EPiC Series in Computing

Volume 95, 2023, Pages 320–331

Proceedings of European University Information Systems Congress 2023

Automatic and Interactive Validation of Study Regulations in Accreditation Processes of Higher Education Institutions

Markus von der Heyde[1*], Chukwunwike Otunuya[2#], Matthias Goebel[3*], Dietmar Zoerner[4+], Ulrike Lucke[4+]

[1,2] SemaLogic UG, Germany

[3,4,5] University of Potsdam, Germany

[*] firstname.lastname@semalogic.de,

[#] lastname@uni-potsdam.de,

[+] firstname.lastname@uni-potsdam.de

Abstract

As digital workflows evolve around the curriculum life cycle in higher education institutions, advanced digital tools are needed to automate the processing of study regulations. In particular, the use of formal logic is beneficial for any type of validation in the accreditation process. Three key challenges are addressed in this paper: how to model study regulations, how to validate rules contained in modelled programmes, and how to package the contained logic of study rules in flexible communication between different AI services. This report on a case study demonstrates a solution that enables a continuous workflow from editing the rules to automated validation scenarios that support the administrative staff. The use of symbolic logic in conjunction with formal specification languages offers various forms of use cases within the curriculum life cycle.

Keywords: validation, specification language, answer set programming, higher education, artificial intelligence, curriculum life cycle, student information system

[1] https://orcid.org/0000-0002-6026-082X
[2] https://orcid.org/0000-0002-1012-7887
[3] https://orcid.org/0000-0002-4671-0900
[4] https://orcid.org/0000-0003-4049-8088

1 Introduction

In higher education, typical problems with study regulations can arise from a number of factors, including conflicting interpretations of rules, inconsistent enforcement of regulations, and lack of clear guidance on policy implementation. These problems can result in confusion among students, faculty, and administrators, and can negatively impact the overall educational experience. Another issue is the lack of standardisation across institutions, which can result in confusion for students who transfer between institutions or for those who are pursuing degrees from multiple institutions. In some cases, regulations may also be outdated or inconsistent with current best practices, leading to problems in their implementation and application. To address these issues, Higher Education Institutions (HEI) regularly review and update their study regulations.

The approach of formalising study regulations in logic-based languages had been applied before (Habtemariam, 2006; Nguyen, 2012). Various problems had been addressed, in particular the solution of the combinatorial scheduling of timetable requirements (Banbara et al., 2013, Banbara et al., 2019). However, the models of the formalised study regulations always required a manual translation from the original legal document, often as a PDF, defining the study programme and the derived model.

The use of SemaLogic (von der Heyde & Goebel, 2021) as a formal specification language provides an additional link between natural language and logic programs. Thus, the processes related to the description, validation, and interpretation of study regulations can be enhanced by utilising AI technology. This approach employs a logic-based method that merges natural language and symbolic rules, allowing the study regulations to be understood by both humans and machines. In order to enhance AI competency and empower subject representatives to design and use the approach, an offer of advanced training is being developed as part of the project. The overarching goals of the project are to increase the structural validity and consistency of study regulations through automated testing, improve the understanding of AI among university stakeholders, and enhance the quality of higher education by implementing AI technology in the organisation of studies. This is achieved through a generic microservice approach, ensuring that all components are robust, scalable, and easily transferable to other higher education institutions (von der Heyde et al., 2023).

This paper focuses on the formalised modelling of inherently flexible rules which define a study program, their generic coding, and the validation in multiple stages. Motivated by a number of use cases, the communication protocol was standardised to cover the logic contained in the study regulations. We place significant emphasis on the formal modelling of study regulations and their validation at multiple levels. This paper aims to provide a comprehensive explanation of the research programme and accompanying applications associated with this approach. Section 2 will outline the level of modelling applied to the study programmes and additional constraints. Section 3 will introduce the specific levels of validation. Finally, the paper will summarise the overall approach to formally validate study regulations based on their formal specification language descriptions.

2 Formal modelling of study regulations

Formal specification languages are languages that are used to describe the behaviour and properties of systems in a precise and mathematical manner. These languages provide a systematic and unambiguous means of describing complex systems by simplifications and describing the problem on an abstract level, thus making it easier to understand, analyse, and verify the systems' correctness. Formal specification languages typically include a set of symbols, rules, and syntax used to specify the behaviour of systems, and often have a theoretical foundation in formal logic and mathematics. They are widely used in various domains, including software engineering, computer science, business contracts (He et al., 2018, Sharifi et al., 2020), and engineering (Gacek et al., 2015) to specify the

behaviour of specific systems in their domain. The use of formal specification languages enables the systematic and rigorous analysis of systems, reducing the risk of errors and improving the overall quality of the described systems. Soavi et al. (2022) compiled a systematic literature review on the application of specification languages to legal documents and, in particular, smart contracts. The translation from natural language to formal specification language was divided into four phases to structure the body of literature: (a) structural and semantic annotation based on an ontology; (b) identification of relationships for concepts identified in the previous step; (c) formalisation of terms into a domain model; and (d) generation of formal expressions. The translation process is not yet performed automatically by any of the reviewed publications for the generic case.

Study regulations are written in natural languages, but still aim to be understandable and concise. As they are legal documents, they typically use a formal language style. However, they do not use formal specification languages. The utilisation of formal specification languages for study regulations facilitates the application of formal logic and mathematical constraints in order to assess their logical consistency, completeness, and accuracy (Habtemariam, 2006). Therefore, the level of abstraction typically used in study regulations as a set of common operators, rules or constraints builds a foundation for their validation as simplification.

The logical operators contained in typical study regulations have been analysed in the past (von der Heyde & Goebel, 2020). Capturing this semantic structure as a nested or hierarchical tree of boolean operators connecting modules belonging to a specific programme is one option in modelling. Additional structural constraints are, e.g., differentiation between focus topics, temporal requirements or prerequisites. SemaLogic has been designed to be able to capture a broad variety of study regulations found in the initial analyses. Based on the formal rule statements, the number of alternative module combinations can be calculated (von der Heyde & Goebel, 2021).

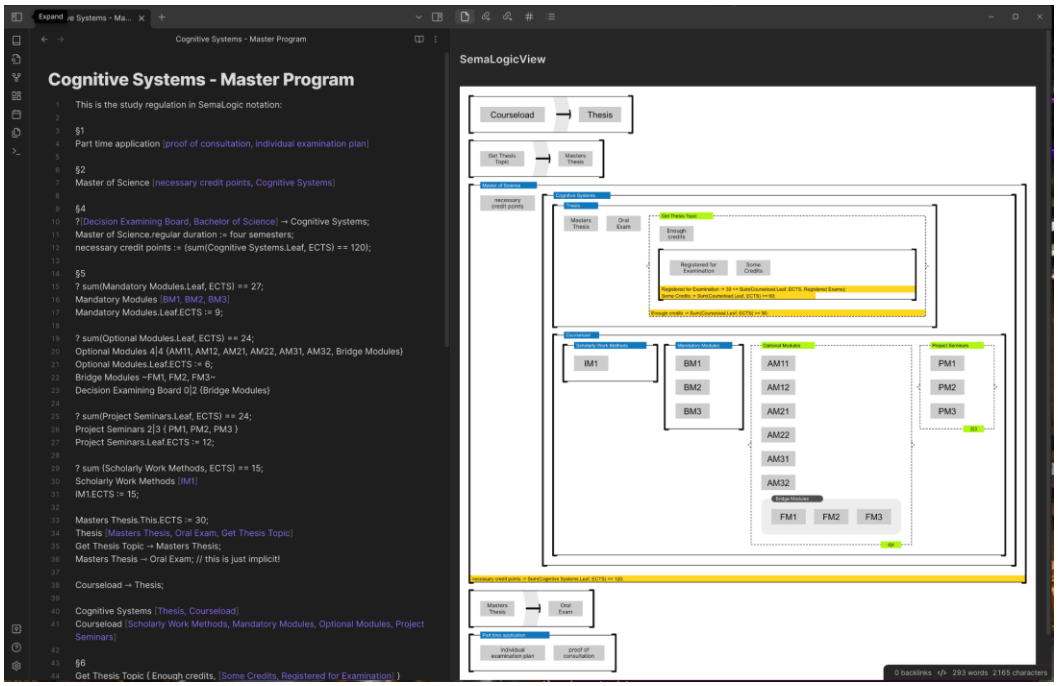

Figure 1: Example of the editing of technical language (left hand side) and the derived graphical representation of the study programme (right hand side).

Adding additional data attributes to modules allows for the capture of the expected student workload (e.g., as credit points), reference to required or allowed exam types, or similar data. In the case of credit points, simple mathematical operations may add to the list of constraints (e.g., a group of modules are required to sum to or exceed a specific number of credit points).

The formal model of the Cognitive Systems master programme of the University of Potsdam was written using SemaLogic as technical language (see Figure 1). Further steps in the modelling will involve the rewriting of the rules in the formal specification language which is closer to the natural legal language which normal study regulations would apply.

3 Validation of study regulations

Validation of study regulations is an essential process in ensuring the quality of a study programme. This procedure involves evaluating the rules against a set of predefined criteria to determine their suitability for the intended purpose. The validation process is typically conducted by subject matter experts and involves a systematic review of the rules, including examining their logical consistency, completeness, and accuracy. The outcome of the validation process is documented, and any discrepancies or issues identified are resolved before the rules are published and applied. Typically, this procedure is performed during any formal accreditation of the study program. This process helps to minimise the risk of errors and ensures that the rule sets are consistent with the overall goals and objectives of the educational system.

In order to introduce the targeted use cases within the validation domain, first the core principles and main use cases are summarised. To allow automatised support subsequently, two complementary approaches are employed: a low-level real-time validation checking, which is effective for a subset of constraints and types of inconsistencies, and a full combinatorial solving process using Answer Set Programming (ASP), which checks all conditions and includes additional constraints. These approaches are linked and the latter incorporates additional data at the ASP solver level, which is detailed in a follow-up subsection. This section concludes with a brief overview of future development.

3.1 General use cases

Using SemaLogic as a formal specification language, validation mainly addresses constellations that can be described by numbers and facts. The following list illustrates the breadth of questions that can be answered automatically to support less error-prone study regulations and accreditable programmes:

- Do the regulations comply with the basic regulations of the University of Potsdam?
- Is the programme manageable? Can the study requirements be met in the given time? Where are the bottlenecks in the sequence that pose a risk to students?
- Could the programme be studied with the current course offerings? If changes were made, could it be studied with equal or higher success rates? Where are systematic bottlenecks?
- What alternatives in the course sequence (i.e., what degrees of freedom) do students have to meet the European Credit Transfer and Accumulation System (ECTS) requirements? How many choices do they have to make?
- How do different study programmes relate to each other in the above respects? Are there critical inconsistencies with the examination regulations? To what extent are the identified characteristics compatible with the strategic direction of the university?
- What teaching capacity is required for which student numbers? What distribution of teaching capacity do the defined elective options and frequencies of modules entail?
- How robust is the study program? How many faculty can be absent for what period of time without jeopardising course offerings? Where are sabbaticals systematically impeded?

While some of these steps are already performed efficiently, others can benefit from more advanced support. As an example, we present here the automatic generation of a table that compares modules from a formally modelled study regulation with the university's existing course offerings.

The table displayed in Figure 2 is used for quality assurance by visualising gaps in the range of courses with regard to the individual modules, overlaps, and inappropriate multiple use of courses. Previously, such mappings had to be created manually, while our approach allows for automated generation of such visualisations in the course of editing new study regulations.

Other activities in the validation process are omitted here for the sake of a concise presentation. Regardless of this simple example, the symbolic models of study regulations already have a high value in study operations, because the formalisation generates a deeper understanding of processes and dependencies, reveals subjective scope for interpretation, and thus reduces potential for misunderstandings.

	AM11 - Current Topics in Computational Linguistics 1	AM12 - Current Topics in Computational Linguistics 2	AM21 - Current Topics in Machine Learning 1	AM22 - Current Topics in Machine Learning 2	AM31 - Current Topics in Computational Intelligence 1	AM32 - Current Topics in Computational Intelligence 2	BM1 - Advanced Natural Language Processing	BM2 - Machine Learning and Data Analysis	BM3 - Advanced Problem Solving Techniques	FM1 - Foundations of Mathematics	FM2 - Foundations of Computer Science	FM3 - Foundations of Linguistics	IM1 - Individual Research Module	PM1 - Project in Computational Linguistics	PM2 - Project in Machine Learning	PM3 - Project in Computational Intelligence
Individual Research													x			
Intelligente Datenanalyse & Maschinelles Lernen I								x								
Solver Construction																x
Declarative Problem Solving																x
Intelligent Logistics Technology																x
Multimodal Dialogue in Human-Robot-Interaction														x	x	
Language, Vision, and Interaction														x	x	
Cognitive and Sensorimotor Development					x	x										
Multi-agent path finding					x	x										
Advanced Declarative Problem Solving and Optimization					x	x										
Bayesian statistical inference 2			x	x												
Atelier in Experimental and Computational Phonology	x	x														
Coreference Resolution	x	x														
Deep Learning for NLP	x	x														
Natural Language Understanding: What Does it Mean?	x	x														
Individuelles Praktikum 2			x	x											x	
Individuelles Praktikum 1			x	x											x	

Figure 2: Mapping of study regulations to course offerings are generated to support validation

3.2 Real-time low-level validation

Validating study regulations that comprise a substantial number of alternatives rapidly results in a large number of module combinations that comply with the rules, referred to as a solution. These alternatives may relate to optional modules, areas of specialisation, or the sequence in which modules are taken. As the number of rules that introduce alternatives increases, the complexity of detecting rule violations also increases. Specialised root causes for the lack of a solution in study regulations have been identified, which can be checked with a quadratic effort in relation to the number of rules or symbols used [$O(n^2)$]. The effort to check all combinations of a set of symbolic values typically grows exponentially, making it efficient to check a subset of typical violations prior to using a full factorial approach.

The semantic representation selected in SemaLogic is optimised for the efficient evaluation of a number of conditions. The language features were intentionally constrained to serve the purpose of an effective evaluation. Simultaneously, users are provided with the capability to explicitly incorporate variations into the rules. This is achieved by capturing the range of permitted values using intervals. This approach is also used to specify the number of alternatives in a selection, thereby defining the minimum and maximum required count of the local solution that satisfies the rules. Many logical constraints are limited to check satisfiability only for the upper and lower bound, thus avoiding the exponential effort.

Detection of the following logical rule violations was implemented at this stage:

- Loop: recognises recursive definitions of terms as well as prerequisites or temporal order.
- Completeness: detects if symbols are left undefined when they are needed.
- Compartmentalisation: detects if the rule set is split into multiple partitions, i.e., the defined symbols have no logical reference between them.
- Ranges: detects if instances of variables do not match the defined range of values.
- Conflicting AND and OR conditions: detects whether the requirements of the AND statements conflict with the constraints of the concurrent OR statements, leaving the solution set empty.
- Empty dynamic groups: detects whether a dynamic group defined by an interval of symbolic names has no members.

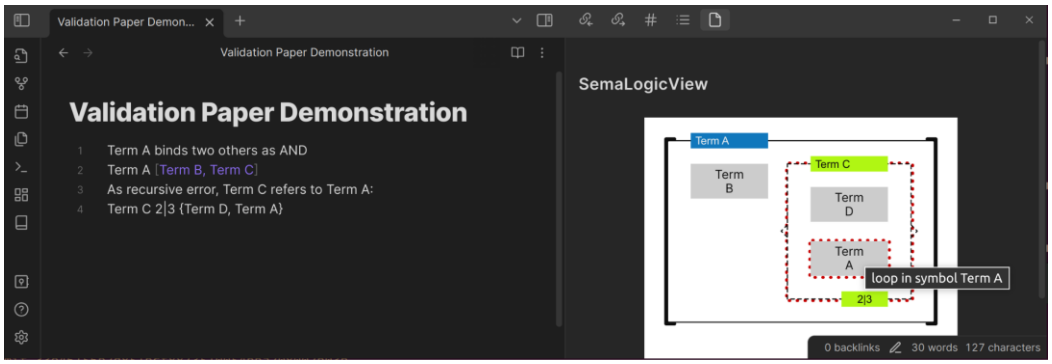

Figure 3: Integration of feedback into the Obsidian editor via interactive SVG graphics. An example for user feedback via the bubble help function is embedded into the SVG.

The user feedback is provided within the various data formats offered via the OpenAPI interface, as depicted in Figure 3: Within the JSON format provided for further evaluation through ASP, the errors are embedded into a JSON sub-structure. The SemaLogic semantic tree output provides a plaintext list of errors before the actual content (left). Finally, the automatically generated graphical representation using SVG (right) offers additional advice by means of the bubble help feature of HTML links. They

are embedded into the structure causing the inconsistency of the rule set. Thus, the user is directly aware of the source of the violation.

Additional conditions have been identified on the theoretical level, but not yet discovered in real study regulations. As the body of modelled regulations grows, additional checks will be implemented to match the need.

In sum, the low-level validation can be applied as a microservice into the online editing of study regulations and constantly helps to evaluate the defined rules in real-time.

3.3 Validation using ASP

ASP is a declarative programming paradigm for solving combinatorial search problems (Schaub & Woltran, 2018; Lifschitz, 2008). In ASP, the problem to be solved is first specified in a declarative fashion. This specification contains rules and constraints which define what a valid solution to the problem is. This is further explained below.

The resulting output from a SemaLogic model is utilised in the ASP solving process. As can be observed from Figure 4 below, each study regulation model in SemaLogic is convertible into a JSON object. This JSON object contains both data and rules found in the study regulation text. In the ASP Service, these rules and data are represented as data and are collectively taken as the ASP instance. This instance is fed into an ASP encoding, which contains the rules and constraints in a study regulation text, and then fed to the ASP solver to find one or many study paths (if they exist). A valid study path (or solution) is simply a set of modules which satisfy all the constraints. This separation of ASP instance and encoding effectively avoids rule generation for ASP, hence making our solving process easy to scale and maintain.

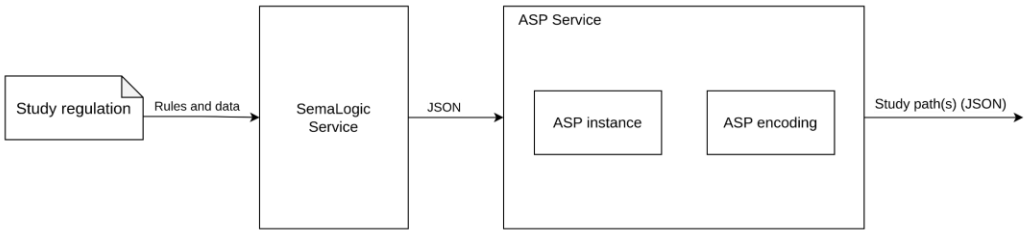

Figure 4: Generic workflow in processing study regulations towards any of the use cases.

The output of SemaLogic is checked for consistency. This checking process is preceded by a generation step, where combinations of the modules in a study regulation are created. This step does not involve checking the validity or studyability of each combination; the rules of the study regulation are not considered here. One could imagine the combinatorial blowup which this step will create: a study regulation with 10 modules will have $2^{10} = 1024$ combinations.

The checking step now applies all the rules to each element in the combination. Those that satisfy the rules remain while the others are excluded. If one or many solutions remain after the checking step then the rules are satisfiable. If no solution is produced then the rules are unsatisfiable. This ‘generate (or guess) and check’ methodology is commonly seen in ASP programs (Schaub & Woltran, 2018; Falkner et al., 2018).

The following features are currently implemented:

1. Fact format generator: the JSON output from SemaLogic is not fed into ASP as-is; rather, it goes through some preprocessing where it is converted into predefined ASP facts format. This feature serves as a kind of middleware between SemaLogic and the ASP encoding. This generator leverages the generic structure of the SemaLogic JSON output which features

modules, module groups, module properties, group aggregate functions and terms as it closely captures the information in the study regulation text.

2. ECTS range checker: modules are usually in groups with each group having properties unique to them. One such property is the ECTS. The ECTS of a module group has the maximum and sometimes minimum number of credit points required for that group. For compulsory module groups, the minimum and maximum ECTS is the same value, and for elective module groups the maximum ECTS is higher. With ASP we ensure this vital constraint is satisfied for all module groups present.
3. Module group cardinality checker: this feature ensures that the total number of modules chosen within a module group is between the minimum and maximum number of modules of the module group. A module group of 5 modules could require a minimum of 2 and a maximum of 3 modules to be selected in order to satisfy the cardinality constraint.

Part of our vision is to have all of these processes happen automated, ideally in real-time, i.e., as a study regulation for a programme is being crafted.

3.4 Transmission of structured logic information

The combination of real-time feedback of SemaLogic during the editing process and the additional validation results of ASP also depends on the communication between the two AI units. Only if the semantically coded knowledge is consistently interpreted will the user be able to equally benefit. Since the study regulation should be limited as little as possible, the coding between SemaLogic and ASP needs to be highly flexible. At the same time, the coding has to comply with existing standards. This section will demonstrate how flexible semantic structures, which arise during the writing process, can be transferred via constant interface definitions, such as a JSON construct defined by an OpenAPI 2.0 interface (Goebel & von der Heyde, 2023).

The goal is to use formalised study regulations in the microservice architecture at various places beyond the validation, e.g., using it as input to student information systems. It is essential that the generally formulated rules can be transferred without human intervention while retaining the extracted logic. The main challenge is to keep the interfaces constant while allowing maximal flexibility in the definition process. Hence, the JSON interface was divided into three generic sections of information:

1. Attributes: as symbols and their properties as attributed values
2. Groups: to flexibly name the segmentation of all symbols
3. Terms: to cover the logical relation between symbols as constraints

The first information concerns the “attribute” definitions. Symbols are defined in SemaLogic simply as anything which is not a specific grammatical token. Symbols represent modules, for example, whereas their attribute “ECTS” may represent the student workload as credit points. For each symbol, all occurring attributes of these symbols are transferred as a key-value pair. The specific naming of symbols and attributes are known at the time of usage, but are not part of the interface definition. For easier handling of further processing systems such as ASP, the data type is explicitly stored in addition to the attributed value.

The second information coded in the interface concerns the “group” definitions. A group definition represents an assignment of a symbol to a group of symbols or further groups. In this JSON fragment, all relevant individual modules or other groups can be assigned to a group symbol such as "mandatory modules". During the JSON export, SemaLogic resolves semantically defined dynamic groups into their individual elements.

In the third section, “terms” are coded as the essential logical relationships of the semantic objects. In analogy to the other two sections, we need to be able to map mathematical functions and logical constraints without adaptation of the interface. Each service can interpret specific logical or mathematical operations of the coded study regulations.

To accommodate this construct and the requirement for extensibility, the “terms” were defined as the combination of a designating symbol, a term type, and the associated parameters. The symbol is used to link to other semantic objects as defined in the other two sections. Thus, a term type of the "AND" operator can define "mandatory modules" for a group of otherwise defined symbols. The definition of further parameters would not be necessary in this case, since all symbols used by the AND condition were already defined in the group section. Other logical constraints require additional parameters. For example, an OR condition may be limited by a minimum and a maximum of required group members.

Even though different parameters are to be transferred for each term type, the definition of the individual parameters is again implemented as an array of a key-value transfer structure, as already defined in the first section. The parameter names (=key) represent unambiguous function parameters specifically adapted to the respective term type, which are to be interpreted by the services. New functions can thus be embedded in the data interface without changing the underlying JSON structure, and without this having to be made known to all other services, since the services only process those term types that are known to the respective service. This flexible coding with a constant interface enables a loose coupling between all services. The overall communication is implemented via the general UseCaseController as described in the generic architecture (von der Heyde et al., 2023).

3.5 Outlook for future development

Further development will focus on the realisation of the different use cases mentioned in Section 3.1. As the generic communication structure exists, additional data exported or derived from the university's student information system can be added to the validation process.

Additional constellations of constraints will cover the recommendation system. So far, the rules for a finite set of solutions have been specified for each study regulation. The individual study path of students may differ, even if the set of modules they have passed remains identical. Thus, during the course of study, students and student advisors need recommendations on the next steps for a specific student. SemaLogic is able to encode operators such as prerequisites, temporal constraints and recommendations. All three operators possibly influence the recommended order in which individual modules or groups of modules are completed.

ASP will capture the temporal features of a regulation and be used to generate, via a rating system, recommendations of all possible solutions, by prioritising those matching the additional and potentially weighted constraints and user-specified preferences. Also we will model and deduce the consequences of the following concepts: modules already passed by students, specialisations within a study program, temporal features in a regulation, module dependency and recommendation, study path recommendation based on student preferences (e.g., career path tailored modules, maximum number of ECTS per semester), and the effect of the absence or presence of one or more modules in a student programme in a particular semester or year.

4 Conclusions

The validation of new study programs or changes to existing programs usually requires considerable effort. The accreditation processes in higher education institutions are mainly a combination of non-automated evaluation and documentation processes. Access to automated processing in this area but also all other areas of the Higher Education Reference Model (HERM) as discussed in (von der Heyde et al., 2023) require clear semantics and data modelling. In order to provide advanced digital tools, the study regulations, which form the core of the defined curriculum, need to be formalised and made accessible to algorithms.

As formal specification languages provide a systematic and unambiguous means of describing complex systems, we have applied this principle to the modelling of study regulations. This makes it possible to understand, analyse and verify the correctness of the regulations. SemaLogic, as a specialised form of a formal specification language, includes a set of operators and a flexible syntax for specifying the semantic cascading of logically nested structures. To communicate the automatically extracted and evaluated logical content of the study regulation at different levels in the validation process, a generic OpenAPI-based JSON definition was chosen. ASP, a combinatorial problem-solving paradigm, was ultimately used in the study regulation validation process.

First use cases have been implemented. For example, the quality management process (as required by the accreditation) at the University of Potsdam regularly monitors the mapping of modules defined by the study regulations to course offerings in each semester. Further use cases have been defined and will be addressed in the near future. Also, the feasibility of the approach will be further analysed on the basis of a growing number of study regulations that provide a representative picture of the existing diversity of Bachelor and Master programmes.

5 Acknowledgements

This work is funded by the German Federal Ministry of Education and Research under contract number 16DHBKI024. We are deeply grateful to the whole CAVAS+ project team as well as to our partners in various organisational units at the University of Potsdam. Without their help and support, the work presented here would not have been possible.

References

Banbara, M.; Inoue, K.; Kaufmann, B.; Okimoto, T.; Schaub, T.; Soh, T.; Wanko, P. (2019). teaspoon: solving the curriculum-based course timetabling problems with answer set programming. Annals Operation Research, 275(1), 3–37. DOI: 10.1007/s10479-018-2757-7

Banbara, M., Soh, T., Tamura, N., Inoue, K., & Schaub, T. (2013). Answer set programming as a modeling language for course timetabling. Theory and Practice of Logic Programming, 13(4–5), 783–798. DOI: 10.1017/S1471068413000495

Falkner, A. A., Friedrich, G., Schekotihin, K., Taupe, R., & Teppan, E. C. (2018). Industrial Applications of Answer Set Programming. *Ki - Künstliche Intelligenz, 32*(2–3), 165–176. DOI: 10.1007/s13218-018-0548-6

Gacek, A., Katis, A., Whalen, M.W., Backes, J., & Cofer, D. (2015). Towards realizability checking of contracts using theories. In: Havelund, K., Holzmann, G., Joshi, R. (eds.) NFM 2015. LNCS, vol. 9058, pp. 173–187. Springer, Cham. DOI: 10.1007/978-3-319-17524-913

Goebel, M.; von der Heyde, M. (2023). OpenAPI Interface Definition for the Communication of Study Regulations between SemaLogic and Answer Set Programming (ASP). Zenodo. DOI: 10.5281/zenodo.8082268

Habtemariam, D. T. (2006). Simulation von Prüfungsordnungen und Studiengängen mit Hilfe von Constraint-logischer Programmierung [PhD Thesis]. https://d-nb.info/1046848909/34

He, X., Qin, B., Zhu, Y., Chen, X., & Liu, Y. (2018). SPESC: A Specification Language for Smart Contracts. 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), 2, 132–137. DOI: 10.1109/COMPSAC.2018.00025

Lifschitz, V. (2008). What is answer set programming. *National Conference on Artificial Intelligence*, 1594–1597. https://aaaipress.org/Papers/AAAI/2008/AAAI08-270.pdf

Nguyen, D. M. T. (2012). Modellierung, Pflege und Compliance-Regeln von Studienprozessen [Bachelor, University of Ulm]. http://dbis.eprints.uni-ulm.de/843/

Schaub, T., & Woltran, S. (2018). Answer set programming unleashed! *KI - Künstliche Intelligenz, 32*(2–3), 105–108. DOI: 10.1007/s13218-018-0550-z.

Sharifi, S., Parvizimosaed, A., Amyot, D., Logrippo, L., & Mylopoulos, J. (2020). Symboleo: Towards a Specification Language for Legal Contracts. 2020 IEEE 28th International Requirements Engineering Conference (RE), 364–369. DOI: 10.1109/RE48521.2020.00049

Soavi, M., Zeni, N., Mylopoulos, J., & Mich, L. (2022). From Legal Contracts to Formal Specifications: A Systematic Literature Review. SN Computer Science, 3(5), 345. DOI: 10.1007/s42979-022-01228-4

von der Heyde, M.; Goebel, M. (2020). Die Sprache «SemaLogic» als semantische Repräsentation—Eine anforderungsbasierte Sprache zur Modellierung von Prüfungsordnungen und Abbildung von Studienverläufen. INFORMATIK 2020, Back to the Future, 521–536. DOI: 10.18420/inf2020_48

von der Heyde, M.; Goebel, M. (2021). Structural comparison of curriculum design — Modelling international study programs using a logical language and its graphical representation. 15th annual International Technology, Education and Development Conference (INTED). DOI: 10.21125/inted.2021.0631.

von der Heyde, M., Goebel, M., Zoerner, D., & Lucke, U. (2023). Integrating AI Tools with Campus Infrastructure to Support the Life Cycle of Study Regulations. European University Information Systems Conference (EUNIS) 2023. EPiC Series in Computing, EasyChair.

Author biographies

Dr. von der Heyde received his PhD with topics in cognition research at the Max Planck Institute for Biological Cybernetics in Tübingen. Since 2011, Dr. von der Heyde has been advising colleges, universities, and public cultural and research institutions on a wide range of digitalisation topics (governance, organisation, strategy, research data management, information security, IT service management) as part of vdH-IT, and conducts independent research on these topics (see ResearchGate). Since 2018, he has been an Adjunct Professor at the School for Interactive Arts and Technology (SIAT) at Simon Fraser University, Vancouver. Dr. von der Heyde is also active as a volunteer in a variety of non-profit organisations (GI, ZKI, EUNIS, Educause). In 2020, he founded SemaLogic UG to use semantic and structural logic technologies to automatically map and validate natural language regulatory texts. The application of these technologies to study regulations and accreditation is currently being implemented with partners from the university environment. See further details at LinkedIn, or Google Scholar.

Chukwunwike Otunuya received his MSc with a project on university examination timetable scheduling in ASP at Covenant University, Ogun state, Nigeria. His work in ASP continued when he joined the Knowledge Representation and Reasoning (KRR) research group at the University of Potsdam in 2021, working as a Scientific Researcher. Currently he is also a doctoral student in Computer Science doing work and research on regulations at the University of Potsdam. You can contact him via LinkedIn.

Matthias Goebel has been active in numerous IT projects for the introduction or optimisation of SAP systems and SAP-based applications since 2000. For more than 10 years he managed the SAP divisions of various companies with regard to the company-wide SAP strategy and architecture. Application-related focal points are enterprise application integration, programme and DB-based performance optimisation, data warehousing and the redesign and modification of digitally supported processes.

Dr. Dietmar Zoerner studied computer science at TU Berlin. He has experience in the areas database and software design, software engineering and software architecture. From 2010 to 2018 he worked as a research assistant as a member of the chair for Complex Multimedia Application Architectures. In 2021 he did his PhD in the field of digital game-based learning. Since December 2021 he is working as a project coordinator at the chair for Complex Multimedia Application Architectures.

Dr. Ulrike Lucke is professor of computer science at the University of Potsdam, Germany. She obtained her PhD at the University of Rostock, Germany. Her research activities include institutional infrastructures for education, research and administration. Currently, among other activities, she coordinates a large-scale national initiative to create a digital ecosystem for education across Germany and acts as an independent evaluator for two European projects on policy experimentation in the educational sector. Until 2018, she was responsible for e-learning and IT strategy as Chief Information Officer of the University of Potsdam. She is a founding member and was vice chair of the German University CIO Association until 2020. Since 2020, she is a Vice President of the German Informatics (GI) society.