Mathematical Aspects
of
Data Structure

Marcelo Fiore

Computer Laboratory
University of Cambridge

Kiryu
9.IV.2012

Mathematical Structures in Computer Science

- Logic in circuit design.
- Graph theory in networking.
- Fourier analysis in image processing.
- Linear algebra in quantum computation.
- Mathematical analysis in algorithms.
- Automata theory in compilers.
- Markov models in bioinformatics.
- Cryptography in security.
- Game theory in economics.
- Foundations in formal methods.

Mathematical Structures in Computer Science

- Logic in circuit design.
- Graph theory in networking.
- Fourier analysis in image processing.
- Linear algebra in quantum computation.
- Mathematical analysis in algorithms.
- Automata theory in compilers.
- Markov models in bioinformatics.
- Cryptography in security.
- Game theory in economics.
- Foundations in formal methods.
- Algebra, algorithmics, analysis, combinatorics, logic, . . . in programming language theory.

Data Structuring in Programming Languages

1950s	FORTRAN	
1960s	LISP	
1960s	Algol	Simula
1970s	Pascal	Smalltalk
1980s	ML	
1990s	Standard ML	
2000s	Java, Scala	
2010s	Haskell 2010	
	Coq, Agda	

Data Structuring in Programming Languages

1950s	FORTRAN		
1960s	LISP		S-expressions lists
1960s	Algol	Simula	
1970s	Pascal	Smalltalk	
1980s	ML		ADTs
1990s	Standard ML		
2000s	Java, Scala		
2010s	Haskell 2010		GADTs
	Coq, Agda		IFs

Symbolic Expressions

$$
\begin{array}{rcll}
S & ::= & a & (\text{atoms}) \\
 & \mid & S\,.\,S & (\text{pairs})
\end{array}
$$

Symbolic Expressions

$$
\begin{array}{rcll}
S & ::= & a & (\text{atoms}) \\
  & | & S \,.\, S & (\text{pairs})
\end{array}
$$

Binary Trees

Specification:

$$
\begin{array}{rcll}
\mathsf{T} & ::= & \bullet & (\text{nil}) \\
  & | & \mathsf{T} \,,\, \mathsf{T} & (\text{cons})
\end{array}
$$

Semantics:

$$\mathcal{T} \cong \mathbf{1} + \mathcal{T} \times \mathcal{T}$$

Seven Trees in One

Claim I. There is a bijective program of constant complexity

$$\mathsf{T}^7 \cong \mathsf{T} \quad .$$

Seven Trees in One

Claim I. There is a bijective program of constant complexity

$$T^7 \cong T \ .$$

Claim II. This program can be built from programs for the basic bijections:

$$T \cong 1 + T \times T$$

$$1 \times A \cong A \ , \quad (A \times B) \times C \cong A \times (B \times C)$$

$$A \times B \cong B \times A$$

$$(A + B) + C \cong A + (B + C) \ , \quad A + B \cong B + A$$

$$A \times (B + C) \cong (A \times B) + (A \times C)$$

An algebraic proof

1. $T = 1 + T^2$

$$\implies T^2 = T - 1$$

$$\implies T^{n+2} = T^{n+1} - T^n$$

An algebraic proof

1. $T = 1 + T^2$

$$\implies T^2 = T - 1$$
$$\implies T^{n+2} = T^{n+1} - T^n$$

2. $$\begin{aligned} T^7 &= T^6 - T^5 \\ &= T^5 - T^4 - T^5 \\ &= -T^4 \\ &= -T^3 + T^2 \\ &= -T^2 + T + T^2 \\ &= T \end{aligned}$$

Soundness and Completeness of the Algebraic Method

Theorem. Let $p, q_1, q_2 \in \mathbb{N}[x]$ be such that

– p is of degree ≥ 2 with $p(0) \neq 0$, and

– q_1, q_2 are of degree ≥ 1.

If

$$x = p(x) \implies q_1(x) = q_2(x)$$

in the theory of rings

then,

for the data type $D \cong p(D)$,

there is a bijection of constant complexity

$$q_1(D) \cong q_2(D) \quad .$$

Corollary. The word problem in $\mathbb{N}[x]$ modulo $x = p(x)$ is decidable.

Two Problems

1. Investigate the decidability of the word problem for the general case $\mathbb{N}[x_1, \ldots, x_m]$ modulo $p_1 = q_1, \ldots, p_n = q_n$.

2. Is there a mathematical theory underlying the following observation?

   Note that

   $$T = 1 + T^2 \implies T = \frac{1}{1-T} = \sum_{n\in\mathbb{N}} T^n = T^*$$

   and that for

   $$T \cong 1 + T^2$$

   there is a *primitive recursive* bijection

   $$T \cong T^* \quad .$$

The Arithmetic of Types

- In the type theory of $+$ and $\times$, type isomorphism is axiomatised by the laws of arithmetic (*i.e.* commutative semiring structure).

The Arithmetic of Types

- In the type theory of $+$ and $\times$, type isomorphism is axiomatised by the laws of arithmetic (*i.e.* commutative semiring structure).

- In the type theory of $\times$ and $\Rightarrow$, type isomorphism is axiomatised by the laws of arithmetic; *i.e.* the commutative monoid laws of $\times$ and the laws of exponentiation:

$$A \Rightarrow (B \times C) \quad \cong \quad (A \Rightarrow B) \times (A \Rightarrow C)$$

$$(A \times B) \Rightarrow C \quad \cong \quad A \Rightarrow B \Rightarrow C$$

- In the type theory of $+$, $\times$, and $\Rightarrow$, type isomorphism is <u>not finitely axiomatisable</u>.

- In the type theory of $+$, $\times$, and $\Rightarrow$, type isomorphism is not finitely axiomatisable.

  The proof uses the lemma:

$$\frac{A \times D \cong C \times B \qquad U \times V \cong X \times Y}{\begin{array}{l} V \Rightarrow \big[(U \Rightarrow A) + (U \Rightarrow B)\big] \\ \quad \times\, Y \Rightarrow \big[(X \Rightarrow C) + (X \Rightarrow D)\big] \\ \cong \\ Y \Rightarrow \big[(X \Rightarrow A) + (X \Rightarrow B)\big] \\ \quad \times\, V \Rightarrow \big[(U \Rightarrow C) + (U \Rightarrow D)\big] \end{array}}$$

  in connection with Tarski's High School Algebra Problem in mathematical logic.

- In the type theory of $+$, $\times$, and $\Rightarrow$, type isomorphism is <u>not finitely axiomatisable</u>.

  The proof uses the lemma:

$$\frac{A \times D \cong C \times B \qquad U \times V \cong X \times Y}{\begin{array}{l} V \Rightarrow \big[(U \Rightarrow A) + (U \Rightarrow B)\big] \\ \quad \times\, Y \Rightarrow \big[(X \Rightarrow C) + (X \Rightarrow D)\big] \\ \cong \\ Y \Rightarrow \big[(X \Rightarrow A) + (X \Rightarrow B)\big] \\ \quad \times\, V \Rightarrow \big[(U \Rightarrow C) + (U \Rightarrow D)\big] \end{array}}$$

  in connection with <u>Tarski's High School Algebra Problem</u> in mathematical logic.

  NB: The lemma provides a combinatorial proof of a number-theoretic identity.

Tree Navigation

The operations down and up

Tree Navigation

The operations down and up:

$$\mathsf{down}_1(\ t.t'\ ,\ \Gamma\ ) = (\ t\ ,\ (2, t') :: \Gamma\ )$$

$$\mathsf{down}_2(\ t.t'\ ,\ \Gamma\ ) = (\ t'\ ,\ (1, t) :: \Gamma\ )$$

Tree Navigation

The operations down and up:

$$\text{down}_1(\ t.t'\ ,\ \Gamma\ ) = (\ t\ ,\ (2, t') :: \Gamma\ )$$

$$\text{down}_2(\ t.t'\ ,\ \Gamma\ ) = (\ t'\ ,\ (1, t) :: \Gamma\ )$$

$$\text{up}(\ t\ ,\ (1, t') :: \Gamma\ ) = (\ t'.t\ ,\ \Gamma\ )$$

$$\text{up}(\ t\ ,\ (2, t') :: \Gamma\ ) = (\ t.t'\ ,\ \Gamma\ )$$

Tree Navigation

The operations down and up:

$$\text{down}_1(\ t.t'\ ,\ \Gamma\ ) = (\ t\ ,\ (2, t') :: \Gamma\ )$$

$$\text{down}_2(\ t.t'\ ,\ \Gamma\ ) = (\ t'\ ,\ (1, t) :: \Gamma\ )$$

$$\text{up}(\ t\ ,\ (1, t') :: \Gamma\ ) = (\ t'.t\ ,\ \Gamma\ )$$

$$\text{up}(\ t\ ,\ (2, t') :: \Gamma\ ) = (\ t.t'\ ,\ \Gamma\ )$$

and their types:

$$\text{down} : \mathbf{2} \rightarrow \mathsf{T} \times \mathsf{C} \rightarrow \mathsf{T} \times \mathsf{C}$$

$$\text{up} : \mathsf{T} \times \mathsf{C} \rightarrow \mathsf{T} \times \mathsf{C}$$

where $\mathsf{C} = (\mathbf{2} \times \mathsf{T})^*$

Mathematical Structure of ADTs Navigation

$D \cong p(D)$, with $p(X) = \sum_{k \in K} X^{A_k}$

Mathematical Structure of ADTs Navigation

For

$$D \cong p(D) \quad , \quad \text{with } p(X) = \sum_{k\in K} X^{A_k}$$

we have the operations

$$\text{down}^{(k)} : \; A_k \to D \times C \to D \times C \qquad (k \in K)$$

$$\text{up} : D \times C \to D \times C$$

Mathematical Structure of ADTs Navigation

For

$$D \cong p(D) \quad , \quad \text{with } \; p(X) = \textstyle\sum_{k \in K} X^{A_k}$$

we have the operations

$$\mathsf{down}^{(k)} : \; A_k \to D \times C \to D \times C \qquad (k \in K)$$

$$\mathsf{up} : D \times C \to D \times C$$

where

$$C = \big(\, p'(D) \,\big)^*$$

with

$$p'(X) = \textstyle\sum_{k \in K} A_k \times X^{A_k - 1}$$

the derivative of p.

- The type descriptions

$$\mathsf{down}^{(k)} : \ A_k \to D \times C \to D \times C \qquad (k \in K)$$

are adequate, but not precise.

Dependent Types

- The type descriptions

  $$\mathsf{down}^{(k)} : \; A_k \longrightarrow D \times C \longrightarrow D \times C \qquad (k \in K)$$

  are adequate, but not precise.

- For precision:

  $$\mathsf{down} : \; (k : K) \longrightarrow \; A_k \longrightarrow D^{A_k} \times C \longrightarrow D \times C$$

  dependent types are needed.

Generalised ADTs

- Exponential lists.

$$\begin{array}{rl} \mathsf{Lexp}\ \alpha & \\ = & \mathtt{nil} : \mathsf{Lexp}\ \alpha \\ \mid & \mathtt{cons} : \alpha \times \mathsf{Lexp}(\alpha \times \alpha) \rightarrow \mathsf{L}\,\mathrm{exp}\ \alpha \end{array}$$

Generalised ADTs

- Exponential lists.

$$\begin{array}{l} \mathsf{Lexp}\ \alpha \\ \quad = \quad \mathtt{nil} : \mathsf{Lexp}\ \alpha \\ \quad\ \ \mid \quad \mathtt{cons} : \alpha \times \mathsf{Lexp}(\alpha \times \alpha) \to \mathsf{L}\exp\ \alpha \end{array}$$

Generates

$[\,]$,

$[a_1]$,

$[a_1, (a_2, a_3)]$,

$\big[a_1, (a_2, a_3), \big((a_4, a_5), (a_6, a_7)\big)\big]$,

. . .

i.e., lists of $2^n - 1$ elements.

$$\mathsf{Lexp}\,\alpha \;\cong\; 1 + \alpha \times \mathsf{L}\exp(\alpha \times \alpha)$$

Inductive Families

- Natural numbers.

$$\mathsf{Nat} \;=\; \mathsf{zero} : \mathsf{Nat} \;\mid\; \mathsf{succ} : \mathsf{Nat} \to \mathsf{Nat}$$

$$\mathsf{Nat} \;\cong\; \mathbf{1} + \mathsf{Nat}$$

Inductive Families

- Natural numbers.

$$\mathsf{Nat} \;=\; \mathsf{zero} : \mathsf{Nat} \;\mid\; \mathsf{succ} : \mathsf{Nat} \to \mathsf{Nat}$$

$$\mathsf{Nat} \;\cong\; \mathbf{1} + \mathsf{Nat}$$

- Finite sets.

$$\begin{aligned} &\mathsf{Fin}(n : \mathsf{Nat}) \\ &\quad = \; \mathsf{z} : \mathsf{Fin}(\mathsf{succ}\ n) \\ &\quad \mid \; \mathsf{s} : \mathsf{Fin}(n) \to \mathsf{Fin}(\mathsf{succ}\ n) \end{aligned}$$

$$\left\{ \begin{array}{rcl} \mathsf{Fin}(\mathsf{zero}) & \cong & \mathbf{0} \\ \mathsf{Fin}(\mathsf{succ}\ n) & \cong & \mathbf{1} + \mathsf{Fin}(n) \end{array} \right.$$

- λ-terms (modulo α-equivalence a la de Bruijn).

$$\begin{array}{rl} \mathsf{Lam}(n : \mathsf{Nat}) & \\ = & \mathsf{var} : \mathsf{Fin}(n) \to \mathsf{Lam}(n) \\ \mid & \mathsf{apl} : \mathsf{Lam}(n) \times \mathsf{Lam}(n) \to \mathsf{Lam}(n) \\ \mid & \mathsf{abs} : \mathsf{Lam}(\mathsf{succ}\ n) \to \mathsf{Lam}(n) \end{array}$$

$$\begin{array}{l} \mathsf{Lam}(n) \\ \quad \cong \mathsf{Fin}(n) + \mathsf{Lam}(n) \times \mathsf{Lam}(n) \\ \qquad + \mathsf{Lam}(\mathsf{succ}\ n) \end{array}$$

Mathematical Structure of GADTs & IFs

- Generalise from polynomial constructions:

$$X \mapsto \sum_{k \in C} \prod_{\ell \in A_k} X$$

Mathematical Structure of GADTs & IFs

- Generalise from polynomial constructions:

$$X \mapsto \sum_{k \in C} \prod_{\ell \in A_k} X$$

to *multivariate power-series* constructions:

$$\left(X_i \right)_{i \in I} \mapsto \left(\sum_{k \in C_i} \prod_{\ell \in A_k} X_{\alpha(k,\ell)} \right)_{i \in I}$$

Mathematical Structure of GADTs & IFs

- Generalise from polynomial constructions:

$$X \mapsto \sum_{k \in C} \prod_{\ell \in A_k} X$$

to <u>*multivariate power-series*</u> constructions:

$$\left(X_i \right)_{i \in I} \mapsto \left(\sum_{k \in C_i} \prod_{\ell \in A_k} X_{\alpha(k,\ell)} \right)_{i \in I}$$

- <u>Differential calculus</u> of partial derivatives.

Mathematical Structure of GADTs & IFs

- Generalise from polynomial constructions:

$$X \mapsto \sum_{k \in C} \prod_{\ell \in A_k} X$$

to <u>*multivariate power-series*</u> constructions:

$$(X_i)_{i \in I} \mapsto \left(\sum_{k \in C_i} \prod_{\ell \in A_k} X_{\alpha(k,\ell)} \right)_{i \in I}$$

- <u>Differential calculus</u> of partial derivatives.

The type of the navigation context for

$$D \cong P(D) \quad , \quad \text{with } P : \mathrm{Fam}(I) \to \mathrm{Fam}(I)$$

is

$$C \in \mathrm{Fam}(I)$$

given by

$$C(i) \cong \mathbf{1} + \textstyle\sum_{j \in I} \frac{\partial P_j}{\partial i}(D) \times C(j)$$

where $\frac{\partial P_j}{\partial i}$ is the <u>Jacobian</u> of P.

Research Themes

- Integration of programming languages and logical systems.
- Reasoning principles and computation by induction and coinduction.
- Algebraic model theory and its applications.
- Induction-recursion and universes in type theory.
- Programming with computational effects and control operators.

Areas

Algebra – Categories – Compilers
Logic – Semantics – Languages – Types