JOETSR-International Journal of Emerging Trends in Scientific Research
Volume 4, Issue 1 | January-2026
e-ISSN: 2997-8718 Print ISSN: 2997-870X

Unifying Pedagogical Approaches to Strengthen Abstract Thinking Across Mathematics and Computer Science Education

GSDV Prasad Sharma
Faculty of Computer Science Keshav Memorial Engg College
E mail: gpsnow86@gmail.com
Received: 01-11-2025; Revised: 24-11-2025; Accepted: 17-12-2025; Published: 10-01-2026

Abstract
A common concept between mathematics and computer science is abstraction, which allows learners to deal with complex phenomena, learn to generalize patterns and represent systems in an effective way. In spite of its significance, abstraction is a pedagogical difficulty because it is inherently abstract. The given paper examines the integrated teaching methods that close the gap between mathematical and computational abstraction. It presents a cross- disciplinary framework and relies on cognitive science, educational theory, and classroom practice to put forth clear and visual reasoning, analogy-, layering and problem-based learning, which facilitate conceptual clarity. The target is to develop abstract thinking ability of the students intuitively and transferrable between the realms.

Keywords: ***Abstraction, Mathematics Education, Computer Science Education, Conceptual Understanding, Cross-Disciplinary Teaching, Pedagogical Strategies, Computational Thinking, Problem-Based Learning, Cognitive Development, Instructional Design.***

1.Introduction
Abstraction has been the intellectual strength of computer science and matters of mathematics, which has given us an effective way to streamline complication, issue uniformisation, and a chance to think systematically. The effective computational thinking, as Keith Devlin aptly puts it, is impossible without the abstraction, which is part of creating, manipulating, and examining software systems. Nevertheless, although it is a fundamental concept, abstraction has proved one of the most elusive concepts as far as students are concerned. They tend to be drawn to concrete products of such as executable code or numeric output, and stay away from the abstractions that define good software design or formal logic. It is an educational divide which has created a demand of combined pedagogy which demystifies abstraction, and makes it seem approachable, learnable(1).

They adopt abstraction in different ways in both fields but they have one thing in common: that is, they aim at doing away with irrelevant specificities so as to bring to light key structures. Rather, mathematics does the same by using axiomatic systems, symbol manipulation, and generalization; computer science does the same by means of modeling languages, hierarchies of classes and algorithm Hennessy and Ely design. Traditionally these two approaches have been taught separately: mathematics through symbolic paper and pencil exercises, and computer science in coding labs. This division however does not highlight the strong interconnectedness between the two which leads to the tendency of the students to compartmentalize their education thereby undermining their ability to transfer abstract reasoning across disciplines. As a result, students can prove successful at solving procedural problems or at successfully writing functional code but fail to achieve success at the higher levels of design and conceptual reasoning.

There also exists this split which requires educational re-conceptualization, which is the one that combines teaching mathematics and computer science because both have a common focus on abstraction. One way that seems productive is to incorporate symbolic modeling tools into undergraduate study in mathematics to offer a continuum between symbolic mathematics and applied computing. More serious goals also lead to the Formal modeling languages such as Alloy which is a lightweight and expressive language that is used to specify abstract relations and constraints. The main power of Alloy is that its accessibility: it codifies ideas in terms of simple set-theoretic and relational logic, understood by any student who has been exposed to discrete mathematics. Meanwhile, it facilitates the tasks of practical modeling, corresponding to UML or object-oriented design, thus, intuitively associating mathematical concepts with software engineering concepts.

The key feature of Alloy in the educational context, which makes it especially interesting, is that it can be used both to develop a visual representation of abstract models and to emulate logical constraints. Students are able to build models and immediately be shown counterexamples or examples which either confirm or disapprove their

hypothesis through machines such as the Alloy Analyzer. This constant back and forth also encourages exploratory learning, so that students can learn iteratively and actively as opposed to passively. This kind of interaction does not only enforce conceptual precision but also inculcates critical thinking and intellectual possession over learning results. In addition, visual part of Alloy makes abstract structures concrete, rendering such concepts as inheritance, transitivity, injectivity, or composition of relations easier to grasp and less threatening.

An important pedagogical contribution provided by this approach is the adoption of the Action Process Object (APO) learning model that follows the internalization process of learners on mathematical concepts. The first type of images involves concrete activity around objects in students, that is mediated with the help of tools and systematic exercises(2). These performances transform into abstract levels of thought when the learner starts thinking conceptualizing the operations. Ultimately such processes are reified into objects, mental entities manipulable, reasoned about, and assembled into larger structures. With the match between this cognitive trajectory and Alloy-based exercises, educators can scaffold abstraction in a manner that reflects that of the learner. As an example, having predetermined examples, one can simulate relational mappings, analyze counter examples, and broadly construct their own models are stepping along toward the further understanding and solidifying their knowledge.

In practice a modeling-first practice can be incorporated in the early coursework in mathematics and computing directly. Instead of pushing formal techniques off to a later phase with more complicated material, rotated courses in logic, sets and discrete structures could include Alloy-based projects that do abstraction-level, as well as modelling. As an example, relations and functions (normally a set of taught symbolic operations) can be illustrated via Alloy diagrams and assigned an interpretation to real word systems, e.g. digital libraries, social networks, or access control systems. This way the students learn to be precise in mathematics and this is combined with the realization that this is applicable in software designing.

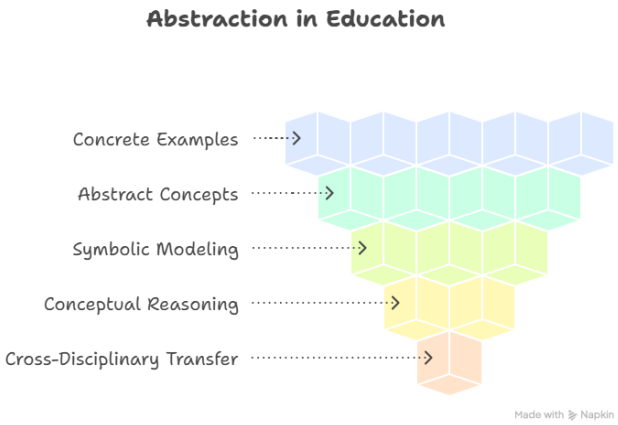

FIGURE 1 Abstraction in Education

The advantages transfer to fundamental programming aspects of computer science like object-oriented programming, design of database schema, and system specification. Such concepts as inheritance, composition and polymorphism, which are commonly presented using syntax-filled coding examples, can be first presented using Alloy, where they are simplified to structural basics. The basic concept students will have learned is that, subclass is a subset, or that set-based relation between parts and wholes are achieved because of composite. Early insights make learning to code less painful and challenging; they serve to ensure that programming is not embedded in trial-and-error explorations but based on an understanding of concepts.

No less important is the role that Alloy plays in introducing the students to theory behind computing without the demands of learning complex skills in proving theorems. With Alloy, they are able to model specifications, to put constraints on what the system can be, to run finite state machines, and to investigate the correctness of algorithms by symbolic execution(3). This offers a declarative step into formal methods, introducing the students to the idea that logic and rigor is not just an academic abstract concept but a useful resource in building software. In that way, learners comprehend the field of computer science as a design practice students learn to value that in the first place: structure, correctness, generality instead of the immediate functionality.

Active learning using formal tools is challenging though it offers several benefits. Possession of alien approaches might first provoke resistance in the students who find them more taxing than using conventional lectures or coding

exercises. However, as research and classroom practice has revealed, the awkwardness most often is replaced by greater involvement and stronger comprehension with the time flow. It lies in the continued support of the instructor and cross-disciplinary between faculty as well as the curriculum that supports modeling exercises with the mathematical theory as well as the software development.

2.Related Works

The implementation of computer-based tools in math and computer science education has developed increased interest among instructors who want to acquire higher cognitive thinking and abstract thinking in learning among learners. A vast number of investigations and projects have undertaken during the last twenty years the study of technological platforms, such as symbolic algebra systems to modeling languages, as a platform facilitating an active learning and a conceptual understanding. This chapter draws a review of some of the pioneering contributions related to this field and how our proposal adds and differs with the previous work especially in terms of the focus on lightweight formal approaches and cross-functional approach to teaching.

The application of computer aids in math teaching has been especially discussed. Computer Algebra Systems (CAS), Dynamic Geometry Software (DGS), and Spreadsheet Programs (SP) are platforms, which have proven efficient in encouraging more involvement of students and beyond concrete learning. As an illustration Gregory (2004) highlighted the possibilities of employing the web-based technologies in making computing students interested in working with mathematics that is usually not relevant to them or excessive in its theoretical nature(4). Such tools give the students the ability to manipulate mathematical objects interactively and they give reinforcement through experimentation. In a similar way, Siller and Greefrath (2009) investigated the application of modeling environment during the classroom activities, emphasising the understanding of importance of the application of mathematics to situations that are realistic. These systems, however, although they facilitate mathematical modeling, may be restricted to numerical or geometric contexts, and are not easily mapped to the computational abstractions of current computer science courses (object-oriented modeling, system specification, logic-based design.)

Our method best falls in line with the constructivist pedagogical desire by Fenton and Dubinsky (1996) who in coming-up with the Action Process Object (APO) model thought of a means through which learners develop mental images regarding mathematic ideas. Students in their work are moved through concrete manipulations to abstract reasoning and end in internalizing ideas as mathematical objects. They support this cognitive trajectory in their curriculum through ISETL a programming language that teaches discrete mathematics. Even though we too believe in the APO model, we differ in approach both in its reach and in its application. First, to encourage abstraction to be more disciplined and axiomatic, we will exploit Alloy as our formal specification verb rather than a conventional programming language. Second, we link our courses cross-disciplinarily between the programming courses and the mathematics courses, such that abstraction conceiving may proceed in parallel across the domains.

A number of scholars have also looked into the introduction of object-oriented (OO) design which takes place through modeling tools. As Hadar and Hadar (2007) pointed out, UML has great advantages in teaching OO concepts especially because it imposes abstraction barriers and displays images of class structures. The same set of features - abstraction, encapsulation and visual modeling can be found in Alloy where it adds additional insight to the process because it introduces formal constraints of the model, as well as allows students to communicate with models using logic-based commands. Bennedsen and Caspersen (2004) also suggested a model-first programmer approach in which UML is used to provide a broad picture of a program before a student goes anywhere near the implementation. We go forward with this concept by not only visualization, but also giving students a mathematical view of object-oriented concepts in order to have a more solid and transferable concept.

The instructional worth of formal approaches has been admitted in a number of institutional enterprises. As an example, Lutz (2006) and Naveda et al. (2009) report their experience at Rochester Institute of Technology (RIT) where the curriculum was reorganized to incorporate both formal and informal modeling techniques in all of the software engineering classes. Discrete mathematics gets an early introduction and then formal techniques like Z, VDM, and Alloy. These courses had sounds similar to our conviction that formal reasoning is not to be deferred until an upper-level course but must be incorporated at the very beginning to influence the mental models that students develop about computation and design.

The other uses of Alloy to teach students add to the importance of this tool as an educational tool. An example is the description of the usage of Alloy in a module of stand-alone software design (Boyatt and Sinclair, 2008). They show the abilities of Alloy to experiment and visualize, through which the students can iterate to improve their knowledge and get knowledge through modeling. Nonetheless, the barrier with their approach is that the usage of Alloy is limited to one course thereby limiting its capacity as a long-term scaffolding tool throughout the curriculum. Contrarily, the approach that we propose treats Alloy as a pedagogical continuum--starting with the presentation of introductory set-theoretic relations, and going up to modeling complex software systems. Constancy of use strengthens the abstract thinking and avoids the tendency of students to consider formal practices as distant and irrelevant.

A less fragmented approach to the combination of mathematics and software engineering has even been prepared by some teachers in a bid to minimize subject isolation of mathematics and software engineering. Sekerinski (2009) suggested that mathematics was the best idea to be applied in the entire concepts of software design. His framework focuses on formal methods of the program specification, proofs of correctness, and verification of algorithms by using formal approaches, such as Weakest Preconditions Calculus. His approach enhances theory rigor of computing education though it is more proof intensive and correctness driven in nature. Conversely, we are interested in mathematical modeling, that enables design thinking and abstraction without formal verification per se.

As a conclusion, Cohoon and Knight (2006), university of Virginia investigated the role that software engineering problems can play in inciting the introduction of discrete mathematical concepts. It is a problem based approach: the students are exposed to a software modeling problem which inherently demands either logical or set-theoretic reasoning. Although such trend sounds encouraging, at our levels modeling activities are implemented more systematically because they are incorporated directly into mathematics and computing labs, and students can move between abstract and implementation levels in a single mental domain.

3.Leveraging Alloy in Learning Environments

The implementation of computer-based tools in math and computer science education has developed increased interest among instructors who want to acquire higher cognitive thinking and abstract thinking in learning among learners. A vast number of investigations and projects have undertaken during the last twenty years the study of technological platforms, such as symbolic algebra systems to modeling languages, as a platform facilitating an active learning and a conceptual understanding. This chapter draws a review of some of the pioneering contributions related to this field and how our proposal adds and differs with the previous work especially in terms of the focus on lightweight formal approaches and cross-functional approach to teaching(5).

- **Integration of Computer Tools:**
  - Computer Algebra Systems (CAS), Dynamic Geometry Software (DGS), and Spreadsheet Programs (SP) are widely used tools that enhance students' engagement in mathematics learning.
  - These technologies support learning that goes beyond concrete computation and fosters deeper conceptual understanding.
- **Enhanced Student Involvement:**
  - Such systems promote active participation and interactive exploration by allowing students to manipulate mathematical objects directly.
  - Learning through experimentation and immediate feedback reinforces comprehension.
- **Gregory (2004) Study:**
  - Highlighted the potential of web-based technologies in motivating computing students.
  - These tools make mathematics more accessible and relevant, especially for students who find traditional, theory-heavy approaches unengaging.
- **Siller and Greefrath (2009) Study:**
  - Investigated the use of modeling environments in classroom activities.
  - Emphasized developing students' understanding of how mathematics applies to real-world contexts.

Limitations of Current Systems:

Despite their benefits, these tools are mostly effective in numerical or geometric modeling.

They are less suited for abstract, computational contexts such as object-oriented modeling, system specification, or logic-based design in computer science courses.

Our method best falls in line with the constructivist pedagogical desire by Fenton and Dubinsky (1996) who in coming-up with the Action Process Object (APO) model thought of a means through which learners develop mental images regarding mathematic ideas. Students in their work are moved through concrete manipulations to abstract reasoning and end in internalizing ideas as mathematical objects. They support this cognitive trajectory in their curriculum through ISETL a programming language that teaches discrete mathematics. Even though we too believe in the APO model, we differ in approach both in its reach and in its application. First, to encourage abstraction to be more disciplined and axiomatic, we will exploit Alloy as our formal specification verb rather than a conventional programming language. Second, we link our courses cross-disciplinarily between the programming courses and the mathematics courses, such that abstraction conceiving may proceed in parallel across the domains.

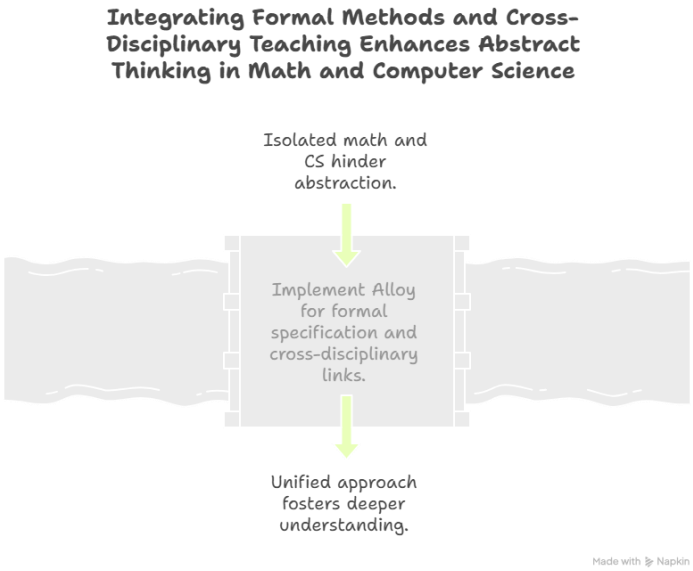

FIGURE 2 Integrating Formal Methods and Cross-Disciplinary

A number of scholars have also looked into the introduction of object-oriented (OO) design which takes place through modeling tools. As Hadar and Hadar (2007) pointed out, UML has great advantages in teaching OO concepts especially because it imposes abstraction barriers and displays images of class structures. The same set of features - abstraction, encapsulation and visual modeling can be found in Alloy where it adds additional insight to the process because it introduces formal constraints of the model, as well as allows students to communicate with models using logic-based commands. Bennedsen and Caspersen (2004) also suggested a model-first programmer approach in which UML is used to provide a broad picture of a program before a student goes anywhere near the implementation(6). We go forward with this concept by not only visualization, but also giving students a mathematical view of object-oriented concepts in order to have a more solid and transferable concept.

The instructional worth of formal approaches has been admitted in a number of institutional enterprises. As an example, Lutz (2006) and Naveda et al. (2009) report their experience at Rochester Institute of Technology (RIT) where the curriculum was reorganized to incorporate both formal and informal modeling techniques in all of the software engineering classes. Discrete mathematics gets an early introduction and then formal techniques like Z, VDM, and Alloy. These courses had sounds similar to our conviction that formal reasoning is not to be deferred until an upper-level course but must be incorporated at the very beginning to influence the mental models that students develop about computation and design.

The other uses of Alloy to teach students add to the importance of this tool as an educational tool. An example is the description of the usage of Alloy in a module of stand-alone software design (Boyatt and Sinclair, 2008). They show the abilities of Alloy to experiment and visualize, through which the students can iterate to improve their knowledge and get knowledge through modeling. Nonetheless, the barrier with their approach is that the usage of Alloy is limited to one course thereby limiting its capacity as a long-term scaffolding tool throughout the

curriculum. Contrarily, the approach that we propose treats Alloy as a pedagogical continuum--starting with the presentation of introductory set-theoretic relations, and going up to modeling complex software systems. Constancy of use strengthens the abstract thinking and avoids the tendency of students to consider formal practices as distant and irrelevant(7).

A less fragmented approach to the combination of mathematics and software engineering has even been prepared by some teachers in a bid to minimize subject isolation of mathematics and software engineering. Sekerinski (2009) suggested that mathematics was the best idea to be applied in the entire concepts of software design. His framework focuses on formal methods of the program specification, proofs of correctness, and verification of algorithms by using formal approaches, such as Weakest Preconditions Calculus. His approach enhances theory rigor of computing education though it is more proof intensive and correctness driven in nature. Conversely, we are interested in mathematical modeling, that enables design thinking and abstraction without formal verification per se.

As a conclusion, Cohoon and Knight (2006), university of Virginia investigated the role that software engineering problems can play in inciting the introduction of discrete mathematical concepts. It is a problem based approach: the students are exposed to a software modeling problem which inherently demands either logical or set-theoretic reasoning. Although such trend sounds encouraging, at our levels modeling activities are implemented more systematically because they are incorporated directly into mathematics and computing labs, and students can move between abstract and implementation levels in a single mental domain.

4.Constructing Mathematical Meaning Through Interactive Modeling

The necessity to provide efficient measures to explain mathematical principles in STEM studies, especially abstract ones, has been gaining ground in our modern society. Although conventional pedagogical practices are highly based on symbol manipulation and rote learning and solving of problems, these pedagogical methods fail most times to develop a form of conceptual knowledge as a base of higher level reasoning in mathematics and computing. In this light, interactive modeling environments provide a way out. Such tools facilitate the involvement of the learners in mathematics in a constructivistic action-oriented way promoting the learning beyond filling in the blanks of the formulas into the active role of meaning building. Alloy is one such platform, but it is the subject of remarkable power, of a multi-leveled, graphic, formal treatment of the abstract objects sets, relations, and logical inference.

The main idea expressed in this model-based approach to pedagogy is that mathematical knowledge can be constructed most fruitfully through action. It is not only important that learners can observe mathematical structures, but they should also have to construct, experiment and reconstruct them. Alloy supports this in a technique that gives a formal language to specify sets and constraints, and a visualizer that displays instances of the model and point to counterexamples. These qualities allow a strongly interactive process of learning where students can prove, experiment, and revise their knowledge after correction. Such interactivity is very coherent with constructivism theories of learning, where knowledge is not transferred but is actively created on the basis of experience.

In order to organize this kind of construction of knowledge, the teachers will be able to use the Action Process Object (APO) model, which can be defined as the cognitive framework created within the purpose to explain the mental development of mathematical concepts(8). Based on this model learners go through three processes. They have to act, and there is a characteristic they act on, which can be called actions or algorithmic procedures or steps on objects. As learners get into practice such actions turn into processes and learners can manipulate such concepts in their mind without requiring physical implementation of the action. Later, the processes themselves were captured as objects, entities to be thought of and acted upon, similar in that regard to physical objects. Alloy facilitates this model where the students are moved on the scale between manually manipulating concrete model instances to the more abstract levels of reasoning.

Let us take a classroom example of using visible relations. It has a set-theoretic model built around the thematic use of a fictional but logically dense situation in which the characters are called gunmen, diggers, holes, and treasures. One is asked to define sets like Gunman, Digger and Hole, together with relations like threat, dig, and contain. They build these relationships using Alloy, and test constraints of exclusivity, injectivity, and composition. Such thematic modeling activity automatically turns an abstract term such as binary relation or domain into a concrete object that can be graphically visualized and operated upon. To give an example, the

dig.contain composition can be examined not just algebraically, but also using the graphical output of Alloy, strengthening intuitive sense of chained relations.

During early levels of teaching, learners will be able to utilize preset examples of models to make sure that they concentrate on interpreting as opposed to building. These predetermined templates which come with already established sets and relations assist in scaffolding the action stage of the APO model. Operations like relation composition, projection, inversion etc. are taught to students with check commands usually helping to check properties, or run commands to follow instance behavior. A practice can request them to deduce a composed relation and contrast Alloy counterexamples with their anticipated outcomes. This type of activities encourages reflection and learning by doing the same tasks again, as the students are to reconcile their mental model with the feedback coming out of the system.Higher abstractions are presented in the curriculum as students get more confident. As opposed to interpreting relationships only, they are provoked to conceptualize the relationships. As an illustration, they can be requested to declare a new relation named threatWith that will associate gunmen with other gunmen who have a joint target. They have to create an amalgamation of known relations and use logic to solve this activity. They should also take into account such mathematical properties as reflexivity or symmetry and apply their definitions in the Alloy. Here they are moving out of the process stage onto the object stage of the APO model instead of treating composed relations as objects that are to be transformed and analysed(9).

An additional level of abstraction is reached via meta-modeling tasks, i.e., activities during which students decompose or build models at a higher theoretical level. They can be requested to judge the validity of a model, or to suggest changes to remove inconsistencies, or to demonstrate a general mathematical behaviour both in Alloy and in the traditional style of symbolic proofs. An example is that a classical problem like, whether every symmetric and transitive relation is reflexive is an interactive inquiry. Alloy allows students to specify such a relation, instantiate a counterexample search and read the visual display to reject naive guesses. Then they pass to a paper proof, bridging empirical discovery and formal proving.

Importantly, the reason why Alloy is not a replacement of other mathematical methods is that it adds to them and makes abstract concepts more accessible. It is a conceptual sandbox, as the students are free to test there hypothesis not afraid to be punished, and your get the instant feedback, which is visual. This is a maturity process in mathematics as students are being taught to stop perceiving symbols as hooked arbitrarily character but the transmitter of meanings, which are anchored in clearly defined structures. What is more, working with the same tool to investigate the concepts of both mathematics and software design models students start to realize the unity of the arts. The prerequisites are no longer abstract as the set theory and logic approaches become the computational language of the thought.

Compartmentalization by students so that they perceive Alloy to be a software, whereas mathematics represents a paper-based activity is another pedagogical battle that is constantly fought. In response to that, the instructors should be always focused on emphasizing the linkage of Alloy modeling to mathematical thinking and employ bridge tasks that necessitate students to move modalities. An example could be that the end of a lab session will consist of a direction to demonstrate, in paper, a property they have established using Alloy, or vise versa. Not only will this strengthen the concepts, it will also place the students in training in terms of transferable skills or in other words the capacity to work seamlessly across formal languages, symbolic thinking, and visual instinct(10).

In short, interactive modeling using Alloy is a revolutionary strategy that delivers the concept of teaching mathematical abstraction. It enables the students to build sense in an active way, to see logical frameworks, and to engage in active development of their knowledge. Through its adherence to and harmony with cognitive models such as APO and its complete and smooth integration with domain-theoretical and domain-applied material, Alloy becomes not a means, but a vehicle of pedagogical approach to the acquisition of mathematical understanding and computational facility. The importance of the formal modeling in the context of mathematics education is here to increase, as far as educators attempt to develop the level of the comprehension deeper in the interdisciplinary world.

5.Conclusion

The question of how to teach abstraction in mathematics and computer science has been a classic worry that has been haunting STEM education. Abstraction, which is by far considered one of the most potent intellectual instruments of dealing with complexity and generalising the knowledge, is a mystery to many learners. It is elusive in the sense that it cannot be easily understood particularly to those students who have been socialized to embrace

tangible things like production of procedural solutions, running codes or learn to think in numbers. As it has been argued in this paper, an incremental change is necessary regarding the introduction and development of abstraction; a cross-disciplinary, modeling first, computer supported method that combines mathematical reasoning and computational practice. The focus of this pedagogical innovation is Alloy, a lightweight formal modeling and it allows lecturers to demonstrate an expressive medium that can also be a visual reasoning tool. The designed syntax and the tutor analyzer allow students to work with abstract relationships, formal constraints, counterexamples, and, what is more crucial, visualize logical structures on the spot thanks to Alloy.

Other than being a tool that is incorporated in learning, the introduction of Alloy in the curriculum marks a shift in the paradigm of education. The presented approach suggests to view mathematics and computer science as two disciplines that have a common goal the development of abstract thinking the way it is achieved through playing with formal systems. This combination is practical and theoretical. On the one hand, it encourages mathematical maturity because the students learn to operate on the building blocks used in mathematics in a practical and problem solving context. Conversely, it is good at computational fluency, an actual bridge to software modeling, object-oriented concepts and system specification without the cognitive fatigue of syntax-saturated programming languages. Learners get to realize, through Alloy, that abstraction is not a hindrance to learning but as the means through which mathematical structures and software systems can be comprehended, manipulated and mastered.

One of the most valuable aspects of such a strategy is that it follows the Action Process Object (APO) model in cognitive development. The structure of learning activities sequence, on a path of guided tasks of concrete relational composition to free tasks of generalization and modeling, is reflected in progressive transition to abstract development of processes up to conceptual objects. Indeed, when students define relations in Alloy and write them out and visualize their output and iteratively refine their models, they are going through precisely this developmental process. Initial activities can be related to using constant examples or model scenarios prepared in advance that allow students to complete certain tasks and get immediate responses. As their confidence builds up they switch to developing their own constraints and investigating such theoretical properties as being able to be symmetrical, reflexive and transitive- again reifying the mathematical processes as conceptual tools.

The interesting part in Alloy in relation to any other modeling or programming tool is that it is able to develop active learning without any coding concerns such as complex syntax and the technical experience before the use. Alloy provides a logical, but beginner-friendly set-theoretic vocabulary, in contrast to programming languages which sometimes obscure their meaning with syntactical requirements. Further, its interactive analyzer enhances a learning process that is hypothesis driven and students ask questions, build models and check a logical outcome. This creates intellectual independence and it motivates learners wanting to own their knowledge. The visual result of the tool, diagrams of model instances, graphical displays of relational compositions, or counterexample generation, is a key factor linking more formal approaches to reasoning to more informal intuitive approaches. By so doing, Alloy helps students build two representations of abstract structures, the first one formal and symbolic, the second one visual and experiential.

Moreover, the fact that Alloy is a programming language that can be used as a language to model mathematics as well as a software design language implies that it can bridge the disciplinary gap between mathematical and computer sciences. It aids the formation of concepts, and strengthens logical argumentation in the mathematics classroom and present conjectures to be experimentally verified. It teaches the students the most important concepts of object-oriented programming, such as inheritance, composition and abstraction, in a rigorous, yet approachable form in the context of computer science. Due to the uniformity of language in the two domains, the student can start to see a flow of abstraction, such that the set theoretic logic and relations logically continues to describe practical software systems. In conventional curricula, this continuity is not normally a part, so students find it difficult to relate formal arguments to programming, or to understand how theoretical mathematics supports computational design.

In spite of its merits, there are some difficulties associated with the introduction of this model. The active engagement of students (particularly those who are used to rote learning), has to be met persistently, scaffolded, and driven by the instructor. At the beginning, there is some resistance, since students might find the modeling exercises cognitively heavy or unrelated to the tasks at hand: codings. Yet, along the longitudinal track, the observed trends are that with familiarity with the tool and the conceptual advantages of it, students engage more and learn better. Alloy labs also implies joint curriculum design and joint instruction with participation of a mathematics educator and a computer science educator, and adding this would indeed help improve conceptual coherence and integrity. In this cross-dis-ciplinary collaboration a second educational advantage can become a

model to students the type cross boundary thinking that is becoming an essential requirement in modern problem solving.

The level of this approach is proven by assessment data and observations that take place in the classroom. It is seen that students are able to retain mathematical concepts better, their ability to reason is improved and they are able to model complex systems. They also tell of having more value towards the applicability of abstract mathematics to their computing course work. Although Alloy does not completely remove the kind of tension that students associate with abstraction, it offers them a concrete means of interaction that will de-mystify the process and generate confidence in the long term. The students, who are educated according to such a paradigm, in the long run are much better prepared to move in the theoretical and practical aspects of their academic and professional activity.

Finally, the same study argues in support of a pedagogical approach, whose focus is not on abstraction as a by-product but one on which focus is intentionally put, developing abstraction via modeling activities with the support of computers. The Alloy-based approach provides a scalable cross-disciplinary method innovating the teaching and learning process of abstraction. It makes it possible to develop strong conceptual frameworks (and fields) in mathematics and computer science by rooting less concrete ideas in engaging visual, interactive and applied environments. The outcome is not just better understanding of the material in a discipline, but also acquisition through meta-cognitive ability which enables student to deal spectacularly with abilities of structural reasoning, problem decomposition, and formal modeling many of which are essential in information oriented world today. Education is changing and strategies such as these will prove to be critical in the training of agile minds, people who will be able to move through the complexity of the world by abstraction, formalism and connecting different disciplines.

Acknowledgement: Nil

Conflicts of interest

The authors have no conflicts of interest to declare

References

1. Kaufmann P, Moss J, Osana HP. The role of abstraction in mathematical and computational problem-solving: A review. Educational Psychology Review. 2021;33(2):495–517.
2. Mannila L, Peltomäki M, Salakoski T. What about a simple language? Analyzing the difficulties in learning programming. Computer Science Education. 2006;16(3):211–227.
3. Wing JM. Computational thinking. Communications of the ACM. 2006;49(3):33–35.
4. Armoni M, Gal-Ezer J. Teaching abstraction in computer science to high-school students. ACM Transactions on Computing Education. 2014;14(2):8.
5. Brown N, Sentance S, Crick T, Humphreys S. Restart: The resurgence of computer science in UK schools. ACM Transactions on Computing Education. 2014;14(2):9.
6. Barany A, Tesar M. Bridging disciplines: Mathematics and computer science abstraction in early education. British Journal of Educational Technology. 2020;51(3):758–772.
7. Kirschner PA, Sweller J, Clark RE. Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. Educational Psychologist. 2006;41(2):75–86.
8. Hooper SF, Rieber LP. Teaching with technology. In: Berliner DC, Calfee RC, editors. Handbook of Educational Psychology. Macmillan Library Reference; 1996. p. 553–584.
9. Tedre M, Denning PJ. The long quest for computational thinking. ACM Inroads. 2016;7(1):48–57.
10. Weintrop D, Wilensky U. To block or not to block, that is the question: Students' perceptions of blocks-based programming. Proceedings of the 14th International Conference on Interaction Design and Children. 2015:199–208.