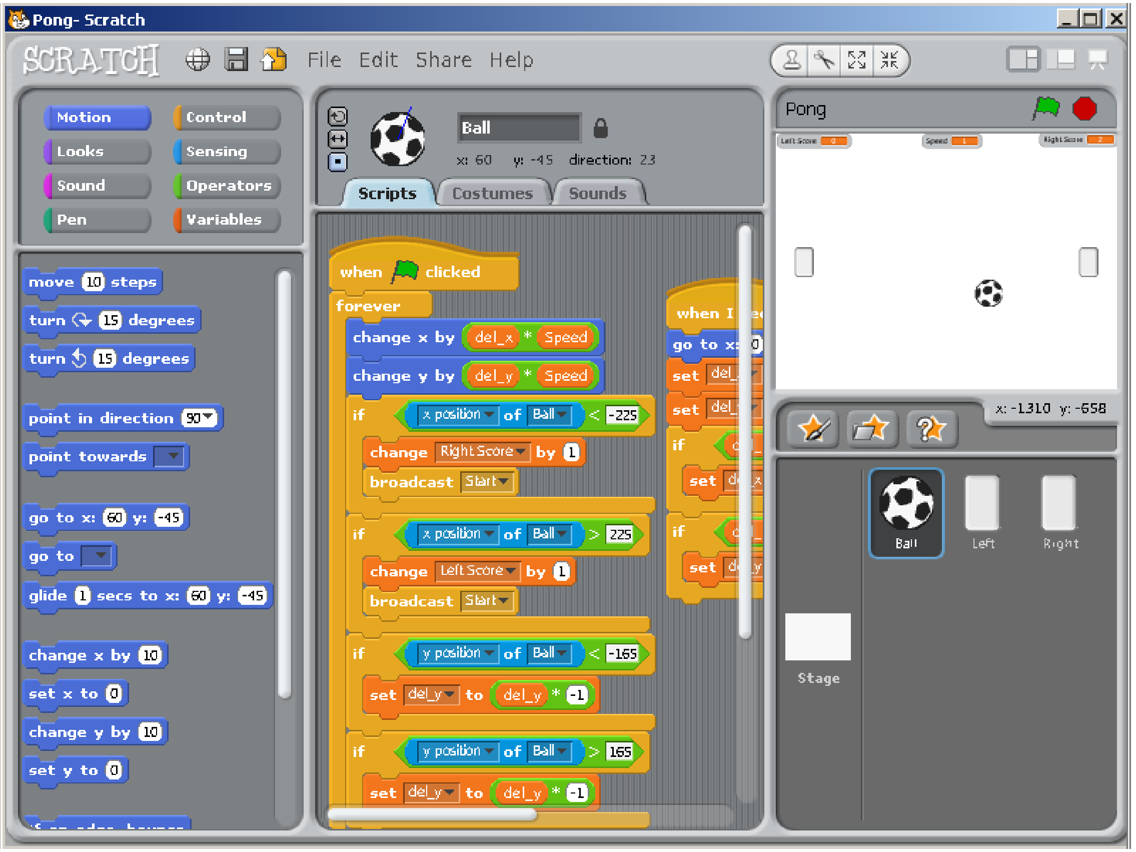
THEY ALREADY KNOW THE SYNTAX!

The Case for Spreadsheets in Programming Education

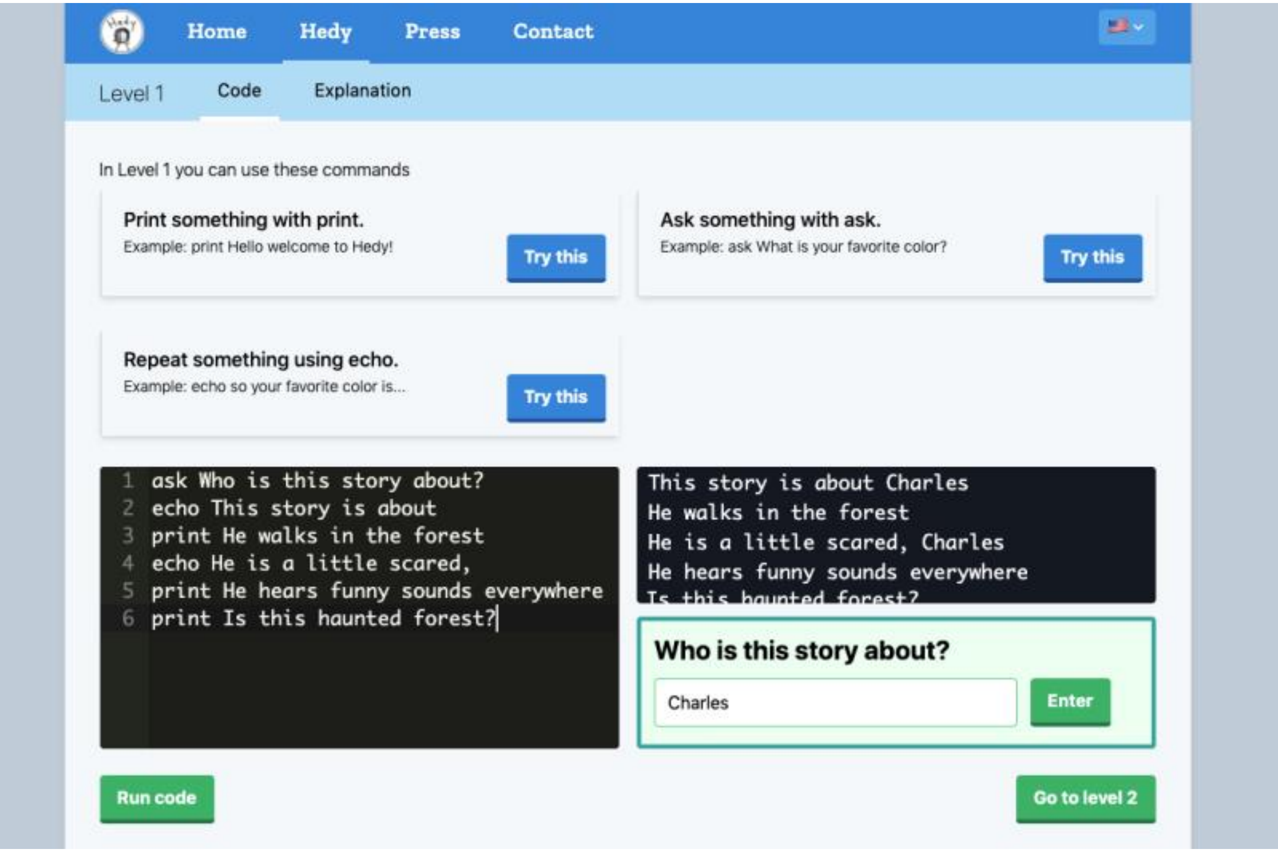
Learning Syntax is Known to be an Obstacle in Programming Education

Responses: Block Coding (replace syntax with shapes) & Gradual Languages (relaxed syntax rules)

Scratch [MIT Media Lab]

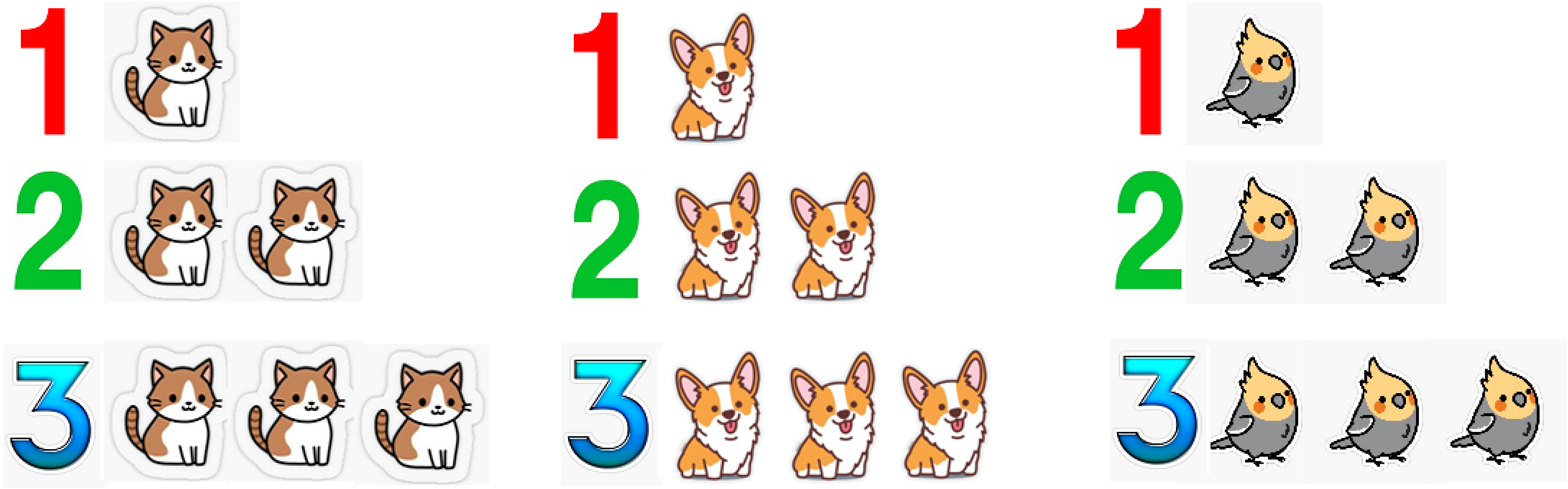


Hedy [Felienne Hermans, Leiden University]

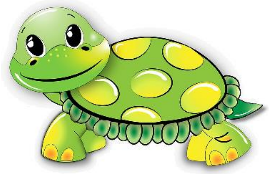


Back to basics - let’s reminisce our early computing education:

Math Abstractions … Baby Steps



I am **one** baby syntax turtle ➔



Positional numeral system, operations, operator precedence (!) …

Math Abstractions …

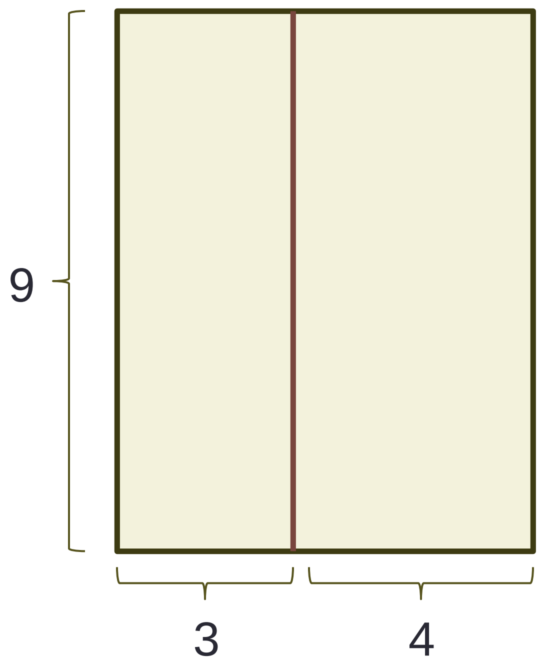
11 +
31

42

6 x 7 = 42

2 / 7 + 3 / 7 = 5 / 7

1,000,000 – 997 = 999,003



9 x (3 + 4) = 63

The turtles … they keep … multiplying (!) – here we are, at the end of middle school and early high school:

Math Abstractions

$$E = mc^2$$

$$y = m \cdot x + q$$

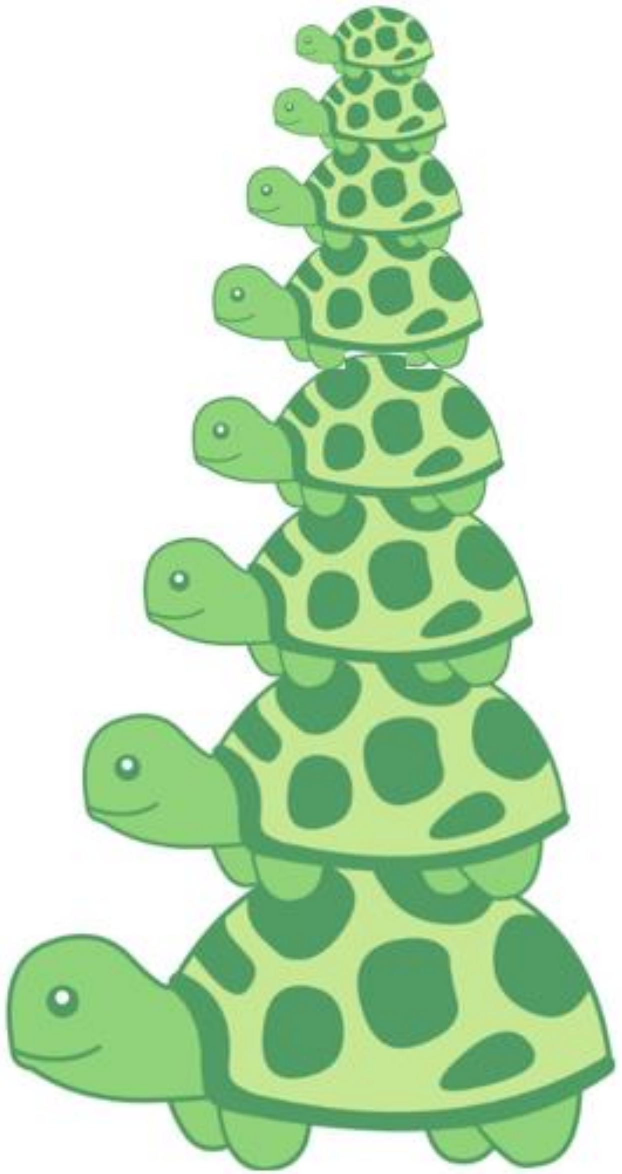
$$sin^2 x + cos^2 x = 1$$

$$\frac{sin\ x}{cos\ x} = tan\ x$$

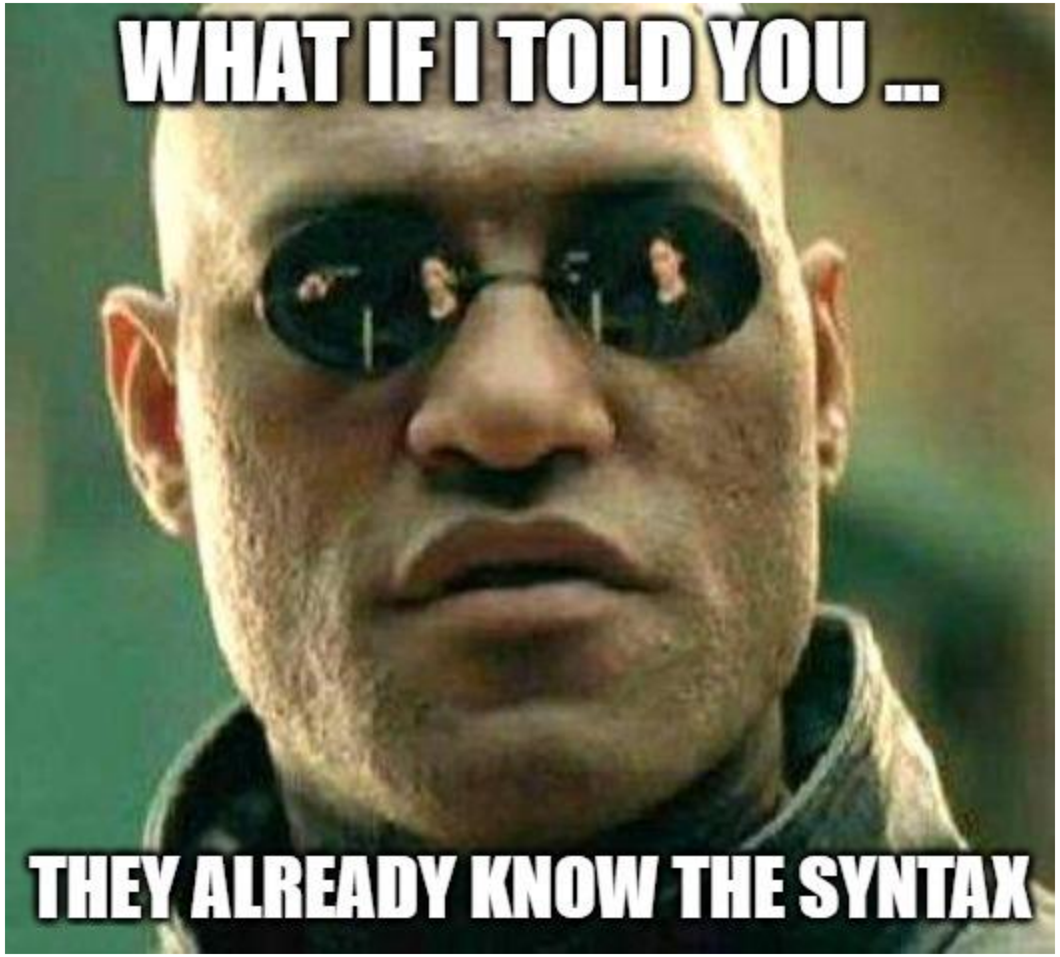
$$(a - b)(a + b) = a^2 - b^2$$

$$D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

$$F = G\ \frac{m_1\ m_2}{d^2}$$



How much ***more*** “syntax” must we feed students to start learning programming?



Note: G.A.M.N. stands for Generally accepted Math notation

Students Already Know (most of) The Syntax!

	Raise to Power	Multiplication	Division	Modulo	Addition	Subtraction	String Concatenation
MATH	x^y	xy | $x * y$	$x \div y$ | x / y	$x \bmod y$	$x + y$	$x - y$	xy | $x \cdot y$
FORTRAN	x ** y	x * y	x / y	mod(x, y)	x + y	x – y	x // y
LISP	(pow x y)	(* x y)	(/ x y)	(mod x y)	(+ x y)	(– x y)	(concatenate x y)
C / C++	pow(x, y)	x * y	x / y	x % y	x + y	x – y	x + y
Haskell	x ^ y | x ** y	x * y	x / y	mod x y	x + y	x – y	x ++ y
Python	x ** y	x * y	x / y	x % y	x + y	x – y	x + y
Java	Math.pow(x, y)	x * y	x / y	x % y	x + y	x – y	x + y
JavaScript	x ** y	x * y	x / y	x % y	x + y	x – y	x + y
OCaml	x ** y	x * y | x *. y	x / y | x /. y	x mod y	x + y | x +. y	x – y | x –. y	x ^ y
MS-Excel	x ^ y	x * y	x / y	mod(x, y)	x + y	x – y	x & y

.:. OK, sure, but those are ***just expressions***. That's not programming, right? I mean, expressions aren't enough, right? ➔

A User-Centred Approach to Functions in Excel
30th June 2003

Simon Peyton Jones
Microsoft Research

Alan Blackwell
Cambridge University

Margaret Burnett
Oregon State University

"It may seem odd to describe a spreadsheet as a programming language. Indeed, one of the great merits of spreadsheets is that users need not think of themselves as doing "programming", let alone functional programming — rather, they simply "write formulae" or "build a model". However, one can imagine printing the cells of a spreadsheet in textual form, like this:

A1 = 3

A2 = A1-32

A3 = A2 * 5/9

and then it plainly is a (functional) program."

"***just expressions***": yes, to program with spreadsheets all the syntax you need to know is that of expressions!

Critique of the Traditional Spreadsheet Core

- **Lack of functional abstraction**
  - Considerable research work has been done on this
  - December 3rd, 2020: Microsoft Research announced LAMBDA

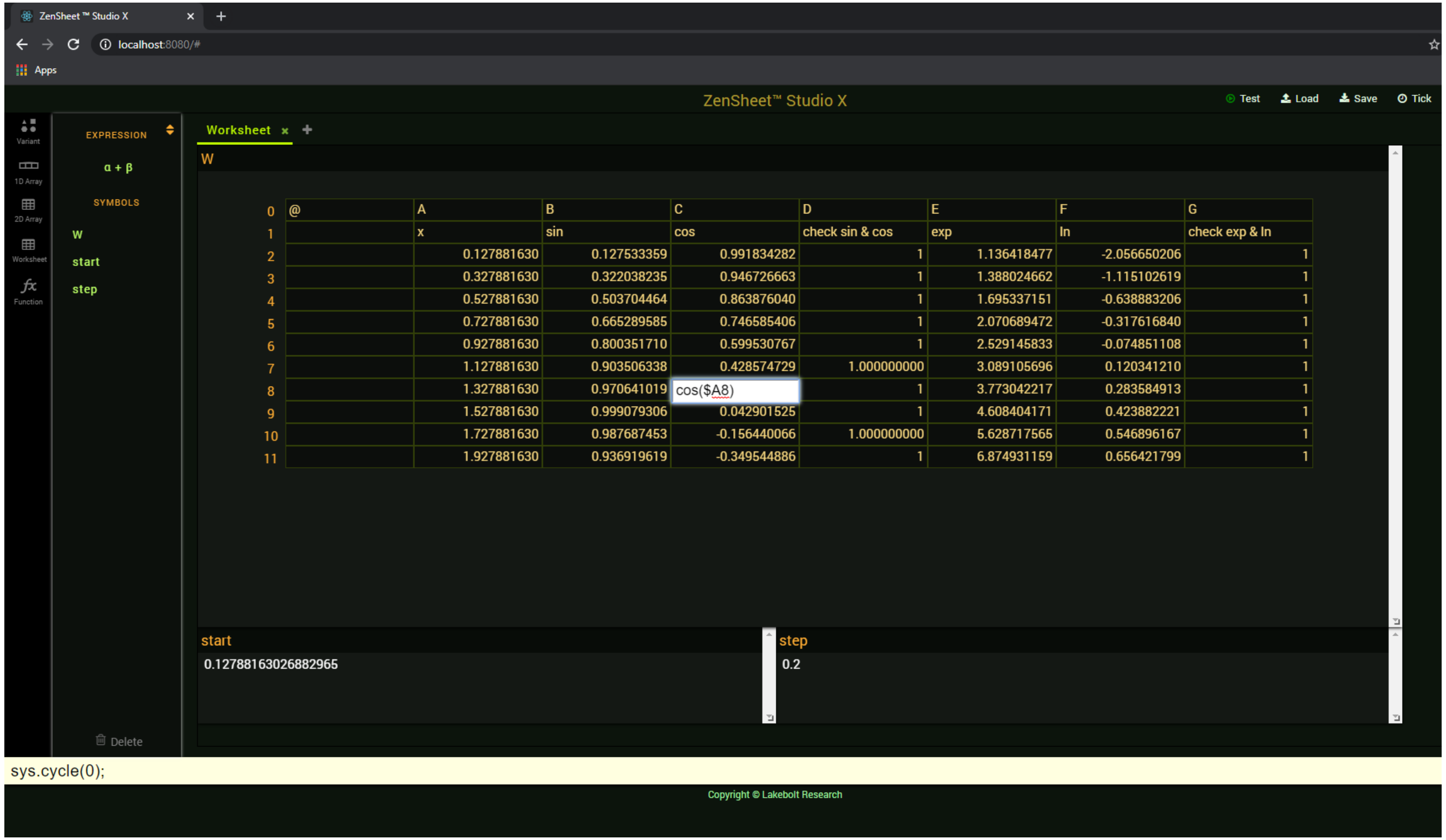
- **Overly simplistic type system**
  - All top-level variables must be a worksheet
  - Worksheets are non-composable cell containers
  - All cells are unitype and must be referenced via coordinates
  - **A1 notation should be considered harmful**

- **Entanglement of model and visualization**
  - Worksheets are the only true variables of the core
    - They are containers that hold state, which includes unreduced expressions
  - Worksheets are also the primary element of the presentation
    - They play an important role as UI layout managers

A language-centric redesign of spreadsheets has been shown to work well

ZenSheet / Lilly

ZenSheet supports composable data structures and functional abstraction.
2D arrays can be used as worksheets: it truly generalizes spreadsheets!



Types

ZT.1) T → **null** | **error** | **bool** | **number** | **string**

ZT.2) T → **fun**(T, …, T) => T

ZT.3) T → **array**[, …,] => T

ZT.4) T → **struct**(T, …, T)

ZT.5) T → **lazy** T

ZT.6) T → **var**

ZT.7) T → <symbol>

Expressions

XLS.1) E → ? | <error> | **true** | **false** | <number> | <string>

XLS.3) E → < A1 > | <symbol> ! < A1 >

ZSE.1) E → <symbol>

ZSE.2) E → λ(T <symbol>, …, T <symbol>) -> E

ZSE.3) E → E(E, …, E)

ZSE.4) E → (E, …, E)

ZSE.5) E → [E, …, E]

ZSE.6) E → E[E, …, E]

ZSE.7) E → E:E

ZSE.8) E → E..E

ZSE.9) E → ‘E’

Actions

ZSA.1) A → **type** <symbol> = T;

ZSA.2) A → T <symbol> := E;

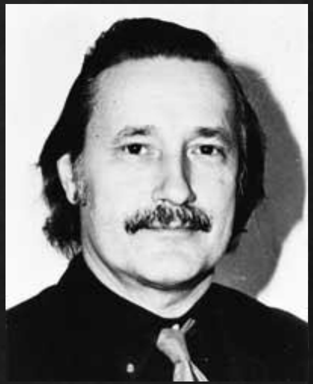
ZSA.3) A → E := E;

Listing 3: abstract syntax of Lilly

But to achieve this result we had to extend a traditional model of computing

Extending Christopher Strachey's Model

Traditional



- LVALUE
- CVALUE
- RVALUE

Assume φ has l-value, then:

- RVALUE(φ) = CVALUE(LVALUE(φ))

Reassignment:

- <lhs> := <rhs>;
    - Post: CVALUE$_p$(<lhs>) = RVALUE$_b$(<rhs>)

Lilly

- LVALUE
- CVALUE
- RVALUE

Assume φ has l-value, then:

- RVALUE(φ) = RVALUE(CVALUE(LVALUE(φ)))

Reassignment:

- <lhs> := <rhs>;
    - Post: CVALUE$_p$(<lhs>) = RVALUE$_b$(<rhs>)

Lilly natively supports a function *formula*(φ) that returns CVALUE(φ) without computing RVALUE(φ)

Experience Report: Lilly in Action

Our Experience Report:
Complementing Programming Education with Lilly and ZenSheet

- Setting
  - Online classes (university is still closed) with no technical support
    - Students set up their own lab, with material and assistance provided by yours truly
    - GitHub (https://github.com/), MSYS2 (https://www.msys2.org/), …
  - Students must learn C in 10 weeks
    - Nearly all of them have no programming experience
  - An old professor and I have been advocating to modernize the curriculum
    - … in fact, we have already started to do so, under the RADAR
    - … and have been gaining support from researchers and institutions

  - Lilly will officially be part of the course in the period that starts next week

Lilly REPL Session

The response to an **action** can be an ACK, followed by an echo of the command, or an ERROR with a descriptive message.

The response to an **expression** can be an OK followed by a description of the reduction, or an ERROR with a descriptive message.

Variable z in this session is initialized with a quoted expression, therefore inferred to have a lazy type

```
ZenSheet REPL - Beta 0.1
Connecting via net protocol to localhost:3899 ...

< ZR > :: data := [ 0, 1, 42, 67, 3, 7, 997, 8 ];
ACK: :: data := [ 0, 1, 42, 67, 3, 7, 997, 8 ];

< ZR > filter(fn(x) -> x < 50, data)
OK: filter(fn(x) -> x < 50, data) ==> [0, 1, 42, 3, 7, 8]

< ZR > :: predicate := fn(x) -> x < 50;
ACK: :: predicate := fn(x) -> x < 50;

< ZR > :: z := 'filter(predicate, data)';
ACK: :: z := 'filter(predicate, data)’;

< ZR > z
OK: z ==> [0, 1, 42, 3, 7, 8]

< ZR > predicate := fn(x) -> x % 2 = 0;
ACK: predicate := fn(x) -> x % 2 = 0;

<13: 7/7 => 7> z
OK: z ==> [0, 42, 8]
```

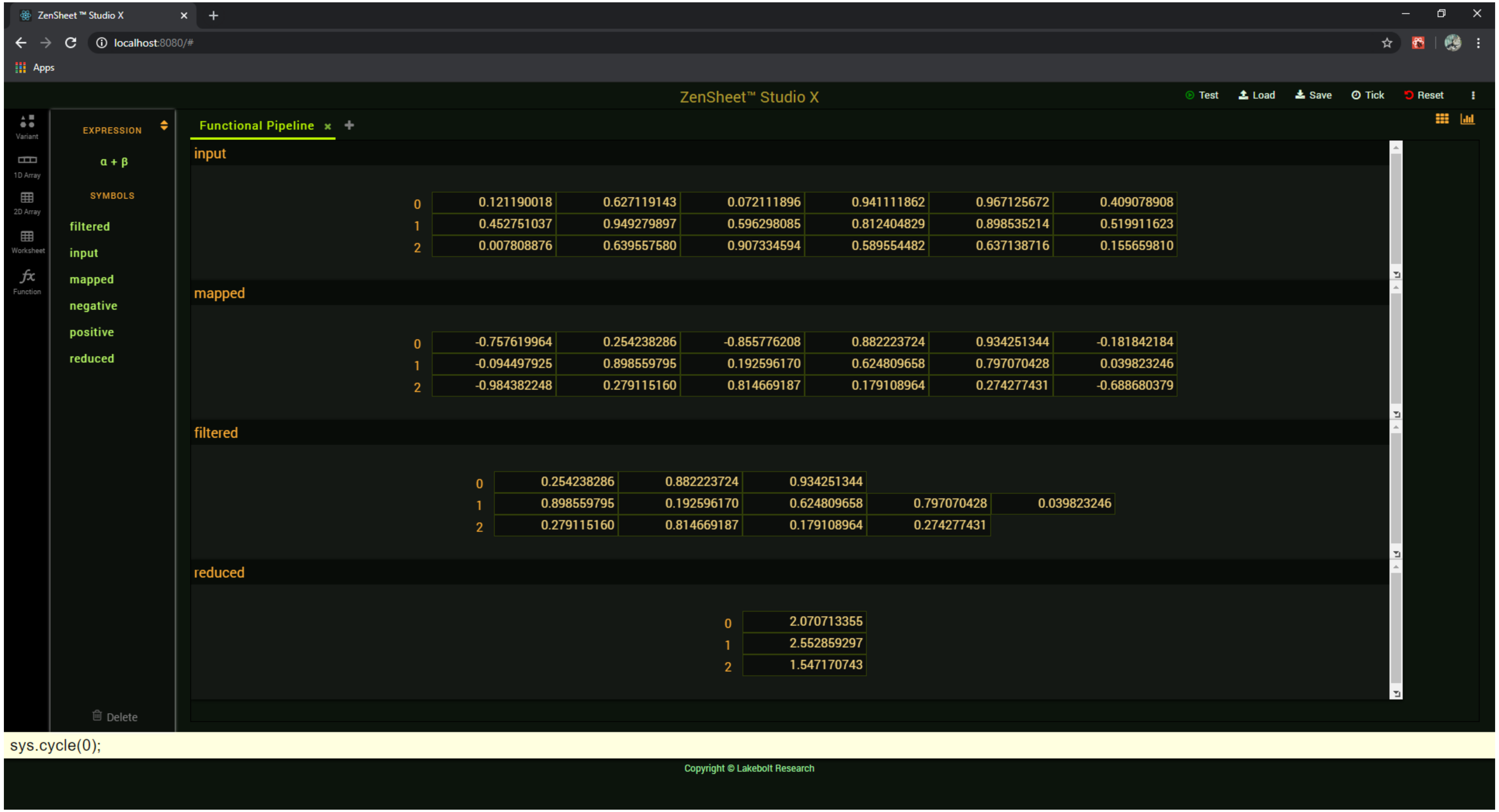
Higher Order Functional Abstraction Example
User implementation of map, filter, fold

```
// map
:: mapz := fn(f, seq) ->
          if(empty(seq), seq, cons(f(head(seq)), mapz(f, tail(seq))));

// filter
:: filterz := fn(pred, seq) ->
          if(empty(seq), seq, if(pred(head(seq)), cons(head(seq), filterz(pred, tail(seq))), filterz(pred, tail(seq))));

// fold
:: foldz := fn(mfn, init, seq) ->
          if(empty(seq), init, foldz(mfn, mfn(init, head(seq)), tail(seq)));
```

Higher Order Functions: a Reactive Pipeline Model

```
///
/// hof.sym
///
/// higher order functions example: reactive pipeline model
///

array[] => array[] => lazy double input := [
  ['uniform()', 'uniform()', 'uniform()', 'uniform()', 'uniform()', 'uniform()'],
  ['uniform()', 'uniform()', 'uniform()', 'uniform()', 'uniform()', 'uniform()'],
  ['uniform()', 'uniform()', 'uniform()', 'uniform()', 'uniform()', 'uniform()']
];

lazy var negative := '/.(x) -> x < 0';
lazy var positive := '/.(x) -> x >= 0';

lazy var mapped := 'map(/.(row) -> map(/.(x) -> 2 * x - 1, row), input)';

lazy var filtered := 'map(/.(row) -> filter(positive, row), mapped)';

lazy var reduced := 'map(/.(row) -> sum(row), filtered)';
```

Reactive Pipeline Model – rendered in ZenSheet UI



Findings and Preliminary Decisions

- Post grade survey: nearly all students reported that Lilly was valuable or very valuable to their learning experience. No one considered it detrimental.
- Tech issues: deployment turned out to be even more of a challenge than anticipated ☹
- Focusing on concepts and paradigms, showing how they are supported in different languages, reduces the “are we learning the right language” worries.
- Adding **Lilly** and **JavaScript** appears to help overcome syntax-related anxiety.
- We are replacing C with **C++** (already used C++ last trimester) even more.
- We also plan to use **Haskell** to show examples of parametric polymorphism.

References

- [1] F. Hermans, “Keynote: How to teach programming and other things?,” *https://www.youtube.com/watch?v=UJxXgugvXmE*, 2018. .
- [2] S. P. Jones, A. Blackwell, and M. Burnett, “A user-centred approach to functions in Excel,” in *ACM SIGPLAN Notices*, 2003, vol. 38, no. 9, pp. 165–176, doi: 10.1145/944746.944721.
- [3] R. Abraham, M. Burnett, and M. Erwig, “Spreadsheet Programming,” in Wiley *Encyclopedia of Computer Science and Engineering*, 2009, pp. 1–10.
- [4] S. P. Jones, M. Burnett, and A. Blackwell, “Spreadsheets : functional programming for the masses.” https://www.slideshare.net/kfrdbs/peyton-jones.
- [5] Microsoft Research, “Future of Spreadsheeting,” 2019. https://www.microsoft.com/en-us/research/video/future-of-spreadsheeting/.
- [6] Microsoft Research, “LAMBDA,” 2020. https://techcommunity.microsoft.com/t5/excel-blog/announcing-lambda-turn-excel-formulas-into-custom-functions/ba-p/1925546.
- [7] M. McCutchen, J. Borghouts, A. D. Gordon, and S. P. Jones, “Elastic Sheet-Defined Functions : Generalising Spreadsheet Functions to Variable-Size Input Arrays ∗,” vol. 1, no. March, 2018.
- [8] M. Figuera, “ZenSheet Studio: A Spreadsheet-Inspired Environment for Reactive Computing,” in *SPLASH Companion 2017 - Proceedings Companion of the 2017 ACM SIGPLAN International Conference on Systems, Programming, Languages, and Applications: Software for Humanity*, Oct. 2017, pp. 33–35, doi: 10.1145/3135932.3135949.
- [9] E. Alda and M. Figuera, “ZenSheet: a live programming environment for reactive computing,” 2017. https://2017.splashcon.org/details/live-2017/5/ZenSheet-a-live-programming-environment-for-reactive-computing.
- [10] Microsoft Research, “Preview of Dynamic Arrays in Excel,” 2018. https://techcommunity.microsoft.com/t5/excel-blog/preview-of-dynamic-arrays-in-excel/ba-p/252944.
- [11] J. G. Siek and W. Taha, “Gradual typing for objects,” in *ECOOP’07*, 2007, pp. 2–27.
- [12] E. Alda and J. Lopéz, “Lambda Days 2020,” 2020. https://www.youtube.com/watch?v=mJa0_gKE6xo.