project, constituted of four clubs of users: a club of industrial users, a club of academic users, a club of users in education, and a club of users in publishing. These clubs, already in construction, will grow during the project. These are the foundations of the Logipedia community on which the infrastructure will expand. These groups are located in 11 European countries. Some of these countries have a large number of participants, some others fewer, reflecting the diversity of maturity of the research on formal methods in Europe. This project will contribute to develop the formal proof culture in the European countries where it is still inceptive. We should also point out the originality of this project within the European strategy of Research infrastructure. ► A novelty of this project is that it investigates how infrastructures can be used in mathematics and in computer science. ► It investigates how immaterial infrastructure can be used to structure a research community, just like material ones do. ► It is focused on mathematical *a priori* knowledge, while most infrastructures are focused on *a posteriori* knowledge, issued from measures, observations, experiments, and field surveys. ► It also investigates how a common infrastructure articulates the relation between research and industry.
Networking activities, trans-national access, joint research activities	The development of Logipedia requires the creation of a new community that integrates the fragmented existing communities around each proof system. Three work packages described in Part 3 are dedicated to networking activities to bring together these communities and collect the data from each of their respective systems: the formal proofs. Logipedia will be publicly and freely accessible online through any web browser and trough a package management tool, so trans-national and virtual access are directly provided. Its administration will be decentralised in various places in Europe with two mirror sites in Saclay and in München. But, beyond trans-national and virtual access, access is at the centre of our project, with two dedicated work packages. Also, this project fosters new joint research activities: between the members of the consortium, between the members of the consortium and other academic partners (in particular those developing automated theorem proving systems), between the consortium and the industrial users of proof systems, between the industrial users themselves, as using proofs developed by others will foster joint developments. Joint research activities currently exist in Europe to some extent. This project is however a unique opportunity to strengthen the existing ones and create new ones. Two more work packages are dedicated to these research activities.

Towards standardisation	This project will be a stepping stone for a possible standardisation of proof languages, for instance with the World Wide Web Consortium or the International Organization for Standardization. Unlike other communities in computer science, the formal proof community is somewhat sceptical with respect to standardisation. And indeed, it would not make sense to standardise a theory for proof systems, just like it would make no sense to propose Euclidean geometry as a standard all geometers should use. Instead, such a cooperative effort can lead to the standardisation of a logical framework, that is a language to express theories and proofs in any of these theories. By promoting cooperation between the communities of different proof systems and demonstrating that many different theories can coexist in a single encyclopedia, we may thus change the attitude towards standardisation in the formal proof community and make a standardisation process more widely accepted.
Inter-disciplinarity	The development of Logipedia requires mathematicians, logicians and computer scientists. Therefore, it is clearly inter-disciplinary. Yet, Logipedia is inter-disciplinary in an even more fundamental way. Proving properties of a piece of software driving a car or piloting an aircraft require formalising part of the physical world in which this piece of software evolves. In the same way proving properties of simulation software, requires to formalise some properties of the simulated object. Thus, just like mathematics and software are involved in any part of modern science, formal proof will eventually spread, over time, in all areas of science. One major obstacle to the formalisation of, for example, simulations is that they typically depend on large bodies of knowledge in mathematics and physics. Different proof systems may be better suited for different applications, thus there is a natural tendency for a diversity of proof formats, just like there is a natural tendency for diversity in the use of programming languages. But, when it comes to formalising a particular physical simulation, one needs to have all the necessary knowledge accessible from a single infrastructure. Because interdisciplinarity is required to formalise physical world simulations, a unifying language, such as Logipedia, for exchanging formal proofs between proof systems is a must-have.

1.3 Concept and methodology

(a) Concept

In Section 1.1, we have defined seven objectives: three focused on the integration of different systems and communities around them, two on the development and structuring of the encyclopedia itself to make it easily accessible for all interested parties, and two focused on exploring new possibilities for joint research created by the existence of Logipedia. We now detail the key scientific and technological concepts to achieve these objectives. Before that, we detail the transverse notion of logical framework.

Logical Frameworks

The multiplication of systems and theories jeopardises the universality of logical truth. But it is not the first time in history this happened. For instance, in the 19^{th} century, the universality of logical truth had been challenged, in a similar way, by the non-Euclidean geometries, as some statements could be true in one geometry, but false in others. At the beginning of the 20^{th} century, a solution to this

problem was found. The definition of the various geometries in predicate logic[9] allowed mathematicians to understand which axiom was used in which proof, restoring the universality of the truth of judgements of the form "the proposition B is true under the axioms $A_1, \ldots, A_n$".

Predicate logic is not a theory *per se*, but a framework in which one can express theories, as sets of axioms: a "logical framework". As we shall see, expressing the various theories implemented in Coq, Matita, Isabelle/HOL, PVS, etc. in a common logical framework will permit, in a similar way, to analyse which "axiom" is used in which proof, and hence, in which theories this proof can be expressed, and in which systems it can be used.

In 1928, predicate logic was a huge success, since three important theories used at that time (geometry, arithmetic, and set theory) could be expressed in it. But it also has limitations, which is the reason that another of the major theories used at that time (Russell's type theory, from *The Principia Mathematica*) has not been expressed in it. Since then, several other theories, such as Church's Simple Type Theory[10], Martin-Löf's Type Theory[11], and the Calculus of Constructions[12], have also been defined as autonomous theories, and not in predicate logic.

However, a different line of research has attempted to understand the limitations of predicate logic and to propose more powerful logical frameworks. The most prominent limitations of predicate logic are the lack of function symbols binding variables, the lack of a syntax for proof-terms (that is a syntax for proofs that goes beyond a mere derivation tree), the lack of a notion of computation, the lack of a notion of proof reduction for axiomatic theories, and the impossibility to express constructive proofs. These limitations have led to the development of other logical frameworks such as λ-Prolog[13], Isabelle[14], the $\lambda\Pi$-calculus (also called the "Edinburgh logical framework")[15], Deduction Modulo Theory[16], Pure Type Systems[17], and Ecumenical logics[18].

[9] D. Hilbert and W. Ackermann. *Grundzüge der theoretischen Logik*. Springer-Verlag, 1928.

[10] A. Church. "A Formulation of the Simple Theory of Types". In: *The Journal of Symbolic Logic* 5.2 (1940), pp. 56–68.

[11] P. Martin-Löf. *Intuitionistic type theory*. Vol. 1. Studies in proof theory. Bibliopolis, 1984. ISBN: 978-88-7088-228-5.

[12] T. Coquand and G. Huet. "The calculus of constructions". In: *Information and Computation* 76.2 (1988), pp. 95–120.

[13] D. Miller and G. Nadathur. *Programming with Higher-Order Logic*. Cambridge University Press, 2012. ISBN: 978-0-521-87940-8.

[14] L. C. Paulson and T. Nipkow. *Isabelle Tutorial and User's Manual*. Tech. rep. 189. Computer Laboratory, University of Cambridge, 1990.

[15] R. Harper, F. Honsell, and G. Plotkin. "A framework for defining logics". In: *Journal of the ACM* 40.1 (1993), pp. 143–184.

[16] G. Dowek, T. Hardin, and C. Kirchner. "Theorem Proving Modulo". In: *J. Autom. Reasoning* 31.1 (2003), pp. 33–72

G. Dowek and B. Werner. "Proof normalization modulo". In: *J. Symb. Log.* 68.4 (2003), pp. 1289–1316.

[17] S. Berardi. "Towards a mathematical analysis of the Coquand-Huet Calculus of Constructions and the other systems in Barendregt's cube". 1988

J. Terlouw. "Een nadere bewijstheoretische analyse van GSTT's". 1989.

[18] D. Prawitz. "Classical versus intuitionistic logic". In: *Why is this a Proof?, Festschrift for Luiz Carlos Pereira.* College Publications, 2015

G. Dowek. "On the definition of the classical connectives and quantifiers". In: *Why is this a Proof?, Festschrift for Luiz Carlos Pereira.* College Publications, 2015

L.C. Pereira and R.O. Rodriguez. "Normalization, Soundness and Completeness for the Propositional Fragment of Prawitz'Ecumenical System". In: *Revista Portuguesa de Filosofia* 73.3-4 (2017), pp. 1153–1168.

Church's type theory expressed in Dedukti

$$
\begin{array}{rcl}
type & : & Type \\
\eta & : & type \to Type \\
o & : & type \\
nat & : & type \\
arrow & : & type \to type \to type \\
\varepsilon & : & (\eta\ o) \to Type \\
\Rightarrow & : & (\eta\ o) \to (\eta\ o) \to (\eta\ o) \\
\forall & : & \Pi a : type\ (((\eta\ a) \to (\eta\ o)) \to (\eta\ o))
\end{array}
$$

$$
\begin{array}{rcl}
(\eta\ (arrow\ x\ y)) & \longrightarrow & (\eta\ x) \to (\eta\ y) \\
(\varepsilon\ (\Rightarrow\ x\ y)) & \longrightarrow & (\varepsilon\ x) \to (\varepsilon\ y) \\
(\varepsilon\ (\forall\ x\ y)) & \longrightarrow & \Pi z : (\eta\ x)\ (\varepsilon\ (y\ z))
\end{array}
$$

Figure 2: Church's type theory can be expressed in Dedukti, with 11 "axioms": 8 declarations and 3 computation rules.

All these logical frameworks have been unified in the $\lambda\Pi$-calculus modulo theory[19], implemented in the system Dedukti[20] on which Logipedia is based.

Geometry, arithmetic and set theory, but also Russell's type theory, Church's type theory, Martin-Löf's type theory, and the Calculus of constructions can be expressed in this framework. For instance, Church's type theory can be expressed in Dedukti with 11 "axioms": 8 declarations and 3 computation rules, as given in Figure 2.

So the theories implemented in various systems can all be expressed in Dedukti and the proofs developed in these systems can be translated to Dedukti. Just like in the case of non-Euclidean geometry, this allows the symbols and computation rules used in each proof to be analysed[21] (a domain traditionally called "reverse mathematics"[22]) and to deduce in which systems each proof can be used. This analysis is the basis of the interoperability between proof systems.

So, to make a formal proof, developed in some system X, accessible to the users of other systems, the first step is to express the theory $D[X]$, implemented in the system X, in the logical framework Dedukti. Then, we must instrument of the system X, that is, modify the system X, in such a way that it exports proofs in $D[X]$. Next, we need to analyse this proof in order to determine which symbols, axioms and computation rules of $D[X]$ it actually uses and, thus, in which alternative theories it can be expressed. Finally, we must align its concepts with the definitions already present in Logipedia and decide where it fits in the general structure of the encyclopedia.

Integration of proof systems

Consider a system X at Logipedia Integration Level (LIL) 1 so that the basic theory $D[X]$ has already been expressed in Dedukti. The next step is to implement a method to automatically translate proofs from system X to its expression in Dedukti, and make these proofs available in Logipedia, so they can be exported to other systems. This step is called the "instrumentation" of system X.

[19] D. Cousineau and G. Dowek. "Embedding Pure Type Systems in the Lambda-Pi-Calculus Modulo". In: *Typed Lambda Calculi and Applications, 8th International Conference, TLCA 2007*. Vol. 4583. LNCS. 2007.

[20] A. Assaf et al. "Dedukti: a Logical Framework based on the lambda-Pi-Calculus Modulo Theory". 2016.

[21] F. Thiré. "Sharing a Library between Proof Assistants: Reaching out to the HOL Family". In: *Proceedings of the 13th International Workshop on Logical Frameworks and Meta-Languages: Theory and Practice, LFMTP@FSCD 2018*. Vol. 274. EPTCS. 2018

G. Dowek. "Analyzing Individual Proofs as the Basis of Interoperability between Proof Systems". In: *Proceedings of the Fifth Workshop on Proof eXchange for Theorem Proving, PxTP 2017*. Vol. 262. EPTCS. 2017.

[22] H. Friedman. "Some systems of second-order arithmetic and their use". In: *Proceedings of the International Congress of Mathematicians (1974)*. Vol. 1. 1975

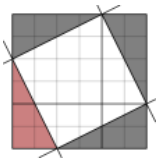
S. Simpson. *Subsystems of second-order arithmetic*. Cambridge University Press, 2009.

We consider three broad classes of proof systems: those based on dependent type theory, those based on simple type theory, and those based on set theory and predicate logic.

Systems based on dependent type theory (Agda, Coq, and Matita)

Coq is an interactive theorem prover developed at Inria since 1984. It is based on the Calculus of Constructions and was used to formally verify the correctness of both industrially relevant software such as the CompCert C compiler and complex mathematical proofs such as those of the Four Colour theorem and the Feit-Thompson theorem. In 2013, Coq received the ACM System Award.

Matita is an interactive theorem prover developed at the University of Bologna and used for teaching logic courses and to verify software and mathematical proofs, with special attention to predicative foundations. The first generation of the system (up to version 0.5.9) was born as a by-product of the MoWGLI FET-Open Project, it was compatible with the logic of Coq and it could re-use its libraries. It was an important test-bench for the integration of Mathematical Knowledge Management techniques with Interactive Theorem Proving, featuring for example a library of theorems distributed over multiple servers, innovative indexing and search techniques and automatic translation of proofs between declarative and procedural styles. The second generation of the system (up to the current version 0.99.3) was a re-implementation from scratch that departed from the logic of Coq and that experimented with the most concise ways to implement an efficient theorem prover. Several ideas later migrated into Coq. The currently available largest library is the formal certification of a complexity-preserving and cost-model-inducing compiler from C to MCS-51 machine code, developed in the FET project CerCo (Certified Complexity).



Agda is a dependently typed programming language and interactive proof assistant developed at Chalmers University of Technology as well as other places in Europe. The theory of Agda is similar to Coq and Matita, but it is more focused on direct manipulation of proof terms (in contrast to using a tactic language to generate proof terms). To support the construction of proof terms, Agda provides powerful features such as dependent pattern and co-pattern matching, η-equality for functions and record types, first-class universe polymorphism, and definitional proof irrelevance. In addition, Agda provides an experimental option for extending the language with user-defined computation rules, which are very similar to the computation rules provided by Dedukti.

Systems based on simple type theory (HOL4, HOL Light, and Isabelle/HOL)

HOL4 is home to a few medium to large scale specifications and associated proof developments that have value outside of HOL4. These specifications include the formal semantics of the CakeML language (and its certified compiler) and an extensive specification of the ARM instruction set architecture (ISA) as formalised by Anthony Fox at the University of Cambridge.

HOL Light is an implementation of Simple Type Theory, used at Intel and Amazon Web Services, among other places.

Isabelle/HOL is an implementation of Simple Type Theory in the Isabelle logical framework. It sits in between type-theory proof systems (like Coq or Agda) and classic LCF-style systems (like HOL Light or HOL4).

Systems based on set theory and predicate logic (Atelier B, Rodin)

Atelier B and Rodin are platforms to develop models written using the B method, a version of set theory expressed in predicate logic. A model in these systems expresses a state machine constrained by invariant properties. Verification of the correctness of the model requires discharging a number of proof obligations produced by a weakest precondition calculus. For example, spanning tree algorithms, distributed

algorithms, and access control policies have been formalised respectively in the Event
B and classical B methods. The development process for both versions of the B method is based on formal proofs: proof obligations are automatically generated and must be proven by automatic or interactive provers. This can include external solvers such as SMT solvers.

Despite the differences between these three classes of proof systems, there are multiple common points between them where solutions that work for one system can also be applied to other systems. Working together on implementing these tools will allow the researchers from these different communities to learn more about the similarities and differences between these systems.

Defining a basic instrumentation of some of these systems is straightforward, but building tools that also work well in practice raises new research challenges:

(a) Improving encodings. Even when we have understood how to express the basic theory of these systems, implementation often reveals some advanced features of these systems that require further work. Moreover, even when we know how to express all features of some system *in theory*, depending on the choice of expression, the instrumentation may be simpler or harder in practice. For example:

- Many Agda libraries (as well as some Coq libraries) heavily rely on type-directed conversion rules such as η-equality for functions and record types, or definitional proof irrelevance. These rules have to be expressed by adding type information to the proof terms. This can in turn lead to a large increase in the size of those proof terms, and thus greatly increase the cost of type checking. This blow-up has to be tamed by the choice of a clever expression of the theory.
- Coq, Agda, and Matita all provide support for coinductive (infinite) structures, which can be expressed by computation rules in Dedukti, but this expression must be carefully chosen so that the proof checker does not go into an infinite loop.
- A few of the systems rely on AC rewriting (that is, rewriting modulo associativity and commutativity of certain operations), for example to express universe polymorphism in dependent type theories. While Dedukti has support for AC rewriting, the current implementation is very slow, which makes typechecking the proofs in Dedukti infeasible. Thus, expressing these theories efficiently requires improvements in Dedukti itself.

Finding better ways to express such features could have a great impact on the quality and speed of the translation process. We plan to investigate two possible approaches to improve the expression of common features of proof assistants in Dedukti. First, we will investigate whether some information in the current expressions is redundant and can thus be omitted. Second, if this is not possible, we will investigate what minimal changes need to be made to Dedukti itself in order to overcome these limitations.

(b) Reconstructing transient proof components. In all proof systems, part of the information created during the proof checking process is not stored in the final proof term. In systems such as Coq, Agda, and Matita, this concerns for example the types of each subexpression. In other systems such as HOL4, Isabelle, and Atelier B, even more information about the proof is discarded and only the high-level proof steps remain. However, the translation from the system to Dedukti might rely on this transient information. Developing a method to reconstruct this transient information is thus crucial.

In order to instrument systems with transient proof information, the proof terms need to be complemented with this transient information by either logging or re-synthesising it on demand. Both approaches may be used, depending on the trade-off between computation time and space for storage.

However, this approach is insufficient for systems that rely on external provers such as Isabelle or Atelier B. In order to handle these cases, we will instrument these external provers so they can produce the missing parts of the proof terms.

(c) Producing more compact proofs. When translating a proof from some external system to Dedukti, we want to produce proof terms that are as small as possible, so it is easy to store them, recheck

them, or export them to a different system. However, there are at least two reasons why proof terms might become large. First, a typical proof in an external system may be large (because big parts of it are constructed automatically). Second, translating a proof to Dedukti may increase the size of the proof, for example by normalising it or by adding type annotations. The translation of the proofs can take a long time, the obtained proof may be difficult to use. This also has an obvious impact on the exporting of large libraries to Dedukti, discussed below.

To reduce the size of proof terms produced by the translation to Dedukti, we plan to investigate how to avoid unnecessary normalisation or duplication of (parts of) proofs. We will also investigate what parts of each proof can be safely omitted because they can be inferred from the rest of the proof.

For each of the systems considered in this project, we can identify some preliminary work. We plan to build upon the previous work done for the following systems:

- The standard and arithmetic libraries of Matita have been the first libraries to be exported to Logipedia using Krajono, a fork of Matita. We plan to make this translation a part of the code of Matita itself so that it is maintained with the rest of the system.
- The standard libraries of HOL Light and of FoCaLiZe already have been translated to Dedukti.
- CoqInE is a prototype tool that can translate Coq proofs to Dedukti. Recent work to include advanced features of Coq, such as universe polymorphism, has dramatically increased the coverage of this translation. We plan to make this translation a part of the code of Coq itself so that it is maintained with the rest of the system.
- 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 (about 25%) to a form that can be checked in Dedukti. This exploratory work uncovered several challenges and opportunities for further work (see research challenges above).
- The instrumentation of the inference kernel of Isabelle has already been implemented to output proofs as λ-terms that can be understood by Dedukti. However, this has so far been only used for small examples[23]. The challenge is to make Isabelle proof terms work robustly for the basic libraries and reasonably big applications. Preliminary work by Wenzel (2019) has demonstrated the feasibility for relatively small parts of Isabelle/HOL, but this requires scaling up.
- HOL4 has support for exporting proofs to the OpenTheory proof exchange format, and there has been some work on importing OpenTheory proofs into Dedukti.
- In the context of the BWare project, an expression of the set theory of the B method has been provided as a theory modulo, that is computation rules rather than a set of axioms. This expression is used by the automatic prover Zenon modulo which features a back-end to Dedukti. Thus, as a first step towards the instrumentation of Atelier B, proof obligations coming from Atelier B can be proved by Zenon modulo producing Dedukti proofs, hence providing a better confidence in the proofs produced by Atelier B[24].

The success of this integration project is measured by the LIL of the various considered systems at the end of the project.

Automatic theorem provers

We have previously discussed the instrumentation of *interactive* proof systems such as Coq, Isabelle/HOL, Agda, etc., where the users build formal proofs with the help of the machine. Automatic theorem provers are another class of systems, where the machine builds proofs without any human intervention. These systems are called automatic theorem provers, SAT solvers, SMT solvers, etc. The

[23] S. Berghofer and T. Nipkow. "Proof terms for simply typed higher order logic". In: *Theorem Proving in Higher Order Logics: TPHOLs 2000*. Vol. 1869. 2000.

[24] G. Bury et al. "An Automation-Friendly Set Theory for the B Method". In: *Abstract State Machines, Alloy, B, TLA, VDM, and Z - 6th International Conference, ABZ 2018*. Vol. 10817. LNCS. 2018.

proofs built by these systems are often expressed in simpler theories than those developed using interactive proof systems, but they are not of a different nature. They cover various domains and various kinds of applications, for example combinatorial mathematics[25], where they are expected to solve one large propositional problem, or verification of programmes[26], where they are given thousands of small problems in a combination of quantified theories.

Including proofs built by such automatic systems in Logipedia is both a goal *per se*, and a tool to help to integrate proofs developed in interactive proof systems and make a coherent whole out of Logipedia. A fruitful interaction between formal proofs requires low-level glue which falls in the scope of automatic theorem provers. For instance, they will be employed to fill the holes that appear when considering provers with various granularity, to reduce the gaps between proof systems, and to discharge proofs of concept alignment.

On the other hand, automatic theorem provers will also benefit from Logipedia. Obviously, Logipedia will be an extensive library of formal statements that can be used and combined by automatic theorem provers in their proof search. This is not trivial though: automatic theorem provers have to select the right lemmas for a proof, which becomes more difficult the more lemmas there are. Logipedia can also be a source of benchmarks for evaluating the expressiveness and automation of automatic tools. Lastly, Logipedia and Dedukti will form a framework to make automatic theorem provers cooperate with each other and other tools, in a safe way, the Logipedia infrastructure standing as a certifier for the end result.

From automatic theorem provers to Dedukti

Similarly to interactive theorem provers, connecting automatic theorem provers to the Logipedia infrastructure strongly relies on the ability for automatic theorem provers to import statements from Dedukti and export proofs in some theory in Dedukti.

(a) Instrumentation of automatic theorem provers to produce proof traces.

The first step for connecting automatic theorem provers to the Logipedia infrastructure, and the library of proofs, is that automatic theorem provers should actually output some kind of proof, without enforcing strong requirement on the format or even the level of granularity of those proofs. SAT (satisfiability) solvers for propositional logic do have a perfectly well specified format for proof traces[27] and most SAT solvers are actually able to produce proof traces in that format. So, considering SAT solvers, the current status already meets the needs of Logipedia on this aspect.

However, instrumenting other automated reasoners is more challenging. Some SMT (Satisifiability Modulo Theories) solvers can produce some kind of proof trace, but the format is specific to the solver, and the proofs are sometimes difficult to replay due to their (lack of) granularity. Most mainstream theorem provers for predicate logic output proofs in a standardised language[28], but this language does not clearly specify the semantics of the proof steps.

Our goals are to improve reasoning tools to demonstrate the feasibility of producing sufficiently detailed proofs for connecting to Logipedia, and to design a set of theoretical methods and practical

[25] B. Konev and A. Lisitsa. "Computer-aided proof of Erdős discrepancy properties". In: *Artif. Intell.* 224 (2015), pp. 103–118

M. J. H. Heule, O. Kullmann, and V. W. Marek. "Solving and Verifying the Boolean Pythagorean Triples Problem via Cube-and-Conquer". In: *Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference*. Vol. 9710. LNCS. 2016. ISBN: 978-3-319-40969-6.

[26] J. Filliâtre and A. Paskevich. "Why3 - Where Programs Meet Provers". In: *Programming Languages and Systems - 22nd European Symposium on Programming, ESOP 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013*. Vol. 7792. LNCS. 2013. ISBN: 978-3-642-37035-9

J. Protzenko et al. "Verified low-level programming embedded in F". in: *PACMPL* 1.ICFP (2017), 17:1–17:29.

[27] N. Wetzler, M. Heule, and W. A. H. Jr. "DRAT-trim: Efficient Checking and Trimming Using Expressive Clausal Proofs". In: *Theory and Applications of Satisfiability Testing - SAT 2014 - 17th International Conference, Held as Part of the Vienna Summer of Logic, VSL 2014*. Vol. 8561. LNCS. 2014. ISBN: 978-3-319-09283-6.

[28] G. Sutcliffe. "TPTP, TSTP, CASC, etc". In: *Computer Science - Theory and Applications, Second International Symposium on Computer Science in Russia, CSR 2007*. Vol. 4649. LNCS. 2007. ISBN: 978-3-540-74509-9.

tools that can be used to further connect Logipedia to the other existing automated reasoners of the same kind. We will mainly work on two SMT solvers (Alt-ergo, VeriT) and one prover for predicate logic (the E-prover). We will also consider more specific reasoning tools, with the aim to demonstrate that this approach also applies to more specific reasoners. We envision to experiment with this approach on a reasoner for Coherent Logic. These reasoners are particularly useful for geometric reasoning, and as such, they are quite complementary to the other considered automatic reasoners. They are also very suitable as a first experiment, since their proofs are fairly straightforward to interpret. They thus constitute a very interesting low-hanging fruit.

It is not possible, and not even desirable, to require all tools to directly be fluent in the language of Logipedia. Indeed, proof trace languages that are specific to one kind of reasoning tool are more appropriate than Dedukti for the instrumentation of large pieces of software, enabling quick output, and allowing post-processing the produced proof traces at the right level of abstraction. Furthermore, provided that those proofs are detailed enough, translation of traces to Dedukti will not be a difficult task, and the work necessary to translate proof traces for a myriad of very different reasoners will be implemented in a single tool (Ekstrakto, see below) in order to take advantage of the fact that there is quite a lot of sharing of reasoning techniques, and thus proof methods, among provers and solvers.

(b) Translate traces of automatic theorem provers into Dedukti.

As pointed out above, it is easier to instrument provers to make them output traces, instead of directly provide Dedukti proofs. The second step to connect automatic theorem provers to the Logipedia infrastructure is to reconstruct the proof traces in order to build Dedukti proofs from them. The proposed process is the following: each step of the trace is transformed into an independent sub-problem; each of these sub-problems is given to a prover that can output Dedukti proofs; proofs of the sub-problems are then combined to produce a global proof of the original problem. Since sub-problems correspond to atomic steps of the proof trace, they are relatively simple, so that we are confident that the prover producing Dedukti proofs will not struggle to find a proof. This process is quite similar to what is done by the hammer tools of interactive theorem provers (Sledgehammer in Isabelle/HOL, HOLyHammer for HOL4, etc.) which reconstruct proofs from traces produced by automated theorem provers.

This scheme has already been prototyped in a tool called Ekstrakto. Ekstrakto takes a TSTP file, as can be produced by, for example, the E prover for first-order logic, and it uses Zenon Modulo and ArchSAT to prove the sub-problems. Ekstrakto was designed to be agnostic with respect to the prover producing the trace; in particular it does not depend on the specific set of inference rules of the prover. It was also designed to be agnostic w.r.t. the prover used to prove the sub-problems; it is only required that the prover can output a Dedukti proof in the correct expression of predicate logic.

Although the experience we already have with Ekstrakto gives us a total confidence that the approach will be successful, it is work in progress. In particular, the following issues will be addressed in the project:

- Up to now, Ekstrakto can only understand traces in TSTP format as input. We plan to make it accept traces in other formats, notably traces from SMT solvers, as well as all formats that will appear when instrumenting other automatic theorem provers.
- Some steps in the proof traces are not provable: their conclusion is not a logical consequence of their premises. However, they preserve provability: the original problem has a proof if and only if the problem with the conclusion of the step also has a proof. This is the case for instance for the Skolemisation step performed by an automated theorem prover for predicate logic, for the introduction of new definitions, as well as for the RAT property in traces produced by SAT solvers. The approach of Ekstrakto cannot be used here, because the sub-problem corresponding to the step cannot be proved. However, since provability is preserved, it should be possible to transform a proof using the conclusion of the step into a proof using its premises. Such a transformation depends on the nature of the step that has been used. We will include in

Ekstrakto a mechanism to handle Skolemisation and definition introduction, which are sufficient for managing all traces from the major theorem provers for predicate logic.

- Dedukti-producing provers used by Ekstrakto, namely Zenon Modulo and ArchSAT, are meant for pure predicate logic. However, it will be necessary to deal with proof traces that use some specialised theory, for example arithmetic or bit-vectors, as could be output by SMT solvers. Although such theories could be presented as a set of axioms in predicate logic, it is almost certain that neither Zenon Modulo nor ArchSAT could be able to find non-trivial proofs using these axioms. Here, the idea is to develop small provers dedicated to a particular theory that can output Dedukti proofs. Such provers would be called when a step in the trace relies on said theory. These provers need not be very optimised, since trace steps are relatively small; this should help to produce Dedukti traces. A way to achieve this could be to extend Zenon Modulo: indeed, Zenon modulo can find proofs modulo arithmetic, but it cannot produce a Dedukti proof yet.

From Dedukti to automatic theorem provers

In the other direction, Logipedia will constitute a source of knowledge for automatic theorem provers. For this to be affordable, this project will tackle the following challenges.

(a) Translate Dedukti statements into automatic theorem provers inputs.

Automatic theorem provers are mostly based on (fragments of) predicate logic. Logipedia theorems, which mostly come from interactive provers, will be expressed in the Dedukti encodings of much more expressive logics, such as dependent type theory or simple type theory. Theorem statements thus need to be expressed in a different way so that automatic theorem provers can deal with them.

Encodings of expressive logics in predicate logic already exist, and are used for instance in hammers for using automatic theorem provers inside interactive theorem provers[29]. In these tools, the encodings are specific to one system and tuned to its logic. In Logipedia, we have to combine and take benefit from statements coming from the encodings of different systems based on different logics. The key challenge here are thus:

- to avoid the loss of meaning arising from a succession of encodings; and
- to encompass statements coming from different systems.

We plan to investigate a new approach where, instead of hammers, the encoding is a succession of fine-grained encodings dedicated to one aspect. These "small" encodings will offer the possibility to be activated independently, depending on the origin of the statement. In addition, they have the other advantages of being modular, easily extensible, and more reliable: each encoding is simple, and may output proofs, for example using Ekstrakto. They will be performed modulo the concept alignment.

It is common in the automatic theorem proving community to evaluate the performance of a particular tool on sets of benchmarks. As a side effect of our work on integrating automatic theorem provers, we will be able to extract a new set of benchmarks from Logipedia to measure the performance of automatic theorem provers on problems coming from different logics, and from a combination of these logics. In our case, this will not only allow automatic theorem provers to be compared against each other but also to measure the success of this task by evaluating the quality of the encodings, and comparing the impact of each of our "small" encodings by activating or deactivating them individually.

[29] L. C. Paulson and J. C. Blanchette. "Three years of experience with Sledgehammer, a Practical Link Between Automatic and Interactive Theorem Provers". In: *The 8th International Workshop on the Implementation of Logics, IWIL 2010*. Vol. 2. EPiC Series in Computing. 2010

L. Czajka and C. Kaliszyk. "Hammer for Coq: Automation for Dependent Type Theory". In: *J. Autom. Reasoning* 61.1-4 (2018), pp. 423–453.

(b) Logipedia as a source of knowledge for automatic theorem provers.

The amount of knowledge available in the whole of Logipedia is significantly larger than in any library of an individual proof assistant, and selecting the knowledge relevant for a particular goal might require more complex techniques than before[30]. We will experiment with integrating techniques from machine learning to the construction of formal proofs. This includes both techniques for selection of relevant knowledge for a goal and for the selection of most promising constructors in the direct proof term construction[31].

Large scale application: formal verification of C code

The use of formal methods in industry requires approaches that apply to a wide variety of cases. For the verification of C code, the Frama-C platform features numerous techniques. One of them relies on automatic solvers: Frama-C-WP. Since automatic theorem provers are built by making many choices, such as which heuristics or which algorithms to use, they display blind spots, i.e., specific cases that are not solved efficiently. To overcome this limitation, it is necessary to use a wide variety of solvers in a portfolio manner. For instance, the Why3 tool features the ability to send problems to lots of provers in a uniform way. However, with so many tools written by different teams, the meaning of the same concept in the different tools could be different. This can lead to errors in the overall verification results. Moreover, in order to be applied more widely, formal methods must handle more concepts such as floating-point numbers whose exact meaning is less clear than mathematical integers. This leads to more chances for two tools to interpret the same concept differently. Finally, industrial users sometimes need to add a form of reasoning or a simplification that is specific to their particular concept, so that it is better handled by automatic solvers. It is very easy to make an error in those simplifications.

In order to overcome these problems, we propose to improve the assurances in the interaction between these different tools, and to speed up the addition of new features for handling new cases by using proof objects:

- For the industrial users who need to define the concepts they want to verify in their C code, we will add the possibility to import concepts from Logipedia. The user gains time by not having to define these concepts themselves and by being able to use proofs for accompanying lemmas.
- For writing the simplification rules for industrial use cases, we will instrument Frama-C-WP to generate Ekstrakto input in order to produce a proof of the correctness of these simplifications.
- For transforming problems to fit the different provers in Why3, we will take inspiration from the fine-grained encodings (see above) to also generate an Ekstrakto proof.
- In the end, we will gather the Dedukti or Ekstrakto proofs of the provers, the encodings, and the simplification rules, in order to assemble them in a coherent whole.

This part thus combines and validates most of the previous aspects of automatic theorem proving, but also constitutes the challenge to instrument a large-scale formal tool for transforming and combining proofs extracted from automatic theorem provers.

Automatic theorem provers to increase Logipedia readiness

Providing automation at the level of an encyclopedia of formal proofs is both interesting and challenging. Indeed, making so many different formal systems cooperate requires a lot of proof manipulation, transformations, and gluing, which can only be reasonably done with automation. On a higher level, being able to more efficiently write proofs directly in Dedukti would in itself be of interest to the community. Thus, a major challenge is to make this automation practicable and scalable.

(a) Automatic theorem provers for Dedukti

[30] G. Irving et al. "DeepMath - Deep Sequence Models for Premise Selection". In: *NeurIPS 2016*. 2016.

[31] M. Zielenkiewicz and A. Schubert. "Automata Theory Approach to Predicate Intuitionistic Logic". In: *Logic-Based Program Synthesis and Transformation - 26th International Symposium, LOPSTR 2016, Revised Selected Papers*. 2016.

Automation has been a key to the success of proof assistants, since it allows for much more efficient formalisation[32]. Many different techniques for providing automation to proof assistants have been developed over the years: techniques based on rewriting, tableaux[33], or even the integration of efficient superposition-based provers for predicate logic[34] and simple type theory[35]. Most recently, various machine learning techniques have been developed for proof assistants allowing for more precise selection of relevant knowledge[36] or even the prediction of useful proof techniques[37].

The automation techniques offered by different proof assistants differ significantly. For systems mostly based on classical predicate logic it is often possible to provide sound and complete translations to existing efficient theorem provers for predicate logic by expressing the intricacies of their type systems[38]. This allows for straightforward native proof reconstruction. On the other end of the spectrum, for systems based on involved foundations, such as the constructive type theory behind Coq and its variants, providing even a slightly useful automation is a significant challenge. Most efficient automation still relies on translations to external tools, but the current translations from such logics to the logic of SMT solvers[39] or predicate logic[40] are incomplete and sometimes (partially) unsound. This means that to reconstruct such proofs in the logic of the proof assistant separate intricate components need to be developed.

As part of the project we will experiment with various kinds of proof automation on the level of Dedukti and verify their applicability to the different libraries imported as part of the Logipedia project. Furthermore, we will experiment with machine learning techniques for formal proofs. This includes both techniques for selection of relevant knowledge for a goal and for the selection of most promising constructors in the direct proof term construction[41]. Finally, we plan to look at computational proof reconstruction, to complement the link to automated theorem provers developed in the other parts.

(b) Automation for Logipedia.

The aforementioned automation will be crucial to enable full proof verification. Considering systems one at a time, a number of proof exports are not complete, that is the actual systems do not provide all the proof step details, and internal automation for Logipedia would be necessary. More demandingly, automation will be used to fill the gaps between systems. In particular, concept alignments need to be established formally. More generally, proof obligations arising from translations, encodings into Dedukti, reverse mathematics etc. must be discharged as automatically as possible.

As it has been the case for interactive theorem provers, automation will also provide support for Logipedia developers and users. During the project, many Dedukti developments shall be performed, in particular to define theories in Dedukti. In the longer term, Dedukti will also be used, for example for new proof transformations or to align distant concepts, which will be affordable with automation.

[32] T. Hales. "Developments in Formal Proofs". In: *Séminaire Bourbaki* 1086 (2013–2014). CoRR/abs 1408.6474.

[33] L. C. Paulson. "A Generic Tableau Prover and its Integration with Isabelle". In: *J. Universal Computer Science* 5.3 (1999), pp. 73–87.

[34] J. Hurd. "First-Order Proof Tactics in Higher-Order Logic Theorem Provers". In: *Design and Application of Strategies/Tactics in Higher Order Logics (STRATA 2003)*. NASA Technical Reports NASA/CP-2003-212448. 2003.

[35] A. Asperti and E. Tassi. "Higher order Proof Reconstruction from Paramodulation-Based Refutations: The Unit Equality Case". In: *Mathematical Knowledge Management (MKM 2007)*. Vol. 4573. LNCS. 2007.

[36] J. C. Blanchette et al. "Hammering towards QED". in: *J. Formalized Reasoning* 9.1 (2016), pp. 101–148.

[37] T. Gauthier, C. Kaliszyk, and J. Urban. "TacticToe: Learning to Reason with HOL4 Tactics". In: *LPAR-21. 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning*. Vol. 46. EPiC Series in Computing. 2017.

[38] C. Kaliszyk and J. Urban. "MizAR 40 for Mizar 40". In: *J. Autom. Reasoning* 55.3 (2015), pp. 245–256.

[39] M. Armand et al. "A Modular Integration of SAT/SMT Solvers to Coq through Proof Witnesses". In: *Certified Programs and Proofs - First International Conference, CPP 2011*. Vol. 7086. LNCS. 2011. ISBN: 978-3-642-25378-2.

[40] Czajka and Kaliszyk, see n. 29.

[41] Zielenkiewicz and Schubert, see n. 31.

The success of the integration of proofs coming from automated theorem proving systems, SAT solvers, SMT solvers, first-order provers and other automatic reasoning tools, can be measured by the size of the proofs imported from these systems to Logipedia. The success of the export of Logipedia proofs into automated provers can be measured by the number of exported theorems. The number and complexity of C programmes verified with *a posteriori* proof checking will measure the success of combining automatic theorem provers with Logipedia.

Large libraries

We have seen how to instrument proof systems and automated theorem provers to produce proofs in Dedukti, and how to share these proofs in Logipedia.

Very large proofs, requiring large libraries of auxiliary results, have been developed in some proof systems. These large proofs and libraries are interesting for our project in two respects. First, they constitute an ambitious benchmark for the methods developed above. Secondly, they will serve as the source for the lion's share of proofs in Logipedia for end-user applications.

The libraries we target are dedicated to particular application areas. They provide a substantial coverage of one application area and do so in a structured manner. Integrating these libraries into Logipedia thus requires to preserve this structure according to application specific ontologies, that support browsing and searching. This structure leverages the infrastructure and will be a stress test for the results of structuring discussed below.

The libraries we focus on have been selected according to the following criteria:

- Relevance: the libraries should support core areas of mathematics and computer science.
- Coverage: the libraries should have a wide coverage of an application area.
- Maturity: the libraries should have already been used in a significant application.

As a result we selected the following libraries: the Archive of Formal Proofs, Isabelle's revised Analysis and Probability library, GeoCoq, Flyspeck and CakeML. Other interesting libraries such as MathComp, Coq's revised Analysis library, CompCert, seL4, and selected Mizar and PVS libraries are left for future work.

Isabelle's Archive of Formal Proofs (AFP) is a growing user-contributed online library for Isabelle.

In Feb-2020, the Archive of formal proofs consisted of more than 500 entries (articles of formalised mathematics) by 340 authors, and required approximately 60h CPU time for checking (using many gigabytes of memory). The purpose of this is to scale up the Isabelle instrumentation for Dedukti further, to cover major parts of this library. We do not intend to restructure or document the library further beyond what is provided by the authors of each article and by the hierarchical dependencies among the articles.

The key challenge is scaling. The ultimate aim is to export the main substance of the Archive of formal proofs without promising full coverage: some entries with prohibitive resource requirements will be omitted. The Archive of formal proofs is continuously growing at a high rate and thus represents a moving target.

The Isabelle Analysis & Probability Theory Library consists of more than 200.000 lines of definitions and proofs, corresponding to almost 4000 printed pages. It covers topology, homology, multivariate, functional and complex analysis, measure and probability theory, and a dedicated library for ordinary differential equations. It is fair to say that it is the most advanced machine-checked library in the area of analysis and probability theory. The Isabelle analysis and probability theory library is in part based on the extensive library of the HOL Light system but has generalised it from real numbers to appropriate algebraic structures and includes areas absent from the HOL Light library (in particular measure theory and ordinary differential equations). It is currently used by 60 out of 500 articles in the Archive of Formal Proofs, a user-contributed library of Isabelle/HOL proofs from all areas of computer science

and mathematics. Moreover, it is the only existing library in this area where proofs are structured and readable. Due to its size and generality and because analysis and probability theory are pervasive in any application in the engineering and natural sciences and economics, this library is a key exploitable result of the project: it is a fundamental enabling resource for almost any formal verification activity in these areas, for example autonomous driving and mathematical finance. We therefore need to structure, document and develop this library for optimal accessibility, ease of use and exhaustiveness. The goal is to have a curated library for applications. So, some material will need some revisions and some new material will need to be added as well.

Accessibility: The library is a large collection of theories with a hierarchical dependency relation. However, this structure has grown over time and does not always reflect the abstract mathematical dependencies. In short, the structure needs to be modularised. This requires a significant refactoring effort. At the same time we need to add metadata to the source material to turn this structured collection of theorems and proofs into a curated library at the Dedukti level (see below). The Analysis & Probability Theory library will be used as a first use case of the infrastructure for handling metadata that will also be developed in this project, in particular the ability to describe ontologies. This will notably entail annotating those statements in the library that correspond to proper mathematical theorems as opposed to auxiliary lemmas required only for the benefit of the theorem prover. To increase accessibility we will also include links from the library into Wikipedia and in particular in the other direction to raise the awareness of Logipedia.

Development: Although the library support for integrals is extensive, it suffers from the (necessary) coexistence of different kinds of integrals. This complicates proofs about integrals and so needs to be unified, which entails further refactoring. The rest of the development adds further essential material:

- Fourier analysis because of its extreme importance both for pure mathematics and for engineering and physics (especially the Fourier transform). A formalisation of basic properties of the Laplace transform is already available in the Archive of formal proofs.
- Stability theory for differential equations and dynamical systems, in particular Lyapunov functions, because of their relevance for the verification of cyber-physical systems.
- Stochastic differential equations to model a large range of dynamical systems with stochastic components, from physics to financial markets.

The GeoCoq library. The GeoCoq library consists of more than 100.000 lines of definitions and proofs in geometry. It is mostly based on synthetic approaches, where the axiom system is based on some geometric objects and axioms about them, but, following Descartes and Tarski, it proves that the analytic approach can be derived, where a field F is assumed (usually $\mathbb{R}$) and the space is defined as F^n. Moreover, it contains a model, based on the analytic approach, of one of the synthetic approaches present in the library, namely Tarski's system of geometry, thus establishing the connection between these two approaches in the opposite direction.

The fact that the analytic approach can be derived from the synthetic one is called the arithmetisation and coordinatisation of geometry and it represents the culminating result of both Hilbert[42] and Tarski[43]. As of now, there is no other formalisation of this result inside a proof assistant, making the GeoCoq library the most advanced machine-checked library in the area of synthetic geometry. This formalisation enables to obtain automatic proofs based on geometric axioms using algebraic automated deduction methods, some of which will be covered by automated theorem proving.

The main axiom system in this library is the one of Tarski, but Hilbert's axiom system and a version of Euclid's axioms are sufficient to prove the propositions in Book 1 of Euclid's Elements are also defined. Thanks to the proof that Tarski's axioms (except continuity) are equivalent to Hilbert's axioms (except continuity), the arithmetisation and coordinatisation of geometry are available for both systems. In the library, the focus is not only on axiom systems but also on axioms themselves. Eleven continuity

[42] D. Hilbert. *Grundlagen der geometrie*. Teubner, 1899.

[43] W. Schwabhäuser, W. Szmielew, and A. Tarski. *Metamathematische Methoden in der Geometrie*. Springer-Verlag, 1983.

axioms are available and are hierarchically organised. Finally, it contains a new refinement of Pejas' classification of parallel postulates together with proofs of the classification of 34 versions of the parallel postulate.

Challenges: A lot of preliminary work has been done on making the GeoCoq library available in Logipedia. One of the remaining obstacles is the use of computational steps in Coq proofs. The issue is that proofs making use of proof by reflection reach a level of complexity that currently makes verification by Dedukti impractical, but these proofs are very frequent in Coq. One possible approach is to isolate these proofs by reflection so that they are not perceived as simple conversion steps in the type theory proofs, but marked as proofs to be treated by an automatic tool. The coherent logic theorem prover should handle part of these proofs. The other ones should be handled by specific provers for geometry, as they correspond to proofs obtained by the Gröbner basis method in GeoCoq. Another challenge is that CoqInE, a tool developed to translate Coq proofs into Dedukti type-checkable terms, produces terms in an encoding of the Calculus of Inductive Constructions in Dedukti. Currently, it is not possible to export these Dedukti terms to other proof assistants. However, another tool, Universo, has been developed and paves the way for the export of these terms. The last challenge is that part of GeoCoq relies on the MathComp library, which should also be exported to Dedukti to complete the integration of GeoCoq into Logipedia.

Refinements: The pragmatic approach that has been adopted so far has motivated a few choices that should be reconsidered once the obstacles with the instrumentation of Coq, encountered throughout this process, have been solved. So far, the solution that has been preferred in most cases was to modify the Coq script so that it does not rely on features, such as universe polymorphism, that were not handled by CoqInE. For example, the use of module functors was removed from GeoCoq, some proofs produced by tactics, such as 'intuition', that relied indirectly on parts of the standard library of Coq that were not available with CoqInE, were reworked by hand to avoid such dependencies, or direct dependencies were avoided. Another choice that could be reconsidered is to use automated theorem provers to replace the proofs by reflection.

The Flyspeck library. The Flyspeck library is the result of the Flyspeck project, a formal proof of the Kepler conjecture. The statement and proof is based on an original proof of Thomas Hales[44], and is formalised and proved largely in HOL Light. HOL Light has a large and rich analysis and geometry library, with some results not formalised in any other system. This motivates the project of importing the HOL Light library and the Flyspeck project into Dedukti and thus into Logipedia.

Challenges: Proofs coming from the HOL systems, including HOL Light, are known to be very large, adding to the already existing issue of the scalability of exporting software for large libraries[45]. As an example, the size of the standard library is 1.5MB in HOL Light files, while the same files once exported as Dedukti files grow to 2GB, and the whole HOL Light library and the Flyspeck project are each as large as 30MB. The multitude of automatic tactics, their increasing complexity, and the number of dependencies suggests that the generation time, size, and rechecking time of exported proofs will not increase linearly. Scalable export techniques for HOL Light proofs have been investigated[46] and can provide a solid base to this project. As we aim at significantly reduce the size of generated Dedukti files, the time of export could at least be reduced.

[44] T. C. Hales et al. "A formal proof of the Kepler conjecture". In: *CoRR* abs/1501.02155 (2015).

[45] W. Wong. "Recording and Checking HOL Proofs". In: *Higher Order Logic Theorem Proving and Its Applications, 8th International Workshop*. Vol. 971. LNCS. 1995. ISBN: 3-540-60275-5

S. Obua and S. Skalberg. "Importing HOL into Isabelle/HOL". in: *Automated Reasoning, Third International Joint Conference, IJCAR 2006*. Vol. 4130. LNCS. 2006. ISBN: 3-540-37187-7

C. Keller and B. Werner. "Importing HOL Light into Coq". In: *Interactive Theorem Proving, First International Conference, ITP 2010*. Vol. 6172. LNCS. 2010. ISBN: 978-3-642-14051-8

R. Kumar. "Challenges in Using OpenTheory to Transport Harrison's HOL Model from HOL Light to HOL4". In: *Third International Workshop on Proof Exchange for Theorem Proving, PxTP 2013*. Vol. 14. EPiC Series in Computing. 2013.

[46] C. Kaliszyk and A. Krauss. "Scalable LCF-Style Proof Translation". In: *Interactive Theorem Proving, ITP 2013*. Vol. 7998. LNCS. 2013.

The main milestones of this task are the further automation of the translation from HOL Light to Dedukti, the import of the multivariate analysis library, the import of the whole HOL Light library, and the import of the Flyspeck in Dedukti.

The CakeML compiler library. The CakeML compiler library[47] consists of the CakeML programming language definition, its compiler, and the correctness proofs about the compiler.

This is a cutting edge compiler library and one of only two verified compilers for real languages, the other being CompCert. Its export to Dedukti is one of the key exploitable results of this project.

The CakeML compiler development lives within the HOL4 prover. HOL4 can export definitions and proofs in the OpenTheory format, which can in turn be translated into Dedukti. This link from HOL4 via OpenTheory to Dedukti exists but, in its current state, fails to scale to the task of transporting something as sizeable as the CakeML compiler development. This part of this task will rework the route via OpenTheory to scale better, possibly taking inspiration from an OpenTheory-like approach that scaled well for the HOL light prover[48].

Challenges. The two key challenges are scalability (pushing the technology to cope with the size of proofs) and accessibility (presenting the library in an accessible manner to end-users). This will be a stress test for the infrastructure.

Contributions. The aforementioned libraries will populate Logipedia with mature, application-level mathematical theories. That is, proofs about important areas of mathematics (for example, analysis and algebra) and computer science (for example, compilers) that support applications in engineering as well as security and in particular in interdisciplinary areas such as cyber-physical systems and autonomous vehicles.

As one of the challenges is scalability, one measure of success is how much of the libraries we will be able to import into Logipedia. A second measure of success is how well we are able to restructure the libraries for end-user access and how well this structure can be transferred to Logipedia and thus to end-users.

Infrastructure of the encyclopedia

In the three last sections dedicated to integration, automated theorem proving, and large libraries, we have described the concepts that are at the centre of the networking activities that will populate Logipedia with formal proofs coming from the libraries of various systems.

Let us now turn to the development of the integrating infrastructure itself and its accessibility. To make such an infrastructure usable, besides trans-national and virtual access, we must also provide an ergonomic web interface, a package distribution system, a search engine for mathematical formulas, a structure of encyclopedia that relies upon an enrichment of the data with metadata.

Sharing a common platform. The libraries of proofs described in previous sections can be quite large: several gigabytes for millions of mathematical objects (definition or theorem) in thousands of files. Translating them to Dedukti may increase this size by one order of magnitude as the Dedukti proof format is of a lower level than the high level languages used by humans in interactive proof assistants for defining objects or proving properties.

Moreover, these mathematical objects can have many dependencies: a proof of a theorem usually relies on the definition of various objects and other proofs as well (lemmas). Hence, verifying the

[47] R. Kumar et al. “CakeML: a verified implementation of ML”. in: *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL ’14*. 2014.

[48] Kaliszyk and Krauss, see n. 46.

correctness of a proof requires to check the correctness of other proofs, and translating a proof requires the translation of many other definitions and proofs.

We want to provide a unique identifier for each proof in Logipedia in order for researchers and publishers to easily refer to their work, and allow reproducibility in mathematics as the correctness of Dedukti proofs is formally checked.

For all these reasons, it is convenient to gather on a single server all these proofs, as well as all the new proofs that will be developed in the future. We will need some mirror-site though for ensuring the permanent availability of the service even in case of technical problems or system updates.

Generating those Dedukti proofs can also be quite demanding in terms of computation power and computation time: depending on their size or complexity, translating those libraries to the Dedukti format may require just a few minutes, a few hours, or up to a few days.

It is therefore convenient that the consortium shares high performance computers to generate those proofs, and all the proofs that will be developed in the future.

Here are examples of large (sets of) libraries with the number of files, the number of objects (definition or theorem), the size (in Megabytes) and the current estimation of the multiplying size factor when for translating them into Dedukti:

Library	Nb files	Nb objects	Size (Mb)	Dedukti Size
Opam-Coq	16,000	473,000	235	$\times 7$
AFP	7,000	90,000	135	$\times 30$
MML	1,400	77,000	101	$\times 5 - \times 14$
Flyspeck	50	25,368	34	$\times 67$

Continuous integration and deployment. Modifying the Logipedia platform in order to add new features or implement new formal proofs will be a collaborative work. To ensure the reliability of the platform and, in the meantime, to be able to efficiently and frequently update it on the production infrastructure, it is necessary to rely on best software development practices. We will use a continuous integration and continuous deployment pipeline to manage the development process.

The pipeline is composed of several open source tools orchestrated to automatically deliver new versions of the Logipedia platform.

- A Version Control System stores and manages the source code and allows its modification securely and with traceability. Git is the leader open source A Version Control System tool and will be used for Logipedia source code. Git implements a distributed version control system that applies perfectly to the context of a European collaborative project. For Logipedia, the Version Control System will handle the source code of the Logipedia website itself as well as the formal proof libraries.
- A continuous integration and continuous deployment platform is used to automate the integration of modifications as soon as a new version is pushed to the Version Control System. The automatic integration consists in building the new application, executing tests to ensure the quality is maintained, and producing packages of the application. Jenkins is a widely used continuous integration and continuous deployment tool and will be used for Logipedia. Once packages are built, the continuous integration and continuous deployment platform can trigger the automatic deployment of the new version on the target infrastructure. This step is called continuous delivery.

In practice, the automatic delivery is often done on a test environment to allow users to verify that the new version of the platform behaves as expected. When manual tests are done, maintainers can push the new version to the production environment.

Automating integration and delivery ensures that the full process will always be the same, regardless of who made the modification of the source code. It also ensures that tests are executed and passed to keep a good quality level and a high availability of the platform.

Online access of findable, accessible, interoperable, and reusable formal proofs. The vast majority of formal proofs developed in the world are publicly available, released under some free licence, and accessible on the Web. Even in the industry, only the small parts that are very specific to a project or a client need to be protected and are not publicly available.

In order to provide Logipedia to the largest audience in research, education, industry, and publishing, we have to make the platform shared by the consortium easily and freely accessible online. This will make these data findable, accessible and reusable, making them interoperable being at the heart of the Logipedia project.

On the other hand, for security reasons, the access to the computation power of the platform has to be allowed to pre-identified persons only.

Search tools. Again, to make the encyclopedia usable, it is necessary to add search functionalities to efficiently find specific theorems and proofs of interest by a specific user. It should be possible to search by name, other metadata, semantic annotations, and by the actual structure and content of these theorems and proofs.

Searching in a collection of millions of regular documents, representing gigabytes of data or more, is a problem easily solved by standard *information retrieval* techniques[49], which routinely scale to several orders of magnitude higher when indexing huge corpora, such as the World Wide Web. Elasticsearch is an example of a highly customisable and scalable search engine for large document collections. But these techniques and tools are adapted to the case of (semi-structured) natural language documents by full text queries, along with other metadata. But they are not adapted to the case of mathematical statements with complex content and structure, as well as semantic annotations.

The case of semantic annotations is relatively easy to solve. SPARQL[50] is a query language for (RDF) semantic annotations expressing complex semantic queries, for which numerous query engines exist; popular (and open-source) ones include Apache Jena and Virtuoso. These systems also support efficient indexing and querying of textual content[51], which allows queries combining textual search and semantic annotations.

Exploiting the structure and content of mathematical statements is a more challenging, open, and interesting issue.

Early efforts have simply considered basic exact keyword and regular expression matching[52], or searching for the exact type of a lemma, which requires type isomorphism tests[53].

One approach to improve on such techniques is to adapt standard information retrieval techniques to work with terms occurring in mathematical tools. This has been done[54] with *latent semantic indexing*, an indexing and retrieval technique based on co-occurrences of terms within documents, which is adapted to work with statements from the Mizar Mathematical Library (MML): terms, which are (lemmatised or stemmed) words in regular natural language documents, are chosen to be type names, operators, etc., in the mathematical language of the MML. Similarly to regular information retrieval techniques, however, such a search cannot interpret complex structures.

To deal with the actual (hierarchical) content of mathematical formulas and statements, systems such as MathWebSearch[55] (in the context of mathematical formulas in MathML within XHTML documents)

[49] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008. ISBN: 978-0-521-86571-5.

[50] W3C. *SPARQL Query Language for RDF*. 2008.

[51] E. Minack, W. Siberski, and W. Nejdl. "Benchmarking Fulltext Search Performance of RDF Stores". In: *The Semantic Web: Research and Applications, 6th European Semantic Web Conference, ESWC 2009*. Vol. 5554. LNCS. 2009.

[52] G. Bancerek and P. Rudnicki. "Information Retrieval in MML". in: *Mathematical Knowledge Management, Second International Conference, MKM 2003*. Vol. 2594. LNCS. 2003.

[53] D. Delahaye. "Information Retrieval in a Coq Proof Library Using Type Isomorphisms". In: *Types for Proofs and Programs, International Workshop TYPES'99*. Vol. 1956. LNCS. 1999.

[54] P. A. Cairns. "Informalising Formal Mathematics: Searching the Mizar Library with Latent Semantics". In: *Mathematical Knowledge Management, Third International Conference, MKM 2004*. Vol. 3119. LNCS. 2004.

[55] M. Kohlhase, B. Matican, and C. Prodescu. "MathWebSearch 0.5: Scaling an Open Formula Search Engine". In:

and Whelp[56] (in the context of the standard library of Coq) propose to index not only individual terms, but the relation that these terms have with each other, allowing to search for somewhat complex hierarchical patterns within mathematical statements.

As an ambition of the Logipedia project, we aim to build search tools that allow combining full-text, semantic, and content search, with state-of-the-art information retrieval and indexing techniques, adapted to the case of a library of mathematical results and proofs in a diversity of languages and formalisms.

Access on everyone's computer. To allow researchers, students and engineers to study and reuse the proofs of Logipedia in their own work environment and tools, it is important to provide an application enabling anyone to download and install Logipedia proofs on one's own computer, and to translate those Dedukti proofs to the proof format (Coq, Isabelle/HOL, etc.) used by the user on her machine.

This tool will have to manage in a transparent way the many dependencies between mathematical objects. Hence, when the user will request a given proof, the tool will automatically download and install all the definitions and proofs it relies on as well.

Interface for teaching. The existing user interfaces for proving theorems in the classroom can be sorted roughly in two categories: the systems built by the community of educators and the systems built by specialist of interactive theorem proving.

The Geometry Tutor[57], Mentoniezh[58], Defi[59], Chypre[60], Cabri-Eulide[61], Geometrix[62] and Baghera[63], 2002) systems belongs to the first category. Using these systems the student can produce proofs interactively using a set of known theorems. In most of these aforementioned systems, the student cannot invent a proof very different from what the programme had pre-computed using automated theorem proving methods. As far as we know, the exception is Cabri-Euclide which contains a small formal system and therefore gives more freedom to the student. Baghera includes also e-learning features, such as task management and network communication between teachers and their students.

In the category of system which have developed by people from the interactive theorem proving community, we can distinguish between system which add syntactic sugar over a state-of-the-art proof assistant: PCoq[64], Coq Web[65], ProofWeb[66], Edukera[67]; and those who use an automatic theorem prover

Intelligent Computer Mathematics - 11th International Conference, AISC 2012, 19th Symposium, Calculemus 2012, 5th International Workshop, DML 2012, 11th International Conference, MKM 2012, Systems and Projects, Held as Part of CICM 2012. Vol. 7362. LNCS. 2012.

[56] A. Asperti et al. "A Content Based Mathematical Search Engine: Whelp". In: *Types for Proofs and Programs, International Workshop, TYPES 2004*. Vol. 3839. LNCS. 2004.

[57] J. R. Anderson, C. F. Boyle, and G. Yost. "The geometry Tutor". In: *IJCAI Proceedings*. 1985, pp. 1–7.

[58] D. Py. "Reconnaissance de plan pour l'aide à la démonstration dans un tuteur intelligent de la géométrie". PhD thesis. Université de Rennes, 1990.

[59] Ag-Almouloud. "L'ordinateur, outil d'aide à l'apprentissage de la démonstration et de traitement de données didactiques". PhD thesis. Université de Rennes, 1992.

[60] P. Bernat. *CHYPRE: Un logiciel d'aide au raisonnement*. Tech. rep. 10. IREM, 1993.

[61] V. Luengo. "Cabri-Euclide: Un micromonde de Preuve intégrant la réfutation". PhD thesis. Univ. Joseph Fourier, 1997.

[62] J. Gressier. *Geometrix*. 1988.

[63] N. Balacheff et al. *Baghera*. 1999.

[64] A. Amerkad et al. "Mathematics and Proof Presentation in Pcoq". In: *Workshop Proof Transformation and Presentation and Proof Complexities in connection with IJCAR 2001*. 2001.

[65] J. Blanc et al. "Proofs for freshmen with Coqweb". In: *PATE'07* (2007). bibtex: blanc2007proofs, p. 93.

[66] C. Kaliszyk et al. "Deduction using the ProofWeb system". In: (2008). bibtex: kaliszyk2008deduction.

[67] B. Rognier and G. Duhamel. "Présentation de la plateforme edukera". In: *Vingt-septièmes Journées Francophones des Langages Applicatifs (JFLA 2016)*. Ed. by J. Signoles. 2016.

above a pseudo natural language input: SAD[68], Naproche[69], Lurch[70], ELFE[71], CalcCheck[72].

Current formal proof systems require to learn a specific formal language and a command-line interface to compile and execute the language. This level of technicality is not compliant with mass adoption by teachers and students outside computer science. For educational purposes, it is therefore mandatory to develop an intuitive and easy-to-handle user-interface for the Logipedia formal proof system. This interface, based on a "What You See Is What You Get" design, should provide three main features:

- Proofs must be displayed in a structured and readable way. In particular, mathematical symbols must be drawn with high-quality typography equivalent to current standards (Latex, Mathjax,. . .)
- It must be possible to build proofs with simple point-and-click interactions. These interactions include applying theorems (or axioms or lemmas) to statements and rewriting rules to a selected element of a statement; this is done in deductive or abductive mode (resp. forward or backward).
- The interface should provide several types of presentations of the structure of a formal proof: deductive, a stack of scopes (assumptions and a statement to be justified), sequent tree style, and so on. The digital nature of formal proofs enables the interactive exploration of a proof to understand its structure: eagle-eye view, folding/unfolding of scopes, highlighting the use of variables (on hover), showing/hiding context, and so on.

One of the difficulties is to provide a way for the students to discover and integrate hundreds of lemmas and rewriting rules necessary to solve exercises. The solution consists of a mixture of step-wise tutorials and training exercises, plus an adapted ergonomic design of the application menus. In this regard, one should be able to search for theorems and rewriting rules.

The research of proof must be automated at some point in order to ease and speed up the process and to comply with the level of detail required in education.

Source Level Structure and Metadata

Like programmes, formal proofs exist at two levels: the user-written source code and the kernel representation.

The expression of theories in Dedukti focuses on integrating the low level representation obtained by the instrumentation of the kernels. Thus, a single architecture such as Dedukti can cross-verify proofs from any number of proof assistants.

But this eliminates a lot of salient information that is present in the original sources, including

- library structure in the form of documents, sections, theories, or modules,
- high-level statements and operators (which proof assistants typically elaborate when compiling the sources),
- extra-logical statements such as comments, metadata annotations, or processing instructions.

So in order to provide a structured view to the theories expressed in Logipedia, we need to complement the current expression of proofs in Dedukti and Logipedia, by integrating source level representations.

We will survey the most commonly used and most interoperability-relevant language features and add those to Dedukti. These will complement the existing language features in such a way that encodings in Dedukti can choose flexibly when and how source level features are preserved. We thus will annotate the kernel level representations with source level representations. We will develop a semantic web-style ontological representations of the libraries, in particular an ontological definition language to *define*

[68] A. Lyaletski, A. Paskevich, and K. Verchinine. "SAD as a mathematical assistant—how should we go from here to there?" In: *Journal of Applied Logic*. Towards Computer Aided Mathematics 4.4 (2006), pp. 560–591. ISSN: 1570-8683.

[69] M. Cramer et al. "The Naproche Project Controlled Natural Language Proof Checking of Mathematical Texts". In: *Controlled Natural Language*. Ed. by N. E. Fuchs. bibtex: 10.1007/978-3-642-14418-9_11. Springer Berlin Heidelberg, 2010, pp. 170–186. ISBN: 978-3-642-14418-9.

[70] N. C. Carter and K. G. Monks. "Lurch: a word processor built on OpenMath that can check mathematical reasoning". In: *Conferences on Intelligent Computer Mathematics*. 2013.

[71] M. Doré. "The ELFE Prover". In: *25th Automated Reasoning Workshop*. bibtex: dore2018elfe. 2018, p. 16.

[72] W. Kahl. "CalcCheck: A Proof Checker for Teaching the "Logical Approach to Discrete Math"". en. In: *Interactive Theorem Proving*. Lecture Notes in Computer Science. Springer, Cham, 2018, pp. 324–341. ISBN: 978-3-319-94820-1 978-3-319-94821-8.

meta-data in the style of Isabelle/DOF[73] that has specific features to annotate, track, and validate formal and semi-formal information.

This information heavily abstracts from the content of the libraries and is not meant to capture neither its syntax nor its semantics, but they are tremendously useful for shallow interoperability such as dependency management, navigation, and searching and querying. For example, searching for a particular theorem based on some information where it might be used can be enough to find it. We will develop a reference ontology for formal libraries and extend our encodings to make use of it. Moreover, we will develop *domain-specific* ontologies allowing, for examples, engineers to search for mathematical theorems relevant to a particular engineering discipline[74].

Annotation with meta-data is particularly relevant in connection with the concept alignments described below. Because, in its simplest form, alignment is just a binary relation between identifiers in different libraries that mean the same thing, it can be integrated with this ontology directly. It then enables queries that span multiple libraries, for example to find all theorems about a certain concept that have been in some provers but not others.

These data structures developed will be used critically in development of search engines for Logipedia. This includes the use of source level syntax in queries that is matched against corresponding source syntax in the libraries as well as semantic web-style queries in languages like SPARQL.

New theories, new systems

Having discussed the concepts related to the networking activities required for collecting the content of Logipedia and to the development of the infrastructure needed to make it accessible, we will describe joint research projects.

So far, we have considered systems that were already at LIL 1, that is, systems that implement a theory that we know how to express in Dedukti. Other relevant systems are still at level 0, that is, we do not yet know how to express them. Part of our project is to work on their theories: those of Mizar, TLA$^+$, PVS, Minlog, ProvenTools, K Prover, and Homotopy Type Theory, in order to understand how to express their theory in Dedukti.

These systems range from specialised systems such as Minlog used in academia to mainstream systems such as Mizar or PVS that have been used in major scientific or industrial projects. They provide specific features that are relevant to their target domains and are therefore preferred by users for developing theories and proofs in these areas. The objective is to bring them to LIL 2 or higher in order to prepare the grounds for their integration into Logipedia in the longer term.

There are various motivations for including these systems in Logipedia. First, being able to export theories and proofs developed in these systems makes them available to systems with more advanced integration level. For example, accessing Mizar's rich library of formalised mathematics would be beneficial to developers of machine-checked mathematical results. Second, certain systems such as TLA$^+$ have been used for specific tasks such as reasoning about distributed algorithms, but they do not provide rich general-purpose standard libraries, for instance, about integer or real arithmetic as the more mature systems. Finally, for some of the systems there exists only one implementation able to check the proofs. By exporting these proofs stemming from the target systems we will be able to cross-verify them. Neither goal will be fully achieved at the end of the project since higher Logipedia integration levels are necessary for exporting or importing full libraries. Nevertheless, the work carried out here is a necessary stepping stone and enabler for making Logipedia a universally accepted encyclopedia of formal proofs.

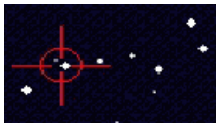
[73] A. D. Brucker and B. Wolff. "Isabelle/DOF: Design and Implementation". In: *Software Engineering and Formal Methods - 17th International Conference, SEFM 2019.* Vol. 11724. LNCS. 2019. ISBN: 978-3-030-30445-4

A. D. Brucker et al. "Using The Isabelle Ontology Framework: Linking the Formal with the Informal". In: *Conference on Intelligent Computer Mathematics (CICM)*. LNCS 11006. Hagenberg, Austria, 2018

A. D. Brucker and B. Wolff. "Using Ontologies in Formal Developments Targeting Certification". In: *Integrated Formal Methods (iFM)*. LNCS 11918. Bergen, 2019. ISBN: 3-540-25109-X.

[74] Brucker and Wolff, "Using Ontologies in Formal Developments Targeting Certification", see n. 73.

Mizar is one of the earliest proof assistants. It was initially created as a typesetting system for mathematics, with proof checking functionality added later. The language is designed to be as close as possible to the traditional language of mathematics where basic elements are represented as nouns, whereas their properties and operations can be represented using adjectives, predicates, and functors, as well as a hierarchy of soft types. Proofs are expressed in a declarative way based on natural deduction.



The Mizar community maintains a centralised database, the Mizar Mathematical Library (MML). Based on the axioms of Tarski-Grothendieck set theory, Mizar[75] provides a framework on top of which new theories can be built in the form of "articles". The library constitutes one of the largest collections of formal mathematics with a number of results that are not present in other systems. Apart from mathematical formalisations, Mizar has been also used for developing formal descriptions of theoretical models of computation, cryptography, digital circuit design, and software verification. MML provides a database for training automated theorem provers and other projects based on machine learning techniques.

The Mizar proof checker differs from other proof assistants in that the semantics of a proof step corresponds to the notion of "obviousness" to a human mathematician. Mizar contains modules that check different aspects of the correctness of Mizar articles in a compiler-like manner, from purely syntactic concerns all the way to logical completeness and consistency. Mizar has not been designed to produce proof objects, and much background knowledge (for example, stemming from soft types) is used implicitly. The main challenges of encoding Mizar in Dedukti will be to represent the different forms of Mizar definitions in a uniform way and to instrument the system to make this implicit knowledge explicit and allow Mizar proofs to be checked by Dedukti.

TLA$^+$[76] is a specification language based on Zermelo-Fraenkel set theory and the Temporal Logic of Actions, a dialect of linear-time temporal logic.

It is intended for the precise description of discrete systems and in particular of distributed algorithms and systems.

TLA$^+$[77] has been substantially adopted by companies working on distributed and cloud systems[78]. The two main software tools for analysing and verifying TLA$^+$ specifications are the model checker TLC[79] and TLAPS, the TLA$^+$ Proof System[80].

TLA$^+$ proofs are written in a declarative, hierarchical style based on natural deduction that allows the user to break down the proof of high-level theorems into lower-level steps. The proof obligations corresponding to the leaves of this proof tree are discharged by automated prover back-ends, including SMT solvers, the Zenon tableau prover, an encoding of TLA$^+$ set theory as an object logic Isabelle/TLA$^+$ in the logical framework Isabelle, and a decision procedure for linear-time temporal logic.

Our objective in this project is to lift the non-temporal fragment of TLA$^+$ (which covers the overwhelming majority of proof obligations arising in system verification) from LIL 0 to at least LIL 2. The main challenges are encoding the untyped set theory underlying TLA$^+$ in Dedukti and to instrument TLA$^+$ in order to export proof traces that can be checked in Dedukti, starting with the Zenon and Isabelle back-ends.

One of the case studies will be the specification of one of the software systems developed within MED-EL, which requires ad-hoc modifications of instances of a pre-defined process. Every new change to the process might contain fewer, more or different steps that should be performed by an end user (a patient), but the requirement is that the already running instances should get updated according to the new process definition. The formal verification of this system should guarantee that the process executed

[75] G. Bancerek et al. "Mizar: State-of-the-art and Beyond". In: *Intl. Conf. Intelligent Computer Mathematics (CICM 2015)*. Vol. 9150. LNCS. 2015.

[76] L. Lamport. *Specifying Systems*. Addison-Wesley, 2002.

[77] Lamport, see n. 76.

[78] C. Newcombe et al. "How Amazon web services uses formal methods". In: *CACM* 58.4 (2015), pp. 66–73.

[79] Y. Yu, P. Manolios, and L. Lamport. "Model checking TLA+ Specifications". In: *Correct Hardware Design and Verification Methods (CHARME'99)*. Vol. 1703. LNCS. 1999.

[80] D. Cousineau et al. "TLA$^+$ Proofs". In: *18th Intl. Symp. Formal Methods (FM 2012)*. Vol. 7436. LNCS. 2012.

by the particular MED-ELs user, is never out of sync, meaning, it does not contain inconsistencies in data presented and generated by the user.

PVS[81] is a specification system and proof assistant that is based on Simple Type Theory, extended with a rich type system including predicate sub-typing. This feature is used extensively in PVS, for example for defining a hierarchy of numerical types. Predicate sub-typing makes type checking in PVS undecidable, and PVS emits corresponding proof obligations in the form of type checking constraints that require interactive proof by the user. For proof checking, PVS provides a rich set of automated tactics and back-end provers but does not generate explicit proof objects.

PVS has been used in numerous academic and industrial projects for verifying hardware systems, fault-tolerant algorithms or dynamically linked Java programmes, among others. It is particularly known for its applications to avionics systems and airborne conflict detection and avoidance systems[82].

Our objective in this project is the translation of a substantial part of the PVS Prelude (the standard library of PVS) into Logipedia. A core version of the PVS logic, restricted to a fragment with decidable type checking, has been defined in Dedukti[83]. The main challenges for achieving our objective are to effectively extend this fragment to full PVS, in a way that identifies superficially different proofs of the inhabitedness of sub-types, and to instrument PVS in order to generate proof traces that can be checked by Dedukti.

Minlog is based on the Curry-Howard correspondence between constructive proofs in natural deduction and terms in typed λ-calculus. It supports extracting programmes from constructive proofs, and has also been extended for extracting programmes from proofs in classical logic. Minlog supports function definitions by computation and computation rules and identifies terms with a common reduct. Its main applications are in constructive analysis[84]: it can extract computational content from a proof of a mathematical statement in an extension of Gödel's system T and generate a formal proof that the extracted term realises the statement, viewed as a specification.

Real numbers are best represented as streams of (signed) digits, and handling such infinite objects requires co-induction, which gives rise to co-recursion in the extracted terms. Minlog accommodates co-recursion via partial functionals in the Scott-Ershov model $\mathcal{C}$ and restricts variables to extensional objects of type level at most 1, where extensionality and totality coincide.

Besides extracting programmes from proofs in constructive analysis, other notable applications of Minlog include the unification of λ-calculus and combinatory logic, embedding classical logic in minimal implicational logic, and the extraction of the DPLL algorithm underlying propositional satisfiability solving from its proof.

In this project, we aim to lift Minlog from LIL 0 to (at least) LIL 3. This is facilitated by the fact that the foundations of Minlog and Dedukti are rather close (both have proof terms and Deduction modulo theory); the main challenges are support for co-induction and co-recursion. We plan to support programme extraction to Dedukti through realisability, and also aim to be able to import a subset of Dedukti proofs into Minlog in order to benefit from the libraries available in Logipedia.

[81] S. Owre, J. M. Rushby, and N. Shankar. "PVS: A Prototype Verification System". In: *11th Intl. Conf. Automated Deduction (CADE-11)*. Vol. 607. LNCS. 1992.

[82] C. A. Muñoz, A. Narkawicz, and A. Dutle. "From Formal Requirements to Highly Assured Software for Unmanned Aircraft Systems". In: *22nd Intl. Symp. Formal Methods (FM 2018)*. Vol. 10951. LNCS. 2018.

[83] F. Gilbert. "Extending higher-order logic with predicate subtyping: application to PVS". PhD thesis. Université Sorbonne Paris Cité, 2018.

[84] K. Miyamoto and H. Schwichtenberg. "Program extraction in exact real arithmetic". In: *Mathematical Structures in Computer Science* 25 (2015), pp. 1692–1704.

ProvenTools. ProvenTools is an interactive environment, developed at Prove & Run, for designing programmes and proofs. Programmes and specifications are written in Smart, a functional language that emphasises a separation between control flow and data flow and brings a uniform monadic style to handling exceptions and different return cases. ProvenTools can also be used to generate efficient C code from a subset of the Smart language, and the tool performs static analysis to ensure that the extraction is safe and correct.

ProvenTools has been used to design, develop, and formally verify ProvenCore[85], a general-purpose micro-kernel that has been certified at the highest Common Criteria level (EAL7). The proofs represent about 400.000 lines of code of Smart, covering all process management including page tables, shared memory, process creation, code loading, etc. ProvenTools has also been used to develop user-side applications, in particular in the case of filter applications[86]. The proofs ensure that the data exchanged conforms with the intended protocol, guaranteeing that no agent attacks another one by producing ill-formed messages.

The logic underlying ProvenTools is similar to that of other programme verification environments such as Why3. The main challenge in translating Smart models and proofs to Dedukti is that ProvenTools does not manipulate formulas or proof terms explicitly. Instead, models are translated to an intermediate format called Smil that resembles control flow graphs with sink states representing erroneous behaviour. Proof obligations are generated in the form of virtual execution paths that must be proved infeasible. A further challenge resides in the scalability of an encoding of models and proofs to the level of industrially relevant applications. An ultimate benchmark would be to be able to handle the standard library of ProvenTools. A main benefit of encoding ProvenTools in Dedukti will be to enable cross-checking proofs using Dedukti. This would be very valuable in the context of certified developments.

K Prover is an implementation of Matching Logic[87], designed as a unifying foundational logic for specifying and verifying programming languages, extending predicate modal logic by features such as many-sorted universes and least fixpoints.

Matching Logic designates a family of logics that underlie a framework for describing and reasoning about both static structures and dynamic features by means of patterns and pattern matching. A language definition in Matching Logic presents both the operational and the axiomatic aspects in a uniform way, with the semantics of the language expressed as a set of computation rules, which are special patterns in Matching Logic. A language-independent proof system can then be used to derive both the operational behaviour and formal properties of a programme.

The K framework allows the designer of a programming languages to specify its formal semantics and to derive analysis tools for the language in a correct-by-construction manner. K specifications (which are Matching Logic theories) exist for a variety of languages including C, Java, JavaScript, and the Ethereum Virtual Machine. The K prover[88] supports coinductive proofs of reachability properties that underlie the correctness of programmes defined using K. It is based on Reachability Logic, formalised in Matching Logic.

Challenges of encoding Matching Logic and proofs generated by the K prover in Dedukti include the translation of the Matching Logic theory expressing the definition of a particular language (where the Matching Logic axioms are patterns expressing transition relations), the translation of the Matching Logic proof system, including its coinductive reachability rules, and the certification of the automatic theorem provers (such as SMT solvers) that the system relies upon. Our objective in this project is to

[85] S. Lescuyer. "ProvenCore: Towards a Verified Isolation Micro-Kernel". In: *International Workshop on MILS: Architecture and Assurance for Secure Systems, MILS@HiPEAC 2015*. 2015.

[86] D. Bolignano and F. Plateau. "Security Filters for IoT Domain Isolation". In: *8th Intl. Symp. Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2018)*. Vol. 11247. LNCS. 2018.

[87] G. Ro su. "Matching Logic". In: *Logical Methods in Computer Science* 13.4 (2017), pp. 1–61.

[88] A. Stefanescu et al. "Semantics-Based Program Verifiers for All Languages". In: *ACM SIGPLAN Intl. Conf. Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2016)*. 2016.

have translated into Dedukti the semantics of (at least) a simple programming language such as IMP, the proof system for reachability properties and a few proofs of paradigmatic programme properties.

Homotopy and cubical type theory (HoTT). Introduced by Vladimir Voevodsky, Michael Warren and others, HoTT aims at connecting type theory and homotopy theory, the study of abstract shapes. It is considered as an alternative to set theory for providing foundations for all areas of mathematics. In particular, paths and homotopies are primitive objects rather than mere sets of points, and isomorphic mathematical structures cannot be discriminated—unlike set theory where accidental identifications can arise depending on how structures are represented.

In its simplest form, HoTT is an extension of Martin-Löf type theory by the univalence axiom. More modern forms include Cubical Type Theory (CubTT[89]) that is both constructive (that is, proving the existence of an object provides an algorithm for effectively computing the object) and where univalence is a theorem. All type-theoretic proof systems have very small steps that are omitted in proofs (for example, trivial equalities such as $1 + 1 = 2$). In the context of HoTT, these are called coherence conditions. Two-level type theory (2LTT[90]) was introduced as a stepping stone where the small proof steps are made explicit, and we will start by expressing 2LTT in Dedukti and then move to CubTT. This will allow us to gain experience with the extent to which coherence conditions can be supported as rewriting steps in Dedukti, and which conditions require proof elaboration.

Challenges

As explained above, each of the systems considered poses some specific challenges that will be addressed in the tasks corresponding to the individual systems. We list some cross-cutting research challenges that this project will allow us to address synergistically.

Rich syntactic and semantic features. Several systems provide support for representing and reasoning about binders, definitional constructions, or have rich type systems that are not easily encoded in Dedukti. Although the details vary between the systems, there are commonalities that should be addressed jointly. For example, K Prover and Minlog have strong support for co-induction and co-recursion for which foundational support should be provided in Dedukti. Mizar and (to a lesser degree) TLA$^+$ make use of soft types for assisting proof search, and PVS emphasises predicate sub-typing, which is also used in the SMT encoding used in TLA$^+$.

Set-theoretic vs. type-theoretic foundations. HoTT and Minlog are based on type theories with explicit proof objects and are in this respect close to Dedukti, although the details differ. K Prover and PVS have a type system reminiscent of Simple Type Theory but add features such as modal logic or predicate sub-typing, whereas Mizar and TLA$^+$ are based on classical set theories. All systems except HoTT and Minlog do not provide proof objects and require instrumentation for providing traces that can then be interpreted by Dedukti. In all cases, a crucial point to consider for defining an encoding in Dedukti, is the exploitation of Dedukti's computation capabilities to control the size of the proof objects.

Automatic theorem provers. Many of the systems that we consider call upon automatic back-ends for proof search and in many cases, these do not natively produce traces. Even when they do, they are not in a form that is suited for replay in Dedukti. There are two ways how we will try to tackle the problem of missing proofs. First, we will attempt to instrument these tools in order to record and export proof traces, and there are clear opportunities for sharing notions of proof traces and concrete proof formats

[89] C. Cohen et al. "Cubical Type Theory: A Constructive Interpretation of the Univalence Axiom". In: *IfCoLog J. of Logic and its Applications* 4.10 (2017), pp. 3127–3170.

[90] D. Annenkov, P. Capriotti, and N. Kraus. "Two-Level Type Theory and Applications". In: *CoRR* abs/1705.03307 (2017).

in order to avoid duplicate effort. Second, we will use native Dedukti automation to re-create the proof parts that are still missing. The research to be carried out here benefits greatly from the network with project partners who consider the integration of automatic theorem provers in Dedukti and Logipedia.

The main measure of success for this work is provided by the Logipedia integration levels of the considered systems, and the fragments for which theories and proofs can be exported. A secondary measure is the overhead incurred by the encodings and instrumentations and thus the complexity of proof checking by Dedukti.

Proof engineering

The development of formal reasoning tools has led to an ever-growing corpus of mechanised mathematical theories. As these libraries span a wide spectrum of proof assistants, the underlying logical foundations of these systems and libraries can differ greatly. For example, some proof assistants use set theory as a foundation while others use Simple type theory[91], Martin-Löf's type theory[92], the Calculus of Constructions[93], etc. In order to combine the power of different proof assistants and their respective communities, it is critically important that these different logical foundations can be related to each other in ways that ensure their interoperability. However, using a common logical foundations is just a starting step. Even when one has successfully related these different logical foundations, one must also align the many different mathematical concepts (for example, sets, numbers, functions) and theorems that are in the libraries produced by the various proof assistants. The goal of proof engineering is to archive a successful interoperability of both proof assistants and formalised mathematical libraries. This starts with identifying the correspondences between the concepts underpinning these formalisations, that is their alignment. Even when the various proof assistants are all exporting their proofs in Dedukti, the proper alignment of the theories underpinning those systems is an essential task: for example, simply combining all the different theories without adaptations can create an inconsistent and, therefore, useless combination. Alignments can also be useful for joining formalisations done within a single prover, since different formal libraries developed within a single system can introduce different definitions of the same mathematical concept (each more suitable to the library where it is introduced).

In the simplest case, an alignment is a pair of symbol identifiers from two different libraries such that both symbols are "the same informal mathematical concept". For example, although real numbers can be defined by Cauchy sequences, Dedekind cuts, and also stream representations using co-recursion, all such different formalisations introduce essentially the same mathematical concept. More precisely, following previous work[94], we consider an alignment to be a pair of:

- two identifiers c, c' (usually from two different libraries L, L'), which are considered to represent the same mathematical concept,
- additional data that governs how L-expressions using c and L'-expressions using c' correspond to each other.

The source of alignments can vary and include manual collections[95] and automatic, machine-learned collections[96].

To explain the need for concept alignment, we can compare this aspect of the Logipedia project to the virtualisation of computer operating systems. In such virtualised systems, it is not sufficient for the user to be able to run system A inside system B, as she also needs to have access to some device of the

[91] Church, see n. 10.

[92] Martin-Löf, see n. 11.

[93] Coquand and Huet, see n. 12.

[94] D. Müller et al. "Classification of Alignments between Concepts of Formal Mathematical Systems". In: *Intelligent Computer Mathematics*. Ed. by H. Geuvers et al. Springer, 2017, pp. 83–98

P. Dehaye et al. "Interoperability in the OpenDreamKit Project: The Math-in-the-Middle Approach". In: *Intelligent Computer Mathematics - 9th International Conference, CICM 2016*. Vol. 9791. LNCS. 2016.

[95] D. Müller et al. "Alignment-based Translations Across Formal Systems Using Interface Theories". In: *Proof eXchange for Theorem Proving*. Ed. by C. Dubois and B. Woltzenlogel Paleo. Open Publishing Association, 2017, pp. 77–93.

[96] T. Gauthier and C. Kaliszyk. "Aligning concepts across proof assistant libraries". In: *Journal of Symbolic Computation* 90 (2019), pp. 89–123.

host system from within the guest system. For example, all operating systems have the concept of file systems, keyboards, mice, etc, but they are all using technically different definitions of these concepts. For virtualisation to succeed, significant work is required to build bridges that formally connect the different versions of these concepts.

Our goals are first to understand the concept of alignment, then to identify alignments across the various proof assistants, and finally to build tools to construct and use these alignments. We will develop standards, tools, and techniques that will allow concepts to be aligned so that formal proof developments from one proof assistant can be meaningfully used in other systems. For example, fundamental properties of natural numbers proved for one definition of natural numbers, must be transported to any isomorphic structure, regardless of the way it has been defined.

Proof engineering happens at different levels: logical foundations, concepts such as types, constants, and theorems, and proofs and proof techniques such as tactics and inference rules. We will primarily focus our efforts on the alignment of logical foundations and concepts.

Logical foundations. Just like Euclidean and non-Euclidean geometries are not completely independent theories but differ by just a few axioms, the various type theories or set theories implemented in the proof systems have some axioms in common. We want to divide these theories in clusters, according to their key parameters, and then to build a web of syntactic translations between systems. We do not expect full back-and-forth translations, as some systems are well-known to be proof-theoretically stronger than others, but we will seek to establish suitable equiprovability results for fragments of the relevant languages.

In order to align logical foundations we will develop novel ways of relating theories represented in Dedukti, building on our extensive experience in type theory, computer-assisted formalisation, proof theory and reverse mathematics. We will articulate our work according to the following key parameters for distinguishing between theories: classical vs. constructive logic, predicative vs. impredicative definitions, treatments of equality, and presence and absence of certain additional axioms, for example the axiom of choice.

One of the major goals will be the formulation and representation in Dedukti of a core logical system that is both common to all logical systems and supportive of modular analysis of its extensions. We believe that such a logical system can be obtained as a type-theoretic version of geometric logic, a certain well-behaved fragment of predicate logic. Such a treatment of geometric logic will be of interest because we believe that, over theories in such a logic, we can transform classical proofs, using the law of excluded middle and the axiom of choice, into constructive proofs. This transformation will not only be a theoretical result but also have practical value since we will seek to obtain feasible algorithms to transform proofs based on classical principles to constructive proofs. This effort will add a new dimension to a long-standing development of proof theory that is still very much unexplored.

A further reason for the interest in this type theory is that it could provide, with a sequence of extensions, a new setting for the development of reverse mathematics in a type-theoretic setting. This setting is in contrast to the standard reverse mathematics developed in the context of second-order arithmetic. Again, the presence of proof terms (expressed using Dedukti) adds a new dimension to a well-established topic.

Concept alignment (case studies). Since sets, functions, relations and numbers are ubiquitous in all formalised mathematics, we will pay special attention to their alignment. Additionally, geometry is a very interesting case study, since it is traditionally introduced in many quite different ways (both analytically, using various number fields, and synthetically, using different axiom systems), and we shall pay special attention to aligning different existing formalisations of geometry.

There are several ways to define numbers. For the natural numbers, one can, for example, choose a definition based on Peano's axioms which relies on a unary representation of numbers which is suitable for proving theorems, but for computing, a definition based on a binary representation is better, and

these different definitions can be aligned[97]. For real numbers and functions, their definitions often differ even between different libraries of the same proof assistant. Real numbers, for example, can be defined using Bishop style modulated Cauchy sequences, regular Cauchy sequences, Dedekind cuts, and stream representations using co-recursion.

Similarly, geometry can be defined synthetically using Euclid's, Hilbert's, or Tarski's axioms (as in GeoCoq library) or analytically using coordinate systems and algebra. Therefore, our first step will be to import into Dedukti the equivalence theorems between these different axiomatisations. In practice, we want to use a theorem $\Gamma \vdash_S T$ expressed in system S based on some axioms Γ in a system S' based on different axioms Γ' and different definitions.

Automated proof engineering. Although alignments can be manually discovered and described, the sheer size of existing formal libraries forces us to develop automated support for proof engineering.

We propose a workflow based on database and semantic web methodologies. Specifically, we aim at building an automatic alignment engine on top of an ontology framework that defines mappings between base concepts belonging to different theories. Relying on a graph of dependencies that indicates the structure of the theories within a given library, we set to propagate the alignments with the help of a certified matching-based engine. The engine will provide a way of inferring new alignments, based on the alignment of the primitive concepts and on a set of rules that will describe further derivations. In order to enable flexible alignments, we are prepared to allow not only exact matches, but also matches with respect to a similarity measure, as in[98].

An alternative method for discovering ontologies of alignments is to use unsupervised learning methods to find alignments between proof assistant repositories. For this, heuristic algorithms and dynamic processes have been tried in the past[99]. As part of the project we intend to also use unsupervised machine translation algorithms[100] to directly find correspondences between statements and their constituent constants and types in proof assistant libraries imported in Logipedia.

Proof engineering services. Each service implemented on top of Dedukti—such as accessing a library via browsing and searching or interoperability and reuse provided by translations across libraries—will benefit from working up-to-alignments. For example, a user of system L can go to Logipedia to find a theorem about t' about concept c', defined in L' that is missing in the library of L. Even if L contains the corresponding concept c, because t' is stated and proved in the logic of L', it will use a different definition of c' in L' than the one of c in L. Therefore, t' cannot be directly used in L. By translating t' along the alignment (c, c'), it induces a theorem t that can be used in L. The details are subtle both in theory and in practice, and this task will explore how to scale up the alignments to provide strong services.

Implementation of services will leverage this simple idea in multiple different ways. Firstly, we will use alignments to search across libraries. In its simplest form, this involves a simple search interface into which the user enters the term t and the system finds t' because it uses the alignment $\langle c, c' \rangle$. Secondly, we will develop translation services that use alignments to transport proofs across libraries. This translation will allow transporting the proof of t' in L' to a (potential) proof of t in L.

Challenges

Often there are many notions that are tightly connected but not equivalent. For example, some definitions are only special cases of others, some definitions are valid in any dimension and some are valid only in some specific dimension, and some definitions differ only in the treatment of certain corner cases (for example, Hilbert's axioms of geometry use the strict betweenness while Tarski's axioms use the

[97] N. Magaud. "Changing Data Representation within the Coq System". In: *Theorem Proving in Higher Order Logics*. 2003.

[98] Gauthier and Kaliszyk, see n. 96.

[99] Gauthier and Kaliszyk, see n. 96.

[100] G. Lample et al. "Phrase-Based & Neural Unsupervised Machine Translation". In: *CoRR* abs/1804.07755 (2018).

non-strict betweenness relation). Although alignments of such definitions are also needed, they are very hard to detect and use. An automatic prover would fail to formally relate such a pair of concepts since they are not equivalent: all the same, such a pair can usually be related by an equivalence "up-to" some special cases. While some equivalences are trivial, others equivalences cannot be expected to be proved automatically.

Contribution

The alignment on concepts between large libraries of formal proofs has never been attempted before. Doing so requires a joint research activity of a large number of European researchers. Alignment based services will significantly facilitate access to the available body of formalised mathematical and computer science knowledge.

One of the final products of this work is alignment aware services that enable both searching and browsing modulo alignment as well as automated translation of theorem proving objects and proofs. However, before such an ambitious goal can be achieved, many lessons must be learned and several milestones must be achieved such as:

- A very precise definition of alignments should be given and a format (and an ontology) for describing and organising alignments must be described.
- Proof theoretical results relating different logical foundations must be obtained and an efficient algorithm for translating statements from one logical system to another must be implemented.
- Alignment of basic mathematical objects (sets, functions, relations, numbers) must be ensured.
- A method for automated detection of alignments must be devised, implemented and applied on a large corpus of formal libraries.
- A case study of geometry must show that it is possible to align large formal theories developed in different theorem provers, based on different approaches (synthetic vs. analytic).

Synergy with related projects

European projects

Alexandria	Alexandria is an ERC project led by Lawrence Paulson aiming at the creation of a proof development environment for working mathematicians. Synergy between Alexandria and Logipedia is clear as Alexandria is built on top of Isabelle and Isabelle and including Isabelle proofs in Logipedia is one of our main objectives. The participation of Angeliki Koutsoukou-Argyraki to our club of academic users is also a factor facilitating the synergy.
Matryoshka	Matryoshka is an ERC Starting Grant led by Jasmin Blanchette aiming at delivering very high levels of automation to users of proof assistants by fusing and extending two lines of research: automatic and interactive theorem proving. Synergy between Matryoshka and Logipedia is clear as Matryoshka could produce proofs for Logipedia and use Logipedia proofs. The participation of Pascal Fontaine to both projects is also a factor facilitating the synergy.
OpenDreamKit	OpenDreamKit is an European Research Infrastructure project led by Nicolas Thierry which ran from 2015 to 2019 that has built the joint ecosystems of software for mathematics. OpenDreamKit and Logipedia's goal are related but different as OpenDreamKit targets software for mathematics, whereas Logipedia focuses on formal proofs. But both type of objects are used by mathematicians, computer scientists, scientists in general and engineers. The continuity between the two projects is manifested by several common members: Michael Kohlhase, Dennis Müller, Florian Rabe, and Makarius Wenzel.

EUTypes	EUTypes is a COST action led by Herman Geuvers coordinating research on type theory and its applications in computer science. It promotes a strong synergy between computer scientists, logicians and mathematicians. Synergy between Types and Logipedia is clear, as six partners of Logipedia are related to Types. But our focus is at the same time more general and narrower. We do not focus on type theory, but we also want to include in Logipedia proofs in set theory, non constructive proofs, etc. On the other hand, as our goal is to build an infrastructure, all our project is oriented towards this goal, while EUTypes gathers researchers with more diverse research directions.

Other projects (not directly funded by the European Union)

DeepSpec	DeepSpec is a project led by Andrew Appel aiming at eliminating software bugs that can lead to security vulnerabilities and computing errors by improving the formal methods by which software is developed and verified. Although DeepSpec and Logipedia pursue different goals, the DeepSpec library can be an interesting benchmark for Logipedia's project to express proofs in different theories and proof systems.
Formal Abstracts	Formal Abstracts is a project lead by Thomas Hales, aiming at formalisation of the main results (constructions, definitions, proofs, conjectures) of a piece of informal mathematics, such as a research paper. Proofs of statements are omitted. Synergy with between Formal Abstracts and Logipedia is clear as Formal Abstracts provide a first step towards formalising all the mathematical knowledge. The two projects are very complementary as Formal abstracts offers to formalise everything except proofs and Logipedia to formalise proofs.
ProofPeer	ProofPeer is a British project led by Jacques Fleuriot that brings together interactive theorem proving technology with the power of the social web, allowing many peers to develop a proof in a cooperative way. Synergy between ProofPeer and Logipedia is clear as ProofPeer could produce proofs for Logipedia and use Logipedia proofs. We also share with ProofPeer — working with Matita, Mizar, Coq, and with any proof system in principle — the will to transcend the use of a single system. The participation of Jacques Fleuriot to our club of academic users is also a factor facilitating the synergy.
Software Heritage	Software Heritage is a project lead by Roberto Di Cosmo and hosted at Inria, aiming at collecting and preserving software in source code form. Software Heritage and Logipedia focus on different objects: programmes and proofs, but they share the same values and the same belief that these objects embody our technical and scientific knowledge and humanity cannot afford the risk of losing it. Also as Software heritage is hosted at Inria, the communication is easier.
StarExec	StarExec is a cross community logic solving service developed at the University of Iowa under the direction of principal investigators Aaron Stump (Iowa), Geoff Sutcliffe (University of Miami), and Cesare Tinelli (Iowa). Its main goal is to facilitate the experimental evaluation of logic solvers. Synergy between StarExec and Logipedia is clear as statements stored in Logipedia can serve as a source of benchmarks for testing logic solvers. The participation of Aaron Stump to the advisory board is a factor facilitating the synergy.

Xena	Xena is a British project, headed by Kevin Buzzard which aims at introducing mathematicians and students to formal proofs. Although Xena and Logipedia use different means, they share the same values, in particular the dissemination of formal proofs. The examples developed in Xena could be an excellent benchmark for Logipedia's project to express elementary proofs in different theories and proof systems. The participation of Kevin Buzzard to our club of academic users and in the club of users in education is also a factor facilitating the synergy.

(b) Methodology

Developing a new research infrastructure that integrates existing ones through data exchange requires a wide range of activities, described in the work-plan. These different activities will require different methodologies: developing new pieces of software, modifying existing ones, proving theorems, experimenting, etc.

We review here the different methodologies used in the project, again objective by objective.

Integration

To integrate existing proof systems into Logipedia, we need to instrument these systems to export their proofs to Dedukti. The methodology required to achieve this goal varies a lot with respect to the proof system considered. With respect to the integration methodology, we distinguish three broad categories of systems:

- Some existing proof systems, the "Automath-like systems", such as Coq, Agda, and Matita, already have a notion of proof-term, that is an explicit representation of proofs. In theory, it is sufficient to translate these terms into the language of Dedukti. As discussed in the Section 1.3, in practice, this may require to reconstruct transient proof components, or to modify the term language of the instrumented system so that these components are included in the proof language itself.
- Some other proof systems, the "LCF-like systems" such as Isabelle/HOL, HOL4, and HOL Light, have a small kernel, that is the only piece of code that needs to be instrumented. This requires some expertise in the instrumented system, but, with this expertise, this instrumentation is doable. In fact some of these systems already have been instrumented and can export proofs expressed in a format called OpenTheory. So, we can reap the benefit of these developments and just translate the OpenTheory proofs to Dedukti.
- A third category of proof systems require a different and more complex methodology. For these systems, such as PVS or Mizar, all that we can extract is a proof-trace, that is a sequence of propositions, such that the transition from one to the next is "easy". To fill the gaps between the propositions of the proof-trace, we thus need to use an external automated theorem prover. In addition, we need to instrument this automated theorem prover itself, so that we can build a complete proof from the trace.

In all these cases, we need to take also into account that these proof systems may rely on external tools to build automatically some parts of the proofs, for instance automated theorem provers, SAT solvers, SMT solvers, etc. In this case the external tool has to be instrumented as well, to that it produces a proof that can be used by the proof system to produce its own proof.

A methodological issue is that all proof systems are actively developed systems that are constantly evolving, even if the theory they implement evolves at a slower pace. In order to avoid having translations that always work for an obsolete version, we need to include these translators to Dedukti to the code of the systems themselves, so that the translations are maintained together with the proof system itself. This is our plan to migrate the translations from Matita, Coq, and HOL Light to the systems themselves, and all the other translations, as soon as they reach the relevant maturity.

Automatic theorem provers

Instrumenting automatic theorem provers, SAT/SMT, predicate solvers, etc. raises different methodological questions, as these systems must be very efficient to be useful. This yields two new issues.

First, these systems include many optimisations that make their code very complex and hence difficult to instrument. Then, instrumentation often slows down the instrumented system slightly and if this is acceptable for interactive proof systems—the user being anyway slower than the software—this is often not acceptable for automated theorem provers. So, these systems must be instrumented in a minimal way, so that their execution is not impacted by the instrumentation. Most of the time, we can just extract a proof trace, as in the third case above, and fill the gaps in these proofs using another, less powerful but better instrumented automated theorem prover.

Some automated theorem provers already output proof traces, meaning that we can see them as black-boxes and concentrate on the translation from these traces to Dedukti. To this end, we plan to extend the prototype tool Ekstrakto to handle more proof formats and increase its expressivity. For the others, a preliminary task is to instrument them. We target proof formats that will be already integrated into Dedukti, in particular TSTP, for automated theorem provers, and LSFC, for SMT solvers.

When an automated theorem prover is called from Dedukti to fill the gap between two propositions, we need also to translate theorem statements expressed in Dedukti, into the logic understood by automatic provers. We will target a pivot language for which translations to major existing automatic provers already exist, such as TIP[101] or the input of Why3[102]. A research challenge is to make this encoding preserve the logical meaning and deal with statements coming from multiple tools. Our approach will be to build the encoding by composing independent "small" encodings that can be activated individually, according to which ones are needed to align the concepts in the various tools with those of Logipedia.

In order to make use of the large amount of theorems stored in Logipedia from automated theorem provers, we need tools to select the knowledge relevant for particular goals. We will look at machine learning methods to select relevant knowledge in the Logipedia corpus as well as to select potential proof term components.

Large libraries

Translating a large library from a given system to Dedukti is almost an experimental science: in principle, if we can translate a small library, we can translate a large one. In practice, things are more complicated. Hence, the first thing we need to do is to experiment with the theories and the translations we do have. Most likely, some efficiency issues will manifest, and we will have to understand where these efficiency issues come from:

- It may be because we are using a computer that is too slow or does not have enough memory.
- It may be because Dedukti itself is too slow and we need to modify Dedukti itself, as we have done in the past to increase, for instance, the efficiency of Dedukti's conversion test.
- It may be because the proof term produced by the translation is too big, as has happened in the past with translations that did not take sharing into account.
- It may be because the expression of the theory itself has to be revised, as has happened in the past with the translation of inductive types.

Depending on the result of this analysis, the computer we use, Dedukti, the translation function, or the theory itself has to be improved, until the full library can be handled.

Infrastructure

Developing an infrastructure to make the proofs more easily accessible raises different methodological questions. Here the technology of web services, distribution package, ergonomic user interfaces, search engines is mature and we need to rely on this existing technology.

Yet this raises two methodological issues. The first is that some of these technologies must be adapted for the case of proofs. This is, for instance, the case of the search engines technology, as plain text search

[101] K. Claessen et al. "TIP: Tons of Inductive Problems". In: *Intelligent Computer Mathematics - International Conference, CICM 2015*. Vol. 9150. LNCS. 2015. ISBN: 978-3-319-20614-1.

[102] Filliâtre and Paskevich, see n. 26.

is not adapted to formal proofs. First because logical formulas are not made of words, themselves made of letters, but directly of symbols. Second because all formulas are, more or less made with the same symbols in different orders—for instance the equal sign is in every equation, so that it does not mean anything to search a formula on the equal sign.

The second aspect which is both methodological and managerial is that these technologies are mastered by other people than researchers and engineers on formal proofs. So we need to develop interdisciplinary teams, which we do by incorporating several partners in the consortium who will bring this expertise.

Structuring the encyclopedia

Structuring the encyclopedia raises the same methodological issues as developing the infrastructure to make the proofs accessible.

The technology of ontologies, for instance, is mature. But is has been developed in different contexts—for instance for medical ontologies—and we need to understand how it can be adapted to formal proofs.

Again, this technology is mastered by other people who are not researchers and engineers on formal proofs. Fortunately, we have succeeded to have in the consortium researchers who will bring their expertise on this topic.

New theories, new systems

Understanding how to express new theories in Dedukti is, from a methodological point of view, closer to mathematics and logic. The main difficulty is to prove the properties (such as confluence, termination, etc.) of the proposed theories and to prove that they are indeed a sound and complete expression in Dedukti of the original theories.

This work can of course benefit from the existing expression of the more mature systems described before: although every system has its idiosyncrasies that require special techniques, there are also many common aspects. In particular, the systems considered here are based on similar foundations (more or less elaborate type theories, set theories, etc.) as those considered above, and ideas that have proved successful there can be transferred *mutatis mutandis*. For instance, issues such as proof-irrelevance, induction, co-induction, etc. are similar across theories and systems.

Yet, an interesting methodological issue, manifests when we attempt to express new theories in Dedukti. In some cases, we do not succeed, but we understand that a minor modification of Dedukti would help. There is here a subtle balance to find between modifying Dedukti each time we want to express a new feature in it, and having a rigid Dedukti that would never evolve. As any logical framework, Dedukti should remain simple enough and neutral with respect the various theories it permits to express, but also flexible enough so that it evolves according to the needs of its users.

Proof engineering

Like the previous themes that also belong to joint research activities, proof engineering first requires some mathematical developments, for instance to establish that proofs in one theory can or cannot always be translated to another theory. But there is also a more empirical aspect in this work: for instance, although in principle not all Matita proofs can be translated to Isabelle/HOL, we have established that the full arithmetic library of Matita could. So it is important to consider specific case studies, such as constructivisation, elimination of universes, elimination of dependent types, elimination of the axiom of choice, etc.

It is important to notice that Logipedia allows bridges to be built between theoretical work and experimental work. For instance, there are many mathematical algorithms, such as constructivisation algorithms or algorithms to eliminate the axiom of choice that have been developed in theoretical papers. **Logipedia gives us a unique opportunity to experiment with these algorithms for translating proofs from a classical to a constructive theory on real size proofs that have been developed for other purposes.**

Another methodological innovation we propose is that of small scale concept alignment. For instance, a small step concept alignment might be to identify that an algebraic structure, called N in one theory, and another, called nat in a different theory, refer to the same object, which means that these structures are isomorphic. Before proving this isomorphism, either by hand or automatically, we need to discover such potential isomorphisms. For this task, heuristics and machine learning seem to be the right methods. So this work on Logipedia is also part of a current to apply machine learning in automated theorem proving and in formal proof technology. In some sense, building a large encyclopedia paves the way to consider proofs as big data and apply statistical methods to proofs, including machine learning.

Finally, to conduct the project as a whole, the critical feed back from the users is essential to have "falsify" wrong directions. This is why the four club of users are a key element to our methodology.

1.4 Ambition

Progress beyond state of the art

Networking activities	In order to foster a culture of cooperation between scientific communities that are often too focused on their own problems, this project will develop several types of networking activities. The initial phase of the project, the development of a common infrastructure, will be the first step in the networking activities of the twenty-eight project partners. Thereafter, collaborations between partners will grow, allowing them to better use the Logipedia infrastructure to meet their needs as well as those of users outside the project. This effort will develop the formal proof community in the countries where it is still in an early stage of development. To structure this growth, several user groups will be created: in industry, in academia, in education, and in publishing. The industrial partners of the project and the **club of industrial users** will strengthen a culture of cooperation between scientific researchers and engineers in academia and in industry. Currently, this cooperation is too often point-to-point (for instance, an enterprise uses one system and develops links to a group of academic researchers developing this system). But we need to develop a more comprehensive culture of cooperation, around common languages and a common research infrastructure. Our action in this direction is twofold. First, some enterprises that already have a strong commitment in formal methods (Clearsy, CEA, Runtime Verification, etc.) are included in the project as partners. Other enterprises that could not be part of the project because of its limited size, or that have a less developed culture in formal methods, are members of the club of industrial users. Among other goals, this club participates to develop a culture of technology watch in formal methods in industry, which is a stepping stone for developing a strong industrial awareness in safety and security in European industry. This club is also only in its initial phases, but we are confident that the initial members of this club will be followed by others. The **club of academic users** will develop a similar culture of technology watch for researchers who are outside formal methods. It gathers colleagues from our community that could not participate to this project, as well as colleagues from outside the formal methods community, in particular mathematicians who are understanding the long term impact formal proofs will have on mathematics.

	The **club of users in education** will also to develop a similar culture of technology watch for teachers, including high school teachers who are investigating how a library of formal proofs can be used with younger students. Here we do not advocate teaching formal proofs to pre-schoolers, but we aim at fostering a culture of rigour in high-school mathematical thinking, by having rigorous statements of theorems in high-school textbooks (including all the corner cases that are often omitted), as well as a clear structure of how each proof of a theorem relies on other theorems. This improved rigour is useful for both the students who will study sciences at University and to those who will not. In particular, in this way we want to modestly contribute to the renewal of the logical thinking culture, which is of prime importance in our "post-truth era". We also believe that an early exposition to formal statements and / or formal proofs may contribute to compensate the gender disparity in our research community. Here also, we must remain modest, as achieving a decent gender-balance will require more than a single action, but is obviously much more effective to try to promote contemporary scientific ideas, and encourage scientific carriers, to women at the high school level rather that at the university level, because at the university level there are already too few women in our auditoriums. The more inceptive **club of users in publishing** will develop networking activities with publishers (Elsevier, Springer, etc. but also, ArXiv, HAL, Wikipedia, etc.) such that formal proofs mentioned in research papers and in other encyclopedia can be permanently made accessible, in Logipedia, while they currently often are just made accessible on the web page of the authors. These four clubs of users: industrial users, academic users, users in education, and users in publishing, together with the partners of the project, will also prepare a larger community that will develop Logipedia beyond the end of the project, in order to achieve our ultimate goal of making Logipedia contain all the formal proofs then developed in twenty years. Finally, this project also prepares the exploitation of Logipedia beyond the duration of the project. Of course, we also plan to organise other networking activities, such as a yearly conference with associated workshops on applications in industry, research, education, and publishing, a general assembly where the research directions can be discussed collectively. We also plan to organise summer schools, especially at the beginning of the project in order to train the new participants (doctoral students, post-docs, etc.) to the technology developed in Dedukti and Logipedia (see Section Dissemination and Communication.) So developing jointly such an infrastructure will not only reconfigure the research community on formal methods, but also the industrial community on formal methods, and will build bridges with the communities of education and of publishing.
Trans-national access and virtual access	Logipedia will be accessible online. It will therefore be accessible at no cost, and without identification, from every country in Europe and beyond, just like, for instance, Wikipedia is.

	The licence chosen for the Logipedia proofs does not need to be the same for all proofs, because some proofs already have a licence before being imported in Logipedia and, in some cases, this licence must be preserved. Yet, in general we will favour a Creative Commons licence and, in particular, cc-by. Such a licence allows a free access, a findable, accessible, interoperable, and reusable data management. It will contribute to the development of the Open data / Open science / Open innovation philosophy and be in line with European policies encouraging open science. Being a central infrastructure, Logipedia will contribute to strengthen the links within the European research area, that it too scattered: for instance, currently, researchers and engineers often use a system developed in their own country (only a few systems having an international community of users), and the libraries of formal proofs specific to this system. As explained above, our effort on access goes beyond providing a transnational and virtual access, as accessibility depends also on developing an ergonomic web interface, a package distribution system, a search engine, and an ontology of mathematical concepts. The public targeted by these interfaces also has to be taken into account, a secondary school student looking for a theorem in geometry needs a different interface from a engineer looking for the correctness proof of an algorithm.
Joint research activities	The project includes joint research activities that would not be possible without this project of co-building an infrastructure. Logipedia will allow joint research projects between the users of this infrastructure that will be able to develop new proofs together using different systems. Then, Logipedia raises new research problems. Some of them have already been solved in the past and must be implemented jointly. Others are newer. Using a common infrastructure requires to describe the theories implemented in the different systems in a common logical framework. We have already discussed how the expression of geometry, arithmetic, and set theory in predicate logic has enabled a renewal of logic at the end of the 1920's, allowing all the logicians to speak the same language. Here we can expect a similar renewal, fostering new joint research through the sharing, not only of a common infrastructure, but also of a common language. The development of a common encyclopedia, such as Logipedia, also raises completely new problems such as automatic concept alignment, structuring a large body of knowledge, or the development of interfaces. These problems will trigger new cooperation between the partners of the project and beyond.

Innovation potential

Europe is a leader in the adoption of formal methods for industrial use. Yet, the multiplicity of approaches used by the different actors impedes the broader application of such techniques. Thus, the wide diffusion of scientific innovation requires building bridges between communities. Europe's leadership will be strengthened by Logipedia that will enable cross-fertilisation of results and resources between the different formal methods communities.

This project to integrate the scientific and technological effort around formal proofs in Europe is thus a way to foster the development of formal methods in industry, as the economic spinoffs from the project will benefit the European industry, mainly by reducing the cost of this technology.

First, reusing a proof produced with a particular tool in another, will become possible in a much more systematic way and at a lower cost. In the uncommon case where a proof relies on a theory which is not

compatible with the target tool, it will be easier to understand why and determine whether adapting it is manageable. Furthermore, as an infrastructure, Logipedia will help users in finding existing proofs of properties, making their verification process faster.

Second, checking a proof will remain possible over time. Today, it requires either to maintain proofs along new versions of the tools, which can represent a significant maintenance cost, or to archive them together with a version of the tool used to produce it. In this situation, Logipedia will help on two aspects. First, the common format will guarantee that proofs can be checked by any tool implementing it, thus reducing proof maintenance cost. Second, by providing a common proof database, general interest proofs can be stored and maintained in the infrastructure, allowing industrial users to focus their resources for their specific needs only.

Finally, Logipedia will increase the trust in formal proofs. For example, in a certification context, the use of a particular tool for the verification of the candidate system must be approved. If the certification body is not familiar with the tool, producing a justification for it can represent a significant amount of time. For a new tool, adoption will be made faster, as not only certification bodies but also potential users could question its soundness despite the potential advantages it could provide. The common, co-built, format offered by Logipedia will provide an answer to this lack of trust. It will help the certification bodies, as they will not have to learn about a new tool for each new certification process, as any tool implementing the format will be suitable for them to check the proof. Thus they only have to trust one. Consequently, it will also help industrial users, as justifying the use of a particular tool will be easier as long as they can provide a proof in the right format. And finally, it will encourage the development of new tools, as potential users will not have to trust these tools directly, but only the proofs that they provide, which can be checked easily.

Each industrial partner in the project of course has their own agenda.

CEA	CEA has been developing Frama-C since 2005, which is used for the analysis and the verification of safety critical applications, which ranges from aeroplanes to nuclear plants or railway systems. In these domains, the confidence into the proofs produced by the analysis tools is an important concern. For deductive verification of C code, Frama-C relies on a complex proof architecture that first simplifies the properties to prove, and then transmit them to the Why3 tool, which will itself perform other simplifications before transmitting them to different SMT solvers. All these steps, while bringing the benefit to get the best of each tool, comes with a risk of error during simplifications, or inconsistency when passing from a tool to another, thus questioning the confidence one can have into the obtained proof. Furthermore, it is common that lemmas developed for a particular use cases are useful in many other use cases. Logipedia is thus an opportunity to create a common database of lemmas that could be reused by the different users of Frama-C. Logipedia will foster the innovation potential of CEA on three aspects: ► Frama-C will be able to gather the proof traces produced by the tools it relies on, and the traces produced by its own simplification process. Thus enabling the possibility to check the complete traces, increasing the confidence into the verification. ► The availability of these traces will also benefit to the validation process of the Frama-C tool itself, using Dedukti as a cross validator of the tests results. ► Sharing of lemmas through Logipedia will benefit to the users of both Frama-C and other tools, avoiding to develop proofs for already known results, thus making their proof process more efficient.

Clearsy	In B software development, each industrial user enriches Atelier B's provers with lemmas which must be qualified by tools or proof-reading. It is often the case that lemmas are qualified multiple times, even within the same company. Clearsy has been involved for years in projects to allow the integration into AtelierB of proof tools developed by other industrial or academic teams (Bware, LCHIP, DISCONT) using modern proof techniques. Eventually, industrial integration raises issues related to the qualification of the tools in relation to industrial safety standards (Common Critera, DO-178, 50129, etc). Logipedia is a unique opportunity for Clearsy to consolidate relationships with leading European actors in the research field for machine-based theorem proving. The tools used by Clearsy and its client companies, namely Atelier B and its provers, will benefit directly from the base of lemmas that will be developed and maintained by Logipedia. For example, a typical railway application requires that tens of thousands of proofs are performed. The rate of automation being typically around 80 %, this leaves hundreds if not thousands of proofs that have to be conducted with a highly specialised human interaction. Such interactions sometimes require more than one thousand manual steps and the creation of lemmas that have to be later verified by a third-party expert. Logipedia, by enabling both the application of new proof tools in this context, would both increase the ratio of automatic proof and facilitate the verification of manually introduced lemmas. Industrial development based on the B method should improve significantly by both cost reduction and increased reliability. The following challenges met by Clearsy are directly addressed in Logipedia: ► leverage advances in proof techniques developed in other communities due to the disparity of expressiveness in different languages, ► certifying the results of the tools used for proof (including Clearsy's own tools), ► integration of new technologies in an industrial context demanding qualification with respect to safety standards.
Edukera	Over the years Edukera has experienced an increasing need for formal methods in education. This is due to the emergence of new industrial activities (cryptocurrency and the verification of smart contracts, software dependant autonomous vehicles, etc.). This is why Edukera considers the existence of a solid open-source education-dedicated interface for formal proofs, to be critical to the upcoming industrial ecosystem. Edukera has developed an online educational application to teach mathematics which relies on the use of the formal proof assistant Coq. However, several issues in the design of the current Edukera application prevent a larger user base and need to be solved: developers in formal methods need to create their own theories and teachers must be able to develop their exercises; plus the core Edukera solution should be open-sourced for anyone to use, fix and improve when necessary. The use of Logipedia as the mathematical foundation of the Edukera application can solve these critical issues: indeed Logipedia federates a large scientific community around a unique language, and a large developer community around a unique open-source mathematical framework, which is beyond the financial means and traction of the Edukera company on its own. Students may already access the Edukera application and exercises. Hence, the existence of an open-source version of the Edukera application will not affect the business model which is based on the billing of class administration features (activity reports, homework schedule, connection to the LMS, etc.).

MED-EL	Given strict and specific quality requirements in the health care domain, medical software needs already a different approach to testing and much stricter verification of requirements than a traditional software. Health care software should provide for reliable data exchange, save patients and professionals time and effort on routine procedures, show the stable performance, and securely deal with the sensitive data. Therefore, such a software should be validated from the perspectives of interoperability, usability, performance, and compliance with requirements, risk factors as well as industry regulations. MED-EL considers that a traditional software testing might be still enhanced with methods assuring that under every condition medical software provides desired results. Hence, by adding formal verification to medical software testing toolkit we can considerably increase probability of delivering new functionalities without unexpected behaviours. Such development of formal proofs must be facilitated in the medical software industry by being able to reuse formal proofs developed for other purposes.
OCamlpro	Logipedia will foster the innovation potential of OCamlpro in two directions. ► The development of proof trace outputs for automatic theorem prover, allowing an ever increasing confidence in the results of formal verification is a major milestone toward their usefulness and actual usage in concrete industrial projects. It also helps mathematicians by allowing more automation in their formalisation of mathematics in assistant theorem provers. ► OCamlpro also supports the dissemination of source code thanks to the Opam open-source package manager, already used to distribute packages for the OCaml programming language and the Coq assistant theorem prover. With the addition of Dedukti proofs in this project, this will demonstrate the capacity of opam to be language-agnostic and help disseminate knowledge.
Prove & Run	Prove & Run uses its innovative verification framework ProvenTools to develop highly-secure bricks in order to help its customers resolve the security challenges linked to the large-scale deployment of connected devices and of the Internet of Things. Formal methods are used to ensure the highest security requirements possible, such as CC EAL7. As such, Prove & Run's interest in Logipedia is twofold. ► In order to obtain an EAL7 Common Criteria certificate, formal verification by itself is not enough and a lot of other constraints including documentation, testing and software development hygiene come into play. The formal proofs themselves must be auditable and the underlying framework documented at length to ensure it is used in a consistent fashion. Logipedia would bring the ability to cross-check proofs obtained in a framework with Dedukti, as well as other frameworkds compatible with Dedukti, and thus bring much more confidence in our proofs and streamline the evaluation process. ► When formally verifying low-level bricks of critical systems, one particularly tedious and somewhat error-prone task is to get the models of underlying architecture right. They are hard to design and hard to test against actual hardware. Being able to share such models with other key players, via translation to Dedukti, would improve the confidence in models and speed up the inclusion of new hardware in formal developments.
Runtime Verification	Runtime Verification is a research and development start-up advocating for the industrial adoption of formal programming language design using Matching Logic (ML), a general purpose logic well-suited for programming language specification. Runtime Verification promotes a "one semantics to rule them all" philosophy; that is, the usage of the (testable, executable) operational semantics of a language to derive formal analysis and verification tools for that language. An important aspect of the formal program verification process is the generation of (checkable) proof objects as justification for the verification conditions. Logipedia will foster the innovation potential of Runtime Verification in several directions:

	► As part of Runtime Verification's involvement in the Logipedia project, the infrastructure of Runtime Verification's ML prover will be adapted to allow the generation of proof traces which can be converted into proof objects. ► Program verification crucially depends on SMT solvers for discharging path conditions. The fact that Logipedia aims to incorporate several such SMT solvers and the fact that ML itself will be integrated into Logipedia will allow to combine ML proofs with the SMT solver proofs into a common proof object. Moreover, since many logics / calculi / models, especially those important in the study of programming languages, can be defined as theories in ML, and since these logics are to be incorporated into Logipedia, Dedukti can be used as a vehicle to smoothly integrate proofs from these logics into more complex combined proof objects. ► Integrating ML with Logipedia will allow for the proof objects to be exported and checked in all logics which are part of the project, thus increasing the confidence in the proof itself by allowing one to use their trusted proof infrastructure to explore and check the proofs.

The members of the club of industrial users also focus on various aspects of Logipedia. Many of them, including **Alstom**, **Mitsubishi Electric**, **Nomadic Labs**, **IBM Research**, **Origin Labs**, **RATP**, etc. insist on the importance of sharing formal proofs and reusing existing ones. Others, for instance **Systerel**, are more interested in increasing the trust in formal developments, in particular by allowing an external verification of the proofs built with the automated theorem provers and the SMT solvers. Some others, for instance **Trusted Labs** and **Onera** insist on the transformation of the certification process. **Arm** insists on the importance of having system independent specifications of pieces of software and hardware, **Facebook Artificial Intelligence Research** on the importance of having large databases of proofs to train machine learning algorithms, and **Siemens** on the fact that Logipedia is also a communication tool to remain up to date with the latest developments.

Readiness of the project

To conclude this part on Excellence, let us mention that this idea of building such a standard for proofs has already been investigated in the past, such as in the QED manifesto[103], but has produced limited results.

Our thesis is that, since the times of QED, the situation has radically changed. After thirty years of research, we have 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 designed several logical frameworks, extending predicate logic, to express these theories. And we have developed reverse mathematics algorithms to analyse which axioms and rules are used in each proof and algorithms, such as constructivisation algorithms, to translate proofs from one theory to another.

So, while the QED manifesto was premature, the time is ripe to remember the values it defended: knowledge sharing, safety, security, privacy, open access, education, etc. A boost on formal methods in Europe is needed, at a time when bugs have killed enough people, and threaten the existence of enough enterprises.

We need to strengthen the citizens' confidence in science and technology.

As the technology is ready, let us make it happen all together.

[103] "The QED Manifesto". In: *Automated Deduction - CADE-12, 12th International Conference on Automated Deduction*. Vol. 814. LNCS. 1994.

Section 2

Impact

2.1 Expected impacts

Wider, simplified, and more efficient access	Logipedia will provide universal access to formal proofs. Each user, in the fields of research, industry, education, and publishing will be able to find the formal proofs she needs in the logic she wants, independently of the logic and system in which this proof has been developed. Alignment of isomorphic structures, both across libraries, and within each library, will provide a wider and simplified access. Such a central encyclopedia can foster projects that would make less sense for one specific library. For instance ► indexing mechanisms for mathematical formulas, and search engines that are specialised to query such formulas, ► a structure of mathematical knowledge divided in theories, books, chapters, categories of users (high school students, researchers), that can be inherited from the libraries imported in Logipedia, from concept alignment, and from clustering algorithms in the dependence graph of the encyclopedia, ► ergonomic user interfaces, that allows a navigation in the structure of the encyclopedia, and that can be specialised for some domains (safety, geometry...) or some category of users, ► programmatic access through an Application Programming Interface, and a package distribution system, so that the mathematical knowledge can be used not only by humans, but also by software. An effort will be made to familiarise new researchers and engineers, who want to contribute to Logipedia: an online tutorial to explain how to use the infrastructure, summer schools, etc. **Performance indicators:** • The number of new users on the web increases by 50% every year. • The number of downloads from the server increases by 50% every year. • The number of users of the teaching interface increases by 50% every year. • The number of teachers and courses using Logipedia increases by 50% every year.
New and more advanced research infrastructure	Logipedia is a research infrastructure of a completely new kind, as it integrates proof systems through data sharing. Such a new research infrastructure will, of course, impact research in many ways:

	▶ Computer scientists will be able to prove safety and security properties of the software they develop faster and at a lower cost because they can access to already developed proofs, independently of the system they use. ▶ When building new proof systems, computer scientists will not need to start from scratch, but will be able to start with an already existing database of proofs. ▶ In Automatic Theorem Proving, computer scientists will be able to use already proved theorems as axioms, to enhance the power of their tools. ▶ Computer scientists conducting research on machine learning using mathematical data sets will have access, thanks to Logipedia, to a wider database of formal proofs. ▶ When they are unsure of the correctness of a proof (*cf.* the recent controversy on the abc conjecture), mathematicians will be able to formalise it faster and at a lower cost because they can access to already developed proofs, independently of the system they use. ▶ Mathematicians interested in reverse mathematics (that is the analysis of the minimal means to prove a theorem) will be able to analyse, in an easier way, the axioms used in each proof. ▶ Scientists, who provide access to formal proofs of the results they present (as required or suggested by several conferences and journals, in computer science), will be able to use Logipedia as a universal repository for such proofs, some of which would be available across systems, and these proofs will remain available permanently. ▶ As mathematics and software are ubiquitous in modern science, Logipedia will also have an impact on other sciences. In particular, because proving properties of a piece of software driving a car or piloting an aircraft require to formalise part of the physical world in which this piece of software evolves. ▶ The results will be archived for a long time, increasing the reproducibility of results, both in mathematics and computer science. **Performance indicators:** • The number of proofs in Logipedia is larger than 100 000 at month 12, 200 000 at month 24, 300 000 at month 36, 400 000 at month 48. • The number of proof systems supported by Logipedia is larger than 8 at month 24 and 12 at month 48.
Operators develop synergies	The formal proof community is currently a scattered community, each sub-community being centred around its own system, its own theory, and its own library or libraries. To build a stronger community, where the researchers develop strategies, it is not sufficient to talk to each other, or to organise conferences, but these sub-communities must exchange data and work together on a joint project to exchange those data. Expressing these data in the same encyclopedia will lead the developers of various systems to express the theories they implement in the same logical framework, yielding a better understanding to the similarities and differences between these theories. Developing synergies will induce less work duplication and will increase the efficiency of the community as a whole.

	Working on common projects will not only increase the communication between relatively close communities, such as the Coq and Agda communities, that meet every year at the TYPES conference, but also to more distant communities, such as the TYPES community, the HOL community (that already meets around the OpenTheory standard), the B and TLA$^+$ communities, and the Mizar community. More importantly, this data exchange between researchers and engineers will allow a better cooperation between research and industry and overcome one of the main obstacles to the diffusion of formal proofs in industry. This evolution towards a standard and this restoration of the universality of logical truth will overcome another main obstacles to the diffusion of formal proofs in the community of mathematicians and in 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. Sharing a logical framework will also allow new synergies between the formal proof community and the Automatic Theorem Proving community that currently share the same notion of proof, but do not share the same tools to manipulate them. By providing large data sets of proofs, Logipedia will also allow a better communication with the machine learning community, interested in training algorithms on formal proofs. We have already organised two Logipedia workshops in January 2019 and January 2020, that have proven to be very valuable to develop joint projects and synergies. This project will allow the organisation of wider international events on this topic of sharing formal proofs, in academia, industry, education, and publishing. **Performance indicators:** • 20 % of proofs developed in one system can used in another at month 24 and 40 % of proofs developed in one system can used in another at month 48. • Number of interactive proof systems generating a verifiable certificate is larger than 8 at month 24 and 12 at month 48. • Number of automatic theorem provers generating a verifiable certificate is larger than 4 at month 24 and 6 at month 48. • Activity of the clubs (one workshop a year for each club).
Innovation is fostered through a reinforced partnership with industry	Formal methods are everywhere and mastering formal methods is key to give Europe a competitive advantage in conquering the market of autonomous cars, trains, planes or drones, or the blockchains and cryptocurrencies. However, this adoption of formal methods by the industry faces the following barrier: 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 the industry by avoiding the continuous re-development of elementary proofs again in each system. We will instead benefit from sharing work on formalisation across communities. Seven European enterprises are partners of the project. Others are members of the club of industrial users. From the point of view of the industry, this project has several types of key exploitable results:

► 1. **Cross-verifying proofs.** In the current state of affairs, when a enterprise proves a piece of software correct using a system X, its client, or the certification authorities can check the proof developed by this enterprise, but they can only do so by using the same system X. A barrier, in the current state of affairs, is that they need to trust the system X. Several actors want to be able to check the proofs using an independent proof system, or even a proof system they have developed themselves to check Logipedia proofs.
A side effect of the construction of the Logipedia platform is to incentivise all the proof systems to be able to produce proofs in a common language. Such proofs can then all be checked by Dedukti, and several other systems.
Moreover, the certification authorities can develop their own proof-checker (the development of such a proof-checker takes a few months) so that they do not need to trust anyone apart from themselves.
► 2. **Paving the way towards standardisation.** The lack of standards is currently a major obstacle to the development of formal methods in industry. Although we consider starting a standardisation process to be still premature, this project will allow us to experiment with a common language that we shall improve until we reach the point where it can be proposed as a standard.
► 3. **Ensuring sustainability.** When an industrial project is over, it is sometimes difficult, especially for small enterprises, to keep an archive of their work over a long period of time. In formal methods, most of the projects are a two-stage rocket. The first stage of the rocket contains basic developments that can be shared between the enterprise and its competitors. The second contains developments that are specific to the project and that often contains industrial secrets. Sharing the first stage on a public encyclopedia, while keeping the second secret, contributes to the sustainability of the developments. They can, for instance, be reused decades later, and will not be lost by the enterprise.
► 4. **Fostering interoperability.** Some industrial program verification tools rely only on a single proof system, so any missing feature of its library forces the end-user to prove complicated theorems that probably exist in other proof systems. Logipedia will make it possible to access all standard libraries and proofs coming from all systems. It hence makes the overall process of proving program correctness much less costly. Other industrial program verification tools use several proof systems. The reason for this is that some proof systems are better for some specific application domains, and also because these enterprises hire researchers and engineers that have different cultures and are more efficient using the tools they know. A positive side effect of the construction of the Logipedia platform is that such cross-system developments are made interoperable: proofs developed in one system can be translated into another, and proofs developed in different systems can be combined in Logipedia itself.

The participation of industry to the project is twofold. First, some enterprises with a strong formal proof activity (CEA LIST, Clearsy, Edukera, MED-EL, OCamlPro, Prove&Run, and Runtime Verification) are full partners of the project and contribute to its work packages and tasks to build the infrastructure.

Second, we will build over the project duration time a *club of industrial users*. This club already has 18 members. Their letters of intent are at the end of the document for (Section 6).

Current members of the club of industrial users

Alstom (*Luis-Fernando Meija*), Arm (*Dominic Mulligan*), ClearSy (*David Déharbe*), Edukera (*Benoit Rognier*), Facebook France (*François Charton*), IBM Research (*Louis Mandel*), MED-EL (*Michal Zaremba*), Mitsubishi Electric R&D Centre Europe (*Denis Cousineau*), Nomadic Labs (*Raphaël Cauderlier*), OCamlPro (*Guillaume Bury*), Onera (*Virginie Wiels*), Origin Labs (*Fabrice Le Fessant*), Prove&Run (*Stéphane Lescuyer*), RATP (*Julien Ordioni*), Runtime Verification (*Grigore Rosu*), Siemens (*Danko Ilik*), Systerel (*Laurent Voisin*), TrustedLabs (*Quang-Huy Nguyen*), TrustInSoft (*Benjamin Monate*).

These enterprises are working in the area of transportation, health care, energy, cybersecurity and blockchain. The club will organise a meeting every year where the industrial members will:
▶ learn about the advancement and new features of the infrastructure,
▶ give feedback on the use of the infrastructure,
▶ provide test sets and proofs to include into Logipedia,
▶ propose to the steering committee new features, services or research directions for the project,
▶ participate to the project self-assessment by providing some feedback on the activities of the consortium.

From the point of view of its members, this club is a unique opportunity for watching the current state of technology. Our empirical observations show that many enterprises, working in safety-critical areas, would like to be more involved in the development of formal methods, but that the first step into using such methods is often too costly. This club will offer them a smoother way to get into this technology. Such a club where the industrial users will develop a common culture in formal methods, share experience with other enterprises, and conduct technology watch on a regular basis is an efficient way to disseminate formal methods in the European industry.

Several key results of the project can lead to an industrial exploitation. This is the case of Logipedia and Dedukti themselves, which cannot be marketed as products but that can be used to develop safer software at a lower cost, to simplify simultaneous certification processes in different countries, and to preserve formal developments over time. More specifically, a shared library of mathematical analysis is a Key Exploitable Result, as is a shared specification of the semantics of programming languages, that can be used both for proving properties of programmes and in particular of compilers.

Impact on the transportation industry

For the transportation industry, the railway industry has been historically a important user of mechanical theorem provers to support the design of software sub-systems. For instance, Clearsy, one of our industrial partners, has been applying proof-based development and maintenance of automatic train control software for large European enterprises. More recently, it has expanded its offer to formal systems and software analysis services, supported by formal methods and mechanical proof systems. By promoting an open framework for the combination of proof systems, including independent proof verification, and the constitution of open libraries of mathematical lemmas, Logipedia will produce an ecosystem that will simplify the constitution of safety cases for product qualification as well as reduce the cost of proof-based developments. This will increase competitiveness without comprising safety.

Impact on the health care industry

Given strict and specific quality requirements in the health care domain, medical software already requires a different approach to testing and much stricter verification of requirements than a traditional software. Software, in the field of health care, should provide for reliable data exchange, save patients and professionals time and effort on routine procedures, show the stable performance, and securely deal with sensitive data. Therefore, such a software should be validated from the perspectives of interoperability, usability, performance, and compliance with requirements, risk factors as well as industry regulations. We consider that a traditional software testing might be still enhanced with methods ensuring that under every condition medical software provides desired results. Hence, by adding formal verification to medical software testing toolkit we could considerably increase probability of delivering new functionalities without unexpected behaviours. In particular, the Logipedia approach will allow sharing proofs, which will facilitate adaptation to the medical field.

Impact on the energy industry

Energy, especially nuclear energy, is a one of the key industrial sectors where safety and security are of prime importance. The certification process in the energy industry depends a lot on a given country. Indeed, they do not share common certification practices, unlike in the aeronautic industry. So an enterprise who certified critical software used in a nuclear plant in one country needs to redo the certification in another country using different tools. The tools recognised by one certification authority are different from one another. Logipedia will allow proof and models to be shared and will therefore ease the adaptation for one certification authority to another.

Impact on the blockchain industry

The goal of public blockchains is to remove the need for trusted third parties. Formal proofs, as promoted by the Logipedia project, can be checked independently of any central authority. Thanks to this key aspect of formal proofs, they can be used in blockchain protocols. For example, the Zen protocol[1] uses formal proofs to guarantee resource bounds on the execution of its smart contracts (a smart contract is a program running on a blockchain).

Blockchain protocols require standardisation of data formats but, because the blockchain technology and its applications evolve quickly, keeping these standards as generic as possible is needed to avoid forbidding future yet-unknown use-cases. Formal proof systems are very generic languages but Logipedia pushes genericity even further by promoting a proof format that is powerful enough to encode most (if not all) other proof formats.

Last but not least, the importance of formal methods is acknowledged by a growing proportion of the blockchain community. In the case of the Tezos[2] blockchain for example, applicability of formal methods has even been one of the main advertised features since the start of the project. Parts of the Tezos node software are certified in F* and Coq and various formal verification systems are used or being developed to certify Tezos smart contracts. Because of the decentralised nature of blockchains, software is hard to update in this industry. Moreover, software in the blockchain industry often manages valuable digital assets, evaluating the risk of software bugs and budgeting for formal verification is easier in this industry than in others. Blockchain systems are however complex because they are related to many fields of computer science (cryptography, theory of programming languages, network security, decentralised computing, game theory) so verifying their code usually requires a combination of specialised tools which raise the question of interoperability between these specialised proof systems. Answering this question of interoperability is one of the main purposes of the Logipedia project.

Performance indicators:

- The number of members of the club of industrial users grows by 20% every year.
- The club of industrial users meets once a year.
- Two theses with an academic and an industrial co-advisor are started every year.
- More than two proof systems developed by industry are integrated during the course of the project.

A new generation of researchers is educated	The availability of a formal online encyclopedia that offers, in a single place, the communication, archival and verification of mathematical knowledge will be of prime importance for teachers.

[1] *An Introduction to the Zen Protocol*. 2017.

[2] L. Goodman. *Tezos: A Self-Amending Crypto-Ledger. White paper*. 2014.

PhD students. The project will contribute to the training of a new generation of researchers, through theses and post-doc contracts.

At the university. Education of formal methods in computer science and of formal proofs in mathematics always hits the same obstacle: the need to choose a specific theory or system. This forces the focus on foundational issues, which is in contradiction with the claimed universality of logical truth. By demonstrating that this choice amounts to include or exclude certain axioms and computation rules, Logipedia will bring back this universality to the education of formal methods. Interactive theorem proving is already used in major universities to introduce students to the concept of proof and to teach software foundations. But, currently it cannot be used for courses with significant mathematical content because of the lack of a large standard libraries of formalised results. By collecting the results already available in the different interactive proof systems, Logipedia will ease the adoption to teach formal maths at university level.

In secondary education. The usage of formal proofs in the classroom is also slowed down by the lack of a large and well organised library of results. Logipedia, through a specific interface, can be used to teach mathematics, for instance geometry, in secondary school. It can also be used to write textbooks that promote mathematical rigour by including only theorems that have formal proofs, even if the proofs in the textbook are not presented formally. This is why we have included, in the project, an enterprise that has already experimented the development of software to teach rigorous proof in high school. We also defend that this renewal of logic in secondary education is of prime importance in our "post-truth era". Because several of us are involved in the renewal of teaching mathematics and computer science in secondary education and in the first years of university, we felt the need to include in the project a club of users in education.

Performance indicators:

- Twelve students start a PhD during the project.
- Two summer schools are organised during the course of the project.
- The number of members of the club of users in education grows by 20% every year.
- Each member of the club of users in education organised a experimented in the classroom.
- The number of users of the teaching interface to Logipedia grows by 20% every year.

Current members of the club of users in education

Jeremy Avigad (Carnegie Mellon University), Thibaut Balabonski (Université Paris Sud), Jean-Paul Bodeveix (Université Paul Sabatier), Quentin Bramas (Université de Strasbourg), Kevin Buzzard (Imperial College, UK), Richard Cabassut (Université de Strasbourg), James Davenport (University of Bath), Marc De Falco (Centre international de Valbonne), David Delahaye (Université de Montpellier), Viviane Durand-Guerrier (Université de Montpellie), Séverine Fratani (Université de Provence), Tetsuo Ida (University of Tsukuba), Bartzia Iro (Université de Montpellier), Mathieu Jaume (Sorbonne Université), Magdalena Kobylanski (Université Gustave Eiffel), Zoltán Kovács (Johannes Kepler University, Frédéric Le Roux (Sorbonne Université), Joao Marcos (Universidade Federal do Rio Grande do Norte), Erik Martin-Dorel (Université Paul Sabatier), Patrick Massot (Université Paris-Sud), Antoine Meyer (Université Paris Est), Simon Modeste (Université de Montpellier), Walther Neuper (Graz University of Technology), Paige Randall (North University of Birmingham), Marc Pantel (Université de Toulouse), Vincent Pavan (Aix-Marseille Université), Pedro Quaresma (University of Coimbra), Jean-Baptiste Raclet (Université Paul Sabatier), Vincent Rahli (University of Birmingham, UK), Philippe Richard (Université de Montreal), Damien Rouhling (Université de Strasbourg), Sylvain Salvati (Inria), Gert Smolka (Saarland University), Martin Strecker (Université Paul Sabatier), Pierre-Yves Strub (Ecole Polytechnique), Christine Tasson (Université Paris Diderot), Damien Thomine (Université Paris-Saclay), Karve Vaibhav (University of Illinois), Daniel Violato (University of Brasilia), Théo Zimmermann (Inria).

Closer interaction between a larger number of researchers

We have already discussed lengthily the impact of Logipedia on the community of academic and industrial researchers in formal methods and, more generally, in logic. Logipedia gathers almost all the research groups in formal proofs in Europe, and its long-term ambition is to gather and work with all of them.

But Logipedia can lead to another important evolution, opening the use of formal proofs by a larger group of users, from experts coming from the formal proof community to non experts: mathematicians, researchers in other sciences using mathematics.

We want to insist on this last point: scientific research and education at all levels are concerned with the discovery, verification, communication, archival and usage of mathematical results. These tasks have been supported by physical books, conferences and other means. Everywhere mathematics and computer science are used (in physics, in some parts of biology and social sciences, in engineering, in particular through simulation, etc.) a quest for higher level of rigour will pave the way for a development of formal proofs. But, here also, this development is slowed down by the multiplicity of theories, systems and libraries that make the first step difficult for beginners. A common reference infrastructure should simplify the access of non-specialists to formal methods. For instance, scientists willing to formalise hybrid systems should have access to a good analysis library, whatever system they use.

	Among the communities of researchers, one on which we can have a real impact during the project is the community of mathematicians. Some of them have started using proof systems, and we must take care to their accessibility to the best libraries of basic mathematics, whatever system they use. **Performance indicators:** • A Logipedia conference is organised every year, • The number of member os the club of academic users increases by 20% every year. **Current members of the club of academic users** Jeremy Avigad (Carnegie Mellon University, US), Nicola Botta (PIK Potsdam, DE), Nuria Brede (Universität Potsdam, DE), Achim D. Brucker (University of Exeter, UK), Michael Butler (University of Southampton, UK), Kevin Buzzard (Imperial College, UK), Johan Commelin (University of Freiburg, DE), David Delahaye (Université de Montpellier, FR), Simon Foster (University of York, UK), Georges Gonthier (Inria, FR), Sébastien Gouëzel (Université de Nantes, FR), Angeliki Koutsoukou-Argyraki (Cambridge University, UK), Michael Leuschel (Universität Düsseldorf, DE), Tadeusz Litak (FAU Erlangen-Nürnberg, DE), Patrick Massot (Université Paris-Sud, FR), Assia Mahboubi (Inria, FR), Dale Miller (Inria, FR), Paige Randall North (Ohio State University, US), Karl Palmskog (KTH Royal Institute of Technology, NO), Vincent Rahli (University of Birmingham, UK), Michael Shulman (University of San Diego, US), Gert Smolka (Universität Saarland, DE), Andrew Sogokon (University of Southampton, UK), Martin Suda (Czech Institute of Informatics, CZ), Josef Urban (Czech Institute of Informatics, CZ)
Better management of the continuous flow of scientific data	Logipedia integrates major scientific equipments and knowledge resources (provers and proof libraries). Organizing this continuous flow of data is at the centre of the Logipedia project. We already insisted on the aspects of interoperability, sustainability, cross verification, and ergonomy of the interfaces. The management of this continuous flow of data will be monitored by the metric we have defined: the Logipedia Integration Level. A data management plan will be provided for making these data findable, accessible, interoperable and reusable (FAIR), and Logipedia will participate to the Open Research Data Pilot H2020 project. **Performance indicators:** • The number of proofs in Logipedia is larger than 100 000 at month 12, 200 000 at month 24, 300 000 at month 36, 400 000 at month 48. • Measures have been taken to make these data available after the end of the project.

Integrated and harmonised access to resources facilitating evidence-based policy making	Some certification authorities such as the French National Agency for the Security of Information Systems (ANSSI), Das Bundesamt für Sicherheit in der Informationstechnik (BSI), etc. are using formal proofs for the higher evaluation assurance levels. The ANSSI is interested in having a unique language for formal proofs, rather than several ones which require as many experts as there are languages. Logipedia will be a service to certification authorities, especially in security, as witnessed by the presence of a representative of ANSSI in the advisory board, Prove&Run among the partners, and several other enterprises focused on security in the club of industrial users. Formal methods can also have applications in the verification of legal rules like, for instance, the rules for computing one's income tax[3]. **Performance indicators:** • At least two meetings have been organised with certification agencies.
Socio-economic impact	One aspect of the socio-economic impact of formal methods in general and integration activities such as Logipedia specifically, is that making the digital society safer and more secure contributes to a safer and more secure society in general. It also contributes to a better trust in technology and science. We are also interested in the recent trend to add a third pillar to safety and security: ethics. Software must, of course, be safe and secure, but they must also verify some ethics properties, such as the respect of privacy, or (in the case of electronic voting systems) secret of vote, auditability, etc. The development of formal methods should contribute to prove safety and security properties, but also ethics property. **Performance indicators:** • An open workshop is organised to raise the awareness of ethics committees on the importance of formal proofs for safety, security, and ethics.
Open data, Open science, and Open innovation	Logipedia is clearly part of the Open data, Open science, and Open innovation movement, as it aims at making scientific data accessible to everyone, amateur or professional, in academia, industry, and education. First, the very idea of making formal proofs findable, accessible, interoperable, and reusable, through the construction of an encyclopedia and proof engineering algorithms is *per se* in the movement of Open data, Open science, and Open innovation. Then, the creative common licence, and other free licences, promoted by Logipedia also contribute to this movement. **Performance indicators:** • The platform Logipedia is functional, allowing open access to all the proofs it contains.

[3] D. Merigoux, R. Monat, and C. Gaie. "Étude formelle de l'implémentation du code des impôts". In: *31ème Journées Francophones des Langages Applicatifs*. 2020.

Scientific publishing, proof verification, long-term proof archiving and reproducibility	Published proofs often contain gaps, sometimes errors, and can be difficult to check or complete by humans because of their length or complexity. Frans and Kosolosky noted that, at each level of the publication process, there is a lack of discipline to emphasise the importance of checking and correcting proofs[4]. Given this state of the field, they came up with several ways in which mathematics can and should be improved. One way is related to the reviewing process itself. Another way is to appeal to online databases. Logipedia will provide services to the community of mathematicians and to the community of scientific publishers. Il will provide hyperlinks to Logipedia proof objects, authors can focus on the presentation of their results and help reviewers and editors in evaluating their correctness and interest. Readers can be confident in the correctness of the results as all Logipedia proofs are formally checked. They can also get a better understanding of them by looking at the proofs through some editor or web interface. They can also visualise the axioms and theorems they rely on. Moreover, as Logipedia proofs are represented in a machine-independent format, they can be rechecked and reused as long as they are available, thus improving result verification and reproducibility in mathematical sciences. **Performance indicators:** • The club of users in publishing develops and includes one representative of each major publishing venue. • An action is made towards the steering committees of conferences such as POPL, ITP, FSCD, so that they encourage their authors to sustain their claims with formal proofs permanently accessible in Logipeda.
Providing data to machine learning algorithms	The success of machine learning techniques heavily relies on the availability of big amounts of data. Various enterprises (Wolfram, Facebook, etc.) are interested in applying deep learning techniques to automated theorem proving, symbolic computation, mathematical knowledge, and automatic language translation (between formal languages, or between formal languages and natural languages), with applications in software engineering, education and computer algebra systems. By gathering proofs coming from many different proof systems, in a unique format, Logipedia will be an important source of training data for machine learning algorithms, and more generally for statistical analysis of the properties of mathematical developments. **Performance indicators:** • Scientific papers in the community of machine learning use Logipedia as a dataset.

2.2 Measures to maximise impact

A detailed *Dissemination and communication plan* will be delivered at month 4 (to be updated in the course of the project as necessary). That plan will outline all of the activities to be undertaken, including the identification of partners' participation in international, European and national conferences and events, relevant publication channels and opportunities for collaboration and cooperation with other projects and

[4] J. Frans and L. Kosolosky. "Revisiting the reliability of published mathematical proofs: where do we go next?" In: *THEORIA. An International Journal for Theory, History and Foundations of Science* 29.3 (2014), pp. 345–360. ISSN: 2171-679X.

initiatives. Special attention will be given to the exploitation strategy to align the dissemination objectives with the exploitation goals. The Plan will also:

- increase the awareness of Logipedia in academia,
- use Logipedia as a way to increase the cooperation between academia and industry,
- prepare the sustainability and exploitation of Logipedia before the end of the project, illustrating the philosophy of the 2018 roadmap of the European Strategy Forum on Research Infrastructures (ESFRI): "A robust long-term vision is essential to successfully and sustainably develop, construct and operate Research Infrastructures." even if Logipedia is still in its "incubation phase".

(a) Plan for dissemination and exploitation of results

Dissemination

Logipedia will strictly follow the open access policy of Horizon 2020 by providing online access to scientific information that is free of charge to the end-user and that is reusable. The project will combine different measures to foster open access to knowledge. With reference to Open Data, Logipedia will comply with the requirements of the Open Research Data Pilot fully.

Dissemination of information and knowledge will be achieve through the following means (described in work package 8):

- The participation to conferences.

  In computer science, publishing in conference proceedings are often favoured over journal publication. We have targeted more than ten conferences:

  - CADE (Conference on Automated Deduction)
  - CICM (Intelligent Computer Mathematics)
  - CPP (Certified Programs and Proofs)
  - CSL (Computer Science Logic)
  - FSCD (Formal Structures for Computation and Deduction)
  - FROCOS (Frontiers of Combining Systems)
  - ICALP (International Colloquium on Automata, Languages, and Programming)
  - IJCAR (International Joint Conference on Automated Reasoning)
  - ITP (Interactive Theorem Proving)
  - LFMTP (Logical Frameworks and Meta-Languages: Theory and Practice)
  - LICS (Logic in Computer Science)
  - LPAR (Logic Programming and Automated Reasoning)
  - PxTP (Proof eXchange for Theorem Proving)

- Open Access publications.

- The organisation of our own yearly event, with a conference, parallel specialised workshops, and the general assembly.

  The organisation of a final event to disseminate and transfer the knowledge generated with the aim to enable others to use and take up results, thus maximising the impact of the project. The event will focus on sharing information and gather feedback through a more interactive and dynamic approach to boost the uptake of the Logipedia results.

- The organisation of two summer schools open to anyone and not only the partners. We shall organise several training sessions targeting the different communities of users: master and PhD students to teach them the foundations of Logipedia, teachers to help them use interactive theorem provers at school and university, and to engineers to help them use formal methods tools in their work. Such training sessions are key dissemination events that will accompany the growing of

the Logipedia community and contribute to educate a new generation of researchers, teachers and engineers.

- The supervisation of PhD students to educate a new generation of researchers and engineers, Some will have academic and industrial co-advisors.
- The co-building the Logipedia strategy by participating to joint meetings, such as the clubs and the advisory board.
- The dissemination of Logipedia in in research, in industry, in education, and in publishing. The four clubs contribute to the dissemination of Logipedia in their own ecosystem, by organising talks, courses, meetings.
- A discussion with certification authorities about the use of a common language across the European Union.
- The co-production of textbooks with the members of the club of users in education.
- The increase of reproducibility in science by referencing formal proofs in a single place.
- The use of a free licence for data and software to make the data is findable, accessible, interoperable and reusable and develop Open data, Open science, and Open innovation.

Action	**Target**	**Indicator and schedule**
Participating to conferences	Researchers.	10 papers per year.
Publishing in Open access venues	Scientists and students.	All the publications of the partners are open.
Organising of annual Logipedia conferences	Researchers, industrials.	100 participants each year. Four conferences total.
Organising summer schools and training sessions	Master and PhD students, researchers, engineers.	2 Summer schools. 100 participants total.
Supervising PhD students	PhD students.	3 PhD students start each year.
Co-building the Logipedia strategy	Researchers, industrials.	Participation to the meetings every year.
Disseminating Logipedia in relevant communities	Research, industry, education, publishing.	At least one event organised by each club every year.
Initiating a discussion with certification authorities	Certification agencies and the industry.	Two meetings are organised with several European certification agencies.
Writing textbooks	Under- graduate and secondary education students	At least one textbook during the project.
Using Logipedia to increase reproducibility in science	Publishers and researchers.	Researchers outside the consortium use Logipedia as a reference.
Using a free licence for data and software	Scientists and innovators.	Delivery of Logipedia at month 14.

Exploitation

The objective of the exploitation activities is to to make sure Logipedia remains functional with a healthy community and development roadmap beyond the project. The partners will conduct the following actions.

Action	Stakeholders	Indicator and schedule
1. Build an organisation in charge of managing Logipedia after the end of the project.	The project management team, volunteer members of the consortium, after consulting the advisory board.	The structure is created at month 48 at the latest.
2. Raise funds to manage Logipedia.	The project management team, after consulting the advisory board.	Sponsors and fundings have been identified at month 36 at the latest.
3. Find a server to permanently host Logipedia.	The project management team, volunteer members of the consortium, after consulting the advisory board.	Logipedia is kept functional.
4. Continue developing Logipedia	New developers taking over.	New libraries (for instance that of ACL2 or Nuprl) are integrated. New features are added to Logipedia.
5. Generate new projects, such as "Logipedia, security, and certification", "Logipedia and automated theorem proving", "Logipedia and the B method"...	Special interest groups within Logipedia.	New communities adopt Logipedia.

(b) Communication activities

Communication activities aim at raising the awareness about Logipedia to potential stakeholders that would not be concerned by our dissemination actions. It will ensure the growth of the Logipedia ecosystem and be a way to advertise the work achieved by the partners and the clubs of users. It also serves the purpose of informing the European citizen of the research findings she has been financially contributing to.

Communication activities will be closely linked to dissemination objectives. Six person-months of an experienced communication officer, from the Inria Saclay communication team, are dedicate to the sole task of communication. She will work in close cooperation with the dissemination, communication, and exploitation work package leader and the project management team.

The preparation of a Dissemination and communication plan in line with the project dissemination and communication strategy will be one of the first tasks to be addressed in work package 8. The Plan will identify fundamental elements of the dissemination and communication strategy, including:

- key messages to be conveyed (what),
- tools and channels used (how),
- timing of the planned activities (when),
- geographical level (local, national, European) (where) providing a guide for the project's and partners' dissemination activities to maximise the impact of Logipedia.

In this project, we also have the ambition to communicate to the general public, even if, in the past, these communication activities have been considered as less important than the dissemination towards the research, industry, and education communities.

Action	Target audience	Indicator and schedule
Promoting the existence of the project including defining the project visual identity, creating a web site, publishing on social media, designing flyers, posters, and videos, and publishing press releases.	Research, industry, and education.	Website at month 3.
Promoting the results and the values of the project through outreach actions: publication of articles in popular science magazines, online videos[5], and live events such that the European Researchers Night or *La Fête de la Science*, focusing, as it is our habit.	General public.	At least five partners organise an outreach activity in their country.

[5] such as `https://www.facebook.com/TheatreLaReineBlanche/videos/518698965681202`, a one-minute video hosted by le Théâtre de la Reine Blanche, presenting, in French, the example given at the beginning of this document.

Section 3

Implementation

3.1 Work plan: work packages, deliverables

Overall structure of the work plan

Our work plan is divided into nine work packages. The first group of work packages is dedicated to the networking activities that are needed to gather the proofs today located in different libraries. The second to making these proofs accessible, beyond trans-national and virtual access. The third to joint research activities that prepare the future of Logipedia. Two more work packages are dedicated to dissemination, communication and exploitation and to management.

Networking activities:			
WP1	Integration	Jesper Cockx (Delft)	Instrument the systems for which we already know how to encode the proofs in Dedukti, and make these proofs available in Logipedia.
WP2	Automatic theorem proving	Chantal Keller (Saclay)	Develop automatic theorem provers to populate, help, and benefit from Logipedia.
WP3	Large libraries	Tobias Nipkow (München)	Export large dedicated libraries in curated form to Logipedia for end-user applications.

Trans-national and virtual access:			
WP4	Access	Frédéric Blanqui (Inria)	Define and build the Logipedia hardware and software infrastructure in which the proofs will be integrated.
WP5	Structure of the encyclopedia	Florian Rabe (Erlangen)	Provide infrastructure for the structured ontological representation of libraries and use it to enrich the information about formal libraries in Logipedia.

Joint research activities:			
WP6	Theories	Cezary Kaliszyk (Inssbruck)	Bringing proof systems implementing a theory that has not yet been expressed in Dedukti to LIL 2 or better.
WP7	Proof engineering	Filip Marić (Belgrade)	Investigate methods for detecting concept alignments and apply them to build a library of alignments present across the Logipedia database.

Dissemination, communication, exploitation, and management:			
WP8	Dissemination, communication, and exploitation	Pascal Fontaine (Liège)	Expand the use of Logipedia in research, industry, education, and publishing.
WP9	Management	Gilles Dowek (Inria)	Coordinate this large community, in a benevolent atmosphere, for optimal efficiency.

These work packages are diverse in the number of tasks and partners. Some of them are large, while others are smaller. This reflects the diversity of their natures and goals. The work package "access" for instance has a very definite and critical goal, it must have a small number of partners and be focused on its goal. In contrast, the work package "theories" requires experts in many different systems. It therefore has a larger number of partners.

Detailed work description

All the person-months below include in-kind contributions, detailed in Section 3.4.

Work Package 1: Integration (Networking activity)										**Months:** 1 - 48
Site	**Inria**	**INPT**	**Unibo**	**TUM**	***TUDelft***	**UGot**	**Chalmers**	**IMT**	**ClearSy**	**all**
Effort	6	31	11	5	*16*	4	16	1	14	**104**

Objectives

Bringing systems already at least at LIL 1 to higher LIL levels, that is, instrumenting the systems for which we already know how to encode the proofs in Dedukti, and making available these proofs in Logipedia so that they can be exported to other systems.

Description

The concrete work that has to be performed to integrate a particular system into Logipedia depends on how much of the theory of that system has already been encoded in Dedukti, and how mature the existing (prototype) tools for exporting proofs already are. Thus the tasks in this work package are divided among the different interactive theorem provers we consider: Agda, Isabelle, HOL4, Atelier-B and Rodin, Matita, and Coq.

T1.1 Instrument Agda; Sites: **TUDelft** (lead)
(PM distribution: **TUDelft**: 16, **UGot**: 4)

- Continue the development of the prototype Agda2Dedukti tool to handle core features of Agda such as inductive datatypes and dependent pattern matching in a robust manner.
- Extend Agda2Dedukti with η-equality for record types and definitional irrelevance.
- Extend Agda2Dedukti with first-class universe level polymorphism.

T1.2 Instrument Isabelle; Sites: **TUM** (lead)
(PM distribution: **TUM**: 5)

- Improve the efficiency of important aspects of the Isabelle/HOL logic implementation, such as normalisation of proofs, type-class reasoning, and special representation of derived rules and definition principles.
- Reduce the volume of proof terms in the Dedukti encoding, by taking more structure of the target language (Dedukti) into account and omitting certain low-level reasoning of HOL (e.g. for inductive types).
- Improve memory usage of the Isabelle/ML implementation platform.

T1.3 Instrument HOL4; Sites: **Chalmers** (lead)
(PM distribution: **Chalmers**: 16)
- Develop the current proof-of-concept prototype HOL4-to-OpenTheory and OpenTheory-to-Dedukti tools to the point where they scale (both in terms of time and space) to cope with real examples of interest.

T1.4 Instrument Atelier-B/Rodin; Sites: **IMT** (lead)
(PM distribution: **INPT**: 31, **IMT**: 1, **ClearSy**: 14)
- Continue the encoding of the B set theory in Dedukti to be able to handle all kind of proof obligations in Zenon Modulo.
- Instrument and export B/Event B models to Dedukti.
- Develop a tool for importing lemmas coming from Logipedia into B models.

T1.5 Integrate the Matita translator in Matita itself; Sites: **Unibo** (lead)
(PM distribution: **Unibo**: 5)
- Merge Krajono and Matita, update the code to the latest version and transfer the maintenance effort to the Matita team.
- Extend Krajono in order to record metadata that allow to invert the translation.
- Optimise the code and the translation in order to be able to export all the libraries of Matita.

T1.6 Instrument Coq; Sites: **Unibo** (lead)
(PM distribution: **Inria**: 6, **Unibo**: 6)
- Access Coq internal data structures to gather logical data, such as statements and proof terms (for tasks in WP3).
- Implement the required instrumentation to translate of Coq terms to Dedukti terms as a part of Coq proper.
- Access Coq internal data structures to gather extra-logical data, such as the role played by a constant in the library like begin an implicit cast from one algebraic structure to another (for tasks in WP3).
- Make extra-logical data available in a structured and extensible format (for tasks in WP5).

Deliverables:
D1.1 (Due: 12, Type: OTHER, Dissem.: PU, Lead: TUM) *Export proofs from Isabelle* ⇝**Ms2**
D1.2 (Due: 12, Type: OTHER, Dissem.: PU, Lead: TUM) *Improved memory management and monitoring for Poly/ML*
D1.3 (Due: 16, Type: OTHER, Dissem.: PU, Lead: Chalmers) *Export proofs from HOL4* ⇝**Ms3**
D1.4 (Due: 18, Type: OTHER, Dissem.: PU, Lead: TUDelft) *Export Agda's standard library*
D1.5 (Due: 8, Type: OTHER, Dissem.: PU, Lead: Inria) *Export of proofs from Coq* ⇝**Ms1**
D1.6 (Due: 24, Type: OTHER, Dissem.: PU, Lead: Unibo) *Scalable export of proof terms with meta data from Coq*
D1.7 (Due: 12, Type: OTHER, Dissem.: PU, Lead: Unibo) *Export proofs & metadata from Matita*

D1.8 (Due: 24, Type: DEM, Dissem.: PU, Lead: ClearSy) *A prototype of tool exporting proof terms from Atelier B*
D1.9 (Due: 48, Type: OTHER, Dissem.: PU, Lead: ClearSy) *Harness in Atelier B to export proof terms and import lemmas from Logipedia*
D1.10 (Due: 48, Type: OTHER, Dissem.: PU, Lead: INPT) *Export/import of B/Event B models from/in Rodin*

Work Package 2: Automatic theorem provers (Networking activity)									**Months:** 1 - 48
Site	**UIBK**	**ULiege**	**UBel**	***USaclay***	**IMT**	**OcamlPro**	**CEA**	**DHBW**	**all**
Effort	6	54	12	*11*	9	14	18	2	**126**

Objectives

Using and developing automatic theorem provers to populate, help, and benefit from Logipedia. The focus is networking and research.

- Automatic provers will be instrumented in a two-step process to produce Dedukti statements.
- Dedukti statements will be made understandable and usable by automatic theorem provers.
- Automatic provers will increase Logipedia readiness by automatically filling gaps in proofs.

The final objective is to develop the infrastructure to safely use automation all along the way.

Description

The aim of this work package is to connect automated tools to the Logipedia infrastructure.

The importance of proofs in automated theorem provers, satisfiability modulo theories solvers, propositional satisfiability solvers and model checkers is increasingly recognised. While for the propositional case, the community agrees on a well defined proof format, the situation is not clear for the other kinds 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 concentrate on some provers: the theorem prover for predicate logic E, the SMT solvers veriT and Alt-Ergo, and a coherent logic reasoner, and provide 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 standardised proof format appears for some kind of tools, the conduits will be updated to adopt the new standard.

In this work package, we will also integrate into 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.

Import from Dedukti to automatic provers will be developed, as well as internal automation. Thus automation techniques can be put in practice to produce new results for Logipedia and out of Logipedia.

This work package has the end-to-end objective to build a powerful automated infrastructure: one will 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. A large-scale application about the semi-automatic verification of C code will validate this infrastructure.

T2.1 Instrumenting ATPs to produce traces; Sites: **ULiege** (lead)
(PM distribution: **ULiege**: 31, **UBel**: 6, **IMT**: 1, **OcamlPro**: 12, **DHBW**: 1)
Automated theorem provers should produce detailed proof traces, with sufficient information to be understood in the Logipedia context. We will consider here the theorem prover for predicate logic E, the SMT solvers veriT and Alt-Ergo, and a coherent logic reasoner, and instrument them to produce detailed proof traces. A first prototype will be made available early in the project, to enable work on the next task. The final tools, fully connectable to the Logipedia infrastructure, will be delivered at the end of the project.

T2.2 Translate ATP traces into Dedukti; Sites: **IMT** (lead)
(PM distribution: **ULiege**: 14, **USaclay**: 2, **IMT**: 8, **DHBW**: 1)
This task focuses on the design and implementation of the tool Ekstrakto to translate into Dedukti

the proof traces from the automatic theorem provers from **T2.1**. This task depends on **T2.1**, but can be partly executed in parallel.

T2.3 Translate Dedukti statements into ATPs inputs; Sites: **OcamlPro** (lead)
(PM distribution: **ULiege**: 9, **UBel**: 3, **USaclay**: 3, **OcamlPro**: 2)
To be fully integrated into Logipedia, automatic tools need to understand Dedukti statements. This task is about creating the software infrastructure to translate Logipedia statements into the standardised input format for some automatic tools, namely TPTP and SMT-LIB. This task can be conducted in parallel of tasks **T2.1** and **T2.2**.

T2.4 ATPs to increase Logipedia readiness; Sites: **UIBK** (lead)
(PM distribution: **UIBK**: 6, **UBel**: 3, **USaclay**: 6)
Internal automatic strategies will be defined for Dedukti, based on automatic theorem proving and machine learning. In combination with external ATPs through the results of tasks **T2.1**, **T2.2** and **T2.3**, they will be used to increase Logipedia readiness, e.g. by helping proof checking, by automatically combining proofs, or by discharging alignment concept obligations (in a joint effort with **T7.3** of WP7).

T2.5 Logipedia and ATP end-to-end; Sites: **CEA** (lead)
(PM distribution: **CEA**: 18)
The infrastructure developed in the above tasks will used in the verification *a posteriori* of the Frama-C-WP tool for formal verification of C code and its back-end Why3. In this platform, the correctness of C programs relies on the automatic generation of proof obligations from annotated programs, and their automatic proofs using a combinations of ATPs and user-defined rewrite rules. In this task:

- the proof obligations rewrite engine and the translation of the proof obligations to the ATPs will be instrumented to produce Dedukti proofs, relying on (parts of) Ekstrakto;
- ATPs will produce Dedukti proofs following tasks **T2.1** and **T2.2**;
- users will have access to the Logipedia library following **T2.3**.

In the end, Dedukti proofs will be assembled together in a coherent whole. This task serves as an end-to-end demonstration of the usefulness of the infrastructure.

Deliverables:
D2.1 (Due: 6, Type: DEM, Dissem.: PU, Lead: ULiege) *Prototypes of Trace producing Automatic Theorem Prover and Satisfiability Modulo Theories solver*
D2.2 (Due: 18, Type: DEM, Dissem.: PU, Lead: IMT) *Prototype of a translator for some traces to Dedukti*
D2.3 (Due: 24, Type: DEM, Dissem.: PU, Lead: USaclay) *Prototype of a translator from Dedukti statements into TPTP and SMT-LIB, application to simple libraries*
D2.4 (Due: 36, Type: DEM, Dissem.: PU, Lead: USaclay) *Tools based on ATPs to increase Logipedia Readiness*
D2.5 (Due: 40, Type: OTHER, Dissem.: PU, Lead: ULiege) *Trace producing Automatic Theorem Prover, Satisfiability Modulo Theories solvers, and Coherent Logic Solver*
D2.6 (Due: 48, Type: OTHER, Dissem.: PU, Lead: IMT) *Translate most traces to Dedukti*
D2.7 (Due: 36, Type: OTHER, Dissem.: PU, Lead: USaclay) *Translator from expressive Dedukti statements into TPTP and SMT-LIB, application to large libraries*
D2.8 (Due: 48, Type: R, Dissem.: PU, Lead: UIBK) *Report on Proof Automation in Logipedia*
D2.9 (Due: 48, Type: DEM, Dissem.: PU, Lead: CEA) *A posteriori verification of non trivial C programs in the Frama-C-WP platform*

Work Package 3: Large libraries (Networking activity)					**Months:** 1 - 48
Site	**Inria**	**Unistra**	***TUM***	**Chalmers**	**all**
Effort	20	18	*29*	12	**79**

Objectives

Exporting large dedicated libraries in curated form to Dedukti and thus to Logipedia for end-user applications. The focus is *Access* and *Scalability*.

- This WP is responsible for supplying the lion's share of proofs in Logipedia. As a result it will be a stress test for the results of WP1.
- The target libraries are dedicated to particular application areas. They provide a substantial coverage of that application area and do so in a structured manner. This may require reworking the libraries for better access.
- The libraries are curated for end-user application. That is, they are structured according to application specific ontologies that support browsing and search. The structuring leverages the infrastructures of WP5 and will be a stress test for the results of that WP.

Description

Translating the standard libraries of the systems is part of the WP1. This WP focuses on advanced libraries selected according to the following criteria: relevance, coverage and maturity. As a result we selected the following libraries: the Archive of Formal Proofs, Isabelle's revised Analysis and Probability library, GeoCoq, Flyspeck, and CakeML.

T3.1 Isabelle's Archive of Formal Proofs; Sites: **TUM** (lead)
(PM distribution: **TUM**: 3)
Isabelle's Archive of Formal Proofs (AFP) is a growing user-contributed online library for Isabelle. In Feb-2020, the AFP consisted of more than 500 entries (articles of formalised mathematics) by 340 authors, and required approx. 60h CPU time for checking (using many gigabytes of memory). The purpose of this task is to scale up the Isabelle instrumentation for Dedukti further, to cover major parts of this library. The ultimate aim is to export the main substance of the AFP without promising full coverage: some entries with prohibitive resource requirements will be omitted.

T3.2 The Isabelle Analysis & Probability Theory library; Sites: **TUM** (lead)
(PM distribution: **TUM**: 26)
This library consists of more than 200.000 lines of definitions and proofs, corresponding to almost 4000 printed pages. It is fair to say that it is the most advanced machine-checked library in the area of analysis and probability theory. Because analysis and probability theory are key to many applications in engineering and science, this library will be a key exploitable result of the project: it is a fundamental enabling resource for almost any formal verification activity in these application areas. The purpose of this task is to structure, document and develop this library for optimal accessibility, ease of use and comprehensiveness.
For better access, the library needs to be modularised, which requires a significant refactoring effort. At the same time we need to add metadata (as provided by WP5) to the source material to turn this structured collection of theorems and proofs into a curated library at the Logipedia level.
The following areas of the library need to be developed further. The library support for integrals is extensive but suffers from the coexistence of different kinds of integrals. This requires unification and refactoring. Further essential material for mathematics, physics and engineering needs to be added: Fourier analysis and esp. the Fourier transform; stability theory for differential equations and dynamical systems, in particular Lyapunov functions; stochastic differential equations.

T3.3 The GeoCoq library; Sites: **Unistra** (lead)
(PM distribution: **Unistra**: 18)
The GeoCoq library consists of more than 100.000 lines of definitions and proofs. It is mostly based on synthetic approaches, where the axiom system is based on some geometric objects and axioms about them, but, following Descartes and Tarski, the analytic approach can be derived, where a field F is assumed (usually $\mathbb{R}$) and the space is defined as F^n. Moreover, it contains a

model of Tarski's axioms, based on the analytic approach, thus establishing the connection between these two approaches in the opposite direction. The main axiom system in this library is the one of Tarski, but Hilbert's axiom system and a version of Euclid's axioms, that are sufficient to prove the propositions in Book 1 of Euclid's Elements, are also defined. In the library, the focus is not only on axiom systems but also on axioms themselves. Eleven continuity axioms are available and are hierarchically organised. Finally, it contains a new refinement of Pejas' classification of parallel postulates together with proofs of the classification of 34 versions of the parallel postulate.
One of the remaining obstacles is the frequent use of computational steps in Coq proofs. The issue is that proofs containing "proof by reflection" reach a level of complexity that makes verification by Dedukti impractical. An approach is to isolate these proofs by reflection so that they are not perceived as simple conversion steps in the type theory proofs, but marked as proofs to be treated by an automatic tool. Another challenge is that CoqInE, a tool developed to translate Coq proofs into Dedukti type-checkable terms, produces terms in a expressing of the Calculus of Inductive Constructions in Dedukti. Currently, it is not possible to export these Dedukti terms to other proof assistant. However, another tool, Universo, has been developed and paves the way for the export of these terms.

T3.4 The Flyspeck library; Sites: **Inria** (lead)
(PM distribution: **Inria**: 20)
The HOL Light library is large and varied. One of its key libraries is the multivariate analysis library which spans the fields of metric spaces, topology, homology, linear algebra, convexity, real and complex analysis and transcendentals, derivatives, and integration. The Flyspeck project gives a formal proof of the Kepler conjecture, based on an original proof of Thomas Hales, and formalised largely in HOL Light.Some of these results are not formalised in any other system, motivating the project of importing the HOL Light library and Flyspeck project in the Dedukti system, in view of its integration into Logipedia.
Proofs coming from the HOL systems, including HOL Light, are known to be very large, adding to the issue of the scalability of exporting software for large libraries. Scalable export techniques HOL Light proofs have been investigated and can provide a solid base to this project.
The main milestones of this task are the further automation of the export from HOL Light to Dedukti, the export of the multivariate analysis library, of the whole HOL Light library, and of the Flyspeck to Dedukti.

T3.5 The CakeML compiler library; Sites: **Chalmers** (lead)
(PM distribution: **Chalmers**: 12)
The CakeML compiler (verified with the HOL4 prover) is one of only two verified compilers for real languages, the other being CompCert. Its export to Dedukti is one of the Key exploitable results of this project. HOL4 can export proofs in the OpenTheory format, which can in turn be translated into Dedukti. Currently this link from HOL4 via OpenTheory to Dedukti does not scale to something as sizeable as the CakeML compiler proof. This part of this task will rework the route via OpenTheory to scale better, possibly taking inspiration from an OpenTheory-like approach that scaled well for the HOL light prover.

Deliverables:
D3.1 (Due: 36, Type: DEM, Dissem.: PU, Lead: TUM) *Scalable export of proof terms for major parts of Isabelle/AFP*
D3.2 (Due: 36, Type: DEM, Dissem.: PU, Lead: TUM) *Export of Isabelle's extended analysis and probability theory library*
D3.3 (Due: 18, Type: OTH, Dissem.: PU, Lead: Unistra) *Export of most of GeoCoq library*
D3.4 (Due: 36, Type: OTH, Dissem.: PU, Lead: Inria) *Export some reusable parts of the multivariate analysis library from the Flyspeck library*
D3.5 (Due: 26, Type: DEM, Dissem.: PU, Lead: Chalmers) *Export parts of the CakeML library*

(as a minimum: the source semantics)

Work Package 4: Access (Trans-national and virtual access)			**Months:** 1 - 48
Site	***Inria***	**OcamlPro**	**all**
Effort	*54*	6	**60**

Objectives

- Defining and building the Logipedia hardware and software infrastructure in which the proofs developed in WP1, WP2, WP3 and WP6 will be integrated.
- Giving access to the Logipedia infrastructure to everyone (researchers, engineers, teachers, publishers) by providing a publicly accessible web site for displaying the proofs added in Logipedia, search tools to query the database, and a tool to automatically download and install those proofs in one's own computer.
- Developing the tools necessary to check the correctness of proofs added in Logipedia and transform them from one theory to another in coordination with WP7.

T4.1 Setting up the hardware and software architecture; Sites: **Inria** (lead)
(PM distribution: **Inria**: 6)
The Logipedia platform will reuse efforts done on an existing proof-of-concept application. However, with an ambition to be a reference platform accessible on the Web, it is necessary to meet several requirements for such an application, especially scalability, availability and sustainability. To achieve this goal, we will specify the architecture of the system by:

- collecting and formalising users needs,
- defining the proof submission process,
- defining the global software architecture,
- defining how files will be organised and stored in the Logipedia database,
- defining the reuse strategy in term of software components,
- defining the tooling architecture necessary for continuous integration and continuous deployment of Logipedia.

Several features will be taken into account in this task:

- interface with proof systems through the repository developed in **T4.3**,
- interface with the search tools using inputs from WP5 and **T4.4**,
- capability to integrate with publication databases through permanent links (**T8.9**),
- capability to integrate with third-party applications like Wikipedia and search engines.

Finally, we will size the needed hardware architecture to fit the objectives of the project. The goal is to define a scalable infrastructure in order to be able to manage an increasing traffic on the website to several thousands downloads a day from all over the world.
Security of the database is also important. It must be possible to recover the data in a few minutes at any time to ensure every user will be able to continue working. Defining the redundancy of the infrastructure as well as the backup strategy is key to guarantee the security and high availability of the platform.
The hosting strategy will be decided taking into account costs, efficiency and sustainability. The platform can be hosted on:

- some servers at Inria with redundancy at other sites like München,
- public cloud services like OVH.com, Scaleway or Amazon Web Services.

T4.2 Giving access to the infrastructure on the Web; Sites: **Inria** (lead)
(PM distribution: **Inria**: 12)
In this task, the implementation of the architecture defined in **T4.1** will be done using agile methodology in order to get frequent feedback from end users and adjust the implementation to fit the needs.

In addition, we will develop an ergonomic and user-friendly web interface to access the platform, navigate into the available proofs and download them. To this end, we will use appropriate languages and tools (Unicode symbols, CSS, etc.) to provide a nice rendering of logical formulas and usual mathematical symbols.
We will then deploy the software on the chosen hardware infrastructure.
All this work will be done using continuous integration / continuous deployment tooling for:

- automatic building/deployment of the Logipedia application on the hardware infrastructure,
- automatic integration of new validated proof files in the Logipedia database.

Finally, we will perform unitary tests and integration tests, especially with components developed in other tasks: Opam repository from **T4.3** and search tools from **T4.4**.

T4.3 Giving access to the infrastructure in proof systems; Sites: **OcamlPro** (lead)
(PM distribution: **OcamlPro**: 6)
This task will provide a proof manager for users of Logipedia. This proof manager will enable users to automatically download and install proofs as well as their dependencies in order to ease the integration of proofs from Logipedia in their own developments.
Opam is an open-source source-based package manager, which has been successfully used by the OCaml community since 2012, where it manages 2585 versioned packages for a total of 13196 combinations of package and version, guaranteeing its ability to connect people across large communities. Furthermore, Opam is meant to provide management capabilities not only to OCaml, but to any language, which is why it is already used as a proof manager by the Coq community where it has been proven to be reliable and suited to managing formal proofs. This makes it a prime candidate to be the proof manager for Logipedia.
This task will use the Opam management tool to develop a repository containing all the proofs of Logipedia, allowing users across Europe to automatically and transparently download and install proofs and their dependencies. This requires to develop a tool able to read the proof database of Logipedia and create an Opam repository, and integrate it in the infrastructure built in **T4.2**.

T4.4 Providing search tools; Sites: **Inria** (lead)
(PM distribution: **Inria**: 18)
We will provide users with search tools enabling them to perform queries on Logipedia in order to find specific theorems or proofs. First, users will be able to search libraries theorems by their names and other metadata (see **T5.2**), including complex semantic queries expressed in the SPARQL language for semantic annotations produced in **T5.3** and **T5.4**. Second, it will be possible to search theorems and proofs based on their structure and mathematical content (types, operators, used axioms and rules, etc.), using exact matching, regular expressions over formulas, and deeper content matching, such as the one done in the MathWebSearch system. Users can use this both to find a specific theorem that could be useful in their current development and to analyse the proofs themselves, e.g., to find all proofs using a given set of axioms.

T4.5 Development and maintenance of Dedukti tools; Sites: **Inria** (lead)
(PM distribution: **Inria**: 18)
We will consolidate the source code of Dedukti, the tool that allows to check the correctness of proofs added in Logipedia, so as to handle the proofs generated in WP1, WP2 and WP3. We will also extend it so as to handle the new theories that will be developed in WP6.
We will also consolidate and integrate the existing tools devoted to the translation of proofs in Dedukti from one theory to another one in coordination with WP7, as well as the tools for generating proofs in target systems from Dedukti proofs.
All this work will be done using continuous integration and deployment tools.

Deliverables:
D4.1 (Due: 6, Type: OTHER, Dissem.: PU, Lead: Inria) *Platform architecture*
D4.2 (Due: 18, Type: DEC, Dissem.: PU, Lead: Inria) *Web interface*

D4.3 (Due: 24, Type: OTHER, Dissem.: PU, Lead: Inria) *Improved version of Dedukti and its associated translation tools*

D4.4 (Due: 12, Type: OTHER, Dissem.: PU, Lead: OcamlPro) *Opam repository for Logipedia* ⇝**Ms4**

D4.5 (Due: 36, Type: OTHER, Dissem.: PU, Lead: Inria) *Search tool for Logipedia*

Work Package 5: Structure of the encyclopedia (Trans-national and virtual access)					**Months:** 1 - 48
Site	**INPT**	**Unibo**	**USaclay**	***FAU***	**all**
Effort	8	5	40	*25*	**78**

Objectives

Providing infrastructure for the structured ontological representation of libraries and use it to enrich the information about formal libraries in Logipedia. Enabling the exchange and reuse the knowledge between prover systems.

Description

We proceed in three steps. Firstly, Tasks **T5.1** and **T5.2** extend the Dedukti language with features for high-level representations that are critical for accessing parts of and searching libraries. This includes a framework to *define* typed meta-data in form of ontologies, and to *enforce* them in the Dedukti libraries. Secondly, Tasks **T5.3** builds ontologies serving as technical exchange format as well as domain-specific descriptions of libraries. Thirdly, Tasks **T5.4** fills the ontology with data from both formal libraries and natural language articles and use the ontology to relate to each other.

This work package will be jointly led by Burkhart Wolff at **USaclay** and Florian Rabe at **FAU**. (Where a single leader is needed for formal purposes, the latter site will be the primary leader.) Burkhart Wolff implemented a document ontology framework in Isabelle and developed several applications in the field of formal software engineering. Florian Rabe has extensive experience in designing and implementing knowledge representation languages as well as in exporting theorem prover libraries.

T5.1 Library Structure; Sites: **FAU** (lead)
(PM distribution: **USaclay**: 6, **FAU**: 8)
This task extends the Dedukti language with primitives for representing library, document, informal annotations, and theory structure. This includes in particular the definition of unique identifiers for all declarations, which is critical for alignments.

T5.2 Ontological Framework for Meta-Data; Sites: **USaclay** (lead)
(PM distribution: **USaclay**: 24)
This tasks extends the Dedukti language with a framework for meta-data annotations. This will cover all levels of the structure introduced in **T5.1** as well as the level of subexpressions of Dedukti expressions. It will also provide a mechanism to validate meta-data according to assertions.

T5.3 Reference Ontology; Sites: **USaclay** (lead)
(PM distribution: **INPT**: 8, **USaclay**: 6, **FAU**: 6)
This tasks compiles, integrates, and curates the various ontologies used for describing libraries in the project. These come from several sources:

- The ontology induced by the structuring features built in task **T5.1**.
- The ontologies built by users using the ontology framework built in task **T5.3**.
- Manually written ontologies or imports of existing ontologies for knowledge formalised in prover libraries, such as the Upper Library Ontology and domain-specific ontologies. The latter may include for example the ontologies for engineering and their relation to descriptive models and certification standards that are planned to be developed by **INPT**.

T5.4 Ontological Representation of Formal Libraries; Sites: **FAU** (lead)
(PM distribution: **Unibo**: 5, **USaclay**: 4, **FAU**: 11)
This task extends the exports from Isabelle and Coq developed in WP3 with structural and ontological data that conforms to the language features introduced in Tasks **T5.1** and **T5.2**. It will also build on the ontological export of RDF triples relative to the Upper Library Ontology developed for Isabelle and Coq.
The task leader will collaborate with M. Wenzel for Isabelle and C. Sacerdoti Coen at **Unibo** for Coq, with whom long-standing collaborations on these library exports exist. Wenzel's involvement will take the form of a sub-contract of **FAU** corresponding to roughly 4 person-months, an arrangement that has already been used twice in other projects. The resources for this subcontract are not included in the person-months listed here.

Deliverables:
D5.1 (Due: 28, Type: R, Dissem.: PU, Lead: USaclay) *This deliverable describes the language developed in Tasks 1 and 2.*
D5.2 (Due: 36, Type: R, Dissem.: PU, Lead: USaclay) *This deliverable describes the reference ontology developed in Task 3.*
D5.3 (Due: 48, Type: R, Dissem.: PU, Lead: FAU) *This deliverable describes the representation of major formal libraries developed in Task 4.*

Work Package 6: Theories (Joint research activity)											**Months:** 1 - 48
Site	**Inria**	***UIBK***	**ULeeds**	**LMU**	**UBia**	**UoB**	**MED-EL**	**P&R**	**UAIC**	**RV**	**all**
Effort	87	*11*	3	16	70	3	5	11	6	7	**219**

Objectives

Bringing proof systems that are currently at LIL 0, i.e. whose logic and theories have not yet been expressed in Dedukti, to LIL 2 or so that at least elementary proofs can be exported and checked in Dedukti.

Achieving these goals requires (i) expressing the logics that underlie these systems in Dedukti and (ii) instrumenting the original proof systems so that they can export proofs that can be checked in Dedukti. Much of the work required in this work package is of foundational nature.

Description

The tasks are presented according to the individual systems. Nevertheless, the project addresses cross-cutting concerns and foundational aspects such as set-theoretic vs. type-theoretic foundations, predicate and dependent types, (co-)recursive definitions and (co-)inductive proofs and so on. The network established in this project will avoid "reinventing the wheel", also taking advantage of the experiences of the more mature systems considered in WP1 and of the work carried out in WP2.

T6.1 Express Cubical Type Theory in Dedukti; Sites: **Inria** (lead)
(PM distribution: **Inria**: 46, **ULeeds**: 3, **UoB**: 3)

- Express 2-Level Type Theory (2LTT) as an object theory in Dedukti, as a stepping stone towards encoding more expressive variants of Homotopy Type Theory.
- Express a core Cubical Type Theory (CubTT) in 2LTT: here, the main challenge is that equality in cubical type theory is more expressive than what can be expressed through rewrite rules in Dedukti.
- Define structures such as cartesian cubical type theory on top of the core, and compare these structures.

- Import the UniMath library into Dedukti and translate it into Cubical Type Theory. The UniMath library extends a version of Martin-Löf type theory by the univalence axiom. It provides an interesting case study for our encoding of CubTT.

T6.2 Express Matching Logic in Dedukti and instrument the K prover; Sites: **UAIC** (lead)
(PM distribution: **UAIC**: 6, **RV**: 7)

- Express Kore in Dedukti: Kore is the specification language used for describing Matching Logic theories in K and will serve as the interface between the K prover and Logipedia.
- Instrument the K prover to produce detailed proof traces expressed in Kore.
- Integrate an instrumented automated prover (cf. tasks **T2.1** and **T2.2**) into the K prover for obtaining proofs for the queries generated by the K prover.
- Combine the translations into Dedukti from Kore and from the automated prover into a single Dedukti proof.

T6.3 Express the theory of Minlog in Dedukti; Sites: **LMU** (lead)
(PM distribution: **LMU**: 16)

- Further develop and implement extensional realisability in Minlog as a necessary first step for bridging Minlog and Dedukti.
- Express the core Minlog logic and proofs in Dedukti, benefiting from the work on realisability for exporting programme extraction to Dedukti.
- Properly encode coinduction and corecursion in Dedukti: these concepts are fundamental to Minlog, for example for representing real numbers as streams of digits, but they are not native to Dedukti.
- Import a subset of Dedukti into Minlog, apply program development by proof transformation, and export back. This will make Dedukti a usable tool for the development of proofs and programs in constructive analysis and allow Minlog users to benefit from theories formalised using other proof assistants.

T6.4 Express the theory of Mizar in Dedukti; Sites: **UBia** (lead)
(PM distribution: **UIBK**: 11, **UBia**: 70)

- Express the foundations of Mizar (first-order Tarski-Grothendieck set theory) in Dedukti, together with Mizar's soft type system. Take advantage of Dedukti's rewriting capabilities to automate parts of type inference.
- Instrument Mizar to export type disambiguation data. Exporting the information present in the Mizar types will enable us to optimise the representation of Mizar statements in Dedukti.
- Express Mizar's equality checking and unification steps as a mix of small proof steps and rewrite rules so that Dedukti's proof kernel can verify them. Export necessary semantic information that is not currently available outside of the Mizar checker in order to check the basis of the Mizar library and make it available in Logipedia.

T6.5 Express the theory of PVS in Dedukti; Sites: **Inria** (lead)
(PM distribution: **Inria**: 20)

- Extend the existing encoding in Dedukti, restricted to a fragment of PVS with decidable type checking, and represent proofs of type checking conditions for predicate subtypes.
- Instrument PVS to export proof traces to Dedukti, as PVS tactics do not produce proof terms.
- Design and implement a PVS proof checker in Dedukti based on the reconstruction of proof traces exported from PVS.

T6.6 Express Smart models and proofs in Dedukti; Sites: **P&R** (lead)
(PM distribution: **P&R**: 11)

- Choose a viable translation strategy from Smil programs (the intermediate form of Smart models used by ProvenTools) into Dedukti by experimenting with direct translations or translations through a different tool such as Coq or Why3.
- Develop a prototype capable of translating the definitions and proof obligations corresponding to relevant examples from the Smart standard library.

- Integrate the prototype into ProvenTools: integrate proof rules internal to the prover and evaluate the scalability to cross-verifying real-world use cases.

T6.7 Express the theory of TLA$^+$ in Dedukti; Sites: **Inria** (lead)
(PM distribution: **Inria**: 21, **MED-EL**: 5)

- Express the untyped TLA$^+$ set theory with choice in Dedukti.
- Instrument backends of the TLA$^+$ Proof System to export proofs, taking advantage of Dedukti's rewriting capabilities in order to compress the size of proofs.
- Formalise a distributed assignment of rehabilitation programs to patients and updates of the process in MED-EL's software.

Deliverables:
D6.1 (Due: 18, Type: R, Dissem.: PU, Lead: UIBK) *Report on defining theories in Dedukti*
D6.2 (Due: 48, Type: R, Dissem.: PU, Lead: Inria) *Report on the integration of Cubical Type Theory in Dedukti*
D6.3 (Due: 42, Type: R, Dissem.: PU, Lead: UAIC) *Report on the integration of Matching Logic*

D6.4 (Due: 48, Type: R, Dissem.: PU, Lead: LMU) *Report on the integration of Minlog*
D6.5 (Due: 48, Type: R, Dissem.: PU, Lead: UBia) *Report on the integration of Mizar*
D6.6 (Due: 48, Type: R, Dissem.: PU, Lead: Inria) *Report on the integration of PVS*
D6.7 (Due: 48, Type: R, Dissem.: PU, Lead: P&R) *Report on the integration of Smart*
D6.8 (Due: 48, Type: R, Dissem.: PU, Lead: Inria) *Report on the integration of TLA$^+$*

Work Package 7: Proof engineering (Joint research activity)											**Months:** 1 - 4
Site	**Inria**	**Unistra**	**UIBK**	**Unibo**	***UBel***	**USaclay**	**FAU**	**ULeeds**	**LMU**	**IMT**	**all**
Effort	8	25	8	14	*20*	9	11	17	1	11	**124**

Objectives

Investigating proof engineering methods for detecting corresponding concept, applying those methods to build a library of alignments present across the Logipedia database, and building a set of alignment-based services.

Description

Within tasks **T7.1** and **T7.2** manual investigations will be applied to detect and to align basic mathematical objects (logics, numbers, sets, functions, relations etc.). As a case study, various formalisations of geometry will manually be aligned. Task **T7.3** is devoted to automated detection of an ontology alignments by employing database and semantic-web technology as well as unsupervised machine translation algorithms. Tasks **T7.4** and **T7.5** are devoting to building alignment based services: search and proof-rewriting.

T7.1 Logical foundations; Sites: **ULeeds** (lead)
(PM distribution: **Inria**: 3, **ULeeds**: 17, **LMU**: 1)
Mechanised mathematical theories span a wide spectrum of different logical foundations (e.g., set theory, first-order logic, higher-order logic, or different variants of type theory) and it is critically important that these can be related to each other in ways that will ensure their interoperability. This task will be devoted to aligning logical connectives, classical and intuitionistic logic, eliminating second-order proof, predicative vs impredicative theories, and to identification of abstraction layers. The focus will be on the Coq and Adga systems.

T7.2 Case study: geometry; Sites: **Unistra** (lead)
(PM distribution: **Unistra**: 25, **UBel**: 20)
Several large-scale formalisations of geometry are available and a very interesting case study is to align fundamental objects that are introduced quite differently (both synthetically or analytically). Since geometrical objects are sometimes introduced using real or complex numbers, this case study will also investigate alignment of different numbers ($\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$), as well as alignment of other fundamental objects (sets, relations, functions).

T7.3 Automated proof engineering; Sites: **IMT** (lead)
(PM distribution: **UIBK**: 8, **USaclay**: 9, **IMT**: 11)
Since manual concept alignment is very tedious and time-consuming task, automated methods for detecting and organising alignments will be developed on top of an ontology framework that defines mappings between base concepts belonging to different theories. Unsupervised machine learning methods to directly find correspondences between statements and their constituent constants and types will be developed and applied.

T7.4 Alignment-Based Search; Sites: **FAU** (lead)
(PM distribution: **FAU**: 11)
This task designs and implements alignment-expression translation functions and uses them to realise search and browsing service modulo alignment.

T7.5 Proof-Rewriting; Sites: **Unibo** (lead)
(PM distribution: **Inria**: 5, **Unibo**: 14)
This task will develop methods to automatically rewrite proofs and statements by a mix of ELPI (developed by a join Ubo-Inr team) and Dedukti rewrite rules. Statement rewriting will find direct application to alignment based search and browsing as well (developed in **T7.4**).

Deliverables:
D7.1 (Due: 24, Type: DEM, Dissem.: PU, Lead: ULeeds) *Algorithms for translating between fragments of the theories of Coq and Agda, possibly extended with classical logic*
D7.2 (Due: 24, Type: DEM, Dissem.: PU, Lead: Unistra) *Manually created basic alignments for some of the examples of the study: $\mathbb{N}$, $\mathbb{R}$, $\mathbb{C}$, $\mathbb{R} \to \mathbb{R}$, between Coq and Isabelle and between Coq via Dedukti*
D7.3 (Due: 36, Type: DEM, Dissem.: PU, Lead: UBel) *A manually created alignments between Tarski's geometry defined in Coq, Tarski's geometry defined in Isabelle, Analytic Geometry defined in Coq, Analytic Geometry in Isabelle*
D7.4 (Due: 48, Type: DEM, Dissem.: PU, Lead: IMT) *Implementation of a prototype automated alignment inference engine*
D7.5 (Due: 48, Type: DEM, Dissem.: PU, Lead: FAU) *Implementation of the alignment-based search service*
D7.6 (Due: 48, Type: DEM, Dissem.: PU, Lead: Unibo) *Implementation of the alignment based proof-rewriting service*

Work Package 8: Dissemination, communication and exploitation									**Months:** 1 - 48
Site	**Inria**	**Unistra**	***ULiege***	**IMT**	**ClearSy**	**UoB**	**Edukera**	**ZIB**	**all**
Effort	6	2	*8*	2	2	4	12	14	**50**

Dissemination, communication, and exploitation activities involve all partners. Only those having a coordinating action or an action involving research in this area are listed above.

Objectives

Communicating and disseminating the activities and results generated in the project to target audiences and key stakeholders, and developing the communities of Logipedia users at the European

scale. This includes:

- widely promoting and ensuring the visibility of the Logipedia project through tailored communication tools (web site, social media, etc.),
- ensuring awareness of the results by publishing them in journals and presenting them in international conferences,
- promoting the use of Logipedia and spreading our expertise through summer schools and training sessions,
- developing and structuring the various communities of users of Logipedia (researchers, industrials, teachers, and publishers) through clubs with a general meeting once a year,
- discussing with publishers the integration of links to Logipedia on their platforms,
- providing a teaching interface for using Logipedia at schools and universities.

Scientific research and education at all levels are concerned with the discovery, verification, communication, archival and usage of mathematical results. These tasks have been supported by physical books, conferences and other means. The availability of a formal online encyclopedia which offers in a single place the communication, archival and verification of mathematical knowledge will be of prime importance for researchers, industrials, teachers and editors.

A detailed *Dissemination and communication* delivered at month 4 will outline all activities to be carried out during the course of the project.

T8.1 Communication; Sites: **Inria** (lead)
(PM distribution: **Inria**: 6)
This task comprises all forms of communication activities such as creation of the project website including visitor analysis and monitoring tools, outreach activities, creation of flyers, posters, etc. A dedicated and experienced communication officer from the Inria Saclay communication team coordinates all these activities together with the work package leader.

T8.2 Dissemination; Sites: **ULiege** (lead)
(PM distribution: **ULiege**: 8)
A yearly Logipedia conference will advertise the Logipedia infrastructure, federate forces around the future challenges for further developing the infrastructure, and be a regular meeting place for the various communities of users.

T8.3 Training Logipedia developers and users; Sites: **UoB** (lead)
(PM distribution: **UoB**: 2)
A suitable training for people at various stages of their career will be provided.

- Master and PhD students. We will organise two summer schools on Logipedia and on the translation of proofs across systems. Such events form an opportunity to improve the gender balance in our community by attracting female students.
- Engineers. We will develop training for employees. We will develop pedagogic innovations in the delivery of training, in support of companies.
- Teachers. We will set up a forum for exchange of tips and tricks for the use of computer proof assistants and formal proofs in teaching such classes. Particular emphasis will be on the sharing of teaching materials, which we will encourage to make available under free licenses.

T8.4 Expanding the use of Logipedia in research; Sites: **UoB** (lead)
(PM distribution: **UoB**: 2)
To foster interactions between developers and users of Logipedia, we plan the following activities.

- Logipedia helpdesk: we will establish a "Logipedia helpdesk", as a unique, and easy-to-reach point of contact for any researcher seeking help on the use of Logipedia. The helpdesk will forward any incoming request for help to a suitable expert among the Logipedia developers.
- We will have a "user day" attached to our yearly Logipedia meetings. During the user day, we plan to have both talks on uses of Logipedia, as well as a round table discussion featuring both developers and users, for discussing present problems and future challenges.

- We will have Logipedia workshops affiliated to conferences to present the project to a wide range of researchers not yet involved in the project.
- We will run an electronic seminar series, thus providing an ecological and economical way of presenting one's work and giving feedback.

T8.5 Expanding the use of Logipedia in the industry; Sites: **ClearSy** (lead)
(PM distribution: **ClearSy**: 2)
The use of formal proof systems by companies is still limited, in particular by small and medium-sized enterprises and small and medium-sized industries. The club will be an opportunity to:

- develop and organise a club of industrial users of Logipedia,
- promote the use of Logipedia to more industrial users,
- organise meetings to allow companies to present their results/experiences using Logipedia, present the results of Logipedia to companies, collect their opinions on their experience and needs with respect to Logipedia,
- encourage cooperation between companies from the Logipedia ecosystem, for example, by sharing rules and proofs between companies using the B method.

T8.6 Promoting the use of Logipedia by certification authorities; Sites: **IMT** (lead)
(PM distribution: **IMT**: 2)
With the help of the French national agency of the security of information systems (ANSSI), we will promote use of Logipedia in European certification authorities.

- ANSSI would be interested in having one unique language for formal proofs, rather than several ones which potentially require as many experts as the number of languages. The consortium will have to first evaluate if Logipedia addresses this need.
- Contact other relevant European public institutions (e.g., certification bodies) in order to introduce them to Logipedia, and understand their potential use of Logipedia.

T8.7 Expanding the use of Logipedia in education; Sites: **Unistra** (lead)
(PM distribution: **Unistra**: 2)
The club of users in education gathers teachers who are already actively using formal proof for teaching in computer science, mathematics and logic but are not necessarily members of the community of researchers in formal theorem proving. These early adopters, will provide continuous feedback to the project members on the usability and accessibility of the system from this particular point of view. More than 30 persons have already accepted to take part in this club. The role of the club of users in education will be to:

- express the needs of students and teachers with respect to proof presentations and tools,
- exchange their experience in teaching formal proofs or using theorem provers in class by participating in the ThEdu community,
- provide use cases,
- contribute to the dissemination of information about Logipedia.

T8.8 Expanding the use of Logipedia in publishing; Sites: **ZIB** (lead)
(PM distribution: **ZIB**: 2)
We will create and animate a club of users in publishing. We will invite people and organisation working in the publication of research works to join this club: publication archives (arXiv, CCSD-HAL, Lipics, ACM, zbMath, etc.), conference steering committees (CPP, ITP, POPL, CICM, etc.), journal editorial boards (JFR, JFM, etc.). It will be an opportunity to:

- discuss with them how to use Logipedia to store and check proofs presented or used in scientific publications,
- express their needs with respect to Logipedia.

T8.9 Linking scientific publications to Logipedia; Sites: **ZIB** (lead)
(PM distribution: **ZIB**: 12)
We will build a service for Logipedia which provides an overview about the development in logic and an intuitive access to the following classes of objects - axioms - theories/calculi - software

(theorem provers) - applications based on logic tools (e.g. the metro in Paris, other SAT problems) The objects and the relations between the objects will be realised via a graph. We will:

- define the objects of each class (we will start from the theorem provers, apply the publication-based approach for the software and search for references to axioms, theories, and applications),
- define a special ontology to describe linking and dependencies between the objects (metadata),
- add these metadata to all objects,
- provide a graph visualisation of the network (lattice) and development of a search engine.

This gives the user a comprehensive entry point into an emerging topic.

T8.10 Web teaching interface for doing proofs at school; Sites: **Edukera** (lead)
(PM distribution: **Edukera**: 12)
We will build an web application for the education community on top of Logipedia. It will enable students to solve exercises that require a mathematical proof with a simple user interface. The development tasks are listed below:

- graphical web component to display a Logipedia proof (could be used in Logipedia website),
- point-and-click interaction engine on top of Logipedia,
- interactive proof interface application (Web interface, connection to LMS, etc.)

Deliverables:
D8.1 (Due: 3, Type: DEC, Dissem.: PU, Lead: Inria) *Setting up project website*
D8.2 (Due: 4, Type: DEC, Dissem.: PU, Lead: ULiege) *Dissemination and communication plan*

D8.3 (Due: 36, Type: R, Dissem.: PU, Lead: UoB) *Report on training*
D8.4 (Due: 36, Type: R, Dissem.: PU, Lead: UoB) *Report on the club of academic users*
D8.5 (Due: 36, Type: R, Dissem.: PU, Lead: Inria) *Report on the club of industrial users*
D8.6 (Due: 36, Type: R, Dissem.: PU, Lead: IMT) *Report on the use of Logipedia by certification bodies*
D8.7 (Due: 36, Type: R, Dissem.: PU, Lead: Unistra) *Report on the club of users in education*
D8.8 (Due: 36, Type: R, Dissem.: PU, Lead: ZIB) *Report on the club of users in publishing*
D8.9 (Due: 24, Type: OTHER, Dissem.: PU, Lead: ZIB) *Database and search engine for linking Logipedia formal proofs with scientific publications*
D8.10 (Due: 24, Type: DEM, Dissem.: PU, Lead: Edukera) *Display Logipedia proofs*
D8.11 (Due: 30, Type: OTHER, Dissem.: PU, Lead: Edukera) *Application for education*

Work Package 9: Management		**Months:** 1 - 48
Site	***Inria***	**all**
Effort	*44*	**44**

The ressources allocated to the work package leaders for their participation to the governance of the project are included in their own work package.

Objectives

Managing the project activities led by the consortium. Ensuring the necessary conditions to enable the project to achieve its objectives while meeting its cost, time and quality requirements. This includes the scientific, administrative, financial and legal management.

Gilles Dowek, Inria senior researcher and professor at ENS Paris-Saclay, will be the Coordinator and leader of this work package. Gilles has been PI of many projects. He will also be supported by a Deputy coordinator, Frédéric Blanqui, an European project manager from the Innovation, Partnership and Transfer Office of Inria Saclay and a Chief engineer.

This therefore includes:

- a scientific and technical coordination to create a vibrant scientific and technical environment

within the project,
- the overall management of the project and consortium according to the governance structure and procedures explained in section 3.2.,
- an efficient project management, as specified in 3.2, including: overall administrative and financial project management, including reporting to the European Commission, quality management, and assessment and risk management, including conflict or dispute management.

T9.1 Scientific and technical coordination; Sites: **Inria** (lead)
(PM distribution: **Inria**: 20)
The scientific and technical coordination will be led by Inria senior researcher Gilles Dowek. He will be in charge of ensuring the implementation of the scientific strategy of Logipedia and thereby ensuring the growth of the Logipedia community. This task foresees a key role of scientific animation and to impulse the organisation of scientific activities, together with the work package leader in charge of dissemination. Gilles Dowek will supervise the ongoing scientific and technical coordination and help with the innovation management, together with the Deputy coordinator, the European project manager, the Chief engineer and the steering committee. The Coordinator will also chair the steering committee and the general assembly. The Coordinator will be the scientific point of contact within the consortium and for the consortium when the project needs to be represented.

T9.2 Quality management; Sites: **Inria** (lead)
(PM distribution: **Inria**: 5)
Quality management includes the elaboration and application of a Project Quality Plan, internal guideline detailing project procedures (quality assurance, document management, document templates, etc.), the set-up of the private online workspace for publishing and exchanging documents within the consortium, the monitoring of information management, ensuring good communication in the consortium, and maintenance of Partners' contact information, including emailing lists.

T9.3 Administrative and Financial Management; Sites: **Inria** (lead)
(PM distribution: **Inria**: 15)
A European Project Manager (EPM) from the Transfer and Innovation team of Inria Saclay, who has an extensive experience in handling innovation from research projects such as Logipedia, will be working 50% of her or his time during the course of the project. The consortium will therefore benefit from tailored and on-demand advice regarding the use and potential transfer of the research results during the course of the project. The Project manager will ensure the day-to-day management as it will be the administrative and financial point of contact for the consortium and a dedicated contact point for the European Commission. The meeting preparation and follow-up will be another task of the Project manager and will include organising plenary meeting with the partner organisation, general assembly, minutes and review meeting with the consortium. Financial aspects will be a crucial task of the Project manager and include: payment to partners; ensuring financial monitoring within the consortium and leading the financial reporting to the European Commission. The Project manager will also be responsible, together with the Coordinator, for ensuring the technical work and deliverables meet the technological objectives of the project according to the defined schedule. The Project manager will work very closely with the work package leaders in order to monitor the progress of the technical work and to identify potential risks within each work package. The Project manager also acts as a quality manager to ensure that the content of the deliverables meets the quality standards defined for the project. The Project manager reports to the Coordinator. The development and maintenance of collaborative tools will be ensured by Inria and monitored by the Project manager. A common teleconference tool, storage space, reporting and intranet will be set up and detailed in the collaborative tools deliverable of month 2.

T9.4 Legal Management (data, ethics, GDPR); Sites: **Inria** (lead)
(PM distribution: **Inria**: 4)

The preparation of the Consortium and Grant agreement will be led by the Project manager at Inria Saclay. If a change arises during the course of the project, an amendment will be prepared by the project management team, in close collaboration with Inria legal team. Data Protection, ethics and GDPR Compliance Management will also be ensured by Inria. The main objective here is to provide guidance on data protection for the research activities of the project in the context of the European General Data Protection Regulation (GDPR). If at some point during the course of the project, the consortium or any scientist is unsure about how to handle a particular situation or requires advice on ethical issues, the partners or the individuals, supported by the Project manager, will refer to the operational ethical committee of Inria (the COERLE) before proceeding.

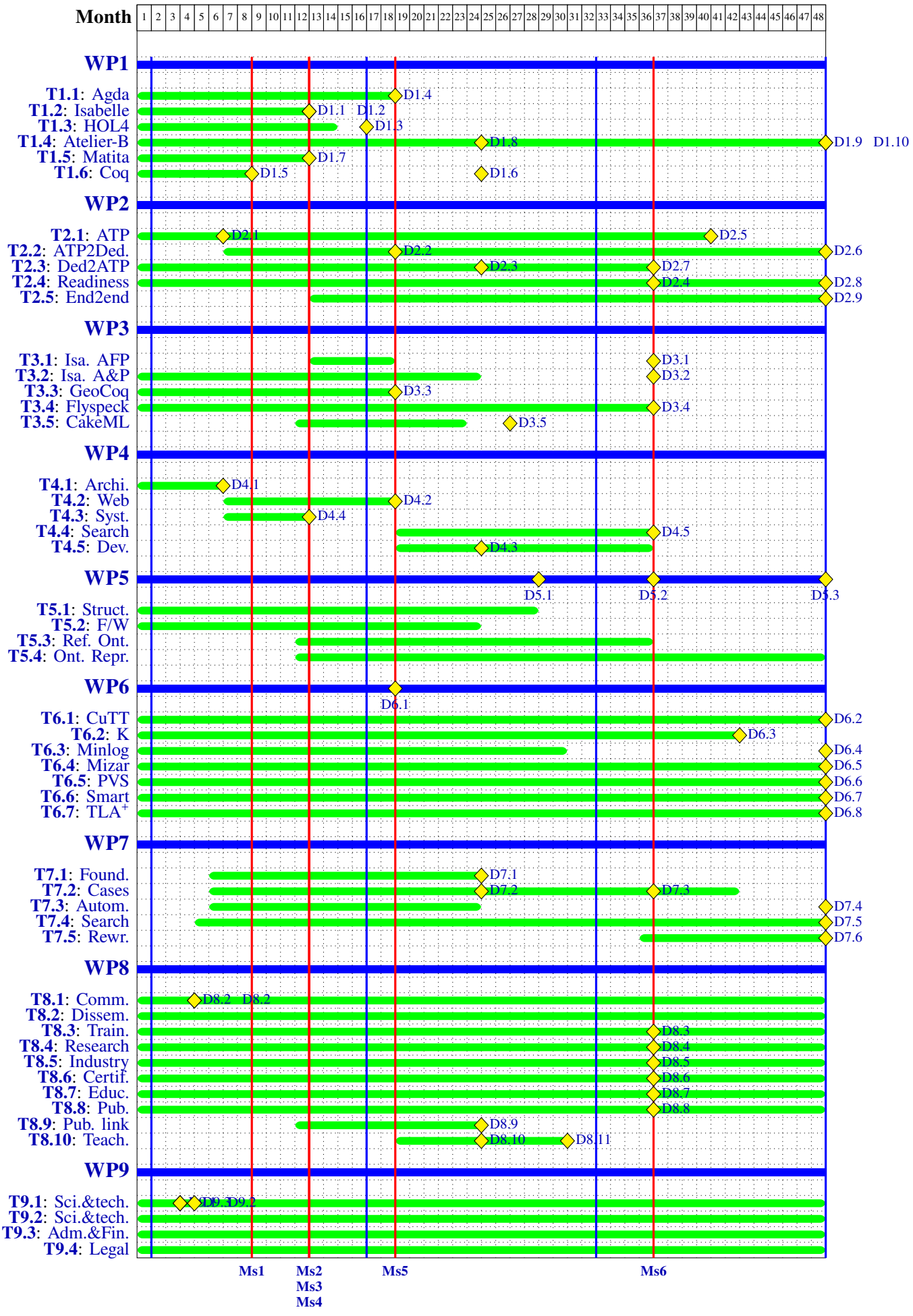
Deliverables:
D9.1 (Due: 3, Type: R, Dissem.: PU, Lead: Inria) *Partner guide: gouvernance, management and collaborative tool*
D9.2 (Due: 3, Type: R, Dissem.: PU, Lead: Inria) *Logipedia quality plan*
D9.3 (Due: 4, Type: R, Dissem.: PU, Lead: Inria) *Data Management Plan*

List of all deliverables

#	Deliverable name	WP	Lead	Type	Level	Due
D8.2	Setting up project website	WP8	**Inria**	DEC	PU	3
D9.1	Partner guide: gouvernance, management and collaborative tool	WP9	**Inria**	R	PU	3
D9.2	Logipedia quality plan	WP9	**Inria**	R	PU	3
D8.2	Dissemination and communication plan	WP8	**ULiege**	DEC	PU	4
D9.3	Data Management Plan	WP9	**Inria**	R	PU	4
D2.1	Prototypes of Trace producing Automatic Theorem Prover and Satisfiability Modulo Theories solver	WP2	**ULiege**	DEM	PU	6
D4.1	Platform architecture	WP4	**Inria**	OTHER	PU	6
D1.5	Export of proofs from Coq	WP1	**Inria**	OTHER	PU	8
D1.1	Export proofs from Isabelle	WP1	**TUM**	OTHER	PU	12
D1.2	Improved memory management and monitoring for Poly/ML	WP1	**TUM**	OTHER	PU	12
D1.7	Export proofs & metadata from Matita	WP1	**Unibo**	OTHER	PU	12
D4.4	Opam repository for Logipedia	WP4	**OcamlPro**	OTHER	PU	12
D1.3	Export proofs from HOL4	WP1	**Chalmers**	OTHER	PU	16
D1.4	Export Agda's standard library	WP1	**TUDelft**	OTHER	PU	18
D2.2	Prototype of a translator for some traces to Dedukti	WP2	**IMT**	DEM	PU	18
D3.3	Export of most of GeoCoq library	WP3	**Unistra**	OTH	PU	18
D4.2	Web interface	WP4	**Inria**	DEC	PU	18
D6.1	Report on defining theories in Dedukti	WP6	**UIBK**	R	PU	18
D1.6	Scalable export of proof terms with meta data from Coq	WP1	**Unibo**	OTHER	PU	24
D1.8	A prototype of tool exporting proof terms from Atelier B	WP1	**ClearSy**	DEM	PU	24
D2.3	Prototype of a translator from Dedukti statements into TPTP and SMT-LIB, application to simple libraries	WP2	**USaclay**	DEM	PU	24
D4.3	Improved version of Dedukti and its associated translation tools	WP4	**Inria**	OTHER	PU	24
D7.1	Algorithms for translating between fragments of the theories of Coq and Agda, possibly extended with classical logic	WP7	**ULeeds**	DEM	PU	24
D7.2	Manually created basic alignments for some of the examples of the study: $\mathbb{N}$, $\mathbb{R}$, $\mathbb{C}$, $\mathbb{R} \to \mathbb{R}$, between Coq and Isabelle and between Coq via Dedukti	WP7	**Unistra**	DEM	PU	24
D8.10	Display Logipedia proofs	WP8	**Edukera**	DEM	PU	24
D8.9	Database and search engine for linking Logipedia formal proofs with scientific publications	WP8	**ZIB**	OTHER	PU	24

#	Deliverable name	WP	Lead	Type	Level	Due
D3.5	Export parts of the CakeML library (as a minimum: the source semantics)	WP3	**Chalmers**	DEM	PU	26
D5.1	This deliverable describes the language developed in Tasks 1 and 2.	WP5	**USaclay**	R	PU	28
D8.11	Application for education	WP8	**Edukera**	OTHER	PU	30
D2.4	Tools based on ATPs to increase Logipedia Readiness	WP2	**USaclay**	DEM	PU	36
D2.7	Translator from expressive Dedukti statements into TPTP and SMT-LIB, application to large libraries	WP2	**USaclay**	OTHER	PU	36
D3.1	Scalable export of proof terms for major parts of Isabelle/AFP	WP3	**TUM**	DEM	PU	36
D3.2	Export of Isabelle's extended analysis and probability theory library	WP3	**TUM**	DEM	PU	36
D3.4	Export some reusable parts of the multivariate analysis library from the Flyspeck library	WP3	**Inria**	OTH	PU	36
D4.5	Search tool for Logipedia	WP4	**Inria**	OTHER	PU	36
D5.2	This deliverable describes the reference ontology developed in Task 3.	WP5	**USaclay**	R	PU	36
D7.3	A manually created alignments between Tarski's geometry defined in Coq, Tarski's geometry defined in Isabelle, Analytic Geometry defined in Coq, Analytic Geometry in Isabelle	WP7	**UBel**	DEM	PU	36
D8.3	Report on training	WP8	**UoB**	R	PU	36
D8.4	Report on the club of academic users	WP8	**UoB**	R	PU	36
D8.5	Report on the club of industrial users	WP8	**Inria**	R	PU	36
D8.6	Report on the use of Logipedia by certification bodies	WP8	**IMT**	R	PU	36
D8.7	Report on the club of users in education	WP8	**Unistra**	R	PU	36
D8.8	Report on the club of users in publishing	WP8	**ZIB**	R	PU	36
D2.5	Trace producing Automatic Theorem Prover, Satisfiability Modulo Theories solvers, and Coherent Logic Solver	WP2	**ULiege**	OTHER	PU	40
D6.3	Report on the integration of Matching Logic	WP6	**UAIC**	R	PU	42
D1.10	Export/import of B/Event B models from/in Rodin	WP1	**INPT**	OTHER	PU	48
D1.9	Harness in Atelier B to export proof terms and import lemmas from Logipedia	WP1	**ClearSy**	OTHER	PU	48
D2.6	Translate most traces to Dedukti	WP2	**IMT**	OTHER	PU	48
D2.8	Report on Proof Automation in Logipedia	WP2	**UIBK**	R	PU	48
D2.9	A posteriori verification of non trivial C programs in the Frama-C-WP platform	WP2	**CEA**	DEM	PU	48
D5.3	This deliverable describes the representation of major formal libraries developed in Task 4.	WP5	**FAU**	R	PU	48
D6.2	Report on the integration of Cubical Type Theory in Dedukti	WP6	**Inria**	R	PU	48
D6.4	Report on the integration of Minlog	WP6	**LMU**	R	PU	48
D6.5	Report on the integration of Mizar	WP6	**UBia**	R	PU	48
D6.6	Report on the integration of PVS	WP6	**Inria**	R	PU	48
D6.7	Report on the integration of Smart	WP6	**P&R**	R	PU	48
D6.8	Report on the integration of TLA$^+$	WP6	**Inria**	R	PU	48
D7.4	Implementation of a prototype automated alignment inference engine	WP7	**IMT**	DEM	PU	48
D7.5	Implementation of the alignment-based search service	WP7	**FAU**	DEM	PU	48
D7.6	Implementation of the alignment based proof-rewriting service	WP7	**Unibo**	DEM	PU	48

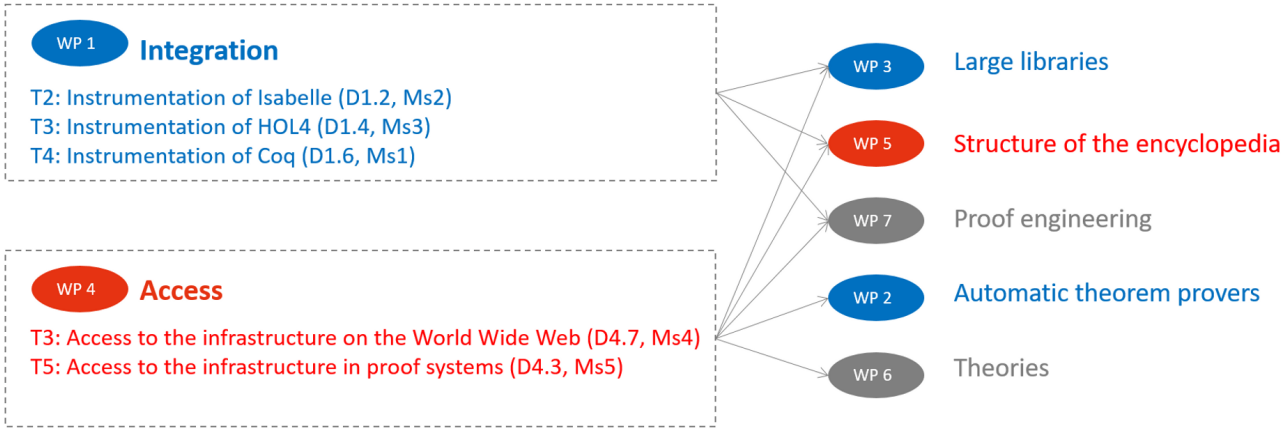
Timing of the different work packages and their components


Month
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
WP1
T1.1: Agda
T1.2: Isabelle
T1.3: HOL4
T1.4: Atelier-B
T1.5: Matita
T1.6: Coq
D1.1 D1.2 D1.3 D1.4 D1.5 D1.6 D1.7 D1.8 D1.9 D1.10
WP2
T2.1: ATP
T2.2: ATP2Ded.
T2.3: Ded2ATP
T2.4: Readiness
T2.5: End2end
D2.1 D2.2 D2.3 D2.4 D2.5 D2.6 D2.7 D2.8 D2.9
WP3
T3.1: Isa. AFP
T3.2: Isa. A&P
T3.3: GeoCoq
T3.4: Flyspeck
T3.5: CakeML
D3.1 D3.2 D3.3 D3.4 D3.5
WP4
T4.1: Archi.
T4.2: Web
T4.3: Syst.
T4.4: Search
T4.5: Dev.
D4.1 D4.2 D4.3 D4.4 D4.5
WP5
T5.1: Struct.
T5.2: F/W
T5.3: Ref. Ont.
T5.4: Ont. Repr.
D5.1 D5.2 D5.3
WP6
T6.1: CuTT
T6.2: K
T6.3: Minlog
T6.4: Mizar
T6.5: PVS
T6.6: Smart
T6.7: TLA+
D6.1 D6.2 D6.3 D6.4 D6.5 D6.6 D6.7 D6.8
WP7
T7.1: Found.
T7.2: Cases
T7.3: Autom.
T7.4: Search
T7.5: Rewr.
D7.1 D7.2 D7.3 D7.4 D7.5 D7.6
WP8
T8.1: Comm.
T8.2: Dissem.
T8.3: Train.
T8.4: Research
T8.5: Industry
T8.6: Certif.
T8.7: Educ.
T8.8: Pub.
T8.9: Pub. link
T8.10: Teach.
D8.1 D8.2 D8.3 D8.4 D8.5 D8.6 D8.7 D8.8 D8.9 D8.10 D8.11
WP9
T9.1: Sci.&tech.
T9.2: Sci.&tech.
T9.3: Adm.&Fin.
T9.4: Legal
D9.1 D9.2
Ms1
Ms2
Ms3
Ms4
Ms5
Ms6
general meetings

Relation between the components

Two work packages are critical for the success of the project, the work package 4, that aims at building the software infrastructure of Logipedia and the work package 1 that aims at populating it with proofs. All the other work packages depend of these two.

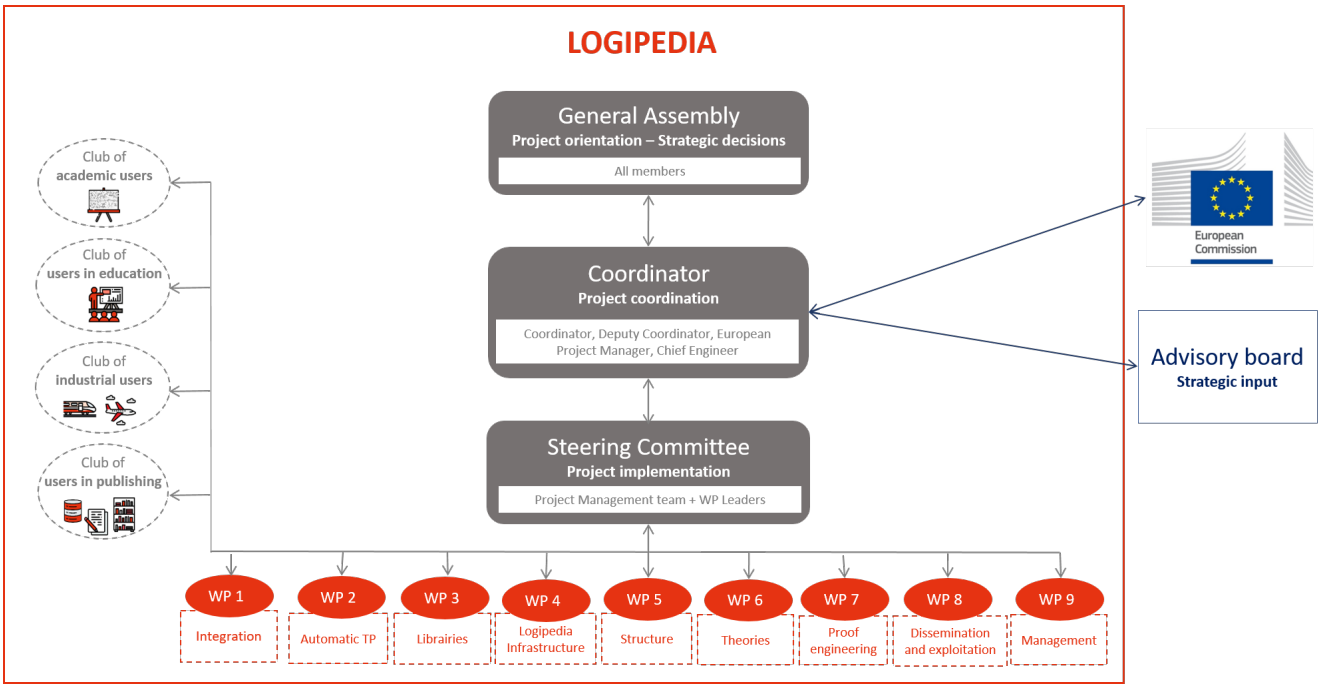
Finally all work packages contribute to just one goal: interoperability, sustainability, and cross-verification of formal proofs.



3.2 Management structure, milestones and procedures

The Logipedia consortium will gather twenty-eight beneficiaries and partners from eleven European countries for four years. The project management structure will be tailored to the specificities and needs of this large consortium and its ongoing network development.

Organisational structure



The project management team

- **The Coordinator** is responsible for the coordination of scientific and technical activities in order to meet the objectives set by the European Commission in the Grant Agreement. The coordinator

works closely with the work package leaders within the steering committee, in order to monitor the progress of the scientific and technical work and identify potential risks within each work package. The coordinator will daily collaborate with the European project manager in charge of the day-to-day management of Logipedia. The project will be managed by Gilles Dowek, permanent senior researcher at Inria Saclay. He will also chair the meetings of both the general assembly and steering committee.

- **The Deputy Coordinator** seconds the coordinator.
- **The European Project Manager** is a team member of the Technology Transfer and Partnership Office of Inria Saclay. He or she is in charge of all administrative, financial and legal management tasks as listed in WP9. The European project manager is the interface between the project and the European Commission as it represents the point of contact for the European Commission. The European project manager has the overall administrative and financial responsibility for the organisation and administrative and financial monitoring of the project.
- **The Chief Engineer** is an experienced research engineer from Inria Saclay and is responsible for ensuring the development and maintenance of tools at Inria Saclay and supervising the development tasks achieved at the other beneficiaries. The chief engineer will ensure the coherence of the Logipedia tools development, according to the defined schedule in the Grant Agreement.

Innovation management and intellectual property rights issues will be handled by the European project manager, supported by the experienced Technology Transfer and Partnerships Office of Inria Saclay. The project management team will establish appropriate policies and rules for the management of intellectual property rights for the knowledge developed within the project, as well as the identification of the opportunities for the exploitation of the project results in innovation activities. Issues related to innovation and/or intellectual property rights management will be tackled at every steering committee meeting.

The operational level

- **The Steering Committee**: The steering committee is composed of the Coordinator, the Deputy coordinator, the Chief engineer, the European project manager and the work package leaders. It is the main decision-making body of the project. It coordinates activities within and among the workpackages, evaluates and validates the progress of the project, assists the scientific coordinator in carrying out his tasks, monitors the technical direction of the project, approves all major technical decisions, approves all deliverables, approves all significant changes in the project work plan, reviews and amends the work plan. The Steering Committee is responsible for the implementation of decisions made by the general assembly. It can formulate proposal for changes in the description of action and the related consortium budget. The steering committee is chaired by the coordinator.
- **The Work Package Leaders**: The work package leaders are responsible for the monitoring and management of the activities and results within their work packages. In particular, work package leaders i) identify deviations from the project plan and report them to the steering committee, ii) manage and supervise the preparation of reports and their timely delivery, iii) control and monitor activities of tasks and regularly meet once per month with task leaders, iv) manage the information flow with other work packages via the steering committee.
- **The Task Leaders**: The task leaders are responsible for coordinating the scientific and technical work in their task and making the day to day technical decisions that solely affect their task. Inter-task decisions are coordinated with the work package leaders.
- **The Club Leaders**: The club leaders are in charge of disseminating of the results and tools developed by the Logipedia consortium in various communities. They organise the activity of the club. They give ongoing feedback to the consortium during the course of the project.

The strategic level

- **The General Assembly**: The general assembly is composed by all the members of the consortium, with each representative having one vote. Every new partner will have a voting right. The general assembly will gather at least once a year, and as many virtual meetings as needed. The general assembly is the main governance and ultimate decision-making body of the consortium. The general assembly must review the project progress, decide on contingency actions in case of deviations from the plan and take final decisions on policy and contractual issues and conflicts as requested by the steering committee.
- **The Advisory Board**: The advisory board is a consultation body to the steering committee and general assembly. It will bring external and non-legally binding perspective on the scientific and technical development of the project, ecosystem building and the future of the encyclopedia. The advisors of this board will attend the yearly general assembly plenary meeting and will be consulted on the strategy of the project. The advisory board should aim at representing the stakeholders of the Logipedia ecosystem without including any beneficiary or associate partner's employees. It will be composed of, among others, industrial and international academic partners (including non-European ones) appointed by the coordinator after consulting the steering committee. To start with, we suggest to include: June Andronick (Data61, Kensington NSW, AU), Denis Cousineau (Mitsubishi Electric R&D Centre Europe, FR), Thomas Letan (ANSSI, FR), Jacques Fleuriot (University of Edinburgh, UK), Natarajan Shankar (SRI, US), Aaron Stump (Iowa University, US), Laurent Voisin (Systerel, FR).

Internal communication and collaborative ecosystem

The consortium will make use of a number of project management tools, such as a visio conferencing tool, a project repository to have an updated account of the project's important documents, the progress of the work packages work and deliverables, all the advances in the project and all the meetings minutes, mailing lists, etc. that facilitate the smooth execution of the project. This collaboration environment will be provided by the coordinator of the project.

Work packages, chaired by work package leaders, will have monthly planned visio conferences and meetings as need by the work plan. Additional technical meetings may be set up by task leaders or individual partners. The steering committee will have monthly visio conferences and will meet twice a year. Dedicated working groups will be planned as needed according to the work plan. All meetings will be documented by minutes listing major decisions and action items.

The project management team will be in charge of all organisational issues in the general assembly meetings, supported by the local partner. The project will organise meetings of the general assembly at least once a year. To equally share travel costs among partners, physical meetings will be located by rotation at partners' locations. Project review meetings will be done on a regular basis according the Grant Agreement provisions.

Quality Control

Quality Control will be part of the Logipedia Quality Plan, which will be a deliverable.

These guidelines adopted by Logipedia will ensure that all participants know who, how and when to act regarding to documentation of project activities, periodic reporting, preparation of financial statements, approval and submission of deliverables, and risk management.

The implementation of the quality management will include the following phases:

- identification of the procedures needed,
- planning, design and development of the procedures and the forms to be implemented,
- development of an implementation guide for all the partners.

Quality assurance is the joint responsibility of all partners and will be applied at all levels of the project's activities. The goal is to ensure the detection of mistakes and deviations in the project's life cycle

as early as possible in order to apply the necessary corrective actions or contingency plans systematically. Every contractual deliverable, prior to its submission to the Commission, will be the subject of a peer review by two persons not directly involved in either the subject matter or the creation of that deliverable. The coordinator will make a final check of the deliverable for consistency and readability.

Decision-making Process

Our approach for the decision-making process is to locate the decision as close as possible to the level responsible for the execution (from task to general assembly level). Decisions are managed within frequent project meetings, either on-site or via teleconference. Decisions can be also managed by consultation. If voting is needed, the agenda should clearly indicate this fact. Quorum and voting rules will be defined in the Consortium Agreement. Decisions are binding once the relevant part of the meeting minutes has been accepted. Any changes to the project plan and scope must be reviewed and approved by all levels of project management, before proposing these changes to the steering committee and any modifications will be considered rejected, after rejection on any of these involved levels.

Another guiding principle is to avoid conflicts. Nevertheless, should one arise, a conflict resolution will be ready to be put in place to deal with it accordingly. The conflict resolution foresees that each conflict will be mediated, solved or decided at the lowest level possible. Attempts to solve issues within the consortium will be carried out in increasing order of authority first at task level (management of task leader), work package level (management of work package leaders), and then following the management bodies till the general assembly.

Before the Logipedia project starts, the consortium partners will sign a Consortium Agreement wherein roles, responsibilities and mutual rights and obligations will be defined. It will be in complete accordance with the rules of the Grant Agreement and will adopt the recommended guidelines laid down by the European Commission.

Monitoring and reporting

Internal reporting

The project management team continuously monitors the project plan with its milestones. Each work package leader will be responsible for the correct execution of the implementation plan for the corresponding work package. In terms of reporting, this means the work package leaders will be in charge of gathering the information related to their own work packages.

Regular audio-conferences of the Steering Committee are foreseen, which allows work package leaders to identify risks and discuss them together. This ensures that management (coordination, European project manager) is aware of potential problems and deviations and can initiate countermeasures long before a situation becomes critical. This ensures to spot the blocking points and implements the solution at the right time.

The following project meetings are planned:

- one kick-off meeting at the beginning of the project,
- one Steering Committee meeting every two months,
- one physical Steering Committee meeting every six months (reporting on work progress), which will be hosted by each work package leader. We will couple these meetings with other ones so to reduce travels,
- optional WP meetings whenever needed,
- four project meetings are planned, jointly with a General Assembly, Advisory Board, Steering Committee and the four Clubs meetings.

The kick-off meeting will serve to launch the project, create a benevolent, trustful, and encouraging atmosphere and adjust expectations.

Reporting to the European Commission

The Logipedia consortium will follow the mandatory reporting period required by the European Commission.

The project management team will provide the necessary templates in order to achieve the reporting in due time. Work package leaders will be asked to gather the relevant information provided by the task leader regarding their work package and to summarise in order to be reviewed by the steering committee. It will then be treated by the coordinator and European project manager and sent to the European Commission.

Significant risks and associated contingency plans

Scientific risks

Description of the risk	**Work packages involved**	**Proposed measure of mitigation**
We do not succeed to express some theories in Dedukti. (Probability: medium. Severity: low.)	WP6	We have carefully divided the systems into two groups: those that do not present risks (WP1) and those that do (WP6). The success we got with the theories implemented in systems of the first group gives us confidence that the theories implemented in those of the second can also be expressed in Dedukti. If one of them happens to be more difficult, we can still build a large encyclopedia with the systems of the first group. We can also extend Dedukti so that it can express more theories, as we have already done in the past.
Some libraries require too much time and memory to be expressed in Dedukti (Probability: medium. Severity: low.)	WP3	There are several ways to mitigate this risk: optimise the representation of data (sharing, elimination of redundancies, etc.), use faster and larger computers. This may also mean that some tasks of this work package are premature and that we have to wait for faster computers, that Moore's law will provide.
No adoption from the community. (Probability: low. Severity: high.)	All	The community may fail to adopt Logipedia for several reasons. Because of a problem of design of Logipedia, in which case we will have to understand what needs to be changed in a second version. It may be because the Logipedia community is too small. This explains that we have decided to include twenty-eight partners in the project. It may also be because of an insufficient dissemination activity. This is why we propose to create the four clubs of users and we will devote time and energy to the animation of these clubs, together with other dissemination activities, such as summer schools and conferences.

Management risks

Brexit disrupts the project (Probability: low. Severity: medium.)	All (specially WP6, WP7)	As we have two partners from the United Kingdom, Brexit could be a risk for our project. Yet, we are quite confident that scientific cooperation will continue after Brexit and that the British partners of the project will continue to be part of it. As this project is submitted during H2020 and nothing changes until the end of 2020, Logipedia is safe for the start of the project. From 2021 onwards, we can hope some agreement will be concluded as the UK already made public its willingness to maintain collaborations. If it were not the case, we would have to reallocate the impacted tasks to other partners.
One partner leaves (Probability: low. Severity: depends on the partner.)	All	The impact of such a default of one partner of course depends on the partner. But, during the preparation of this project, we have been careful to develop an atmosphere of trust and solidarity between the partners. If this happened we would need to adapt the objectives of the work package the partner was supposed to contribute to.
Difficulty to find people (doctoral students, post-docs, engineers, etc.) (Probability: medium. Severity: low.)	All	This project will require hiring a fair number of people. This may be difficult in some European countries. If this happens we will use the size of the network to find candidates in other countries to meet the objectives of the project.

Milestones

As show by the Pert diagram above, two work packages, are critical, as many others depend on them: work package 4 “Access to the encyclopedia” that is focused on the development of the infrastructure itself and work package 1 “Integration” that focuses on populating this infrastructure with theories and proofs. All work packages depend on work package 4, and work packages 3 “Large libraries”, work package 5 “Structure of the encyclopedia”, and work package 7 “Proof engineering” depend on work package 1. As a consequence five of our milestones are completions of the key tasks of work packages 4 and 1. No milestone is the completion of a task of the two, more risky, joint research activity work packages: work package 6 “Theories” and work package 7 “Proof engineering”, which is a sign of robustness of the project.

1. **Milestone Ms1 (Month8) Instrumentation of Coq.** Integration of the Coq standard library in Logipedia.
2. **Milestone Ms2 (Month12) Instrumentation of Isabelle.** Integration of the Isabelle standard library in Logipedia.
3. **Milestone Ms3 (Month12) Instrumentation of HOL4.** Integration of the HOL4 standard library in Logipedia.
4. **Milestone Ms4 (Month12) Opam for Logipedia.** Release of a package distribution system for Logipedia.
5. **Milestone Ms5 (Month18) Logipedia platform.** Release of the web interface to the Logipedia platform.

6. **Milestone Ms6 (Month36) First overall evaluation of the project.** The delivery of the reports of the clubs of users allow to define the strategy for the long term exploitation of Logipedia.

3.3 Consortium as a whole

Building an infrastructure like Logipedia requires that most of the major proof systems contribute to the project. We currently have eighteen out of the twenty-four systems presented in Figure 1, and fourteen out of the sixteen European ones. The only reason why we do not have them all is because of budget constraints.

This explains why the consortium has to be large. We have twenty-eight partners in the consortium which is almost all the groups working on formal proof technology in Europe. A smaller consortium would not reach the universality which is at the heart of the project. Being such a large consortium is a strategic decision.

Each partner brings a different expertise to the project. Eighteen partners provide their expertise for one specific system. Eight bring their expertise on different aspects of automated theorem proving. Four bring their expertise of some specific library. Eleven, bring their expertise on a transversal issue: access (two partners), proof engineering (nine partners), and publishing (one partner).

Finally, there are seven industry partners among the twenty-eight partners. One is an industrial research centre and six are enterprises. Three are transversal and five are focused on one industrial sector: transportation, health care, energy, security, and education. These six enterprises are small and medium-sized enterprises. In contrast the enterprises in the club of industrial users are of very different sizes, from small and medium-sized enterprises to very large enterprises such as Alstom or Siemens.

Partner	**Systems**	**ATP**	**Libraries**	**Transverse**	**Industrial sector**
Institut National de Recherche en Informatique et Automatique	HOL Light, PVS, Coq, TLA$^+$, HoTT		Flyspeck	Proof engineering	
Université de Strasbourg			GeoCoq	Proof engineering	
Institut National Polytechnique de Toulouse	Rodin			Proof engineering	
Universität Innsbruck	Mizar	Using FOL proofs		Proof engineering	
Université de Liège		SMT reasoners and proofs			
Alma Mater Studiorum — Università di Bologna	Matita, Coq			Proof engineering	
Matematički fakultet, Univerzitet u Beogradu		Coherent logic		Proof engineering	
Technische Universität München	Isabelle/HOL		AFP, Probability / analysis		
Technische Universiteit Delft	Agda				

Université Paris-Saclay		Using SMT proofs		Proof engineering	
Friedrich-Alexander Universität Erlangen-Nürnberg				Access, Proof engineering	
University of Leeds	HoTT, Coq			Proof engineering	
Göteborgs Universitet	Agda				
Chalmers Tekniska Högskola	HOL4		CakeML		
Ludwig-Maximilians-Universität München	Minlog				
Institut Mines-Télécom	Atelier B, FoCaLiZe, Coq	ATP proofs for Dedukti		Proof engineering	
Uniwersytet w Białymstoku	Mizar		MML		
OCamlPro		SMT			
ClearSy	Atelier B				Transportation, Energy
University of Birmingham	HoTT		UniMath		
Commissariat à l'Energie Atomique et aux Energies Alternatives	Why3, Frama-C	ATP cooperation			Energy, transportation, security
Duale Hochschule Baden-Württemberg		FOL reasoners and proofs			
Edukera				Access	Education
MED-EL Elektromedizinische Geraete GmbH					Health care
Prove & Run	ProvenTools				Security
Konrad-Zuse-Zentrum für Informationstechnik Berlin				Publishing	
Universitatea Alexandru Ioan Cuza din Iasi	K Prover				
Runtime Verification SRL	K Prover				

3.4 Resources to be committed

Cost breakdown per activity: 82% of the budget funds the seven scientific and technological work packages (16%, 14%, 9%, 7%, 8%, 17%, and 11%), 12% is for the work package "Dissemination, communication, and exploitation", and 6% for the work package "Management".

Cost breakdown of direct cost per type: 87% of the budget is for personnel, 12% for missions, and 1% for subcontracting.

Note that 48% of the budget is allocated to French partners. This was difficult to avoid because the coordinating site is French and also two large formal proof communities, the B community and the Coq community, are French. Nevertheless, the consortium is very international, as it counts eleven countries,

WP	Title	Inria	Unistra	INPT	UIBK	ULiege	Unibo	UBel	TUM	TUDelft	USaclay	FAU	ULeeds	UGot	Chalmers	LMU	IMT	UBia	OcamlPro	ClearSy	UoB	CEA	DHBW	Edukera	MED-EL	P&R	ZIB	UAIC	RV	total
WP1	Integration	6		31			11		5	*16*				4	16		1			14										**104**
WP2	Automatic theorem provers				6	54		12			*11*						9		14			18	2								**126**
WP3	Large libraries	20	18						*29*						12																**79**
WP4	Access	*54*																	6												**60**
WP5	Structure of the encyclopedia			8			5				40	*25*																			**78**
WP6	Theories	87			*11*								3			16		70			3				5	11		6	7	**219**
WP7	Proof engineering	8	25		8		14	*20*			9	11	17			1	11													**124**
WP8	Dissemination	6	2			*8*											2			2	4			12			14			**50**
WP9	Management	*44*																												**44**
totals		**225**	**45**	**39**	**25**	**62**	**30**	**32**	**34**	**16**	**60**	**36**	**20**	**4**	**28**	**17**	**23**	**70**	**20**	**16**	**7**	**18**	**2**	**12**	**5**	**11**	**14**	**6**	**7**	**884**

Efforts in PM; WP lead efforts light gray italicised

Table 1: Work Packages

and most European centres of excellence in formal proofs are represented.

In-kind contributions on the beneficiary's premises

As declared in the online proposal submission form, the section 3 "budget", the Logipedia consortium has declared a number of in-kind contributions. That is why the requested EU Contribution (K) of our consortium is a smaller amount than the max EU contribution (J). The community supporting the Logipedia project is wide but each partner has been thoroughly selected and is necessary for the success of the project. A budget of 5 millions euros is too limited to support all the efforts of the community. The partners are so committed to the successful implementation of the project that they have agreed in-kind contributions. The difference between the total eligible costs for the project and the requested will therefore be supported by some partners as an-kind contribution.

Inria is going to contribute 67 PM in-kind for the whole duration of the project. This contribution is already funded by Inria.

Unistra is going to contribute 7 PM in-kind for the whole duration of the project. This contribution is already funded by the University of Strasbourg.

INPT is going to contribute 28 PM in-kind for the whole duration of the project. This contribution is already funded by the INPT.

UIBK is going to contribute 2 PM in-kind for the whole duration of the project. This contribution is already funded by the University of Innsbruck.

ULiege is going to contribute 9 PM in-kind for the whole duration of the project. This contribution is already funded by the University of Liège.

USaclay is going to contribute 12 PM in-kind for the whole duration of the project. This contribution is already funded by Université Paris-Saclay.

ULeeds is going to contribute 5 PM in-kind for the whole duration of the project. This contribution is already funded by the University of Leeds.

LMU is going to contribute 7 PM in-kind for the whole duration of the project. This contribution is already funded by the Ludwig-Maximilians-Universität München.

IMT is going to contribute 14 PM in-kind for the whole duration of the project. This contribution is already funded by Institut Mines-Télécom.

UBia is going to contribute 22 PM in-kind for the whole duration of the project. This contribution is already funded by the University of Bialystok.

DHBW is going to contribute 2 PM in-kind for the whole duration of the project. This contribution is already funded by the Duale Hochschule Baden-Württemberg.

Including these in-kind contributions the overall budget of the project is 6.1 millions euros (19% of which is these in-kind contributions).

Justification for mission costs

Details for participants whose Other Direct Costs (sum of the costs for' travel', 'equipment', and 'goods and services') exceeds 15% of their Direct Personal Costs (according to the budget table in section 3 of the proposal administrative forms).

ULiege	Cost (EUR)	Justification
Travel & subsistence for trans-national access (if applicable)	15,000	The PhD student co-supervised with **DHBW** will regularly visit **DHBW** for periods that will span several weeks. The rest is dedicated for presenting results at conferences and inviting external (non-EU) experts on some specialised aspects of WP2.
Other travel	100,000	**ULiege** is leading WP8, and will centralise the budget for organising the yearly Logipedia conference. This money is allocated to the organisation of the four Logipedia conferences, at an approximate cost of 25k per conference, for about 100 people attending.
Equipment		
Other goods and services		
Total	115,000	

UBel	Cost (EUR)	Justification
Travel & subsistence for trans-national access (if applicable)	17,150	**UBel** is part of WP2 and WP7. Five researchers from University of Belgrade will visit partner sites and regular project meetings. Since they are going to work jointly with researchers from University of Strasbourg, most visits will be made to Strasbourg. The total budget for missions allocated to **UBel** is aligned with other participants in the project, but, since Serbia is a low income country, researcher salaries are lower and mission budget consumes a higher percentage of the allocated budget.
Other travel		
Equipment		
Other goods and services		
Total	17,150	

IMT	Cost (EUR)	Justification
Travel & subsistence for trans-national access (if applicable)	21,700	Burel is leading **T2.2** on reconstructing Dedukti proofs from proof traces. This task depends on **T2.1** on producing such traces. He will therefore need to meet the participants of these tasks, in particular the PhD student co-supervised by **ULiege** and **DHBW**. Dubois is leading the dissemination task **T8.6** on promoting the use of Logipedia by certification authorities, in particular by European public institutions. This task will require travels to meet these institutions. Furthermore, she leads and participates in **T1.4** on the translation of B models into Dedukti, with **INPT** team. Mission funding will help her to travel to meetings and workshops with this team.

Other travel		
Equipment		
Other goods and services		
Total	21,700	

UoB	Cost (EUR)	Justification
Travel & subsistence for trans-national access (if applicable)	14,000	**UoB** participates in **T6.1**, which is led by Barras and a PhD student in Paris.
Other travel	25,000	**UoB** leads two dissemination tasks: **T8.3** on training and **T8.4** on academic users. Both of these tasks will require frequent travel to meetings and workshops. A large portion of mission funding is for supporting disadvantaged participants at training events and workshops.
Equipment		
Other goods and services		
Total	39,000	

DHBW	Cost (EUR)	Justification
Travel & subsistence for trans-national access (if applicable)	17,000	**DHBW** applies for compensation for necessary travel costs. It provides most its personnel as an in-kind donation to support the project. In particular, it will support WP2 (instrumenting an ATP to provide suitable proof traces) and WP8 (in particular co-supervising a PhD-candidate with **ULiege**). Since **DHBW** does not apply for significant other funds, the mission budget is nearly all of the total budget allocated to it.
Other travel		
Equipment		
Other goods and services		
Total	17,000	

UAIC	Cost (EUR)	Justification
Travel & subsistence for trans-national access (if applicable)	10,000	**UAIC** is leading the task **T6.2** from the work package WP6. Since **T6.2** depends on the tasks **T2.1** and **T2.2**, a part of the budget for mission is dedicated to visit the partners leading these tasks. The rest of the mission budget is dedicated to participating Logipedia meetings/workshops and for disseminating results at conferences. The total budget for missions allocated to **UAIC** is aligned with other participants in the project, but, since Roumania is a low income country, researcher salaries are lower and mission budget consumes a higher percentage of the allocated budget.
Equipment		
Other goods and services		
Total	10,000	