VIENNA SCIENCE
AND TECHNOLOGY FUND

Proposal for Project Funding
Information and Communication Technology 2022

1. Short Proposal

Basic Information

Project ID:	ICT22-026
Project title:	Leveraging Modern Automated Deduction for Program Synthesis
Acronym:	LEMONADE
Keywords:	proofs-as-programs, deductive synthesis, automated theorem proving, logic in computer science, proof theory
Coordinating Principal Investigator:	Florian Zuleger, TU Wien
Indicative duration:	48 months
Indicative budget (in kâ‚¬):	600.0
Project type:	WWTF project

Scientific disciplines relevant to the project:

Main scientific discipline:		Other scientific discipline:		Other scientific discipline:	
102031 - Theoretical computer science	40 %	101013 - Mathematical logic	40 %	102022 - Software development	20 %

Project summary

Program synthesis is a long-standing problem that crosses many areas of computer science and mathematics. Program synthesis holds the promise to increase the reliability of software systems (because the synthesized program are correct by construction) and to increase programmer productivity by enabling programmers to focus on high-level ideas instead of low-level details.

Traditionally, program synthesis has been formalized as a problem in deductive theorem proving: Given a logical formula that states that for all inputs there exists an output such that the required specification holds, one searches for a constructive proof of this formula and then extracts a program from this proof.

However, progress on deductive synthesis has been slow because of a lack of automated and scalable provers. We believe that the impressive progress in automated theorem proving finally has made automated proof search feasible. Thus, the main goal of our project is to leverage state-of-the-art first-order automated theorem provers (such as Vampire) for program synthesis, demonstrating the feasibility of deductive synthesis via automated theorem proving.

Specifically, we plan to 1) adapt and extend the theory on program extraction to the specifics of modern automated theorem provers, and to 2) implement a proof-of-concept tool that demonstrates the feasibility of our approach.

Introduction, background, and state of the art

Programs are usually first written and then tested/verified. In contrast, program synthesis allows a system developer to derive a program in one step. At the same time, program synthesis holds the promise of freeing the programmer from low-level details and of enabling the programmer to focus on the high-level intent. Program synthesis is a long-standing problem that crosses many areas of computer science and mathematics including programming languages, AI, formal methods and proof theory. Traditionally, program synthesis has been formalized as a problem in deductive theorem proving [8]: Given a logical formula that states that for all inputs there exists an output such that the required specification holds, one searches for a constructive proof of this formula and then extracts a program from this proof. A classic route to program extraction is the Curry-Howard correspondence [4] (also known as proofs-as-programs) where one exploits the equivalence between proofs in intuitionistic logic and functional programs in order to directly extract a functional program from an intuitionistic proof. The Curry-Howard correspondence denotes a fundamental relationship between two seemingly unrelated formalisms, namely proofs systems on one hand and computational models on the other hand. The Curry-Howard correspondence has been a major driving force for research spanning functional programming languages, type theory, proof systems and constructive logic. However, progress on the deductive synthesis problem has been slow for many years because of a lack of automated and scalable provers. Instead, a recent variant of the program synthesis problem has enabled dramatic progress (we refer to the special issue [2] and the workshop https://simons.berkeley.edu/workshops/tfcs2021-1 for an overview): The logical specification is accompanied by a syntactic template that constrains the space of possible implementations. The synthesis procedure then searches for/enumerates candidate implementations and either rejects these candidates by finding violating input/output pairs or accepts the generated program by verifying their correctness. The syntactic approach is very versatile and synthesis approaches have been explored that solely start from input/output examples or that mine possible program fragments from public code repositories. While the reported progress is impressive, we believe that the traditional deductive approach to synthesis has been unjustly neglected.

Research questions, hypotheses and objectives

The main goal of this project is to connect the classical approach to deductive synthesis - proofs-as-programs [4] -, with modern automated theorem provers, in particular first-order solvers (including SMT solvers), which have made impressive progress over the recent years (as evidenced by yearly competitions). We believe that this approach offers the following advantages: 1) First-order solvers are particularly suitable to solving quantified formulas as needed for synthesis. 2) Proofs-as-programs represents a longstanding approach to program extraction. That is to say, much groundwork for program extraction from proofs has been laid, we just need to adapt and extend this theory to the specifics of modern automated theorem provers. 3) We are able to take advantage of the highly sophisticated and optimized proof search strategies of first-order solvers instead of needing to define and implement ad-hoc search strategies.

We believe the proposed research is very timely, because improved hardware and the progress in automated theorem proving have finally made automated proof search feasible. We will advance the above main goal by two work packages (WPs):
(WP1) We will lay the theoretical groundwork for program extraction with automated theorem provers. The challenge here is that the proofs-as-programs approach requires proofs in intuitionistic logic, while state-of-the-art theorem provers utilize resolution which is an inherently classical technique. For this, we will adapt and extend techniques from the literature that connect the Curry-Howard correspondence to classical logic.
(WP2) We will implement a proof-of-concept tool that demonstrates the feasibility of our approach. This tool will take first-order specifications as an input, and leverage automated theorem provers to output programs that are correct by construction. We will further experiment with techniques that relate to modern syntax-guided approaches to synthesis.

Expected results, novelty, and relevance

The final goal of our project is to leverage state-of-the-art first-order automated theorem provers, in particular Vampire [6], for program synthesis. This is motivated by the recent and prospective advancements in their capabilities. We will lay the necessary theoretical foundations as well as implement a prototype system to leverage state-of-the-art classical theorem provers for program extraction. At the current time no such system exists. Thus the proposed research promises to fill a longstanding gap in the state-of-the-art.

Much of the theoretical groundwork for implementing such a system is already present, but is scattered and mostly limited to theoretical settings. We sketch here the two main directions that connect the Curry-Howard correspondence to classical logic: 1) The Curry-Howard correspondence can be extended to classical logic using control operators [5], Parigot's lambda,mu-Calculus [9] or CPS-translation. However in this process we must in general give up on some desirable properties, at the very least the realizability of 'exists' and 'or'. 2) Under certain conditions, classically valid proofs can be transformed into intuitionistic proofs, e.g. proofs of Pi2-sentences in Peano Arithmetic [3] (this point is of particular interest for SMT). We will adapt and specialize these techniques from the literature to the proofs produced by automated theorem provers such as Vampire: A) We will develop the first program extraction calculi for resolution resp. superposition proofs. In particular, we need to pay careful attention to the normal-form transformation - a pre-processing step - in automated theorem provers, which also involve equivalences that are only classically valid. B) We will identify assumptions under which our adaptions of the techniques 1) and 2) guarantee that such a program extraction is possible. We are not aware of related results in the literature.

Methods and feasibility

We now give details on WP1:
1) Parigot's lambda,mu-calculus [9] gives a direct translation of classical proofs with datatypes and is essentially ready for implementation. Unfortunately, we lose the realizability of 'exists' and 'or' since non-constructive tautologies like 'A or ¬A' are classically provable. In practice it is sometimes possible to eliminate the non-constructive reasoning steps from proofs [7]. We aim to establish stronger theoretical guarantees on when this is possible but also evaluate this approach when realizability is not ensured by a theorem.
2) Sometimes classical validity guarantees intuitionistic validity, most famously for Pi2-formulas in Arithmetic [3]. There have been many variations of this result, most notably by Schwichtenberg's group [1]. We will collect, adapt and extend these results with regard to our concrete setting. This is particularly promising when applied to SMT, interpreted as a restricted subproblem of the full first-order case [10], or when considering altered specifications (see next point).
3) We investigate a new direction: We aim for a translation between provers rather than proofs. E.g., we study under what conditions classical proofs of a modified input formula allow to extract intuitionistic proofs of the actual input. A motivating example is actually the reverse direction: given an intuitionistic prover one can establish classical provability via double negation translation.

Using the results of (WP1) we will implement a synthesis system on top of Vampire (WP2), targeting a suitable functional programming language. We will experiment with various benchmark problems and compare our results with the SyGuS-competition. We will use ideas from syntax-guided synthesis in order to restrict and guide proof search: We will experiment with different formulations of arithmetic axioms, induction axioms and redundant axioms. We will further examine if proof search is possible when only input/output pairs are given.

Role of team members and collaborative aspects

Florian Zuleger (PI) works at the intersection of logics and automated decision procedures and their application in verification and static analysis. His knowledge about automated decision procedures as well as about the interest of the community in verification/synthesis put him in an ideal position to run the project. Initial results as part of the master thesis of Alexander Pluska make him confident about the success of the project.

Alexander Pluska is a prospective PhD student and brings a firm background in both Mathematics and Computer Science, which is a necessity for the project. This comprises the mathematical rigour to study

the relevant calculi but also experience with the more applied facets of automated reasoning. His master's thesis crosses many aspects of WP1, in particular laying a foundation for Task A.4.

We will closely collaborate with Laura Kovacs at TU Wien, who is one of the developers of Vampire, and who has expressed great interest in the project.

Potential for interdisciplinarity and application to other research fields

Traditional quality assurance techniques rely on first building the desired system and then testing/verifying the system. In contrast, correct-by-construction approaches seek to develop a system that continues to be correct, from the initial design to the final product. Advances in deductive synthesis promise to increase the applicability of the correct-by-construction approach, leading to higher reliability in software development.

Automated theorem proving has made impressive progress in recent years, demonstrated in yearly solver competitions. We will submit our benchmarks to these competitions in order to interest solver developers in improving their tools wrt the specific needs of deductive synthesis.

We expect that a successful prototype for program extraction with automated theorem provers will lead to a revival of proof-as-programs for automated deductive synthesis, inspiring more members of the community to investigate this approach.

Potential medium-term economic / societal benefits

As a correct-by-construction methodology, program synthesis will increase the reliability of important safety-critical systems such as avionics, cars, medical devices, etc.

Enabling programmers to focus on high-level ideas instead of low-level details will increase programmer productivity.

Non-expert/ end-user programmers are also expected to promise from automated synthesis by only needing to communicate their intent to the synthesis engine (instead of writing a whole program).

Key references

[1] Ulrich Berger, Wilfried Buchholz, and Helmut Schwichtenberg. Refined program extraction from classical proofs. 2002.

[2] Dana Fisman, Rishabh Singh, and Armando Solar-Lezama. Special issue on syntax-guided synthesis preface. 2022.

[3] Harvey Friedman. Classically and intuitionistically provably recursive functions. 1978.

[4] Jean-Yves Girard, Paul Taylor, and Yves Lafont. Proofs and types. 1989.

[5] Timothy G Griffin. A formulae-as-type notion of control. 1989.

[6] Laura Kovács, Andrei Voronkov. First-Order Theorem Proving and Vampire. 2013

[7] Yevgeniy Makarov. Practical program extraction from classical proofs. 2006.

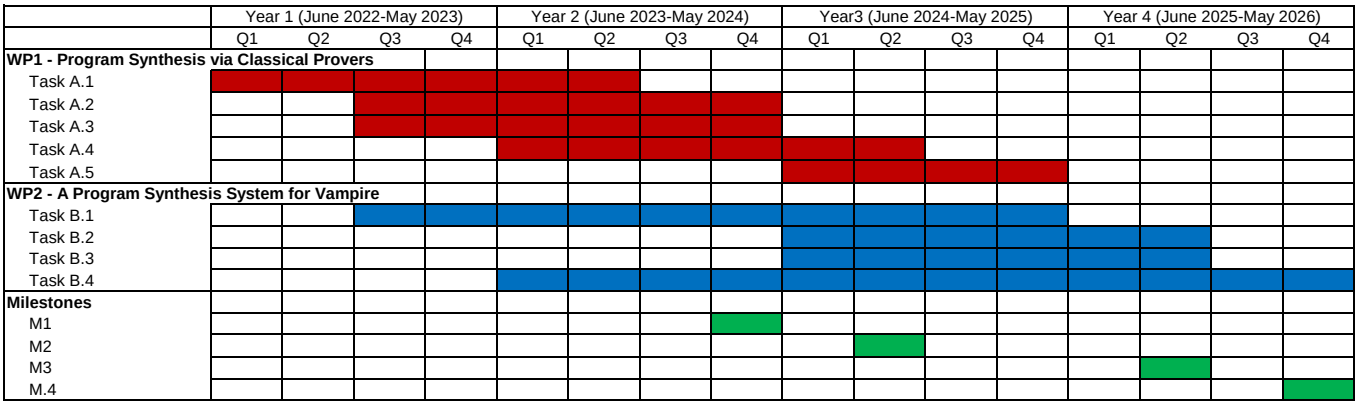
[8] Zohar Manna and Richard Waldinger. A deductive approach to program synthesis. 1979.

[9] Michel Parigot. lambda,mu-calculus: an algorithmic interpretation of classical natural deduction. 1992.

[10] Andrew Reynolds, Morgan Deters, Viktor Kuncak, Cesare Tinelli, and Clark Barrett. Counterexample-guided quantifier instantiation for synthesis in SMT. 2015.

WWTF proposal form: "Information and Communication Technology Call" 2012

PROJECT FLOWCHART



Short description of Tasks

Task A.1: Designing a calculus with control structures suitable for program extraction from Vampire proofs
Task A.2: Establishing under what conditions non-constructive applications of control structures can be eliminated
Task A.3: Collecting characterizations of specifications for which constructiveness of the extracted program can be guaranteed
Task A.4: Examining transformation procedures for input specifications
Task A.5: Evaluating the applicability, robustness and potential shortcomings of the developed mechanisms
Task B.1: Implementing the findings of WP1 into a software system
Task B.2: Utilizing syntactic context for improved proof search
Task B.3: Extending the system to restricted specifications, e.g. input/output pairs
Task B.4: Extensive testing and benchmarking

Staff:

Florian Zuleger (PI)
PhD1 (Alexander Pluska, 48 months)
PhD2 (N.N., 48 months)

Milestones:

M1: Extraction of constructive programs from classical proofs
M2: Classical provers for program synthesis via modified specifications
M3: Fully working implementation of theoretical findings
M4: A usable system for program synthesis built on top of the Vampire theorem prover

Workplan:

The PI Florian Zuleger will be involved in all tasks and lead A.5, B.1, B.4
PhD1 will lead tasks A.1, A.2, B.2 and be involved in A.5,B.1,B.4
PhD2 will lead tasks A.3, A.4, B.3 and be involved in A.1,A.5,B.1,B.4

ethical approval is not necessary.

Budget (in kEUR)

Institution	Personnel costs	Non-personnel costs	Total direct costs	Overheads	Funding applied for	% per institution
TU Wien	500.00	0.00		100.00	600.00	100.00 %
Total funding applied for per cost category	500.00	0.00	500.00	100.00	600.00	
% of total direct costs	100.00	0.00				

Budget per region (in kEUR)

Region	Funding applied for	% per region
Vienna	600.00	100.00 %
Other	0.00	0.00 %

Explanation of cost planning

We plan to hire two PhD students (with 40h contracts) for the duration of the whole project (48months). The tasks of the PhD students will be the development of the theoretical foundations as well as the implementation of a prototype system. As indicated in the Gantt chart, the two PhD students will pursue different approaches to program synthesis. In year 1 and 2, they will mainly explore proofs-as-program approaches with classical provers. In year 3 and 4, they will focus on integrating ideas from syntax-guided synthesis and also study SMT-based synthesis.

Project team

Coordinating Principal Investigator

Name:	Mr. Florian Zuleger
Highest academic title:	Assistant Prof. / Associated Prof. / Univ.-Ass./ao.Univ.-Prof./Univ.-Doz
Staff category:	Senior personnel
Employment at current institution by the time of submission of the proposal:	permanent
Year of birth:	1984
Date of doctorate:	11.05.2011
ORCID/Researcher ID:	0000-0003-1468-8398
Affiliation:	TU Wien
Region:	Vienna
Institute/Department/Research Group:	Institut für Logic and Computation / Formal Methods in Systems Engineering
Address:	Favoritenstraße 9-11
Zip code / city / country:	1040 / Wien / AT
E-mail:	florian.zulegr@tuwien.ac.at
Telephone:	+43 1 58801 18449
Homepage of Institute/Research Group:	https://forsyte.at/
Personal Scientific Website:	https://forsyte.at/people/zuleger/

Scientific Expertise:	Program Analysis, Verification, Formal Methods, Logic in Computer Science

BIOGRAPHICAL SKETCH – Florian Zuleger

PERSONAL INFORMATION

Name: Florian Zuleger
Researcher unique ID(s) ORCID: 0000-0003-1468-8398
Webpage: http://forsyte.at/people/zuleger

HIGHER EDUCATION

2017	Habilitation, TU Wien, Austria
2011	PhD Computer Science, Supervisor: Helmut Veith, TU Wien, Austria
2008	Diploma Mathematics, TU Munich, Germany

APPOINTMENTS/ POSITIONS

2017 – present	Associate Professor at TU Wien
2013 – 2017	Assistant Professor at TU Wien
2011 – 2013	PostDoc at TU Wien
2007 – 2011	University Assistant at TU Munich, TU Darmstadt, TU Wien

FELLOWSHIPS AND AWARDS

2021	Invited talk, Automated termination and complexity analysis, Joint CALCO and MFPS Special Session on Termination Analysis and Synthesis
2021	Bill McCune PhD Award in Automated Reasoning for my student Jens Pagel
2021	Invited to submit to the special issue of ACM Transactions on Programming Languages and Systems (TOPLAS) for selected papers of ESOP 2021
2019	Invited to submit to the special issue of Software Tools for Technology Transfer (STTT) for selected papers of TACAS
2018, 2020	Invited to submit to the special issue of Formal Methods in Systems Design (FMSD) for selected papers of FMCAD
2015	Best paper award for “Verification of Asynchronous Mobile-Robots in Partially-Known Environments” at Principles and Practice of Multi-Agent Systems (PRIMA)

SELECTED MEMBERSHIPS

Since 2020 Founding Member and Chief Financial Officer of the FMCAD Association (formally FMCAD – Verein zur Organisation der Konferenz "Formal Methods in Computer-Aided Design")

SELECTED THIRD PARTY FUNDS/ONGOING PROJECTS

2021 – 2026	Doctoral Program LogiCS@TUWien, Horizon 2020, H2020-MSCA-COFUND-2020, vice chair for research and training, 5.333.760€
2015 – 2018	SHiNE: Systematic Methods in Systems Engineering, National Research Network (NFN) - Austrian Science Fund (FWF), Task Leader, 4,2 Mio €

SELECTED COLLABORATION PARTNERS

- Georg Moser, Complexity Analysis of Purely Functional Data Structures, University of Innsbruck, Austria
- Tomas Vojnar, Shape Analysis for Low-level Programming Languages, Brno University of Technology, Czech Republic
- Radu Iosif, Separation Logic with Inductive Definitions, CRNS, VERMIAG, Grenoble, France

PROFESSIONAL AND SCHOLARLY ACTIVITIES

- Journal Referee: AGNT, JAAMAS, JAR, FORM, Information and Computation, SOSYM, CACM, TCS, TOPLAS, JSC, SCICO, STTT, IPL
- Program committee member: MFCS, SAS, APLAS, CAV, STACS, FMCAD
- Review Editor for Frontiers in Computer Science
- Co-Organizer of Dagstuhl Seminar 17921, Resource Bound Analysis, July 2017

PUBLICATION SUMMARY

53 peer reviewed publications/proceedings (6 journal papers, 43 conference papers, 2 workshop papers, 2 invited papers), cited 1398 times, h-index 19 (as of 04.2022, google scholar), 1 patent

TOP 10 SELECTED PEER-REVIEWED PUBLICATIONS (*most relevant to proposal)

1. Jens Pagel, Florian Zuleger: Strong-Separation Logic. ESOP 2021: 664-692

2. Florian Zuleger: The Polynomial Complexity of Vector Addition Systems with States. FoSSaCS 2020: 622-641

3. (*) Matthias Schlaipfer, Friedrich Slivovsky, Georg Weissenbacher, Florian Zuleger: Multi-linear Strategy Extraction for QBF Expansion Proofs via Local Soundness. SAT 2020: 429-446

4. Jens Katelaan, Christoph Matheja and Florian Zuleger: Effective Entailment Checking for Separation Logic with Inductive Definitions. TACAS (2) 2019: 319-336

5. Ivan Radicek, Gilles Barthe, Marco Gaboardi, Deepak Garg, Florian Zuleger: Monadic refinements for relational cost analysis. Proc. ACM Program. Lang. 2(POPL): 36:1-36:32 (2018)

6.Tomás Brázdil, Krishnendu Chatterjee, Antonín Kucera, Petr Novotný, Dominik Velan, Florian Zuleger: Efficient Algorithms for Asymptotic Bounds on Termination Time in VASS. LICS 2018: 185-194

7. Benjamin Aminof, Aniello Murano, Sasha Rubin, Florian Zuleger:
Prompt Alternating-Time Epistemic Logics. KR 2016: 258-267

8. Sasha Rubin, Florian Zuleger, Aniello Murano, Benjamin Aminof: Verification of Asynchronous Mobile-Robots in Partially-Known Environments. PRIMA 2015: 185-200, best paper award

9. Tomer Kotek, Mantas Simkus, Helmut Veith, Florian Zuleger:
Extending ALCQIO with Trees. LICS 2015: 511-522

10. Thomas Colcombet, Laure Daviaud, Florian Zuleger: Size-Change Abstraction and Max-Plus Automata. MFCS (1) 2014: 208-219

Uploaded file(s) containing signatures:

signatures.pdf [PDF, 2 pages, 489.51 KB]
https://funding.wwtf.at/modules/download.php?key=189011_EN_O&dat=d0c7c882