How to Automate Feedback on Diagrammatic Reasoning With a Relevant Degree of Freedom?

Géraldine Brieven, Lev Malcev, Benoit Donnet
Université de Liège, Institut Montefiore, Belgium

***Abstract*—This paper considers CAFÉ 2.0, an Automated Feedback system designed to support students' diagrammatic reasoning in STEM disciplines. CAFÉ 2.0 relies on a predefined error library, metamodels, and rules to correct students' solutions and deliver formative feedback. Implementing such a system requires a balance between constraining the solution syntax to enable AF and leaving freedom to students to reflect on their solution. This paper aims to evaluate whether the level of freedom provided by our AF system sufficiently prepares students for exams. In the exam, they must reason and construct solutions starting with a blank page.**

This study is conducted in an introductory programming course (CS1), based on two semesters (in 2022 and 2023), where CAFÉ 2.0 supports online homework. Findings reveal a discrepancy between students' performance in online homework and their success on exams. While many students feel comfortable with fill-in-the-blank diagrams in their homework, they struggle with the open-ended nature of exam tasks. Our results show that, among the students who succeeded in their online homework in 2023, 20% were still unable to produce any diagram in the exam. Additionally, 70% of them could not correctly provide a text description of their solution.

To overcome this limitation, this paper proposes an enhanced system that integrates predefined rules with Large Language Models (LLMs). In this framework, LLMs serve as translators. Students can freely create their diagrams and annotate them with their own textual descriptions using a drawing editor. The LLM then maps these representations into a more structured format that aligns with predefined rules. In this way, CAFÉ 2.0 can generate accurate feedback. This transformed representation retains the same informational content as the original, differing only in format. This feature will offer students greater flexibility in constructing their solutions while ensuring that feedback remains precise and consistent by limiting the role of LLMs to translation rather than feedback generation.

***Index Terms*—automated feedback, diagrammatic reasoning, metamodeling, error detection, large language model**

I. INTRODUCTION

To keep students motivated in today's education, regular practice is crucial [1] and must be coupled with effective feedback [2]. This is where Automated Feedback (AF) becomes essential [3]. Our tool, CAFÉ 2.0 [4], addresses this need by supporting online homework where students can progressively refine their solutions using AF. Currently, CAFÉ 2.0 is used in two introductory courses —- Computer Science (CS1) and Physics —- with plans to expand to Chemistry and Mathematics next year. In all these courses, students are asked to model their solution using a diagram (i.e., *diagrammatic reasoning* [5]) before developing it (see Sec. II). Development typically consists of equations, predicates, or pieces of code. Previous research has highlighted the benefits of developing spatial skills for mastering STEM disciplines [6], and more specifically for CS1 courses [7].

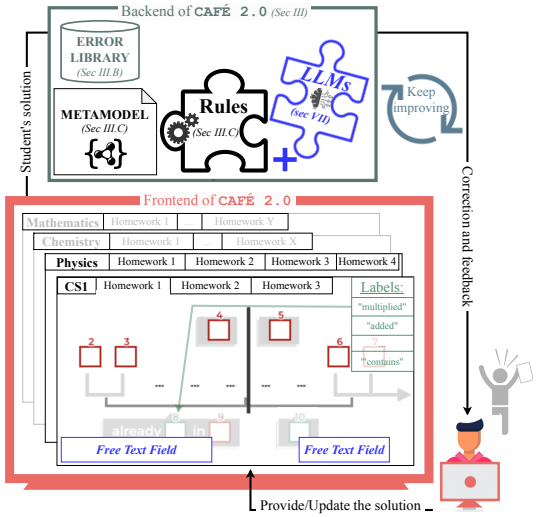


Figure 1. AF on Diagrammatic Reasoning (including free text (TO-BE)).

Fig. 1 illustrates CAFÉ 2.0's interface ("Frontend of CAFÉ 2.0") and the building blocks to automate feedback ("Backend of CAFÉ 2.0"). For each assignment, students work on a prefilled diagram composed of boxes (referred to as *fill-in-the-blank diagram*). Constraining the diagram's shape restricts the range of possible constructions at the syntactic level, simplifying the process of modeling the expected solution (referred to as *metamodeling* [8]). The metamodel, expressed by the instructor in a configuration file, can support the expected content of the boxes, their domain, and the relationships between them. It also contains the error codes each box is exposed to (see Sec. III-C). Once students have filled the diagram, they can submit it for evaluation based on rules implemented by the instructor. They capture students' typical errors, which are stored in an error library (see Sec. III-B). When students receive the feedback, they can use it to improve and resubmit their solution. This is illustrated in Fig. 1 and detailed in Sec. III. This figure also highlights in blue how Large Language Models (LLM) could enhance CAFÉ 2.0 to support a less constrained solution (see Sec. VII). It would align more closely with the exam setting, where students are expected to draw diagrams by hand from scratch. This upgrade is motivated by the results (see Sec. V), which reveal a gap between the online homework and the exam setting in a CS1 course. This gap is measured based on students performance.

We further analyze it by identifying students' mistakes and assessing their ability to transfer their understanding between different online homework and to the exam.

II. Motivation of Diagrammatic Reasoning in STEM Courses

The design approach of CAFÉ 2.0 is relevant for multiple STEM topics. The general goal of this tool is to allow first-year students to informally model their solutions while supporting AF. From a course perspective, this requires instructors to introduce students to Model-Driven Development [8], where students are encouraged to formulate their solutions both textually and graphically. This approach is intended to foster abstract thinking by guiding students to use a limited set of components and symbols to express their reasoning.

In our CS1 course, we apply this approach when students must implement a *loop* (i.e., a sequence of instructions that must be repeated a certain number of times). This task is challenging for beginners [9] and a fundamental building block for effective program design. To guide students in implementing a loop, they may use the *Loop Invariant* [10]. It relies on predicates and describes the state of the variables involved in the solution. The barrier to this approach is that predicates are not accessible for first-year students [11]. To address this, we teach students to design a LOOP DRAWING, an informal version of the Loop Invariant [12]. It is a diagrammatic and textual representation of a predicate, making formalization more accessible by removing the mathematical overhead of a predicate. Similarly, in Kinematics, students may be asked to compute the friction coefficient for a scenario involving forces acting on an object, using equations. To easily derive these equations, it is helpful to first create a force diagram (also called free body diagram) that clearly illustrates the direction and application of each force [13]. In Trigonometry, reasoning over a drawing can help students better grasp the concept of the trigonometric functions [14]. These first three examples are illustrated in Fig. 2. Finally, in Chemistry, first-year students may be asked to write chemical reactions. Before directly writing it, students can be taught to graphically represent each chemical reactant and illustrate the atoms transitions between reactants to form the chemical products [15].

III. Automated Feedback via Café 2.0

A. Syntax Constraints on Students' Model

From a technical perspective, automating assessment of exercises requires finding a balance between pre-structuring the solution (to anticipate and match student answers with feedback messages) and allowing some degree of syntax freedom (to train students to solve problems from scratch).

Fig. 2 illustrates the different levels of constraint a solution format can be subjected to. The highest level of freedom is achieved when students can express their solutions in natural language. At the other end of the spectrum, the most constrained format requires students to select from predefined multiple-choice options. Between these extremes, diagrams with predefined components offer a structured yet flexible way for students to represent their ideas. Fig. 2 shows three diagram formats tailored to different STEM disciplines (Maths, CS, and Physics). In the three lowest diagrams, students must only complete boxes. For example, in the CS1 exercise, each box is filled either with a simple expression, or a predefined symbol. Similarly, in the Kinematics exercise, students must drag symbols (representing gravity, normal force, etc.) to the appropriate arrows on the diagram. This level of freedom is currently the highest one CAFÉ 2.0 can support (indicated by blue frames in Fig. 2). Previous research [16] corroborates this choice by encouraging instructors to privilege students' problem-solving process within structured parameters rather than allowing complete syntax freedom. While this may hold true for learning purposes, in all of these courses, students are ultimately expected to provide a solution from scratch in the exam, along with a text description. That positions the exam setting three levels upper the one adopted by CAFÉ 2.0. With less syntactic constraints, the fill-in-the-blank diagram with free fields allows students to freely write in the fill-in-the-blank boxes, without the support of a symbol table. Finally, getting closer to the exam setting, students should build their diagram from scratch, using a drawing editor. They first select the type of diagram they find relevant. The more abstract the discipline, the less easy this choice. For example, when the problem consists of assessing the friction coefficient so that a car is standing on a steep road, it is straightforward that the type of diagram is a car standing on a steep road and not a ball thrown in the air. On the opposite, when a student must express a cosine, the trigonometric circle is a fully abstract representation they should know already. Between these extremes, in a CS1 course, students should opt to traverse an interval of integers to compute the factorial of a given number. Still, some of them may select an array of characters as type of diagram because they just blindly replicate the solution to other problems they remember. Once students have chosen the type of diagram, they must select, arrange, and complete it with components (lines, arrows, etc.) and texts.

B. The Error Library Construction

To provide and manage AF, typical errors must be identified in advance. For this purpose, we define an error library. It classifies them [17] and maps them to feedback messages.

Fig.3 illustrates how our error library has been populated with typical errors, based on prior experience [18]. Currently, in the Physics course, typical errors have been gathered for one topic – Kinematics – to support exercises with AF. In the CS1 course, typical errors gathered in seven topics have been treated (e.g., Problem Decomposition, Loop Modeling etc.). We analyzed eight midterms, 40 homework submissions (five homework per year), and 24 exams[1] from 2016 to 2023. A majority of typical errors cross-check errors presented in other research [19].

Each student's error was defined through an error code in our error library and mapped to a specific misunderstanding.

[1] In our country, for our course, students have one official exam (in January) with two potential resits (June and August).

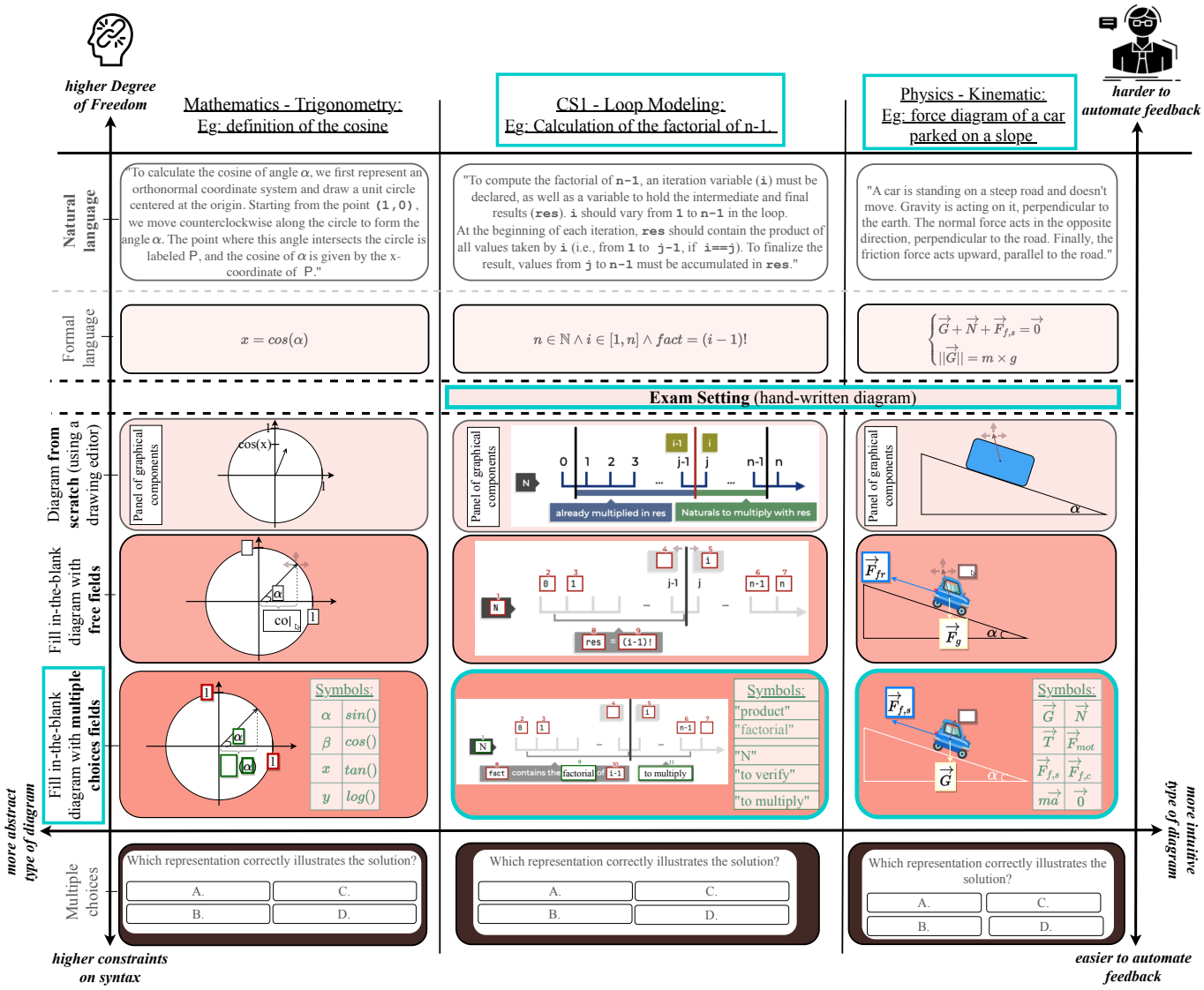


Figure 2. Trade-off when designing au AF system. Blue frames highlight the formats supported by CAFÉ 2.0 (with respect to the exam setting).

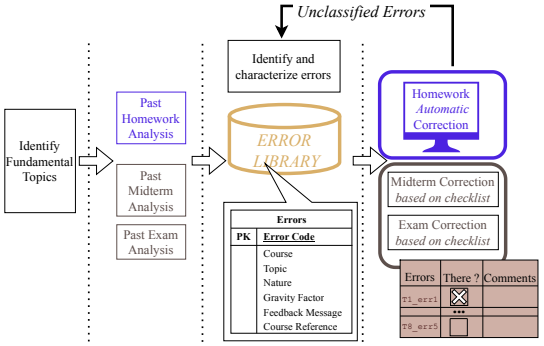


Figure 3. How the error library is fed (for a given topic)?

Defining typical errors requires determining an appropriate level of granularity. For instance, an array indexing error is broader than an off-by-one error, which suggests the student believes arrays are indexed starting from 1. The errors specificity should match the desired accuracy level for feedback. Each typical error is characterized as syntactic or semantic. They are also assigned a gravity factor to aid in grading and prioritizing the feedback messages, with the most significant feedback appearing first. One feedback message reflects one error code. Additionally, feedforward (i.e., guidance for improving the solution) is offered through references in the course that demonstrate the correct application of the concept. A portion of our error library is provided in Appendix B. At the time of writing, the error library includes 136 error codes.

Beyond supporting AF, this error library ensures consistency in course evaluation. For both formative and summative assessments, typical errors from the error library are listed in checklists supporting grading. From a research perspective, it also makes it possible to measure learning gains from instructional experiences by formally assessing students before and after the experience. This approach has supported the

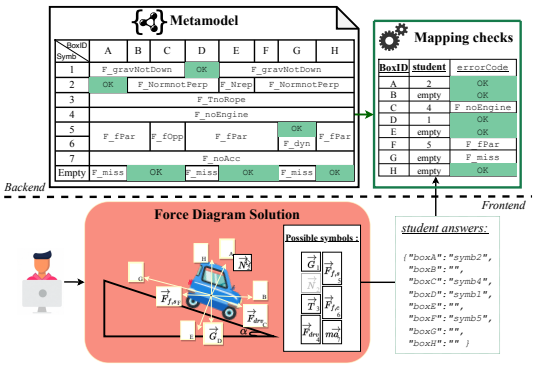


Figure 4. For a given problem statement in Physics, example of fill-in-the-blank diagram, metamodel and check results.

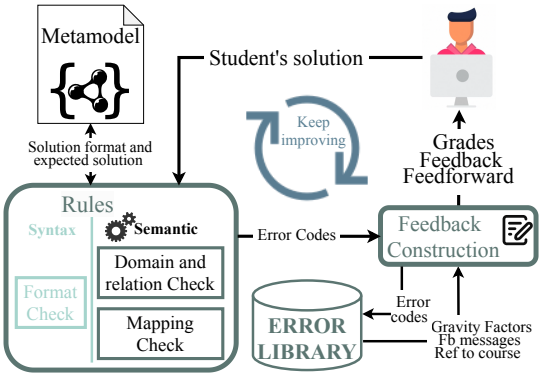
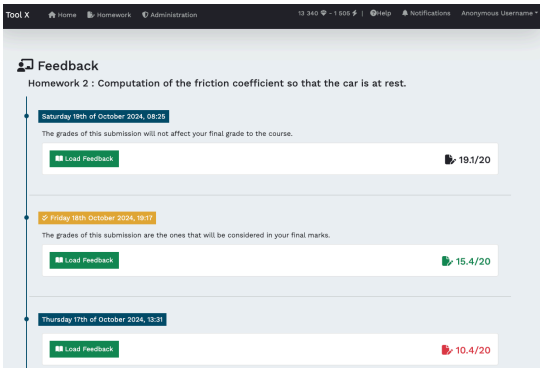


Figure 5. Detailed flow to provide Automated Feedback.

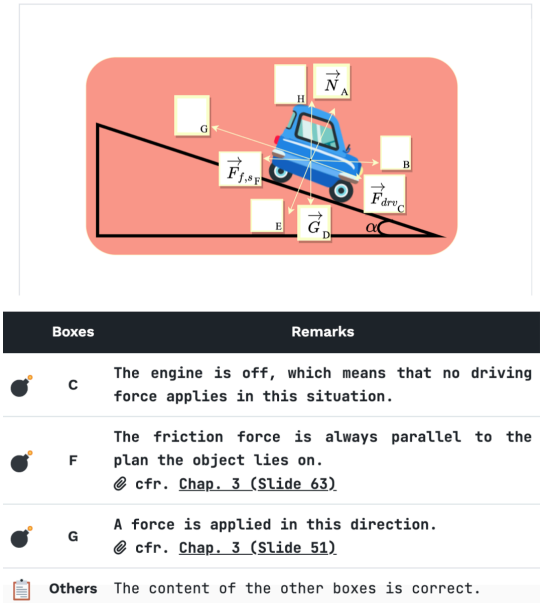
findings of various studies [20], [21].

C. Metamodeling, Correction, and Feedback

As briefly specified in Sec. III-A, when students model a solution in CAFÉ 2.0, they are provided with a fill-in-the-blank diagram with different types of boxes. *Free boxes* expect expressions while *constrained boxes* expect a symbol dragged from a predefined list [12], [4]. The expected content of *free boxes* is characterized by a domain and some relationships with other boxes' content. That is specified in a metamodel. For example, in a CS1 assignment involving iterative constructs, *free boxes* might contain an integer variable index whose domain would be within $[0, N[$ and related to an array size N. More simply, the expected content of *constrained boxes* is a symbol ID, like illustrated in Fig. 4. The figure shows an example of a fill-in-the-blank diagram in Physics containing only *constrained boxes*. In the figure, the student answered both correct and incorrect symbols. The metamodel is represented through a matrix that specifies the expected symbole ID and the typical error to return (if any) for each incorrect symbol. For instance, in box C, the student places a motorized force, while the problem statement indicates that the car is parked. Consequently, the error code `F_noEngine` is triggered during the mapping check. A more complex example of a fill-in-the-blank diagram, metamodel, and associated checks is given in Appendix A, in a CS1 context.



(a) Feedback main page in CAFÉ 2.0.



(b) Example of a piece of feedback in CAFÉ 2.0.

Figure 6. Screenshots of CAFÉ 2.0.

Once all the typical errors made by the student are identified via their error code (as shown in the last column of the "Mapping Checks" section in Fig.4), CAFÉ 2.0 accesses the error library to compute grades and provide feedback and feedforward. This flow is illustrated in Fig.5. It allows students to recognize their misunderstandings and improve their subsequent submissions.

An example of the main feedback page and feedback messages in CAFÉ 2.0 is illustrated in Figure 6. The "bomb" icon highlights pieces of feedback corresponding to a typical error student should fix in subsequent submissions. To draw student's attention, that feedback is also in bold. The "paper clip" icon informs the student about how their submission is processed and interpreted.

IV. METHOD

CAFÉ 2.0 is available in our CS1 course since academic year 2022–2023 and has been deployed in Physics this academic year (2024–2025). Therefore, in this paper, we use data col-

Table I
ONLINE HOMEWORK DESCRIPTION IN THE CS1 COURSE, WITH THE PARTICIPATION RATE FOR EACH ONLINE HOMEWORK.

	Online homework		Participation	
	Subject	Details	2022–2023	2023–2024
#1	Action on digits for all numbers in $[a, b]$	outer loop (spans $[a, b]$) inner loop (counting digits) inner loop (summing digits)	78%	67%
#2	Arrays	Compressing an integer array into another Calculating a checksum	62%	72%
#3	Displaying all numbers in array `T` that follow a property G	Checking whether a number is prime Checking if a number meets the property G Displaying the numbers meeting G	56%	50%

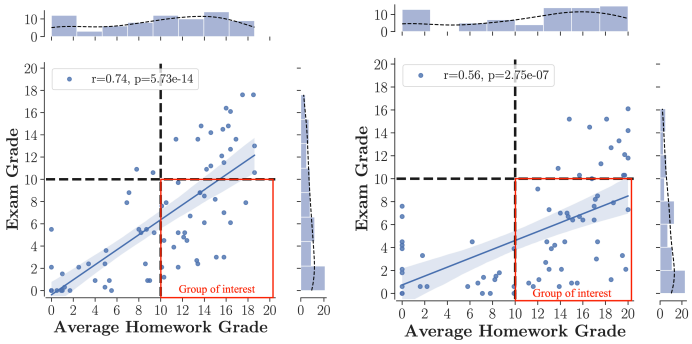
lected during academic years 2022–2023 and 2023–2024 in the context of the CS1 course. In 2022–2023 (resp. 2023–2024), 97 (resp. 101) students registered to the course. 5% (resp. 18%) of them had some prior knowledge in programming. Most of them (68% in 2022–2023, 87% in 2023–2024) were new comers, starting their first-year at the University. Throughout the semester, students participated in ten 2-hour theoretical lectures and ten 2-hour exercise sessions.

Every three weeks, students were also given online homework (five in total over the semester). Solving them usually requires an additional four out-of-class hours of work. The content of these assignments aligns with the chapters taught and practiced in the preceding week(s). Among the five assignments, three focus on training diagrammatic reasoning, specifically within the context of loop programming (see Sec. II). This paper concentrates exclusively on these three online homework. Table I presents a high-level overview of their tasks, including student participation rates. In both years, the participation rate decreased. This trend is due to a drop-out after the midterm [2] as well as the assignments that get more complex.

Online homework is completed across two distinct periods. The first, a formative period, runs from Wednesday to Friday of the same week. During this time, students are allowed up to three attempts, with their final submission determining their grade. Aligned with Sambell et al.'s principles of Assessment for Learning (AfL), this low-stakes activity contributes 10% to the final grade (2% per assignment) [22]. The second period, focused solely on practice, begins immediately after the formative period and extends until the final exam (typically in January in our region). During this phase, students can make unlimited attempts to complete the homework, but their performance does not affect their grade. Consistent with AfL principles, this practice-only period provides opportunities for reinforcement and feedback, helping students prepare effectively for the exam [22].

Online homework is completed using CAFÉ 2.0, which tracks and logs typical errors made by students during each submission. At the start of the semester for each year under study, students were informed about their participation in a research study and given the option to opt out. All students provided their consent, and no one chose to opt out during the semester. Additionally, all data collected by CAFÉ 2.0 was anonymized before being analyzed.

[2]Our students have a midterm for most of their courses. The midterm is usually organized between Homework #1 and #2 in Table I.



(a) 2022–2023 (b) 2023–2024

Figure 7. Correlation between average grades to the online homework and exam for both academic years. Students who did not participate in one or both assessments are not accounted in this graph ($N_{2022} = 74$ and $N_{2023} = 72$).

V. RESULTS

In this section, we aim at assessing the relevance of the trade-off we made to implement CAFÉ 2.0 (see Sec. III-A).

Fig. 7 illustrates the correlation between the average grade for online homework (X-axis) and the corresponding question on the final exam (Y-axis). Four regions are highlighted:

- Bottom-left: Students failing both the online homework and the exam question (29 in 2022–2023, 25 in 2023–2024).
- Top-left: Students failing the online homework but succeeding on the exam question (2 in 2022–2023, 0 in 2023–2024).
- Top-right: Students succeeding in both the online homework and the exam question (18 in 2022–2023, 14 in 2023–2024).
- Bottom-right: Students succeeding in the online homework but failing the exam question (25 in 2022–2023, 32 in 2023–2024).

Fig. 7 suggests that regular practice, which includes AF, benefits some students (top regions). However, a significant proportion of students (our group of interest) failed the exam

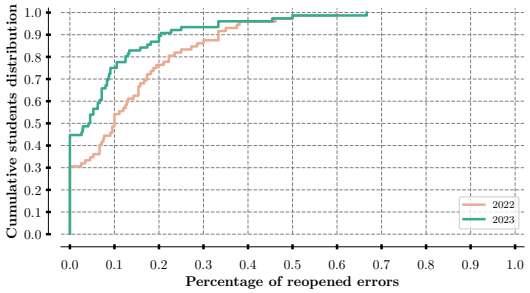


Figure 8. Number of errors students committed in other online homework while they were fixed in previous online homework ($N_{2022} = 72$ and $N_{2023} = 76$).

question despite performing well in the online homework. This discrepancy suggests three possible explanations:

- *Expl. 1*: Some students might rely on external assistance, such as AI tools (e.g., ChatGPT [23]), plagiarism [24][3], or contract cheating [26][4].
- *Expl. 2*: Students may struggle to process feedback and transfer their knowledge to different contexts, a phenomenon known as "compartmentalization of knowledge" [27].
- *Expl. 3*: The final exam question may differ significantly from the online homework tasks. For example, while online homework provides structured prompts (e.g., fill-in-the-blank diagram), the exam requires students to construct solutions from scratch.

This study investigates the latter two hypotheses.

To evaluate *Expl. 2*, we focus on errors students could fix in a given online homework assignment. Then, we inspect how well students avoided repeating them in the context of another online homework assignment. Fig. 8 shows that 30% (2022) and 45% (2023) of students did not repeat previously corrected errors. However, some students repeated up to 60% (2022) and 77% (2023) of their prior errors. Furthermore, 30% (2022) and 17% (2023) of students made between 10% and 25% of the mistakes they had previously corrected. These findings indicate that students generally transfer their understanding across assignments when the tasks have similar syntactic constraints, disproving *Expl. 2* and shifting the focus to *Expl. 3*.

To address *Expl. 3*, we analyzed the groups of interest for each year ($N_{2022} = 26$ and $N_{2023} = 32$) and assessed the percentage of students who: (i) were unable to provide any solution; (ii) failed to represent the appropriate type of diagram aligned with the nature of the problem; and (iii) could not provide a correct text description. Table II presents the results. Results from both years show the same trend, with the most pronounced ones being from 2023. They show that approximately 20% (8% in 2022) of students were unable

[3]We addressed plagiarism by inspecting all student submissions using a script [25] that calculates the percentage of similarity between pairs of submissions from different students. We detected one plagiarism case involving four students in 2022–2023 and two cases in 2023–2024 involving five and three students. Those students were excluded from this study.

[4]Contract cheating means presenting a work written by another person as if it were their own, therefore resorting to a third party to do the assignment.

Table II
ERROR SOURCES IN THE EXAM. PERCENTAGE CONSIDERS ONLY THE STUDENTS WHO SUCCEEDED IN THEIR ONLINE HOMEWORK BUT FAILED IN THE CORRESPONDING EXAM QUESTION.

	Source of error in the exam question	Academic Year 2022-2023	2023-2024
(i)	No solution	7.7%	19.4%
(ii)	Incorrect type of diagram	26.9%	62.5%
(iii)	Incorrect text description	50.0%	68.8%

to provide any diagrammatic representation of their solution, despite successfully doing so in their online homework. Additionally, about 62% (27% in 2022) of students could not identify the correct type of diagram to support their representation. Finally, 69% (50% in 2022) of students failed to correctly textually describe the state of their variables. To deep-dive this last result, we computed that over half (38% in 2022) of the students who struggled to describe variable states on the exam had succeeded when using fill-in-the-blank text with drag-and-drop components in the online homework.

These results suggest that the high level of constraints in the online homework may have limited the effectiveness of AF in preparing students for open-ended exam questions. Sec. VII explores a solution to address this gap, using LLM.

VI. Related Work

In STEM education, Diagrammatic Reasoning (DR) is closely linked to Representational Competence [28] and Computational Thinking [29]. DR specifically promotes problem-solving through visualization, whereas Representational Competence refers to the ability to translate one type of representation into another (e.g., in Kinematics, students might be required to convert a force diagram into equations).

Many tools promote DR (e.g., Scratch, Codex, KiRC inventory [30]). Most of these tools provide passive visualization by either directly depicting potential representations or simulating students' final answers (e.g., code snippets) to enhance their understanding of solution behavior. Some tools take this a step further by enabling interactive, dynamic diagrams that students can manipulate, such as Brilliant [31], which assists in solution discovery. However, these tools are student-driven and lack system checks for consistency.

On the other hand, numerous tools provide AF in STEM disciplines [32]. In CS, specifically, various automated systems have been developed to deliver feedback on programming exercises (e.g.,[33], [34], [35]). Most of these systems rely on test-based feedback, where student code is evaluated using unit tests.

While a comprehensive review of such tools is beyond the scope of this paper, systematic reviews have previously identified and categorized these tools extensively, both within the CS domain [36] and across STEM more broadly [37].

This paper focuses on tools meeting the following criteria: (i) updated after 2022, (ii) promote DR or Writing-to-Learn (WTL), (iii) provide automated feedback (AF), (iv) target pre-university or higher education levels, and, (v) span multiple

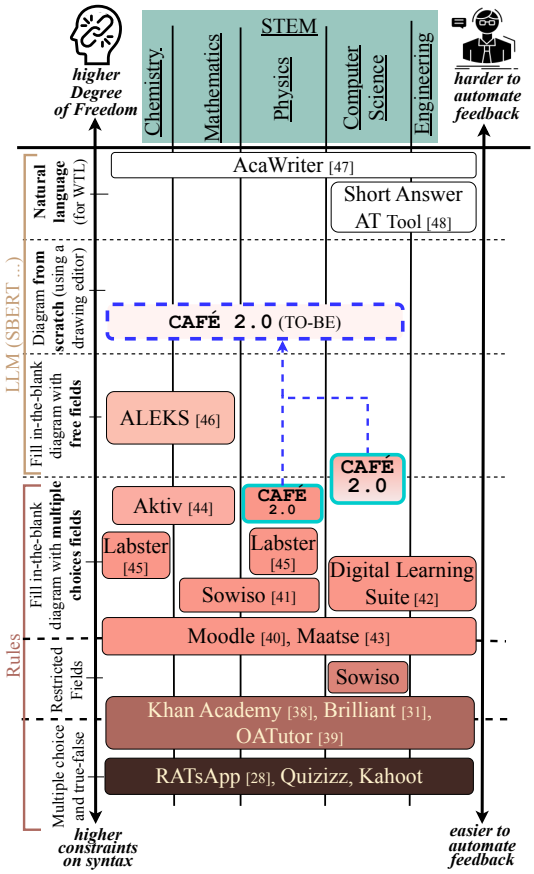


Figure 9. Other AF tools supporting DR in STEM disciplines.

STEM disciplines. Fig. 9 highlights fifteen tools that fulfill these requirements, positioning them upper or lower than CAFÉ 2.0 with respect to their level of syntax constraint.

Kahoot, Quizizz, and RATsApp provide formative feedback for multiple-choice questions, covering a wide range of STEM disciplines. Specifically, RATsApp (Rapid Assessment Tasks App)[28] includes kinematics tasks, with an example similar to ours (see Fig. 4). While RATsApp offers useful features such as scaffolding and a learning dashboard, it lacks flexibility in accommodating diverse students responses. Its primary focus in on expanding topic coverage, rather than increasing the complexity of solution representation.

Khan Academy [38], Brilliant, and OATutor (Open Adaptive Tutor) [39] go beyond supporting multiple-choice questions by incorporating more advanced question types, such as those with restricted input fields (e.g., allowing users to compose expressions like $2x^2 + x - 1$). These tools also stand out for their personalization features. They suggest problem statements based on a student's skill level as determined by their past performance.

At more advanced levels, tools such as Moodle [40], Sowiso [41], and the Digital Learning Suite [42] offer "drag-and-drop" question types. These allow educators to design static diagrams with blank fields that students complete using predefined elements. For example, prior work [40] demonstrates the use of graphical force diagrams in Moodle for this purpose. MAATSE [43] includes a comparable feature but distinguishes itself by offering more advanced feedback through the application of sophisticated rule-based systems.

Building on these tools, Aktiv [44], Labster [45], and ALEKS [46] take feedback to the next level by supporting interactive diagrams that students can freely complete or equations with movable terms in Mathematics, Physics, and Chemistry. These platforms offer feedback on question formats that extend far beyond traditional multiple-choice approaches. CAFÉ 2.0 follows a similar approach but differentiates itself by prioritizing advancements in diagrammatic reasoning practice in future updates, whereas these tools primarily focus on knowledge modeling and content personalization.

Finally, AcaWriter [47] assists in describing a solution. Beyond commenting on syntax (e.g., identifying misspellings), it detects subjective statements or gaps in explanations, thereby capturing the solution's semantics. More specifically, in CS and Engineering, we found a web-based system capable of grading short text answers provided by students [48].

VII. LLM PERSPECTIVE

In Sec. V, the results investigate the gap between online homework and exam conditions. They revealed that while students can semantically construct solutions by selecting correct symbols from a list, they often struggle to formulate their own textual descriptions in the summative assessments. To fill this gap, our goal is to align CAFÉ 2.0 more closely with the exam format by introducing greater flexibility in students' syntax. As shown in Fig. 2, all the solution' representations within the same column convey identical information, differing only in format. To bridge these representations, we propose designing a translator module, capable of converting a solution with minimal syntactic constraints into a predefined format. This would enable the current validation rules to process students's solutions, even if they are initially created with fewer syntactic constraints, as illustrated in Fig. 10. By doing so, to go further, CAFÉ 2.0 could even dynamically adjust the level of syntactic constraints imposed on students based on their abilities or preferences.

For example, a statement that currently requires students to select the symbols "sum", "integers", "0", and "i-1" in order to complete a fill-in-the-blank diagram described by: "The variable [sum] contains the sum of the [integers] from [0] to [i-1]." could be modified to include an empty text field, where the answer would be equivalently interpreted by the LLM. This change would remove a crutch and bring students closer to real exam setting.

The use of LLM offers not only a closer alignment with exam settings, but also advances the adoption of *Writing-To-Learn* (WTL) pedagogy. In STEM courses, WTL involves crafting exercises that prompt students to articulate, in natural language, the components of their solutions before implementing them. This practice enhances conceptual understanding [49], [50], [51]. Similarly, in diagrammatic reasoning, an intermediate stage involves creating drawings rather than

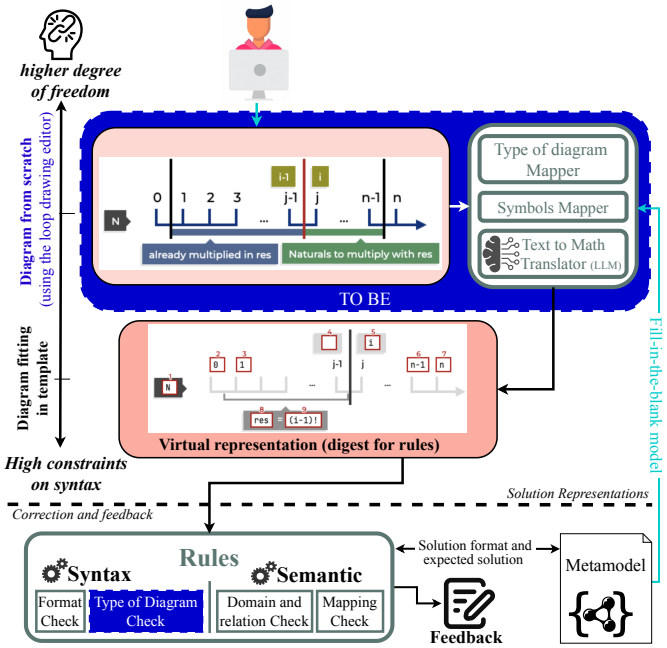


Figure 10. Integrating LLMs in the flow of CAFÉ 2.0.

writing text. This approach leverages the benefits of WTL pedagogy while addressing its primary limitation: the reliance on manual corrections by instructors. From a technical perspective, implementing this feature requires using LLM exclusively to translate free-text fields, as illustrated through Fig. 1.

Importantly, our goal is not to use LLMs to directly generate feedback based on students answers. Other studies [51], [52] highlighted the challenges of this approach. Primarily, feedback generated by LLMs may lack consistency and precision due to the probabilistic nature of these models and potential gaps in domain-specific fine-tuning. LLMs can identify general patterns in responses. However, ensuring consistent and accurate feedback across diverse inputs requires extensive training with a large, high-quality dataset. Additionally, LLMs typically function as "black boxes", making their internal processes difficult to interpret, even when their configuration and tuning are well understood. This lack of transparency can pose issues if students or instructors question the validity of feedback, as it becomes challenging to explain the reasoning behind the generated responses.

The hybrid solution we propose, where an LLM is used to translate free-text responses into structured data processed by a rule checker, provides substantial control over the feedback's content while preserving the flexibility offered by an LLM.

As for the implementation strategy, previous research [52], [53] has identified models such as *SBERT* [54] and the *Universal Sentence Encoder* [55] as effective means for deriving semantically meaningful sentence embeddings from student responses. These models leverage transformer-based architectures that map input sentences into vector spaces, maintaining semantic relationships where similar sentences are represented closer together. By applying these embeddings, we can accurately extract key concepts and semantic structures from student answers, which can be systematically fed into a rule-checking system. This ensures that feedback is not only aligned with expected responses but also controlled with consistent educational input.

VIII. CONCLUSION

To conclude, CAFÉ 2.0 is an interdisciplinary AF tool designed to promote Diagrammatic Reasoning (DR) skills. Its implementation strikes a balance between constraining the syntax of students' solutions (to enable AF) and allowing sufficient freedom for them to develop their DR abilities. To achieve this, when solving a problem, CAFÉ 2.0 presents students with a fill-in-the-blank diagram that includes two types of input fields: *free boxes*, where students enter expressions from a specific domain that satisfy defined relationships, and *constrained boxes*, where students drag and drop predefined symbols. The objective is to prepare students for the exam, where they must create graphical representations of their solutions from scratch (i.e., by hand on paper). To evaluate whether this balance effectively prepares students for the exam, we compared student performance in online homework and corresponding exam questions. Results indicate that while some students perform well in online homework, they face challenges under the exam setting. This highlights the need to better align online homework with exam conditions. To address this, the paper proposes a hybrid approach that integrates predefined rules with LLMs to create more flexible diagrammatic reasoning templates in homework assignments.

SOFTWARE ARTEFACT

CAFÉ 2.0 is written in Python 3. It requires the Pandas library for working properly. CAFÉ 2.0 source code is available at this URL: https://gitlab.uliege.be/cse.

ACKNOWLEDGMENTS

This work is supported by the CyberExcellence project funded by the Walloon Region, under number 2110186.

APPENDIX A
EXAMPLE OF MORE COMPLEX CHECKING RULES

Fig.11 illustrates a fill-in-the-blank diagram containing two types of boxes. *Free boxes* (in red) expect expressions, allowing students the freedom to define their own variables and apply operations to them. *Constrained boxes* (in green) expect a symbol chosen from a predefined list, similar to all boxes in Physics, as shown in Fig.4.

In the configuration file, each box is linked to a semantic defined by typical errors (as in the Physics example). For the checks, *free boxes* are mapped to a domain or a dependency relationship with other boxes, while *constrained boxes* are mapped to a symbol ID. For instance, box E should contain the expression in box D plus 1, and box K should hold the 48th symbol in the list (i.e., "to multiply"). In this example, the student made three mistakes in boxes B, G, and I. In box

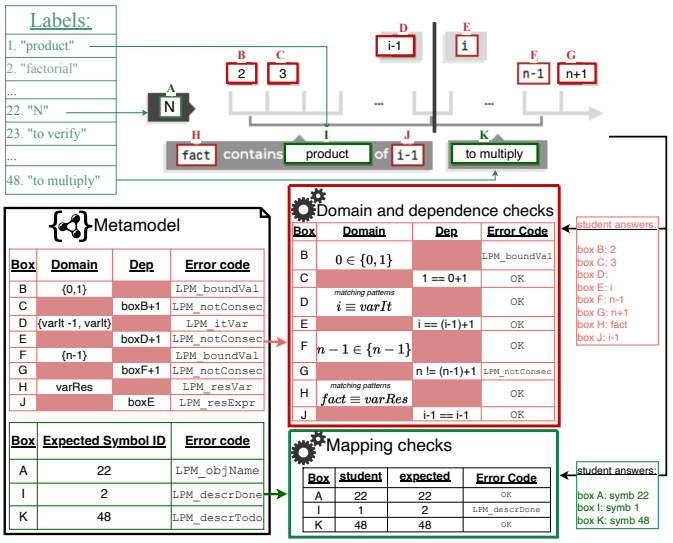


Figure 11. Example of fill-in-the-blank diagram, metamodel and checking rules in CS1. Syntactic checks on *free boxes* are omitted for figure clarity.

B, the student entered 2, while the expected values were 0 or 1, triggering the error code `LPM_boundVal`. Similarly, in box G, the expected content should match the content of box F minus 1 (i.e., `n`), which is not the case. Lastly, in box I, the expected content was the second symbol ("factorial"), but the student selected the first one ("product").

Appendix B
Example of Typical Errors

Table III presents examples of typical errors included in our error library. The first four typical errors are syntactic ones. They appear only during summative assessments, where students must design their solutions from scratch. The next eight typical errors are semantic. For instance, as indicated by its text description, the error code `F_normNotPerp` occurs when students do not realize that the normal force is applied by the plane on the object, which is manifested by the normal force not being perpendicular to the plane.

References

[1] S. Sharmin, D. Zingaro, and C. Brett, "Weekly open-ended exercises and student motivation in CS1," in *Proc. Koli Calling International Conference on Computing Education Research*, November 2020.

[2] C. Burgers, A. Eden, M. D. van Engelenburg, and S. Buningh, "How feedback boosts motivation and play in a brain-training game," *Computers in Human Behavior*, vol. 48, pp. 94–103, July 2015.

[3] S. Buckingham Shum, L. Lim, D. Boud, M. Bearman, and P. Dawson, "A comparative analysis of the skilled use of automated feedback tools through the lens of teacher feedback literacy," *International Journal of Educational Technology in Higher Education*, vol. 20, no. 40, July 2023.

[4] G. Brieven, L. Malcev, and B. Donnet, "Practicing abstraction skills through diagrammatic reasoning over CAFÉ 2.0," in *Proc. IEEE Global Engineering Education Conference (EDUCON)*, 2024.

[5] M. Anderson, B. Meyer, and P. Olivier, Eds., *Diagrammatic Representation and Reasoning*. Springer London, 2002.

[6] L. Mcgarvey, L. Luo, and Z. Hawes, "Spatial skills framework for young engineers," in *Early Engineering Learning*, L. English and T. Moore, Eds. Springer, 2018, pp. 53–81.

[7] J. Parkinson and Q. Cutts, "Investigating the relationship between spatial skills and computer science," in *Proc. ACM Conference on International Computing Education Research (ICER)*, August 2018.

[8] S. Naujokat, M. Lybecait, D. Kopetzki, and B. Steffen, "CINCO: a simplicity-driven approach to full generation of domain-specific graphical modeling tools," *International Journal on Software Tools for Technology Transfer*, vol. 20, no. 3, pp. 327–354, June 2018.

[9] Y. Cherenkova, D. Zingaro, and A. Petersen, "Identifying challenging CS1 concepts in a large problem dataset," in *Proc. ACM Technical Symposium on Computer Science Education (SIGCSE)*, March 2014.

[10] C. A. R. Hoare, "An axiomatic basis for computer programming," *Communications of the ACM*, vol. 12, no. 10, pp. 576–580, October 1969.

[11] O. Astrachan, "Pictures as invariants," in *Proc. ACM Technical Symposium on Computer Science Education (SIGCSE)*, March 1991.

[12] G. Brieven, S. Liénardy, L. Malcev, and B. Donnet, "Graphical loop invariant based programming," in *Proc. Formal Method Teaching Workshop (FMTea)*, March 2023.

[13] T. Huynh, A. T. Alessandrini, L. C. Bauman, O. Sorby, and A. D. Robertson, "Drawing on force ideas for kinematic reasoning in introductory physics," in *Proc. Physics Education Research Conference (PER)*, July 2023.

[14] V. Giardino, "Diagrammatic reasoning in mathematics," in *Springer Handbook of Model-Based Science*, L. Magnani and T. Bertolotti, Eds. Springer, 2017, pp. 499–522.

[15] S. Omar, M. Arshad, M. S. Rosli, and N. Shukor, "Chemistry modelling skills: Students' understanding on transferring simple molecule to model drawing," *Advanced Science Letters*, vol. 23, no. 9, pp. 8259–8263, September 2017.

[16] R. Kadar, N. Wahab, J. Othman, M. Shamsuddin, and S. Mahlan, "A study of difficulties in teaching and learning programming: A systematic literature review," *International Journal of Academic Research in Progressive Education and Development*, vol. 10, no. 3, pp. 591–605, August 2021.

[17] D. Zehetmeier, A. Brüggemann-Klein, A. Böttcher, and V. Thurner, "A concept for interventions that address typical error classes in programming education," in *Proc. IEEE Global Engineering Education Conference (EDUCON)*, April 2016.

[18] V. Almstrum, P. Henderson, V. Harvey, C. Heeren, W. Marion, C. Riedesel, L.-K. Soh, and A. Tew, "Concept inventories in computer science for the topic discrete mathematics," in *Working Group Reports on ITiCSE on Innovation and Technology in Computer Science Education (ITiCSE-WGR)*, June 2006.

[19] L. Jell, C. List, and M. Kipp, "Towards automated interactive tutoring – focussing on misconceptions and adaptive level-specific feedback," in *Proc. European Conference on Software Engineering Education (ECSEE)*, June 2023.

[20] G. Brieven, L. Leduc, and B. Donnet, "Collaborative design and build activity in a CS1 course: A practical experience report," in *Proc. International Conference on Higher Education Advances (HEAd)*, June 2022.

[21] R. Hake, "Interactive-engagement versus traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses," *American Journal of Physics (AMER J PHYS)*, vol. 66, pp. 64–74, January 1998.

[22] K. Sambell, L. McDowell, and C. Montgomery, *Assessment for Learning in Higher Education*. Routledge, 2012.

[23] K. Malinka, M. Peresini, A. Firc, O. Hujnak, and F. Janus, "On the educational impact of ChatGPT: Is artificial intelligence ready to obtain a university degree?" in *Proc. Conference on Innovation and Technology in Computer Science Education (ITiCSE)*, June 2023.

[24] D. Pawelczak, "Effects of plagiarims in introductory programming courses on the learning outcomes," in *Proc. International Conference on Higher Education Advances (HEAd)*, June 2019.

[25] S. Schleimer, D. S. Wilkerson, and A. Aiken, "Winnowing: Local algorithms for document fingerprinting," in *Proc. ACM International Conference on Management of Data (SIGMOD)*, June 2003, see https://theory.standford.edu/~aiken/moss.

[26] E. J. Morris, "Academic integrity matters: Five considerations for addressing contract cheating," *International Journal for Educational Integrity*, vol. 14, no. 15, pp. 1–12, December 2018.

[27] A. Gagatsis, "Compartmentalization in learning," in *Encyclopedia of the Sciences of Learning*, N. M. Seel, Ed. Springer, 2012, pp. 665—-668.

Table III
A TAXONOMY OF TYPICAL ERRORS PROBED BY REVIEWING PAST STATEMENTS ANSWERS.

Error Code	Course	Topic	Nature	Feedback Message
LPM_incorObj	CS1	Loop Modeling	Syntax	The object you have represented doesn't match with any existing one.
LPM_posDemarc	CS1	Loop Modeling	Syntax	Your demarcation line should be before or after a graduation.
F_noOrient	Physics	Kinematics	Syntax	Any force you represent should be an oriented arrow to see in which direction the force is pulling.
F_noName	Physics	Kinematics	Syntax	Any force you represent should have a name so that it can be referred in the equations.
LPM_boundVal	CS1	Loop Modeling	Semantic	The expressions of the boundaries are incorrect with respect to the problem statement.
LPM_notConsec	CS1	Loop Modeling	Semantic	The expression in this box should be consecutive to the one in box {}.
LPM_descrDone	CS1	Loop Modeling	Semantic	The description of the solution state is incorrect.
F_gravNotDown	Physics	Kinematics	Semantic	The gravitational force is always vertical and oriented down, to the Earth.
F_normNotPerp	Physics	Kinematics	Semantic	The normal force is always perpendicular to the plan the object lies on.
F_miss	Physics	Kinematics	Semantic	A force is applied in this direction.
F_fPar	Physics	Kinematics	Semantic	The friction force is always parallel to the plan the object lies on.
F_noEngine	Physics	Kinematics	Semantic	The engine is off, which means that no driving force applies in this situation.

[28] S. Steinert, L. Krupp, K. Avila, A. Janssen, V. Ruf, D. Dzsotjan, C. Schryver, J. Karolus, S. Ruzika, K. Joisten, P. Lukowicz, J. Kuhn, N. Wehn, and S. Küchemann, "Lessons learned from designing an open-source automated feedback system for STEM education," *Education and Information Technologies*, vol. 29, no. 13, pp. 1–42, September 2024.

[29] P. Osztián, Z. Katai, and O. Erika, "On the computational thinking and diagrammatic reasoning of first-year computer science and engineering students," *Frontiers in Education*, vol. 7, pp. 1–9, September 2022.

[30] P. Klein, A. Müller, and J. Kuhn, "Assessment of representational competence in kinematics," *Physical Review Physics Education Research*, vol. 13, p. 010132, July 2017.

[31] Brillant, "Learning by doing," [Last Access: November 12th, 2024]. [Online]. Available: https://brilliant.org/

[32] G. Deeva, D. Bogdanova, E. Serral, M. Snoeck, and J. De Weerdt, "A review of automated feedback systems for learners: Classification framework, challenges and opportunities," *Computers & Education*, vol. 162, p. 104094, 2021.

[33] N. Parlante, "Codingbat: Code practice," 2011, [Last Access: November 13th, 2024]. [Online]. Available: https://codingbat.com

[34] Pearson, "My lab programming," [Last Access: November 13th, 2024]. [Online]. Available: https://www.pearsonmylabandmastering.com/northamerica/myprogramminglab/

[35] R. Lobb and J. Harlow, "Coderunner: a tool for assessing computer programming skills," *ACM Inroads*, vol. 7, no. 1, pp. 47–51, March 2016.

[36] M. Messer, N. C. C. Brown, M. Kölling, and M. Shi, "Automated grading and feedback tools for programming education: A systematic review," vol. 24, no. 1, February 2024.

[37] A. Kulkarni, S. Endait, R. Ghatage, R. Patil, and G. Kale, "Automated answer and diagram scoring in the STEM domain: A literature review," in *Proc. International Conference for Emerging Technology (INCET)*, May 2024.

[38] "Khan Academy," [Last Access: November 12th, 2024]. [Online]. Available: https://blog.khanacademy.org/

[39] Z. Pardos, M. Tang, I. Anastasopoulos, S. Sheel, and E. Zhang, "OATutor: An open-source adaptive tutoring system and curated content library for learning sciences research," in *Proc. Conference on Human Factors in Computing Systems (CHI)*, April 2023.

[40] M. Orthaber, D. Stütz, T. Antretter, and M. Ebner, "Concepts for e-assessments in STEM on the example of engineering mechanics," *International Journal of Emerging Technologies in Learning (iJET)*, vol. 15, no. 12, pp. 136–152, June 2020.

[41] A. Heck, "Using SOWISO to realize interactive mathematical documents for learning, practising, and assessing mathematics," *MSOR Connections*, vol. 15, no. 2, pp. 6–16, 2017, [Last Accessed: December 15th, 2023]. [Online]. Available: https://cloud.sowiso.nl

[42] S. Chee, S. Hoong, and L. Yeng Seng, "Innovating engineering education with blended learning and remote laboratories," in *Proc. IEEE Global Engineering Education Conference (EDUCON)*, May 2024.

[43] A. Rolwes, P. Stellbauer, U. Lungershausen, D. Cubela, K. Böhm, and P. Neis, "MAATSE: Prototyping and evaluating an open and modular e-assessment tool for STEM education," in *Proc. IEEE German Education Conference (GECon)*, August 2023.

[44] Aktiv, "Discover the formula for student success," [Last Access: November 12th, 2024]. [Online]. Available: https://www.aktiv.com

[45] R. Schechter, R. Gross, and J. Cai, "Exploring nationwide student engagement and performance in virtual lab simulations with labster," in *Proc. EdMedia + Innovate Learning*, July 2024.

[46] "Aleks," [Last Access: November 12th, 2024]. [Online]. Available: https://www.aleks.com

[47] S. Knight, A. Shibani, S. Abel, A. Gibson, P. Ryan, N. Sutton, R. Wight, C. Lucas, A. Sando, K. Kotto, M. Liu, R. V. Mogarkar, and S. B. Shum, "AcaWriter: A learning analytics tool for formative feedback on academic writing," *Journal of Writing Research*, vol. 12, no. 1, pp. 141–186, June 2020.

[48] T. Duong and C. Meng, "Automatic grading of short answers using large language models in software engineering courses," in *Proc. IEEE Global Engineering Education Conference (EDUCON)*, May 2024.

[49] P. Anderson, C. Anson, R. Gonyea, and C. Paine, "The contributions of writing to learning and development: Results from a large-scale multi-institutional study," *Research in the Teaching of English*, vol. 50, no. 2, pp. 199–235, November 2015.

[50] A. R. Gere, N. Limlamai, E. Wilson, K. MacDougall Saylor, and R. Pugh, "Writing and conceptual learning in science: An analysis of assignments," *Written Communication*, vol. 36, no. 1, pp. 99–135, January 2019.

[51] F. Watts, A. Dood, and G. Shultz, "Automated, content-focused feedback for a writing-to-learn assignment in an undergraduate organic chemistry course," in *Proc. International Learning Analytics and Knowledge Conference (LAK)*, March 2023.

[52] A. Botelho, S. Baral, J. Erickson, P. Benachamardi, and N. Heffernan, "Leveraging natural language processing to support automated assessment and feedback for student open responses in mathematics," *Journal of Computer Assisted Learning*, vol. 39, no. 3, pp. 823–840, June 2023.

[53] I. Ndukwe, A. Chukwudi, L. Nkomo, and B. Daniel, *Automatic Grading System Using Sentence-BERT Network*. Springer, June 2020, pp. 224–227.

[54] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proc. of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, January 2019, pp. 3973–3983.

[55] D. Cer, Y. Yang, S. Kong, N. Hua, N. Limtiaco, R. St. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, Y.-H. Sung, B. Strope, and R. Kurzweil, "Universal sentence encoder," 2018. [Online]. Available: https://arxiv.org/abs/1803.11175