1
Introduction

A comparison may help to describe the intention of this book: natural sciences and engineering sciences have their differential and integral calculi. Whenever practical work is to be done, one will easily find a numerical algebra package at the computing center which one will be able to use. This applies to solving linear equations or determining eigenvalues, for example, in connection with finite element methods.

The situation is different for various forms of information sciences as in the study of vagueness, fuzziness, spatial or temporal reasoning, handling of uncertain/rough/qualitative knowledge in mathematical psychology, sociology, and computational linguistics, to mention a few areas. These also model theoretically with certain calculi, the calculi of logic, of sets, the calculus of relations, etc. However, for applications practitioners will usually apply PROLOG-like calculi. Hardly anybody confronted with practical problems knows how to apply relational calculi; there is almost no broadly available computer support. There is usually no package able to handle problems beyond toy size. One will have to approach theoreticians since there are not many practitioners in such fields. So it might seem that George Boole in 1854 [26, 28] was right in saying:

> *It would, perhaps, be premature to speculate here upon the question whether the methods of abstract science are likely at any future day to render service in the investigation of social problems at all commensurate with those which they have rendered in various departments of physical inquiry.*

We feel, however, that the situation is about to change dramatically as relational mathematics develops and computer power exceeds previous expectations. Already in [35] an increasingly optimistic attitude is evident, claiming for an approach with matrices that it '. . . permits an algorithmic rather than a postulational-deductive development of the calculus of relations'. There exists, however, Rudolf Berghammers's RELVIEW system for calculating with relations fairly beyond toy size. This is a tool with which many diverse applications have been handled successfully.

With this text we present an introduction to the field which is at the same time

easily accessible and theoretically sound and which leads to (reasonably) efficient computer programs. It takes into account problems people have encountered earlier. Although mainly dealing with discrete mathematics, at many places the text will differ from what is presented in standard texts on that topic. Most importantly, complexity considerations will not be the center of our focus – notwithstanding the significance of these. The presentation favors rather basic principles that are broadly applicable.

In general, we pay attention to the process of delivering a problem to be handled by a computer. This means that we anticipate the diversity of representations of the basic constituents. It shall no longer occur that somebody arrives with a relation in set function representation and is confronted with a computer program where matrices are used. From the very beginning, we anticipate conversions to make this work together.

We aim to provide three forms of work with relations simultaneously, namely *modelling* with relations, *reasoning* about relations, transforming relational terms considered as program transformation, and, finally, *computing* the results of relational tasks. We are convinced that such support is necessary for an extremely broad area of applications.

PART I

REPRESENTATIONS OF RELATIONS

Part I starts by recalling more or less trivial facts on sets, their elements or subsets, and relations between them. It is rather sketchy and will probably be uninteresting for literate scientists such as mathematicians and/or logicians.

Sets, elements, subsets, and relations can be represented in different ways. We will give hints to the programmer how to work with concrete relations and put particular emphasis on methods for switching from one form to another. Such transitions may be achieved on the representation level; they may, however, also touch a higher relation-algebraic level which we hide at this early stage.

We are going to recall how a partition is presented, or a permutation. Permutations may lead to a different presentation of a set or of a relation on or between sets. Functions between sets may be given in various forms, as a table, as a list, or in some other form. A partition may reduce the problem size when factorizing according to it. We show how relations emerge. This may be simply by writing down a matrix, or by abstracting with a cut from a real-valued matrix. For testing purposes, the relation may be generated randomly.

There is a clash in attitudes and expectations between mathematicians and information engineers. While the first group is interested in *reasoning about properties*, the second aims at *computing and evaluating* around these properties and thus tries to have the relations in question in their hands. Logicians will be satisfied with assertions and sketches in theory fragments, while information scientists try to obtain Boolean matrices and operate on these with machine help.

A point deserving special mention is the language TituRel, which is frequently referred to in this book. It has been developed in parallel with the writing of this book and is a thoroughgoing relational reference language, defined in some syntactical way. It may be used in proofs and transformations or for interpretation in some model and environment.

However, one cannot start a book on a not yet commonly known topic using a

sophisticated supporting language. If one is not sufficiently acquainted with relations, one will simply not be able to assess the merits of the language. This enforces an unconventional approach. We cannot start with the language from scratch and have to say first what a relation, a subset, or a point is like in a rather naïve way. Also, one cannot start a book on relations and relational methods without saying what symmetry and transitivity mean, although this can only be formulated later in a theoretically more satisfactory way. We cannot maintain an overly puristic attitude: the potential reader has to accept learning about *concrete* relations first and switching to the intended pure and abstract form only later.

At several places, however, we will give hints when approaching the intended higher level. At such a level it will be possible to write down terms in the language mentioned and – when given a model and an environment – to evaluate and observe the results immediately (of course only for small or medium-sized finite examples). All of the figures and examples of this book emerged in this way.

To sum up: Part I does not resemble the ultimate aims of the present book; rather, it has been inserted as a sort of 'warm up' – with more advanced remarks included – for those who are not well acquainted with relations.

2

Sets, Subsets, and Elements

Usually, we are confronted with sets at a very early period of our education. Depending on the respective nationality, it is approximately at the age of 10 or 11 years. Thus we carry with us quite a burden of concepts concerning sets. At least in Germany, *Mengenlehre* as taught in elementary schools will raise bad memories on discussing it with parents of school children. All too often, one will be reminded of Georg Cantor, the inventor of set theory, who became mentally ill. At a more advanced level, we encounter a number of paradoxes making set theory problematic, when treated in a naïve way. One has to avoid colloquial formulations completely and should confine oneself to an adequately restricted formal treatment.

The situation does not improve when addressing logicians. Most of them think in just one universe of discourse containing numbers, letters, pairs of numbers, etc., altogether rendering themselves susceptible to numerous semantic problems. While these, in principle, can be overcome, ideally they should nevertheless be avoided from the beginning.

In our work with relations, we will mostly be restricted to finite situations, which are much easier to work with and to which most practical work is necessarily confined. A basic decision for this text is that a (finite) set is always introduced together with a *linear ordering* of its elements. Only then we will have a well-defined way of presenting a relation as a Boolean matrix. When we want to stress this aspect, the set will be called a baseset. Other basesets will be generated in a clear way from already given basesets. Furthermore, we distinguish such *basesets* from their *subsets*: they are handled completely differently when they are transferred to a computer. Subsets of basesets necessarily refer to their baseset. We will be able to denote elements of basesets explicitly and to represent basesets for presentation purposes in an accordingly permuted form. Changing the order of its elements means, however, switching to another baseset. Altogether, we will give a rather constructive approach to basic mathematics – as well as to theoretical computer science.

2.1 Set representation

The sets we start with are (hopefully sufficiently small) finite **ground sets** as we call them. To denote a ground set, we need a name for the set and a list of the different names for all the names of its elements as, for example

$$\begin{aligned}
\text{Politicians} &= \{\text{Clinton,Bush,Mitterand,Chirac,Schmidt,Kohl,Schröder,Thatcher,}\\
&\quad\ \text{Major,Blair,Zapatero}\}\\
\text{Nationalities} &= \{\text{US,French,German,British,Spanish}\}\\
\text{Continents} &= \{\text{North-America,South-America,Asia,Africa,Australia,Antarctica,Europe}\}\\
\text{Months} &= \{\text{Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec}\}\\
\text{GermSocc} &= \{\text{Bayern München,Borussia Dortmund,Werder Bremen,Schalke 04,VfB Stuttgart}\}\\
\text{IntSocc} &= \{\text{Arsenal London,FC Chelsea,Manchester United,Bayern München,}\\
&\qquad\qquad \text{Borussia Dortmund,Real Madrid,Juventus Turin,Olympique Lyon,}\\
&\qquad\qquad \text{Ajax Amsterdam,FC Liverpool,Austria Wien,Sparta Prag,FC Porto}\}.
\end{aligned}$$

There is no need to discuss the nature of ground sets as we assume them to be given 'explicitly'. Since such a set is intimately combined with the order of its representation, we will call it a **baseset**. An easy form of representation in a computer language like HASKELL is possible. One will need a name for the set – the first string below – and a list of names for the elements – here delimited by brackets – giving a scheme for denoting a 'named baseset' as

```
BSN String [String].
```

Here `BSN` starts with an upper case letter – without double quotes – and, thus, denotes a 'constructor'. A constructor has been chosen so as to be able to match against it. We would have to write, for example,

```
BSN "Nationalities" ["US","French","German","British","Spanish"].
```

In this way, paradoxes such as the 'set of all sets that do not contain themselves as an element' cannot occur; these are possible only when defining sets 'descriptively' as in the preceding sentence.

A variant form of the ground set is, for example, the '10-element set Y' for which we tacitly assume the standard element notation and ordering to be given, namely[1]

$$Y = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}.$$

Ordering of a ground set

Normally, a set in mathematics is *not* equipped with an ordering of its elements. Working practically with sets, however, this level of abstraction cannot be maintained. Even when presenting a set on a sheet of paper or on a blackboard, we can hardly avoid some ordering. So we demand that ground sets correspond to *lists* and not just to *sets*. As this is the case, we take advantage of it in so far as we are allowed to choose a favorable ordering of elements of a set. This may depend

[1] When implementing this in some programming language, one will most certainly run into the problem that the elements of the set are integers – not strings.

on the context in which the set is presented. The necessary permutation will then somehow be deduced from that context.

As an example, consider the baseset MonthsS of short month names under the additional requirement that month names be presented alphabetically as in

{Apr,Aug,Dec,Feb,Jan,Jul,Jun,Mar,May,Nov,Oct,Sep}.

The necessary permutation shown as numbers to which position the respective original month name should be sent is

[5,4,8,1,9,7,6,2,12,11,10,3].

'Jan', for example, must for this purpose be sent to position 5. Occasionally, it will be necessary to convert such a permutation back, for which we use the inverse permutation

[4,8,12,2,1,7,6,3,5,11,10,9]

sending, for example, 'Apr' back to its original position 4.

Another example of a ground set is that of Bridge card denotations and suit denotations. The latter need a permutation so as to obtain the sequence suitable for the game of Skat[2]

$$\begin{aligned} \text{CardValues} &= \{\text{A,K,Q,J,10,9,8,7,6,5,4,3,2}\} \\ \text{BridgeColors} &= \{\spadesuit,\heartsuit,\diamondsuit,\clubsuit\} \\ \text{SkatColors} &= \{\clubsuit,\spadesuit,\heartsuit,\diamondsuit\}. \end{aligned}$$

A ground set is an object consisting of a name for the ground set – uniquely chosen in the respective situation – and a list of element names. Handling it in a computer requires, of course, the ability to ask for the name of the set, to ask for its cardinality, and to ask for the list of element names. At this point we do not elaborate this any further.

What should be mentioned is that our exposition here does not completely follow the sequence in which the concepts have to be introduced theoretically. When we show that sets may be permuted to facilitate some visualization, we are already using the concept of relations which is introduced later. We cannot avoid using it here in a naïve way.

Constructing new basesets

Starting from ground sets, further basesets will be obtained by construction, as pair sets, as variant sets, as powersets, or as the quotient of a baseset modulo some equivalence. Other constructions that are not so easily identified as such are

[2] Skat is a popular card game in Central Europe with the same suits as Bridge, but ordered differently as mentioned here; card values range from 7 to Ace only.

subset extrusion and also baseset permutation. The former serves the purpose of promoting a subset of a baseset to a baseset in its own right,[3] while the latter enables us, for example, to present sets in a nice way. For all these constructions we will give explanations and examples later.

2.2 Element representation

So far we have been concerned with the (base)set as a whole. Now we concentrate on its elements. There are several methods of identifying an element, and we will learn how to switch from one form to another. In every case, we assume that the baseset is known when we try to denote an element in one of these forms

— as an element number out of the baseset	`NUMBElem BaseSet Int`
— marked **1**/**0** along the baseset	`MARKElem BaseSet [Bool]`
— as a name out of the elements of the baseset	`NAMEElem BaseSet String`
— as a one-element mark in the diagonal	`DIAGElem BaseSet [[Bool]]`.

First we might choose to indicate the element of a *ground* baseset giving the position in the enumeration of elements of the baseset, as in $\texttt{Politicians}_5$, $\texttt{Colors}_7$, $\texttt{Nationalities}_2$. Because our basesets, in addition to what is normal for mathematical sets, are endowed with the sequence of their elements, this is a perfect way of identifying elements. Of course, we should not try an index above the cardinality which will result in an error.

However, there is another form which is useful when using a computer. It is very similar to a bit sequence and may, thus, be helpful. We choose to represent such an element identification as in Fig. 2.1.

$$\texttt{Politicians}_3 \quad \text{Mitterand} \qquad \begin{array}{r} \text{Clinton} \\ \text{Bush} \\ \text{Mitterand} \\ \text{Chirac} \\ \text{Schmidt} \\ \text{Kohl} \\ \text{Schröder} \\ \text{Thatcher} \\ \text{Major} \\ \text{Blair} \\ \text{Zapatero} \end{array} \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{1} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} \quad \begin{array}{r|c} \text{Clinton} & \mathbf{0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0} \\ \text{Bush} & \mathbf{0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0} \\ \text{Mitterand} & \mathbf{0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0} \\ \text{Chirac} & \mathbf{0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0} \\ \text{Schmidt} & \mathbf{0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0} \\ \text{Kohl} & \mathbf{0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0} \\ \text{Schröder} & \mathbf{0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0} \\ \text{Thatcher} & \mathbf{0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0} \\ \text{Major} & \mathbf{0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0} \\ \text{Blair} & \mathbf{0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0} \\ \text{Zapatero} & \mathbf{0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0} \end{array}$$

Fig. 2.1 Element as a marking vector or as a marking diagonal matrix

Again we see that combination with the baseset is needed to make the column vector of **0**s and **1**s meaningful. We may go even further and consider, in a fully

[3] The literate reader may identify basesets with objects in a category of sets. Category objects constructed generically as direct sums, direct products, and direct powers will afterwards be interpreted using natural projections, natural injections, and membership relations.

naïve way, a partial identity relation with just one entry **1**, or a partial diagonal matrix with just one entry **1** on the baseset.[4]

With these four attempts, we have demonstrated heavy notation for simply saying the name of the element, namely `Mitterand` ∈ `Politicians`. Using such a complicated notation is justified only when it brings added value. Mathematicians have a tendency of abstracting over all these representations, and they often have reason for doing so. On the other hand, those using applications cannot switch that easily between representations. Sometimes, this prevents them from seeing the possibilities of more efficient work or reasoning.

An element of a baseset in the mathematical sense is here assumed to be transferable to the other forms of representation via functions that might be called

`elemAsNUMB, elemAsMARK, elemAsNAME, elemAsDIAG,`

as required. Each of these takes an element in whatever variant it is presented and delivers it in the variant indicated by the name of the function. All this works fine for ground sets. Later we will ask how to denote elements in generically constructed basesets such as direct products (pair sets), powersets, etc.

2.3 Subset representation

We now extend element notation slightly to the notation of subsets. Given a baseset, subsets may be defined in at least six different forms which may be used interchangeably and which are also indicated here in a computer usable form:

— as a list of element numbers of the baseset `LISTSet BaseSet [Int]`
— as a list of element names of the baseset `LINASet BaseSet [String]`
— as a predicate over the baseset `PREDSet BaseSet (Int -> Bool)`
— as element in the powerset of the baseset `POWESet BaseSet [Bool]`
— marked **1**/**0** along the baseset `MARKSet BaseSet [Bool]`
— as partial diagonal matrix on the baseset `DIAGSet BaseSet [[Bool]]`.

Again, the capitalized name is a constructor in the sense of a modern functional language; what follows is the type to indicate the set we draw the subset from. This methods works fine as long as the baseset is ground. In this case we have the possibility of giving either a set of numbers or a set of names of elements in the baseset. One has to invest more care for constructed basesets or for infinite basesets. For the latter, negation of a finite subset may well be infinite, and thus more difficult to represent.

[4] We quote from [77]: ... Moreover, the **0** and **1** reflects the philosophical thought embodied in symbols of dualism of the cosmos in *I Ching* (The Classic of Changes or The Book of Divination) of ancient China. That is, **0** represents
Yin (negative force, earth, bad, passive, destruction, night, autumn, short, water, cold, etc.)
and **1** represents
Yang (positive force, heaven, good, active, construction, day, spring, tall, fire, hot, etc.).

CAMBRIDGE

Cambridge University Press
978-0-521-76268-7 - Relational Mathematics
Gunther Schmidt
Excerpt
More information

Given the name of the baseset, we may list numbers of the elements of the subset as in the first variant or we may give their names explicitly as in the second version of Fig. 2.2. We may also use marking along the baseset as in variant three, and we may, as already explained for elements, use a partial diagonal matrix over the baseset.

$$\text{Politicians}_{\{1,2,6,10\}} \qquad \{\text{Clinton,Bush,Kohl,Blair}\}$$

$$\begin{array}{r}\text{Clinton}\\ \text{Bush}\\ \text{Mitterand}\\ \text{Chirac}\\ \text{Schmidt}\\ \text{Kohl}\\ \text{Schröder}\\ \text{Thatcher}\\ \text{Major}\\ \text{Blair}\\ \text{Zapatero}\end{array}\begin{pmatrix}1\\1\\0\\0\\0\\1\\0\\0\\0\\1\\0\end{pmatrix} \qquad \begin{array}{r}\text{Clinton}\\ \text{Bush}\\ \text{Mitterand}\\ \text{Chirac}\\ \text{Schmidt}\\ \text{Kohl}\\ \text{Schröder}\\ \text{Thatcher}\\ \text{Major}\\ \text{Blair}\\ \text{Zapatero}\end{array}\begin{pmatrix}1&0&0&0&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&0&0&0&0\end{pmatrix}$$

Fig. 2.2 Different subset representations

In any case, given that

```
Politicians = {Clinton,Bush,Mitterand,Chirac,Schmidt,Kohl,
               Schröder,Thatcher,Major,Blair,Zapatero}
```

what we intended to denote was simply something like

```
{Clinton,Bush,Kohl,Blair} ⊆ Politicians.
```

Subsets may be given in either of these variants; we assume, however, functions

```
setAsLIST, setAsLINA, setAsPRED, setAsPOWE, setAsMARK, setAsDIAG
```

to be available that convert to a prescribed representation of subsets. Of course, the integer lists must provide numbers in the range of 1 to the cardinality of the baseset.

Another technique should only be used for sets of minor cardinality, although a computer will handle even medium sized ones. We may identify the subset of X as an element in the powerset $\mathcal{P}(X)$ or $\mathbf{2}^X$. For reasons of space, it will only be presented for the subset {♡,♢} of the 4-element Bridge suit set {♠,♡,♢,♣}.[5]

---

[5] Another well-known notation for the empty subset is $\emptyset$ instead of {}.