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
- 1960s Algol Simula
- 1970s Pascal Smalltalk
- 1980s ML
- 1990s Standard ML
- 2000s Java, Scala
- 2010s Haskell 2010

Coq, Agda

S-expressions lists

ADTs

GADTs IFs

Symbolic Expressions

S	::=	a	(atoms)
		S . S	(pairs)

Symbolic Expressions

S ::= a (atoms) | S.S (pairs)

Binary Trees

Specification:

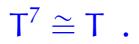
Т	::=	•	(nil)
		Т, Т	(cons)

Semantics:

 ${\mathcal T}\,\cong\, {\mathbf 1}+{\mathcal T} imes {\mathcal T}$

Seven Trees in One

Claim I. There is a bijective program of constant complexity



Seven Trees in One

Claim I. There is a bijective program of constant complexity

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

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.
$$T^7 = T^6 - T^5$$

 $= T^5 - T^4 - T^5$
 $= -T^4$
 $= -T^3 + T^2$
 $= -T^2 + T + T^2$
 $= T$

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 .

lf

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

in the theory of rings

then,

for the data type $D \cong p(D)$, there is a bijection of constant complexity

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

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 $\mathbb{N}[x_1, \dots, x_m]$ modulo $p_1 = q_1, \dots, p_n = q_n$.
- 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

 $\mathsf{T}\,\cong\,\mathbf{1}+\mathsf{T}^2$

there is a *primitive recursive* bijection

 $\mathsf{T}\cong\mathsf{T}^*$.

The Arithmetic of Types

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

The Arithmetic of Types

- In the type theory of + and ×, type isomorphism is axiomatised by the laws of arithmetic (*i.e.* commutative semiring structure).
- ► In the type theory of × and ⇒, type isomorphism is axiomatised by the laws of arithmetic; *i.e.* the commutative monoid laws of × and the laws of exponentiation:

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

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

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

The proof uses the lemma:

 $A \times D \cong C \times B \qquad U \times V \cong X \times Y$

 $V \Rightarrow [(U \Rightarrow A) + (U \Rightarrow B)]$ $\times Y \Rightarrow [(X \Rightarrow C) + (X \Rightarrow D)]$ \cong $Y \Rightarrow [(X \Rightarrow A) + (X \Rightarrow B)]$ $\times V \Rightarrow [(U \Rightarrow C) + (U \Rightarrow D)]$

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

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

The proof uses the lemma:

 $A \times D \cong C \times B \qquad U \times V \cong X \times Y$

 $V \Rightarrow [(U \Rightarrow A) + (U \Rightarrow B)]$ $\times Y \Rightarrow [(X \Rightarrow C) + (X \Rightarrow D)]$ \cong $Y \Rightarrow [(X \Rightarrow A) + (X \Rightarrow B)]$ $\times V \Rightarrow [(U \Rightarrow C) + (U \Rightarrow D)]$

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

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

The operations down and up

The operations down and up:

down₁(t.t', Γ) = (t, (2,t'):: Γ) down₂(t.t', Γ) = (t', (1,t):: Γ)

The operations down and up:

$$down_{1}(t.t', \Gamma) = (t, (2,t')::\Gamma)$$
$$down_{2}(t.t', \Gamma) = (t', (1,t)::\Gamma)$$
$$up(t, (1,t')::\Gamma) = (t'.t, \Gamma)$$

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

The operations down and up:

down₁(t.t',
$$\Gamma$$
) = (t, (2,t'):: Γ)
down₂(t.t', Γ) = (t', (1,t):: Γ)

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

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

and their types:

$$\begin{array}{l} \text{down}: \mathbf{2} \rightarrow \mathsf{T} \times C \rightarrow \mathsf{T} \times C \\ \text{up}: \mathsf{T} \times C \rightarrow \mathsf{T} \times C \\ \text{where } C = (\mathbf{2} \times \mathsf{T})^* \end{array}$$

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

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

we have the operations

$$\label{eq:constraint} \begin{split} & \text{down}^{(k)}: \ A_k \to D \times C \to D \times C \qquad (k \in K) \\ & \text{up}: D \times C \to D \times C \end{split}$$

Mathematical Structure of ADTs Navigation

For

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

we have the operations

down^(k): $A_k \rightarrow D \times C \rightarrow D \times C$ ($k \in K$) up : $D \times C \rightarrow D \times C$

where

 $C = \left(p'(D) \right)^*$

with

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

the <u>derivative</u> of p.

► The type descriptions $\frac{down^{(k)}: A_k \rightarrow D \times C \rightarrow D \times C}{down^{(k)}: but not precise.}$

Dependent Types

- ► The type descriptions $\begin{array}{l} \text{down}^{(k)}: A_k \rightarrow D \times C \rightarrow D \times C \quad (k \in K) \\ \text{are adequate, but not precise.} \end{array}$
- ► For precision:

down : $(k : K) \rightarrow A_k \rightarrow D^{A_k} \times C \rightarrow D \times C$ dependent types are needed.

Generalised ADTs

Exponential lists.

Lexp α

= nil: Lexp α

 $| \text{ cons}: \alpha \times \text{Lexp}(\alpha \times \alpha) \rightarrow \text{L} \exp \alpha$

Generalised ADTs

Exponential lists. Lexp α = nil: Lexp α $| \operatorname{cons} : \alpha \times \operatorname{\mathsf{Lexp}}(\alpha \times \alpha) \to \operatorname{\mathsf{Lexp}} \alpha$ Generates [], $[a_1]$, $[a_1, (a_2, a_3)]$, $[a_1, (a_2, a_3), ((a_4, a_5), (a_6, a_7))]$, . . . *i.e.*, lists of $2^n - 1$ elements.

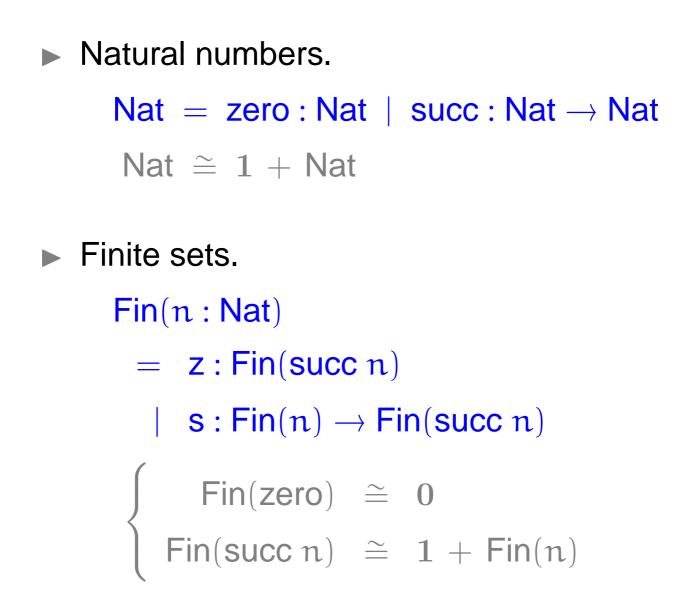
 $\mathsf{Lexp}\,\alpha \ \cong \ \mathbf{1} \ + \ \alpha \times \mathsf{L}\exp(\alpha \times \alpha)$

Inductive Families

Natural numbers.

Nat = zero : Nat | succ : Nat \rightarrow Nat Nat \cong 1 + Nat

Inductive Families



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

```
Lam(n : Nat)
```

- = var : Fin(n) \rightarrow Lam(n)
 - $| \quad \text{apl}: \text{Lam}(n) \times \text{Lam}(n) \rightarrow \text{Lam}(n)$
 - $| \quad \text{abs}: \text{Lam}(\text{succ} \ n) \rightarrow \text{Lam}(n)$

 $\text{Lam}(\mathfrak{n})$

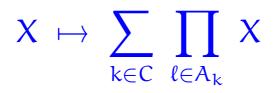
 \cong Fin(n) + Lam(n) × Lam(n)

+ Lam(succ n)

Generalise from polynomial constructions:

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

► Generalise from polynomial constructions:



to *multivariate power-series* constructions:

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

► Generalise from polynomial constructions:

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

to *multivariate power-series* 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.

► Generalise from polynomial constructions:

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

to *multivariate power-series* constructions:

$$\left(X_{i} \right)_{i \in I} \; \mapsto \; \left(\sum_{k \in C_{i}} \prod_{\ell \in A_{k}} X_{\alpha(k,\ell)} \right)_{i \in I}$$

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

The type of the navigation context for

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

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

given by

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

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