OPEN ACCESS
OPUS AUGSBURG

Universität Augsburg
Universitätsbibliothek

A technology-oriented algebra curriculum

Reinhard Oldenburg

Angaben zur Veröffentlichung / Publication details:

Oldenburg, Reinhard. 2023. “A technology-oriented algebra curriculum.” In *Symposium proceedings Innovative Teaching Practices: held at The Queen's College Oxford University, UK, August 14-18, 2023*, edited by Janina Morska and Alan Rogerson, 168–73. Münster: WTM - Verlag für Wissenschaftliche Texte und Medien.
https://doi.org/10.37626/ga9783959872508.0.32.

Nutzungsbedingungen / Terms of use:

licgercopyright

Dieses Dokument wird unter folgenden Bedingungen zur Verfügung gestellt: / This document is made available under these conditions:
Deutsches Urheberrecht
Weitere Informationen finden Sie unter: / For more information see:
https://www.uni-augsburg.de/de/organisation/bibliothek/publizieren-zitieren-archivieren/publiz/

A Technology-oriented Algebra Curriculum

Reinhard Oldenburg, reinhard.oldenburg@math.uni-augsburg.de
Augsburg University

Abstract

The traditional algebra curriculum of (lower) secondary schools is (at least in Germany) plagued by several severe deficits: 1) Students are hardly interested and motivated to learn algebra. 2) The semantics of algebra as discussed in didactics is unnecessary complex and partially incompatible with algebra in upper secondary schools and at the university level. 3) The algebra taught is partly incompatible with the algebra of programming languages and computer algebra systems. This paper gives a short overview of a technology-oriented curriculum that shall overcome these issues and it presents some of the technological tools that have been developed to provide a consistent and motivating learning experience of algebra. An essential feature is that technology is not just a medium, but a tool and benchmark for consistency.

Introduction

The learning and teaching of algebra has always been in the focus of mathematics education (e.g., Drijvers, 2011). However, many studies document that the learning outcome often lacks behind expectations (e.g., Hodgen et al., 2009). While it seems that most activities undertaken to improve the situation aim at the methodology of teaching, there has recently been a renewed interest in changing the curriculum, see e.g., Wolfram (2020) and Strømskag & Chevallard (2022). This prompted me to condense my own experience with algebra education into a curriculum (Oldenburg, 2023). The present paper is partly an introduction to some key ideas if this curriculum, and partly an elaboration of some ideas from another perspective.

The problems with the current curriculum

Students live in a world that is dominated by artefacts from computer science (digitalized sounds, images, videos, (secure) communication, planning, artificial intelligence etc) and processes governed by natural sciences (global warming, spreading of deceases, charging of batteries, measuring all kinds of quantities). However, it is not at all obvious that mathematics and especially algebra has a role in all that, when in schoolbooks algebra is e.g., introduced as means to find out how many people can sit at a row of tables or to find out how old a mother is who was five years ago four times as old as her mother. I don't suggest omitting all these classical riddles and instead train fifth graders to become experts in quantum computers, but I believe that relevance of algebra can become clearer if the curriculum emphasizes that algebraic thinking is a special case of computational thinking (CT, see Wing (2006)) and that this underpins much of thinking in natural sciences and philosophy.

Oldenburg, Reinhard (2023). A Technology-oriented Algebra Curriculum, in Morska, Janina & Rogerson, Alan (Eds) Innovative Teaching Practices, Proceedings of the First International Symposium of The Mathematics Education for the Future Project, Oxford University. Aug 14-18 2023 (pp. 168-173). Münster: WTM. https://doi.org/10.37626/GA9783959872508.0.32

Thinking algebra with CT in mind does not only show motivating examples of applications but it focusses on syntax, semantics, and pragmatics. Algebraic expressions are means to communicate calculation procedures between humans and between humans and machines. A syntax that is unambiguous for machines might be a bit more verbose than traditional syntax, but it proves itself to be clear. The same holds true for the semantics. While didactical considerations of variables are obscured by many opaque concepts (such has changing numbers or variables representing different numbers simultaneously), conceptualising variables in a way that is compatible with programming languages and computer algebra systems guarantees consistency and eases use of such tools. Finally, pragmatics manifests itself e.g., in the role a variable is used in (see. e.g., Sajaniemi, 2002). Thus, application should not just be an add-on in the end but should be integral to the process of developing algebra.

Opaque concepts mentioned above are not only a problem for the teaching about variables. Many schoolbooks simply define an expression as a "sensible" string of symbols used for calculations (to quote from a German textbook: "sinnvoller Rechenausdruck"). Everything else students have to pick up inductively from examples. There is hardly any explanation why x+/2 is not sensible. It is much more precise to build up expressions from the recursive definition that an expression is either a number, or a symbol (variable) or an expression in parentheses, or a sum of two expression etc. This allows to build expressions from the ground up (in fact, to enumerate all of them recursively) and to analyse give expressions into this structure. It links them to expression trees and explains the operation of substitution which is rather important for the learning of algebra (Oldenburg, 2009).

Another opaque concept is that of equivalence of expressions or of equations. E.g., the latter is sometimes defined to mean that two equations have the same set of solutions (which makes $x-1=0$ and $y-1=0$ equivalent, although they have different variables), or in other books by the ability to transform them into each other by application of a set of given rules (which makes $\sin x=2$ and $x^2=-1$ not equivalent over R although they have the same solution set). A sensible definition, of course, is possible along the lines of mathematical logic: One defines an assignment to be a map, that assigns values to variables, such as $x\to 2, y\to 1\}$ such that an assignment turns an expression into a number and an equation into a truth value, and calls to expressions (or equations) equivalent, if they have the same (truth) value under all assignments. This simple definition clarifies the problems just described. Moreover, in CAS one can work with such assignments, e.g. in Mathematica the input x+2*y/.{x->1,y->5} yields 11. This technological example moreover shows that this kind of semantics is compatible with the semantics that comes from Language play (Wittgenstein, 1953), i.e., the transformation of expressions. The interested reader should consult the description of operational and denotational semantics in theoretical computer science, e.g., in the book by Stump (2013).

Overview of the suggested curriculum

The main steps of the suggested curriculum are:

Grade 7: Expressions with variables are introduced in the context of image processing (see below) and applied in simple game programming.
The rationale behind this is that most traditional contexts don't really exhibit the need for generalization well. Different incarnations of programming do this, because one has to specify a recipe in advance (at programming time) before actually calculating (run time).

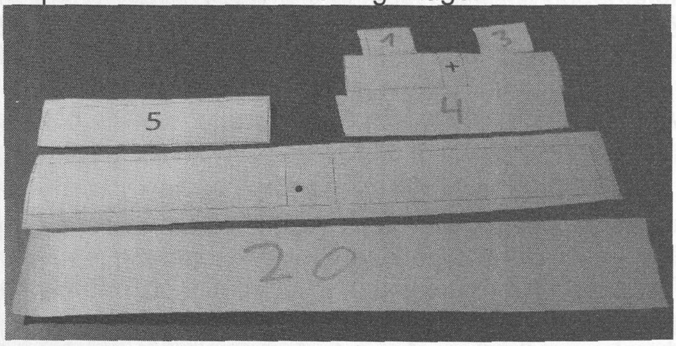
Grade 8: Equations are introduced in applications and with an app (see below) Equations are syntactical objects, just as expressions, hence they fit nicely in the development by extending ordinary expressions that yields numbers to more general expressions that may yield truth values.

Grade 9: Relations and logic: Solution sets are explored systematically and equations coming from geometry (Pythagoras) are explored.

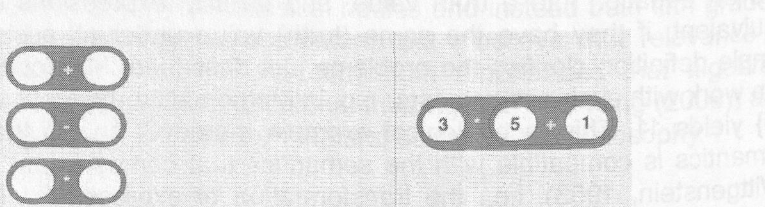
Grade 10: Functions are introduced and used as a means of abstractions and capsulating calculation processes.

A closer look: Tools for expressions

The recursive nature of expressions can be experienced from classical activities, e.g., decomposing a given number into a calculation by means of given paper strips as shown in the following image:

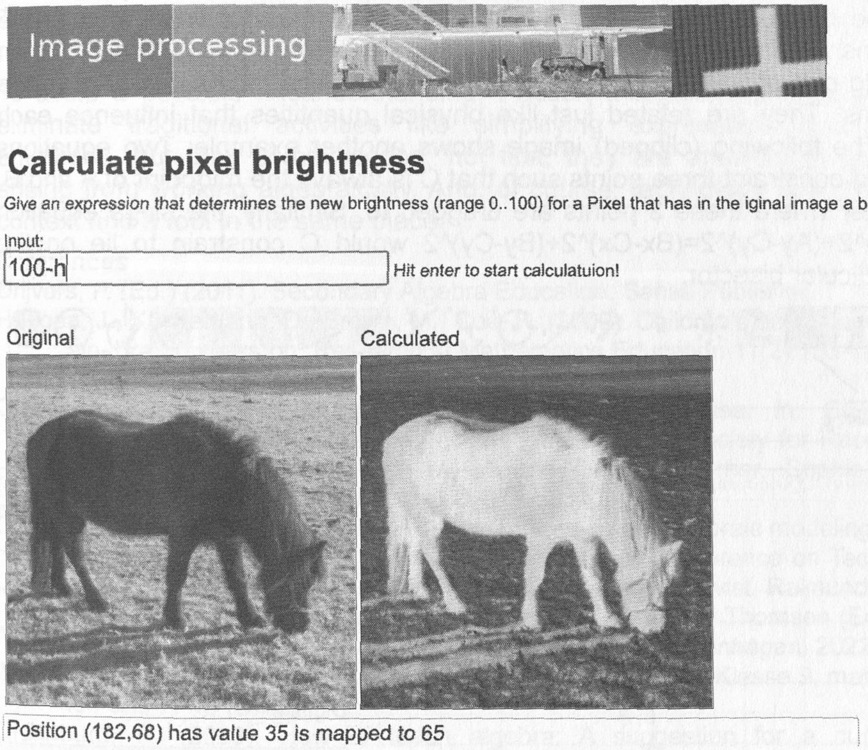


However, the dynamic insertion of operations and operands into placeholders can also be exercised with much more fun when using the Scratch programming language (https://scratch.mit.edu/) which is easy to use even for younger students (see e.g. https://www.ucl.ac.uk/ioe/research/projects/ucl-scratchmaths) and that has pallets from which operations can be taken and combined:



Students should exercise such calculations with Scratch as well as writing the expressions in their notebooks. Of course, this is ideally part of some bigger project like implementing a simple game.

After having some experience with such numerical expressions, variables can be introduced in the context of image processing: The web page https://myweb.rz.uni-augsburg.de/~oldenbre/webBVen/index.html explains a bit about the background and provides access to several applets: https://myweb.rz.uni-augsburg.de/~oldenbre/webBVen/onepix-webcam.html. This applet allows to modify an image by giving an expression describing how the old brightness value (a number from 0 to 100) is transformed into the new brightness. An example is shown below.



Calculate pixel brightness

Give an expression that determines the new brightness (range 0..100) for a Pixel that has in the iginal image a brightness **h**

Input:

100-h

Hit enter to start calculatuion!

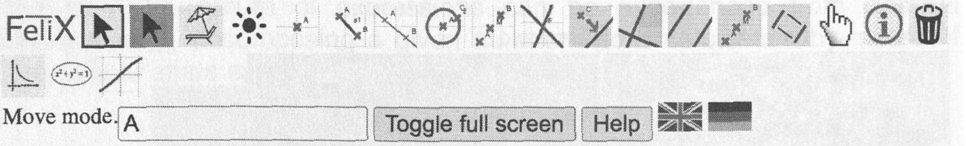
Position (182,68) has value 35 is mapped to 65

Working with this example give students a direct understanding of expressions as "little programs" that describe calculations, and of variables as references that refer to different values in different applications of the same expression.

A closer look: Tools for equations

Many tools (spreadsheets, dynamic geometry systems, function plotters) support functional thinking, i.e., a directional model of thinking where some independent variable is the input (a cell in Excel where a number is entered, a moveable basic point in Geogebra, the input of a function in any calculator) that determines a result or output (the content of a cell calculated by a formula, the position of dependent (constructed) geometric objects, the result of a calculation). Besides such directional *covariation*, also directionless covariation occurs: Quantities can be tight together by *relations* and this requires relational thinking. Geogebra allows e.g., to enter implicit equations like $x^2+\frac{y^2}{4}=1$ or inequations and logical combinations of them like $x^2+y^2<1 \wedge x \cdot y>0$ to be plotted. Variables in such relations cannot be separated into dependent and independent. Activities as plotting such region and modelling regions by algebra are important and rewarding (see Oldenburg, 2022a), but as they show the whole solution set at once they are best suited for more advanced learners. To start relational thinking I suggest use a tool like FeliX (see Oldenburg (2022) & https://myweb.rz.uni-augsburg.de/~oldenbre/jsfelix/F2d/jxfelix.html).

It allows to restrict the freedom of points by equations. In the simplest case, e.g., one may start from two points A and B, glue them to the x-axis by the equations Ay=0 and By=0, and relate their x-position by Ax+2*Bx=10. Then changing one of them, changes the other as well, and they move in opposite directions. They are related just like physical quantities that influence each other. The following (clipped) image shows another example: Two equations are used constraint three points such that C is always the midpoint of A and B, no matter where these 3 points are dragged to. Similarly, the singe equation (Ax-Cx)^2+(Ay-Cy)^2=(Bx-Cx)^2+(By-Cy)^2 would C constrain to lie on the perpendicular bisector.



Object	Va
A	[1.0,2.0]
B	[7.0,8.0]
C	[4.0,5.0]

Equation/expression

Ax+Bx=(2*Cx)

Ay+By=(2*Cy)

Students max discover that the equation 2*Ax+1=5 or 4*Ax+2=10 or Ax=2 all have the same effect, thus getting a feeling for the equivalence of equations. When a point is restricted to a one-dimensional orbit, this orbit can be plotted and its equation can be displayed if it can be calculated by Gröbner basis elimination techniques.

Via inequalities one can also determine that two points must not come too close to each other, so that one can push the other away: (Ax-Bx)^2+(Ay-By)^2>1. Thereby both points are completely equal, there is no distinction in dependent and independent - exactly therefore the relational thinking is promoted.

Conclusion

The examples above have should give an idea of what should be possible when simplifying the algebraic language used in schools and align it to algebraic language use in computer science. Of, course, implementing such a curriculum is an almost impossible enterprise in these days were. However, all the ingredients have been investigated in isolated uses in classroom. Therefore, some experience supports the hypothesis that it is possible to recast the curriculum in this form.

To avoid a possible misunderstanding, I assert that this approach does not eliminate traditional activities like simplifying expressions and solving equations, but they come second, not first, they are answers to questions arising from working algebraically. And, of course, technology is a motivating context and a tool in the same place.

References

Drijvers, P. (Ed.) (2011). Secondary Algebra Education. Sense Publisher.

Hodgen, J., Küchemann, D., Brown, M., Coe, R. (2009). Children's understandings of algebra 30 years on. Research in Mathematics Education 11(2):193-194, DOI: 10.1080/14794800903063653

Oldenburg, R. (2009). Structure of algebraic competencies. In: CERME 6: Proceedings of the Sixth Congress of the European Society for Research in Mathematics Education; ed. by Viviane Durand-Guerrier, Sophie Soury-Lavergne and Ferdinando Arzarello, 579-588.

Oldenburg, R. (2022). Relational thinking supported by an algebraic modeling tool on the web. in: Proceedings of the 15th International Conference on Technology in Mathematics Teaching (ICTMT 15). Uffe Thomas Jankvist, Raimundo Elicer, Alison Cark-Wilson, Hans-Georg Weigand and Marianne Thomsen (Eds.), 89-96 (Danish School of Education, Aarhus University: Copenhagen, 2022)

Oldenburg, R. (2022a) Logisch? - Logisch! Unterrichtsideen ab Klasse 9, mathematik lehren, 44-46 (2022)

Oldenburg, R. (2023). How to teach algebra: A suggestion for a curriculum. https://doi.org/10.35542/osf.io/cf5v3

Sajaniemi, J. (2002). An Empirical Analysis of Roles of Variables in Novice-Level Procedural Programs. Proceedings of IEEE 2002 Symposia on Human Centric Computing Languages and Environments (HCC'02), Arlington, VA. IEEE Computer Society, 37-39.

Strømskag, H., Chevallard, Y. (2022). Elementary algebra as a modelling tool: A plea for a new curriculum. Recherches en didactique des mathématiques. Volume 42 (3).

Stump, A. (2013). Programming Language Foundations. Wiley: Hoboken.

Wing, J. (2006). Computational Thinking. Communications of the ACM, Vol. 49, No. 3

Wittgenstein, L. (1953). Philosophical Investigations. Blackwell. Wolfram, C. (2020). The Math(s) Fix.