

Notation

Pat Hanrahan

"Semantics, Not Syntax"

DSLs

Syntax and Semantics

Functions and sets (Semantics)

Concrete syntax (notation)

Abstract syntax tree (Syntax)

Evaluator (Syntax \rightarrow Semantics)

Laws

```
data Expr a = - Functor
    Add (Expr a) (Expr a) -
    | Mul (Expr a) (Expr a)
    | Con a
```

```
type Syntax = Expr Integer
type Semantics = Integer
```

```
eval :: Syntax -> Semantics
eval (Add e1 e2) = (eval e1) + (eval e2)
eval (Mul e1 e2) = (eval e1) * (eval e2)
eval (Con c)     = c
```

```
// universal algebra / algebraic data type
```

```
// see DSLsofMath
```

– Laws

```
open import Relation.Binary.PropositionalEquality using (_≡_)
open import Data.Nat using (ℕ; _+_, *__)
open import Data.Nat.Properties using
  (+-assoc; +-identityl; +-identityr; +-comm;
  *-assoc; *-identityl; *-identityr; *-distribr-+)

record IsSemiRing {A : Set}
  (_+_ : A → A → A) (e : A)
  (_*_ : A → A → A) (f : A) : Set where
  field
    +-assoc : ∀ (x y z : A) → (x + y) + z ≡ x + (y + z)
    +-identityl : ∀ (x : A) → e + x ≡ x
    +-identityr : ∀ (x : A) → x + e ≡ x
    +-comm : ∀ (x y : A) → x + y ≡ y + x
    *-assoc : ∀ (x y z : A) → (x * y) * z ≡ x * (y * z)
    *-identityl : ∀ (x : A) → f * x ≡ x
    *-identityr : ∀ (x : A) → x * f ≡ x
    *-distrib-+ : ∀ x y z → (x * (y + z)) ≡ ((x * y) + (x * z))
```

Definition: *vector space*

A *vector space* is a set V along with an addition on V and a scalar multiplication on V such that the following properties hold:

- $u + w = w + u$ for all $u, w \in V$;
 - $(u + v) + w = u + (v + w)$ and $(ab)u = a(bu)$ for all $u, v, w \in V$ and all $a, b \in \mathbf{F}$;
 - there exists $0 \in V$ such that $u + 0 = u$ for all $u \in V$;
 - for every $u \in V$, there exists $w \in V$ such that $u + w = 0$;
 - $1u = u$ for all $u \in V$;
 - $a(u + w) = au + aw$ and $(a + b)u = au + bu$ for all $a, b \in \mathbf{F}$ and all $u, w \in V$.
- commutativity
associativity
additive identity
additive inverse
multiplicative identity
distributive properties

Two vectors are equal if they are isomorphic via a linear transformation. That is, the coordinate system is arbitrary

Dual space with inner product

A *tensor* is formed from two vector spaces V and W using the tensor product $V \otimes W$.

With the following laws (bilinearity)

$$s(v \otimes w) = (sv) \otimes w = v \otimes (sw)$$

$$(u \otimes w) + (v \otimes w) = (u+v) \otimes w$$

Can form multilinear products $U \otimes V \otimes W$

Covariant and Contravariant vectors

THE FOUNDATION OF THE GENERAL THEORY OF RELATIVITY

BY A. EINSTEIN

“*Contraction*” of a *Mixed Tensor*.—From any mixed tensor we may form a tensor whose rank is less by two, by equating an index of covariant with one of contravariant character, and summing with respect to this index (“*contraction*”). Thus, for example, from the mixed tensor of the fourth rank $A_{\mu\nu}^{\sigma\tau}$, we obtain the mixed tensor of the second rank,

$$A_{\nu}^{\tau} = A_{\mu\nu}^{\mu\tau} \quad (= \sum_{\mu} A_{\mu\nu}^{\mu\tau}),$$

and from this, by a second contraction, the tensor of zero rank,

$$A = A_{\nu}^{\nu} = A_{\mu\nu}^{\mu\nu}.$$

The theory which is presented in the following pages conceivably constitutes the farthest-reaching generalization of a theory which, today, is generally called the “theory of relativity”; I will call the latter one—in order to distinguish it from the first named—the “special theory of relativity,” which I assume to be known. The generalization of the theory of relativity has been facilitated considerably by Minkowski, a mathematician who was the first one to recognize the formal equivalence of space coordinates and the time coordinate, and utilized this in the construction of the theory. The mathematical tools that are necessary for general relativity were readily available in the “absolute differential calculus,” which is based upon the research on non-Euclidean manifolds by Gauss, Riemann, and Christoffel, and which has been systematized by Ricci and Levi-Civita and has already been applied to problems of theoretical physics. In section B of the present paper I developed all the necessary mathematical tools—which cannot be assumed to be known to every physicist—and I tried to do it in as simple and transparent a manner as possible, so that a special study of the mathematical literature is not required for the understanding of the present paper. Finally, I want to acknowledge gratefully my friend, the mathematician Grossmann, whose help not only saved me the effort of studying the pertinent mathematical literature, but who also helped me in my search for the field equations of gravitation.

Penrose Diagrams

Applications of Negative Dimensional Tensors

ROGER PENROSE

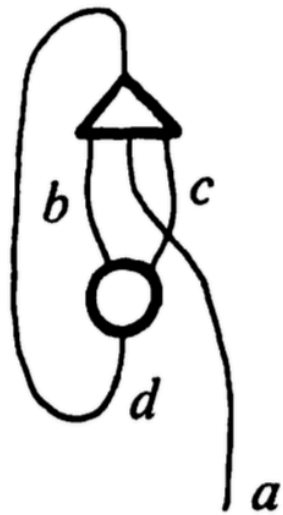
Birkbeck College, University of London, England

I wish to describe a theory of “abstract tensor systems” (abbreviated ATS) and indicate some applications. Unfortunately I shall only be able to give a very brief outline of the general theory here.†

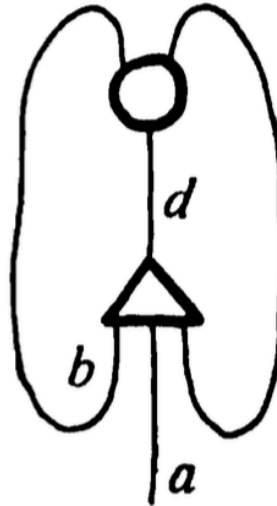
Penrose Diagrams



$$= \theta_c^{af} \chi_{fde}^b \in \mathcal{T}_{cde}^{ab}$$

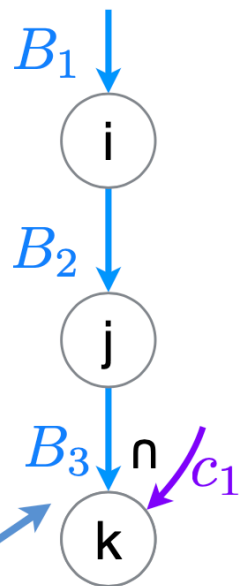


=



$$= \theta_d^{bc} \chi_{bac}^d \in \mathcal{T}_a.$$

$$A_{ij} = \sum_k B_{ijk} c_k$$



```

for (int i = 0; i < m; i++) {
  for (int pB2 = B2_pos[i]; pB2 < B2_pos[i+1]; pB2++) {
    int j = B2_crd[pB2];

    int pA2 = i*n + j;
    int pB3 = B3_pos[pB2];
    int pc1 = c1_pos[0];
    while (pB3 < B3_pos[pB2+1] && pc1 < c1_pos[1]) {
      int kB = B3_crd[pB3];
      int kc = c1_crd[pc1];
      int k = min(kB, kc);
      if (kB == k && kc == k) {
        A[pA2] += B[pB3] * c[pc1];
      }
      if (kB == k) pB3++;
      if (kc == k) pc1++;
    }
  }
}

```

Key operation is to coiterate over data structures

Intersection coiteration

Concrete index notation specifies order of computations and location of intermediate values

$$A_{ij} = B_{ij} + C_{ij}$$



$$\forall_i \forall_j A_{ij} = B_{ij} + C_{ij}$$

$$\alpha = \sum_i b_i c_i$$



$$\forall_i \alpha = b_i c_i$$

$$a_i = \sum_j B_{ij} c_j$$



$$\forall_i \forall_j a_i = t \textbf{ where } t = B_{ij} c_j$$

Representations

Set Representations (Type Isomorphism)

Representation	Union	Find
Cons	$O(n^2)$	$O(n)$
Sorted Cons	$O(n)$	$O(\log(n))$
Hash	$O(n^2)$	$O(1)$
Bit Vector	$O(n)$	$O(1)$
Binary Tree	$O(\log(n))$	$O(n \log(n))$
Union-Find Forest	$O(\alpha(n))$	$O(\alpha(n))$

An Automatic Technique for Selection of Data Representations in SETL Programs, Schonberg, Schwarz, Sharir, 1981

Herb Simon



Nobel Prize in Economics (1977)

"for his pioneering research into the decision-making process within economic organizations"

Turing Award (1975)

"basic contributions to artificial intelligence, the psychology of human cognition, and list processing"

Number Scrabble

Goal: Pick three numbers that sum to 15



A:

B:

Number Scrabble

Goal: Pick three numbers that sum to 15

1

6

7

9

A:

8

4

5

B:

2

3

?

Number Scrabble

Goal: Pick three numbers that sum to 15



A: 8 4 5

B: 2 3 ?

Number Scrabble

Goal: Pick three numbers that sum to 15



A:


8


B:

Number Scrabble

Goal: Pick three numbers that sum to 15



A: 

B: 

Number Scrabble

Goal: Pick three numbers that sum to 15

1

3

5

6

7

9

A:

8

4

B:

2

Number Scrabble

Goal: Pick three numbers that sum to 15

1

5

6

7

9

A:

8

4

B:

2

3

Number Scrabble

Goal: Pick three numbers that sum to 15

1

6

7

9

A:

8

4

5

B:

2

3

?

Problem Isomorphs

Problem Isomorph

4	3	8
9	5	1
2	7	6

Magic Square: All rows, columns, diagonals sum to 15

Transform to Tic-Tac-Toe

4	3	8
9	5	1
2	7	6

Transform to Tic-Tac-Toe

4	3	8
9	5	1
2	7	6

Transform to Tic-Tac-Toe

4	3	8
9	5	1
2	7	6

Transform to Tic-Tac-Toe

4	3	8
9	5	1
2	7	6

Transform to Tic-Tac-Toe

4	3	8
9	5	1
2	7	6

Transform to Tic-Tac-Toe

4	3	8
9	5	1
2	7	6

**“Why is a Picture
(Sometimes) Worth
10,000 Words”**

**Larkin and Simon,
Cognitive Science, 1987**

Why?

Reduce memory load

- ❑ Working memory is limited
- ❑ Store information in the diagram

Reduce search time

- ❑ Pre-attentive (constant-time) search process
- ❑ Spatially-indexed patterns store the “facts”

Allow perceptual inference

- ❑ Map inference to pattern finding

The Representation Effect

Although two representations
may be equivalent,
one is often much “better”
for a given problem

“Better” means

- ❑ Faster
- ❑ Fewer errors
- ❑ Better comprehension
- ❑ ...

Summary of Financial Performance

		Central		East		South		West	
		Total Sales	Total Profit	Total Sales	Total Profit	Total Sales	Total Profit	Total Sales	Total Profit
Coffee	Amaretto	\$14,011	5,105	\$2,993	1,009			\$9,265	-1,225
	Columbian	\$28,913	8,528	\$47,386	27,253	\$21,664	8,767	\$30,357	11,253
	Decaf Irish Cream	\$26,155	9,632	\$6,261	2,727	\$11,592	2,933	\$18,235	-1,305
Espresso	Caffe Latte					\$15,442	3,872	\$20,458	7,502
	Caffe Mocha	\$35,218	14,640	\$16,646	-6,230	\$14,163	5,201	\$18,876	4,064
	DecafEspresso	\$24,485	8,860	\$7,722	2,410	\$15,384	5,930	\$30,578	12,302
	RegularEspresso			\$24,036	10,062				
Herbal Tea	Chamomile	\$36,570	14,434	\$2,194	765	\$11,186	3,180	\$25,632	8,852
	Lemon	\$21,978	6,251	\$27,176	7,901	\$14,497	2,593	\$32,274	13,120
	Mint	\$9,337	4,069	\$11,992	-2,242			\$14,380	4,330
Tea	Darjeeling	\$30,289	10,772	\$14,096	6,497			\$28,769	11,780
	Earl Grey	\$32,881	10,331	\$6,505	3,405			\$27,387	10,425
	Green Tea	\$5,211	1,227	\$11,571	5,654			\$16,063	-7,109

How much mint tea was sold in the west?

Summary of Financial Performance

		Central		East		South		West	
		Total Sales	Total Profit	Total Sales	Total Profit	Total Sales	Total Profit	Total Sales	Total Profit
Coffee	Amaretto	\$14,011	5,105	\$2,993	1,009			\$9,265	-1,225
	Columbian	\$28,913	8,528	\$47,386	27,253	\$21,664	8,767	\$30,357	11,253
	Decaf Irish Cream	\$26,155	9,632	\$6,261	2,727	\$11,592	2,933	\$18,235	-1,305
Espresso	Caffe Latte					\$15,442	3,872	\$20,458	7,502
	Caffe Mocha	\$35,218	14,640	\$16,646	-6,230	\$14,163	5,201	\$18,876	4,064
	DecafEspresso	\$24,485	8,860	\$7,722	2,410	\$15,384	5,930	\$30,578	12,302
	RegularEspresso			\$24,036	10,062				
Herbal Tea	Chamomile	\$36,570	14,434	\$2,194	765	\$11,186	3,180	\$25,632	8,852
	Lemon	\$21,978	6,251	\$27,176	7,901	\$14,497	2,593	\$32,274	13,120
	Mint	\$9,337	4,069	\$11,992	-2,242			\$14,380	4,330
Tea	Darjeeling	\$30,289	10,772	\$14,096	6,497			\$28,769	11,780
	Earl Grey	\$32,881	10,331	\$6,505	3,405			\$27,387	10,425
	Green Tea	\$5,211	1,227	\$11,571	5,654			\$16,063	-7,109

What product in what region sold the most?

Summary of Financial Performance



What product in what region sold the most?

“Number Representations”

Zhang and Norman

Number Representations

Counting – Tallying



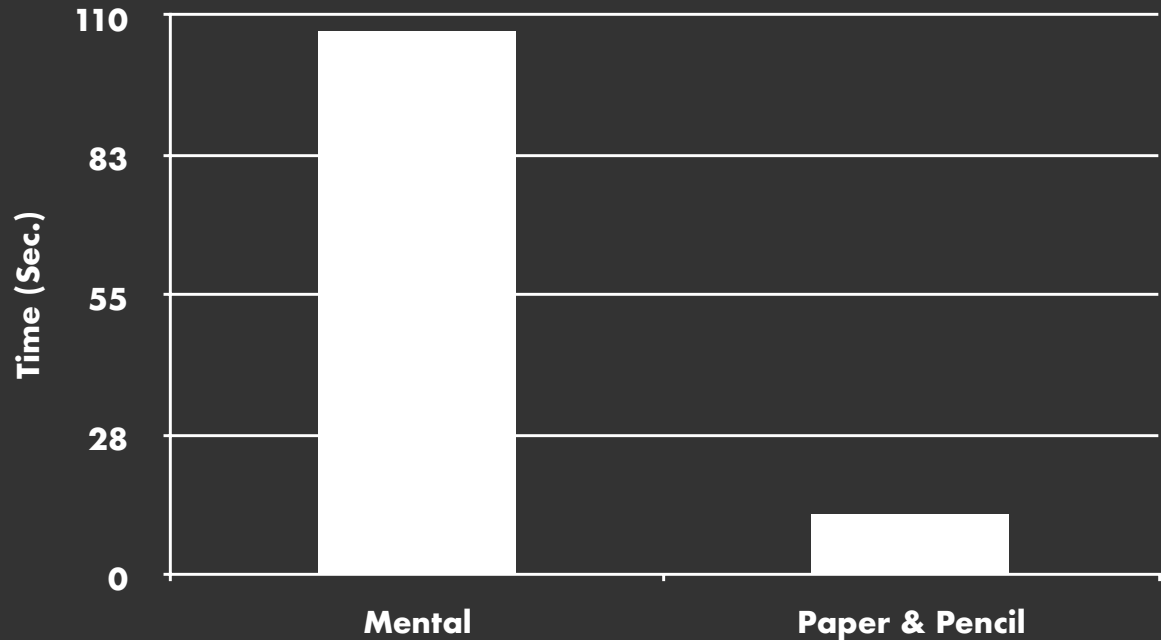
Adding – Roman numerals

$$XXIII + XII = XXXIIII = XXXV$$

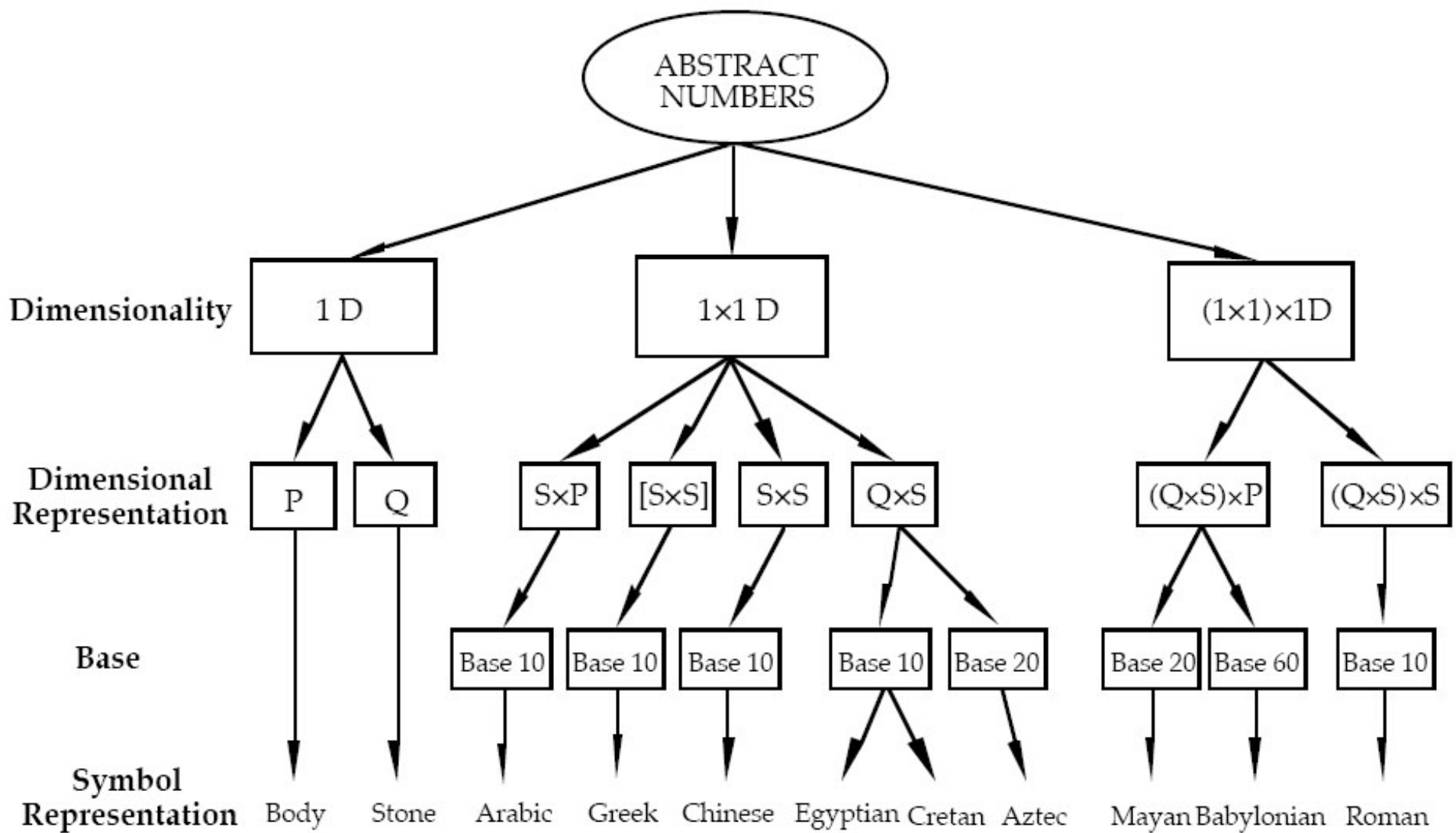
Multiplication – Arabic number systems

Long-Hand Multiplication

$$\begin{array}{r} 34 \\ \times 72 \\ \hline 68 \\ 238 \\ \hline 2448 \end{array}$$



From "Introduction to Information Visualization,"
Card, Schneiderman, Mackinlay



Zhang and Norman, *The Representations of Numbers*,
Cognition, 57, 271-295, 1996

Distributed Cognition

External (E) vs. Internal (I) process

		Roman	Arabic
1.	Separate power & base	I	E
2.	Get base value	E	I
3.	Multiply base values	I	I
4.	Get power values	I	E
5.	Add power values	I	E
6.	Combine base & power	I	E
7.	Add results	I	E

Arabic more efficient than Roman



**Notation
as a Tool of Thought**

Kenneth Iverson

Notation as a Tool for Thought

“The thesis of the present paper is that the advantages of executability and universality found in programming languages can be effectively combined, in a single coherent language, with the advantages offered by mathematical notation”

K. Iverson

Unambiguous executable mathematics

Arithmetic and Algebra in APL (k)

```
> k = 5
```

```
> til k
```

```
0 1 2 3 4
```

```
> 1 + 2 * til k
```

```
1 3 5 7 9
```

```
> +/ 1+2*til k // 1 + 3 + 5 + 7 + 9
```

```
25
```

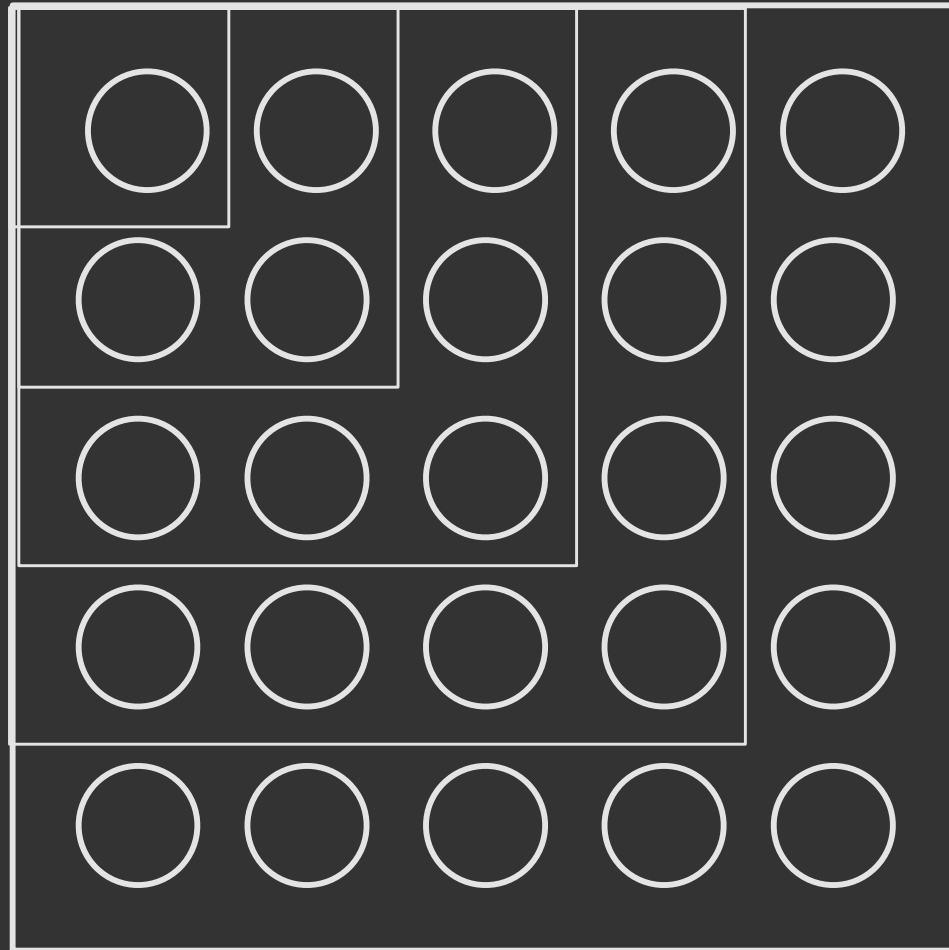
```
> k*k
```

```
25
```

Program Transformations as Proofs

```
// sum of k odd numbers
  +/ (1 + 2 * til k)
== // definition of multiplication
  +/ (1 + (til k) + (til k))
== // addition is commutative and associative
  +/ (1 + (til k) + (reverse til k))
== // 0 1 2 + 2 1 0 = (0+2) + (1+1) + (2+0) = 2+2+2 // +/ k # k-1
  +/ (1 + k # (k-1))
== // scalar + vector causes scalar to be repeated k times
  +/ k#k
== // definition of multiplication as repeated addition: k*k = +/ k#k
  // e.g. 3*3 = +/ 3 3 3      ,
  k*k
```

Visual Proofs



Algebra

$$1+3+5+7+9=5^2$$

data \mathbb{N} : Set where

zero : \mathbb{N}

suc : $\mathbb{N} \rightarrow \mathbb{N}$

+ : $\mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{N}$

zero + n = n

suc m + n = suc (m + n)

+-identity^r : $\forall (m : \mathbb{N}) \rightarrow m + \text{zero} \equiv m$

+-identity^r zero =

begin

zero + zero

$\equiv \langle \rangle$

zero

■

+-identity^r (suc m) =

begin

suc m + zero

$\equiv \langle \rangle$

suc (m + zero)

$\equiv \langle \text{cong suc (+-identity^r m) } \rangle$

suc m

■

Characteristics of Notation

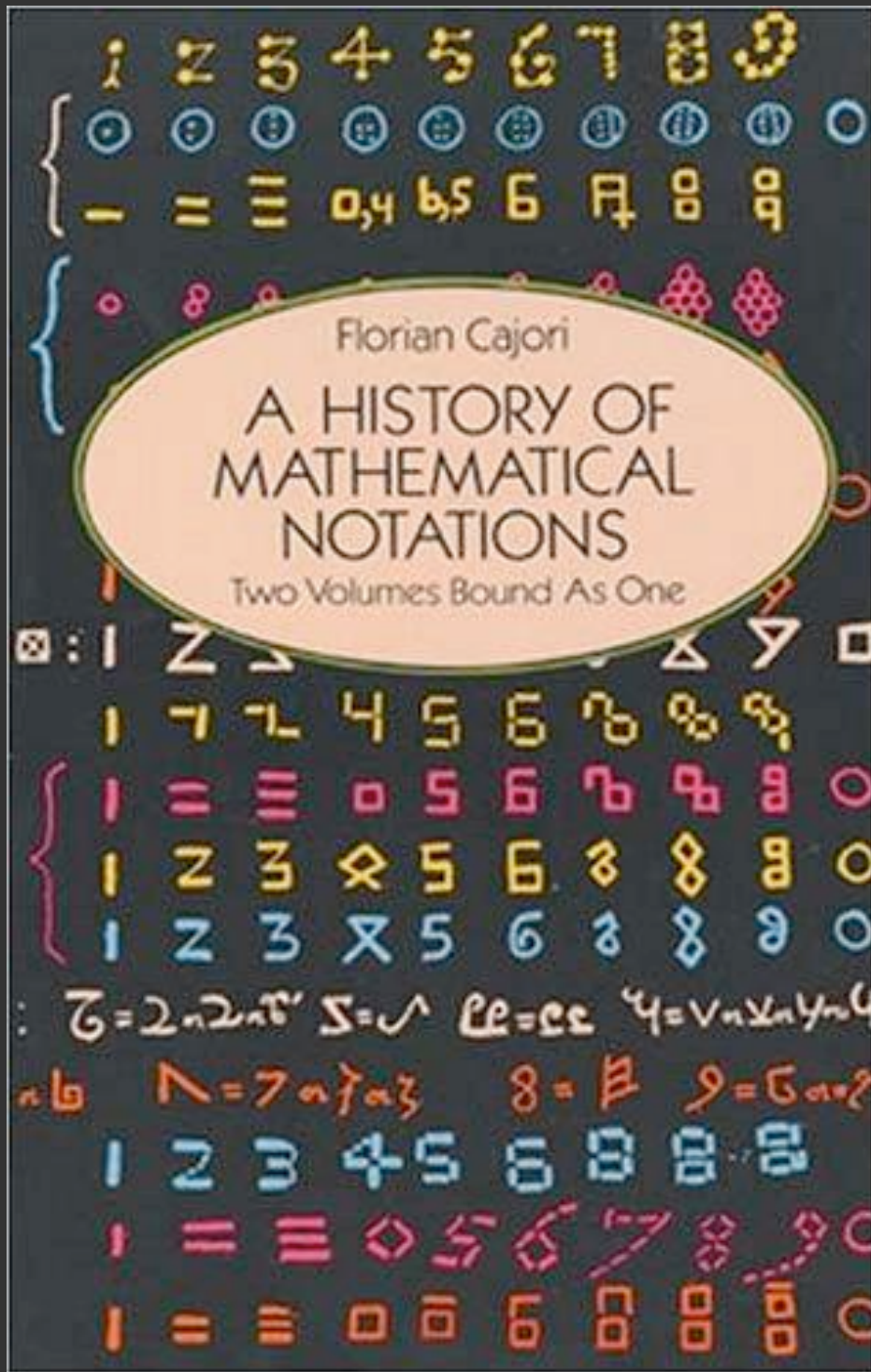
1 Ease of expression (natural)

2 Suggestivity (expose patterns)

3 Subordinate detail (abstract)

4 Economy (concise)

5 Formal (proofs)



Florian Cajori (1859-1930)

History of Mathematics

University of California,

Berkeley

Mathematica

Rich character and symbol set (2500)

Multiple versions of a symbol to resolve ambiguity

- *i*
- double struck $i = \sqrt{-1}$

Two-dimensional (over, subscript, ...)

StandardForm, TraditionalForm, ...

- 100 heuristics to go from traditional to standard

Input macros (map ascii to symbol, ->)

[Mathematical Notation: Past and Future, Stephen Wolfram, 2000](#)

The Incredible Convenience of Mathematica Image Processing

Theodore Gray

ChatGPT

Supporting Notation

```
// Lisp
```

```
// Uniform syntax - s expressions
```

```
(cond
```

```
  ((= n 10) (= m 1))
```

```
  ((> n 10) (= m 2) (= n (* n m)))
```

```
  ((< n 10) (= n 0)))
```

```
// Symbols are characters delimited by
```

```
// spaces and punctuation
```

```
// Macros allow special forms (e.g.
```

```
cond)
```

```
// Minimal notation!
```



```
// Smalltalk
```

```
employee name first
```

```
// unary operators / methods
```

```
// parsed left to right
```

```
// left associative
```

```
// = employee.name.first
```

```
// binary operators
```

```
// left associative, no precedence
```

```
1 + 2 * 3 = (1 + 2) * 3
```

```
// unary ops have precedence over binary
```

```
1 + 4 sqrt = 1 + (4 sqrt)
```

```
// APL (k)
```

```
+ / 1 + 2 * til k
```

```
// monadic (til) and dyadic (+ *)
```

```
//
```

```
// right associative
```

```
// functions have same precedence
```

```
// operators (higher-order functions)
```

```
// operators > functions
```

```
+ / (1 + (2 * (til k)))
```

```
// Haskell
```

```
// sections
```

```
(+) 1 2 = 1 + 2
```

```
// currying, left associative
```

```
((+) 1) 2
```

```
// precedence and associativity
```

```
infixl 6 +
```

o From the `Prelude.Unicode` module

▪ Values

- `not` = (\neg)
- `(&&)` = (\wedge)
- `(||)` = (\vee)
- `(==)` = (\equiv)
- `(/=)` = (\neq) = (\neq)
- `<=` = \leq = \leq
- `>=` = \geq = \geq
- `pi` = π
- `(/)` = (\div)
- `(*)` = (\cdot)
- `(.)` = (\circ)
- `(++)` = $(\#)$
- `elem` = (\in)
- `notElem` = (\notin)
- `undefined` = (\perp)

▪ Types

- `Integer` = \mathbb{Z}
- `Rational` = \mathbb{Q}

Unicode Symbols

```
// Agda
```

```
// define operators with blanks for  
arguments
```

```
_+_ x y =
```

```
_+_ x y = x + y
```

```
// precedence and associativity
```

```
infixl 6 _+_
```

```
// statement forms
```

```
if_then_else_ x y z =
```

```
if x then y else
```

```
// unicode characters in names
```

```
// identifiers separated with spaces
```

Semantics **And Syntax**