Computer Physics Communications 28 (1982) 183–187
North-Holland Publishing Company

COMPUTATION OF OUTER PRODUCTS OF SCHUR FUNCTIONS

Ömer EĞECIOĞLU

Department of Mathematics, University of California San Diego, La Jolla, CA 92093, USA

Received 17 May 1982

PROGRAM SUMMARY

Title of program: Schur

Catalogue number: AAMJ

Program obtainable from: CPC Program Library, Queen's University of Belfast, N. Ireland (see application form in this issue)

Computers upon which the program is operable: Various mini and microcomputers supporting the UCSD Pascal system

Computer: PDP-11/780; *Installation:* UCSD

Operating system: UCSD Pascal system

Programming language used: UCSD Pascal

High speed storage required: 1900 words

Number of bits in a word: 16

Overlay structure: none

Peripherals used: console, disk

Number of cards in combined program and test deck: 498

Keywords: Schur function, outer product, irreducible representation, partition, Ferrers' diagram, standard tableau, skew-tableau, backtracking, depth-first order, balanced tree

Nature of physical problem
To express the (outer) product of an arbitrary number of Schur functions as a linear combination of Schur functions.

Method of solution
A new backtracking algorithm [1] is implemented to generate the partitions that appear in the expansion of a product of Schur functions.

Restrictions on the complexity of the problem
The size of the problem that can be handled by the implementation is restricted by the total number of parts of the input partitions (no more than 255).

Typical running time

$\{5^2\}\{3^22\}$	2 s.
$\{31\}\{2^2\}\{21\}\{1^2\}$	8 s.

(Time indicated includes the disk-write time: 1 and 2 s, respectively.)

Unusual features of the program
The partitions generated by the algorithm are maintained as nodes of a balanced binary tree to minimize list insertion time.

References
[1] J. Remmel and R. Whitney, Multiplication of Schur Functions (to appear).
[2] D.E. Littlewood and A.R. Richardson, Phil. Trans. A 233 (1934) 99.
[3] D.E. Littlewood, The Theory of Group Characters, 2nd ed. (Oxford Univ. Press, London, 1950) p. 94.

LONG WRITE-UP

1. Introduction

The product of two Schur functions $\{\lambda^1\}$ and $\{\lambda^2\}$ of degrees m and n, respectively, can be expressed as a nonnegative integral linear combination of Schur functions $\{\mu\}$ of degree $m+n$:

$$\{\lambda^1\}\{\lambda^2\} = \sum_{\mu \vdash n+m} g^{\mu}_{\lambda^1,\lambda^2}\{\mu\}. \tag{1}$$

The correspondence [2] between the construction of the outer product $\lambda^1 \times \lambda^2$ of two irreducible representations λ^1 and λ^2 of the symmetric group and the (outer) product of the Schur functions determined by these partitions gives the multiplicities of the irreducible constituents of the representation $\lambda^1 \times \lambda^2$, once the expansion (1) is known. Similarly, the coefficient $g^{\mu}_{\lambda^1,\lambda^2,\ldots,\lambda^\tau}$ in the expansion

$$\{\lambda^1\}\{\lambda^2\}\ldots\{\lambda^r\} = \sum_{\mu} g^{\mu}_{\lambda^1\lambda^2,\ldots,\lambda^r}\{\mu\} \tag{2}$$

of the Schur functions $\{\lambda^1\}$, $\{\lambda^2\},\ldots,\{\lambda^r\}$ gives the multiplicity of the corresponding irreducible representation μ in the analysis of $\lambda^1 \times \lambda^2 \times \ldots \times \lambda^r$.

In ref. [2], a combinatorial rule was given to compute the coefficients $g^{\mu}_{\lambda^1,\lambda^2}$ that occur in (1) (See also ref. [3]).

We implement a new method obtained in ref. [1], which makes it possible to multiply an arbitrary number of Schur functions *directly,* and thus obtain the coefficients $g^{\mu}_{\lambda^1,\lambda^2,\ldots,\lambda^r}$ that appear in (2) without excessive computation.

As a surprising by-product of the algorithm, the degrees n_λ of the irreducible representations of the symmetric group can also be computed without much effort in view of the identity

$$\{1\}^n = \sum_{\lambda \vdash n} n_\lambda\{\lambda\}.$$

2. Terminology and the method

A partition $\lambda = (\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_k)$ is pictorially represented via its Ferrers' diagram by taking k left-justified rows of squares where the ith row consists of λ_i squares, $i = 1, 2, \ldots, k$. For example the partition $\lambda = (432)$ is represented as shown in fig. 1. This is also referred to as the *shape* of λ.

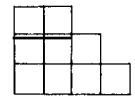

Fig. 1. Representation of the partition $\lambda = (4\ 3\ 2)$.

If $\sum_{i=1}^{k}\lambda_i = n$ then we denote this by $\lambda \vdash n$ and put $|\lambda| = n$. $\mathrm{ST}(\lambda)$ is the collection of all standard tableaux of shape λ and

$$\mathrm{ST}(n) := \bigcup_{\lambda \vdash n} \mathrm{ST}(\lambda).$$

If $\tau \in \mathrm{ST}(\lambda)$ then we set $P(\tau) = \lambda$.

If λ^1, λ^2 are two partitions, the *skew-shape* $F(\lambda^1, \lambda^2)$ is the plane figure (fig. 2).

The *lexicographic filling* $\mathrm{LF}(\lambda^1, \lambda^2)$ of a skew-shape $F(\lambda^1, \lambda^2)$ is obtained by filling the cells of the underlying skew-shape with the integers $1, 2, \ldots, |\lambda^1| + |\lambda^2|$ (in that order) starting from the upper left cell and proceeding row-wise down the figure. These definitions clearly extend to an arbitrary number of partitions $\lambda^1, \lambda^2, \ldots, \lambda^r$. If

$$\{\lambda^1\}\{\lambda^2\}\ldots\{\lambda^r\} = \sum_{\mu \vdash N} g^{\mu}_{\lambda^1,\lambda^2,\ldots,\lambda^r}\{\mu\},$$

where $N = |\lambda^1| + |\lambda^2| + \cdots + |\lambda^r|$ then we can compute the coefficients $g^{\mu}_{\lambda^1,\lambda^2,\ldots,\lambda^r}$ by the following version of the result of ref. [1]:

Theorem. Let $\mathrm{LF}(\lambda^1, \lambda^2, \ldots, \lambda^r)$ be the lexicographic filling of the skew-shape $F(\lambda^1, \lambda^2, \ldots, \lambda^r)$

Fig. 2. Plane figure of the skew-shape $F(\lambda^1, \lambda^2)$.

Fig. 3. $\mathrm{LF}(\lambda^1, \lambda^2)$ with $\lambda^1 = (2\ 1)$ and $\lambda^2 = (1^2)$.

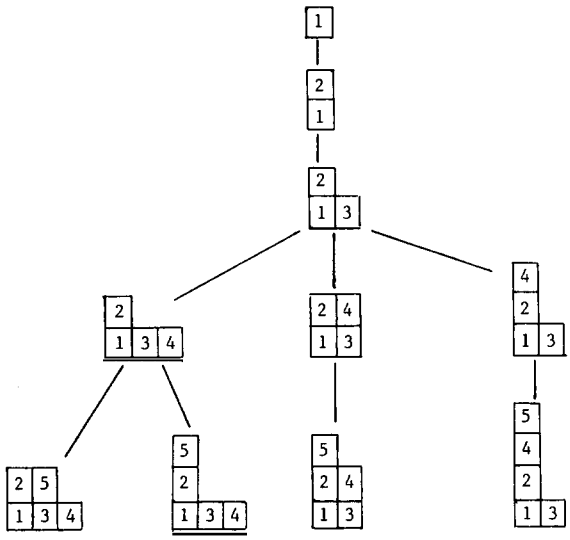

Fig. 4. Tableau-tree in depth-first order.

and let $R(N)$ be the collection of all $\tau \in \mathrm{ST}(N)$ with the following properties:

(i) Whenever $j+1$ is immediately to the right of j in $\mathrm{LF}(\lambda^1, \lambda^2, \ldots, \lambda^r)$, then $j+1$ appears south–east of j in τ.

(ii) Whenever k is immediately below j in $\mathrm{LF}(\lambda^1, \lambda^2, \ldots, \lambda^r)$, then k appears north–west of j in τ.

Then

$$\{\lambda^1\}\{\lambda^2\}\ldots\{\lambda^r\} = \sum_{\tau \in R(N)} \{P(\tau)\}.$$

As an example take $\lambda^1 = (2\ 1)$, $\lambda^2 = (1^2)$. Then $\mathrm{LF}(\lambda^1, \lambda^2)$ is as shown in fig. 3.

To expand $\{\lambda^1\}\ \{\lambda^2\}$, we construct all standard tableaux in ST(5) such that

2 appears N–W of 1,

3 appears S–E of 2,

5 appears N–W of 4.

This is done by constructing the tree, depicted in fig. 4, in depth-first order. Taking the underlying shapes of the terminal nodes of this tree, we have the expansion

$$\{2\ 1\}\{1^2\} = \{3\ 2\} + \{3\ 1^2\} + \{2^2\ 1\} + \{2\ 1^3\}.$$

3. Computer implementation

3.1. An overview

The program generates and maintains two trees. One of these trees (called the tableau-tree) is built to create the restricted family $R(N)$ of standard tableaux, while the other tree (called the list-tree) is a balanced [4] binary tree which keeps track of the list of partitions that have been created by the algorithm.

The parameters $\lambda^1, \lambda^2, \ldots, \lambda^r$ are read from the input file and two global arrays are constructed to record the constraints on the allowable locations of each integer j, $1 \le j \le N$ which determine the tableaux in the collection $R(N)$. This is accomplished without actually generating the objects $F(\lambda^1, \lambda^2, \ldots, \lambda^r)$ and $\mathrm{LF}(\lambda^1, \lambda^2, \ldots, \lambda^r)$ themselves.

The tableaux-tree is constructed by the procedure CONSTRUCT recursively using backtracking. Since only the shape of a tableau that appears as a terminal node is required, the data structure employed keeps track of the sizes of the parts of the tableaux being generated rather than the tableaux themselves. The allowable rows for branching at each particular node of the tableau-tree are passed as arguments in the next call to procedure CONSTRUCT. Once a terminal node of the tableau-tree is reached, the rowlengths are converted into a string to facilitate data manipulation at later stages of the program. The encoded partition is then inserted into the list-tree which maintains the list of partitions previously generated together with the number of times each one was generated (multiplicity). If the partition already appears in the current list-tree then its multiplicity is incremented by one and control returned to the main program. Otherwise a new leef node is created and the tree is balanced if necessary.

After the generation of the collection $P(R(N))$, the list-tree is traversed in depth-first order and the list of generated partitions together with their multiplicities are written to a user-specified output file.

3.2. Description of the procedures

The names and brief descriptions of the procedures used are as follows:

DECLARATIONS
In this procedure, the parameters that are used by the main program SCHUR and their data types are declared. The problem size is controlled by a single constant declared as MAXPROBSIZE.

SCHUR
This is the main driver of the program.

OUTPUT
Traverses the list-tree in depth-first order. The multiplicity, the number of parts and the parts themselves of each partition are written to an output file.

CONSTRUCT
This procedure constructs the collection $P(R(N))$. The input arguments are as follows:

(i) The element j to be added to the present tableau.
(ii) The minimum and the maximum indices of the rows that j can be attached to as determined by the elements in $\mathrm{LF}(\lambda^1, \lambda^2, \ldots, \lambda^r)$ which lie immediately above and to the left of j (if any). Thus the present node in the tableau-tree can have at most (maximum − minimum) subtrees.
(iii) The parts of the underlying partition which can be incremented by one without violating the monotonicity condition of the rowlengths. The indices of these rows are kept in the array GOOD_ROWS.
(iv) The array ROW_LENGTHS which stores the parts of the underlying partition.

These variables must be initialized by the main driver and passed as value parameters.

INSERT
Inserts a partition into the list-tree and balances the tree if necessary.

SET_CONSTRAINTS
Reads the input from the specified input file and initializes the variables. The two global arrays that are used to determine $R(N)$ are constructed as follows:

DOWN_RIGHT $[k]: = j$ if j is immediately above k in the lexicographic filling of the associated skew-shape,
DOWN_RIGHT $[k]: = 0$ if no such entry exists;
UP_LEFT $[j+1]: = j$ if j is immediately to the left of $j+1$ in the lexicographic filling in the associated skew-shape,
UP_LEFT $[j+1]: = 0$ if no such entry exists.

These assignments are made as the input partitions are read.

UPDATE_TABLEAU
The current shape ROW_LENGTHS is updated and the available rows for the insertion of the next element is recorded. The array HEIGHT marks the indices of the rows occupied by the numbers that are already in the tableau. In conjunction with the information in the arrays DOWN_RIGHT and UP_LEFT, the maximum and the minimum indices of the rows available for the next element are determined.

3.3. I/O format

The user is prompted to enter the names of the input and output files. The default is the console device. The input file is expected to be a linear list where the input partitions are formated as

$$k\ \lambda_1\ \lambda_2\ \ldots\ \lambda_k\ \langle \mathrm{cr} \rangle,$$

where the entries are separated by blanks and $\langle \mathrm{cr} \rangle$ denotes carriage return.

The partitions generated by the program are written to the specified output file in the form

$$m_\mu\ l\ \mu_1\ \mu_2\ \ldots\ \mu_l.$$

Here m_μ is the multiplicity of the Schur function $\{\mu\}$ in the expansion of the product of the input Schur functions.

If the output device is other than disk or the console then the *write* and *writeln* statements that appear in procedure OUTPUT should be modified accordingly.

3.4. Performance

The complexity of the algorithm is determined by the list-insertion and maintenance portion of the program which is of $\mathcal{O}(n \log n)$, where n is the number of terminal nodes generated in the tableau-tree. The main body of the program which actually generates the collection $P(R(N))$ can be altered to perform in linear time by judicious use of linked-list structures to pass the list of available rows in the recursive calls to procedure CONSTRUCT. Moreover, by suitably encoding the partitions created, bucket-sort can be applied to generate the output list in linear time. Therefore theoretically the algorithm can be implemented in linear time. For large size problems though, the exponential growth of the partition function renders bucket-sort quite impractical in terms of space requirements.

4. Remarks

The correctness of this implementation has been checked extensively by making use of the tables of outer products of Schur functions that appear in ref. [5].

The standard reference of the Pascal programming language is ref. [6]. The UCSD Pascal system is described in detail in ref. [7].

Acknowledgements

Special thanks go to Mark Erikson for coding and fine-tuning the present version of program Schur. The Electric Engineering and Computer Science department of UCSD kindly provided access to their computational facilities.

References

[1] J. Remmel and R. Whitney, Multiplication of Schur Functions (to appear).
[2] D.E. Littlewood and A.R. Richardson, Phil. Trans. A 233 (1934) 99.
[3] D.E. Littlewood, The Theory of Group Characters, 2nd ed. (Oxford Univ. Press, London, 1950) p. 94.
[4] N. Wirth, Algorithms + Data Structures = Programs (Prentice-Hall, Englewood Cliffs, NJ, 1976).
[5] B.G. Wybourne, Symmetry Principles of Atomic Spectroscopy (John Wiley, New York, 1970).
[6] K. Jensen and N. Wirth, Pascal User Manual and Report (Springer-Verlag, New York, 1975).
[7] K.L. Bowles, Beginner's Manual for the UCSD Pascal System (McGraw-Hill, Peterborough, New Hampshire, 1979).