UNIKASSEL VERSITÄT

HOCHSCHULE FULDA
UNIVERSITY OF APPLIED SCIENCES

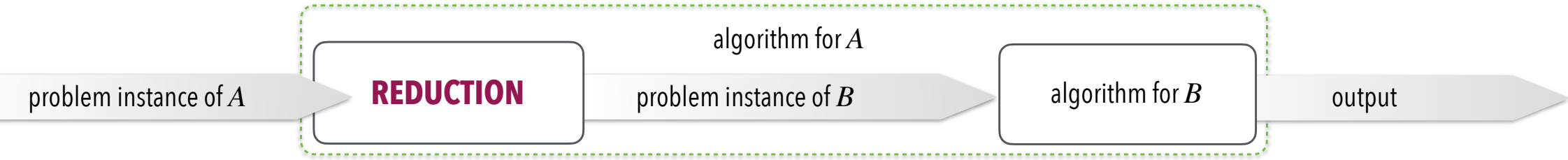
Towards Computer-Aided Teaching of Reductions in Theoretical Computer Science

Maurice Herwig[1], Norbert Hundeshagen[1], Marit Kastaun[2], and Cedric Kollenberg[3]

[1]University of Kassel, Germany, Theoretical Computer Science / Formal Methods / [2]University of Kassel, Germany, Didactics of Biology / [3]Hochschule Fulda – University of Applied Sciences, Germany, Applied Computer Science

Reductions – A Formal Definition

- A problem A is called reducible onto B, if there is a computable function f such that $x \in A \Leftrightarrow f(x) \in B$ (f.a. x from the domain of A).
- Essentially, the above definition expresses the ability to **use a solution for B to solve problem A.**



Motivation & Research Questions

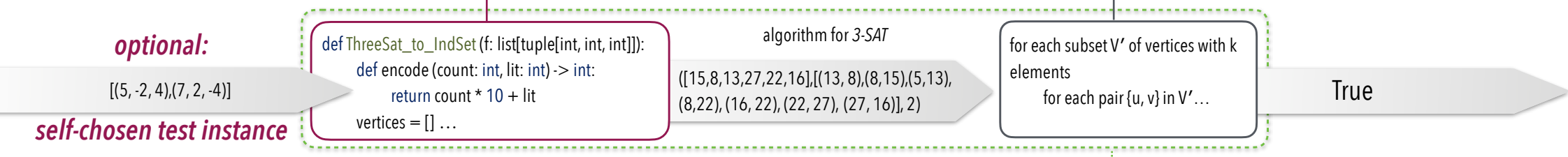
- "Reduction is an important method for solving problems in mathematics and science in general and is one of the main principles of computer science".[1]
- There are different tools to support teaching and learning, mostly relying on reductions given in a domain-specific language. [e.g. 2, 3]
- Our approach: Reductions as a programming exercise in a common and widespread language.

How can the abstraction level of reductions be minimized by testing worked examples in and with Python?

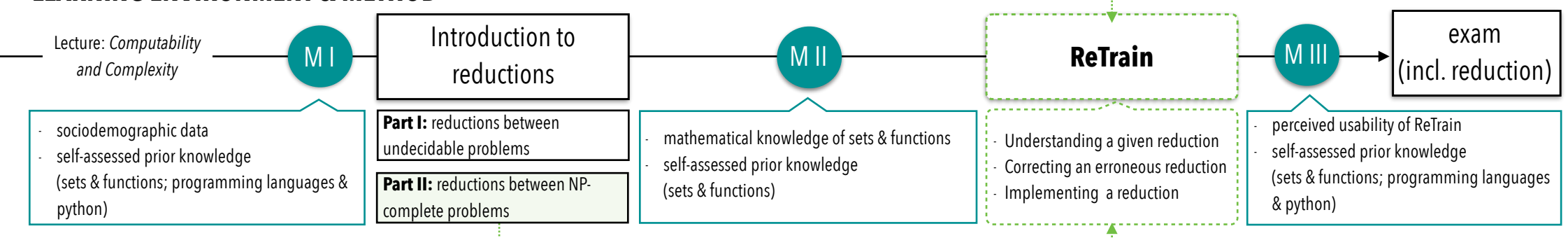
How and to what extent should automated feedback be provided to support learners in their learning process of reductions?

Tool-Based Teaching of Reductions – A Pilot Study

PROTOTYPE - ReTrain



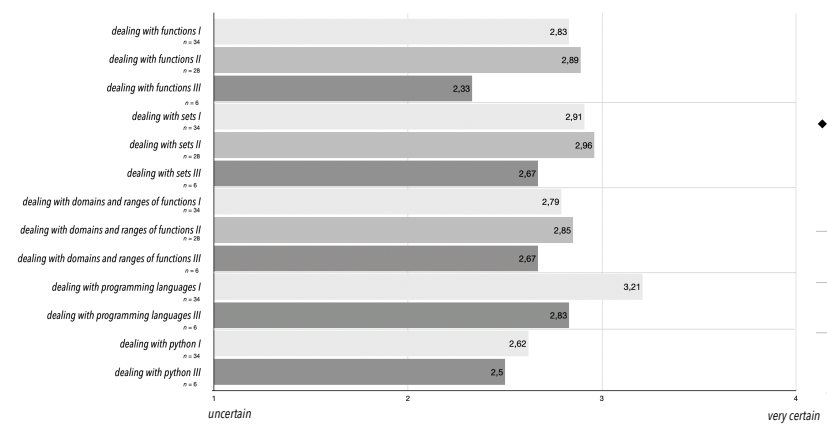
LEARNING ENVIRONMENT & METHOD



Results

n = 41; 1st - 3rd semester = 47.1 %; 4th-7th semester = 26.5 %; >7th semester = 26.5 %

- At the measurement points students assessed their mathematical (MI - MIII) and programming skills (MI & MIII) in different ways.



- The students demonstrated a range of proficiency in mathematical skills, with specific attention to reductions.

(1= incorrect; 2 = correct)

- *knowledge of sets: M* = 1.50; *SD* = .23
- *knowledge of functions: M* = 1.67; *SD* = .13

- There is a partial correlation between the self-assessed prior knowledge, (i.e. dealing with sets) and mathematical skills.

	knowledge of sets	*knowledge of functions*
dealing with functions	$r = -.028, p = .893$	$r = .34, p = .103$
dealing with sets	$r = .051, p = .804$	$r = .35$, $\mathbf{p = .043}$, $d = .24$
dealing with domains and ranges of functions	$r = -.008, p = .971$	$r = .38$, $\mathbf{p = .048}$, $d = .28$

"sometimes a bit fiddly" (student 4th - 7th semester)

"For me personally, the code window was sometimes not displayed at all." (student 1st - 3rd semester)

"I think it is very good that there is the possibility of testing reductions using pre-defined queries. However, for some problems it is quite complicated to specify the reduction, as many transformations have to be made due to the encoding." (student 1st - 3rd semester)

Results of telemetry data and qualitative feedback:

- testing of instances is perceived as helpful but troublesome due to abstract encodings
- only a few self-chosen examples were tested
- time consuming implementation of reductions

A Future Learning Tool for Reductions

EXTENSIBILITY

simplified generation of new reduction tasks

- support via pre-defined libraries and reusability of data from existing problems
- easy-to-understand and uniform description language for decision procedures [cf. 2]

support of undecidable problems

- only limited feedback via testing (e.g. for semi-decidable problems)
- further research needed (e.g. identification of "didactic" subclasses that permit certain automatic feedback methods)

USABILITY

focus on side-aspects such as problem encodings needs to be reduced

- encodings/data-structures should be pre-defined and if possible visualized (e.g. graph representation)

more problem-specific feedback is needed

- type hints for instances of reduction problems
- (semi-)automatic assessment of correctness (e.g. by static code analysis)
- further research needed on typical classes of errors [cf. 4]

individualized and simplified instance testing

- automatic generation of examples and provision of edge cases

Literature

[1] Armoni, M.; Gal-Ezer, J.: Reduction – an Abstract Thinking Pattern: The Case of the Computational Models Course. In: Proc. of the 37th SIGCSE Technical Symposium on Computer Science Education. pp. 389–393, 2006.

[2] Creus, C.; Fernández, P.; Godoy, G.: Automatic Evaluation of Reductions between NP- Complete Problems. In: Sinz, C. & Egly, U. (Eds.) *Theory and Applications of Satisfiability Testing – SAT 2014.* Springer International Publishing, pp. 415–421, 2014.

[3] Zhang, C.; Hartline, J. D.; Dimoulas, C.: Karp: A Language for NP Reductions. In: Association for Computing Machinery (Eds.) *Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation.* PLDI 2022, pp. 762–776, 2022.

[4] Gal-Ezer, J.; Trakhtenbrot, M.: Identification and addressing reduction-related misconceptions. *Computer Science Education 26*(2-3), pp. 89–103, 2016.

Acknowledgements

The authors would like to thank Kathrin Lehman, Clemens Weiße, and Marco Sälzer for their work in developing ReTrain. Furthermore, the work is partially funded by Stiftung Innovation in der Hochschullehre within the project UKS_digi.