

*Reasoning about Study Regulations in Answer Set Programming**

SUSANA HAHN

*University of Potsdam, Germany
Potassco Solutions, Germany*

CEDRIC MARTENS

University of Potsdam, Germany

AMADE NEMES

University of Potsdam, Germany

HENRY OTUNUYA

University of Potsdam, Germany

JAVIER ROMERO

University of Potsdam, Germany

TORSTEN SCHAUB

*University of Potsdam, Germany
Potassco Solutions, Germany*

SEBASTIAN SCHELLHORN

University of Potsdam, Germany

Abstract

We are interested in automating reasoning with and about study regulations, catering to various stakeholders, ranging from administrators, over faculty, to students at different stages. Our work builds on an extensive analysis of various study programs at the University of Potsdam. The conceptualization of the underlying principles provides us with a formal account of study regulations. In particular, the formalization reveals the properties of admissible study plans. With these at end, we propose an encoding of study regulations in Answer Set Programming that produces corresponding study plans. Finally, we show how this approach can be extended to a generic user interface for exploring study plans.

KEYWORDS: Answer Set Programming, Study regulations and plans

1 Introduction

Study regulations govern our teaching at universities by specifying requirements to be met by students to earn a degree. Creating a study program involves different stakeholders: faculty members designing study programs, administrative and legal staff warranting criteria, like studyability, faculty members teaching the corresponding programs as well

* A preliminary version of this article appeared in (Hahn et al. 2023).

as supervising their execution on examination boards, study advisors consulting students, and of course, students studying accordingly.

Given this impressive spectrum of use-cases, it is quite remarkable that study regulations are usually rather sparse and leave many aspects to the commonsense of the respective users. This is needed to cope with their inherent incomplete, inconsistent, and evolving nature. For instance, often study regulations leave open minor dependencies among modules. Sometimes associated courses overlap and certain modules cannot be taken in the same semester. And finally, studying happens over time, students' perspectives may change and faculty may rotate. Often these phenomena are compensated by changes, preferences, recommendations, defaults, etc. In fact, this richness in issues and notions from Knowledge Representation and Reasoning (KRR) makes study regulations a prime candidate for a comprehensive benchmark for KRR formalisms.

Our approach is adaptable, though not universally applicable. Its foundations lie in the principles of Europe's Bologna Process, designed to harmonize higher education across the continent. European curricula, for example, are structured around credit points, often allocated to individual modules. The European Credit Transfer and Accumulation System (ECTS) provides guidance on program design and credit allocation. Similarly, our approach utilizes modules and credit points as fundamental building blocks. Our goal is to encompass all study regulations at the University of Potsdam, which share additional key principles. However, adaptation or even redesign may be necessary for other institutions with differing structures or requirements.

In fact, this work is part of a project conducted at the University of Potsdam to assist different users by automatizing study regulations. These users range from study administrators, over faculty in different functions, to prospective and advanced students. We started by analyzing more than a dozen different study regulations in order to identify their underlying principles. The conceptualization of the basic principles led us to a formal account of basic study regulations, presented in Section 2. For illustration, we provide the formalization of the master program *Cognitive Systems*. We refine this in Section 3 by showing how modules are passed by accomplishing their associated examination tasks. The formalization of study regulations reveals the properties of admissible study plans. To automatize reasoning about study regulations and their study plans, we capture their properties in Answer Set Programming (ASP; Lifschitz (2002)), a declarative problem solving paradigm, tailored for knowledge representation and reasoning. The ASP-based encoding of basic study regulations is discussed in Section 4. Moreover, we show in Section 5 how this encoding can be used together with an ASP-driven user interface to browse through study plans of given study regulations. We conclude in Section 7.

2 Conceptualizing study regulations

The basic concept of our study regulations are modules. Accordingly, a semester is composed of a set of modules and a study plan is a finite sequence of semesters. More formally, given a set M of modules, a *study plan* of n semesters is a sequence $(S_i)_{i=1}^n$ where $S_i \subseteq M$ for $1 \leq i \leq n$. Study regulations specify legal study plans. To capture this, we propose an abstract characterization of study regulations and show how they induce legal study plans.

A basic *study regulation* is a tuple $(M, G, c, s, l, u, R_g, R_t)$, where

1. M is a set of modules,
2. $G \subseteq 2^M$ distinguishes certain groups of modules,
3. $c : M \rightarrow \mathbb{N}$ gives the credits of each module,
4. $s : M \rightarrow \{w, s, e\}$ assigns a regular semester to a module,
5. $l : G \rightarrow \mathbb{N}$ returns the lower-bound of the credits of a module group,
6. $u : G \rightarrow \mathbb{N}$ returns the upper-bound of the credits of a module group,
7. $R_g \subseteq 2^{(2^M)^n}$ is a set of global constraints expressing study regulations, and
8. $R_t \subseteq 2^{(2^M)^n}$ is a set of temporal constraints expressing study regulations.

The module groups in G allow us to structure the modules and to express group-wise regulations. Functions c and s give the credit points of a module and its turnus,¹ viz. in winter, summer, or each semester (indicated by w, s, e), respectively. The elements l and u are partial functions delineating the number of credits obtained per module group; a specific number of credits is captured by an equal lower and upper bound. The regulation constraints in R_g and R_t are represented extensionally: each constraint r in $R_g \cup R_t$ is represented by the sequences of sets of modules $r \subseteq (2^M)^n$ satisfying it. While the sets of constraints R_g and R_t share the same mathematical structure, they differ in purpose and are therefore separated for clarity's sake. R_t expresses temporal constraints over the sequence of semesters, while R_g does not make use of this sequential structure, and rather expresses global constraints over the entire set of modules.

A study plan for a basic study regulation is a finite sequence of sets of modules satisfying all regulation constraints. More precisely, a sequence $(S_i)_{i=1}^n$ of modules of length n such that $S_i \subseteq M$ for $1 \leq i \leq n$ is a *study plan* for $(M, G, c, s, l, u, R_g, R_t)$ if $(S_i)_{i=1}^n \in \bigcap_{r \in R_g \cup R_t} r$.

Finally, we call modules exogenous if they are imposed by external means, e.g. by an examination board. The specific choice of these modules is determined case by case.

Example 1 (Cognitive systems)

As an example, consider the study regulations of the international master program *Cognitive Systems* offered at the University of Potsdam.² This program offers a combination of modules in Natural language processing, Machine learning, and Knowledge representation and reasoning.

These subjects are reflected by the three mandatory base modules, joined in B in (3). Since each module yields 9 credits, their obligation is achieved by requiring that the modules stemming from B must account for 27 credits.³ While the amount is specified in (12) and (13), the actual constraint is imposed in (15). The same constraint is used for choosing two among three possible project modules from P . The optional modules in group O are handled similarly, just that only 24 credits from 36 possible credits are admissible. That is, four out of nine modules must be taken. The freedom of which four the student may choose is restricted by the examination board by imposing the study of up to two foundational modules $E \subseteq F$, which must be the only modules taken from module group F , as formalized in constraint (14). The total number of credits over all

¹ Winter and summer semesters are associated with odd and even positions in a sequence, respectively (see below).

² Available at https://www.uni-potsdam.de/fileadmin/projects/studium/docs/03_studium_konkret/07_rechtsgrundlagen/studienordnungen/St0_CogSys_EN.pdf

³ This is not our way of modeling mandatory courses but rather reflects the actual regulation.

modules must equal 120. Finally, an internship, im , and a thesis, msc , are imposed in (16). This brings us to the temporal regulation in (20) requiring that at least 90 credits are accumulated before conducting a thesis. The temporal regulations in (18) and (19) ensure that modules are taken in the right season. And finally (17) makes sure that modules are chosen at most once.

Let $E \subseteq F$ be some exogenous set of modules in the following example; and let \bar{S} stand for $\bigcup_{i=1}^n S_i$, and $M_w = \{m \in M \mid s(m) = w\}$ and $M_s = \{m \in M \mid s(m) = s\}$. Then, the study regulations of the master program *Cognitive Systems* with respect to E can be formalized as follows.

$$M = B \cup F \cup A \cup P \cup \{im, msc\} \quad (1)$$

$$G = \{B, F, A, O, P, M\} \quad (2)$$

$$B = \{bm_i \mid i = 1..3\} \quad (3)$$

$$F = \{fm_i \mid i = 1..3\} \quad (4)$$

$$A = \{am_{i,j} \mid i = 1..3, j = 1, 2\} \quad (5)$$

$$O = F \cup A \quad (6)$$

$$P = \{pm_i \mid i = 1..3\} \quad (7)$$

$$c = \{bm_i \mapsto 9 \mid i = 1..3\} \cup \{am_{i,j} \mapsto 6 \mid i = 1..3, j = 1, 2\} \cup \quad (8)$$

$$\{fm_i \mapsto 6 \mid i = 1..3\} \cup \{pm_i \mapsto 12 \mid i = 1..3\} \cup \{im \mapsto 15, msc \mapsto 30\} \quad (9)$$

$$s = \{bm_1 \mapsto w, bm_2 \mapsto s, bm_3 \mapsto w\} \cup \{am_{i,j} \mapsto e \mid i = 1..3, j = 1, 2\} \cup \quad (10)$$

$$\{fm_i \mapsto w \mid i = 1..3\} \cup \{pm_i \mapsto e \mid i = 1..3\} \cup \{im \mapsto e, msc \mapsto e\} \quad (11)$$

$$l = \{B \mapsto 27, O \mapsto 24, P \mapsto 24, M \mapsto 120\} \quad (12)$$

$$u = \{B \mapsto 27, O \mapsto 24, P \mapsto 24, M \mapsto 120\} \quad (13)$$

$$R_g = \{ \{(S_i)_{i=1}^n \subseteq M^n \mid |E| \leq 2, \bar{S} \cap F = E\}, \quad (14)$$

$$\{(S_i)_{i=1}^n \subseteq M^n \mid l(H) \leq \sum_{m \in H \cap \bar{S}} c(m) \leq u(H)\} \text{ for all } H \in \{B, O, P, M\}, \quad (15)$$

$$\{(S_i)_{i=1}^n \subseteq M^n \mid \{im, msc\} \subseteq \bar{S}\} \quad (16)$$

$$R_t = \{ \{(S_i)_{i=1}^n \subseteq M^n \mid S_i \cap S_j = \emptyset, 1 \leq i < j \leq n\} \quad (17)$$

$$\{(S_i)_{i=1}^n \subseteq M^n \mid M_w \cap S_{2k} = \emptyset, 1 \leq 2k \leq n, k \in \mathbb{N}\} \quad (18)$$

$$\{(S_i)_{i=1}^n \subseteq M^n \mid M_s \cap S_{2k-1} = \emptyset, 1 \leq 2k-1 \leq n, k \in \mathbb{N}\} \quad (19)$$

$$\{(S_i)_{i=1}^n \subseteq M^n \mid msc \in S_k, \sum_{1 \leq i < k} \sum_{m \in S_i} c(m) \geq 90, k \in \mathbb{N}\} \quad (20)$$

If the set of exogenous modules given by the examination board is, for example, $E = \{fm_1\}$, one admissible study plan spanning four semesters is $S = (S_i)_{i=1}^4$, where

$$S_1 = \{bm_1, bm_3, fm_1, am_{1,2}\} \quad (21)$$

$$S_2 = \{bm_2, am_{2,1}, pm_1\} \quad (22)$$

$$S_3 = \{im, pm_3, am_{3,1}\} \quad (23)$$

$$S_4 = \{msc\} \quad (24)$$

This plan comprises 120 credits, although the load per semester varies.

For illustration, let us verify that our study plan belongs to the ones in (14) and (15) for $H = O$. Indeed constraint (14) is satisfied as we have $F \cap \bar{S} = \{fm_1\} = E$, and thus

S is an element of constraint (14). With regards to (15), we have

$$O \cap \bar{S} = \{fm_1, am_{1,2}, am_{2,1}, am_{3,1}\} \quad (25)$$

which makes us check whether our study plan satisfies

$$l(O) = 24 \leq \sum_{m \in \{fm_1, am_{1,2}, am_{2,1}, am_{3,1}\}} c(m) \leq 24 = u(O) \quad (26)$$

This is indeed the case since $c(fm_1) + c(am_{1,2}) + c(am_{2,1}) + c(am_{3,1}) = 24$. Hence, our study plan is an element of constraint (15).

Although the above specification reflects the legal study regulation, it leaves many ambiguities behind. For instance, the number of credits per semester is left open, as is the order of the modules. The guideline is usually to take around 30 credits per semesters but this is not enforced. Similarly, basic modules in B should be taken before advanced ones in A , again this is neither enforced nor always possible. Since these constraints are usually soft, they are left to the students and/or their study advisors.

Our previous preliminary work (Hahn et al. 2023) outlined further extensions to basic study regulations (subarea specializations, module dependencies, blocking modules). The following section delves into a more novel aspect: how modules are passed by accomplishing their associated examination tasks.

3 Examination Tasks

Modules have specific examination tasks that define the criteria for successful completion. Students typically complete these tasks within courses, where they are carefully designed to align with the requirements of specific modules.

Study regulations focus on modules and their associated examination tasks. It is the responsibility of each department to provide a course selection that enables students to complete the modules required by their study program. Therefore, we concentrate in what follows on modules and their associated examinations, rather than courses.

Each module has at least one examination task⁴ and these tasks are unique to each module. We distinguish between *primary examinations* (e.g. written or oral exams) and *secondary examinations* (e.g. weekly exercises), and denote them by E_p and E_s , respectively. In analogy to study plans, *examination plans* are sequences of form $(E_i)_{i=1}^n \subseteq (E_p \cup E_s)^n$.

Interestingly, a single module can offer flexibility through different combinations of primary and secondary examination tasks. Given a set of modules M and sets E_p and E_s of primary and secondary examinations, we define the functions $e_p : M \rightarrow 2^{E_p}$ and $e_s : M \rightarrow 2^{E_s}$ to associate a module with different combinations of primary and secondary examination tasks, respectively.

For illustration, suppose module bm_1 requires as primary examination task either a written exam or a final project report, identified by $ep_{bm_1,1}$ and $ep_{bm_1,2}$. Also, two secondary examination tasks, a lecture attendance of 50% and the successful completion

⁴ All this follows the general regulations for study examinations for BSc and MSc degrees at <http://www.uni-potsdam.de/am-up/2013/ambek-2013-03-035-055.pdf> (last accessed on 30th of April 2024).

of all weekly exercises, viz. $es_{bm_1,1}$ and $es_{bm_1,2}$, are required by the module. This is captured as $e_p(bm_1) = \{\{ep_{bm_1,1}\}, \{ep_{bm_1,2}\}\}$ and $e_s(bm_1) = \{\{es_{bm_1,1}, es_{bm_1,2}\}\}$.

Next, we define the completion of modules in terms of examinations.

To this end, we define a function a associating examination plans with sequences of modules as $a : (E_i)_{i=1}^n \mapsto (S_i)_{i=1}^n$ where for $1 \leq i \leq n$ we have

$$S_i = \{m \in M \mid V \cup W \subseteq \bigcup_{j=1}^i E_j, V \cup W \not\subseteq \bigcup_{j=1}^{i-1} E_j \text{ for some } V \in e_s(m), W \in e_p(m)\}$$

The idea is to complete modules as early as possible. Once a module is completed for a specific V and W combination, it cannot be repeated for credit in later semesters. A module may appear multiple times in a module sequence if it offers several valid combinations of primary and secondary examination tasks, and these combinations are spread across different semesters within the examination plan. However, study plans become invalid if they include the same module in multiple semesters, as this violates a core regulation (see also (17)). Also, different examination plans may result in the same module sequence, especially if some examination tasks do not immediately contribute to completing a module.

For illustration, let us continue our example with $e_p(bm_1)$ and $e_s(bm_1)$ as above. Furthermore, suppose we have the primary examination tasks

$$e_p(m) = \{\{ep_{m,1}\}\} \text{ for } m \in \{bm_3, fm_1, am_{1,2}, bm_2, am_{2,1}, pm_1, im, pm_3, am_{3,1}, msc\}$$

along with the following secondary examination tasks $e_s(bm_2) = \{\{es_{bm_2,1}\}\}$, $e_s(bm_3) = \{\{es_{bm_3,1}, es_{bm_3,2}\}\}$, $e_s(fm_1) = \{\{es_{fm_1,1}\}\}$, and

$$e_s(m) = \{\emptyset\} \text{ for } m \in \{am_{1,2}, am_{2,1}, pm_1, im, pm_3, am_{3,1}, msc\}.$$

Now, consider the examination plan $(E_i)_{i=1}^4$ where

$$\begin{aligned} E_1 &= \{ep_{bm_1,1}, es_{bm_1,1}, es_{bm_1,2}, ep_{bm_3,1}, es_{bm_3,1}, es_{bm_3,2}, ep_{fm_1,1}, es_{fm_1,1}, ep_{am_{1,2},1}\} \\ E_2 &= \{ep_{bm_2,1}, es_{bm_2,1}, ep_{am_{2,1},1}, ep_{pm_1,1}\} \\ E_3 &= \{ep_{im,1}, ep_{pm_3,1}, ep_{am_{3,1},1}\} \\ E_4 &= \{ep_{msc}\}. \end{aligned}$$

In fact, this examination plan induces the study plan given in Section 2, that is, $a((E_i)_{i=1}^4) = (S_i)_{i=1}^4$ as given in (21) to (24).

Similarly, the examination plan $E' = (E_1, E_2 \cup \{ep_{bm_1,2}\}, E_3, E_4)$ induces $a(E') = (S_1, S_2 \cup \{bm_1\}, S_3, S_4)$. Since $ep_{bm_1,1} \in E_1$ and $e_p(bm_1) = \{\{ep_{bm_1,1}\}, \{ep_{bm_1,2}\}\}$ indicates that only one primary examination tasks is needed to accomplish module bm_1 , the occurrence of $ep_{bm_1,2}$ in the second semester of E' is redundant and reflected by the second occurrence of module bm_1 in $a(E')$.

The relation between primary and secondary examinations tasks is even more intricate since secondary tasks may need to be accomplished before primary ones. We represent such dependencies by a relation $D \subseteq 2^{2^{E_s}} \times 2^{E_p}$ between alternative sets of secondary examination tasks and sets of primary ones.⁵ More precisely, any such dependency in D

⁵ For simplicity, we refrain here from defining these relations in a module dependent way. Without loss of generality, we can thus assume that only examination tasks associated with the same module are put in correspondence.

expresses a temporal constraint requiring that one of the sets of secondary examination tasks must be accomplished before or in the same semester as the set of primary constraints.

For example, the dependencies $(\{\{es_{bm_1,2}\}\}, \{ep_{bm_1,1}\})$ and $(\{\{es_{bm_1,2}\}\}, \{ep_{bm_1,2}\})$ express that weekly exercises of module bm_1 (viz. $es_{bm_1,2}$) must be successfully completed before a student can take a written exam ($ep_{bm_1,1}$) or hand in the final project report ($ep_{bm_1,2}$). None of the primary examinations depend on secondary examination task $es_{bm_1,1}$, which is only needed to accomplish the module itself.

We are now ready to formalize the concept of an admissible examination plan.

A *basic study regulation problem with examination tasks* is a pair $(\mathcal{B}, \mathcal{E})$, where \mathcal{B} is a basic study regulation problem and $\mathcal{E} = (E_p, E_s, e_p, e_s, D, R_{eg}, R_{et})$ where E_p, E_s, e_p, e_s, D are as defined above, and $R_{eg} \subseteq 2^{(2^{E_p \cup E_s})^n}$ and $R_{et} \subseteq 2^{(2^{E_p \cup E_s})^n}$ are sets of global and temporal constraints on examination plans, respectively, as given next.

In doing so, we extend the above abbreviation of $(E_i)_{i=1}^n$ by \overline{E} to sets of sets, that is, $\overline{X} = \bigcup_{x \in X} x$ for any set X of sets. Now, given a set of modules M along with primary and secondary examination tasks $E_p \cup E_s$, we define the following constraints:

$$R_{eg} = \{ \{ (E_i)_{i=1}^n \subseteq (E_p \cup E_s)^n \mid \overline{E} \cap \overline{e_p(m) \cup e_s(m)} \neq \emptyset \text{ implies} \quad (27)$$

$$\overline{E} \cap \overline{e_p(m)} \in e_p(m) \text{ for all } m \in M \}, \quad (28)$$

$$\{ (E_i)_{i=1}^n \subseteq (E_p \cup E_s)^n \mid \overline{E} \cap \overline{e_p(m) \cup e_s(m)} \neq \emptyset \text{ implies} \quad (29)$$

$$\overline{E} \cap \overline{e_s(m)} \in e_s(m) \text{ for all } m \in M \} \} \quad (30)$$

$$R_{et} = \{ \{ (E_i)_{i=1}^n \subseteq (E_p \cup E_s)^n \mid E_i \cap E_j = \emptyset, 1 \leq i < j \leq n \}, \quad (31)$$

$$\{ (E_i)_{i=1}^n \subseteq (E_p \cup E_s)^n \mid \text{for all } (X, W) \in D, \quad (32)$$

$$W \subseteq \overline{E} \text{ implies there is some } V \in X \text{ such that} \quad (33)$$

$$V \subseteq \overline{E} \text{ and } \max(\{i \mid V \cap E_i \neq \emptyset\}) \leq \min(\{i \mid W \cap E_i \neq \emptyset\}) \} \} \quad (34)$$

The two global constraints in R_{eg} , ranging from (27) to (30), ensure that if an examination task associated with a module m is part of an examination plan, then a valid combination of primary and secondary examination achieving module m must also be a part of the examination plan, with no further unnecessary examination tasks associated with the module being taken. The first temporal constraint in R_{et} , given in (31), forbids the same examination task from being completed in two distinct semesters. Finally, the second temporal constraint in (32) to (34), implements the meaning of dependencies as presented when introducing the concept.

Last but not least, an examination plan $(E_i)_{i=1}^n \subseteq (E_p \cup E_s)^n$ for a basic study regulations problem with examination tasks $(\mathcal{B}, \mathcal{E})$ is *admissible* if $a((E_i)_{i=1}^n)$ is a study plan for \mathcal{B} and $(E_i)_{i=1}^n \in \bigcap_{r \in R_{eg} \cup R_{et}} r$.

For illustration, let us return to examination plan $(E_i)_{i=1}^4$. We have already seen in Section 2 that $a((E_i)_{i=1}^4)$ constitutes a valid study plan for the *Cognitive Systems* program. Similarly, the actual examination plan $(E_i)_{i=1}^4$ satisfies all constraints in (27) to (34), which warrants its admissibility. Unlike this, the aforementioned modified examination plan E' fails to be admissible. This is because, first, R_t , or more specifically (17), forces modules to occur at most once in a study plan and, second, R_{eg} forbids to have both $ep_{bm_1,1}$ and $ep_{bm_1,2}$ in $\overline{E'}$. Hence, both $a(E')$ and E' violate constraints on study and examination plans, respectively.

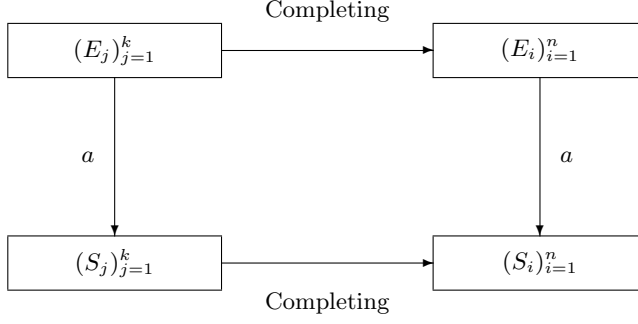


Fig. 1. Relations between partial and total study and examination plans

Although we focused so far on total study and examination plans, in practice, a common use-case consists in completing partial ones. The above formalization directly carries over to partial plans. Even though we do not formally elaborate this, we can show the relationships among partial and total study and examination plans given in Figure 1.

4 Encoding study regulations

In this section, we present an ASP-based approach to represent study regulations and generate valid study plans. We explain in detail the representation of basic study regulations, and we discuss briefly its extension to examination tasks.⁶ As usual, the representation is divided in two parts: a specific instance and a general encoding. The instance represents the elements of a specific study regulation by a set of facts, while the encoding provides the semantics associated with study regulations. Given an instance that represents one study regulation, the answer sets of the encoding together with the instance correspond to the study plans for that study regulation.

We try to keep the notation as close as possible to the formalizations of the previous sections. We use the same symbols as before for sets and functions, but always in lowercase, to adapt to the conventions of ASP. For example, the sets \bar{S} , S_i and M_w are denoted by the terms \mathbf{s} , $\mathbf{s(i)}$ and $\mathbf{m(w)}$, respectively. In what follows, we may use the logic programming notation to refer to those sets and functions.

Listing 1 shows the first part of the instance `cogsys.lp` for the *Cognitive Systems* master, that specifies the sets and functions of the study plan (without examination tasks). Sets are defined using atoms of the form `in(e,a)` that represent that the element \mathbf{e} belongs to the set \mathbf{a} . For example, the atom `in(bm1,b)` expresses that $\mathbf{bm1} \in \mathbf{b}$. Functions are defined similarly, using atoms of the form `map(f,e,v)` that represent that the value of the function \mathbf{f} applied to the element \mathbf{e} is \mathbf{v} . For example, `map(c,bm1,9)` expresses that $\mathbf{c(bm1) = 9}$. To define the facts more compactly, we make extensive use of pooling using the operator `';`. For example, the three facts `in(bm1,b)`, `in(bm2,b)`, and `in(bm3,b)` defining the set \mathbf{b} are captured by the single rule `in((bm1;bm2;bm3),b)` in Line 2.

⁶ The complete encoding for study regulations with examination tasks is available at <https://github.com/potassco/study-regulations/tree/v1.0.0>

```

1  % b, f, a, o and p                                % c
2  in ((bm1 ; bm2 ; bm3), b).                        map (c, (bm1 ; bm2 ; bm3), 9).
3  in ((fm1 ; fm2 ; fm3), f).                        map (c, (fm1 ; fm2 ; fm3), 6).
4  in ((am11 ; am12 ; am21), a).                     map (c, (am11 ; am12 ; am21), 6).
5  in ((am22 ; am31 ; am32), a).                     map (c, (am22 ; am31 ; am32), 6).
6  in (E,o) :- in (E, (f ; a)).                      map (c, (pm1 ; pm2 ; pm3), 12).
7  in ((pm1 ; pm2 ; pm3), p).                        map (c, im, 15).
8                                                    map (c, msc, 30).

10 % m                                                  % s
11 in (E,m) :- in (E, (b ; f ; a ; p)).              map (s, bm1, w ; s, bm2, s ; s, bm3, w).
12 in ((im ; msc), m).                               map (s, (fm1 ; fm2 ; fm3), w).
13                                                    map (s, (am11 ; am12 ; am21), e).
14                                                    map (s, (am22 ; am31 ; am32), e).
15                                                    map (s, (pm1 ; pm2 ; pm3), e).
16                                                    map (s, (im1 ; msc1), e).

18 % e                                                  % l and u
19 in (fm1, e).                                       map (l, b, 27 ; l, o, 24 ; l, p, 24 ; l, m, 120).
20                                                    map (u, b, 27 ; u, o, 24 ; u, p, 24 ; u, m, 120).

```

Listing 1. *First part of the instance of the Cognitive Systems master in cogsys.lp.*

The definitions of the constraints in (14)-(20) provide the conditions that every study plan $(S_i)_{i=1}^n \subseteq M^n$ must satisfy. These conditions usually refer to operations over sets, that we represent in ASP using prefix notation. For example, the condition of (14) refers to the intersection of the sets \overline{S} and F , that is denoted in the logic program by the term $\text{int}(s, f)$. Other terms can be used to denote the union, subtraction and complement of sets. The regulation constraints could in principle be very diverse, but in our investigation of various study regulations we have found that they can be captured by a few types of general constraints, that we represent in ASP by different predicates. The general encoding gives their semantics, while the specific instance of each study regulation provides facts over those predicates to represent the corresponding constraints. Listing 2 shows the second part of our example instance, that specifies the constraints of the *Cognitive Systems* master. It uses atoms of the following form, with the associated meaning (where A and B denote sets, F denotes a function, and L and U are integers):

- $\text{empty}(A)$ means that $A = \emptyset$,
- $\text{equal}(A, B)$ means that $A = B$,
- $\text{subseteq}(A, B)$ means that $A \subseteq B$,
- $\text{card}(A, \text{leq}, U)$ means that $|A| \leq U$,
- $\text{sum}(A, F, \text{bw}, (L, U))$ means that $L \leq \sum_{e \in A} F(e) \leq U$, and
- $\text{sum}(A, F, \text{geq}, L)$ means that $L \leq \sum_{e \in A} F(e)$.

The general encoding includes more predicates to represent other relations among sets, like proper subset or superset, and it also allows other types of comparisons within atoms of the predicates $\text{card}/3$ and $\text{sum}/4$. Using these predicates, the global constraints (14)-(16) are captured in Lines 23-25. The first constraint consists of two conditions, and this is accordingly represented by two facts. Line 24 uses pooling to refer to all the sets in $\{b, o, p, m\}$, and atoms over $\text{map}/3$ to capture the values L and U of the functions l and u applied to those sets. The last rule of the block defines a new set gc3 that consists of im and msc , and compares it via subseteq with s . Temporal constraints are represented in Lines 28 to 31. The first three use atoms of the predicate $\text{empty}/1$ that refer to $m(w)$, $m(s)$, and the specific sets of modules $s(i)$ of each semester i . The last one defines the set tc4

that consists of **msc**, applies to it a new kind of temporal operator called **before**, and uses the resulting term in an atom of predicate **sum/4**. The term **before(tc4)** denotes the set of modules that occur in the study plan before some element of **tc4**; in this case, before the module **msc**. Using our previous mathematical notation, the set **before(tc4)** is $\{m \in M \mid m \in S_i \text{ and there is some } m' \in \text{tc4} \cap S_k \text{ such that } i < k\}$. The general encoding includes other similar operators like **after** or **between**.

```

22 % global constraints
23 card( e,leq,2 ). equal( int( s,f ),e ).
24 sum( int( H,s ),c,bw, ( L,U ) ) :- H = ( b;o;p;m ), map( l,H,L ), map( u,H,U ).
25 in( im,gc3 ). in( msc,gc3 ). subseq( gc3,s ).

27 % temporal constraints
28 empty( int( s(I),s(J) ) ) :- I = 1..n, J = 1..n, I < J.
29 empty( int( m(w),s(2*K) ) ) :- K = 1..n, 1 <= 2*K, 2*K <= n.
30 empty( int( m(s),s(2*K-1) ) ) :- K = 1..n, 1 <= 2*K-1, 2*K-1 <= n.
31 in( msc,tc4 ). sum( before( tc4 ),c,geq,90 ).

```

Listing 2. *Second part of the instance of the Cognitive Systems master in cogsys.lp.*

Listing 3 shows the general encoding in **encoding.lp**. It takes as input the constant **n** that gives the length of the study plan. This constant is used by the choice rule in Line 2 to generate the possible study plans, represented by the sets **s(i)** for **i** between 1 and **n**. Then, Line 5 defines the set **s** as the union of all **s(i)**'s, and Line 8 defines the sets **m(w)** and **m(s)**. After this, Lines 11-20 handle the additional sets that may occur in the constraints. The first block of rules identifies the sets that occur as arguments in the constraints. Then, the rules in Lines 16 and 17 recursively look for the sets occurring inside the operators **int** and **before**. The encoding contains other similar rules for the other operators, but we do not show them here. Once all the new sets have been identified, additional rules provide their definition. Line 19 defines the intersection of two sets, and Line 20 defines the modules occurring before some module of another set. The complete encoding includes further rules for the other operators. The next part of the encoding, in Lines 23-33, enforces the constraints. The first ones about **empty/1**, **subseq/2** and **equal/2**, use the predicate **in/2** to eliminate the cases that are not consistent with the constraints, while those about **card/3** and **sum/4** rely on cardinality and aggregate atoms for that task. For example, the condition $|A| \leq U$ for **card(A,leq,U)** is captured by the cardinality atom $\{ \text{in}(E,A) \} U$, and the condition $L \leq \sum_{e \in A} F(e) \leq U$ for **sum(A,F,bw,(L,U))** is captured by the aggregate atom $L \# \text{sum}\{ V,E : \text{in}(E,A), \text{map}(F,E,V) \} U$. Finally, the last block of statements in Lines 36 and 37 displays the sets **s(i)**.

We can now run the ASP solver *clingo* with the instance for the *Cognitive Systems* master and the general encoding. For **n=4** we obtain, among others, an answer set that corresponds to the admissible study plan *S* of Example 1:

```

clingo -c n=4 cogsys.lp encoding.lp
...
Answer: 1
(bm1,1) (bm3,1) (fm1,1) (am12,1) (bm2,2) (am21,2) (pm1,2)
(im,3) (am31,3) (pm3,3) (msc,4)

```

This problem is solved in less than a second, but we have not evaluated the scalability of our approach yet. Note also that our current encodings are designed for readability, but we foresee that they can be made more efficient.

```

1  % generate
2  { in(E,s(I)) } :- in(E,m), I=1..n.

4  % s = s(1) U ... U s(n)
5  in(E,s) :- in(E,s(I)).

7  % m(w) and m(s)
8  in(E,m(X)) :- X = (s;w), in(E,m), map(s,E,X).

10 % additional sets
11 set(A) :- empty(A).
12 set(A) :- subseq(A,B).      set(A) :- equal(A,B).
13 set(B) :- subseq(A,B).      set(B) :- equal(A,B).
14 set(A) :- card(A,R,L).      set(A) :- sum(A,M,R,L).
15 %
16 set(A) :- set(int(A,B)).     set(B) :- set(int(A,B)).
17 set(A) :- set(before(A)).
18 %
19 in(E, int(A,B)) :- set(int(A,B)), in(E,A), in(E,B).
20 in(E1,before(A)) :- set(before(A)), in(E1,s(I)), in(E2,A), in(E2,s(J)), I < J.

22 % constraints
23 :- empty(A), in(E,A).
24 %
25 :- subseq(A,B), in(E,A), not in(E,B).
26 %
27 :- equal(A,B), in(E,A), not in(E,B).
28 :- equal(A,B), not in(E,A), in(E,B).
29 %
30 :- card(A,leq,U), not { in(E,A) } U.
31 %
32 :- sum(A,F,bw,(L,U)), not L #sum{ V,E : in(E,A), map(F,E,V) } U.
33 :- sum(A,F,geq,L), not L #sum{ V,E : in(E,A), map(F,E,V) }.

35 % display
36 #show .
37 #show (M,I) : in(M,s(I)).

```

Listing 3. Encoding for all study regulations in *encoding.lp*.

The extension to handle examination tasks is not involved. The first part of the instance is extended by the specification of the sets E_p , E_s and D using predicate `in/2`, and of the functions e_p and e_s using predicate `map/3`. The second part specifies the constraints in R_{eg} and R_{eg} . As an example, the first global constraint is represented by the rule

```

implies(
  neg(empty(int(ee,expand(union(EP,ES))))),
  in'(int(ee,expand(EP)),EP)
) :- in(M,m), map(ep,M,EP), map(es,M,ES).

```

Looking at the body, variable M represents a module, EP denotes $e_p(M)$, and ES denotes $e_s(M)$. In the second line, the term `ee` refers to the set \overline{E} , defined in the extended general encoding, while the set operator `expand` takes one set of sets and returns the union of the elements of that set. In the third line, relation `in'` is a version of `in` where the first argument is a set and the second is a set of sets. With this, the second line refers to the set $\overline{E} \cap \overline{e_p(M) \cup e_s(M)}$ and checks whether that set is not empty, and the third line checks if $\overline{E} \cap e_p(M) \in e_p(M)$. All together, the rule states that for every module M the condition of the second line implies the condition of the third one, just like the constraint (27)-(28).

The general encoding for examination tasks is extended accordingly to accomodate the new set operations, like `expand`, and the logical connectives, like `implies` or `neg`. The

main change of the encoding takes place at the generation part, where the choice rule of Line 2 is replaced by another choice rule that generates possible examination plans together with a normal rule that implements function *a* and defines the corresponding possible study plans.

5 ASP-driven user interface

In this section, we sketch our interactive prototypical User Interface (UI) for creating study plans in accordance with study regulations. Notably, the UI is generated and driven by ASP, more precisely the *clinguin* system.⁷



Fig. 2. User interaction via mouse actions in *clinguin*.

A detailed description of the UI can be found in our preliminary publication, where *tkinter* is used for rendering. For a complement, we concentrate in what follows on the modern web-based front-end *Angular*.^{8,9} To this end, *clinguin* uses a few dedicated predicates to define UIs and the treatment of user-triggered events. This simple design greatly facilitates the specification of continuous user interactions with an ASP system, in our case *clingo* (Kaminski et al. 2023). More precisely, the UI is defined by predicates `elem/3`, `attr/3` and `when/4`, for specifying the UI’s layout, style and functionality, respectively.

We show in Listing 4 the relevant sections of the encoding used to generate the UI snapshots in Figure 2.¹⁰ This encoding is passed along with our study regulations encoding from Section 4 to *clinguin*, which allows for browsing and completing (partial) study plans.

Line 1 creates a window element. Line 3 creates a container for each semester, which is placed in the window. Lines 5 and 6 define the title container for each semester, assigning values to the attribute `class` to define the blue background, bold font, padding and margin. Lines 9 and 10 define a dropdown menu with the text ‘Assign module’.

The possible modules to be assigned are defined in predicate `possible_module/1` on Line 12, using the assignments that appear in any model (union) but are not in all models (intersection), via predicates `_any/1` and `_all/1`, respectively. This predicate is used in Lines 13 to 15 for defining the items in the dropdown menu.

In the second snapshot of Figure 2, we notice that module *bm₂* does not appear in the options since there is no model where this module is assigned to the third semester. When

⁷ <https://clinguin.readthedocs.io/en/latest>

⁸ <https://angular.io>

⁹ A detailed account of *clinguin* has also been submitted to ICLP’24.

¹⁰ The full encoding can be found in <https://github.com/potassco/study-regulations/tree/v1.0.0>

```

1  elem(w, window, root).

3  elem(s(I), container, w) :- semester(I).

5  elem(s_t(I), container, s(I)) :- semester(I).
6  attr(s_t(I), class, ("bg-primary"; "bg-opacity-50";
7                      "fw-bold"; "p-2"; "m-1"      )) :- semester(I).

9  elem(s_dd(I), dropdown_menu, s_t(I)) :- semester(I).
10 attr(s_dd(I), selected, "Assign module") :- semester(I).

12 possible_module(I,E) :- _any(in(E,s(I))), not _all(in(E,s(I))).
13 elem(s_ddi(I,E), dropdown_menu_item, s_dd(I)) :- possible_module(I,E).
14 attr(s_ddi(I,E), label, E) :- possible_module(I,E).
15 when(s_ddi(I,E), click, call, add_assumption(in(E,s(I)))) :-
16     possible_module(I,E).

18 assigned_module(I,E) :- _all(in(E,s(I))).
19 assigned_module(I,E) :- in(E,s(I)), _clinguin_browsing.
20 elem(s_module(I,E), container, s_modules(I)) :- assigned_module(I,E).
21 attr(s_module(I,E), height, C*10) :- assigned_module(I,E), map(c,E,C).

```

Listing 4. An encoding for the prototype UI (*ui.lp*)

the module bm_3 is clicked, Line 15 adds the corresponding atom as an assumption (Andres et al. 2012; Alviano et al. 2023). Lines 18 and 19 define the modules that are assigned to a semester (gray boxes). These can be the ones appearing in all models (Line 18) due to a user selection or inference, or the ones in the current model when the user is browsing solutions, as indicated by atom `_clinguin_browsing`.

Predicate `assigned_modules/2` is then used in Lines 20 and 21 to create the corresponding container, using the number of credits of the module to define the height in Line 21. The third snapshot shows module ‘ bm_3 ’ assigned to the third semester after the previous click. Modules selected by the user include a button marked with ‘x’ to remove the selection; unlike assignments forced by the encoding, such as module ‘ m_{sc} ’ in the fourth semester. Upon clicking in the ‘Next’ button, the last snapshot shows one full study plan consistent with the user’s selections.

6 Related work

ASP, event calculus and process mining techniques were already used in (Wagner et al. 2023) for solving study regulation problems. However, Wagner et al. (2023) presents an overview of the *AIStudyBuddy* project, and contains neither a formalization of study regulations nor any implementation details. Unlike this, Samaranyake et al. (2023) presents a web-based Decision Support System for a degree planning problem along with a mathematical formalization. Degree requirements are mentioned but not formalized. Also, Baldazo et al. (2023) considers educational planning problems. This includes stress and learning effects on students in a personalized study plan generation. It aims at reducing the duration of student plans and is implemented via integer linear programming. Schneider et al. (2018) present a curriculum timetable validation tool by modeling constraints in language *B*. Finally, Shen et al. (2022) present a data-driven approach for implementing

a course recommendation algorithm. A traditional collaborative filtering algorithm is extended to consider additional course path data.

7 Summary

We have introduced a conceptualization of study regulations based on set-based constraints. This formalization is both simple and general to capture a wide range of study regulations. We indicated how the basic formulation easily extends to more complex constructions. This will be further elaborated in future work. The identification of basic principles in study regulations also allowed us to obtain a very general ASP encoding. The building blocks of each study regulation are captured in terms of facts so that the actual encoding is also applicable to a large range of study programs. Finally, we have described an ASP-driven user interface for interactive elaboration of study plans. Again, the interface is designed in a generic way and broadly applicable. Moreover, this case study serves as a nice illustration of *clinguin* and how it can be used for interactive ASP applications.

Finally, study regulations offer a very rich playground for applications of knowledge representation and reasoning techniques. Study plans have a light temporal flavor and resemble finite traces in linear temporal logic (De Giacomo and Vardi 2013). The creation of a study plans amounts to a configuration task, which also brings about interaction and explainability. Finally, we have so far only been concerned with the hard constraints emerging from study regulations but there is so much commonsense knowledge involved, like defaults, preferences, deontic laws, updates, etc.

Acknowledgments. This work was partly funded by DFG grant SCHA 550/15 and BMBF project CAVAS+ (16DHBKI024).

References

- ALVIANO, M., DODARO, C., FIORENTINO, S., PREVITI, A., AND RICCA, F. 2023. ASP and subset minimality: Enumeration, cautious reasoning and MUSes. *Artificial Intelligence*, 320, 103931.
- ANDRES, B., KAUFMANN, B., MATHEIS, O., AND SCHAUB, T. Unsatisfiability-based optimization in clasp. In DOVIER, A. AND SANTOS COSTA, V., editors, *Technical Communications of the Twenty-eighth International Conference on Logic Programming (ICLP'12)* 2012, volume 17, pp. 212–221. Leibniz International Proceedings in Informatics (LIPIcs).
- BALDAZO, J., YASMÍN, R., AND NIGENDA, R. 2023. Scheduling personalized study plans considering the stress factor. *Interactive Learning Environments*, 1–20.
- DE GIACOMO, G. AND VARDI, M. Linear temporal logic and linear dynamic logic on finite traces. In ROSSI, F., editor, *Proceedings of the Twenty-third International Joint Conference on Artificial Intelligence (IJCAI'13)* 2013, pp. 854–860. IJCAI/AAAI Press.
- HAHN, S., MARTENS, C., NEMES, A., OTUNUYA, H., ROMERO, J., SCHAUB, T., AND SCHELLHORN, S. Reasoning about study regulations in answer set programming (preliminary report). In ARIAS, J., BATSAKIS, S., FABER, W., GUPTA, G., PACENZA, F., PAPADAKIS, E., ROBALDO, L., RÜCKSCHLOSS, K., SALAZAR, E., SARIBATUR, Z., TACHMAZIDIS, I., WEITKÄMPER, F., AND WYNER, A., editors, *Proceedings of the International Conference on Logic Programming Workshops* 2023, volume 3437 of *CEUR Workshop Proceedings*. CEUR-WS.org.
- KAMINSKI, R., ROMERO, J., SCHAUB, T., AND WANKO, P. 2023. How to build your own asp-based system?! *Theory and Practice of Logic Programming*, 23, 1, 299–361.

- LIFSCHITZ, V. 2002. Answer set programming and plan generation. *Artificial Intelligence*, 138, 1-2, 39–54.
- SAMARANAYAKE, S., GUNAWARDENA, A., AND MEYER, R. 2023. An interactive decision support system for college degree planning. *Athens Journal of Education*, 10, 101–116.
- SCHNEIDER, D., LEUSCHEL, M., AND WITT, T. 2018. Model-based problem solving for university timetable validation and improvement. *Formal Aspects of Computing*, 30, 545–569.
- SHEN, Y., LI, H., AND LIAO, Z. Online education course recommendation algorithm based on path factors. In *Proceedings of the Fifth International Conference on Information Systems and Computer Aided Education (ICISCAE'22)* 2022, pp. 257–260. IEEE Computer Society Press.
- WAGNER, M., HELAL, H., ROEPKE, R., JUDEL, S., DOVEREN, J., GOERZEN, S., SOUDMAND, P., LAKEMEYER, G., SCHROEDER, U., AND VAN DER AALST, W. A combined approach of process mining and rule-based ai for study planning and monitoring in higher education. In *Proceedings of the International Conference on Process Mining (ICPM'22): Process Mining Workshops* 2023, pp. 513–525. Springer-Verlag.