*Jl. of Interactive Learning Research* (2000) **11**(3/4), 465-484

# Arithmeticus: A DPS-Based Model for Arithmetical Competence

JOSEPH KLEP
*Institute for Curriculum Development*
*Boulevard 1945 3, 7511 CA Enschede*
*P.O. Box 2041, 7500 CA Enschede, The Netherlands*
J.Klep@slo.nl
http://www.slo.nl/network/engels.html

## The Position of Arithmeticus Between other Student Models

Petrushin and Sinitsa (1990) offer a short classification of learner models. They distinguish between fixing and simulating models. Both kinds of models are based on a set of expert-rules. The learner model is a description of students behaviour related to that set of expert-rules.

Arithmeticus does not contain expert-rules in the field of mathematics, but meta-mathematical rules, telling which transitions are permitted in calulations or in argumentations. Using those rules, Arithmeticus can construct lines of arguments.

If a student has expressed his solution of a problem, Arithmeticus tries to realise (for itself) this expressed solution by constructing a fitting line of arguments (or algorithm). So, Arithmeticus offers a generic description of mathematical knowledge based on meta-mathematical rules. The kernel of Arithmeticus is not a psychological, but an epistemological, generic description of mathematics (arithmetics).

Around this kernel, we created a psychological shell by annotating solutions of children. It works like this: In MathMirror (front end), a student can express calculations by manipulating mathematical objects. Those manipulations (solutions) are recorded with time annotations. Comparing these students' expressions with solutions produced by Arithmeticus gives a lot of information about the students work. So, it is possible to qualify solutions in terms as effectiveness, speed, degree of automation, and used rote knowledge. Those qualifications are based on learning history. For instance, newer solutions can repress older ones, and automatisms can improve or deprove. Arithmeticus describes annotated and qualified students' solutions. Solutions are added to the set of rules that Arithmeticus uses in producing calculations or argumentations.

Arithmeticus can “learn” rules that are constructed by Arithmeticus and recognised in students' work. This way, Arithmeticus is a generic representation of what a student might learn.

Arithmeticus is not a psychological model of how a student thinks. It hasn’t that pretention. We think of Arithmeticus as a student's mathematical mate, who tries to know what his friend is doing. Arithmeticus remembers what the student-friend has done before and can understand the students more and more. So, large pieces of argumentations can remain implicit after some communication between a student and Arithmeticus. If the student-friend falls back (regression) or produces something new, Arithmeticus can mark it.

A student can fail in constructing or in expressing an algorithm or argumentation. There are two levels of errors:

- Not acceptable transitions: logical or syntactical errors.
- Dirty argumentations: formally correct but inefficient or regressive argumentations.

Logical and syntactical errors are detected immediately in the user interface. Dirty argumentations are detected by Arithmeticus immediately after a student has performed a given solution. Arithmeticus does not produce errors like perturbation models. In fact, errors are not very important in Arithmeticus because this system is interested in how a student can combine acquired personal knowledge to create interesting solutions. Arithmeticus can produce reactions such as, “Well, it is correct, but you can produce a shorter solution.” Nevertheless, it is possible to add error generation to Arithmeticus, too.

Arithmeticus is an epistemological, generative description of solutions that a student is able to generate. A psychological shell qualifies students' solutions in terms of knowledge and skills, which are stored in students’ learning history databases. Arithmeticus learns with a student, and can communicate with the student about qualities of students' solutions. In this context, some epistemological aspects, ideas about learning mathematics, DPS-based microworlds, and generic planning of exercises are discussed.

### Education

Thinking is a chaotic process, but we can assess flows of thoughts by expressing them in a language. Which expressions are acceptable and which are not is culturally determined. So, concepts and lines of argument are not necessarily successful representations of thinking. In language, philosophy concepts and logic are products of the culture of a person, a group, or a society. The scholastic way of defining concepts by describing essences is only one of the possible ways of representing knowledge, maybe not very interesting as far as learning is concerned.

In expert systems, models of cognition are often based on scholastic knowledge representations. That might be useful in illustrating, explaining, and criticising scientific results and methods. It might be less useful in provoking creative thinking.

In this article, we concentrate on the question of how we can enrich creative thinking and how we can coach learning. A student in mathematics is not someone who has to "behave like an expert mathematician," but someone who is creative in thinking and sensible in communication between mathematicians.

Instruction (in mathematics) should concentrate on the enrichment of creative thoughts, facilitating expressions, and on reflection on communication (between mathematicians)**.** Learning (mathematics) is not learning to behave like an expert (mathematician)**.**

## ABOUT CONCEPTS

From a (socio-) constructivist point of view (Varela, 1990), man is constructing knowledge by expressing thoughts with physical and linguistic acts, and reflecting on the effects of those expressions in the physical and social environment. Concepts are not defined by essences and attributes, but concepts are clusters of experiences that people talk about in the same kind of words. The relation "… is (an) …" ( or nearly equivalent: "…is a subset/element of …") is fundamental in traditional logic.

*A butterfly is symmetric.*

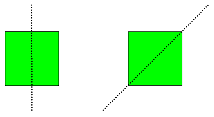
*A square is a quadrangle with sides of equal length and right angles.*

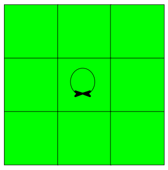
Relations like "…is like…" are important in language philosophy or in constructivist psychology.

*A butterfly opens and closes itself like a book: both the wings of the butterfly and the pages of the book fit precisely on each other.*

*Something is symmetrical if we can agree it is like a butterfly.*

*Something is a square if it is just like a piece of paper that can be fold ed symmetrical in two ways.*





*Or something is a square if it is just like a flag that can be rotated in a grid of flags.*

The scholastic or Aristotelian way of modelling concepts in terms of essences is a very important logical scientific method. But maybe it is not very helpful when thinking about thought. In language philosophy it is not important to ask, “What is a book ?” but to ask, “In what circumstances do people use the word ‘book?'” People do not use the word "book" because all books have the same book-ness as common essence, but they use “book” for things that are less or more alike in some conventional sense.

This seems a very loose way of defining concepts, but it seems to be more like what really happens between people, and maybe more like what happens in people's minds. Following the ideas of Varela, I prefer to say, “We think by imagination.” Our thinking is a flow of imagination. Sometimes, that flow is well known, based on experience, sometimes it is very chaotic and directed by mutually competitive associations.

Realising or understanding what has been said or done is the other way around to express thoughts. Realising is trying to have a fitting flow of thoughts, and fitting means: “I could have expressed this flow of thoughts in the same way as I have heard or seen now.”

## MATHEMATICAL METHOD

In the ISMA project, we proposed to make a difference between thinking and reasoning. To think is a creative, very loose, and unstructured process (Varela). To reason is expressing a flow of thoughts in lines of arguments in a language. A mathematician tries to assess thoughts and intuitions (mathematical method) by expressing them in a formal language in which the validity of lines of arguments is based on the conventions in mathematical logic. That convention, itself, is a result of analysis of lines of arguments that are widely accepted in a mathematical community.

Someone can learn mathematics in a community by trying to express thoughts about numbers and structures, and by reflecting on the reactions of others on those expressions. So, someone learns mathematics by practising this mathematical method.

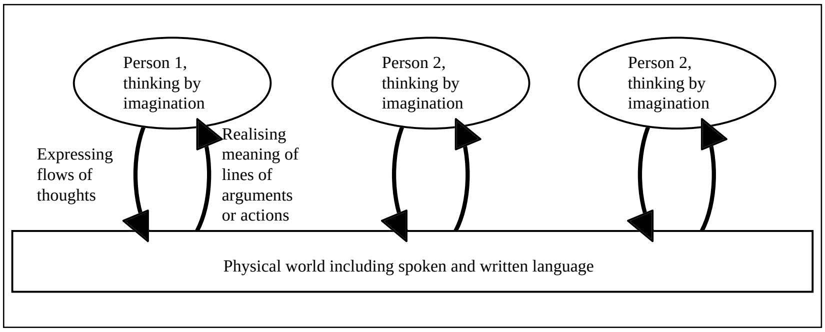

Person 1, thinking by imagination
Person 2, thinking by imagination
Person 2, thinking by imagination
Expressing flows of thoughts
Realising meaning of lines of arguments or actions
Physical world including spoken and written language

**Figure 1.** Communication in a mathematical community: expressing and realising are culturally bound.

## THINKING AND REASONING

Reasoning is a language-bound part of thinking. Before saying or doing something, one can express thoughts in language and critique whether those expressions will be successful or not. An expression is successful if someone else can recognize the expression and agree with it. So, thinking is not language bound, but it is a flow of imagination. A part of that thinking is reasoning, and that is creating and criticising expressions. Reasoning is method based, thinking is not.

In mathematics, methods are very rigorous. It is possible to construct lines of arguments in a logical, correct way, and logically correct lines of arguments have a very good chance of being accepted. Expressing flows of thoughts implies reflection on the acceptability of formulated expressions, that is, applying mathematical or other methods.

In mathematics communication, there are mainly two kinds of reactions to an expression:

1. Formal reactions: I think your expression is (not) logically correct.
2. Semantic reactions: I can recognize what you expressed and I agree (not).

## Dynamic Problem Spaces (DPS) and Mathematical MicroWorlds (MMW).

Students who have to solve a mathematical or arithmetical problem have to find an adequate orientation to their task. If a mathematical problem is submitted to a student, pieces of his (mathematical) knowledge are associated with that problem. This associated knowledge is not a static, but a dynamic knowledge that changes with the state of the problem-solving process. These pieces of actual knowledge are called a dynamic problem space (DPS) (Newell & Simon, 1972).

In Klep (1992), mathematical microworlds (MMW) have been presented as learning environments:

- They help children to actualise their DPS for the current problem state.
- Students can express their reasoning or calculations.
- MMWs offers feedback on the progression or regression of mathematical activity of the student.

A good MMW should reflect most relevant elements of the current DPS of a child, corresponding to the current problem state in the MMW. A good MMW should also support reflection of a student on solutions or actions. MathMirror is such a MMW.

In order to present suitable information, suggestions, and tools in an MMW, we need a Students Math Model (PMM), in which the mathematical knowledge of a student can be represented and from which actual DPS's can be generated. Arithmeticus is such a PMM.

Given a problem P, the student has a DPS associated with P. If the student has a poor DPS, the problem is not very meaningful. If the student has a rich DPS, then the problem is more meaningful, and thestudent might have a good chance to have a strategy to construct a solution for that problem.

## Experts, Expert-Knowledge and Expert Systems.

Experts in any area are people with very rich DPSs that can be activated when the expert is thinking about that area. An expert is well trained in using conventional methods concerning that area.

In this epistemological view on experts, it is important to understand that someone is free to use a method or not. Sometimes, a mathematician can say, “Well, your proof is correct, but I feel your thesis is not good.” One of the worst things a student in mathematics can do is try to find a solution by applying lines of arguments that are known by heart.

In most expert systems, expert knowledge is modelled in facts and rules. Those facts and rules are found by knowledge elicitation. But this kind of expert system represents only the lines of arguments and the methods used by experts, that is, not the experts’ knowledge. In terms of Figure 1, expert systems represent objects, actions, and written and spoken language in a formal (written) world and the (logical) relations between them. They do not represent the thinking process itself.

## Competence versus Performance in Reasoning

The difference between thinking and reasoning is made clear when a child can say, “I think I can understand this, but I cannot explain it.” In school, good performance (correct lines of arguments and computations) is often identified with “someone is good in mathematics.” That is not necessary true. A child might know one correct and well-trained type of solutions. In that case, they have very poor DPSs in thinking. Maybe they are not very creative in mathematics.

Bad performance is often identified with bad thinking. That seems to be too fast. There is not much interest in the problems children can have expressing their mathematical imaginations.

Because thinking and performance are often identified, much instruction is designated to tell children “how to solve” a problem or “how to think.” That kind of instruction neglects the nature of thinking processes in a child’s mind. So, there is a gap between thinking and performance.

Nevertheless, lines of arguments and calculations are the only things that can be assessed because thinking is very hard to observe. But my assumption is that most of the time, there are some successful thoughts behind successful expressions. So, a good opportunity to enrich thinking is to remind a child of earlier, successful thoughts and give suggestions that are easily relatable to those earlier thoughts. In this epistemology, even the most rigid instruction is a kind of enrichment of thinking (and DPS's), based on assessment of the mathematical performance of children.

## EDUCATION

Good education offers children a rich MMW that reflects, in some sense, elements of their DPS of a problem. More exactly, a good MMW offers expressions that can be realised, easily, by the student. So, education is not offering "what a good expert would do." That might be helpful for the enrichment of a child's DPS, but is rather indirect because offering expert behaviour is rather prescriptive and not an aid for creative thinking and formulating thoughts.

## MAN—COMPUTER DIALOGUES IN THE ISMA PROJECT

In natural person-to-person communication, people give answers after realising what the other person says (Figure 1). What could be the idea of man-computer interaction? In the ISMA project, we have chosen an interaction such as Figure 2. In this communication, Aritmeticus tries to interpret students' (incomplete) expressions as a line of argument (a calculation).

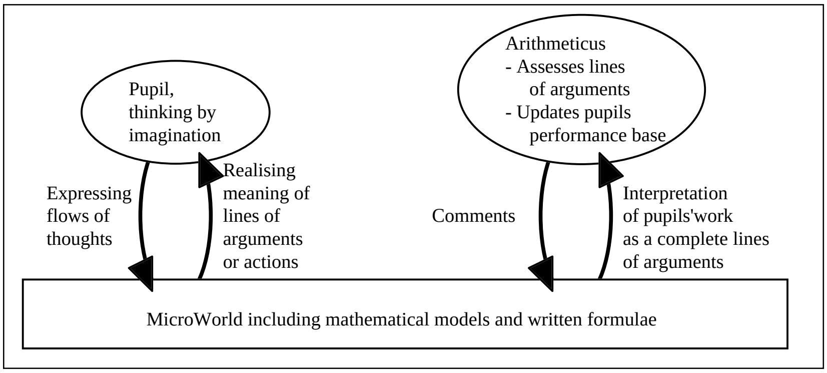

Pupil, thinking by imagination
Arithmeticus
- Assesses lines of arguments
- Updates pupils performance base
Expressing flows of thoughts
Realising meaning of lines of arguments or actions
Comments
Interpretation of pupils'work as a complete lines of arguments
MicroWorld including mathematical models and written formulae

**Figure 2.** Man-computer communication in the ISMA project

In the man-computer interaction in the ISMA project, the computer is a rather poor listener compared to a person. It only compares and assesses actual calculations and lines of argument of a child with "what a student is able to do." The front-end in which a child works in is called "MathMirror," because the program reflects a description of students' reasoning performance, the DPS of a child, and the quality of the solution compared to earlier work. When starting with a new Student Arithmeticus, the knowledgebase is nearly empty; it only contains rules determining how to create algorithms, based on a set of elementary rules and relations. Basic knowledge is addition and subtraction with 0 and 1. How these programs work will be explained in next paragraphs.

## Students Errors

Until now, the ISMA project did not concentrate on procedural errors. In fact, only formal errors were reflected to make the child think about "a failing communication." If an error occurs, the child had to make a desicion and recreate the steps or solution.

Qualities of solutions are compared to the child's learning history and the solutions that should be calculated based on skill level. Correct solutions, solutions without formal errors, are correct for Arithmeticus. Nevertheless, they can be ineffective, too long or complex. The system can give comments such as, "I think you can find a shorter solution."

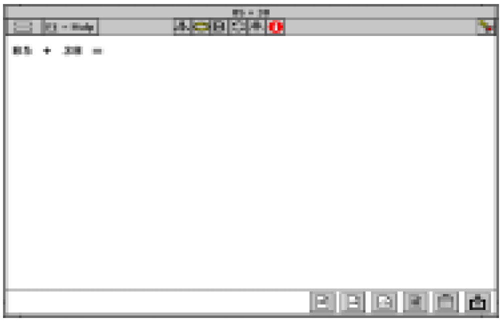
## MathMirror

In the next paragraphs, MathMirror will be presented. This is followed by a student's solution in MathMirror that will be discussed, and then Arithmeticus will be presented. In the last paragraph, implications of Arithmeticus will be discussed. MathMirror[1] is an experimental MMW in which children can solve arithmetical problems.

MathMirror, in its experimental setting, offers a modest presentation. MathMirror and Arithmeticus are test cases to recognize and test the epistemological and educational concepts of DPS and MMW as described previously. Early mathematics is a small but very rich area in which ideas can be tested.

MathMirror is a front-end, and Arithmeticus is a student's model for learning early mathematics. They provide children a program for constructing and training strategies for reducing addition, subtraction, multiplication and division in early mathematics. (Age 7-12). Both in MathMirror and Arithmeticus, fractions and real numbers are provided in the program's architecture. (In an experimental setting they can be used already.) The concept of

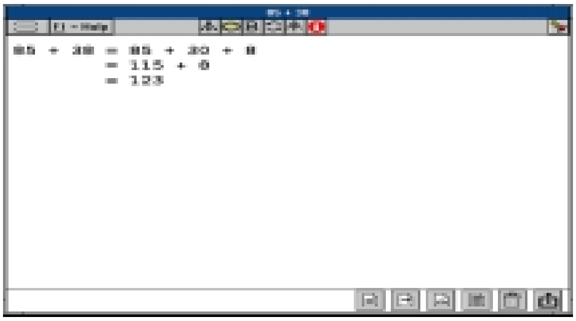
Arithmeticus is very general: all formal knowledge and algorithms can be handled in the same way.



```
85 + 38 =
```

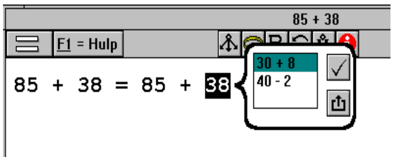
**Figure 3.** “Worksheet”

In Figure 3 a “worksheet” is shown in which the formula 85 + 38 is presented. The task is to reduce 85 + 38 to 123. If a student knows the answer, 123 can be written, and MathMirror will accept that. If a child does not know the result by heart, these steps can be written:



```
85 + 38 = 85 + 30 + 8
        = 115 + 8
        = 123
```

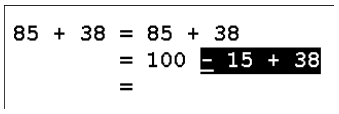
If a student does not know what to do, the small icons in the menu can be used. Marking the number 38 in 85 + 38 and clicking , give the result:



```
85 + 38 = 85 + 38
```

That means: 38 = 30 + 8 or 38 is only “2 away from 40.” The student can choose one of the two suggestions or quit. “ ” offers possible splits of 38 referring to “somewhat further than 30” or “somewhat before 40”. The second suggestion only will be presented if the distance from 38 to 40 is well known. In the same way, 85 can be split up into 80 + 5, 90 - 5, or 100 - 15.

Suppose the child chooses 100 - 15, then the task becomes:



```
85 + 38 = 85 + 38
        = 100 - 15 + 38
        =
```

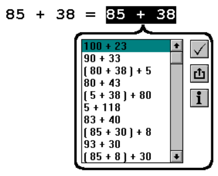
“”

The child might decide to reduce -15 + 38 first. Marking -15 + 38 and clicking “ ” gives a commutation. ( Marking 15 + 38 and clicking “  ” give an error message: “You should mark the ‘-’ before 15, too.”) And maybe the child reduces 38-15 to 23 by heart:

```
85 + 38 = 85 + 38
        = 100 - 15 + 38
        = 100 + 38 - 15
        = 100 + 23        “”
        = 123
```

The last step, 100 + 23 = 123, is easy.

A quite different way of thinking about 85 + 38 is thinking in terms of nice numbers. Marking 85 + 38 and clicking “ ” give a list of alternatives related to what a child has proved to know to the system.



```
85 + 38 = 85 + 38
100 + 23
90 + 33
( 80 + 38 ) + 5
80 + 43
( 5 + 38 ) + 80
5 + 118
83 + 40
( 85 + 30 ) + 8
93 + 30
( 85 + 8 ) + 30
```

Some elements of the list are:

- 85 is near to 100, the difference is 15, so we need 38 - 15 = 23 more: 100 + 23.
- 85 is near to 90, the difference is 5, so we have: 90 + 33.
- 85 is near to 80, the difference 5 can be neglected for a moment: first 80 + 38, and then +5. In short: (80 + 38) + 5.

Another related strategy is:

- 85 is near to 80, the difference 5 has to be added, well lets do it to 28, so we have: 80 + 43.

A nice one is, I have got to add something to 85. Well, I first will add 35, near to 38, but ending on a 5, so I have 120. And then: I know I need 3 more, because 35 is 3 less then 38.

This kind of strategy is very refined and complicated. It is not the calculus that is complex. It is the massive need of knowledge and experience needed for this kind of approximation. MathMirror only offers a suggestion to a child if needed knowledge is available, indeed. In the first example, for instance, the distance from 85 to 100 is needed, 38-15 is needed, and these efforts are only useful if 100 + 23 is easy. So, MathMirror has to evaluate whether these steps are available, before offering the suggestion 85 + 38 = 100 + 23.

Other tools in the upper toolbar will not be discussed here. These examples make it clear that the tools in the upper bar offer a part of the algebraic DPS of 85+38.

The mathematical idea of substitution is supported by marking and replacing. For example, in case of:

3 + 5 + 7 + 5, where a student can mark 5 + 5 and replace it by 10.

Or in case of:

85 + 38 = 80 + 5 + 38, where a student marked 85 and replaced it by 80+5.

This mark-and-replace function is supported by a detailed error control for grammar and logical mistakes.

At the lower bar on the right side we see other icons:

The two at the right side are not important now. They can be used to exit the worksheet in some way. The left three offer the opportunity to take 85+38 or a marked sub-problem to another representation:

arrow-language

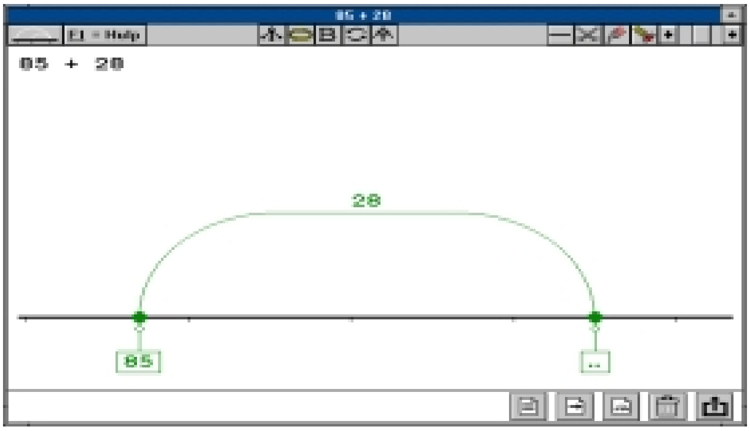
$$85 \xrightarrow{+\ 38}$$

$$85 \xrightarrow{+\ 10} 95 \xrightarrow{+\ 10} 105 \xrightarrow{+\ 10} 115 \xrightarrow{+\ 5} 120$$

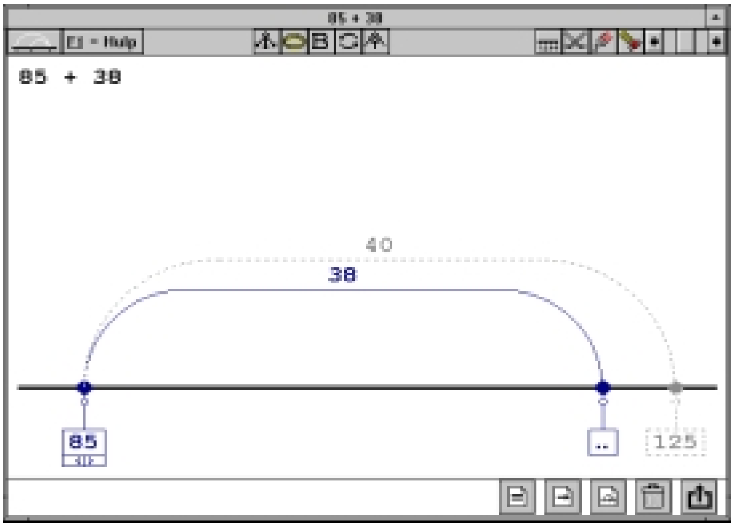
$$120 \xrightarrow{+\ 3} 123$$

In this = 95, 95 + 10 = 105 and so on. Adding tens and 5 and 3 is “stringing” steps until you are far enough. In the “arrow language worksheet” and in the numberline worksheet the same tool-icons are available as in the “=”- worksheet.

numberline



In the numberline worksheet, many techniques for drawing on the numberline are supported in order to express a strategy. For instance, someone can decide to approximate 85 + 38 by 85 + 40:

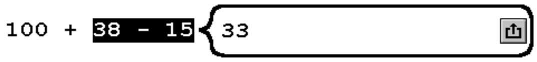


In this numberline representation, the approximation is very easy to understand: 125 is a little bit too far. The difference is 40 - 38.

or to go to another "="-worksheet.

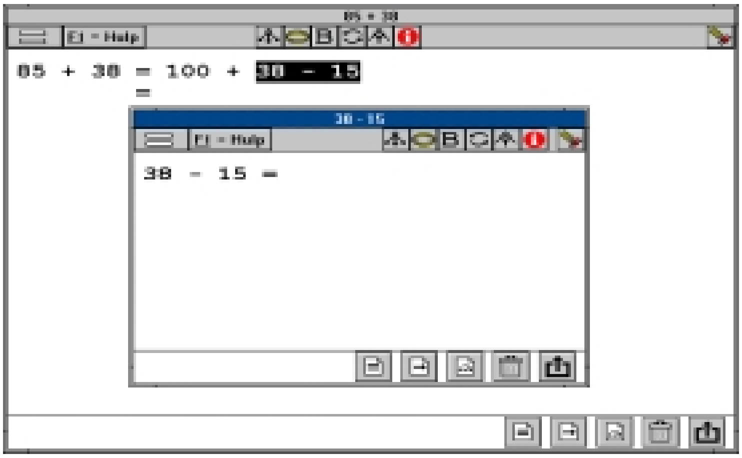
This option is automatically used in case of a wrong replacement:

Suppose someone has split 85 + 38 into 100 - 15 + 38, commutes -15 + 38, marks 38 - 15, and replaces it by 33:



100 + 38 - 15 33

then MathMirror reacts with:



85 + 38

85 + 38 = 100 + 38 - 15

=

38 - 15

38 - 15 =

The mistake is implicitly marked, and the wrong reduction is offered in a separate worksheet with all tools available.

MathMirror offers a counterpart of the student's DPS of 85 + 38 and its sub-problems. Tools are sensible for the objects to which they are applied , and the tools of the MMW (worksheet) follow the child’s solution of the problem and offer the child ideas that might belong to the child’s DPS. “Might” means that the current student has the knowledge involved in the strategy available.

## ARITHMETICUS: A STUDENTS MODEL

In this paragraph, how Arithmeticus produces solutions of exercises is sketched. After that, how Arithmeticus “learns” arithmetic is presented. And in the end, discussion includes how a zone of next development can be defined using Arithmeticus and how Arithmeticus can “reflect” on quality of solutions.

## Arithmeticus: Modelling Student's DPS's

In the description of MathMirror, 85 + 38 is used as an example. A student can generate a solution in several ways:

- The reduction can be known by heart.
- An algorithm that is (rather) well known can be remembered.
- The expression can approximated.
- The expression can be split up or rounded up one or more numbers.
- Algebraic rules can be used, like commutation and substitution.

A DPS of 85 + 38 is a set of this kind of associations. Solving 85 + 38 can be understood as generating a chain of associations. Reminding the DPS is changing with the state of the solution. A student's reduction can be modelled as a formal representation of that chain of associations: a sequence of mathematical transformations. This is a DPS-based solution.

Sometimes it is rather difficult to understand the strategy that a child has followed. What is -for example- the strategy behind: 45 + 19 = 60 + 4 = 64? Reasonable interpretations are:

**Table 1**
Some Solutions for 45+19

| | | | | |
|---|---|---|---|---|
| 1. | A. | B. | C. | D. |
| 2. | **45+19** | **45+19** | **45+19** | **45+19** |
| 3. | 45+10+9 | (45-1) + (19+1) | 45+20-1 | 50+14 |
| 4. | 45+10 | 44+20 | 45-1 | **60+4** |
| 5. | 55 | 40+4+20 | 44 | **64** |
| 6. | 55+9 | 40+20 | 44+20 | |
| 7. | 55+5+4 | 60 | 40+4+20 | |
| 8. | 55+5 | **60+4** | 40+20 | |
| 9. | 60 | **64** | 60 | |
| 10. | **60+4** | | **60+4** | |
| 11. | **64** | | **64** | |

- Interpretation A is very common: 19 has been split up.
- Interpretation B is based on the rule: $a + b = (a - c) + (b + c)$. That rule is applied in order to change 19 into 20. That's easier for addition.
- Interpretation C is based on the idea that 19 is just near to 20.
- Interpretation D is also based on the rule $a + b = (a - c) + (b + c)$. First, 45 is rounded to 50, and 10 of 14 has been moved towards 50. Then 60 + 4 = 64 remains.

Suppose we know that this student knows reductions like 45 + 10=55 and 45 + 20 = 65. (These reductions are based on, for example, “counting forward in steps of 10, from any number.”) Then, the interpretations B, C, and D are not very likely:

- In interpretation B, steps 5 - 8 seem superfluous.
- In interpretation C, steps 7 - 10 seem redundant.
- In interpretation D, 50 + 14 seems to be known by heart (but for commutation), so 60 + 4 seems to be superfluous.

Nobody gives a fully detailed reduction of an arithmetical expression; only a few steps will be given which are necessary as an aid to memory. Therefore, most solutions of arithmetical reduction tasks cannot be completely understood without any knowledge of what a student might know already. A good teacher who sees a student’s solution like 45 + 19 = 60 + 4 = 64, thinks, “What algorithms related to my instruction, give solution’s steps that include 45 + 19 = 60 + 4 = 64?” A more refined question might be: “Given the facts and algorithms a student knows, what algorithms using that knowledge give solution steps, that include 45 + 19 = 60 + 4 = 64?”

For the interpretation of a student’s reductions, we need “the reductions a student is able to make.” We could collect reductions or reduction strategies from schoolwork. The problem is there are thousands of possible reductions. So, we have a representation problem and a collecting problem. And if we have that collection of reductions, we still have the problem of how to match those reductions with actual students' work. A problem becomes clear in solution A:

| | | |
|---|---|---|
| 1. | A. | A’. |
| 2. | 45+19 | 45+19 |
| 3. | 45+10+9 | 45+10+9 |
| 4. | 45+10 | 45+10 |
| 5. | 55 | 55 |
| 6. | 55+9 | 55+9 |
| 7. | **55+5+4** | |
| 8. | **55+5** | |
| 9. | **60** | |
| 10. | **60+4** | |
| 11. | 64 | 64 |

Steps 7-10 can be omitted, when a student can do 55 + 9 very fast and by heart. So algorithm A’ is shorter than algorithm A. But it might be possible that a child writes:

(i) 45 + 19 = 55 + 9 = 60 + 4 = 64.

in place of the shorter:

(ii) 45 + 19 = 55 + 9 = 64.
(ii) fits to algorithm A’, (i) is a regression in algorithm A’ or it is fitting to algorithm A.
If a student uses often A’, I prefer to say (i) is a regression.

This example shows we need a dynamic set of students' facts and algorithms changing with students' learning. In the next paragraph, a DPS-based model of students' facts and algorithms is shown that permits complete interpretation of students' solutions and can record progression and regression in individual algorithms.

## ARITHMETICUS: GENERATION OF ALGORITHMS A STUDENT IS ABLE TO

Arithmeticus produces reductions of any arithmetical expression by combining transformations. Arithmeticus is simulating DPS-based solutions by chaining elementary transformations (rules).

Arithmeticus is an inference engine with static and dynamic transformation rules. There are two kinds of static rules: algebraic rules and rules that change representations. Dynamic rules are facts and algorithms that are available for a current student.

A fact is a memorised relation like 3 + 4 = 7. An algorithm is a reduction of an expression in a number of steps, as in Table 1. Algorithms are represented as a recursive, structured sequence of specified transformations. Algorithms are not related to an exercise. Arithmeticus can test whether an algorithm can be applied to an expression or not. In the students’ model, algorithms can be qualified as strategy, routine, or automatism, depending on the speed and correctness of student’s solutions.

In Arithmeticus concepts like “easy” and “is in the neighbourhood of” have been defined. These concepts are used in strategies like: 6+7 is in the neighbourhood of 6 + 6, or 63 - 48 is nearly 63 - 50.

Given an expression (as “85 + 38”), Arithmeticus can produce algorithms that a current student is supposed to be able to produce. This way Arithmeticus can produce hundreds of different algorithms to reduce “85 + 38.”

An algorithm is a sequence of static and dynamic transformations. Dynamic transformations are facts and algorithms that a current student learned before. Each time Arithmeticus determines an algorithm that is new to the current student, that new algorithm is added to the student’s knowledge base. The same is done with facts. So, the set of dynamic rules is growing. In other words, the DPS-model of a current student is growing.

Arithmeticus, with its dynamic and static rules, gives a generic model of the algorithms that a student is able to do. New student’s reductions are interpreted in that generic model.

Student’s reductions are interpreted in two or more steps:

1. Ask Arithmeticus which reductions of a formula are possible for the current student.
2. Match the Arithmeticus solutions with the current student’s reductions.

Sometimes, there is ambiguity in the matching process. In that case, a dialogue is started in which the current student can explain his algorithm in more detail.

This matching covers the case of regression. If the matching process expects a solution by heart, but meets some steps belonging to what has to be done by heart, a regression is established.

## ARITHMETICUS: SUPPORTING REFLECTION

A very interesting feature of Arithmeticus is the possibility to compare algorithms. Length, complexity, numbers of integers, level of practice, and other properties can be compared. On the basis of those properties, the quality of algorithms can be compared. Arithmeticus can see whether an algorithm is used with less expressed steps or faster in time. Arithmeticus can compare a current algorithm with other earlier-used students' algorithms for the same type of reductions. And Arithmeticus can compare a solution with the algorithms that a student is able to calculate. So, comments can be given like:

“This is a nice new solution of yours.”

“Compliments, you proved you can do this algorithm by heart.”

“This is a good solution, but you know a faster one.”

“Please, do this reduction again and use the DPS-icons in MathMirror.”

So Arithmeticus can produce comments for a student in order to make the studen reflect on the reduction.

## ARITHMETICUS: GENERIC PLANNING OF EXERCISES

One of the most exciting aspects of Arithmeticus is the possibility of planning exercises by calculating a zone of next development. Given a set of goal exercise types and a current state of Arithmeticus, it is possible to calculate which type of exercises can be done on what kind of level by the current student. A planning algorithm chooses, from this zone of next development, sets of exercises and individual exercises to be used whether a student has requested knowledge available or not.

This way of planning is very different from the usual curriculum definition. There is no sequence of exercises that defines the curriculum. There is only a set of goal exercise types. By implication, the learning path of individual students can differ. The learning paths are not redefined but are generically defined by the set of goal-exercises, the current state of Arithmeticus for the current student, and the planning algorithm that tells which exercises from the zone of next development will be chosen first.

Up to now, there are two goal exercise types: formula types and algorithm types. Formula types are, for example, addition exercises with operands from a certain domain. Algorithm types are sets of formulae that have to be solved by the same algorithm.

## IMPLICATIONS OF ARITHMETICUS: GENERIC STUDENT- VERSUS EXPERT-MODELLING

Arithmeticus and MathMirror offer a learning environment in which a child can develop personal strategies. There are no predefined expert solutions. A solution can be good or nicely related to a child's learning history. The more strategies or facts a child learns, the smarter Arithmeticus will be and the more smart solutions Arithmeticus will expect of a student. In some sense Arithmeticus learns the arithmetic of a child and reflects all previous work in the MMW MathMirror in order to have a child recognize the DPS of a current problem. What is smart for Arithmeticus is smart for the student, and the other way round. What Arithmeticus can do is combine all a child knows and elementary mathematical transformations so it can create "what a student is supposed to be able to do." Arithmeticus and MathMirror can feed a student's creative thinking by offering a student a meaningful but formal based dialogue in which a child can enrich personal thinking.

## References

Klep, J. (1992). Learning elementary mathematics: A discussion of MicroWorlds. In P. Kommers et al.: *Cognitive tools for Learning,* NATO ASI Series, Vol. F81, Berlin Heidelberg: Springer-Verlag.

Newell, A., & Simon H. ( 1972). *Human problem solving*. Englewood Cliffs, N.J.: Prentice Hall.

Petrushin, V.A., & Sinitsa K.M. (1990) Learner's knowledge adaptive testing based on the Bayesian approach to decision making (in Russian). *Computerized technologies in education* (pp. 71-76). Kiev: Glushkov Inst. for Cybernetics.

Varela, F.J. (1990). *Kognitionswissenschaft-Kognitionstechnik: Eine skizze aktueller Perspektiven*. Suhrkamp, Frankfurt am Main.

## Note

1. A school and a home version of MathMirror ("Plato and his MathMirror") was edited by Zwijsen, Tilburg, Netherlands in September/December, 1997.