SHEFFIELD 1987

From Cayley Diagrams to Computer Aided Design

Meurig Beynon
Department of Computer Science
University of Warwick

A Cayley diagram is a particularly beautiful example of a geometrical object which has an independent algebraic interpretation. On the one hand, it is a graph with a high degree of symmetry; on the other, it is a finite automaton which captures the structure of a group presentation. An ability to identify and reason simultaneously about two or more semantic models of a single object, such as a Cayley diagram, is characteristic of mathematical - perhaps even human - intelligence.

Examples of objects which admit extraordinarily diverse abstract models - VLSI circuits, systems of communicating processes, pictorial images - abound in computing applications, and the problem of representing and manipulating such abstractions is of central importance in modern computer science.

This talk describes how the development of an interactive system for displaying and manipulating Cayley diagrams has suggested a "definitive" (ie definition-based) programming paradigm within which several forms of abstraction commonly encountered in Computer Aided Design can be represented.

Preface: Computer Science as a mathematical discipline

Problem: representation and manipulation of Cayley diagrams

What is? Finite state machine recogniser for the set of representations of the identity in a finite group specified via a particular presentation (set of generators + relations). Vertices represent the elements of the group (once the identity has been selected), and edges the effect of multiplication by the generators.

How to draw a CD? Automatic layout of a Cayley Diagram algorithm? Functional / equational description of a Cayley diagram? Interactive dialogue to locate the vertices, specify the diagram?

What use / how to use a CD? Aesthetic / practical concern: must be readable. Want to be able eg to identify orders of the elements, to construct and display the group table, to relate the vertices to group elements, to change the generators, to label vertices by a permutation representation.

Solution adopted in ARCA (named after ARthur CAyley): use a *definitive notation*

Definitive notation = notation based upon definition of variables

Underlying algebra of values, and various modes of abstraction in defining variables

(etc)
Example of use - in the old format

Discussion of the new approach, using an auxiliary definitive notation.
Templates and homomorphisms of the underlying algebra

Analysis: to involve consideration of abstract merits of definitive notations
(with reference to alternative paradigms for graphics etc)
and range of possible further applications (eg DoNaLD example)

Points to make:

Nature of what has been done in ARCA has more to do with data representation than with manipulation. How to create an environment for manipulating groups and Cayley Diagrams, rather than a suite of clever algorithms. Note that this wouldn't really have resulted from alternative approaches.

Generalisability of principles: choice of algebra. DoNaLD contrast. Virtues of definitive notation for interaction and abstraction. Application to communicating systems of processes in LSD.

Future applications: CAD and data bases. Where limitations, and connection with other programming paradigms.

May I first thank you for the invitation to present a joint seminar to the Mathematics and Computer Science Departments. As some of you may know, my own roots are very firmly based in mathematics, and I have retained a strong interest in pure mathematics despite - if that is the right word - 12 years in a Computer Science Department. The relationship between Mathematics and Computer Science has been very much in my mind over recent weeks, and I think it will be helpful to say a few words on that general topic, since it bears upon the particular subject I have chosen to address.

I - like many others, I suspect - have long considered the question of what Computer Science can properly be regarded to comprise. For most of us, this is an issue much complicated by diverse influences over the 25 years since the subject was first conceived. There are those in better established fields of scientific research who have disparaged Computer Science, and those within the subject who have - for their own good reasons - wished to represent the subject as essentially non-mathematical in nature. The enormous commercial pressures which have surrounded the development of industrial computing, and the advent of popular recreational software, have also played their part in obscuring the profound problems at the core of a young and as yet undeveloped discipline.

In seeking to understand what constitutes "computer science", I find it helpful to return to the work of pioneering figures in the subject, such as Backus, McCarthy and Minsky who were involved about 1960 in developing the first applications of computers. Perhaps simplistically, I think of the subject which they initiated at that time as centrally concerned with two related problems viz the representation and manipulation of data. Of course, neither of these concerns was entirely alien to mathematics: on the contrary, the choice of a symbolism and the form of an algorithm had long been recognised as very significant in mathematical computation of every kind. The legacy of many great mathematicians is strongly linked with just such computational concerns - Newton and Liebnitz (the calculus), Gauss and Galois (arithmetic with numbers and polynomials), Hilbert and Turing (constructive logic). What distinguishes the study of algorithms for the electronic computer from previous research in mathematical algorithms is the diversity of the representations and manipulations of data which have to be considered, and the formality with which these have to be described. Characteristic of computer science is a need for models of data of many different kinds at many different levels of abstraction, and for efficient methods of translating data between formats suitable for human interpretation and for mechanical processing.

Much more than simply a renaissance in study of conventional algorithms: though it has entailed that (eg identification of class of NP-complete problems, Hendrik Lenstra's recent breakthrough on factorisation techniques).

coherence of the data model cf mathematical notation / conceptual structure
- two motivations: eg need to translate great variety of data manipulations into "binary arithmetic"
  analysis and representation of syntax of a pl in terms of formal languages

need to do more than describe an algorithm relative to an ad hoc data representation
- media for expressing algorithms: programming paradigms (related to choice of machine model)
- representation data for purposes such as "retrieval from a db" etc, where emphasis is no longer on a primitive algorithmic task (related to knowledge representation)
- rel between representation and semantics becomes very important here

Contrast practical success of different paradigms in different applications: theory of formal languages vs theory of databases vs vision. Particular interest here in how best to model data for purposes of interaction.

Illustrations: spreadsheet vs calculator, Data and Reality: way in which conceptual models of data prove to be limited both in practical and philosophical terms.

Computer Science as a mathematical discipline

Pioneers (eg Backus, Minsky, McCarthy) initiated subject around about 1960
Centrally concerned with **data representation** and **manipulation**

Not a new theme:

Arabic vs Roman numerals - arithmetic
Newton / Liebnitz - calculus
Gauss / Galois - algebraic numbers and polynomials
Boole / Frege - propositional and predicate logic
Hilbert / Turing - constructive proof, decideability

What **is** new:

tremendous variety of different sorts of data to be represented
diversity of representations and manipulations to be considered
formality with which have to describe

Characteristic is:

need to model data at many different levels of abstraction
need to translate data between form suitable for human interpretation
and form suitable for mechanical processing

To this extent, computer science could be viewed simply as an enhanced "mathematical theory of algorithms". Advent of CS has enriched this theory: eg Hendrik Lenstra's integer factorisation algorithm, probabilistic primality testing, identification of NP-complete problems

In fact, there is much more than this involved in modern CS.

What else does data representation and manipulation for computers entail?

Instructive to draw parallel with the classical work of Gauss (Disquisitiones Arithmeticae) on the solution of quadratic equations in integers. Gauss described a complete algorithmic solution to
"given a quadratic equation with integer coefficients, what are all its solutions?"
- including cases when there are infinitely many solutions, and when there are none. In the process: Gauss developed a coherent framework (the theory of quadratic forms) within which all the data manipulations required could be expressed. By a fanciful analogy, could see this as "devising a programming language, or equivalently an abstract machine model, within which all the subroutines needed to solve the algorithmic problem could be programmed".

Computer Science **must** be concerned with just such coherence in representations eg because
1 - practical need to translate all data manipulation into a common machine model
2 - identifying a good abstract machine model for a specific class of algorithms
is essential for comprehensibility and effective analysis
3 - need to develop "abstract machine models" for general purpose algorithms

In response to these challenges, have eg:
1 - theory of compilers and data structures
2 - theories such as formal languages and automata for parsing
3 - different programming languages and programming paradigms

These are only the first-order effects. Of particular importance at present are the issues raised by data representation for interactive applications, and those (such as data bases) in which the problem is to develop a unifying framework within which to perform many different related algorithmic tasks. The contrast between a calculator and a spreadsheet illustrates one way in which a difference of emphasis enters here. For such applications, it is arguable that satisfactory mathematical models of data have yet to be discovered - it is easy to identify ways in which the use of relational data base models fail to fulfil practical and philosophical requirements for instance (see Kent: Data and Reality).