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

Markus von der Heyde[1], Chukwunwike Otunuya[2], Matthias Goebel[1], Dietmar Zoerner[2], Ulrike Lucke[2]

[1] SemaLogic UG, Germany
[2] University of Potsdam, Germany

EUNIS VIGO 2023

EUNIS Congress 2023, Vigo, Spain

What if…

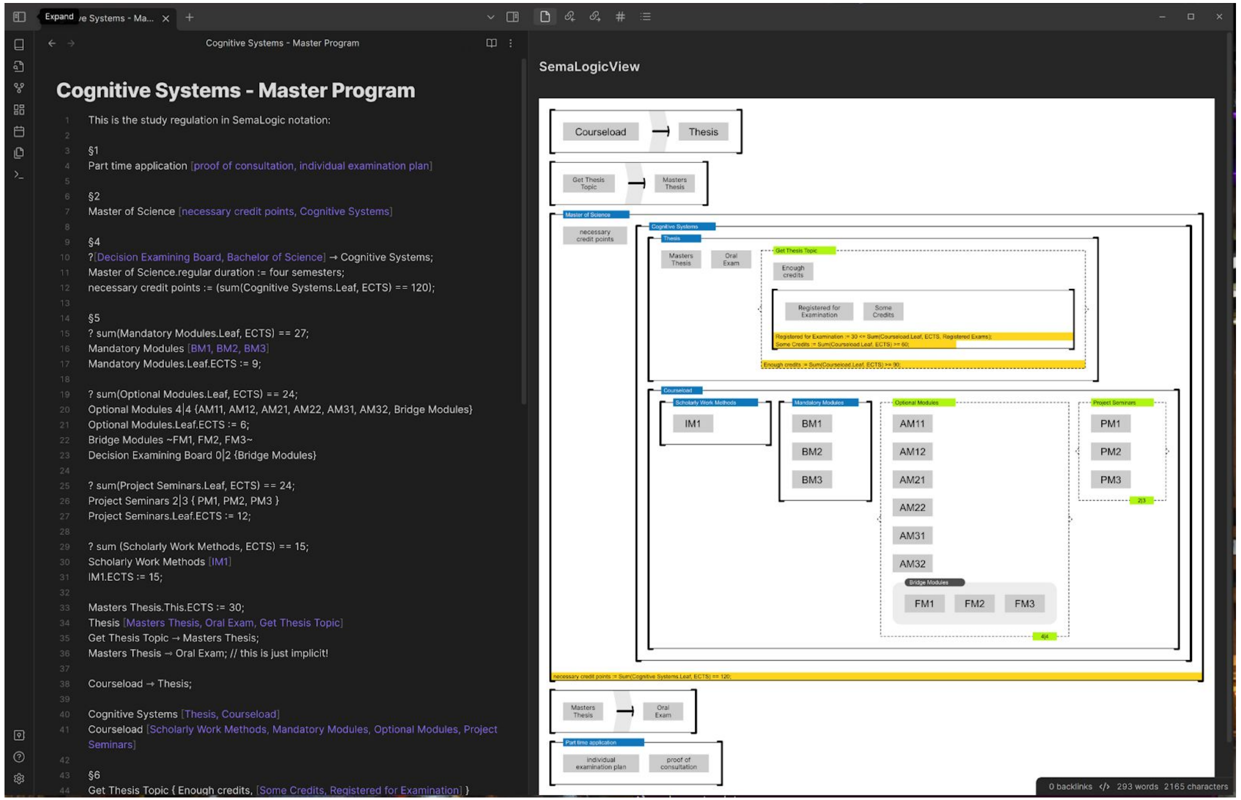
- we would get real time feedback
- on validity and misconceptions
- while writing or changing a study regulation.

We would need to…

- edit a text,
- do real time parsing,
- validate contained semantics,
- transform into other representations,
- generate feedback,
- and display feedback along with the text (or elsewhere).

How to edit

- technical language
- dynamically generated SVG
- MarkDown notation
- export PDFs (for archival)
- Obsidian plug-in to connect to REST API

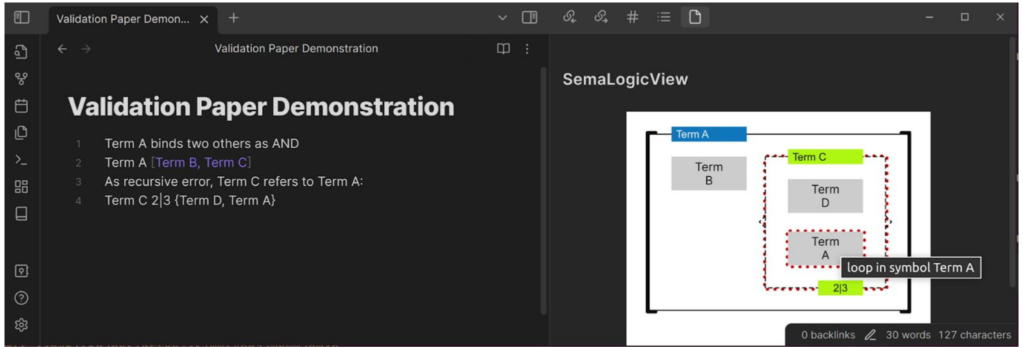


Cognitive Systems - Master Program

```
This is the study regulation in SemaLogic notation:

§1
Part time application [proof of consultation, individual examination plan]

§2
Master of Science [necessary credit points, Cognitive Systems]

§4
?[Decision Examining Board, Bachelor of Science] → Cognitive Systems;
Master of Science.regular duration := four semesters;
necessary credit points := (sum(Cognitive Systems.Leaf, ECTS) == 120);

§5
? sum(Mandatory Modules.Leaf, ECTS) == 27;
Mandatory Modules [BM1, BM2, BM3]
Mandatory Modules.Leaf.ECTS := 9;

? sum(Optional Modules.Leaf, ECTS) == 24;
Optional Modules 4|4 {AM11, AM12, AM21, AM22, AM31, AM32, Bridge Modules}
Optional Modules.Leaf.ECTS := 6;
Bridge Modules ~FM1, FM2, FM3~
Decision Examining Board 0|2 {Bridge Modules}

? sum(Project Seminars.Leaf, ECTS) == 24;
Project Seminars 2|3 { PM1, PM2, PM3 }
Project Seminars.Leaf.ECTS := 12;

? sum (Scholarly Work Methods, ECTS) == 15;
Scholarly Work Methods [IM1]
IM1.ECTS := 15;

Masters Thesis.This.ECTS := 30;
Thesis [Masters Thesis, Oral Exam, Get Thesis Topic]
Get Thesis Topic → Masters Thesis;
Masters Thesis → Oral Exam; // this is just implicit!

Courseload → Thesis;

Cognitive Systems [Thesis, Courseload]
Courseload [Scholarly Work Methods, Mandatory Modules, Optional Modules, Project
Seminars]

§6
Get Thesis Topic { Enough credits, [Some Credits, Registered for Examination] }
```

Simple rule violations

- **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.



Validation Paper Demonstration

```
Term A binds two others as AND
Term A [Term B, Term C]
As recursive error, Term C refers to Term A:
Term C 2|3 {Term D, Term A}
```

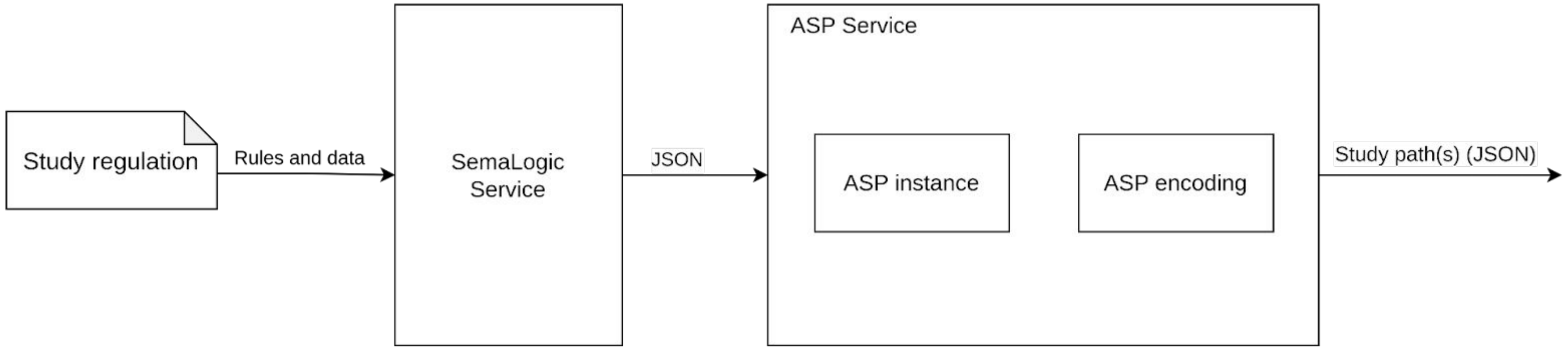
Generic JSON representation

Three generic sections of information:

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

ASP workflow (1/2)

- Beyond simple validation → use of highly specialized AI tool for Answer Set Programming (ASP)



ASP workflow (2/2)

- ASP was used to encode and solve the combinatorial problem.
- The generated SemaLogic JSON object serves as imput.
- Our encoding contains the study regulations constraints which feeds on the data.
- The solving process returns an output of study paths which satisfy the constraints in the regulation. It returns no solution if the constraints are not satisfied.

Live demonstration

- Simple editing…
- What was understood by the system?
- SemaLogic / JSON / SVG output
- Display of validation results

Summary

- Use of Formal Specification Language
- Automatic parsing and further processing
- Simple (SemaLogic-based) and complex (ASP-based) validation scenarios
- Real time capabilities for OpenAPI based REST interface
- Integration in Obsidian editor by plug-in mechanism

→ Writing and reading study regulations, while receiving real time validation feedback from SemaLogic and ASP.