Analysing mathematical programming schemes using different lenses

ANDREAS BORG AND MARIA FAHLGREN

The use of programming in mathematics education is undergoing a renaissance and, in this paper, we analyse students' handling of programming in mathematics using the *Instrumental approach* as a theoretical lens. We are especially interested in analysing the development of mental schemes using two analytical frameworks which are compared and contrasted according the idea of networking theories. The study illustrates that the frameworks' detail of richness can have both advantages and disadvantages and that one of the frameworks are more customed to be applied when analysing students' instrumental genesis concerning the use of a programming environment as a mathematical artefact.

During the past two decades, programming has in many countries been given a more prominent role in lower and secondary school, often linked to the subject mathematics. As a consequence, there has been renewed interest in mathematics education research regarding how the use of programming could help fostering conceptual mathematical understanding as well as developing students' computational thinking. In this paper we focus on students' use of programming when solving a mathematical problem, using the *Instrumental approach* as a theoretical starting point. We are especially interested in applying two different analytical frameworks for describing the utilization schemes developed by students when engaging in mathematical activities involving the use of programming. Based on the two analyses, the aim of the paper is to compare and contrast the two frameworks' affordances when analysing the instrumental genesis of students using programming in mathematics education.

Andreas Borg, *Karlstad University*
Maria Fahlgren, *Karlstad University*

The renaissance of programming in school mathematics

During the 1970s and 1980s, programming became more frequently used for educational purposes. As an advocate for the use of programming in mathematics education, Papert (1980) argued that programming could help students learn the language of mathematics and thereby develop new cognitive skills when engaging in pre-designed programming tasks referred to as microworlds. Sutherland (1994) argued that programming could be seen as a form of experimental mathematics and could also, according to Noss (1986), ”provide children with a framework on which further [mathematical] learning may be based” (p. 353). Although an extensive amount of research agreed on the benefits of using programming in mathematics education, the role of programming in schools diminished during the 1990s.

Since the 2010s, programming has once again been given a more prominent role in education, much due to the technical development in society which has called for an increased digital competence including a more developed computational thinking. Wing (2006) argues that computational thinking involves ”solving problems, designing systems, and understanding human behaviour, by drawing on the concepts fundamental to computer science” (p. 33). Computational thinking could also be developed during mathematical activities which according to Kallia et al. (2021) are characterised by a ”structured problem-solving approach in which one is able to solve and/or transfer the solution of a mathematical problem to other people or a machine by employing thinking processes” (pp. 20–21).

When reintroducing programming in school curricula, different countries have taken different approaches. In England, the subject *Computing* was introduced in 2014 with the aim of learning all students the fundamental ideas of programming (Sangwin & O’Toole, 2017). In countries such as Sweden, Finland (Bocconi et al., 2018), and France (Gueudet et al., 2018), programming has been integrated within the subject of mathematics and regarded as a digital tool for solving mathematical tasks through the design of algorithms. However, Bråting and Kilhamn (2021) argue that the use of programming in mathematics education is not a straightforward process, for example, different roles and meanings of the equal sign and variables in mathematics compared to programming could cause students difficulties. In Denmark, a pilot project was implemented 2018 related to the introduction of the subject *Technology comprehension* (Elicer & Tamborg, 2021). The project has evaluated two ways of implementing this new subject; (1) as a separate subject and (2) integrated into other subjects (e.g. mathematics).

If students' use of programming should contribute to the development of their computational thinking, the students' technical handling of the programming environment is essential. Programming environments such as Scratch, using visual programming, has decreased the technical difficulties relating to the syntax (Resnick et al., 2009) which 30 years ago was one of the reasons why the use of programming in school mathematics drastically decreased. Yet, it could be argued that using programming for mathematical purposes should not be regarded as a straightforward endeavour. Drijvers and Trouche (2008) claim that when students use digital tools in order to solve mathematical tasks, there always exists an intertwining between the technical handling of the tool and the conceptual understanding relating to the mathematical objects which the tool should act upon. Thus, it could be argued that in order to successfully integrate programming as a tool in school mathematics, this intertwining, which is the core of the Instrumental approach, is of special interest.

The Instrumental approach

During the past decades, the Instrumental approach has frequently been used in mathematics education research in order to reach a better understanding of how the appropriation of digital artefacts affect students' building of mathematical knowledge (Buteau, Muller et al., 2020). The instrumental genesis involves "the construction of personal schemes or, more generally, the appropriation of social pre-existing schemes" (Artigue, 2002, p. 250) and during this process, the artefact transforms into an instrument, a psychological construct including not only the artefact itself but also the mental schemes associated with the use of the artefact based on the given purpose (Rabardel, 2002). A mental scheme can be regarded as an "invariant organization of behavior for a certain class of situations" (Vergnaud, 1998, p. 167) or as "a more or less stable way to deal with specific situations or tasks, guided by developing knowledge" (Drijvers et al., 2013, p. 27). Since the existence of the schemes directly relates to students' ability to appropriate a technical artefact for given mathematical purposes, the development of such utilisation schemes is of special interest when investigating students' instrumental genesis. Utilisation schemes consist of two levels of schemes; usage schemes, linked to secondary tasks (i.e., tasks relating to the subject's handling of the artefact), and instrumented action schemes relating to the primary tasks (i.e., actions directed towards the subject's main objective) (Rabardel, 2002). Instrumented action schemes thus consist of usage schemes.

The concept of schemes could be used to understand subjects' stable behaviours but also to describe and gain knowledge about problem-solving processes (Vergnaud, 1998). In order to analyse such processes, schemes can be described using its four components; *goals and anticipations*, *rules of action*, *operational invariants* and *possibilities of infe-rences*. Vergnaud (1998) states that every mathematical activity and consequently every associated scheme is based on more or less reachable goals and anticipations. The rules of actions within the scheme are the gene-rative parts that ”generates behaviours as a function of some situation variables” (p. 229). The operational invariants are, according to Vergnaud (2009), the epistemic aspects of the scheme and comprise concepts-in-action and theorems-in-action. Concepts-in-action are used to select and categorise information relevant in order to solve a specific task and are vital bricks of the theorems-in-action which are propositions held to be true by the subject when she or he acts toward the given object (Vergnaud, 1998). During an activity there also exists possibilities of inferences, that is, ”possibilities of adaptation to the specific features of the situation” (Buteau, Muller et al., 2020, p. 373) which, according to Gueudet et al. (2020), can lead to the development/use of additional rules of action, operational invariants or even completely new schemes.

The instrumental genesis works in two directions (Artigue, 2002). It is directed towards the artefact, a process called *instrumentalization*. During this process, the subject learns about possibilities and constraints given by the artefact (Trouche, 2005) which allows the subject to personalise and modify the artefact in order to better fulfil her/his needs given the object on which the artefact should act upon. *Instrumentation* is the second process of instrumental genesis which is directed toward the subject. During the instrumentation, the artefact prints its marks on the subject and the instrumented action schemes are developed.

Solving a specific task using a technical artefact involves the use of an *instrumented technique* which according to Artigue (2002) is ”a manner of solving a task and, as soon as one goes beyond the body of routine tasks for a given institution, each technique is a complex assembly of reasoning and routine work” (p. 248). Instrumented techniques are often looked upon and assessed based on their pragmatic value ”by focusing on their productive potential (efficiency, cost, field of validity)” (Artigue, 2002, p. 248), but could also have epistemic values since they may help developing the subject's conceptual understanding relating to the object(s) in question. Consequently, when students solve mathematical tasks using a technical artefact, they apply an instrumented technique which potentially both could ease the practical efforts to solve the task and increase students' conceptual understanding relating to the object of which the artefact is used to act upon.

It should be stressed that Vergnaud's (1998) psychological notion of scheme which served as the foundation for the concept of instrumental genesis (Rabardel, 2002) was overlaid by the anthropological notion of technique when the idea of instrumental genesis was taken up by mathematics education researchers (Artigue, 2002).

Analysing scheme development using two different lenses

The aim of this paper is to network two analytical frameworks which previously have been used to analyse scheme development as a part of students' instrumental genesis. According to Prediger et al. (2008), networking theories has in mathematics education research been regarded important due to the variety of conceptual theories existing within the field. They claim that the process of networking theories involves four different networking strategies: *understanding others and making own theories understandable; comparing and contrasting; coordinating and combining;* as well as *synthesizing and integrating*. This paper will focus on the networking strategy comparing and contrasting which aims at highlighting "similarities and differences, possible connections as well as complementary aspects" (Drijvers et al., 2013, p. 25). The aim of this strategy is according to Drijvers et al. (2013) not necessarily to decide which theories are most appropriate to use, but rather to investigate "if the results from both analyses shed new lights on the phenomenon under consideration" (p. 25).

It is important to stress that the aim of this paper is not networking different theoretical frameworks but rather networking two different *analytical* frameworks used within the same theoretical frame (i.e., the Instrumental approach). Yet, we argue that the strategies for networking theories presented in this section could also be applied when analysing the practise of two different analytical frameworks. We argue that analysing data using different analytical frameworks related to the Instrumental approach and networking these frameworks potentially may highlight different aspects of the subjects' schemes relating to the use of specific artefacts as well as differences and similarities between the frameworks.

The first approach for analysing students' instrumented action schemes presented in this paper has been operationalised by Buteau, Gueudet et al. (2020) when investigating the instrumental genesis of university students trying to solve mathematical problems using a programming environment. Buteau, Gueudet et al. (2020) describe the instrumented action schemes using the scheme components introduced by Vergnaud (1998) (*goal and anticipations*, *rules of action*, *operational invariants* and *possibilities of inferences*). They argue that using scheme components when describing students' instrumented action schemes and analysing students' instrumental genesis "deepen[s the] understanding of what is at stake in terms of students' learning in this particular context" (p. 1036). Furthermore, they claim that the instrumented action schemes used by the students to solve a specific task may involve different sets of scheme components due to students' different problem-solving approaches. Gueudet et al. (2020) argue that during a mathematical problem-solving activity involving the use of a programming environment, students develop a complex network of schemes involving mathematical schemes (m-schemes), programming schemes (p-schemes) , and combined programming and mathematics schemes (p + m-schemes). When analysing different schemes developed by students using a programming environment (the artefact) during a mathematical activity, Buteau, Muller et al. (2020) characterise schemes involving goals in which programming and mathematics are closely linked together as *p + m-schemes*. Such schemes could, according to Gueudet et al. (2020), serve as a bridge between mathematical and digital competencies and the operational invariants of such schemes illustrate the relationship between mathematics and programming. But during a mathematical activity involving the use of a programming environment, there may also exist *p-schemes* that only involves rules of action linked to the use of the artefact and not associated with any mathematical content as well as *m-schemes* not associated with the use of the artefact but solely related to the mathematical object(s).

The second approach for describing utilization schemes presented in this paper has been frequently used when analysing students' instrumental genesis relating to their handling of technical mathematical artefacts such as calculators, CAS, and dynamical geometry software (Drijvers & Gravemeijer, 2005; Fahlgren, 2017; Trouche, 2005; Turgut & Drijvers, 2021). Drijvers and Gravemeijer (2005) share the description of schemes introduced by Vergnaud (1998) but claim that schemes are hidden inside the head of the subject and thus cannot be directly observed. Instead, Drijvers and Gravemeijer (2005) argue that the instrumental genesis is best understood by analysing the subject's *instrumented technique* which they describe as "a set of rules and methods in a technological environment that is used for solving a specific type of problem" (Drijvers & Gravemeijer, 2005, p. 169). They argue that the instrumented techniques, which are the technical aspects of the instrumented action schemes made visible for the researcher when studying subjects' use of the artefact, are "the gateway to the analysis of instrumental genesis" (p. 169). When investigating the instrumental genesis through students' instrumented techniques, Drijvers and Gravemeijer (2005) suggest identifying key elements of the instrumented action schemes which could have technical as well as conceptual character. They argue that there is a two-dimensional relationship between the two types of elements since

> on the one hand, the possibilities and constraints of the artifact shape the conceptual development of the user; the conceptions of the user, [and] on the other hand, change the ways in which he or she uses the artifact, and may even lead to changing the artifact or customizing it (Drijvers & Gravemeijer, 2005, p. 168)

which is also emphasised in the differences between instrumentalization and instrumentation. Although this analytical framework has been commonly used in mathematics education research, it has never, as far as we know, been applied when studying the instrumental genesis during mathematical activities where a programming environment constitutes the artefact.

The aim of this paper is to assess the affordances of the two abovementioned analytical frameworks when analysing the instrumental genesis of students using programming during mathematical activity and the specific research question is:

When examining students' use of programming in mathematics, how does the framework for analysing utilization schemes based on the notion of a scheme presented by Vergnaud (1998) compares and contrasts to the well-established analytical framework for analysing schemes based on instrumented techniques (Drijvers & Gravemeijer, 2005)?

Method

The comparison of the two different analytical frameworks will be operationalised by analysing existing data from a study conducted by the first author (Borg, 2021). One of the aims of that study was to analyse the instrumental genesis of Swedish upper secondary school students using programming to solve mathematical problems. The students participating in the study were taking their third course in mathematics (Mathematics 3c) and an introductory course in programming (Programming 1). Although the students had basic knowledge about programming concepts, they had limited or no experience of using programming for mathematical purposes. Due to students' prior experience, Java was the syntax-based programming language used by the students during the intervention together with the programming environment NetBeans 8.2. During the intervention, the students worked in pairs and the activity on their screens together with their mutual conversations were recorded and served as the main data of the study.

The data analysed in the following section concerns a problem-solving activity during which the students were asked to solve a mathematical problem concerning Cinderella who has had her wish not to age fulfilled. Her two older sisters, of whom the oldest was 15 years older than Cinderella inform Cinderella that her wish means that they will lose 480 gold coins the following year, as the sisters each year receive the product of their ages in gold coins from their father. Based on this information, the students were asked to determine the ages of the three siblings. Although the problem can be mathematised algebraically in terms of relationships involving the ages it could not be solved algebraically using analytical methods familiar to the students. Instead, it was intended that the students should solve the problem by conducting an exhaustive trial (using programming) involving the use of loops in order to systematically combine the ages of the sisters and conditionals to test if each combination of ages fulfilled the conditions stated in the task.

Data analysis

When the first author (Borg, 2021) analysed the schemes of the Swedish upper secondary school students using *scheme components* as described by Vergnaud (1998), rules of action were highlighted by the action taken by the students during the mathematical activity (Gueudet et al., 2020). The rationales for the students' actions were, in this way of analysing data, interpreted as operational invariants. Such rationales involving concepts used by the students were categorised as concepts-in-action and rationales involving propositions held to be true by the students were categorised as theorems-in-action. During the analysis, the first author's (Borg, 2021) intention was to stay true to the way Buteau, Gueudet et al. (2020) and Gueudet et al. (2020) analysed schemes developed by university students engaging in problem-solving using programming as a mathematical tool. Table 1 illustrates an example of the scheme components identified by Buteau, Gueudet et al. (2020) when describing a scheme of a student validating the programmed mathematics.

When, in this paper, analysing the same data by scrutinising students' *instrumented techniques*, the authors' intention has been to describe such techniques in the same manner as Drijvers and Gravemeijer (2005). Thus, we have, based on the actions taken by the students, identified core elements of the instrumented action scheme. During the analysis, elements are regarded to have a conceptual character if they are relating to the mathematical object in question and regarded to have a technical character if they are relating to the handling of the technical artefact. Table 2

Table 1. *Components of a student's scheme for validating the programmed mathematics (Buteau, Gueudet et al., 2020, pp. 1032–1033)*

Goal and anticipation	Rules of action	Operational invariants
GoA: Validating the programmed mathematics.	RoA-1: Testing of each vb.net function in isolation, i.e. that it runs and that the expected output is correct. RoA-2: Combining them in simple ways and testing the combination, i.e. that it runs and that the expected output is correct. RoA-3: Checking that the more elaborated combinations of functions runs without crashing or does not ”do something completely unexpected”. RoA-4: Testing the final mathematics result, i.e. that the output is correct.	CiA-1: Programming vb.net function. CiA-2: Method of ”combining functions in simple ways”. CiA-3: ”Unexpected end math result”. CiA-4: Input and output of a program. CiA-5: Mathematical model and process. CiA-6: ”Working” program. TiA-1: ”If every vb.net function works properly and the method you use to join the vb.net function is working properly, and the final product is working properly, then you kind of can assume that the program itself is working properly”. TiA-2: If nothing is really acting unexpectedly, then it is assumed that the program works. TiA-3: If running the program with specific input values of a ”complex” computational mathematical model returns a correct math output, then it is assumed that the program works.

Note. CiA = concept-in-action; TiA = theorem-in-action

Table 2. *Core elements for solving parameterized equations using a symbolic calculator (Drijvers & Gravemeijer, 2005, p. 174)*

Conceptual elements	Technical elements
CE-1: Knowing that the Solve command can be used to express one of the variables in a parameterized equation in other variables. CE-2: Knowing the difference between an expression and an equation. CE-3: Realizing that an equation is solved with respect to an unknown and being able to identify the unknown in the parameterized problem situation. CE-4: Being able to interpret the result, particularly when it is an expression, and to relate it to graphical representations.	TE-1: Remembering the TI-89 syntax of the Solve command, that is Solve (equation, unknown). TE-2: Being able to type in the Solve command correctly on the TI-89.

illustrates an example of core elements identified by Drijvers and Gravemeijer (2005) when describing elements of a scheme for solving parameterized equations using a symbolic calculator.

It should be stressed that the analysis only involves data from a single intervention and will only describe the instrumented action schemes used by the students at this particular moment. The authors therefore make no claims of analysing the students' instrumental genesis in detail which would have required a more longitudinal approach.

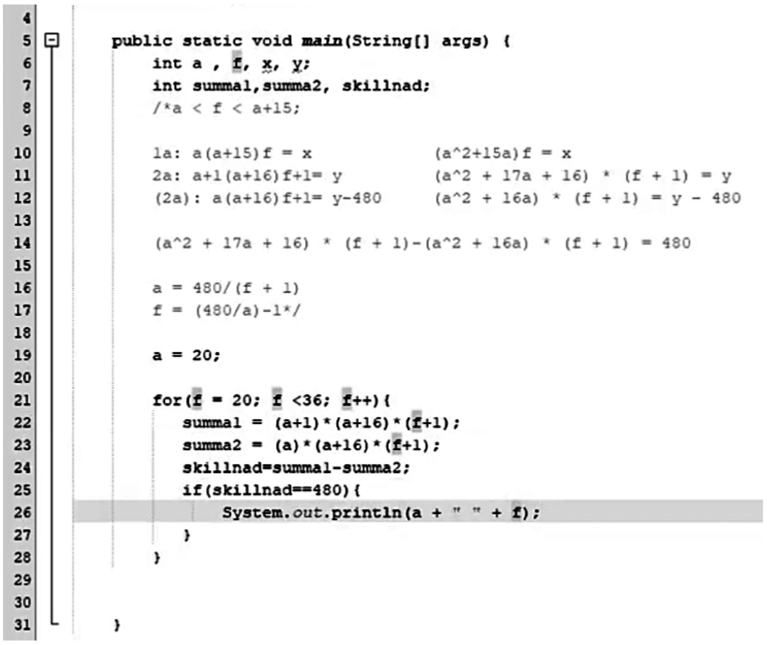
Results

The episode narrated in this section has been chosen on the basis that it contains both mathematical aspects concerning students' use of problem-solving strategy as well as technical aspects relating to the implementation of the strategy using programming. When trying to solve the mathematical problem described in the previous section, the students George and Henrik successfully stated correct mathematical relationships relating to the ages of the sisters but then struggled to articulate an adequate problem-solving strategy and understanding the potential role of programming. Instead, the students tried to simplify the relationships using algebraic manipulations. In the following excerpt, the students discuss their ongoing work with the researcher (first author).

Res.:	And now you have tried to expand and multiply.
Henrik:	Yes, expanded it, multiplied. Kind of to get even more removed. I replaced f in the first so you only got *a*. But I do not know if it helps. But it is in the back of my mind that you have to test because I do not know what else you would use programming for.
Res.r:	No, that may be wise.
Henrik:	But I do not know what to test in that case.
Res.:	No, but you mentioned that you wanted to test so that you sort of could find the optimal ... So that it was true.
George:	Yes, but I don't know how to do it. If you ... No, I don't know.
Res.:	Because then it might be important to test with different values for *a* and different values for *f* and see if you can find a combination that works.
Henrik:	Yes. George, we could actually do that. We ignore everything we've done so far.

The students now start coding and first construct a code (figure 1) which only vary the value of the variable (*f*) corresponding to the age of the mid sister. The variable corresponding to the age of Cinderella (*a*) is given a fixed value in line 19. Within the loop (line 21–28), the variable *skillnad* is assigned a value corresponding to the number of lost gold coins, where the variables *summa1* and *summa2* correspond to the number of gold coins given to the sisters if Cinderella did age or not. The IF statement in line 25 tests if the value of *skillnad* equals 480 (the number of lost coins) and if so, the value of *a* and *f* should be printed.

The students execute the code but their program does not generate any output. At this stage, George's and Henrik's intention is to manually alter the value of *a* each time the program has been run.



```
public static void main(String[] args) {
    int a , f, x, y;
    int summa1,summa2, skillnad;
    /*a < f < a+15;

    1a: a(a+15)f = x                (a^2+15a)f = x
    2a: a+1(a+16)f+1= y             (a^2 + 17a + 16) * (f + 1) = y
    (2a): a(a+16)f+1= y-480         (a^2 + 16a) * (f + 1) = y - 480

    (a^2 + 17a + 16) * (f + 1)-(a^2 + 16a) * (f + 1) = 480

    a = 480/(f + 1)
    f = (480/a)-1*/

    a = 20;

    for(f = 20; f <36; f++){
      summa1 = (a+1)*(a+16)*(f+1);
      summa2 = (a)*(a+16)*(f+1);
      skillnad=summa1-summa2;
      if(skillnad==480){
          System.out.println(a + " " + f);
      }
    }


}
```

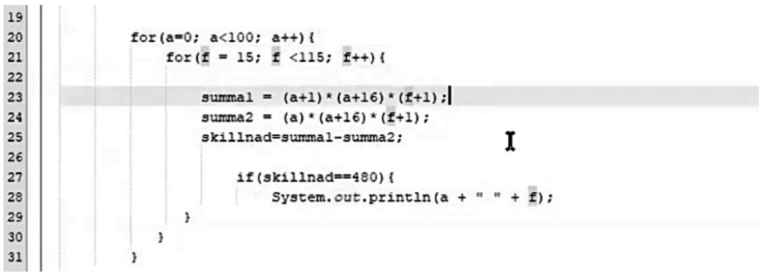
Figure 1. *Code involving the use of a single loop*

George: It will probably not find anything because it (*a*) is not 20. No.

Henrik: She's not 20 then. But then we can only test upwards.

George: Or we do another FOR loop.

The students now add another FOR loop controlling the values of the variable *a* (figure 2) and this loop is also nested with the existing FOR loop controlling the variable *f* (although the initial value of *f* is not fitting based on the task). When executing the code, the program generates four outputs (solutions) and the students start discussing the validity of the solutions (0, 29), (4, 23), (8, 19), and (14, 15).



```
for(a=0; a<100; a++){
    for(f = 15; f <115; f++){

        summa1 = (a+1)*(a+16)*(f+1);
        summa2 = (a)*(a+16)*(f+1);
        skillnad=summa1-summa2;

            if(skillnad==480){
                System.out.println(a + " " + f);
        }
    }
}
```

Figure 2. *Code involving the use of nested loops*

Henrik: 14 and 15 could work. It's the only one that works.

George: Why couldn't Cinderella be... Cinderella could be 8?

Henrik: No, I was joking. She can be 8 or she can be 14. But 23 and 4, and 0 and 29 do not work.

George: Why not?

Henrik: Try to take 29 minus 0, it's greater than 15. Right? And 23 and 4. 23 minus 4 is also greater than 15. So, Cinderella is 8 or 14.

Henrik realises that two of the solutions do not meet the condition $f < a + 15$ and then adds such a condition within the IF statement. Although their code has some minor deficiencies, the executed program now generates two correct solutions.

Describing the scheme using a scheme components (SC) analysis

In this sub-section, the scheme developed/used by the students will be described using scheme components (Vergnaud, 1998). The scheme, which is summarised in table 3, is a combined programming and mathematics scheme (i.e., a p + m-scheme) (Gueudet et al., 2020) since the aim of the activity is to use the programming environment to conduct an exhaustive trial in order to solve a mathematical problem involving algebraic relationships.

Based on the first excerpt involving a discussion between the researcher and the students, Henrik argues that "you have to test" which could be regarded as a *proposition* or a theorem-in-action (TiA-1) that a possible problem-solving strategy for solving the problem would involve the use of an exhaustive trial. TiA-1 is in turn based on two corresponding concepts-in-actions relating to the *concept* of stating mathematical relationships between the variables in play (CiA-1) and the concept of an exhaustive trial (CiA-2). During the discussion, Henrik also realises that an exhaustive trial should be based on the idea of systematically combining variables (CiA-3) which forms a corresponding theorem-in-action (TiA-2).

The students' initial idea when constructing their code in order to conduct an exhaustive trial was to vary only one of the variables using a FOR loop and then vary a second variable manually each time the code was executed. The *actions* taken by the students, when starting their problem-solving process is categorised as rules of action (RoA-1, RoA-2, RoA-3, RoA-5, RoA-6). When Henrik later realised that they could use another loop to vary the variable *a* he brought into action serval new components within their instrumented action scheme relating to the use of nested loops (CiA-4, CiA-5, CiA-6, TiA-3, RoA-4).

Table 3. *Components of George's and Henrik's instrumented action scheme*

Goal and anticipation	Rules of action	Operational invariants
GoA: To use a programming environment to conduct an exhaustive trial in order to solve a mathematical problem involving algebraic relationships between variables.	RoA-1: Formulate the problem situation as amenable to solution through exhaustive trial. RoA-2: Declare computational variables and (where given) assign variables values. RoA-3: Create an iteration and define conditions for the loop. RoA-4: Create nested loops in order to systematically combine variables. RoA-5: Make use of the conditional operator IF to (a) evaluate given conditions in order to (b) perform different actions based on the validity of the given conditions. RoA-6: Create an output which shows the values of variables. RoA-7: Coordinate necessary rules of action (as efficiently as possible) so as to construct a solution procedure. RoA-8: Make sure that the program takes account of all the information in the problem statement.	CiA-1: The idea of expressing a mathematical problemin terms of (one or more) algebraic relationships between variables. CiA-2: The idea of conducting an exhaustive trial. CiA-3: The idea of systematically combining variables. CiA-4: The idea of establishing a loop relating to each of the variables in play. CiA-5: The idea of nesting loops (and statements within them) in order to achieve an appropriate sequence of variable-related actions. CiA-6: The idea of using conditions within loops and conditional operators in order to extract solutions within a given range during an exhaustive trial. TiA-1: Conducting an exhaustive trial is a means of solving mathematical problem involving algebraic relationships between variables. TiA-2: Systematically combining variables is a means for conducting an exhaustive trial. TiA-3: Establishing a loop for each variable in play and nesting these loops is a means for systematically combining variables.

Note. CiA = concept-in-action; TiA = theorem-in-action

The students seemed confused by the fact that their program then generated four different outputs. It could be argued that testing the generated solutions meant that the students subsequently applied new rules of action relating to coordination of the necessary rules of action (RoA-7) as well as to the fact that all the information in the problem should be taken into account (RoA-8).

Since the data analysed in this paper only concerns short excerpts from students' use of the programming environment during a single lesson, it was not possible to analyse the existence of *possibilities of inferences*.

Describing the scheme using an instrumented techniques (IT) analysis

In accordance with the work of Drijvers and Gravemeijer (2005), the instrumented action scheme of the students George and Henrik will, in this sub-section, be described by distinguishing conceptual and technical elements (table 4) based on the students' instrumented techniques.

Table 4. *Core elements of George's and Henrik's instrumented action scheme*

Conceptual elements	Technical elements
CE-1: Realising the possibility to solve a mathematical problem involving algebraic relationships between variables by using an exhaustive trial.	TE-1: Being able to define computational variables and assigning such variables values.
CE-2: Knowing that an exhaustive trial implies that every single combination of ages must be evaluated in order to determine if they fulfil the conditions given in the mathematical problem.	TE-2: Being able to create a FOR loop and define conditions for the loop.
CE-3: Realising the importance of stating ranges for the variables in play based on the condition(s) given in the mathematical problem.	TE-3: Being able to create nested loops in order to systematically combine variables.
	TE-4: Being able to create IF statements to (a) evaluate given conditions in order to (b) perform different actions based on the validity of the given conditions.
	TE-5: Being able to create an output which shows the values of variables.

Based on the conversation in the first excerpt, Henrik seems to realise that conducting an exhaustive trial would be a possible way of finding solutions to the problem, which could be described as an important conceptual element (CE-1) of the developing scheme since it is foremost concerning the mathematical objects relating to the problem. After the discussion with the researcher, Henrik also realises that such a trial has to include testing all relevant combinations of ages towards the conditions given in the problem which is regarded as another conceptual element (CE-2). He also appreciates the potential role played by the technical artefact in order to conduct such a trial which brought into action several technical elements (TE-1, TE-2, TE-4, TE-5) relating to the specific handling of the programming environment (the artefact). After constructing a code that used a single loop to vary only one of the variables, the students used nested loops in order to systematically combine two variables (TE-3).

Conceptual elements could guide the activity but could also be the outcome of the activity (Drijvers et al., 2013) which was illustrated when the two students discussed the validity of the four solutions generated by their code. The use of their initial IF statement had printed combinations of ages which only fulfilled the condition related to the number of gold coins lost by the sisters. But the output of their program together with the following discussion made it evident to the students the necessity of being thorough when stating variable ranges for the variables relating to the conditions given in the problem (CE-3). This outcome could also be seen as an example of instrumentation, where the artefact influences the subjects' perception regarding the mathematical object.

Discussion

The previous section has offered a description of an instrumented action scheme using two different analytical frameworks. The scheme was described by studying the students' instrumented techniques (IT) and defining core elements of the scheme as well as by describing the scheme using scheme components (SC). In this section, relating to the research question of this paper, we compare and contrast the two analytical frameworks from a networking perspective (Prediger et al., 2008) based on the outcome of the two separate analyses including a discussion concerning the affordances of the analytical frameworks when analysing the instrumental genesis of students using programming in mathematics.

Comparing and contrasting the two analyses

A clear difference between the schemes described using the two different analytical frameworks is the richness of detail. Whereas the scheme described using the IT analysis includes eight core elements divided into two categories, the SC analysis describes the scheme by using 18 components of three different types. In accordance with the view of Buteau, Gueudet et al. (2020), it could be argued that the richness of detail illustrates the complexity of an instrumented action scheme. Although no such comparison was made in this paper, it could be argued that a more detailed description of a scheme would also facilitate comparisons between schemes developed by different students. But the fact that the scheme generated through the IT analysis was less detailed could also be seen as an advantage since the focus remains on the conceptual and technical aspects of the scheme and how these aspects or elements are intertwined (Drijvers & Gravemeijer, 2005). In line with Drijvers and Gravemeijer (2005), we argue that, especially for novice researchers, the meaning and application of conceptual and technical elements are relatively easy to grasp whereas, in our view, the implications of rules of action, concept-in-action, and theorems-in-action could be perceived as less accessible.

We argue that the nature of the artefact could also affect the applicability of the frameworks. IT analyses in existing literature often concern the use of calculators, CAS, or dynamical geometry software whereas the SC analysis in the form presented in this paper only has been applied when studying students' use of programming in a mathematical context. An important difference between programming environments and the other technical artefacts mentioned in relation to the IT analysis, is that a syntax-based programming environment is not primarily designed for educational purposes, and more important not designed as a mathematical tool (Buteau, Gueudet et al., 2020). Whereas the syntaxes used when handling for example a handheld calculator are pre-designed in order to conduct mathematical actions, the corresponding actions using a programming environment often has to be designed by the user with the help of programming concepts, a design which could be hampered by the different meanings existing between symbols used in both programming and mathematics (e.g. variables and the equal sign) (Bråting & Kilhamn, 2021). Since previous research (Drijvers & Gravemeijer, 2005) has shown that students' handling of syntaxes related to the use of calculators are far from straightforward we claim that the use of a syntax-based programming language for mathematical purposes could potentially be perceived as even more complicated. There could thus exist a great *distance* (Haspekian, 2005) between how students solve mathematical problems using paper and pencil compared to how similar problems are to be solved using programming. This distance, which could hamper the instrumental process, is among other things affected by the different meanings of symbols used in both mathematics and programming (Bråting & Kilhamn, 2021). It could thus be argued that the adaption of the programming environment for mathematical purposes is an essential part of the instrumental genesis during which the subject modifies the artefact in order to better fulfil her/his needs (Rabardel, 2002). We argue that the more detailed SC analysis has the potential to highlight this adaption process and in accordance with Gueudet et al. (2020) we claim that the SC analysis in a clear way visualises the bridge between mathematical and digital competencies and the adaption of the programming environment for mathematical purposes by referring to key operational invariants related to both mathematical and programming knowledge (i.e., CiA-2, CiA-3, CiA-4, CiA-5, TiA-1, TiA-2). We argue that this bridge is less evident in the IT analysis and in order to highlight the complexity of using technical artefacts, such as programming environments, not specifically designed as mathematical tools, we suggest that the IT analysis should not only involve conceptual elements relating to mathematical objects but also *conceptual programming elements* relating to the use of the artefact for mathematical purposes.

Comparing and contrasting the two frameworks

Since the instrumented techniques and the components of schemes have different functions in the Instrumental approach, the two analytical frameworks applied in this paper to describe instrumented action schemes use different notations. But we argue that it is still possible to establish links between them (table 5). We regard rules of action as the generative part of the scheme and thus, like the instrumented techniques, the most observable part of the scheme. Rules of action just like instrumented techniques could also have a pragmatic as well as a

Table 5. *The relationships between scheme components and core elements*

SC analysis	IT analysis
Rules of action	*Instrumented techniques*
Examples: Create an iteration and define conditions for the loop (RoA-3). Create nested loops in order to systematically combine variables (RoA-4).	Examples: Students' use of loops in order to combine values of variables involving technical elements such as being able to create a FOR loop and define conditions for the loop (TE-2) and being able to create nested loops in order to systematically combine variables (TE-3).
Concepts-in-action	*Conceptual elements*
Examples: The idea of expressing a mathematical problem in terms of (one or more) algebraic relationships between variables (CiA-1). The idea of conducting an exhaustive trial (CiA-2). The idea of nesting loops (and statements within them) in order to achieve an appropriate sequence of variable-related actions (CiA-5).	Examples: Realising the possibility to solve a mathematical problem involving algebraic relationships between variables by using an exhaustive trial (CE-1). Knowing that an exhaustive trial implies that every single combination of ages must be evaluated in order to determine if they fulfil the conditions given in the mathematical problem (CE-2).

epistemic function (Ruthven, 2002). We also argue that concepts-in-action (relating to mathematical concepts) could be viewed as the psychological correlate of conceptual elements, whereas theorems-in-action, which we regard as the bridge between concepts-in-action and rules of action, lacks an obvious equivalent in the IT analysis. Describing the scheme using SC could thus be considered as a more nuanced approach, recognising the interweaving within the instrumental genesis of what could be referred to as technical and conceptual aspects of mathematical capability.

Conclusions

In this paper, we have tried to describe students' instrumented action schemes using two different analytical frameworks. Since the analysed data is limited and since the work of the participating students only involves one single intervention, we cannot comment on the students' instrumental genesis. Also, we cannot claim that the scheme analysed really is an invariant organization of behaviour for this specific type of situation. Yet, we argue that the scheme analysed at least could be regarded as a scheme in progress, which development together with the artefact initiates the creation of the instrument.

The analytical frameworks used to analyse the same data in this paper describe schemes differently. Since Drijvers and Gravemeijer (2005) argue that the schemes are hidden for a researcher inside to head of the subject, they are restricted to describing the core elements of the scheme illustrated through the instrumented techniques. This means that the description of a scheme becomes less fine-grained than the schemes described by Buteau, Gueudet et al. (2020) who regard the subject's actions and the verbal justification of such actions during a mathematical activity as evidence for the components that form the scheme. Although it could be argued that the detail of richness in the SC analysis could potentially highlight aspects which would be hidden when only discussing core elements of the scheme, it could also be discussed in how much detail you can describe a mental construct such as a scheme. It could, in accordance with Drijvers and Gravemeijer (2005), be argued that it is impossible to directly observe a scheme which implies that descriptions of schemes using SC are based on hypothesis of the observed behaviours. There is thus a risk that the level of detail gained when describing a scheme using SC comes with a cost, the reliability of the analyses.

We have also discussed that using a programming environment, an artefact not designed primarily as a tool for mathematical purposes, may increase the complexity of the instrumented action schemes developed by the students. Through the operational invariants linking mathematics and programming knowledge together, this complexity is more naturally visualized during a SC analysis. Therefore, we argue that there is a need for expanding the conceptual elements in an IT analysis to also include conceptual programming elements relating to the mathematical possibilities and constraints offered by the artefact.

Our intention in this paper has been to compare and contrast two different analytical frameworks for analysing instrumented action schemes as a part of students' instrumental genesis. When comparing and contrasting the two analytical frameworks, our aim has not primarily been to decide which of these lenses are the most suitable to use, but to highlight the pros and cons relating to each framework. Since this paper concerns a limited amount of data restricted to the Swedish context, we finally call for further research which applies the SC analysis in situations concerning the use of other artefacts than programming. We also call for more research, similar to that of Haspekian (2005), trying to operationalise the IT analysis when students engage in mathematical activities using artefacts which are not primarily designed as mathematical tools.

References

Artigue, M. (2002). Learning mathematics in a CAS environment: the genesis of a reflection about instrumentation and the dialectics between technical and conceptual work. *International Journal of Computers for Mathematical Learning*, 7 (3), 245–274. doi: 10.1023/a:1022103903080

Bocconi, S., Chioccariello, A. & Earp, J. (2018). *The Nordic approach to introducing computational thinking and programming in compulsory education*. Report prepared for the Nordic@BETT2018 Steering Group. doi: 10.17471/54007

Borg, A. (2021). *Designing for the incorporation of programming in mathematical education : programming as an instrument for mathematical problem solving* [Licentiate thesis]. Karlstads University. http://urn.kb.se/resolve?urn=urn:nbn:se:kau:diva-85625

Bråting, K. & Kilhamn, C. (2021). Exploring the intersection of algebraic and computational thinking. *Mathematical Thinking and Learning*, 23 (2), 170–185. doi: 10.1080/10986065.2020.1779012

Buteau, C., Gueudet, G., Muller, E., Mgombelo, J. & Sacristán, A. I. (2020). University students turning computer programming into an instrument for "authentic" mathematical work. *International Journal of Mathematical Education in Science and Technology*, 51 (7), 1020–1041. doi: 10.1080/0020739X.2019.1648892

Buteau, C., Muller, E., Mgombelo, J., Sacristán, A. I. & Dreise, K. (2020). Instrumental genesis stages of programming for mathematical work. *Digital Experiences in Mathematics Education*, 6 (3), 367–390. doi: 10.1007/s40751-020-00060-w

Drijvers, P., Godino, J. D., Font, V. & Trouche, L. (2013). One episode, two lenses. *Educational Studies in Mathematics*, 82 (1), 23–49. doi: 10.1007/s10649-012-9416-8

Drijvers, P. & Gravemeijer, K. (2005). Computer algebra as an instrument: examples of algebraic schemes. In D. Guin, K. Ruthven & L. Trouche (Eds.), *The didactical challenge of symbolic calculators: turning a computational device into a mathematical instrument* (pp. 163–196). Springer.

Drijvers, P. & Trouche, L. (2008). From artifacts to instruments: a theoretical framework behind the orchestra metaphor. In K. Heid & G. Blume (Eds.), *Research on technology and the teaching and learning of mathematics* (pp. 363–392). Information Age.

Elicer, R. & Tamborg, A. L. (2021). *Nature of the relations between programming and computational thinking and mathematics in Danish teaching resources*. Paper presented at ICTMT 15, Copenhagen.

Fahlgren, M. (2017). Redesigning task sequences to support instrumental genesis in the use of movable points and slider bars. *The International Journal for Technology in Mathematics Education*, 24 (1), 3–15. doi: 10.1564/tme_v24.1.01

Gueudet, G., Bueno-Ravel, L., Modeste, S. & Trouche, L. (2018). Curriculum in France: a national frame in transition. In D. R. Thompson, M. A. Huntley & C. Suurtamm (Eds.), *International perspectives on mathematics curriculum* (pp. 41–70). Information Age.

Gueudet, G., Buteau, C., Muller, E., Mgombelo, J. & Sacristán, A. I. (2020). Programming as an artefact: What do we learn about university students' activity? In T. Hausberger, M. Bosch & F. Chellougui (Eds.), *INDRUM2020 Proceedings* (pp. 443–452). University of Carthage and INDRUM.

Haspekian, M. (2005). An "Instrumental approach" to study the integration of a computer tool into mathematics teaching: the case of spreadsheets. *International Journal of Computers for Mathematical Learning*, 10 (2), 109–141. doi: 10.1007/s10758-005-0395-z

Kallia, M., van Borkulo, S. P., Drijvers, P., Barendsen, E. & Tolboom, J. (2021). Characterising computational thinking in mathematics education: a literature-informed Delphi study. *Research in Mathematics Education*, 1–29. doi: 10.1080/14794802.2020.1852104

Noss, R. (1986). Constructing a conceptual framework for elementary algebra through Logo programming. *Educational Studies in Mathematics*, 17 (4), 335–357. doi: 10.1007/BF00311324

Papert, S. (1980). *Mindstorms: children, computers, and powerful ideas*. Basic Books.

Prediger, S., Bikner-Ahsbahs, A. & Arzarello, F. (2008). Networking strategies and methods for connecting theoretical approaches: first steps towards a conceptual framework. *ZDM*, 40 (2), 165–178. doi: 10.1007/s11858-008-0086-z

Rabardel, P. (2002). *People and technology: a cognitive approach to contemporary instruments*. https://hal.archives-ouvertes.fr/hal-01020705

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E. et al. (2009). Scratch: programming for all. *Communications of the ACM*, 52 (11), 60–67. doi: 10.1145/1592761.1592779

Ruthven, K. (2002). Instrumenting mathematical activity: reflections on key studies of the educational use of computer algebra systems. *International Journal of Computers for Mathematical Learning*, 7 (3), 275–291. doi: 10.1023/A:1022108003988

Sangwin, C. J. & O'Toole, C. (2017). Computer programming in the UK undergraduate mathematics curriculum. *International Journal of Mathematical Education in Science and Technology*, 48 (8), 1133–1152. doi: 10.1080/0020739X.2017.1315186

Sutherland, R. (1994). The role of programming: towards experimental mathematics. In R. Biehler, R. W. Scholz, R. Sträßer & B. Winkelmann (Eds.), *Didactics of mathematics as a scientific discipline* (pp. 177–187). Springer.

Trouche, L. (2005). An instrumental approach to mathematics learning in symbolic calculators environments. In D. Guin, K. Ruthven & L. Trouche (Eds.), *The didactical challenge of symbolic calculators: turning a computational device into a mathematical instrument* (pp. 137–162). Springer.

Turgut, M. & Drijvers, P. (2021). Instrumentation schemes for solving systems of linear equations with dynamic geometry software. *International Journal for Technology in Mathematics Education*, 28 (2), 65–80.

Vergnaud, G. (1998). A comprehensive theory of representation for mathematics education. *The Journal of Mathematical Behavior*, 17 (2), 167–181. doi: 10.1016/S0364-0213(99)80057-3

Vergnaud, G. (2009). The theory of conceptual fields. *Human Development*, 52 (2), 83–94. doi: 10.1159/000202727

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49 (3), 33–35. doi: 10.1145/1118178.1118215

Andreas Borg

Andreas Borg is a doctoral student in educational work at Karlstad University, Sweden. His research focus is on upper secondary students' use of programming for mathematical purposes as well as on the orchestration of learning activities involving the use of programming as a mathematical tool.

andreas.borg@kau.se

Maria Fahlgren

Maria Fahlgren is an associate professor in mathematics education at Karlstad University, Sweden. Her research focus is on the use of digital technologies in the teaching and learning of mathematics, with a particular focus on task design, at both the upper school level and tertiary level.

maria.fahlgren@kau.se