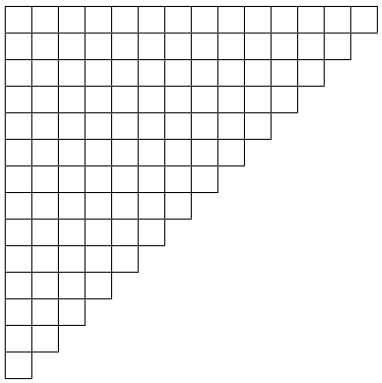
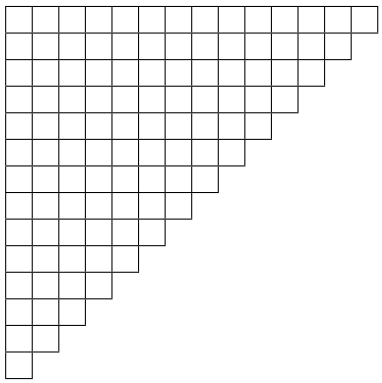




SCHUR

An interactive programme
for
calculating properties
of
Lie groups
and
symmetric functions

Steven M. Christensen
P. O. Box 16175
Chapel Hill, NC 27516 USA
Phone/FAX: 919-967-9853
Email: stevc@smc.vnet.net
WEB: http://scm.vnet.net/Christensen.html

The information in this manual is valid for version 5.3 of Schur.

Preface

Group theory, and the theory of symmetric functions, continues to make major inroads into problems in chemistry, mathematics and physics. This is natural in a wide range of fields of study where symmetry is exploited. Numerous examples involving the full range of Lie groups, and their extensions, can be drawn from such diverse areas as ligand field theory, electronic states of molecules, atomic physics, nuclear models, particle physics and string theory. In mathematics, statistics and economics symmetric functions are at the foundations of much combinatorial analysis while to physicists they are making almost unexpected appearances in topics such as the quantum Hall effect and the KdV equation for solitons.

The algorithms required to perform calculations are often quite complex. While simple examples may be done with pencil and paper the student and professional alike soon find the labour required for more detailed problems almost, if not, beyond human endeavour. Such calculations often require such a mastery of a vast number of algorithms that there is almost no time left for the problem at hand. SCHUR has been developed to permit the user to carry out complex calculations reliably but without requiring the user to be familiar with the many diverse algorithms or any particular programming language. It has been created as a tool for both learning and research. It provides an environment for the student to do independent learning and for the teacher wishing to create realistic examples.

Many of the commands in SCHUR have arisen from direct application to research problems. These have included applications to the study of Hecke algebras, the interacting boson model in nuclear physics, the normal forms for tensor polynomials, symplectic models of nuclei etc. As a consequence of this experience it has been possible to identify many of the requirements of users. Much thought has gone into the analysis of input and output and the use of unequivocal notation. No software package is universal in its application and SCHUR is no exception. Within its domain there are a vast range of problems that can be tackled and there are equally well a vast range of problems that will forever lie outside its possibilities for reasons of time and memory. There are problems that are simple to state but which would require times many times greater than the age of the universe.

Additional information, downloadable papers, examples, etc relating to SCHUR may be found at the WEB site:- http://www.phys.uni.torun.pl/∼bgw/

It is hoped that the user will find SCHUR both useful and enjoyable. The quotation on page 11 is with the kind permission of Anita Brookner.

B.G.Wybourne
Instytut Fizyki, Uniwersytet Mikołaja Kopernika
ul. Grudziądzka 5/7
87-100 Toruń Poland

CONTENTS

TABLES

0

Introducing SCHUR

What is SCHUR?

■ 0.1 What is SCHUR ?

SCHUR©is an interactive package for calculating properties of representations of Lie groups and yet it is much more. SCHUR has been designed to answer questions of relevance to a wide range of problems of special interest to chemists, mathematicians and physicists. These include not only problems directly relating to Lie groups but also many properties of symmetric functions.

As the name of the package suggests much of the structure of SCHUR derives from the pioneering work of the mathematician Issai Schur on symmetric functions and the representation theory of matrix groups. Many persons find in their work they need specific knowledge relating to some aspect of Lie groups or symmetric functions and yet do not wish to be encumbered with complex algorithms. The objective of SCHUR is to supply results with the complexity of the algorithms fortunately hidden from view. The user should not only be able to obtain results but should be able to use SCHUR effectively as a scratch pad, obtaining a result and then using that result to derive new results in a fully interactive manner. In using SCHUR interactively the user should be able to exploit the many commands built into SCHUR and to define his or her own command structures. The range of possible problems that can be attacked by SCHUR is largely limited by the ingenuity of the user.

■ 0.2 What can SCHUR do ?

Among the many tasks amenable to attack by SCHUR are the following:

1. The calculation of Kronecker products for all the compact Lie groups and for the ordinary and spin representations of the symmetric group. Not only for individual irreducible representations but also lists of irreducible representations. List handling is a general feature of SCHUR.
2. The calculation of branching rules with the ability to successively branch through a chain of nested groups.
3. The calculation of the properties of irreducible representations such as dimensions, second-order Casimir and Dynkin invariants, the trace of the $n-$th order Casimir invariants and the conversion between partition and Dynkin labelling of irreducible representations.
4. The handling of direct products of several groups.
5. The computation of a wide range of properties related to Schur function operations such as the Littlewood-Richardson rule, inner products, skew products, and plethysms as well as the inclusion of commands for generating the terms in infinite series of Schur functions up to a user defined cutoff.
6. The computation of the properties of the symmetric $Q-$functions with respect to operations such as the analogous Littlewood-Richardson rule, skew and inner products.
7. The standardisation of non-standard representations of groups by the

use of modification procedures.

8. Calculation of properties of the classical symmetric functions including expansions between various types of symmetric functions.
9. Kronecker products, plethysms and branching rules involving the non-compact groups $Mp(2n)$, $Sp(2n, R)$, $SO^*(2n)$ and $U(p, q)$.

Among the special features of SCHUR are:

1. All operations can be made on lists of irreducible representations and not just single irreducible representations.
2. Sequences of instructions may be set as functions (which may be saved on disk) allowing easy extension of SCHUR to implement user defined rules.
3. Results of a session with SCHUR may be saved as a logfile for future record or editing.
4. Over 200 commands allow a wide variety of applications of SCHUR.
5. SCHUR can be a valuable tool in the teaching of the properties of groups as students and teachers can readily create examples. Taken with this manual it can be used as a self-paced learning tool.
6. SCHUR can be used as a research tool in many studies.

■ 0.3 What has SCHUR done ?

SCHUR has found important applications in diverse research topics such as,

a . Constructing character tables for the Hecke algebras $H_n(q)$ of type A_{n-1}.

b . Symmetry properties of the Riemann tensor.

c . Group properties of the Interacting Boson Model of nuclei.

d . Non-compact group properties such as branching rules and Kronecker products.

e . Problems in supersymmetry.

f . Evaluation of the properties of one- and two-photon processes in rare earth ions.

g . Symplectic models of nuclei.

i . Studies of the mathematical properties of the exceptional Lie groups.

j . Studies of the symmetric functions such as Schur functions, Q-functions and Hall-Littlewood polynomials.

k . Expansion of powers of the Vandermonde determinant in the quantum Hall effect.

l . Discovery of new $S-$function identities.

m . Discovery of new identities for plethysms in the non-compact group $Sp(2n, R)$.

■ 0.4 Working through the Manual

Every user of a new software package is keen to get started even before reading the manual. SCHUR is a relatively sophisticated package and to gain maximum benifit from the package it is absolutely essential that the intending user first read at least some relevant sections of the manual. Different users will come to SCHUR with differing backgrounds and a uniform approach to the manual will not be suitable for all users. All users should make a careful reading of Chapter One on the important issues of Input/Output in SCHUR. Those familiar with the concepts of partitions of integers should have no difficulties. Others may wish to first read relevant sections of Chapters Two and Three and then return to Chapter One. Only then should the user move on to the tutorials. These tutorials have been designed to take the user through SCHUR systematically starting with the easier sections first and ending up with an advanced tutorial where the user constructs his or her own defined functions for carrying out complex tasks.

For every complex question there is a simple answer
— and it's wrong.
— H. L. Mencken

1

Basics of SCHUR

Input and Output

Introduction

In this chapter we discuss the basic structure of SCHUR especially in regard to input and output issues. It is essential that the user becomes familiar with the integer notation and general syntax used in SCHUR. A careful reading of this chapter should be made before attempting to use SCHUR. The user who has patience, and who resists the temptation to start using SCHUR without reading the manual, will be well rewarded.

1.1 Input of Lists to SCHUR

The input of SCHUR consists of commands and integers while the output consists of written statements and integers enclosed in a variety of brackets. SCHUR is case independent so that upper and lower case letters may be freely used in all input statements. Spaces have no significance and may be inserted anywhere for the sake of clarity. Spaces, or commas, are mandatory between entries of the same type as for example between two commands or two numbers. Nevertheless spaces are forbidden inside an entity. For example, within a command or a number. A line of input is up to 120 characters long. The ampersand "&" is reserved for use as a sign of a continued line so in practice input may be continued over as many lines as desired. Every line of input is followed by a carriage return (henceforth indicated as <CR>). Up until <CR> the line may be edited but after <CR> no further editing is possible. The user is free to use brackets chosen from the set "{, }, (,), <, >" in any desired order or combination, to clarify input which can be convenient in deeply nested command sequences however SCHUR does no checking on inputted brackets and treats them in a completely neutral manner. The brackets "[,]" are reserved for use in the DPMode and may not be used otherwise.

SCHUR knows only of the existence of integers and hence all input and output of numbers is in the form of integers. Most frequently input into SCHUR involves lists of numbers. Each item in a list will normally involve up to five distinct sub-items entered in the prescribed order

$$sign \quad multiplicity \quad prefix \quad partition \quad label$$

The *sign* will always be "+" or "−". If the item is the *first* item in the list then the "+" is discretionary whereas "−" is always compulsory.

The *multiplicity* is the multiplier associated with the item and is always a positive integer. If $multiplicity = 1$ then it is discretionary and would normally be omitted. In normal usage if $multiplicity \geq 10$ the multiplicity should be *preceded* by an exclamation mark "!" (e.g. !137). If the multiplicity is present it *must always* be terminated with a period "." (e.g. !137.).

The *prefix* will normally be used only in the case of indicating spin representations for which the letter "s" is used.

The *partition* is one of the most important objects in SCHUR. Users unfamiliar with the concepts of partitions of integers should read §2.1 before

proceeding further. In reading in partitions SCHUR is unconcerned as to the order of the integers making up the partition. In normal usage SCHUR assumes that most partitions inputted will have parts ≤ 9 in which cases the integers may be simply entered together with no spaces required. Thus the partition "9 4 3 2 1" could be entered simply as 94321. A partition containing a negative part, say "$9-4321$", is entered with a "~" replacing the "$-$" (i.e. as 9~4321). If a part of a partition is ≥ 10 it is entered prefixed with the exclamation sign and the part terminated by a space or a non-digit. Thus "137 29 10 9 4 1" could be entered as !137!29!10 941, note the important space separating the 10 from the 941 . Frequently the user will encounter partitions involving several equal parts. In these cases the user has the option to either enter the repeated parts or if say the integer i occurs n times ($n \leq 9$) it may be entered as $i\hat{}n$. The partition "9 9 9 9 4 4 3 3 3 3 3 3 2 2 1 1" could be entered either as 9999443333332211 or as 9^44^23^62^21^2 or perhaps with greater clarity as 9^4 4^2 3^4 2^2 1^2. In a similar manner the partition "12 12 12 11 11 999921" could be entered as !12^3!11^29^421 or for clarity as !12^3 !11^2 9^4 21 . In rare cases the user may need to input a partition having a part i which is repeated $n \geq 10$ times. In that case the parts i may be entered as $i\hat{}!n$. Thus the partition "9 9 9 9 9 9 9 9 9 9 9 2 2 2 1" would be entered as 9^!12 2^3 1 , note the important space following the number !12 . Finally we note that inputting !12^!13 !10^!31 9^3 21 would correspond to the partition $12^{13}10^{31}9^{3}21$. SCHUR accepts partitions into a maximum of 100 parts

In reading in partitions SCHUR is unconcerned as to the order of the integers making up the partition with one exception - it expects that for partitions have more than one part the last part will be non-zero. Thus entering 3202001 and 3202001000 will both be seen by SCHUR as 3202001. In other words trailing zeroes are ignored but not zeroes followed by a non-zero integer. A particular application involving trailing zeroes is discussed in Chapter Five.

The *label* is a single character, most commonly a "+", "$-$" or a "#". They are normally discretionary and would be omitted in constructing an item. They become compulsory only in the case of certain representations of the orthogonal, special orthogonal, symmetric and alternating groups.

As mentioned earlier lines of input are limited to 120 characters and the ampersand & is used to continue lines. The ampersand is placed *after* the *sign* (*not* after a *label*) "+" or "$-$" of the item that is to appear on the next line after making the <CR> .

Input normally requires no use of brackets. The exception is in the inputting of lists in the DPMode of SCHUR where square brackets "[ ]" are compulsory for enclosing items belonging to lists. In the DPMode the items of a list are of the form

$$sign \quad multiplicity[groupitem * groupitem * \ldots * groupitem]$$

where the *sign* and *multiplicity* are exactly as earlier described with the exception

that *if* the *multiplicity* is placed prior to the "[" bracket then it may take the form of any integer up to up to *MaxInt* and must not be preceded by "!" nor followed by either a "." or a space " ". The number of *groupitems* is equal to the number of groups set with each *groupitem* being separated by a compulsory "∗". Each *groupitem* is characterised by a:-

$$prefix \quad partition \quad label$$

with each sub-item defined and specified exactly as before.

1.2 Output of Lists from SCHUR

The usual action of SCHURfollowing the input of a set of commands either generating a list or acting upon a list, or lists, is to produce an output to screen involving an echoing of the input and output statements and a new list. The new list is output in a manner fitting with the types of groups set or functions that are indexed by partitions. The output list may involve many items spanning many lines. The items in the output list will normally be of the form:-

$$sign \quad multiplicity \quad letter \quad bracket \quad outputitem \quad bracket \quad label$$

The *sign, multiplicity* and *label* have the same meanings as for the input except that the multiplicity always appears as an integer of any magnitude up to the *MaxInt* of the installation. The *letter* (m, e, f, h, p, Q or P) only appears when monomial−, elementary−, forgotten−, homogeneous−, power sum−, $Q-$ or $P-$symmetric functions are being output.

The *outputitem* normally involves a possible *prefix* and a *partition* both as described earlier. The partitions will normally be output in power notation with appropriate spaces with no need for exclamation marks to indicate integers ≥ 10. The *outputitem* is enclosed in brackets appropriate to the nature of the *outputitem*. The brackets used to enclose particular types of *outputitems* are specified in Table 1.1.

Table 1.1 Brackets used in the output of lists by SCHUR.

Brackets	Output items
{ }	$U_n,\ SU_n,\ U_{m/n},\ SU_{m/n},\ S_n,\ A_n,\ S-functions$
[ ]	$O_n,\ SO_n$
< >	$Sp_{2k},\ S_n(reduced)$
[>	$OSp_{m/n}$
( )	$G_2,\ F_4.\ E_6,\ E_7,\ E_8$

The precise form of the output depends on the MODE in which SCHUR is operating. We now give an introduction to these modes and will then be able to give sample lines of input and output for the user to study.

1.3 The SCHUR Modes

SCHUR operates within four distinct modes that perform distinct tasks, these

are the DPMode, REPmode, BRMode and SFNmode. The brief features of each mode are as follows:-

DPMode:

This mode handles direct products of groups and all those operations, such as branching rules where the number, and type, of the groups set change. Direct products involving up to six different groups may be set. This is also the natural mode to use when setting up functions and using the very important command **Rule**. The direct product representations used in this mode are referred to as DPreps.

REPmode:

This mode executes operations involving the representations (frequently referred to as "reps") of a single group and does not permit operations that change the group. It yields more information on the properties of reps than does the DPMode and has a simpler input.

BRMode:

This mode only executes branching rules and is useful when a given branching rule is used repeatedly such as when constructing tables. No other operations are permitted in this mode and data cannot be transported from this mode to other modes.

SFNmode:

This is the natural mode to use when carrying out operations on lists of symmetric functions such as the $S-$ or $Q-$functions. This is the mode that should be explored first by novice users.

Each of the four modes is the subject of a separate tutorial and the intending user is strongly advised to systematically follow through the tutorials in their given sequence.

■ 1.4 Sample Input and Output Lists

The input of a list of items without any command to act on the list results in SCHUR producing an output list with each item in a form appropriate to the current group settings and SCHUR mode but with no action on the items in the list and no attempt to order or standardise items in the list. Following the output list by the command **LA**st results in the most recently output list being re-echoed but with the items in the list presented as a sorted and ordered list.

We now illustrate these features of SCHUR by giving some examples of typical input lists, the lists as echoed by SCHUR and finally the sorted and ordered list re-echoed by SCHUR following the issuing of the command **LA**st. The first examples all take place in the SFNmode and hence the output *outputitems* are enclosed in { } brackets.

Sample: 1
Input: 4321+21+621+54+341+1+321+4~321+0+71
Output: $\{4321\}+\{21\}+\{621\}+\{54\}+\{341\}+\{1\}+\{321\}+\{4-321\}$

+{0} + {71}

Last: {71} + {621} + {54} + {4321} + {4 − 321} + {341} + {321} + {21} +{1} + {0}

Notice that in the **Input:** list no attempt has been made to order the items and two non-standard partitions have been included. The **Output:** list has faithfully echoed the **Input:** list except for enclosing the items in the appropriate brackets. The **Last:** list has ordered the items in the list with the partition with largest first part heading the list.

Sample: **2**
Input: !12.321+4^32^21−2.4^32^21+3^!12 21^6+2.!13!11 721
Output: 12{321}+{4^3 2^2 1}−2{4^3 2^2 1}+{3^12 21^6 }+2{13 11 721}
Last: 2{13 11 721} − {4^3 2^2 1} + {3^12 21^6 } + 12{321}

Notice that in this sample the segment 4^32^21−2.4^32^21 occurs in both the **Input:** and **Output:** lists and the obvious cancellation is made in the **Last:** list.

Sample: **3**
Input: !1371.321+21−31^6+2^!12+31+41+61+71−&
8321+421
Output: 1371{321} + {21} − {31^6 } + {2^12 } + {31} + {41} + {61} + {71} − {8321} + {421}
Last: −{8321} + {71} + {61} + {421} + {41} + 1371{321} − {31^6 } + {31} + {2^12 } + {21}

This sample illustrates the use of the ampersand "`&`" to continue a line. Immediately after the `&` a <CR> was issued and then the next line of input written and the final <CR> issued and SCHUR responded with the **Output:** list. Note carefully the placement of the `&` *after* the "−" in the **Input:** list.

We now turn to a sample constructed in the REPmode where the group has been set as SO(8). This sample illustrates the input of *prefix* and *label* and in **Last:** the normal ordering adopted for irreducible representations such as those for SO(8) by SCHUR.

Sample: **4**
Input: 1111++s0−+21+s21++3111−+3111++s2111−+0
Output: [1^4] + + [s; 0] − + [21] + [s; 21] + + [31^3] − + [31^3] + + [s; 21^3] − + [0]
Last: [31^3] + + [31^3] − + [s; 21^3] − + [s; 21] + + [21] + [1^4] + + [s; 0] − + [0]

Notice the very compact input used. For clarity spaces could have been inserted. Nevertheless SCHUR has been able to correctly display the items and in **Last:** produce an orderly looking list.

As a final sample we consider a case from the DPMode where three groups have been set as SU(3) ∗Sp(4) ∗SO(6).

Sample: **5**
Input: 13[21∗2∗31]+[31∗42∗321+]+2[42∗31∗s21−]−6[21∗21∗21]

Output: $2\{42\} < 31 > [s; 21] - \ + \{31\} < 42 > [321] +$
$- \ 6\{21\} < 21 > [21] \ + \ 13\{21\} < 2 > [31]$
Last: $2\{42\} < 31 > [s; 21] - \ + \{31\} < 42 > [321] +$
$- \ 6\{21\} < 21 > [21] \ + \ 13\{21\} < 2 > [31]$

Notice that in the DPMode SCHUR echoes the **Output:** list as an *ordered* list exactly as in the **Last:** list. Notice also the ordering that SCHUR has adopted, the first group in the list of direct product groups takes precedence and so on. Changing the order of the groups will change the order of the output list.

■ 1.5 Commands and Expressions

In §1.2 to §1.4 the input and output of lists of items was considered. SCHUR can create lists in response to the input of commands. Commands can also act on lists to produce new lists. We shall frequently use the ambiguous term EXPRession usually abbreviated to just EXPR to stand for a list or a command or a series of commands that finally produces a list. This usage reflects the interactive nature of SCHUR, it allows the user to act on the output to create new output. In many cases a user may nest commands inside commands with only the final output being displayed by SCHUR with all the intermediate output being suppressed. Thus, for example, COMmand1,COMmand2,EXPR would have the action of COMmand2 acting on EXPR to produce a new EXPR which would then be acted on by COMmand1 to produce a final EXPR which would be ported to the screen. It is just this feature that makes SCHUR so versatile. Given a large set of basic commands the user is free to combine them in a multitude of different ways to achieve the desired results. These features are most strongly exploited in the possibility of user defined functions that the user can create for particular tasks. These are fully described in the appropriate tutorials.

■ 1.6 Accessing Help Files

SCHUR contains a large number of Help files that may be accessed within SCHUR. A complete listing is given in Appendix B. These are pure ASCII files and the user is free to modify them and indeed add his or her own help files.

> *Ah, he thought, the truth bursting on him suddenly, nobody grows up. Everyone carries around all the selves that they have ever been, intact, waiting to be activated in moments of pain, of fear, of danger. Everything is retrievable, every shock, every hurt. But perhaps it becomes a duty to abandon the stock of time that carries within oneself, to discard it in favour of the present, so that one's embrace may be turned outwards to the world in which one has made one's home (page 210)*
>
> — Anita Brookner, *Latecomers*

2

Mathematical Background to SCHUR

Combinatorics and Schur functions

■ Introduction

In this chapter we outline the combinatorial concepts used in SCHUR. While the reader is spared most of the technical details - after all that is the purpose of SCHUR - to allow you to obtain results without the tedium of constructing the algorithms - it is most desirable that the user be familiar with matters of notation. Even those familiar with matters of notation should at least skim this chapter before using SCHUR.

The key object in SCHUR is the partition of an integer. From that object we can define various symmetric functions, most notably the Schur function (or briefly, the $S-$function). From the properties of $S-$functions we are able to define certain infinite series of $S-$functions that play a key role in developing branching rules etc. Various operations associated with $S-$functions such as outer products and skews, via the Littlewood-Richardson rule, inner products and plethysm are presented. This material is essential to understanding the approach to Lie groups used by SCHUR. In SCHUR the entire theory of Lie groups is developed in terms of partition labelled representations. This has the advantage of being closely related to the tensor and spinor notations familiar to many physicists. The relationship to the weight labelling methods that find their expression in terms of the Coxeter-Dynkin diagrams is outlined. The mathematical background for the Lie group applications are covered in the succeeding chapter.

■ 2.1 Partitions

We shall take a partition to be any finite sequence of integers

$$\lambda = (\lambda_1 \lambda_2 \ldots \lambda_p) \tag{2.1}$$

which unless otherwise stated we shall assume to be non-negative integers arranged in non-increasing order:

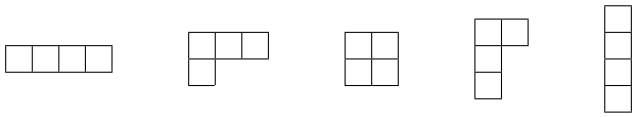
$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p \geq 0 \tag{2.2}$$

Normally we will omit zeros. The non-zero λ_i form the *parts* of λ. The number of parts is the *length*, ℓ_λ, of λ while the sum of its parts, w_λ, is the *weight* of λ. If $w_\lambda = n$ then λ is said to be a *partition* of n. We shall often write $\lambda \vdash n$ to indicate that λ is a partition of n. Repeated parts of a partition will frequently be indicated as i^{m_i} where m_i is the number of times the part i occurs in the partition. Thus we shall write the partitions for $n = 6$ as

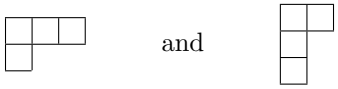
$$(6) \quad (51) \quad (42) \quad (41^2) \quad (3^2) \quad (321) \quad (31^3) \quad (2^3) \quad (2^2 1^2) \quad (21^4) \quad (1^6)$$

■ 2.2 Young Diagrams

Every partition $\lambda \vdash n$ may be associated with a *Young frame* or *shape* F^λ involving n cells, dots, circles or boxes in ℓ_λ left-adjusted rows with the $i-$th row containing λ_i cells {SCHUR draws either boxes (if the extended IBM characters are available) or circles otherwise}. For $n = 5$ we have the five diagrams



The *conjugate* of a partition λ is a partition λ' whose diagram is the transpose of the diagram of λ. If $\lambda' \equiv \lambda$ then the partition λ is said to be *self-conjugate*. Thus



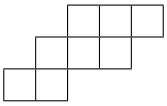
and

are conjugates of one another while

is self-conjugate. Herein primed lower case greek letters are used to indicate the conjugate of the partition designated by the corresponding unprimed greek letter.

■ 2.3 Skew Frames

Given two partitions λ and μ such that $\lambda \supseteq \mu$ implies that the frame F^{λ} contains the frame F^{μ}, i.e. that $\lambda_i \geq \mu_i$ for all $i \geq 1$. The difference $\rho = \lambda - \mu$ forms a *skew* frame $F^{\lambda/\mu}$. Thus, for example, the skew frame $F^{542/21}$ has the form



Note that a skew frame may consist of disconnected pieces.

■ 2.4 Frobenius notation for partitions

There is an alternative notation for partitions due to Frobenius. The *diagonal* of the nodes in a Young diagram beginning at the top left-hand corner is called the *leading diagonal*. The number of nodes in the leading diagonal is called the *rank* of the partition. If r is the rank of a partition then let a_i be the number of nodes to the right of the leading diagonal in the $i-$th row and let b_i be the number of nodes below the leading diagonal in the $i-$th column. The partition is then denoted by Frobenius as

$$\begin{pmatrix} a_1, & a_2, & \ldots, & a_r \\ b_1, & b_2, & \ldots, & b_r \end{pmatrix} \tag{2.3}$$

We note that

$$a_1 > a_2 > \ldots > a_r$$
$$b_1 > b_2 > \ldots > b_r$$

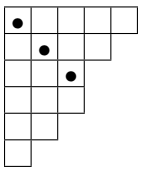
and

$$a_1 + a_2 + \ldots + a_r + b_1 + b_2 + \ldots + b_r + r = n$$

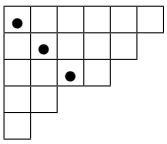
The partition conjugate to that of Eq.(2.3) is just

$$\begin{pmatrix} b_1, & b_2, & \ldots, & b_r \\ a_1, & a_2, & \ldots, & a_r \end{pmatrix} \tag{2.4}$$

As an example consider the mutually conjugate partitions $(5\,4\,3^2\,2\,1)$ and $(6\,5\,4\,2\,1)$. Drawing their diagrams and marking their leading diagonal we have



and

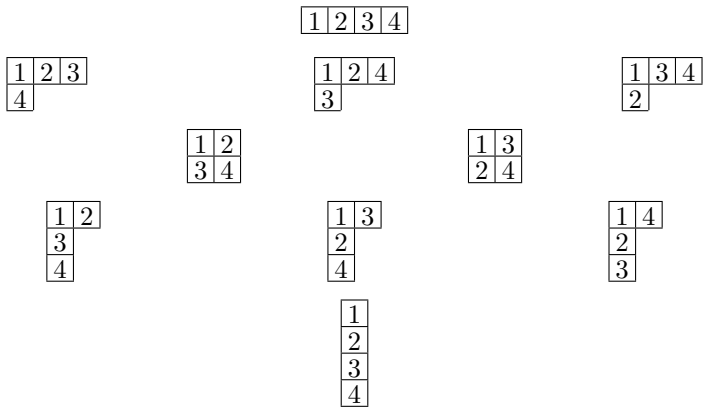


from which we deduce the respective Frobenius designations

$$\begin{pmatrix} 4 & 2 & 0 \\ 5 & 3 & 1 \end{pmatrix} \qquad \text{and} \qquad \begin{pmatrix} 5 & 3 & 1 \\ 4 & 2 & 0 \end{pmatrix}$$

■ 2.5 Young Tableaux

A Young tableau is an assignment of n numbers to the n cells of a frame F^λ with $\lambda \vdash n$ according to some numbering sequence. A tableau is *standard* if the assignment of the numbers $1, 2, \ldots, n$ is such that the numbers are strictly increasing from left to right in rows and down columns from top to bottom. Thus for the partitions of the integer 4 we have the standard Young tableaux

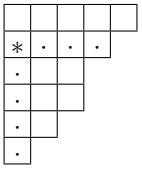


We notice in the above examples that the number of standard tableaux for conjugate partitions is the same. Indeed the number of standard tableaux associated with a given frame F^λ is the *dimension* f_n^λ of an irreducible representation $\{\lambda\}$ of the symmetric group $\mathcal{S}_n$.

■ 2.6 Hook lengths and dimensions for $\mathcal{S}_n$

The *hook length* of a given box in a frame F^λ is the length of the right-angled path in the frame with that box as the upper left vertex. For example, the hook

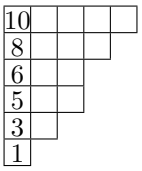
length of the marked box in



is 8. SCHUR makes extensive use of the concept of hook lengths in computing dimensions of representations of various groups and in its implementation of modification rules for modifying non-standard labelled representations.

Theorem 2.1: *To find the dimension of the representation of $\mathcal{S}_n$ corresponding to the frame F^λ, divide $n!$ by the factorial of the hook length of each box in the first column of F^λ and multiply by the difference of each pair of such hook lengths.*

Thus for the partition $(5\,4\,3^2\,2\,1)$ we have the hook lengths



and hence a dimension

$$f_{18}^{543^221} = 18!\frac{2\times 4\times 5\times 7\times 9\times 2\times 3\times 5\times 7\times 1\times 3\times 5\times 2\times 4\times 2}{10!\times 8!\times 6!\times 5!\times 3!\times 1!}$$
$$= 10720710$$

It is not suggested that you check the above result by explicit enumeration! The above evaluation can also be equivalently made by computing the hook lengths h_{ij} for every box at position (i,j) and then noting that

$$f_n^\lambda = \frac{n!}{\prod_{(i,j)\in\lambda} h_{ij}} \tag{2.5}$$

which is the celebrated result of Frame, Robinson and Thrall.

■ 2.7 Unitary Numbering of Young Tableaux

Many different prescriptions can be given for injecting numbers into the boxes of a frame. We have already noted the standard numbering which is intimately associated with the symmetric group $\mathcal{S}_n$. Another important numbering prescription is that of *unitary* numbering where now numbers 1, 2, ..., d are injected into the boxes of a frame F^λ such a way that:

i. Numbers are non-decreasing across a row going from left to right.

ii. Numbers are strictly increasing in columns from top to bottom.

The first condition permits repetitions of integers. Thus using the numbers 1, 2, 3 in the frame $F^{2\,1}$ we obtain the 8 tableaux

$$\begin{array}{|c|c|}\hline 1&1\\\hline 2\\\cline{1-1}\end{array}\quad\begin{array}{|c|c|}\hline 1&1\\\hline 3\\\cline{1-1}\end{array}\quad\begin{array}{|c|c|}\hline 1&2\\\hline 2\\\cline{1-1}\end{array}\quad\begin{array}{|c|c|}\hline 1&2\\\hline 3\\\cline{1-1}\end{array}\quad\begin{array}{|c|c|}\hline 1&3\\\hline 2\\\cline{1-1}\end{array}\quad\begin{array}{|c|c|}\hline 1&3\\\hline 3\\\cline{1-1}\end{array}\quad\begin{array}{|c|c|}\hline 2&2\\\hline 3\\\cline{1-1}\end{array}\quad\begin{array}{|c|c|}\hline 2&3\\\hline 3\\\cline{1-1}\end{array}$$

Had we chosen $d = 2$ we would have obtained just two tableaux while $d = 4$ yields twenty tableaux. In general, for a frame F^λ a unitary numbering using the integers 1, 2, ..., d leads to

$$f_d^{\{\lambda\}} = \frac{G_d^\lambda}{H_\lambda} \tag{2.6}$$

where H_λ is the product of the hook lengths h_{ij} of the frame and

$$G_d^\lambda = \prod_{(i,j)\in\lambda} (d+i-j) \tag{2.7}$$

Thus for $d = 5$ and $\lambda = (421)$ we obtain for $G_5^{\{421\}}$ the numbering

$$\begin{array}{|c|c|c|c|}\hline 5&6&7&8\\\hline 4&5\\\cline{1-2}3\\\cline{1-1}\end{array}$$

leading to $G_5^{\{421\}} = 100800$ while for $H_{\{421\}}$ the numbering

$$\begin{array}{|c|c|c|c|}\hline 6&4&2&1\\\hline 3&1\\\cline{1-2}1\\\cline{1-1}\end{array}$$

leads to $H_{\{421\}} = 144$ from which we deduce that

$$f_5^{\{4\,2\,1\}} = 700$$

which is the dimension of the irreducible representation $\{4\,2\,1\}$ of the general linear group GL_5. In general, $f_d^{\{\lambda\}}$ is the dimension of the irreducible representation $\{\lambda\}$ of GL_d. Since the representations of GL_d labelled by partitions λ remain irreducible under restriction to the unitary group U_d Eq.(2.6) is valid for computing the dimensions of the irreducible representations of the unitary group U_d.

The same rules for a unitary numbering may be applied to the skew frames $F^{\lambda/\mu}$ introduced in §2.3. Thus for $F^{542/21}$ an allowed unitary numbering using just the integers 1 and 2 would be

$$\begin{array}{|c|c|c|}\hline 1&1&1\\\hline 1&2&2\\\cline{1-3}1&2\\\cline{1-2}\end{array}$$

Note that our unitary numbering yields what in the mathematical literature are commonly referred to as *semistandard* Young tableaux. Other numberings are possible and have been developed for all the classical Lie algebras.

■ 2.8 Young Tableaux and Monomials

A numbered frame may be associated with a unique monomial by replacing each integer i by a variable x_i. Thus the Young tableau T

1	1	2	4	5
3	3	3	5	
4	6	7		
5	7	8		
6	8			
7				

can be associated with the monomial $\mathbf{x}^T = x_1^2\, x_2\, x_3^3\, x_4^2\, x_5^3\, x_6^2\, x_7^3\, x_8^2$

■ 2.9 Monomial symmetric functions

Consider a set of variables $(\mathbf{x}) = x_1, x_2, \ldots, x_d$. A *symmetric* monomial

$$m_\lambda(\mathbf{x}) = \sum_\alpha \mathbf{x}^\alpha \tag{2.8}$$

where $\mathbf{x}^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_d^{\alpha_d}$ and the sum is over all distinct permutations $\alpha = \alpha_1, \alpha_2, \ldots, \alpha_d)$ of $(\lambda) = (\lambda_1, \lambda_2, \ldots)$. Thus if $(\mathbf{x}) = (x_1, x_2, x_3)$ then

$$m_{21}(\mathbf{x}) = x_1^2\, x_2 + x_1^2\, x_3 + x_1\, x_2^2 + x_1\, x_3^2 + x_2^2\, x_3 + x_2\, x_3^2$$
$$m_{1^3}(\mathbf{x}) = x_1\, x_2\, x_3$$

The unitary numbering of $(\lambda) = (2\,1)$ with 1, 2, 3 corresponds to the sum of monomials

$$m_{21}(\mathbf{x}) + 2m_{1^3}(\mathbf{x})$$

The same linear combination occurs for any number of variables with $d \geq 3$. However, for two variables just $m_{21}(\mathbf{x})$ survives while in terms of a single variable neither monomial survives.

The monomials $m_\lambda(\mathbf{x})$ are *symmetric functions*. If $\lambda \vdash n$ then $m_\lambda(\mathbf{x})$ is homogeneous of degree n. Unless otherwise stated we shall henceforth assume that $\mathbf{x}$ involves an infinite number of variables x_i.

The *ring of symmetric functions* $\Lambda = \Lambda(\mathbf{x})$ is the vector space spanned by all the $m_\lambda(\mathbf{x})$. This space can be decomposed as

$$\Lambda = \oplus_{n \geq 0} \Lambda^n \tag{2.9}$$

where Λ^n is the space spanned by all m_λ of degree n. Thus the $\{m_\lambda : \lambda \vdash n\}$ form a basis for the space Λ^n which is of dimension $p(n)$ where $p(n)$ is the number of partitions of n.

■ 2.10 The Classical Symmetric Functions

There exist various symmetric functions that can also form bases for Λ^n and are hence expressible in terms of monomials of degree n. Among these are the bases that derive from the classical symmetric functions p_n, e_n, h_n, and f_n known

as the *power sum*, *elementary*, *homogeneous* and *forgotten* symmetric functions respectively. They may each be defined in terms of monomials with:

$$p_n = m_n = \sum_{i \geq 1} x_i^n$$

$$e_n = m_{1^n} = \sum_{i_1 < \ldots < i_n} x_{i_1} \ldots x_{i_n}$$

$$h_n = \sum_{\lambda \vdash n} m_\lambda = \sum_{i_1 \leq \ldots \leq i_n} x_{i_1} \ldots x_{i_n}$$

and the *forgotten* symmetric functions being defined under an involution ω such that

$$f_\lambda = \omega(m_\lambda)$$

for all λ.

For example, with $n = 3$

$$\begin{aligned}
p_3 &= m_3 = x_1^3 + x_2^3 + x_3^3 + \ldots \\
e_3 &= m_{1^3} = x_1x_2x_3 + x_1x_2x_4 + x_1x_3x_4 + x_2x_3x_4 + \ldots \\
f_3 &= m_3 = x_1^3 + x_2^3 + x_3^3 + \ldots \\
h_3 &= m_3 + m_{21} + m_{1^3} \\
&= x_1^3 + x_2^3 + \ldots + x_1^2x_2 + x_1x_2^2 + \ldots + x_1x_2x_3 + x_1x_2x_4 + \ldots
\end{aligned}$$

To form a basis for Λ^n there must be one element for every $\lambda \vdash n$. To that end *multiplicative* symmetric functions

$$z_\lambda = z_{\lambda_1} z_{\lambda_2} \ldots z_{\lambda_k}$$

with $z = p, e, h$ or f are introduced. The symmetric functions

$$m_\lambda, p_\lambda, e_\lambda, h_\lambda, f_\lambda \qquad \lambda \vdash n$$

form five distinct bases for the ring Λ^n of symmetric functions of degree n and are indexed by partitions.

2.11 The Schur Functions

An alternative very important basis of Λ is realised in terms of the Schur functions, or for brevity, *S−functions*, s_λ, which may be variously defined. Combinatorially they may be defined as

$$s_\lambda(\mathbf{x}) = \sum_T \mathbf{x}^T \tag{2.10}$$

where the summation is over all semistandard $\lambda-$tableaux T. For example, consider the $S-$functions s_λ in just three variables (x_1, x_2, x_3). For $\lambda = (2\,1)$ we have the eight tableaux T found earlier

$$\begin{array}{|c|c|}\hline 1&1\\\hline 2\\\cline{1-1}\end{array} \quad \begin{array}{|c|c|}\hline 1&1\\\hline 3\\\cline{1-1}\end{array} \quad \begin{array}{|c|c|}\hline 1&2\\\hline 2\\\cline{1-1}\end{array} \quad \begin{array}{|c|c|}\hline 1&2\\\hline 3\\\cline{1-1}\end{array} \quad \begin{array}{|c|c|}\hline 1&3\\\hline 2\\\cline{1-1}\end{array} \quad \begin{array}{|c|c|}\hline 1&3\\\hline 3\\\cline{1-1}\end{array} \quad \begin{array}{|c|c|}\hline 2&2\\\hline 3\\\cline{1-1}\end{array} \quad \begin{array}{|c|c|}\hline 2&3\\\hline 3\\\cline{1-1}\end{array}$$

Each tableaux T corresponds to a monomial $\mathbf{x}^T$ to give

$$s_{2\,1}(x_1,\,x_2,\,x_3) = x_1^2\,x_2 + x_1^2\,x_3 + x_1\,x_2^2 + x_1\,x_2\,x_3 + x_1\,x_2\,x_3 + x_1\,x_3^2 + x_2^2\,x_3 + x_2\,x_3^2 \tag{2.11}$$

We note that the monomials in Eq.(2.11) can be expressed in terms of just two *symmetric monomials* in the three variables $(x_1,\,x_2,\,x_3)$ to give

$$s_{2\,1}(x_1,\,x_2,\,x_3) = m_{2\,1}(x_1,\,x_2,\,x_3) + 2m_{1^3}(x_1,\,x_2,\,x_3) \tag{2.12}$$

In an arbitrary number of variables

$$s_{2\,1}(\mathbf{x}) = m_{2\,1}(\mathbf{x}) + 2m_{1^3}(\mathbf{x}) \tag{2.13}$$

This is an example of the general result that an $S-$function may be expressed as a linear combination of symmetric monomials as indeed would be expected if the $S-$functions are a basis of Λ^n. In fact

$$s_\lambda(\mathbf{x}) = \sum_{\mu \vdash n} K_{\lambda\mu} m_\mu(\mathbf{x}) \tag{2.14}$$

where $w_\lambda = n$ and $K_{\lambda\lambda} = 1$. The $K_{\lambda\mu}$ are the elements of an upper triangular matrix K known as the Kostka matrix. K is an example of a *transition matrix* that relates one symmetric function basis to another. The symmetric functions m_λ, e_λ, f_λ, h_λ and s_λ have the important property of forming an *integral* basis of Λ, whereas the p_λ form a *non-integral* basis of Λ. This means that transformations between the symmetric functions m_λ, e_λ, f_λ, h_λ and s_λ all involve integer coefficients and SCHUR will perform such transformations. The transition matrix relating p_λ to s_λ is just the character table of S_n for $\lambda \vdash n$ and involves just integer coefficients and is available in SCHUR.

$$p_\rho = \sum_\lambda \chi_\rho^\lambda s_\lambda$$

The inverse transformation involves non-integral coefficients and are implemented in SCHUR by presenting the coefficients multiplied by $|\rho|!$.

■ 2.12 Calculation of the Elements of the Kostka Matrix

The elements $K_{\lambda\mu}$ of the Kostka matrix may be calculated readily by the following algorithm :

i. Draw the frame F^λ.

ii. Form all possible semistandard tableaux that arise in numbering F^λ with μ_1 ones, μ_2 twos etc.

iii. $K_{\lambda\mu}$ is the number of semistandard tableaux so formed.

Thus calculating $K_{(42)\,(2^2\,1^2)}$ we obtain the four semistandard tableaux

1	1	2	2
3	4		

1	1	2	3
2	4		

1	1	2	4
2	3		

1	1	3	4
2	2		

and hence $K_{(42)\,(2^2\,1^2)} = 4$.

For practical reasons SCHUR uses a much faster algorithm to calculate $K_{\lambda\mu}$ and if desired can present on screen the entire Kostka marix for a given value of n.

■ 2.13 Classical Definition of the S-function

The classical definition of the $S-$function, as opposed to the equivalent combinatorial definition given in Eq.(2.10), was first given by Jacobi as,

$$s_\lambda(\mathbf{x}) = s_\lambda(x_1, x_2, \ldots, x_d) = \frac{a_{\lambda+\delta}(\mathbf{x})}{a_\delta(\mathbf{x})} \tag{2.15}$$

where λ is a partition of length $\leq n$ and $\delta = (d-1, d-2, \ldots, 1, 0)$ with

$$a_{\lambda+\delta}(\mathbf{x}) = det(x_i^{\lambda_j+d-j})_{1\leq i,j\leq d} \tag{2.16}$$

and

$$a_\delta(\mathbf{x}) = \prod_{1\leq i,j\leq n}(x_i - x_j) = det(x_i^{n-j}) \tag{2.17}$$

is the *Vandermonde determinant*.

■ 2.14 Non-standard $S-$functions

The $S-$functions are symmetric functions indexed by ordered partitions λ. We shall frequently write $S-$functions $s_\lambda(\mathbf{x})$ as $\{\lambda\}(\mathbf{x})$ or, since we will generally consider the number of variables to be unrestricted, just $\{\lambda\}$. This bracket notation is used extensively in SCHUR. As a matter of notation the partitions will normally be written without spaces or commas separating the parts where $\lambda_i \leq 9$. A space will be left after any part $\lambda_i \geq 10$. Thus we write $\{12, 11, 9, 8, 3, 2, 1\} \equiv \{12\ \ 11\ \ 98321\}$. While we have defined $S-$functions in terms of ordered partitions (λ) we sometimes encounter $S-$functions labelled by partitions that are not in the standard form and we must convert such *non-standard* $S-$functions into standard $S-$functions. Inspection of the determinantal form (2.15-17) of the $S-$function leads to the following *modification rules* :

$$\{\lambda_1, \lambda_2, \ldots, -\lambda_\ell\} = 0 \tag{2.18}$$

$$\{\lambda_1, \ldots, \lambda_i, \lambda_{i+1}, \ldots, \lambda_\ell\} = -\{\lambda_1, \ldots, \lambda_{i+1} - 1, \lambda_i + 1, \ldots, \lambda_\ell\} \tag{2.19}$$

$$\{\lambda\} = 0 \qquad \text{if } \lambda_{i+1} = \lambda_i + 1 \tag{2.20}$$

Repeated application of the above three rules reduces any non-standard $S-$function to either zero or to a signed standard $S-$function. In the process of using the above rules trailing zero parts are omitted. In most cases SCHUR automatically carries out the standardisation process which is normally hidden from the user.

■ 2.15 Skew $S-$functions

The combinatorial definition given for $S-$functions in Eq.(2.10) is also valid for skew tableaux and can hence be used to define *skew* $S-$functions $s_{\lambda/\mu}(\mathbf{x})$ or $\{\lambda/\mu\}$. Since the $s_{\lambda/\mu}(\mathbf{x})$ are symmetric functions they must be expressible in

terms of $S-$functions $s_\nu(\mathbf{x})$ such that

$$s_{\lambda/\mu}(\mathbf{x}) = \sum_\nu c^\lambda_{\mu\nu} s_\nu(\mathbf{x}) \tag{2.21}$$

It may be shown that the coefficients $c^\lambda_{\mu\nu}$ are necessarily non-negative integers and symmetric with respect to μ and ν. The coefficients $c^\lambda_{\mu\nu}$ are commonly referred to as *Littlewood-Richardson* coefficients.

■ 2.16 The Littlewood-Richardson rule

The product of two $S-$functions can be written as a sum of $S-$functions, viz.

$$s_\mu(\mathbf{x}).s_{\nu(\mathbf{x})} = \sum_\lambda c^\lambda_{\mu\nu} s_\lambda(\mathbf{x}) \tag{2.22}$$

The Littlewood-Richardson coefficients $c^\lambda_{\mu\nu}$ in Eqs. (2.21) and (2.22) are identical, though the summations are of course different. In both cases $w_\mu + w_\nu = w_\lambda$. A rule for evaluating the coefficients $c^\lambda_{\mu\nu}$ was given by Littlewood and Richardson in 1934 and has played a major role in all subsequent developments and is central to most of the operations in SCHUR. The rule may be stated in various ways. We shall state it first in terms of semistandard tableaux and then also give the rule for evaluating the product given in Eq.(2.22) which is commonly referred to as the *outer multiplication* of $S-$functions. In each statement the concepts of a *row-word* and of a *lattice permutation* is used.

Definition 2.1 A word

> *Let T be a tableau. From T we derive a row-word or sequence $w(T)$ by reading the symbols in T from right to left (i.e. as in Arabic or Hebrew) in successive rows starting at the top row and proceeding to the bottom row*

Thus for the tableau

1	1	2	2	3
2	2	3	3	
4	4			
5	6			
7				
8				

we have the word $w(T) = 322113322446578$ and for the skew tableau

		1	1	1
	1	2	2	
1	2			

we have the word $w(T) = 11122121$.

Definition 4.2 A lattice permutation

A word $w = a_1 a_2 \dots a_N$ in the symbols $1, 2, \dots, n$ is said to be a lattice permutation if for $1 \leq r \leq N$ and $1 \leq i \leq n-1$, the number of occurrences of the symbol i in $a_1 a_2 \dots a_r$ is not less than the number of occurrences of $i+1$.

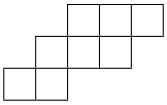
Thus the word $w(T) = 322113322446578$ is clearly not a lattice permutation whereas the word $w(T) = 11122121$ is a lattice permutation. The word $w(T) = 12122111$ is not a lattice permutation since the sub-word 12122 has more twos than ones.

Theorem 2.2 *The value of the coefficient $c^{\lambda}_{\mu\nu}$ is equal to the number of semi-standard tableaux T of shape $F^{\lambda/\mu}$ and content ν such that $w(T)$ is a lattice permutation.*

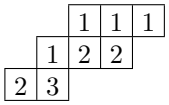
By *content* ν we mean that each tableau T contains ν_1 ones, ν_2 twos, etc.

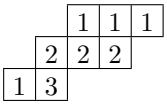
Example

Let us evaluate the coefficient $c^{542}_{431\ 21}$. We first draw the frame $F^{542/21}$.

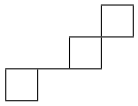


Into this frame we must inject the content of 431 i.e. 4 ones, 3 twos and 1 three in such a way that we have a lattice permutation. We find two such numberings

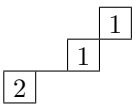


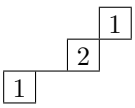


and hence $c^{542}_{431\ 21} = 2$. Note that in the evaluation we had a choice, we could have, and indeed more simply, evaluated $c^{542}_{21\ 431}$. In that case we would have drawn the frame $F^{542/431}$ to get



Note that in this case the three boxes are disjoint. This skew frame is to be numbered with two ones and one 2 leading to the two tableaux





verifying the previous result. Theorem 2.2 gives a direct method for evaluating the Littlewood-Richardson coefficients. These coefficients can be used to evaluate both skews and products. It is sometimes useful to state a procedure for directly evaluating products.

Theorem 2.3 *To evaluate the $S-$function product $\{\mu\}.\{\nu\}$*

1. *Draw the frame F^μ and place μ_1 ones in the first row, μ_2 twos in the second row etc until the frame is filled with integers.*
2. *Draw the frame F^ν and inject positive integers to form a semistandard tableau such that the word formed by reading from right to left starting at the top row of the first frame F^μ and moving downwards along successive rows to the bottom row and then continuing through the second frame F^ν is a lattice permutation.*
3. *Repeat the above process until no further words can be constructed.*
4. *Each word corresponds to an $S-$function $\{\lambda\}$ where λ_1 is the number of ones, λ_2 the number of twos etc.*

As an example consider the $S-$function product $\{21\}.\{21\}$. Step 1 gives the tableau

$$\begin{array}{|c|c|}\hline 1&1\\\hline 2\\\cline{1-1}\end{array}$$

Steps 2 and 3 lead to the eight numbered frames

$$\begin{array}{|c|c|}\hline 1&1\\\hline 2\\\cline{1-1}\end{array}\quad\begin{array}{|c|c|}\hline 1&1\\\hline 3\\\cline{1-1}\end{array}\quad\begin{array}{|c|c|}\hline 1&2\\\hline 2\\\cline{1-1}\end{array}\quad\begin{array}{|c|c|}\hline 1&2\\\hline 3\\\cline{1-1}\end{array}\quad\begin{array}{|c|c|}\hline 1&3\\\hline 2\\\cline{1-1}\end{array}\quad\begin{array}{|c|c|}\hline 1&3\\\hline 4\\\cline{1-1}\end{array}\quad\begin{array}{|c|c|}\hline 2&3\\\hline 3\\\cline{1-1}\end{array}\quad\begin{array}{|c|c|}\hline 2&3\\\hline 4\\\cline{1-1}\end{array}$$

Step 4 then lead to the eight words

$$112112 \quad 112113 \quad 112212 \quad 112213 \quad 112312 \quad 112314 \quad 112323 \quad 112324$$

from which we conclude that

$$\{21\}.\{21\} = \{42\} + \{41^2\} + \{3^2\} + 2\{321\} + \{31^3\} + \{2^3\} + \{2^2 1^2\}$$

While it is instructive for the reader to carry out some simple examples of the evaluation of Littlewood-Richardson coefficients it will quickly become apparent that such hand calculations are, for all but the simplest cases, tedious with a high probability of error, it is here that SCHUR comes into its own. The corresponding problem for expressing the product of a list of monomials with a list of $S-$functions to produce a list of $S-$functions is accomplished in SCHUR using the so-called Gordan's formula.

■ 2.17 Relationship to the Unitary Group

Various symmetric functions indexed by partitions may be defined on sets of variables. The variables can admit many interpretations. We may, for example, choose as the set of variables a set of matrices. The link between $S-$functions and the character theory of groups is such that, if λ is a partition with $\ell_\lambda \leq N$ and the eigenvalues of a group element, g, of the unitary group U_N are given by $x_j = \exp(i\phi_j)$ for $j = 1, 2, \ldots, N$ then the $S-$function

$$\{\lambda\} = \{\lambda_1\lambda_2 \ldots \lambda_N\} = s_\lambda(\mathbf{x}) = s_\lambda(\exp(i\phi_1)\exp(i\phi_2)\ldots\exp(i\phi_N))$$

is nothing other than the character of g in the irreducible representation of U_N conventionally designated by $\{\lambda\}$.

The Littlewood-Richardson rule gives the resolution of the Kronecker product $\{\mu\} \times \{\nu\}$ of U_N as

$$\{\mu\} \times \{\nu\} = \sum_{w_\lambda = w_\mu + w_\nu} c^\lambda_{\mu,\nu} \{\lambda\} \tag{2.23}$$

where the $c^\lambda_{\mu,\nu}$ are the usual Littlewood-Richardson coefficients. Equation (2.23) must be modified for partitions λ involving more than N parts. Here the *modification rule* is very simple. We simply discard all partitions involving more than N parts.

■ 2.18 Inner Products of S-functions

The ordinary, or *outer*, product $\{\lambda\} \cdot \{\mu\}$ where $\lambda \vdash n$ and $\mu \vdash m$ corresponds to inducing $\{\lambda\} \cdot \{\mu\}$ from $S_n \times S_m$ to S_{m+n}. The Kronecker, or *inner* product $\{\lambda\} \circ \{\mu\}$, now with $\lambda, \mu \vdash n$ corresponds to the tensor product of the representations $\{\lambda\}$ and $\{\mu\}$ in S_n. This serves to define the inner product of $S-$functions. Thus in terms of the characters of S_n if

$$\chi^\mu \chi^\nu = \sum_{\lambda \vdash n} g^{\mu\nu}_\lambda \chi^\lambda \tag{2.24}$$

then

$$\{\mu\} \circ \{\nu\} = \sum_{\lambda \vdash n} g^{\mu\nu}_\lambda \{\lambda\} \tag{2.25}$$

■ 2.19 Reduced Inner Products

The tensor irreducible representations of the symmetric group S_n are customarily labelled as $\{\lambda\}$ where $\lambda \vdash n$. Such a notation is clearly $n-$dependent. It is often useful to introduce a *reduced notation* that is $n-$independent. This is accomplished for

$$\{\lambda\} = \{n - m, \mu\} \qquad \mu \vdash m \tag{2.26}$$

by writing $\{\lambda\}$ in reduced notation $< \mu >$. Thus in reduced notation $< 21 >$ corresponds to $\{321\}$ in S_6 or equally $\{721\}$ in S_{10} etc. SCHUR distinguishes the reduced notation by always enclosing the partition in the angular brackets $<, >$.

The resolution of a reduced inner product yields reduced representations and the product rule is $n-$independent. Thus

$$< \mu > \circ < \nu > = \sum_\lambda r^{\mu\nu}_\lambda < \lambda > \tag{2.27}$$

for some non-negative integers $r^{\mu\nu}_\lambda$ For example, SCHUR readily produces the $n-$independent result

$$\begin{aligned} &< 21 > \circ < 1 > \\ &= < 31 > + < 3 > + < 2^2 > + < 21^2 > + 2 < 21 > + < 2 > + < 1^3 > \\ &+ < 1^2 > \end{aligned} \tag{2.28}$$

For a specific value of n the result is found by prefixing the part (n-m) to each partition and changing the brackets to $\{,\}$. For small values of n non-standard partitions arise which must be modified using the modification rules given in §2.14. Thus for $n = 5$ we obtain from Eq. (2.28) the inner product result

$$\{2^2 1\} \circ \{41\} = \{32\} + \{31^2\} + \{2^2 1\} + \{21^3\}$$

whereas for $n = 7$ we obtain

$$\begin{aligned}&\{421\} \circ \{61\}\\ &= \{52\} + \{51^2\} + \{43\} + 2\{421\} + \{41^3\} + \{3^2 1\} + \{32^2\} + \{321^2\}\end{aligned}$$

For $n \geq 7$ the same number of terms is found, the difference is only in the first part of each partition.

■ 2.20 Plethysm of S-functions

The plethysm of $S-$functions plays an important role in many applications. The concept was originally introduced by D. E. Littlewood as arising in the formation of an invariant of an invariant matrix written as $|A^{\{\lambda\}}|^{\{\mu\}}$ which he took to define the operation of $S-$function plethysm as

$$\{\lambda\} \otimes \{\mu\} = \sum_{\nu} p^{\nu}_{\lambda\mu} \{\nu\} \tag{2.29}$$

where the partitions (ν) are all of weight $w_\lambda . w_\mu$ and the coefficients $p^{\nu}_{\lambda\mu}$ are non-negative integers . The $n-$th outer multiplication of an $S-$function, say $\{\lambda\}^n$, may be resolved into a sum of symmetrised powers of $S-$functions by writing

$$\{\lambda\}^n = \sum_{\mu \vdash n} f^{\mu}_{n} \{\lambda\} \otimes \{\mu\} \tag{2.30}$$

where f^{μ}_{n} is the dimension of the irreducible representation $\{\mu\}$ of S_n. Thus, for example,

$$\{21\}^3 = \{21\} \otimes \{3\} + 2(\{21\} \otimes \{21\}) + \{21\} \otimes \{1^3\}$$

The operation of outer $S-$function plethysm has a close connection with branching rules. Thus if $\{\lambda\}$ is an $S-$function corresponding to the character of an irreducible representation of a unitary group U_n of dimension m then that irreducible representation may be embedded in the vector representation, of character $\{1\}$, of the unitary group U_m. Under $U_m \rightarrow U_n$ we have $\{1\} \rightarrow \{\lambda\}$ while for an arbitrary irreducible representation of character $\{\mu\}$ the branching rule becomes

$$U_m \rightarrow U_n \qquad \{\mu\} \rightarrow \{\lambda\} \otimes \{\mu\} \tag{2.31}$$

with the evaluation of the plethysm following from Eq. (2.29) with the resulting $S-$functions being interpreted as characters of irreducible representations of the subgroup U_n with any $S-$functions associated with partitions having more than n non-zero parts being discarded. Given this close connection with branching rules physicists have stayed with Littlewood's original definition writing plethysms as in Eq.(2.29) and this notation is used in SCHUR.

Mathematicians usually give a combinatorial interpretation of the operation of plethysm and instead of writing $s_\lambda \otimes s_\mu$ would write $s_\mu[s_\lambda]$. In this way the operation of $S-$function plethysm is viewed as a substitution process. We illustrate the general idea with an example. Suppose we wish to evaluate $s_2[s_{1^2}](x_1,\ldots,x_4)$. We express $s_{1^2}(x_1,\ldots,x_4)$ as a sum of monomials

$$s_{1^2}(x_1,\ldots,x_4) = x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4 \tag{2.32}$$

Now regard s_2 as a function in as many monomials as in Eq. (2.32) writing

$$s_2[s_{1^2}](x_1,\ldots,x_4) = s_2(x_1x_2, x_1x_3, x_1x_4, x_2x_3, x_2x_4, x_3x_4) \tag{2.33}$$

Very tediously the right-hand-side of Eq. (2.33) may be expanded as a sum of monomials which may then be expressed in terms of $S-$functions to yield finally

$$s_2[s_{1^2}](\mathbf{x}) = s_{2^2}(\mathbf{x}) + s_{1^4}(\mathbf{x})$$

which in Littlewood's notation would be

$$\{1^2\} \otimes \{2\} = \{2^2\} + \{1^4\}$$

Noting Eq.(2.30) we could further deduce that

$$\{1^2\} \otimes \{1^2\} = \{21^2\}$$

The above example illustrates some of the complexities that can arise in evaluating $S-$function plethysms. To date no really efficient algorithms exist for evaluating plethysms though SCHUR will patiently plod through such calculations, the limit being dictated by time and memory considerations.

The algebra of plethysms is governed by the following rules

$$A \otimes (B \pm C) = A \otimes B \pm A \otimes C \tag{2.34a}$$
$$A \otimes (BC) = (A \otimes B) \cdot (A \otimes C) \tag{2.34b}$$
$$A \otimes (B \otimes C) = (A \otimes B) \otimes C \tag{2.34c}$$
$$(A + B) \otimes \{\mu\} = \sum_\zeta (A \otimes \{\mu/\zeta\}) \cdot (B \otimes \{\zeta\}) \tag{2.34d}$$
$$(A - B) \otimes \{\mu\} = \sum_\zeta (-1)^{\omega_\zeta} (A \otimes \{\mu/\zeta\}) \cdot (B \otimes \{\zeta'\}) \tag{2.34e}$$
$$(AB) \otimes \{\mu\} = \sum_\rho (A \otimes \{\rho\}) \cdot (B \otimes \{\mu \circ \rho\}) \tag{2.34f}$$

SCHUR makes use of the above relations in many of its algorithmic implementations.

■ 2.21 Inner Plethysm

The plethysm just discussed is often referred to as the outer plethysm of $S-$functions. It is also possible to define an *inner* plethysm of $S-$functions based upon

the $n-$th inner product power of $S-$functions so that for S_n

$$\{\lambda\}\circ^n = \sum_{\mu\vdash n} f_n^{\mu}\{\lambda\}\odot\{\mu\} \tag{2.35}$$

where $\{\lambda\}\odot\{\mu\}$ is an inner plethysm. We may write

$$\{\lambda\}\odot\{\mu\} = \sum_{\rho\vdash\omega_\lambda} q_{\lambda\mu}^{\rho}\{\rho\} \tag{2.36}$$

In general, such plethysms are fearsome objects to evaluate and no general algorithm has, as yet, been incorporated in SCHUR. Such objects are especially important in developing branching rules for the embedding of the symmetric group S_n in the full orthogonal group O_n. In this case the reduced notation given in §2.19 is particularly useful. Plethysms arise of the type
$\{n-1,1\}\otimes\{\mu\} =< 1 >\otimes\{\mu\}$. They may be evaluated by use of the identity

$$< 1 >\otimes\{1^m\} =< 1^m > \tag{2.37}$$

and noting that any $S-$function $\{\mu\}$ may be expanded as sums of products of $S-$functions of type $\{1^k\}$ using the determinantal expansion

$$\{\mu\} = \left|1^{\mu'_s-s+t}\right| \tag{2.38}$$

where μ' is the partition conjugate to μ.

SCHUR evaluates plethysms of the type $< 1 >\otimes\{\mu\}$ automatically. For example, SCHUR readily finds that

$$\begin{aligned}
< 1 >\otimes\{3\} &=< 3 > + < 2 > + < 1^2 > +2 < 1 > + < 0 > \\
< 1 >\otimes\{21\} &=< 21 > + < 2 > + < 1^2 > + < 1 > \\
< 1 >\otimes\{1^3\} &=< 1^3 >
\end{aligned}$$

which may be compared with the reduced triple product

$$\begin{aligned}
&< 1 > \cdot < 1 > \cdot < 1 > \\
&=< 3 > +2 < 21 > +3 < 2 > + < 1^3 > +3 < 1^2 > +4 < 1 > + < 0 >
\end{aligned}$$

Making the partitions up to weight 6 we then deduce the S_6 inner plethysms

$$\begin{aligned}
\{51\}\odot\{3\} &= \{3^2\} + \{42\} + \{41^2\} + 2\{51\} + \{6\} \\
\{51\}\odot\{21\} &= \{321\} + \{42\} + \{41^2\} + \{51\} \\
\{51\}\odot\{1^3\} &= \{31^3\}
\end{aligned}$$

which may be compared with the triple inner product in S_6:

$$\begin{aligned}
&\{51\}\circ\{51\}\circ\{51\} \\
&= \{6\} + 4\{51\} + 3\{42\} + 3\{41^2\} + \{3^2\} + 2\{321\} + \{31^3\}
\end{aligned}$$

■ 2.22 S-function Series

Infinite series of $S-$functions play an important role in determining branching rules and furthermore lead to concise symbolic methods well adapted to computer implementation. Truncated series are used extensively in SCHUR. As an example consider the infinite series generated by

$$\begin{aligned} L &= \prod_{i=1}^{\infty}(1-x_i) \\ &= 1 - \sum x_1 + \sum x_1 x_2 - \ldots \end{aligned} \tag{2.39}$$

where the summation is over all distinct terms. e.g.

$$\sum x_1 x_2 = x_1 x_2 + x_1 x_3 + \ldots + x_2 x_3 + x_2 x_4 + \ldots$$

Each monomial in Eq. (2.39) corresponds to an $S-$function $\{1^m\}$ such that

$$\begin{aligned} L &= 1 - \{1\} + \{1^2\} - \{1^3\} + \ldots \\ &= \sum_{m=0}^{\infty}(-1)^m\{1^m\} \end{aligned} \tag{2.40}$$

Taking the inverse of Eq. (2.39) gives another infinite series

$$\begin{aligned} M &= \prod_{i=1}^{\infty}(1-x_i)^{-1} \\ &= 1 + \{1\} + \{2\} + \{3\} + \ldots \\ &= \sum_{m=0}^{\infty}\{m\} \end{aligned} \tag{2.41}$$

Clearly

$$LM = 1 \tag{2.42}$$

a result by no means obvious by simply looking at the product of the two series expressed as $S-$functions. In practice large numbers of infinite series and their associated generating functions may be constructed. The particular series implemented in SCHUR include:

A $= \sum_\alpha (-1)^{w_\alpha}\{\alpha\}$	B $= \sum_\beta \{\beta\}$	C $= \sum_\gamma (-1)^{w_\gamma/2}\{\gamma\}$
D $= \sum_\delta \{\delta\}$	E $= \sum_\epsilon (-1)^{(w_\epsilon+r)/2}\{\epsilon\}$	F $= \sum_\zeta \{\zeta\}$
G $= \sum_\epsilon (-1)^{(w_\epsilon-r)/2}\{\epsilon\}$	H $= \sum_\zeta (-1)^{w_\zeta}\{\zeta\}$	L $= \sum_m (-1)^m\{1^m\}$
M $= \sum_m \{m\}$	P $= \sum_m (-1)^m\{m\}$	Q $= \sum_m \{1^m\}$

(2.43)

where (α) and (γ) are mutually conjugate partitions, which in the Frobenius notation take the form

$$(\alpha) = \begin{pmatrix} a_1 & a_2 & \dots & a_r \\ a_1+1 & a_2+1 & \dots & a_r+1 \end{pmatrix} \quad (\gamma) = \begin{pmatrix} a_1+1 & a_2+1 & \dots & a_r+1 \\ a_1 & a_2 & \dots & a_r \end{pmatrix} \tag{2.44}$$

(δ) is a partition into *even parts* only and (β) is conjugate to (δ). (ζ) is any partition and (ϵ) is any self-conjugate partition. r is the Frobenius rank of (α), (γ) and (ϵ).

These series occur in mutually inverse pairs:

$$AB = CD = EF = GH = LM = PQ = \{0\} = 1 \tag{2.45}$$

Furthermore,

$$\begin{aligned} LA = PC = E \quad & MB = QD = F \\ MC = AQ = G \quad & LD = PB = H \end{aligned} \tag{2.46}$$

SCHUR also makes use of the series

$$R = \{0\} - 2\sum_{a,b}(-1)^{a+b+1}\binom{a}{b} \qquad S = \{0\} + 2\sum_{a,b}\binom{a}{b} \tag{2.47}$$

where we have again used the Frobenius notation, and

$$\begin{aligned} V = \sum_{\omega}(-1)^q\{\omega'\} \quad & W = \sum_{\omega}(-1)^q\{\omega\} \\ X = \sum_{\omega}\{\omega'\} \quad & Y = \sum_{\omega}\{\omega\} \end{aligned} \tag{2.48}$$

where (ω) is a partition of an even number into at most two parts, the second of which is q, and ω' is the conjugate of ω. We have the further relations

$$RS = VW = \{0\} = 1 \tag{2.49}$$

and

$$\begin{aligned} PM = AD = W \quad & LQ = BC = V \\ MQ = FG = S \quad & LP = HE = R \end{aligned} \tag{2.50}$$

Finally, there is the series

$$T = 1 + \{1\} + \{21\} + \{321\} + \{4321\} + \dots \tag{2.51}$$

2.23 Symbolic Manipulation

The above relations lead to a method of describing many of the properties of groups via symbolic manipulation of infinite series of $S-$functions. Thus if $\{\lambda\}$ is an $S-$function then we may symbolically write, for example,

$$\{\lambda/M\} = \sum_{m}\{\lambda/m\} \tag{2.51}$$

We can construct quite remarkable identities such as:

$$BD = \sum_{\zeta} \{\zeta\} \cdot \{\zeta\} \tag{2.52}$$

or for an arbitrary $S-$function $\{\epsilon\}$

$$BD \cdot \{\epsilon\} = \sum_{\zeta} \{\zeta\} \cdot \{\zeta/\epsilon\} \tag{2.53}$$

Equally remarkably we can find identities such as

$$\{\sigma \cdot \tau\}/Z = \{\sigma/Z\} \cdot \{\tau/Z\} \qquad \text{for} \qquad Z = L, M, P, Q, R, S, V, W \tag{2.54a}$$

$$\{\sigma \cdot \tau\}/Z = \sum_{\zeta} \{\sigma/\zeta Z\} \cdot \{\tau/\zeta Z\} \qquad \text{for} \qquad Z = B, D, F, H \tag{2.54b}$$

$$\{\sigma \cdot \tau\}/Z = \sum_{\zeta} (-1)^{w_\zeta} \{\sigma/\zeta Z\} \cdot \{\tau/\zeta' Z\} \qquad \text{for} \quad Z = A, C, E, G \tag{2.54c}$$

These various identities lead to a symbolic method of treating properties of groups particulary amenable to computer implementation.

■ 2.24 The $U_n \to U_{n-1}$ branching rule

As an illustration of the preceding remarks we apply the properties of $S-$functions to the determination of the $U_n \to U_{n-1}$ branching rules. The vector irreducible representation$\{1\}$ of U_n can be taken as decomposing under $U_n \to U_{n-1}$ as

$$\{1\} \to \{1\} + \{0\} \tag{2.55}$$

that is into a vector $\{1\}$ and scalar $\{0\}$ of U_{n-1}. In general, the vector spaces corresponding to tensors for which a particular number of indices, say m, take on the value n, define invariant subspaces. Such indices must be mutually symmetrised. The irreducible representations specified by the quotient $\{\lambda/m\}$ are those corresponding to tensors obtained by contracting the indices of the tensor corresponding to $\{\lambda\}$ with an $m-$th rank symmetric tensor. Thus we may symbolically write the general branching rule as simply

$$\{\lambda\} \to \{\lambda/M\} \tag{2.56}$$

In this example the infinite $M-$series is truncated by the requirement that $\lambda_1 \geq m$ to yield a non-vanishing skew diagram. Thus for example under $U_3 \to U_2$ we have

$$\begin{aligned} \{21\} &\to \{21/M\} \\ &\to \{21/0\} + \{21/1\} + \{21/2\} \\ &\to \{21\} + \{2\} + \{11\} + \{1\} \end{aligned}$$

■ 2.25 Schur's Q-functions

Schur's $Q-$functions play an important role in determining the properties of the spin (or projective) representations of the symmetric group S_n. They are made

use of in SCHUR in calculating $O_n \to S_n$ branching rules for the spin irreducible represention of the full orthogonal group O_n and in evaluating Kronecker products in S_n involving the spin representations. SCHUR also contains a number of commands for obtaining properties of $Q-$functions such as those for calculating the analogue of the Littlewood-Richardson rule. Within SCHUR, hidden from the user, are routines that make use of the concept of shifted Young diagrams and tableaux as well as many other properties such as shifted lattice properties etc. These will be familiar to the experts and for the most part of minor interest to the average user of SCHUR. Here we sketch only the briefest of details.

A shifted diagram is a diagonally adjusted Young diagram with the restriction that the $(i+1)-$th row does not exceed the $i-$th row. This condition restricts us to partitions λ involving only distinct parts. That is $\lambda \in \mathbf{DP}$ where $\mathbf{DP}$ stands for the set of partitions involving distinct parts. Formally we have

Definition *For each* $\lambda \in \mathbf{DP}$ *there is an associated shifted diagram defined as*

$$\mathrm{D}'_\lambda = \{(i,j) \in \mathrm{Z}^2 : i \le j \le \lambda_i + i - 1, 1 \le i \le \ell_\lambda\}$$

As an example, for the partition (65421) we have the unshifted and shifted diagrams

Shifted Young tableaux may be defined as follows. Let $\mathbf{P'}$ denote the ordered alphabet $\{1' < 1 < 2' < 2 \ldots\}$. The letters $1', 2', \ldots$ are said to be *marked*.

Definition *A shifted tableau* T *of shape* λ *is an assignment* $T : \mathbf{D}'_\lambda \to \mathbf{P'}$ *satisfying the following conditions:*

1. $T(i,j) \le T(i+1,j), \quad T(i,j) \le T(i,j+1)$
2. *Each column has at most one* $k \quad (k = 1, 2, \ldots)$
3. *Each row has at most one* $k' \quad (k' = 1', 2', \ldots)$

A typical shifted Young tableau associated with the partition (65421) and satisfying the above conditions is

$1'$	1	1	$2'$	2	$6'$
	2	2	2	$5'$	7
		$3'$	$5'$	5	8
			$5'$	6	
				7	

Let γ_k denote the number of boxes $(i,j) \in \mathbf{D}'_\lambda$ such that $|T(i,j)| = k$ where $|k'| = |k| = k$. Then the *content* $\gamma = (\gamma_1, \gamma_2, \ldots)$ and we may associate a monomial $\mathbf{x}^T = \mathbf{x}^\gamma = x_1^{\gamma_1} x_2^{\gamma_2} \ldots$ with the tableau T. Schur's $Q-$function $Q_\lambda(\mathbf{x})$

in the variables $\mathbf{x} = (x_1, x_2, \ldots)$ with $\lambda \in \mathbf{DP}$ may be defined as

$$Q_\lambda(\mathbf{x}) = \sum_{T:D'_\lambda \to \mathbf{P}'} \mathbf{x}^T \tag{2.57}$$

where the summation is over all shifted tableaux T.

In exactly the same manner a skew $Q-$function may be defined in terms of shifted skew tableaux as

$$Q_{\lambda/\mu}(\mathbf{x}) = \sum_{T:D'_{\lambda/mu} \to \mathbf{P}'} \mathbf{x}^T \tag{2.58}$$

where the summation is over all the shifted skew tableaux $T(\lambda/\mu)$. The skew $Q-$function may be expanded as a sum of $Q-$function

$$Q_{\lambda/\mu} = \sum_\nu f^\lambda_{\mu\nu} Q_\nu \tag{2.59}$$

where $\mu, \nu, \lambda \in \mathbf{DP}$, $w_\lambda = w_\mu + w_\nu$, $\lambda \supseteq \mu, \nu$. The coefficients $f^\lambda_{\mu\nu}$ are non-negative integers. The same coefficients appear in the $Q-$function outer product

$$Q_\mu \cdot Q_\nu = \sum_\lambda 2^{[\ell_\mu + \ell_\nu - \ell_\lambda]} f^\lambda_{\mu\nu} Q_\lambda \tag{2.60}$$

Occasionally SCHUR makes use of the $P-$functions which are related to the $Q-$functions.

$$P_\lambda(x) = 2^{-\ell_\lambda} Q_\lambda(x)$$

SCHUR contains routines that compute properties of $Q-$functions such as outer products and skew quotients as well as various other properties including inner and reduced inner products etc.

■ 2.26 Non-Standard Q-functions

Schur's $Q-$functions are indexed by ordered partitions having distinct parts. However, situations can arise involving non-standard partitions. In those cases modification rules must be applied. Any list of $Q-$functions can be converted into a list of standard $Q-$functions by sequential application of the following four rules to the list:

1. The parts of partition are first ordered so that the absolute magnitude of the parts are in descending order when read from left to right. This is achieved by repeated use of

   $$Q_{(\ldots,\lambda_i,\lambda_{i+1},\ldots)} = -Q_{(\ldots,\lambda_{i+1},\lambda_i,\ldots)}$$

   whenever $|\lambda_{i+1}| > |\lambda_i|$, remembering that $Q_{(\mu,0)} = Q_{(\mu)}$.
2. $Q-$functions indexed by partitions with *consecutive* repeated parts are null.

3. $Q-$functions where a negative part $-\lambda_p$ precedes λ_p are modified by use of the identity

$$Q_{(\lambda_1,\ldots,-\lambda_p,\lambda_p,\ldots,\lambda_k)} = (-1)^{\lambda_p} 2 Q_{(\lambda_1,\ldots,\lambda_k)}$$

4. Any remaining $Q-$function indexed by a partition containing a negative part is null.

The above operations are automatically implemented in SCHUR.

2.27 Young's Raising Operators

The Young raising operator R_{ij} operates on a partition (λ) by *increasing* the part λ_i by one and *decreasing* λ_{i+1} by one with $i < j$. The operators

$$\prod_{i<j}(1 \pm R_{ij}) \quad \text{and} \quad \prod_{i<j}(1 \pm R_{ij})^{-1} \tag{2.61}$$

play an important role in the theory of symmetric functions such as the $S-$functions and the $Q-$functions. In the process of using them non-standard $S-$ or $Q-$functions may arise which must be modified using the appropriate standardisation rules. These rules may only be applied *after* all the operations of the Young operators have been applied. All four operators defined in Eq. (2.61) are implemented in SCHUR for action of $S-$functions and $Q-$functions as appropriate.

Young raising operators may also be defined which act on partitions in reduced notation.

You boil it in sawdust: you salt it in glue;
You condense it with locusts and tape
Still keeping one principal object in view
To preserve its symmetrical shape.
— Lewis Carroll *The Hunting of the Snark*

3

Mathematical Background to SCHUR

Notation for Lie Groups

Introduction

This is not the place for a detailed discussion of the properties of Lie groups. Books in abundance exist on that subject. However to make effective use of SCHUR it is essential to have an understanding of the notation used in SCHUR for individual Lie groups and in particular the choice adopted for referring to labelling irreducible representations. As noted earlier SCHUR uses partition based labels as distinct from the more widely used Dynkin labelling scheme. However SCHUR does allow the user to relate its partition labels to the corresponding Dynkin labels.

3.1 Unitary Group Labels

Irreducible representations of the unitary group U_n are labelled by partitions λ, into at most n parts, of ordered integers enclosed in brackets {, } to give $\{\lambda\}$. The partition $\lambda \vdash \ell$ serves to specify the symmetry properties of the corresponding $\ell-$th rank *covariant* tensor forming the basis of the representation. There also exist inequivalent irreducible representations of U_n associated with the $m-$th rank *contravariant* tensors specified by $\{\overline{\mu}\}$. Irreducible representations associated with *mixed* tensors are designated by writing $\{\overline{\mu};\lambda\}$ where the partition $\lambda \vdash \ell$ specifies the symmetry of the ℓ covariant indices and $(\overline{\mu})$, with $\mu \vdash \ell$, that of the m contravariant indices.

3.2 Orthogonal and Symplectic Group Labels

The covariant tensor irreducible representations $\{\lambda\}$ of U_n form a basis for the tensor irreducible representations of subgroups of U_n, including the orthogonal group O_n and, if n is even, the symplectic group Sp_n. However these representations are, in general, reducible due to the existence of the symmetric and antisymmetric metric tensors of O_n and Sp_n respectively. The representations may be reduced by extracting all possible trace terms formed by contraction with the metric tensor. In this way irreducible representations of O_n and Sp_n are obtained. In SCHUR the tensor irreducible representations of O_n are labelled by ordered partitions λ into at most k parts (where $n = 2k+1$ or $n = 2k$) with the partition enclosed in brackets [,] to give $[\lambda]$. Likewise the irreducible representations of Sp_{2k} are labelled by ordered partitions λ having at most k parts with the partition enclosed in brackets $<, >$ to give $< \lambda >$.

As well as the tensor irreducible representations labelled by $[\lambda]$, the group O_n also has *double-valued* or *spin* representations denoted by $[\Delta;\lambda]$ where Δ is the fundamental spin irreducible representation of dimension 2^k. To avoid introducing special type SCHUR outputs the spin irreducible representations as $[s;\lambda]$.

3.3 Associate Irreducible Representations

For all linear groups there exist among the irreducible representations a one-dimensional irreducible representation, denoted here by ϵ, which maps each group element onto the value of its determinant. By definition all the elements of the unimodular groups SU_n, SO_n and Sp_n have determinant $+1$, so that irreducible

representation ϵ for these groups coincides with the identity irreducible representations $\{0\}$, $[0]$ and $< 0 >$ respectively. However, for U_n and O_n this is not the case. For U_n ϵ is the irreducible representation $\{1^n\}$ with an inverse $\epsilon^{-1} = \overline{\epsilon} = \{\overline{1^n}\}$. For O_n all the group elements have determinant ± 1 and hence $\epsilon^{-1} = \overline{\epsilon}$ and $\overline{\epsilon} \times \epsilon = [0]$.

The product of ϵ with any irreducible representation is also an irreducible representation, and inequivalent irreducible representations related by some power of ϵ are said to be *associated*. For U_n there are an infinite number of inequivalent irreducible representations associated with a given irreducible representation, one of which will be specified by a partition having less than n parts. For instance $\{\overline{66521}\}$, $\{\overline{5541}\}$, $\{\overline{443};1\}$, $\{\overline{332};21\}$, $\{\overline{221};32\}$, $\{\overline{11};43\}$,$\{541\}$, $\{65211\}$, ... are all mutually associated irreducible representations of U_5. More generally the Kronecker product $\epsilon^r\{\overline{\mu};\lambda\} = \epsilon^r \times \{\overline{\mu};\lambda\}$ is an irreducible representation of U_n associated to the irreducible representation $\{\overline{\mu};\lambda\}$ for any real value of r. If r is not an integer then such irreducible representations are strictly speaking not true irreducible representations of U_n since they are multivalued. In particular, if r is half an odd integer they are analogous to the spinor irreducible representations of O_n which are of course double-valued. The fact that on restriction from U_n to SU_n $\epsilon \rightarrow \{0\}$ implies that all mutually associated irreducible representations of U_n give equivalent irreducible representations of SU_n under this restriction. Moreover, each inequivalent irreducible representation of SU_n may be denoted by means of a partition into less than n parts.

■ 3.4 Irreducible Representations of O_n and SO_n

In the case of O_n any given irreducible representation can possess at most one inequivalent associated irreducible representation. Irreducible representations for which the character is zero for all the group elements having determinant -1 possess an associate which is equivalent to itself. Such irreducible representations are termed *self-associate*. For O_{2k} all the spin irreducible representations and the tensor irreducible representations labelled by partitions having exactly k non-zero parts are self-associate. The remaining tensor irreducible representations of O_{2k} and all irreducible representations , spin and tensor, of O_{2k+1} are not self-associate. Associated pairs of irreducible representations are denoted in SCHUR by $[\lambda]$ and $[\lambda]\# = \epsilon \times [\lambda]$ and by $[\Delta;\lambda]$ and $[\Delta;\lambda]\# = \epsilon \times [\Delta;\lambda]$.

Under the restriction $O_n \rightarrow SO_n$ the distinction between an irreducible representation and its associate is lost. It might be expected that the labels $[\lambda]$ and $[\Delta;\lambda]$ would suffice for the irreducible representations of SO_n. This is not the case. Only those irreducible representations of O_n which are not self-associate remain irreducible under the restriction $O_n \rightarrow SO_n$. In contrast, each self-associate irreducible representation of O_{2k} yields on restriction to SO_{2k} two inequivalent irreducible representations of the same dimension. These pairs of irreducible representations are conviently specified by the additional labels $\pm$ to give $[\lambda]\pm$ and $[\Delta;\lambda]\pm$ where in the former case λ is necessarily a partition into k non-zero parts.

■ 3.5 Irreducible Representations of Exceptional Groups

The irreducible representations of the five exceptional groups G_2, F_4, E_6, E_7 and E_8 may be conveniently labelled by exploiting the notation introduced to label the irreducible representations of maximal classical subgroups of the same rank. Many possible maximal embeddings exist. In SCHUR the embeddings have been chosen as those associated with $G_2 \supset SU_3$, $F_4 \supset SO_9$, $E_6 \supset SU_2 \times SU_6$, $E_7 \supset SU_8$ and $E_8 \supset SU_9$. As a consequence every irreducible representation of the exceptional Lie groups may be uniquely labelled by a constrained set of partitions. SCHUR outputs the labelled irreducible representations of the exceptional groups enclosed by brackets (,). The partition labels associated with the exceptional groups E_6, E_7 and E_8 carry the additional constraint that the weight of a partition label for E_6 or E_7 must be *even* while that for E_8 must be 0 mod 3.

In the case of G_2 it is common for physicists to employ the Racah notation $(u_1 u_2)$ with $u_1 \geq u_2$ instead of the SU_3 based $(\ l_1,\ l_2)$. The two labelling schemes are related by

$$l_1 = \mu_1 - \mu_2 \qquad \text{and} \qquad l_2 = \mu_2 \tag{3.1}$$

■ 3.6 The Super Lie Groups

SCHUR handles some of the properties of the super Lie groups $U_{m/n}$, $SU_{m/n}$ and $OSp_{m/n}$. Again certain irreducible representations may be labelled by partitions. For $U_{m/n}$ and $SU_{m/n}$ SCHUR outputs the partitions enclosed in the brackets $\{,\}$ while for the orthosymplectic groups $OSp_{m/n}$ the bracket pair $[,>$ is used.

It should be pointed out that although mixed tensor characters $\{\overline{\mu};\lambda\}$ of $U_{m/n}$ and $SU_{m/n}$ are well defined, they do not in general correspond to irreducible representations. The same is true, in general, of the characters $[\lambda >$ od $OSp_{m/n}$. Only the characters $\{\lambda\}$ of $U_{m/n}$ and $SU_{m/n}$ correspond unequivocally to irreducible representations.

■ 3.7 Notation for the Symmetric and Alternating Groups

In the case of the symmetric and alternating groups, S_n and A_n, the ordinary irreducible representations are labelled by the partitions $\lambda \vdash n$ and are output by SCHUR enclosed in the brackets $\{,\}$. Conjugate pairs of irreducible representations of A_n are distinguished by attachment of the labels $\pm$. The two-valued spin representations of S_n correspond to single-valued representations of the covering group Γ_n and may be labelled uniquely by partitions $\lambda \vdash n$ having distinct parts. In SCHUR we exploit the fact that under $O_n \to S_n$ the basic spin representation Δ of O_n is irreducible if n is *even* or otherwise it decomposes into an associated pair of irreducible representations of S_n. In order to maintain the close connection between the two groups we choose to label the spin irreducible representations of S_n using the notation $\{\Delta;\lambda\}$ where λ is a partition into distinct

parts with the added constraints

$$n \quad odd \quad \lambda_1, \omega_\lambda \leq \frac{n-1}{2} \qquad n \quad even \quad \lambda_1 + 1, \omega_\lambda \leq \frac{n}{2} \tag{3.2}$$

If λ is a partition into k parts and $(n-k)$ is *even* we obtain an associated pair of irreducible representations which we distinguish by attaching the labels $\pm$ by analogy with SO_n. A similar labelling is adopted for the spin irreducible representations of A_n. SCHUR will normally output the spin representations using s.

■ 3.8 Standard Labels for Lie Groups

The labels introduced here serve to emphasise the tensor or spin nature of the irreducible representations. The labels adopted in SCHUR for the Lie groups of rank k are summarised in Table 3.1.

Table 3.1 Standard labels for irreducible representations of the Lie groups of rank k.

Group	**Label**	**Constraints**
U_n	$\{\overline{\mu};\lambda\}$	$\ell_\lambda + \ell_\mu \leq k = n$
SU_n	$\{\lambda\}$	$\ell_\lambda \leq k = n-1$
O_{2k+1}	$[\lambda], [\lambda]\#$	$\ell_\lambda \leq k$
	$[\Delta;\lambda], [\Delta;\lambda]\#$	$\ell_\lambda \leq k$
SO_{2k+1}	$[\lambda], [\Delta;\lambda]$	$\ell_\lambda \leq k$
Sp_{2k}	$<\lambda>$	$\ell_\lambda \leq k$
O_{2k}	$[\lambda], [\lambda]\#$	$\ell_\lambda < k$
	$[\lambda]$	$\ell_\lambda = k$
	$[\Delta;\lambda]$	$\ell_\lambda \leq k$
SO_{2k}	$[\lambda]$	$\ell_\lambda < k$
	$[\lambda]\pm$	$\ell_\lambda = k$
	$[\Delta;\lambda]\pm$	$\ell_\lambda \leq k$
G_2	(λ)	$\ell_\lambda \leq 2, \lambda_1 \geq 2\lambda_2$
F_4	(λ)	$\ell_\lambda \leq 4, \lambda_1 > \lambda_2 + \lambda_3 + \lambda_4$
	$(\Delta;\lambda)$	$\ell_\lambda \leq 4, \lambda_1 > \lambda_2 + \lambda_3 + \lambda_4$
E_6	$(s;\lambda)$	$\ell_\lambda \leq 5, s \geq \lambda_1 + \lambda_2 + \lambda_3 - \lambda_4 - \lambda_5$
E_7	(λ)	$\ell_\lambda \leq 7, \lambda_1 \geq \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 - \lambda_6 - \lambda_7$
E_8	(λ)	$\ell_\lambda \leq 8$
		$\lambda_1 \geq 2\lambda_2 + 2\lambda_3 + 2\lambda_4 - \lambda_5 - \lambda_6 - \lambda_7 - \lambda_8$
$U_{m/n}$	$\{\overline{\mu};\lambda\}$	$\lambda_m \leq n$
$SU_{m/n}$	$\{\lambda\}$	$\lambda_{m+1} \leq n$
$OSp_{m/n}$	$[\lambda>, [\Delta;\lambda>$	$\lambda_m \leq n$

■ 3.9 Standard Labels and Dynkin Labels

The standard or natural labels λ used by SCHUR to label irreducible representations of Lie groups gives a unique prescription. Other methods of labelling irreducible representations exist, most notably that of Dynkin wherein every irreducible representation of a semisimple Lie group is uniquely labelled by a vector **a** of non-negative integers with components $a_i \quad (i = 1, \ldots, k)$ associated with each simple root $\mathbf{r}(i)$ of the corresponding Lie algebra of rank k. This amounts to assigning non-negative integers to the nodes of the appropriate Coxeter-Dynkin diagram. The precise relationship between SCHUR 's standard partition labels λ and the corresponding Dynkin labels are given in Table 3.2a for the classical Lie groups and in Table 3.2b for the exceptional groups. In the latter case they are appropriate to the classical group embeddings given in §3.5.

Table 3.2a Relationship between standard SCHUR labels and the corresponding Dynkin labels for the classical Lie groups.

Group	**Dynkin label**	**Standard Label**
SU_{k+1}	$a_1 = l_1 - l_2$	$l_1 = a_1 + a_2 + \ldots + a_{k-1} + a_k$
	$a_2 = l_2 - l_3$	$l_2 = a_2 + \ldots + a_{k-1} + a_k$
	$\vdots$	$\vdots$
	$a_{k-1} = l_{k-1} - l_k$	$l_{k-1} = a_{k-1} + a_k$
	$a_k = l_k$	$l_k = a_k$
SO_{2k+1}	$a_1 = l_1 - l_2$	$l_1 = a_1 + a_2 + \ldots + a_{k-1} + \frac{a_k}{2}$
	$a_2 = l_2 - l_3$	$l_2 = a_2 + \ldots + a_{k-1} + \frac{a_k}{2}$
	$\vdots$	$\vdots$
	$a_{k-1} = l_{k-1} - l_k$	$l_{k-1} = a_{k-1} + \frac{a_k}{2}$
	$a_k = 2\, l_k$	$l_k = \frac{a_k}{2}$
Sp_{2k}	$a_1 = l_1 - l_2$	$l_1 = a_1 + a_2 + \ldots + a_{k-1} + a_k$
	$a_2 = l_2 - l_3$	$l_2 = a_2 + \ldots + a_{k-1} + a_k$
	$\vdots$	$\vdots$
	$a_{k-1} = l_{k-1} - l_k$	$l_{k-1} = a_{k-1} + a_k$
	$a_k = l_k$	$l_k = a_k$
SO_{2k}	$a_1 = l_1 - l_2$	$l_1 = a_1 + a_2 + \ldots + a_{k-2} + \frac{a_{k-1}}{2} + \frac{a_k}{2}$
	$a_2 = l_2 - l_3$	$l_2 = a_2 + \ldots + a_{k-2} + \frac{a_{k-1}}{2} + \frac{a_k}{2}$
	$\vdots$	$\vdots$
	$a_{k-2} = l_{k-2} - l_{k-1}$	$l_{k-2} = a_{k-2} + \frac{a_{k-1}}{2} + \frac{a_k}{2}$
	$a_{k-1} = l_{k-1} - l_k$	$l_{k-1} = \frac{a_{k-1}}{2} + \frac{a_k}{2}$
	$a_k = l_{k-1} + l_k$	$l_k = \frac{a_{k-1}}{2} - \frac{a_k}{2}$

Table 3.2b Relationship between standard SCHUR labels and the corresponding Dynkin labels for the exceptional Lie groups.

Group	Dynkin label	Standard Label
G_2	$a_1 = l_2$	$l_1 = 2a_1 + a_2$
	$a_2 = l_1 - 2\,l_2$	$l_2 = a_1$
F_4	$a_1 = l_2 - l_3$	$l_1 = a_1 + 2a_2 + \frac{3a_3}{2} + a_4$
	$a_2 = l_3 - l_4$	$l_2 = a_1 + a_2 + \frac{a_3}{2}$
	$a_3 = 2\,l_4$	$l_3 = a_2 + \frac{a_3}{2}$
	$a_4 = l_1 - l_2 - l_3 - l_4$	$l_4 = \frac{a_3}{2}$
E_6	$a_1 = l_2 - l_3$	$l_1 = a_1 + 2a_2 + 3a_3 + 2a_4 + a_5 + 2a_6$
	$a_2 = l_3 - l_4$	$l_2 = a_1 + a_2 + a_3 + a_4 + a_5$
	$a_3 = l_4 - l_5$	$l_3 = a_2 + a_3 + a_4 + a_5$
	$a_4 = l_5 - l_6$	$l_4 = a_3 + a_4 + a_5$
	$a_5 = l_6$	$l_5 = a_4 + a_5$
	$a_6 = \frac{1}{2}(\,l_1 - l_2 - l_3 - l_4 + l_5 + l_6)$	
		$l_6 = a_5$
E_7	$a_1 = l_7$	$l_1 = 2a_1 + 3a_2 + 4a_3 + 3a_4 + 2a_5 + a_6 + 2a_7$
	$a_2 = l_6 - l_7$	$l_2 = a_1 + a_2 + a_3 + a_4 + a_5 + a_6$
	$a_3 = l_5 - l_6$	$l_3 = a_1 + a_2 + a_3 + a_4 + a_5$
	$a_4 = l_4 - l_5$	$l_4 = a_1 + a_2 + a_3 + a_4$
	$a_5 = l_3 - l_4$	$l_5 = a_1 + a_2 + a_3$
	$a_6 = l_2 - l_3$	$l_6 = a_1 + a_2$
	$a_7 = \frac{1}{2}(\,l_1 - l_2 - l_3 - l_4 - l_5 + l_6 + l_7)$	
		$l_7 = a_1$
E_8	$a_1 = l_8$	$l_1 = 2a_1 + 3a_2 + 4a_3 + 5a_4 + 6a_5 + 4a_6 + 2a_7 + 3a_8$
	$a_2 = l_7 - l_8$	$l_2 = a_1 + a_2 + a_3 + a_4 + a_5 + a_6 + a_7$
	$a_3 = l_6 - l_7$	$l_3 = a_1 + a_2 + a_3 + a_4 + a_5 + a_6$
	$a_4 = l_5 - l_6$	$l_4 = a_1 + a_2 + a_3 + a_4 + a_5$
	$a_5 = l_4 - l_5$	$l_5 = a_1 + a_2 + a_3 + a_4$
	$a_6 = l_3 - l_4$	$l_6 = a_1 + a_2 + a_3$
	$a_7 = l_2 - l_3$	$l_7 = a_1 + a_2$
	$a_8 = \frac{1}{3}(\,l_1 - 2\,l_2 - 2\,l_3 - 2\,l_4 + l_5 + l_6 + l_7 + l_8)$	
		$l_8 = a_1$

Users of table 3.2a should note that for SO_{2k} with $\lambda_k \neq 0$

$$[\lambda]_{\pm} \equiv [\lambda_1, \lambda_2, \ldots, \pm\lambda_k]$$

For SO_{2k+1}, O_{2k+1} and O_{2k} the spin representations

$$\begin{aligned}[\Delta;\mu] &\equiv [\mu_1+\frac{1}{2},\mu_2+\frac{1}{2},\ldots,\mu_k+\frac{1}{2}]\\ &\equiv [\lambda_1,\lambda_2,\ldots,\lambda_k]\end{aligned}$$

while for SO_{2k}

$$\begin{aligned}[\Delta;\mu]\pm &\equiv [\mu_1+\frac{1}{2},\mu_2+\frac{1}{2},\ldots,\pm\mu_k\pm\frac{1}{2}]\\ &\equiv [\lambda_1,\lambda_2,\ldots,\lambda_k]\end{aligned}$$

■ 3.10 Modification Rules

Non-standard labelled representations may arise in the process of evaluating Kronecker products or branching rules. These do not conform to the constraints given in Table 3.1. SCHUR contains a set of modification rules that automatically transform non-standard labelled representations into either null objects or into standard labelled representations. These modification rules are accessible to the user. This feature is especially important for users putting together their own functions where it may be essential to employ the modification rules. For the classical Lie groups the modification rules involve drawing the Young frame associated with the non-standard partition λ and removing a continuous strip of boxes of length h, starting at the foot of the first column and working up along the right edge. The strip of boxes removed is known as the *striplength*. The strip removal is symbolised by $\lambda - h$. A phase factor also occurs which is dependent upon the column c in which the removal ends. If the resulting Young diagram corresponds to an ordered partition then $\lambda \rightarrow \lambda - h$, otherwise λ is discarded.

The modification rules appropriate to the classical groups are summarised in Table 3.3. It should be noted that the result of a single modification may still leave a non-standard label and the process must be repeated until either a standard label, or a null result, is obtained. Fortunately the SCHUR modification routines carry out the process to completion. The rules for the exceptional groups and for the super groups in Table 3.1 are implemented in SCHUR. In the case of the super groups the modification rules are such that a single non-standard labelled representation may lead to a string of modified representations.

Table 3.3 The modification rules appropriate to the classical Lie groups.

Group	**modification rule**	**h**
U_n, SU_n	$\{\overline{\mu};\lambda\} = (-1)^{c+d-1}\{\overline{\mu-h};\lambda-h\}$	$h = \ell_\mu + \ell_\lambda - n - 1 \geq 0$
O_{2k+1}	$[\lambda] = (-1)^{c-1}[\lambda-h]\#$	$h = 2\ell_\lambda - 2k - 1 > 0$
	$[\lambda]\# = (-1)^{c-1}[\lambda-h]$	$h = 2\ell_\lambda - 2k - 1 > 0$
	$[\Delta;\lambda] = (-1)^{c}[\Delta;\lambda-h]\#$	$h = 2\ell_\lambda - 2k - 2 \geq 0$
	$[\Delta;\lambda]\# = (-1)^{c}[\Delta;\lambda-h]$	$h = 2\ell_\lambda - 2k - 2 \geq 0$
SO_{2k+1}	$[\lambda] = (-1)^{c-1}[\lambda-h]$	$h = 2\ell_\lambda - 2k - 1 > 0$
	$[\Delta;\lambda] = (-1)^{c}[\Delta;\lambda-h]$	$h = 2\ell_\lambda - 2k - 2 \geq 0$
Sp_{2k}	$<\lambda>= (-1)^{c} <\lambda-h>$	$h = 2\ell_\lambda - 2k - 2 \geq 0$
O_{2k}	$[\lambda] = (-1)^{c-1}[\lambda-h]\#$	$h = 2\ell_\lambda - 2k > 0$
	$[\lambda]\# = (-1)^{c-1}[\lambda-h]$	$h = 2\ell_\lambda - 2k > 0$
	$[\Delta;\lambda] = (-1)^{c}[\Delta;\lambda-h]$	$h = 2\ell_\lambda - 2k - 1 \geq 0$
SO_{2k}	$[\lambda] = (-1)^{c-1}[\lambda-h]$	$h = 2\ell_\lambda - 2k > 0$
	$[\lambda]_\pm = (-1)^{c-1}[\lambda-h]_\mp$	$h = 2\ell_\lambda - 2k > 0$
	$[\Delta;\lambda]\pm = (-1)^{c}[\Delta;\lambda-h]\mp$	$h = 2\ell_\lambda - 2k - 1 \geq 0$

■ 3.11 Fusion Modification Rules

Fusion rules for WZW models based on SU_n and Sp_{2k} are incorporated into SCHUR. In applying such rules the standard modification rules are first applied followed by the fusion modification rules. It seems at this stage desirable to treat these operations separately rather than combining them into a single command.

■ 3.12 Dimensions of Irreducible Representations

SCHUR computes the dimensions of the irreducible representations of Lie groups and the symmetric and alternating groups. The dimensions of whole lists of irreducible representations for either single groups or product of groups may be calculated. Normally the dimensions are calculated using the standard Weyl formulae. However, for the classical Lie groups the user has the option of having SCHUR either use the standard Weyl formulae or the dimensional formulae based upon hooklengths. The latter assume that the irreducible representation labels are already in standard form whereas the former can be used even on non-standard labelled irreducible representations.

■ 3.13 Casimir and Dynkin Invariants

The Casimir and Dynkin invariants play a key role in the calculation of anomaly factors in gauge theories and in the many variants of the interacting boson model of nuclei. SCHUR calculates the eigenvalues of the standard second order Casimir invariant $C_2(\lambda)$ and the second-order Dynkin index $I_2(\lambda)$ for a given irreducible representation (λ) of a semisimple group $\mathcal{G}$ and implements the formula for the

trace of the $n-$th order Casimir invariant

$$I_N^{\{\nu\}}\{\lambda\} = \frac{N}{2^N}\sum_\rho K^\rho_{\nu\lambda}\frac{d(\rho)}{d(\lambda)}[C_2(\rho) - C_2(\nu) - C_2(\lambda)]^N \tag{3.3}$$

where the trace for the representation (λ) is calculated with respect to the representation (ν), $K^\rho_{\nu\lambda}$ is the coefficient of (ρ) in the Kronecker product $(\nu)\times(\lambda)$, $C_2(\alpha)$ is the second order Casimir invariant and $d(\rho)$ is the dimension of (ρ) and $d(\lambda)$ that of (λ).

SCHUR gives the user the choice of either using a command that uses the adjoint representation for (ν) or using a command that allows the user to specify (ν).

Note that as it stands $I_N^{\{\nu\}}\{\lambda\}$ is not necessarily an independent invariant. To form a genuine independent $N-$th order invariant it is necessary to form the appropriate linear combination of invariants. This can be readily determined by using SCHUR to compute the eigenvalues of a few simple invariants. Thus in the case of F_4 SCHUR finds

$$I_2^{(11)}(11) = 18 \qquad \text{and} \qquad I_3^{(11)}(11) = -81$$

However there is no independent third order invariant for F_4 and we have the relation

$$\frac{9}{2}I_2^{(11)}(\lambda) + I_3^{(11)}(\lambda) = 0$$

for all λ.

■ 3.14 Kronecker Products

The Kronecker product of a pair of irreducible representations, say λ and μ, of a Lie group $\mathcal{G}$ results in the creation of a list of irreducible representations

$$\lambda \times \mu = \sum_\nu m^\nu_{\lambda\mu}\nu \tag{3.4}$$

The coefficients $m^\nu_{\lambda\mu}$ are the *multiplicities* which are non-negative integers. The key to resolving Kronecker products is the determination of the multiplicities. SCHUR achieves this for all compact semisimple Lie groups using algorithms that all involve ultimately the use of the Littlewood-Richardson rule for $S-$functions. The algorithms are constructed so that SCHUR can resolve the Kronecker products of lists of irreducible representations not only for a single group $\mathcal{G}$ but also for group products such as $\mathcal{G}_1 \times \mathcal{G}_2 \ldots \mathcal{G}_n$ where n is usually limited to $n \leq 6$. While exact arithmetic is used in calculating properties such as dimensions and Casimir and Dynkin invariants multiplicities are calculated using ordinary integer arithmetic with the consequence that the maximum multiplicity output by SCHUR is limited to the computers value of *maxint*. In nearly all practical cases this imposes no limitation.

■ 3.15 Plethysms in Lie Groups

If λ is an irreducible representation of a semisimple Lie group $\mathcal{G}$ then the $n-$th power of λ may be resolved into a sum of symmetrised powers each corresponding to a plethysm of type

$$\lambda \otimes \{\mu\} = \sum_{\nu} p^{\nu}_{\lambda\mu} \nu \tag{3.5}$$

where $\mu \vdash n$ and the $p^{\nu}_{\lambda\mu}$ are non-negative integers. The ν are irreducible representations of $\mathcal{G}$. SCHUR currently evaluates plethysms for tensor irreducible representations of the groups U_n (but not for mixed tensors), SU_n, O_n, SO_{2k+1}, Sp_{2k} and for the exceptional group G_2. Thus for the 14 dimensional representation (21) of G_2 SCHUR readily finds that

$$\begin{aligned}
(21) \otimes \{3\} &= (63) + (41) + (3) + (21) + (1) \\
(21) \otimes \{21\} &= (51) + (42) + (41) + (31) + (3) + 2(21) + (2) \\
(21) \otimes \{1^3\} &= (42) + (4) + (3) + (2) + (0)
\end{aligned} \tag{3.6}$$

which is consistent with

$$(21) \times (21) \times (21) = (21) \otimes \{3\} + 2[(21) \otimes \{21\} + (21) \otimes \{1^3\}$$

The dimension of a plethysm $\lambda \otimes \{\mu\}$ may be checked by noting that if in $\mathcal{G}$ $Dim_{\mathcal{G}}(\lambda) = p$ then

$$\begin{aligned}
Dim(\lambda \otimes \{\mu\}) &= \sum_{\nu} g^{\nu}_{\lambda\mu} Dim_{\mathcal{G}}(\nu) \\
&= Dim_{U_p}(\{\mu\})
\end{aligned} \tag{3.7}$$

Thus in U_{14} we find $Dim_{U_{14}}\{3\} = 560$. Noting Eq. (3.6) and (3.7) we find after computing the dimensions of the relevant irreducible representations of G_2

$$560 = 273 + 189 + 77 + 14 + 7$$

■ 3.16 Automorphisms and Isomorphisms in Lie Groups

For low rank Lie groups it is sometimes useful to be able to exploit the locally isomorphic sets of Lie groups

$$SO_2 \sim U_1 \tag{3.8a}$$
$$SU_2 \sim SO_3 \sim Sp_2 \tag{3.8b}$$
$$SO_4 \sim SU_2 \times SU_2 \sim SO_3 \times SO_3 \sim Sp_2 \times Sp_2 \tag{3.8c}$$
$$Sp_4 \sim SO_5 \tag{3.8d}$$
$$SO_6 \sim SU_4 \tag{3.8e}$$

and their associated representation correspondences

$$[a] \sim \{a\} \tag{3.9a}$$
$$\{a\} \sim [a/2] \sim < a > \tag{3.9b}$$

$$[a,b] \sim \{a+b\} \times \{a-b\} \sim [(a+b)/2] \times [(a-b)/2]$$
$$\sim < a+b > \times < a-b > \quad (3.9c)$$
$$< a,b > \sim [(a+b)/2,(a-b)/2] \quad (3.9d)$$
$$[abc] \sim \{a+b, a-c, b-c\} \quad (3.9e)$$

In addition there is the outer automorphism of order 3 of SO_8 such that

$$[a,b,c,d] \sim \left[\frac{(\mathrm{a}+\mathrm{b}+\mathrm{c}+\mathrm{d})}{2}, \frac{(\mathrm{a}+\mathrm{b}-\mathrm{c}-\mathrm{d})}{2}, \frac{(\mathrm{a}-\mathrm{b}+\mathrm{c}-\mathrm{d})}{2}, \frac{(-\mathrm{a}+\mathrm{b}+\mathrm{c}-\mathrm{d})}{2}\right]$$
$$\sim \left[\frac{(\mathrm{a}+\mathrm{b}+\mathrm{c}-\mathrm{d})}{2}, \frac{(\mathrm{a}+\mathrm{b}-\mathrm{c}+\mathrm{d})}{2}, \frac{(\mathrm{a}-\mathrm{b}+\mathrm{c}+\mathrm{d})}{2}, \frac{(\mathrm{a}-\mathrm{b}-\mathrm{c}-\mathrm{d})}{2}\right]$$
$$\sim [a,b,c,d] \quad (3.10)$$

This automorphism relates spin and tensor irreducible representations of SO_8. For example

$$\Delta_+ \sim [1] \sim \Delta_- \sim \Delta_+ \quad (3.11)$$

These isomorphisms and automorphism are fully implemented in SCHUR.

■ 3.17 Branching Rules

The decomposition of a representation $\lambda_{\mathcal{G}}$ of a group $\mathcal{G}$ into irreducible representations $\mu_{\mathcal{H}}$ of a subgroup $\mathcal{G} \supset \mathcal{H}$ is a very frequent problem of applications of Lie groups. The subgroup $\mathcal{H}$ need not be a simple Lie group, it may be a product of several simple Lie groups, or possibly a finite group such as arises in $O_n \supset S_n$. In general

$$\lambda_{\mathcal{G}} \to \sum_{\mu_{\mathcal{H}}} m_{\lambda_{\mathcal{G}}}^{\mu_{\mathcal{H}}} \mu_{\mathcal{H}} \quad (3.11)$$

and the fundamental problem is to determine the non-negative integers $m_{\lambda_{\mathcal{G}}}^{\mu_{\mathcal{H}}}$ known as the branching multiplicities. As with Kronecker products SCHUR solves this problem using properties of $S-$functions. While such an approach suffers from an inherent lack of universality it does permit the construction of efficient algorithms for specific branching rules. SCHUR covers a large range of branching rules and allows the user to successively branch through chains of embedded groups. It is possible for the user to define his or her own functions to cover additional branching rules. The case of $SO_7 \supset SU_2 \times SU_2 \times SU_2$ is given as a detailed example of constructing a user defined function in the advanced tutorials on user defined functions. The specific branching rules are restricted to compact Lie groups and the symmetric and alternating groups however it is possible to apply SCHUR even to the determination of the branching content, up to a user defined limit, of group-subgroup combinations such as $Sp_{2k}(R) \supset U_k$.

In handling the decompositions for the spin representations in $O_n \supset S_n$ SCHUR makes extensive use of the properties of $Q-$functions which are thoughtfully hidden from the user.

■ 3.18 Odds and Ends

In this and the previous chapter we have outlined some of the mathematical foundations of SCHUR often giving only the barest details. The user wishing more details should consult the bibliography in Chapter 7. Several features of SCHUR have been omitted in this account as being of interest to only a few specialists. They should consult the Appendix A where every command in SCHUR is detailed giving information on not only the common SCHUR commands but also rather exotic commands of specialist interest only. Of course it is our ultimate hope that eventually every user will become a specialist with a deep understanding of the philosophy behind SCHUR and its many potentialities.

It has been rumoured that the "group pest" is gradually being cut out of quantum physics

—H. Weyl *The Theory of Groups and Quantum Mechanics, 1930*

We wish finally to make a few remarks concerning the place of the theory of groups in the study of the quantum mechanics of atomic spectra. The reader will have heard that this mathematical discipline is of great importance for the subject. We manage to get along without it.

— E. U. Condon and G. H. Shortley *Theory of Atomic Spectra, 1935*

4

THE NON-COMPACT GROUPS Sp(2n,R), Mp(2n) AND $SO^*(2n)$

4.1 Introduction

A knowledge of the properties of compact Lie groups, and their associated Lie algebras, is essential in many areas of physics. The unitary irreducible representations of the compact Lie groups are all of finite dimension and hence it is possible to compute complete results even though at times the dimensions may be very large. Much less is known of the non-compact Lie groups and yet they too find applications in physics. The non-compact symplectic Lie group $Sp(2n, R)$, and its metaplectic covering group, $Mp(2n)$, occur in the theory of three-dimensional harmonic oscillator ($n = 3$) and more generally in symplectic models of nuclei, quantum dots and quantum optics. A fundamental difference between the non-compact and compact Lie groups is that the non-trivial unitary irreducible representations of the former are all of infinite dimension. Thus in most cases it is only possible to compute such things as tensor products and branching rules up to some prescribed cutoff.

In this chapter we outline extensions to SCHUR that permit the calculation of Kronecker products and branching rules involving irreducible representations of the non-compact groups $Mp(2n)$, $Sp(2n, R)$, $SO^*(2n)$ and important subgroups thereof.

4.2 Labelling the Irreps of Non-compact Lie Groups

Here we shall limit ourselves to discussion of the so-called positive discrete unitary irreducible representations of the group $Sp(2n, R)$ and its double covering group, $Mp(2n)$ and the group $SO^*(2n)$. These irreducible representations are all infinite dimensional and are characterised by a *lowest weight* with respect to the ordering of weights of the maximal compact subgroup $U(n)$. There exists a harmonic representation, $\tilde{\Delta}$, associated with the Heisenberg algebra. This is a true, unitary, infinite dimensional irreducible representation of the double covering group $Mp(2n)$ of $Sp(2n, R)$, the so-called *metaplectic group*. This representation is reducible into the sum of two irreducible representations $\tilde{\Delta}_+$ and $\tilde{\Delta}_-$ whose leading weights are $(\frac{1}{2}\frac{1}{2}\ldots\frac{1}{2})$ and $(\frac{3}{2}\frac{1}{2}\ldots\frac{1}{2})$ corresponding to the highest weights of the representations $\varepsilon^{\frac{1}{2}}\{0\}$ and $\varepsilon^{\frac{1}{2}}\{1\}$ which appear in the restriction of $Sp(2n, R)$ to its maximal compact subgroup $U(n)$.

The tensor powers $\tilde{\Delta}^k$ all decompose into a direct sum of unitary irreducible representations of $Mp(2n)$. All those irreducible representations which derive from $\tilde{\Delta}^k$ for some k will be referred to as *harmonic series representaions*. All those irreducible representations that appear in $\tilde{\Delta}^k$ will be labelled by the symbols $\langle\frac{k}{2}(\lambda)\rangle$. The harmonic series representations appearing in $\tilde{\Delta}^k$ are in one-to-one correspondence with the terms arising in the branching rule appropriate to the restriction from $Mp(2nk)$ to $Sp(2n, R) \times O(k)$

$$\tilde{\Delta} \rightarrow \sum_{\lambda} \langle \frac{k}{2}(\lambda)\rangle \times [\lambda] \tag{4.1}$$

where the summation is carried out over all partitions $(\lambda) = (\lambda_1, \lambda_2, \ldots)$ for

which the conjugate partition $(\tilde{\lambda}) = (\tilde{\lambda}_1, \tilde{\lambda}_2, \ldots)$ satisfies the constraints

$$\tilde{\lambda}_1 + \tilde{\lambda}_2 \leq k \tag{4.2a}$$

and

$$\tilde{\lambda}_1 \leq n \tag{4.2b}$$

Irreducible representations of $Sp(2n, R)$ $\langle \frac{1}{2}k(\lambda)\rangle$ satisfying Eq.(4.2) will be said to be *standard* and we may limit our attention to these irreducible representations of $Sp(2n, R)$.

The value of $\frac{k}{2}$ maybe an integer (k *even*) or a half-odd-integer (k *odd*). In terms of inputting and outputting $Sp(2n, R)$ labelled irreducible representations into SCHUR it is useful to introduce the equivalent notation

$$\langle s\kappa; (\lambda)\rangle \equiv \langle \frac{k}{2}(\lambda)\rangle \tag{4.3}$$

where

$$\frac{k}{2} = s + \kappa \tag{4.4}$$

with κ being the integer part of $\frac{k}{2}$ and the residue part is $s = 0$ or $\frac{1}{2}$. Thus we have the typical notational equivalences

$$\langle s1; (\lambda)\rangle \equiv \langle \frac{3}{2}(\lambda)\rangle, k = 3 \quad \langle 1; (\lambda)\rangle \equiv \langle 1(\lambda)\rangle \quad k = 2$$

SCHUR accepts irreducible representation labels in the form of lists of $\langle s\kappa; \lambda\rangle$ and standardises the input in accordance with the constraints of Eq.(4.2) making null all non-standard $Sp(2n, R)$ irreducible representations. As an example taken from SCHUR with the inputs marked $->$ we have

```
->gr spr8
Group is Sp(8,R)
REP>
->2;211 + 2;31 + 2;2211 + s1;21 + s2;32
 <2;(21^2) > + <2;(31)> + <2;(2^2 1^2) > + <s1;(21)>
  + <s2;(32)>
REP>
->std last
  <s2;(32)> + <2;(31)> + <2;(21^2)> + <s1;(21)>
```

The second instruction has applied Eq.(4.2) to the list and eliminated the non-standard $< 2; (2211) >$ label.

4.3 Branching Rules for subgroups of $Mp(2n)$ and $Sp(2n,R)$

The branching rule for the group-subgroup decomposition $Sp(2n,R) \rightarrow U(n)$ has been shown to be

$$\langle \frac{k}{2}(\lambda) \rangle \rightarrow \varepsilon^{\frac{k}{2}} \cdot \{\{\lambda_s\}_N^k \cdot D_N\}_N \tag{4.5}$$

with $N = min(n,k)$. The infinite $S-$function series

$$D = \sum_{\delta} \{\delta\} \tag{4.6}$$

involves a sum over all partitions (δ) whose parts are even. This series is restricted to D_N in Eq.(4.6) involving members (δ) of the $D-$series having not more than N parts. Nevertheless, the series D_N remains as an infinite series of $S-$functions.

The *signed sequence* $\{\lambda_s\}_N^k$ is the set of terms $\pm\{\rho\}$ such that $\pm[\rho]$ is equivalent to $[\lambda]$ under the modification rules of the group $O(k)$. The signed sequence is rendered finite by restriction to terms $\{\rho\}$ involving not more than N parts.

The first $\cdot$ indicates a product in $U(n)$ and the second $\cdot$ a product in $U(N)$ as implied by the final subscript N.

The harmonic discrete series irreducible representations of $Sp(2n,R)$ are all of infinite dimension and hence there are an infinite number of $U(n)$ irreducible representations arising on the right-hand-side of Eq.(4.5). Clearly, in practical implementations of the branching rule a user definable cutoff must be introduced to produce a manageablely finite number of terms. In SCHUR we solve this problem by introducing a user defined integer constant **SETLIM**it that results in the computation of all terms up to a chosen maximal weight partition. SCHUR possesses procedures to generate the necessary signed sequences and $S-$function series, as well as carrying out the relevant Kronecker products and modification rules. A typical example of verbatim SCHUR input and output is given below:

```
->gr spr8
Group is Sp(8,R)
DP>
->br36,8gr1[s1;21]
Group is U(4)
  {s;11 21^2 } + {s;10 31^2 } + {s;10 2^2 1} + {s;941^2 }
+ {s;9321} + {s;921^2 } + {s;851^2 } + 2{s;8421}
+ {s;831^2 } + {s;82^2 1} + {s;761^2 } + {s;7521}
+ {s;7431} + {s;741^2 } + {s;7321} + {s;721^2 }
+ {s;6^2 21} + {s;6531} + {s;651^2 } + {s;64^2 1}
```

```
+ 2{s;6421} + {s;631^2 } + {s;62^2 1} + {s;5431}
+ {s;541^2 } + {s;5321} + {s;521^2 } + {s;4^2 21}
+ {s;431^2 } + {s;42^2 1} + {s;321^2 }
```

The branching rule for the decomposition of the irreducible representation $\tilde{\Delta} \equiv \langle s;(0)\rangle$ of the metaplectic group $Mp(2nk)$ upon restriction to the subgroup $Sp(2n,R)\times O(k)$ follows from implementation of Eq.(4.1) into SCHUR.

We find for example:

```
DP>
->sb_tex true
DP>
->columns3
DP>
->gr mp24
Group is Mp(24)
DP>
->br39,6,4gr1[s;0]
Groups are   Sp(6,R) * O(4)
```

$< 2;(12\ ) > [12\ ]$ + $< 2;(11\ 1) > [11\ 1]$ + $< 2;(11\ ) > [11\ ]$
+ $< 2;(10\ 2) > [10\ 2]$ + $< 2;(10\ 1^2) > [10\ 1^2]$ + $< 2;(10\ 1) > [10\ 1]$
+ $< 2;(10\ ) > [10\ ]$ + $< 2;(93) > [93]$ + $< 2;(92) > [92]$
+ $< 2;(91^2) > [91^2]$ + $< 2;(91) > [91]$ + $< 2;(9) > [9]$
+ $< 2;(84) > [84]$ + $< 2;(83) > [83]$ + $< 2;(82) > [82]$
+ $< 2;(81^2) > [81^2]$ + $< 2;(81) > [81]$ + $< 2;(8) > [8]$
+ $< 2;(75) > [75]$ + $< 2;(74) > [74]$ + $< 2;(73) > [73]$
+ $< 2;(72) > [72]$ + $< 2;(71^2) > [71^2]$ + $< 2;(71) > [71]$
+ $< 2;(7) > [7]$ + $< 2;(6^2) > [6^2]$ + $< 2;(65) > [65]$
+ $< 2;(64) > [64]$ + $< 2;(63) > [63]$ + $< 2;(62) > [62]$
+ $< 2;(61^2) > [61^2]$ + $< 2;(61) > [61]$ + $< 2;(6) > [6]$
+ $< 2;(5^2) > [5^2]$ + $< 2;(54) > [54]$ + $< 2;(53) > [53]$
+ $< 2;(52) > [52]$ + $< 2;(51^2) > [51^2]$ + $< 2;(51) > [51]$
+ $< 2;(5) > [5]$ + $< 2;(4^2) > [4^2]$ + $< 2;(43) > [43]$
+ $< 2;(42) > [42]$ + $< 2;(41^2) > [41^2]$ + $< 2;(41) > [41]$
+ $< 2;(4) > [4]$ + $< 2;(3^2) > [3^2]$ + $< 2;(32) > [32]$
+ $< 2;(31^2) > [31^2]$ + $< 2;(31) > [31]$ + $< 2;(3) > [3]$
+ $< 2;(2^2) > [2^2]$ + $< 2;(21^2) > [21^2]$ + $< 2;(21) > [21]$
+ $< 2;(2) > [2]$ + $< 2;(1^3) > [1^3]$ + $< 2;(1^2) > [1^2]$
+ $< 2;(1) > [1]$ + $< 2;(0) > [0]$

Note that in this case SCHUR has been requested to produce TEX output in

four columns forming a setbox with the appropriate settabs and TEX commands automatically inserted.

Under the restriction $Mp(2n) \rightarrow Sp(2n,R)$ we have

$$\langle s;(0)\rangle \rightarrow \langle s;(0)\rangle + \langle s;(1)\rangle \tag{4.7}$$

It the case of the harmonic oscillator this corresponds to separating the odd and even states. The appropriate branching rules for these two irreducible representations have been implemented in SCHUR. For example,

```
->gr spr24
Group is Sp(24,R)
DP>
->br38,6,4gr1[s;0]
Groups are   Sp(6,R) * O(4)
```

$< 3;(12\ ) > [12\ ]$ + $< 3;(11\ 1) > [11\ 1]$ + $< 3;(10\ 2) > [10\ 2]$
+ $< 3;(10\ ) > [10\ ]$ + $< 3;(93) > [93]$ + $< 3;(91) > [91]$
+ $< 3;(84) > [84]$ + $< 3;(82) > [82]$ + $< 3;(8) > [8]$
+ $< 3;(75) > [75]$ + $< 3;(73) > [73]$ + $< 3;(71) > [71]$
+ $< 3;(6^2) > [6^2]$ + $< 3;(64) > [64]$ + $< 3;(62) > [62]$
+ $< 3;(6) > [6]$ + $< 3;(5^2) > [5^2]$ + $< 3;(53) > [53]$
+ $< 3;(51) > [51]$ + $< 3;(4^2) > [4^2]$ + $< 3;(42) > [42]$
+ $< 3;(4) > [4]$ + $< 3;(3^2) > [3^2]$ + $< 3;(31) > [31]$
+ $< 3;(2^2) > [2^2]$ + $< 3;(2) > [2]$ + $< 3;(1^2) > [1^2]$
+ $< 3;(0) > [0]$

```
->gr spr24
Group is Sp(24,R)
DP>
->br38,6,4gr1[s;1]
Groups are   Sp(6,R) * O(4)
```

$< 3;(11\ ) > [11\ ]$ + $< 3;(10\ 1) > [10\ 1]$ + $< 3;(92) > [92]$
+ $< 3;(9) > [9]$ + $< 3;(83) > [83]$ + $< 3;(81) > [81]$
+ $< 3;(74) > [74]$ + $< 3;(72) > [72]$ + $< 3;(7) > [7]$
+ $< 3;(65) > [65]$ + $< 3;(63) > [63]$ + $< 3;(61) > [61]$
+ $< 3;(54) > [54]$ + $< 3;(52) > [52]$ + $< 3;(5) > [5]$
+ $< 3;(43) > [43]$ + $< 3;(41) > [41]$ + $< 3;(32) > [32]$
+ $< 3;(3) > [3]$ + $< 3;(21) > [21]$ + $< 3;(1) > [1]$

Not surprisingly the sum of the above two results coincides with those for the $Mp(24) \rightarrow Sp(6,R) \times O(4)$ decomposition given earlier.

The final branching rule we require is the general reduction for $Sp(2n,R) \rightarrow Sp(2,R) \times O(n)$. Again the relevant result is available and has been added to SCHUR. As a typical example we have:-

```
->gr spr8
Group is Sp(8,R)
DP>
->br45,8gr1[s1;21]
Groups are   Sp(2,R) * O(4)
```

$< 6;(11\ ) > [10\ 1]$
$+\ 2 < 6;(11\ ) > [9]$
$+\ < 6;(11\ ) > [74]$
$+\ 4 < 6;(11\ ) > [7]$
$+\ 10 < 6;(11\ ) > [61]$
$+\ 6 < 6;(11\ ) > [5]\#$
$+\ 13 < 6;(11\ ) > [41]$
$+\ 7 < 6;(11\ ) > [3]$
$+\ 3 < 6;(11\ ) > [1]$
$+\ < 6;(9) > [7]\#$
$+\ 4 < 6;(9) > [61]$
$+\ 3 < 6;(9) > [5]\#$
$+\ 8 < 6;(9) > [41]$
$+\ 5 < 6;(9) > [3]$
$+\ 2 < 6;(9) > [1]$
$+\ < 6;(7) > [5]\#$
$+\ 4 < 6;(7) > [41]$
$+\ 3 < 6;(7) > [3]$
$+\ 2 < 6;(7) > [1]$
$+\ < 6;(5) > [3]\#$
$+\ < 6;(5) > [1]\#$
$+\ < 6;(3) > [1]$

$+\ < 6;(11\ ) > [92]$
$+\ < 6;(11\ ) > [83]$
$+\ 5 < 6;(11\ ) > [72]$
$+\ < 6;(11\ ) > [65]$
$+\ 3 < 6;(11\ ) > [54]$
$+\ 8 < 6;(11\ ) > [5]$
$+\ 8 < 6;(11\ ) > [32]$
$+\ 7 < 6;(11\ ) > [21]$
$+\ < 6;(9) > [81]$
$+\ 2 < 6;(9) > [7]$
$+\ < 6;(9) > [54]$
$+\ 4 < 6;(9) > [5]$
$+\ 5 < 6;(9) > [32]$
$+\ 6 < 6;(9) > [21]$
$+\ < 6;(7) > [61]$
$+\ 2 < 6;(7) > [5]$
$+\ 3 < 6;(7) > [32]$
$+\ 4 < 6;(7) > [21]$
$+\ < 6;(5) > [41]$
$+\ 2 < 6;(5) > [3]$
$+\ < 6;(5) > [1]$

$+\ < 6;(11\ ) > [9]\#$
$+\ 4 < 6;(11\ ) > [81]$
$+\ 3 < 6;(11\ ) > [7]\#$
$+\ 5 < 6;(11\ ) > [63]$
$+\ 9 < 6;(11\ ) > [52]$
$+\ 6 < 6;(11\ ) > [43]$
$+\ 5 < 6;(11\ ) > [3]\#$
$+\ 2 < 6;(11\ ) > [1]\#$
$+\ < 6;(9) > [72]$
$+\ < 6;(9) > [63]$
$+\ 5 < 6;(9) > [52]$
$+\ 3 < 6;(9) > [43]$
$+\ 4 < 6;(9) > [3]\#$
$+\ < 6;(9) > [1]\#$
$+\ < 6;(7) > [52]$
$+\ < 6;(7) > [43]$
$+\ 2 < 6;(7) > [3]\#$
$+\ < 6;(7) > [1]\#$
$+\ < 6;(5) > [32]$
$+\ 2 < 6;(5) > [21]$
$+\ < 6;(3) > [21]$

The above results give an indication of the application of SCHUR to branching rules involving non-compact groups. The examples have been kept quite small but SCHUR can evaluate terms almost without limit if required.

4.4 Kronecker Products for $Sp(2n,R)$

The evaluation of Kronecker products of harmonic series irreducible representations of the non-compact group $Sp(2n,R)$ have been considered by King and Wybourne. They establish the complete result

$$\langle \frac{k}{2}(\mu)\rangle \times \langle \frac{\ell}{2}(\nu)\rangle = \langle \frac{k+\ell}{2}((\{\mu_s\}^k \cdot \{\nu_s\}^\ell \cdot D))_{k+\ell,n}\rangle \tag{4.8}$$

where $((\lambda))_{k+\ell,n}$ is interpreted as null unless the constraints of Eq.(4.2) are

satisfied.

They have conjectured the validity of a somehat simpler formula

$$\langle\frac{k}{2}(\mu)\rangle \times \langle\frac{\ell}{2}(\nu)\rangle$$
$$= \langle\frac{k+\ell}{2}((\{\mu\}\cdot\{\nu_s\}^{\ell}_{N}\cdot D_N)_N)_n \quad \text{with } N = min(n,\ell) \tag{4.9a}$$
$$= \langle\frac{k+\ell}{2}((\{\nu\}\cdot\{\mu_s\}^{\ell}_{N}\cdot D_M)_M)_n \quad \text{with } M = min(n,k) \tag{4.9b}$$

The symbol $\langle\frac{k+\ell}{2}(\lambda)\rangle_n$ is interpreted as a harmonic series irreducible representation subject to a two stage modification that first modifies (λ) in $O(k+\ell)$ and then modifies in $U(n)$. These modifications are automatically done in SCHUR .

To date no counter-example to Eq.(4.9) is known in spite of much searching. King and Wybourne have presented arguments in favour of the plausibility of their conjecture but to date no formal proof has been offered. Currently there is implementation of both Eq.(4.8) and (4.9) in SCHUR but only when a complete proof is obtained can one consider the results with certainty from Eq.(4.9). As an example of the implementation of Eq. (4.9) we have

```
REP>
->gr spr8
Group is Sp(8,R)
->p s1;21,2;31
```

$< 3;(92) >$ $+ < 3;(91^2) >$ $+ 2 < 3;(83) >$
$+ 4 < 3;(821) >$ $+ 2 < 3;(81^3) >$ $+ 3 < 3;(74) >$
$+ 7 < 3;(731) >$ $+ 4 < 3;(72^2) >$ $+ 5 < 3;(721^2) >$
$+ < 3;(72) >$ $+ < 3;(71^2) >$ $+ 2 < 3;(65) >$
$+ 8 < 3;(641) >$ $+ 8 < 3;(632) >$ $+ 8 < 3;(631^2) >$
$+ 2 < 3;(63) >$ $+ 4 < 3;(62^21) >$ $+ 4 < 3;(621) >$
$+ 2 < 3;(61^3) >$ $+ 3 < 3;(5^21) >$ $+ 7 < 3;(542) >$
$+ 6 < 3;(541^2) >$ $+ 2 < 3;(54) >$ $+ 4 < 3;(53^2) >$
$+ 8 < 3;(5321) >$ $+ 5 < 3;(531) >$ $+ 3 < 3;(52^2) >$
$+ 4 < 3;(521^2) >$ $+ < 3;(52) >$ $+ < 3;(51^2) >$
$+ 3 < 3;(4^23) >$ $+ 4 < 3;(4^221) >$ $+ 3 < 3;(4^21) >$
$+ 4 < 3;(43^21) >$ $+ 4 < 3;(432) >$ $+ 4 < 3;(431^2) >$
$+ < 3;(43) >$ $+ 2 < 3;(42^21) >$ $+ 2 < 3;(421) >$
$+ < 3;(41^3) >$ $+ < 3;(3^3) >$ $+ 2 < 3;(3^221) >$
$+ < 3;(3^21) >$ $+ < 3;(32^2) >$ $+ < 3;(321^2) >$

also in agreement with Eq.(4.8). A boolean **SB_CONJ**ecture has been added to SCHUR to permit the user to toggle between the two equations. Setting the boolean as **TRUE** leads to use of Eq.(4.8) and **FALSE** to use of Eq.(4.9).

4.5 $Sp(2n, R)$ Plethysms

The resolution of symmetrised powers of the fundamental irreps $\langle s;(0)\rangle$ and $\langle s;(1)\rangle$ is important in practical calculations. The $Sp(2n, R)$ content of such plethysms involves an infinite set of irreps and clearly, as with Kronecker products, the output must be truncated. This requires some experience. In making these calculations SCHUR first performs plethysms at the $U(n)$ level and then innverts the resulting $U(n)$ irreps to produce a list of $Sp(2n, R)$ irreps up to the weight prescribed by the value of **SETLIM**it. The plethysms at the $U(n)$ level are rendered finite by setting the value of **SET_PWT**.

The procedure is illustrated by the following example to obtain the $Sp(6, R)$ content of $\langle s;(0)\rangle \otimes \{21\}$ to weight 16 and to produce the output as a TEX table.

```
REP mode
REP>
group spr6
Group is Sp(6,R)
REP>
setlimit16
REP>
set_pwt16
REP>
sb_rev true
REP>
sb_tex true
REP>
columns2
REP>
pl s;0,21
\+$<s1;(2)>$&$ + \ <s1;(4)>$\cr
\+$ + \ <s1;(51)>$&$ + \ <s1;(6)>$\cr
\+$ + \ <s1;(71)>$&$ + \ 2<s1;(8)>$\cr
\+$ + \ <s1;(91)>$&$ + \ 2<s1;(10\ )>$\cr
\+$ + \ 2<s1;(11\ 1)>$&$ + \ 2<s1;(12\ )>$\cr
\+$ + \ 2<s1;(13\ 1)>$&$ + \ 3<s1;(14\ )>$\cr
\+$ + \ 2<s1;(15\ 1)>$&$ + \ 2<s1;(16\ )>$\cr
REP>
```

The group was set as $Sp(6, R)$. The output was limited to terms of

weight ≤ 16 by invoking the SCHUR commands **SETLIM**it and **SET_PWT**. The command **SB_REV TRUE** was issued to have the output list partitions in order increasing weight while the command **SB_TEX TRUE** instructed SCHUR to output the result in plain TEX with **COL**umns¿ set to 4. The resultant output was then capable of being trivially formed into TEX boxes. The table was then formed from the above output by writing

```
\setbox1=\vbox{\settabs3\columns{
\+$<s;(0)>\otimes\{21\}$&$<s1;(2)>$&$ + \ <s1;(4)>$\cr
\+&$ + \ <s1;(51)>$&$ + \ <s1;(6)>$\cr
\+&$ + \ <s1;(71)>$&$ + \ 2<s1;(8)>$\cr
\+&$ + \ <s1;(91)>$&$ + \ 2<s1;(10\ )>$\cr
\+&$ + \ 2<s1;(11\ 1)>$&$ + \ 2<s1;(12\ )>$\cr
\+&$ + \ 2<s1;(13\ 1)>$&$ + \ 3<s1;(14\ )>$\cr
\+&$ + \ 2<s1;(15\ 1)>$&$ + \ 2<s1;(16\ )>$\cr}}
$$\boxit{\boxit{\box1}}$$
```

which when TEX compiled yields

$< s;(0) > \otimes\{21\}$	$< s1;(2) >$	$+ < s1;(4) >$
	$+ < s1;(51) >$	$+ < s1;(6) >$
	$+ < s1;(71) >$	$+ 2 < s1;(8) >$
	$+ < s1;(91) >$	$+ 2 < s1;(10\ ) >$
	$+ 2 < s1;(11\ 1) >$	$+ 2 < s1;(12\ ) >$
	$+ 2 < s1;(13\ 1) >$	$+ 3 < s1;(14\ ) >$
	$+ 2 < s1;(15\ 1) >$	$+ 2 < s1;(16\ ) >$

There are no restrictions placed on the $Sp(2n, R)$ irrep used in the plethysm computation but the evaluation of plethysms for irreps other than the two basic irreps will be very slow.

The group $SO^*(2n)$ has recently been added to SCHUR. Plethysms for the group $SO^*(2n)$ are evaluated as for the group $Sp(2n, R)$ except for the entry of the $SO^*(2n)$ irrep $[k;(\lambda)]$ as $k;\lambda$, likewise for the evaluation of the $SO^*(2n) \rightarrow U_n$ branching rules. Extensive examples of the use of SCHURfor the group $SO^*(2n)$ can be seen at the WEB pages:-
http://www.phys.uni.torun.pl/∼bgw
At the same site there is information on using SCHURfor the non-compact group $U(p, q)$. More technical details on these groups can be found by consulting references [46], [52], [119] and [120].

5

Tutorials in using SCHUR

■ 5.1 Introduction to Tutorials

It is now time for the user to start seriously getting into SCHUR. In this chapter we present four tutorials designed to assist the user in gaining familiarity with the many features of SCHUR. It is highly desirable that the user work through the tutorials in the order they are presented. In Chapter Six we continue with two advanced tutorials which will fully demonstrate the versatility of SCHUR. Finally in Chapter Seven we give some detailed examples drawn from various areas of physics, mathematics and chemistry.

The individual commands of SCHUR are described in detail in Appendix A and the novice will need to refer to the appropriate entries as he or she proceeds through the tutorial. In most cases the abbreviated forms of the commands are used. In order to shorten the length of commands a number of other abbreviations have been used. These are given in the table below. We also repeat here the table of commands as given in Appendix A, with the abbreviated forms being printed in boldface letters.

ch = change,	conv = convert,	e = elementary
fn = function	grN = group number,	h = homogeneous
i = inner,	int = integer	m = monomial
o = outer,	op = operator,	p = power sum
p = pfn,	q = qfn ,	rd = reduce
rk = rank,	rm = remove	rp = replace
s = sfn,	sb = setboolean,	std = standardise
wr = write,	wt = weight	

Table A.1 All the commands in SCHUR.

ABSoluteValue	ADD	ALLskewSfn
ASSOCiate	ATtachPartitionToSfn	AUtoOrIsoMorphism
BRanch	BRMode	
CANcelDatFile	CASIMIRGnthTrace	CASimirNthordertrace
CH_CoeffsToOneForSfns	CH_LabelForOn	CH_PhaseOfSfns
CH_SpinIndex	CH_UoneReps	CLASS
COLumns	COMPare	COMPLement
CONJADD	CONJugateSfnList	CONSPLIT
CONTENT	CONTractGroups	CONTRAGredientRep
CONV_D_TO_Rep	CONV_D_TO_Sfn	CONV_R_TO_Sfn
CONV_S_TO_Rep	COUNTCoeffsInList	COUNTTermsInList
COVariant		
DEAD	DIMension	DPMode

D_TO_Plabel	DYNKINIndex	
END	ENTerVar	EQualSfnList
E_TO_FSymmFn	E_TO_HSymmFn	E_TO_MSymmFn
E_TO_SSymmFn	EXITmode	EXPandSfnList
FFPROD	FIRSTPart	FN
FPROD	FRACAHnotation	FROB
F_TO_ESymmFn	F_TO_HSymmFn	F_TO_MSymmFn
F_TO_SSymmFn	FUSion	
GENERIC	GENprod	GRoup
GWT		
HALLpolynomialProduct	HCLASS	HEAPstatus
HECKE	HELP	HSTD
HSTDList	H_TO_ESymmFn	H_TO_FSymmFn
H_TO_MSymmFn	H_TO_SSymmFn	
INDEXsequence	INSertPartitionIntoSfn	INTegerDivideCoeffs
INVerseseries	I_PLethysmRd	I_QfnProduct
I_sfnProduct	I_SFNQfnProduct	
KINSert	KMatrix	Kostka
LABel	LASTresult	LATticetest
LENgthOfPartitionsSelect	LICENSE	LINES
LOadFile	LOGfile	LSEQuence
MACMixedSeries	MACseries	MAKEwtOfSfnToN
MAXCoeffInList	MIXedTensorReps	M_TImesSfnProduct
M_TO_ESymmFn	M_TO_FSymmFn	M_TO_HSymmFn
M_TO_SSymmFn	MUlt_CoeffsByAnInt	MULT_List
MULT_Ntimes	MULT_PartsByAnInt	MULT_SelectInList
MULT_SPlitIntoTwoLists	MYlistOfSfns	
NLambda	NSKew	NSTDise
ONSCalar	O_PfnProduct	O_QfnProduct
O_Restrict	O_sfnProduct	
PARITYsequence	PAUSE	PLethysm
PLG	ProductKronecker	PROPertyOfRepList
P_TO_Dlabel	P_TO_SSymmFn	

QEXPandSpecialSeries QQEXpandSpecialSeries QQSEries
QSAME QSERies Q_TO_SsymmFn

RACAHnotation RAISEInverseOp RAISEop
RD_I_QfnProduct RD_I_sfnProduct RD_RAISEInverseOp
RD_RaiseOp READFnFromDisk REMark
REPmode RETurn RIEMANNList
RIEMANNPlethList RIEMANNScalarsOrderN RM_EVENPARTS
RM_EVENRkSfnsOnly RM_EVENWtInList RM_FirstPartOfSfn
RM_Group RM_NMP RM_ODDPARTS
RM_ODDRkSfnsOnly RM_ODDWtInList RM_PartitionFromSfn
RM_PARTSequalN RM_RepeatedPartsSfns RM_SOnEvenLabels
RM_UoneWtOverMax RP_FirstPartBySpin RP_RepOrSfnByWt
RP_SfnCoeffByInt RSAMEwtSfnList RULE
RVar

SAMEwtSfns SAVEsetVar SB_Bell
SB_CONjecture SB_CUT SB_Digits
SB_DIMension SB_LISToutput SB_More
SB_POWerNotation SB_PROGress SB_Qfn
SB_RD_notation SB_REVerseOrder SB_TexOutPut
SCALARInner SCHAR SERIESTErmsThatSkew
SERiesToIntWt SETFnVar SETLIMit
SET_PWT SETRVar SETSVar
SETVarInDPmode SFNmode SIGNSEQuence
SK_Pfn SK_Qfn SK_sfn
SMON SNchar SPIN
SPLitIntoSpinAndTensor SPONModify SPRCH
SPREXtend SPSTAR SQuares
STARequivalent STATusOfSchur STD
STD_OneDprep STD_Qfn S_TO_ESymmFn
S_TO_FSymmFn S_TO_HSymmFn S_TO_MSymmFn
STOP S_TO_PsymmFn S_TO_QsymmFn
SUBtract SUM SUMSQuares
SUPpressOutputToScreen SVar SWAPgroups

TABleOfBranchingRules

UONEAddInteger UONEDivInteger UONETrace

VarForDpreps VMult

WHATGroup WITH WRFNTODisk

WRfnToScreen	WSEQuence	WTofRepOrSfnSelect
YHooklengths	YOungDiagrams	YShapeSelect
Zero		

■ 5.2 Tutorial 1. : Getting Started in the SFNmode

The objective of this tutorial is to gain familiarity with input/output in SCHUR and to explore features of the SFNmode. We assume you have loaded the files supplied into the appropriate directories. To start up a SCHUR session simply go to the appropriate directory in your setup and enter the command **SCHUR** to activate the SCHUR.EXE file. (You'll probably want to set up a SCHUR.BAT file to handle your start-up). Your screen should now be exhibiting information similar to that below with a flashing cursor below DP>.

```
(If you wish to EXIT, enter 'END')
(If you wish to obtain HELP, enter '?help')
DPrep Mode (with function)
DP>
_
```

Notice the very important message telling you how to exit SCHUR and how to obtain HELP information. Let us put SCHUR into the **SFN**mode by entering the command **SFN** followed by <CR> to give:

```
sfn
Schur Function Mode
SFN>
_
```

SCHUR is case independent so it matters not that we have entered the command in lower case. Now try and repeat the **Sample: 1** input given on page 8 and follow it with the command **la**stresult. Your screen output should look like:-

```
4321+21+621+54+341+1+4~321+0+71
 {4321} + {21} + {621} + {54} + {341} + {1} + {4-321}
 + {0} + {71}
SFN>
last
 {71} + {621} + {54} + {4321} + {4-321} + {341} + {21}
 + {1} + {0}
SFN>
std last
 {71} + {621} + {54} + {4321} + {21} + {1} + {0}
SFN>
_
```

Notice how even after the command **las**t produced a nice sorted list there remained two non-standard partitions "4-321" and "341". Use of the command sequence **std last** results in a list that is correctly standardised. This is our first example of nested commands. Issue the command **yo last** and we obtain:-

```
SFN>
yo last
  OOOOOOO   OOOOOO   OOOOO   OOOO  OO   O   .
  O         OO       OOOO    OOO   O
            O                OO
                             O
SFN>
_
```

Do not ignore the important "." which represents the empty Young diagram corresponding to {0}.

An example of the Littlewood-Richardson rule is readily generated by entering the command "o21,21".If we wanted to see the Young diagrams drawn we could follow with the command "yo last". We could however get to the desired result by issuing the command sequence "yo o21,21" to give

```
SFN>
yo o21,21
                        2
  0000    0000   000   000   000   00   00
  00      0      000   00    0     00   00
          0            0     0     00   0
                               0          0
SFN>
_
```

Notice that the multiplicities other than +1 appear above the relevant Young diagram. The result of "o21,21" has not been lost as a return of "last" will quickly reveal. Entering the command "countc last" will show that the sum of the multiplicities is indeed 8. More dramatically, enter the command sequence "countc o4321,4321" and note the output is simply "CoeffSum = 930" meaning that the sum of the multiplicities is 930. How many distinct Young diagrams arise in that case? Simply enter "countt last" and find out! The result of "o4321,4321" is still there as the command "last" will quickly fill the screen and the presence of "more" on the screen awaits your pressing a key to give another full screen and some more.

By now you should be getting the idea of inputting into the SFNmode of SCHUR. Appendix A gives many examples of the use of all the commands in SCHUR. Try to reproduce some of these and then make up some examples of your own. Then try making sequences of nested commands. The following exercises are chosen to illustrate further use of SCHUR.

2.1 Study the output produced successively from the following commands "sk21,p" , "sk last,q" and "sk sk21,p,q" and explain the final result.

2.2 Study the output produced successively from the following commands "i32,41", "rd_i2,1" and "makewt5,rd_i2,1". Why is the output from the first and last command the same?

2.3 Consider the command sequence "wt4,o ser4,p,ser4,q". Before entering the command what output do you think will result? Now try it.

2.4 You want a list of all the partitions of all the integers up to n where n is not too large, say $n < 20$. Show that the command "ser n,f" yields the desired result and that "countc ser20,f" shows that the total number of partitions for $n = 0, 1, \ldots, 20$ is 2715.

2.5 Show that by issuing the command "countc wt-20ser20,f" there are 627 partitions of the integer 20. NB. On installations that are short on memory you might want to consider smaller examples.

2.6 Show that the command sequence "wt-105,ser105,t" generates just one

partition. Why?

2.7 Try analysing the following command sequence

"wt4,pl,ser4,m,4"

and then try running it first as a single command and then try to reproduce the result by issuing a sequence of three separate commands. The above sequence is actually relevant to a problem involving the symplectic nuclear model.

2.8 Just to remind ourselves that SCHUR handles lists of objects use SCHUR to calculate the following: "o21+3,21+4", "i21+3,21+4" and "pl21+3,21+4". Think about the second result.

5.3 Tutorial 2 : Exploring the REPmode

The **REP**mode is primarily for calculating the properties associated with a particular group. Only operations that stay within the set group are permitted. That, for example, excludes branching rules that result in a change of the group. Entry into the **REP**mode is made by issuing the command "rep" from either the SFNmode or the DPMode. The first action to be taken upon entering the **REP**mode is to check if a group has been set. If a group is already set, perhaps from a previous excursion into the **REP**mode, it will be obvious to the user because below the REP> will be the statement "The group is ..." with the group name being given. If the group is not set then you *must* set it as described under the command **GR**oup. For the purposes of this tutorial we shall set the group as SO_8 by issuing the command "gr so8".

```
DP>
rep
REP mode
REP>
gr so8
Group is SO(8)
REP>
-
```

At this stage try to repeat the input/output obtained in **Sample:**4 (Page 9). We can obtain information on the properties of an irreducible representation of SO_8 by using the command **prop** followed by one or more irreducible representations . Thus for the irreducible representation [21] we have:-

```
REP>
prop21
<dynkin label>(1100)
dimension=160   48*2nd-casimir=84
2nd-dynkin=60
REP>
-
```

Now evaluate the Kronecker square of the irreducible representation [21] and then determine its properties as follows:-

```
prod21,21
      [42] + [41^2 ] + [4] + [3^2 ] + 2[321] + [31^3 ]+
    + [31^3 ]- + 3[31] + [2^3 ] + [2^2 1^2 ]+ + [2^2 1^2 ]-
    + 2[2^2 ] + 3[21^2 ] + 2[2] + [1^4 ]+ + [1^4 ]-
    + 2[1^2 ] + [0]
REP>
prop last
<dynkin label>(2200) + (3011) + (4000) + (0300) + 2(1111)
            + (2020) + (2002) + 3(2100) + (0022) + (0120)
            + (0102) + 2(0200) + 3(1011) + 2(2000)
            + (0020) + (0002) + 2(0100) + (0000)
dimension=25600   2nd-dynkin=19200
REP>
-
```

We have two checks that indicate the calculation was correctly performed.

1. The dimensional check
$$Dim(\lambda \times \mu) = \sum_{\nu} c^{\nu}_{\lambda\mu} Dim(\nu)$$
which in the above case we have $160^2 = 25600$ as expected.
2. The 2nd-Dynkin index check that
$$\begin{aligned} I_2(\lambda \times \mu) &= I_2(\lambda) \times Dim(\mu) + Dim(\lambda) \times I_2(\mu) \\ &= \sum_{\nu} c^{\nu}_{\lambda\mu} I_2(\nu) \end{aligned}$$
which in the above case gives $160 \times 60 + 60 \times 160 = 19200$.

All the basic ideas found in the SFNmode tutorial of nesting commands carry over into the REPmode. Furthermore, the operations of the SFNmode may be used in the REPmode by making use of the command **conv_s**. Thus if the group had been set as S_{105} we could issue the command "conv_s wt-105ser105,t" followed by further commands to get the following

```
REP>
gr s105
Group is S(105)
REP>
conv_s wt-105ser105,t
      {14 13 12 11 10 987654321}
REP>
dim last
dimension=513782568580731957367019767803085320396631
          27760999759183808656854124180549926912001
REP>
dim conv_s wt-105ser105,t
dimension=513782568580731957367019767803085320396631
          27760999759183808656854124180549926912001
REP>
```

Just to get a feeling for the above result spend a few moments imagining you have a supercomputer that calculates the diagonal matrix elements of the representation matrix for a single group element at 10^9 elements/second. When will the computer finish its task? Would a 10^9 times faster computer help?

By now you should have gained sufficient experience to explore on your own the many commands associated with the REPmode mentioned in Appendix A. Let us try an exercise. We have set the group as U_3 and we want to determine the decomposition of the irreducible representation $\{31\}$ when the group is restricted to U_2. We already mentioned that branching rules are not part of the REPmode but let us see if we can get around that problem. We first set the group as U_3 and calculate the dimension of $\{31\}$ using the sequence:-

```
REP>
gr u3
Group is U(3)
REP>
dim 31
dimension=15
REP>
-
```

It follows from Eq. (2.56) that we need to evaluate $\{31/M\}$ where M is an

$S-$function series. Here we can use the command **conv_s**. Thus

```
REP>
conv_s sk31,m
      {31} + {3} + {21} + {2} + {1^2 } + {1}
REP>
dim last
dimension=45
REP>
-
```

In our enthusiasm we have gone ahead and computed the dimension to check the result and clearly got a wrong answer! We have not reset the group to U_2 which is now the relevant group. Let us fix that now:-

```
REP>
gr u2
Group is U(2)
REP>
dim last
dimension=15
REP>
rem That is correct!!!
REP>
-
```

Notice the use of the command **rem**. Let us repeat the exercise but this time considering the group-subgroup combination $SU_3 \supset SU_2$. Eq. (2.56) is still the relevant result.

```
REP>
gr su3
Group is SU(3)
REP>
dim 31
dimension=15
REP>
gr su2
Group is SU(2)
REP>
conv_s sk31,m
      {31} + {3} + {21} + {2} + {1^2 } + {1}
REP>
std last
      {3} + 2{2} + 2{1} + {0}
REP>
dim last
dimension=15
REP>
_
```

Notice the introduction of the command **std**. This has been introduced in SU_2 because standard irreducible representations of SU_n have $n-1$ or fewer parts so that the two part irreducible representations $\{31\}$, $\{21\}$ and $\{1^2\}$ must be modified to produce a list of standard SU_2 irreducible representations . It is usually good practice to follow the use of **conv_s** with the **std** command just to be on the safe side. The above two examples involved the issuing of several commands to finally yield the result for a single irreducible representation. Clearly the same set of commands could have handled a string of irreducible representations such as $\{31\} + \{2^2\} + \{21\}$ or we could have repeated the commands each time we wanted the branching for a different irreducible representation. Later we shall learn how to encaspulate the set of commands in a user defined function that can be called whenever needed.

Let us conclude this tutorial with some exercises.

3.1 The branching rule for the maximal embedding of Sp_{2k} in U_{2k} is given as $\{\lambda\} \downarrow \{\lambda/B\}$. Inversely, for $Sp_{2k} \uparrow U_{2k}$ we have $< \lambda >\uparrow \{\lambda/A\}$.

(a.) Use SCHUR to show that for $2k = 8$

$$< 21 >\uparrow \{21\} - \{1\} \quad \text{and} \quad \{21\} \downarrow < 21 > + < 1 >$$

(b.) Use the knowledge gained in (a.) to write a *single* sequence of commands to evaluate the Sp_8 Kronecker product $< 21 > \times < 21 >$. Check that your result is dimensionally correct and compare your result with that obtained by simply setting the group as Sp_8 and using the command "prod21,21".

3.2 Read the information in Appendix A about the command **au** and use the command to show that under the SO_8 automorphism $[1] \rightarrow [s;0]- \rightarrow [s;0]+$. Use that observation to obtain the content of the Kronecker products $[s;0] - \times [s;0]-$ and $[s;0] + \times [s;0]+$ from the result obtained from the command "prod1,1".

3.3 Set the group as Sp_8 and use SCHUR to calculate the Kronecker product $< 321 > \times < 421 >$. Now save the result by using the command "setr1 last" . Use the command "zero" to eliminate "last" and then issue the command "rv1" and you should once again see the output of the Kronecker product. Now save the "rvar" to a diskfile using the command "save rvar 'filename' " as described in Appendix A. If you now issue the command "setr1 zero" you will find on saying "rvar 1" the data for the Kronecker product has been lost and "rvar 1" simply reports zero. All is not lost as we can reload, even days later, the saved "rvar1" using the command **lo**ad as described in Appendix A. Thus issuing the command "load rvar'filename' " followed by the command "rv1" will put to screen once again the Kronecker product. Note however, we must set the group as appropriate to that of the saved rvar.

■ 5.4 Tutorial 3 : The Branching Rule Mode

The BRMode is called from the DPMode by the command **BRM**ode. It allows the user to pick one of SCHUR's predefined branching rules and evaluate it repeatedly for different irreducible representations or strings of irreducible representations without having to reset the groups or the rule. However, the results of a calculation, unlike in all other modes of SCHUR, cannot be further processed. SCHUR considers 55 different group-subgroup structures as shown in Table A.2 of Appendix A. In each case the embedding chosen is defined by that of the vector irreducible representation of the group. The user may scroll through the contents of Table A.2 in an active session using the helpfile command **?tab**. A simple example of the use of the BRMode has already been given in Appendix A which the beginning user should now consult and repeat. Let us begin by switching to the BRMode to give:-

```
DP>
brm
Branch Mode
enter branching & rule numbers>
```

Our response is to decide on the rule we wish to invoke and then enter the rule number and the group numbers. Thus to evaluate decompositions for $SO_9 \rightarrow SO_4 \times SO_5$ the rule number is 27 and the group numbers would be 4 and 5. This information is entered as follows:-

```
27,4,5
SO(9) to SO(4) * SO(5)
BRM>
_
```

Note the commas that have been used to separate the numbers. Spaces could have been equally well used. To obtain branching rules we now simply enter irreducible representations or strings of irreducible representations of the group SO_9. For example:-

```
BRM>
1
      [1][0] + [0][1]
BRM>
21
      [21]+[0] + [21]-[0] + [2][1] + [1^2 ]+[1]
       + [1^2 ]-[1] + [1][2] + [1][1^2 ] + [1][0]
       + [0][21] + [0][1]
BRM>
21+1
      [21]+[0] + [21]-[0] + [2][1] + [1^2 ]+[1]
       + [1^2 ]-[1] + [1][2] + [1][1^2 ] + 2[1][0]
       + [0][21] + 2[0][1]
BRM>
s2
      [s;2]+[s;0] + [s;2]-[s;0] + [s;1]+[s;1] + [s;1]+[s;0]
    + [s;1]-[s;1] + [s;1]-[s;0] + [s;0]+[s;2] + [s;0]+[s;1]
    + [s;0]+[s;0] + [s;0]-[s;2] + [s;0]-[s;1] + [s;0]-[s;0]
BRM>
-
```

Note the appearance of the auxilliary $\pm$ labels associated with two-part tensor irreducible representations and all the spin irreducible representations of the subgroup SO_4. Also note the example of using several irreducible representations at once. There is nothing to stop you entering multiple copies of irreducible representations as for example "2.21 + 5.321 + s21". Let us now change the rule to obtain some decompositions for the super Lie group chain

$$SU_{7/11} \rightarrow U_1 \times SU_{4/5} \times SU_{3/6}$$

In this case the relevant rule number is 33 and the group numbers 4,3,5,6 . But first we must issue the command "stop" to change the rule leading to:-

```
BRM>
stop
enter branching & rule numbers>
33,4,5,3,6
SU(7/11) to U(1) * SU(4/5) * SU(3/6)
BRM>
21
      {3}{0}{21} + {-1}{1}{2} + {-1}{1}{1^2 }
       + {-5}{2}{1} + {-5}{1^2 }{1} + {-9}{21}{0}
BRM>
-
```

In the case of rules 40 to 55 the group-subgroup structures are fixed and it suffices to enter just the rule number. Thus we have, for example,

```
BRM>
stop
enter branching & rule numbers>
52
E(8) to SU(3) * E(6)
BRM>
21
      {2^2 }(1:1^5 ) + {21}(2:0) + {21}(0:0) + {2}(1:1)
       + {1^2 }(2:1^4 ) + {1^2 }(1:1) + {1}(2:1^2 )
       + {1}(1:1^5 ) + {0}(2:21^4 ) + {0}(0:0)
BRM>
exit
DPrep Mode (with function)
DP>
-
```

Issuing the command “exit” takes us back to the DPMode the subject of our next tutorial. In this tutorial you should have not only gained some experience in using the BRMode but also noted the frequent occurrence of direct products of groups and that in invoking branching rules the set groups change.

■ 5.5 Tutorial 4. : Introduction to the DPMode

In many respects the DPMode is the most important of all the modes of SCHUR. The data created in all the other modes can be accessed in the DPMode and much more. This is the mode for handling all those situations where more than one group is involved and where certain operations lead to a change in the set groups. It is also the mode in which the user can define his or her own functions and use the important command **rule**. These two uses are discussed in the two advanced tutorials given in Chapter Six.

As with the REPmode it is usually necessary to first set the groups. Note the use of the plural as here we may set products of up to eight groups. The setting of the groups follows use of the command **gr**oups described in Appendix A and following the formats specified in Table A.3. The entry of the direct product irreducible representations, commonly herein referred to as DPreps, follows that of **Sample:** 5 on page 9. It is important to note that most operations in the DPMode act on a particular group and it is necessary to indicate the number of the group being acted upon. The action of some operations will be to change the group to another group, or even groups, and as a result the order of the groups may change. After any group changing operation SCHUR will indicate the new setting of the groups. Where a product of several groups has been set the user needs to be aware that often seemingly simple operations on relatively modest irreducible representations may generate a very large output possibly exhausting the available heap space result in a runtime error being produced.

To illustrate some of the preceding matters let us start in the DPMode by setting the groups as $SU_3 \times Sp_4 \times SO_6$.

```
DP>
gr3su3sp4so6
Groups are   SU(3) * Sp(4) * SO(6)
DP>
-
```

Now work through the following sequences of commands:-

```
DP>
[1*1*1]
      {1}<1>[1]
DP>
dim last
Dimension = 72
DP>
prod last,last
     {2}<2>[2] + {2}<2>[1^2 ] + {2}<2>[0] + {2}<1^2 >[2]
     + {2}<1^2 >[1^2 ] + {2}<1^2 >[0] + {2}<0>[2]
     + {2}<0>[1^2 ] + {2}<0>[0] + {1^2 }<2>[2]
     + {1^2 }<2>[1^2 ] + {1^2 }<2>[0] + {1^2 }<1^2 >[2]
     + {1^2 }<1^2 >[1^2 ] + {1^2 }<1^2 >[0] + {1^2 }<0>[2]
     + {1^2 }<0>[1^2 ] + {1^2 }<0>[0]
DP>
dim last
Dimension = 5184
```

The group Sp_4 is locally isomorphic to SO_5 and hence we should be able to use the command **au** to replace the Sp_4 irreducible representations by the corresponding SO_5 irreducible representations by issuing the command "au gr2,so5" to produce:-

```
DP>
au gr2,so5,last
Groups are   SU(3) * SO(5) * SO(6)
   {2}[1^2 ][2] + {2}[1^2 ][1^2 ] + {2}[1^2 ][0]
   + {2}[1][2] + {2}[1][1^2 ] + {2}[1][0] + {2}[0][2]
   + {2}[0][1^2 ] + {2}[0][0] + {1^2 }[1^2 ][2]
   + {1^2 }[1^2 ][1^2 ] + {1^2 }[1^2 ][0] + {1^2 }[1][2]
   + {1^2 }[1][1^2 ] + {1^2 }[1][0] + {1^2 }[0][2]
   + {1^2 }[0][1^2 ] + {1^2 }[0][0]
DP>
dim last
Dimension = 5184
```

Note that the groups have been changed but as expected the dimension of the

result is unchanged. Let us now branch the group $SO_6 \rightarrow U_1 \times SU_3$ noting Table A.2 of Appendix A. We obtain:-

```
DP>
br26,6gr3last
Groups are   SU(3) * SO(5) * U(1) * SU(3)
      {2}[1^2 ]{4}{2} + {2}[1^2 ]{4}{1^2 }
       + 2{2}[1^2 ]{0}{21} + 2{2}[1^2 ]{0}{0}
       + {2}[1^2 ]{-4}{2^2 } + {2}[1^2 ]{-4}{1}
       + {2}[1]{4}{2} + {2}[1]{4}{1^2 }
       + 2{2}[1]{0}{21} + 2{2}[1]{0}{0}
       + {2}[1]{-4}{2^2 } + {2}[1]{-4}{1}
       + {2}[0]{4}{2} + {2}[0]{4}{1^2 }
       + 2{2}[0]{0}{21} + 2{2}[0]{0}{0}
       + {2}[0]{-4}{2^2 } + {2}[0]{-4}{1}
       + {1^2 }[1^2 ]{4}{2} + {1^2 }[1^2 ]{4}{1^2 }
       + 2{1^2 }[1^2 ]{0}{21} + 2{1^2 }[1^2 ]{0}{0}
       + {1^2 }[1^2 ]{-4}{2^2 } + {1^2 }[1^2 ]{-4}{1}
       + {1^2 }[1]{4}{2} + {1^2 }[1]{4}{1^2 }
       + 2{1^2 }[1]{0}{21} + 2{1^2 }[1]{0}{0}
       + {1^2 }[1]{-4}{2^2 } + {1^2 }[1]{-4}{1}
       + {1^2 }[0]{4}{2} + {1^2 }[0]{4}{1^2 }
       + 2{1^2 }[0]{0}{21} + 2{1^2 }[0]{0}{0}
       + {1^2 }[0]{-4}{2^2 } + {1^2 }[0]{-4}{1}
DP>
dim last
Dimension = 5184
```

Let us now use the command **Cont**ractGroups, described in Appendix A, to combine the two SU_3 groups under the Kronecker product as follows:-

```
DP>
cont1,4,last
Groups are   SU(3) * SO(5) * U(1)
      {42}[1^2 ]{-4} + {42}[1]{-4} + {42}[0]{-4}
       + 2{41}[1^2 ]{0} + 2{41}[1]{0} + 2{41}[0]{0}
       + {4}[1^2 ]{4} + {4}[1]{4} + {4}[0]{4}
       + {3^2 }[1^2 ]{-4} + {3^2 }[1]{-4} + {3^2 }[0]{-4}
       + 4{32}[1^2 ]{0} + 4{32}[1]{0} + 4{32}[0]{0}
       + 3{31}[1^2 ]{4} + 3{31}[1]{4} + 3{31}[0]{4}
       + {3}[1^2 ]{-4} + {3}[1]{-4} + {3}[0]{-4}
       + 2{2^2 }[1^2 ]{4} + 2{2^2 }[1]{4} + 2{2^2 }[0]{4}
       + 4{21}[1^2 ]{-4} + 4{21}[1]{-4} + 4{21}[0]{-4}
       + 6{2}[1^2 ]{0} + 6{2}[1]{0} + 6{2}[0]{0}
       + 6{1^2 }[1^2 ]{0} + 6{1^2 }[1]{0} + 6{1^2 }[0]{0}
       + 3{1}[1^2 ]{4} + 3{1}[1]{4} + 3{1}[0]{4}
       + 2{0}[1^2 ]{-4} + 2{0}[1]{-4} + 2{0}[0]{-4}
```

Finally let us remove the group U_1 by use of the command **rm_g**roup :-

```
DP>
rm_g3,last
Groups are   SU(3) * SO(5)
      {42}[1^2 ] + {42}[1] + {42}[0] + 2{41}[1^2 ]
       + 2{41}[1] + 2{41}[0] + {4}[1^2 ] + {4}[1] + {4}[0]
       + {3^2 }[1^2 ] + {3^2 }[1] + {3^2 }[0] + 4{32}[1^2 ]
       + 4{32}[1] + 4{32}[0] + 3{31}[1^2 ] + 3{31}[1]
       + 3{31}[0] + {3}[1^2 ] + {3}[1] + {3}[0]
       + 2{2^2 }[1^2 ] + 2{2^2 }[1] + 2{2^2 }[0]
       + 4{21}[1^2 ] + 4{21}[1] + 4{21}[0] + 6{2}[1^2 ]
       + 6{2}[1]  + 6{2}[0] + 6{1^2 }[1^2 ] + 6{1^2 }[1]
       + 6{1^2 }[0] + 3{1}[1^2 ] + 3{1}[1] + 3{1}[0]
       + 2{0}[1^2 ] + 2{0}[1] + 2{0}[0]
DP>
dim last
Dimension = 5184
```

Note our frequent use of the dimensional check. It is a good habit. It helps avoid mistakes of input etc, or more likely, to indicate mistakes. The preceding examples illustrate some of the rich features of the DPMode. Note that the REPmode command **conv_s** may be also used in the DPMode by the simple expedience of enclosing the expression in square brackets "[,]". In a similar fashion rvars can be imported into the DPMode. This feature can often be of use in writing functions as discussed in the next chapter.

5.6 Exercises

The following exercises illustrate some problems you should now be able to tackle using SCHUR.

1. The states of the d^n electron configuration may be classified by the following chain of groups leading, finally, to the spectroscopic terms ^{2S+1}L. Use SCHUR to establish the results of Table 5.1 for the spectroscopic terms of the d^5 configuration

$$U_{10} \supset Sp_{10} \supset SU_2 \times SO_5 \supset SU_2 \times SO_3 \sim SO_3^S \times SO_3^L \equiv^{2S+1} L$$

Table 5.1 Spectroscopic terms of the d^5 electron configuration.

Dim	SU_{10}	Sp_{10}	SU_2	×	SO_5	SU_2	×	SO_3	SO_3^S	×	SO_3^L	^{2S+1}L
252	$\{1^5\}$	$<1^5>$	$\{5\}$	×	$[0]$	$\{5\}$	×	$[0]$	$[\Delta;2]$	×	$[0]$	^6S
			$\{3\}$	×	$[2]$	$\{3\}$	×	$[4]$	$[\Delta;1]$	×	$[4]$	^4G
								$[2]$			$[2]$	^4D
			$\{1\}$	×	$[2^2]$	$\{1\}$	×	$[6]$	$[\Delta;0]$	×	$[6]$	^2I
								$[4]$			$[4]$	^2G
								$[3]$			$[3]$	^2F
								$[2]$			$[2]$	^2D
								$[0]$			$[0]$	^2S
		$<1^3>$	$\{3\}$	×	$[1^2]$	$\{3\}$	×	$[3]$	$[\Delta;1]$	×	$[3]$	^4F
								$[1]$			$[1]$	^4P
			$\{1\}$	×	$[21]$	$\{1\}$	×	$[5]$	$[\Delta;0]$	×	$[5]$	^2H
								$[4]$			$[4]$	^2G
								$[3]$			$[3]$	^2F
								$[2]$			$[2]$	^2D
								$[1]$			$[1]$	^2P
		$<1>$	$\{1\}$	×	$[1]$	$\{1\}$	×	$[2]$	$[\Delta;0]$	×	$[2]$	^2D

2. If two sets of states of a d^n configuration transform under SO_5 as the irreducible representations $[\lambda]$ and $[\mu]$ respectively their matrix elements of an operator transforming as $[\nu]$ will certainly vanish unless

$$[\lambda] \times [\mu] \supset [\nu]$$

The Coulomb interaction within the d^n configuration can be expanded

in terms of operators symmetrised with respect to the same groups used to classify the states. One of the relevant operators transforms as $[2^2]$ under SO_5. Let $c([\lambda][\mu][2^2])$ be the number of times $[2^2]$ occurs in the SO_5 Kronecker product $[\lambda] \times [\mu]$. Use SCHUR to construct the entries given in Table 5.2 for the numbers $c([\lambda][\mu][2^2])$.

Table 5.2 The numbers $c([\lambda][\mu][2^2])$ for irreducible representations of SO_5 appropriate to the states of the d^n configurations.

	$[0]$	$[1]$	$[1^2]$	$[2]$	$[21]$	$[2^2]$
$[0]$	–	–	–	–	–	1
$[1]$	–	–	–	–	1	1
$[1^2]$	–	–	1	–	1	1
$[2]$	–	–	–	1	1	1
$[21]$	–	1	1	1	2	1
$[2^2]$	1	1	1	1	1	1

3. Show that for SO_5 the irreducible representation $[2^2]$ occurs once in the symmetric part of the Kronecker square of the irreducible representation $[2^2]$ and once in the antisymmetric part.

5. In the simple SU_3 quark model of baryons and mesons the (u, d, s) quarks span the three-dimensional SU_3 irreducible representation $\{1\}$ while the corresponding antiquarks $(\overline{u}, \overline{d}, \overline{s})$ span the three-dimensional SU_3 irreducible representation $\{1^2\}$. The SU_3 group is sometimes referred to as the *flavour* group SU_3^{fl}.

   a. Show that combining quarks with antiquarks leads to an octet and singlet of mesons.

   b. Show that combining a triple product of quarks leads to a baryon singlet and two baryon octets.

5. A student tries to unite the lowest lying baryons into a single irreducible representation of a Lie group G. Noting that the baryon octet has spin $J^p = \frac{1}{2}^+$ and the decuplet spin $J^p = \frac{3}{2}^+$ identify G and the relevant irreducible representation she found.

6. Assume that you have the group SU_3^{fl}, as in exercise 5. Your quarks are now also endowed with spin (SU_2^s) and *color* (SU_3^c). The total of 18 single quark states span the vector irreducible representation $\{1\}$ of SU_{18} and you have the chain of groups

$$SU_{18} \supset SU_3^{fl} \times (SU_6^{cs} \supset SU_2^s \times SU_3^c)$$

   Assume that each quark is in an $S-$state and that the states of the six quark configuration q^6, span the totally antisymmetric irreducible representation $\{1^6\}$ of SU_{18}. In terms of QCD (Quantum ChromoDynamics) physical states correspond to color singlets i.e. states transforming as

$\{0\}$ under SU_3^c. With respect to the color-spin group SU_6^{cs} this can only happen if the weight w_λ of the partition λ labelling the irreducible representation $\{\lambda\}$ of SU_6^{cs} is divisible by three. That is for irreducible representations of null triality.

a. Determine the irreducible representations of $SU_3^{fl} \times SU_6^{cs}$ contained in $\{1^6\}$ of SU_{18} that have null triality.

b. Determine the spin content of each SU_6^{cs} found in (a).

c. Which of the multi-quark configurations q^7, q^8 and q^9 might you expect to contain color singlet states?

d. In the MIT bag model of multi-quark states there is an energy term proportional to the eigenvalues of the second-order Casimir operator of the color-spin group. Show that the eigenvalues for the irreducible representations $\{21^4\}$, $\{2^3\}$ and $\{3^2\}$ are in the ratio 1:2:3.

7. Use the command **p_to_s** to construct the character table of the symmetric group $\mathcal{S}_4$.
8. Show that the command sequence

$$\textbf{comp}\text{are } \lambda, \textbf{p\_to\_s } \rho \quad \lambda, \rho \vdash n$$

brings to the screen the value of the characteristic χ_ρ^λ of $\mathcal{S}_n$.

9. A person wishes to be able to form the product of two $S-$functions λ and μ using the Littlewood-Richardson rule retaining only terms whose first part is $\leq n$. Show that the command sequence

$$\textbf{conj len n conj o } \lambda, \mu$$

will achieve the desired result.

10. Repeat (9.) for the combination of $Q-$functions.

> — *and many a man lives a long life through, thinking he believes certain universally received and well established things and yet never suspects that if he were confronted by those things once, he would discover he did not really believe them before, but only thought he believed them*
> — Mark Twain, *Roughing it, 1872*

6

Advanced Tutorials in using SCHUR

Introduction

It is the purpose of this chapter to introduce the user to the task of writing user defined functions and using the command **Rule**. These two tutorials are at a more advanced level and assume that the user has already worked through the earlier tutorials gaining some familiarity with most of the available commands. A number of technical points will arise which assume the user is familiar with much of the material in chapters Two and Three. In the first tutorial the writing of relatively simple user defined functions is considered. The second tutorial is at a still more advanced level leading into research applications. For some of these applications the user may wish to consult the reading list in Chapter Eight.

6.1 Advanced Tutorial 1 : Writing User Defined Functions

The objective of writing user defined functions is to be able to create sequences of commands that can be encaspulated into a single function which maybe repeatedly called upon to carry out the same sequences of commands. In the early stages the user will want the function to write to screen every line of output and make frequent dimensional checks until fully convinced that the function fulfils its tasks correctly. At that stage the output can be restricted to final results and some of the checking procedures removed.

While it is entirely possible to write functions using the SCHUR command **SetF**nVar it is usually preferable to use a text editor to write the function line-by-line and then read it into SCHUR using the command **ReadF**nFromDisk.

As a first exercise let us write a simple function that reads in two lists of $S-$functions forms their outer product, writes out the number distinct partitions and the sum of the multiplicities as well as the resultant list of $S-$functions.

Our objective can be attained by writing the following function:-

```
rem This forms outer products of S-functions
sfn
sup false
enter sv1
enter sv2
o sv1,sv2
countt last
countc last
stop
```

We shall assume you have saved this function to your SCHUR directory with the filename 'test.fn'. This function deserves a line-by-line comment. The first line is

a remark that will not appear in the output. The second line "sfn" puts SCHUR into the SFNmode. The command "sup false" ensures that every line of output will go to the screen. The next two lines will lead to SCHUR instructing the user to enter two lists of $S-$functions. The next line "o sv1,sv2" combines the two lists using the Littlewood-Richardson rule. The next line counts the number of distinct terms or partitions while the pentultimate line reports the sum of the coefficients or multiplicities. The final line "stop" (with a <CR>) marks the end of the function. The function is loaded in the DPMode by issuing the command "readf1'test.fn' ". Notice the single quotes about the filename. Failure to include them will result in an error message. We have designated that this function be loaded as function 1. Let us now see the function in use:-

```
DP>
readf1'test.fn'
=-
=-
=-
=-
=-
=-
=-
=-
=-
DP>
wrfn1
rem This forms outer products of S-functions
sfn
sup false
enter sv1
enter sv2
o sv1,sv2
countt last
countm last
DP>
-
```

Having read the function in we can inspect it in the DPMode at anytime by issuing the command "wrfn1". To run the function we simply enter the command "fn1" and then respond to SCHUR's requests as shown on the next page.

```
DP>
fn1
Schur Function Mode
enter sv1
21
enter sv2
31
   {52} + {51^2 } + {43} + 2{421} + {41^3 } + {3^2 1}
 + {32^2 } + {321^2 }
TermSum  = 8
CoeffSum = 9
SFN>
-
```

Note that after finishing its task SCHUR has remained in the SFNmode. We could have made SCHUR return to the DPMode by putting the command "exit" as the pentultimate line in the function. However if we wish to use the function repeatedly we may as well stay in the SFNmode as the function can be accessed directly within the SFNmode as seen in the following:-

```
SFN>
fn1
enter sv1
2+1
enter sv2
2+3
  {5} + {41} + 2{4} + {32} + 2{31} + {3} + {2^2 } + {21}
TermSum  = 8
CoeffSum = 10
SFN>
-
```

As a second exercise suppose we wished to construct a table of $O_6 \rightarrow S_6$ branching rules, obtaining the dimension of the inputted O_6 irreducible representation and with the output in TEX. We could construct a function 'o6s6.fn' of the form:-

```
gr o6
sb_tex true
enter rv1
sup false
dim[rv1]
br18,6gr1[rv1]
dim last
sb_tex false
stop
```

Note that before ending the function we restore sb_tex to its default setting. The entry of the irreducible representation has been given appropriate to the REPmode even though all the action is taking place in the DPMode. The rvar is encased in square brackets. If we load this new function as fn1 it will overwrite the earlier fn1 otherwise we may load it as say fn2 and we may then access either fn1 or fn2. We shall choose to load it as fn2 and run it as follows:-

```
DP>
readf2'o6s6.fn'
=-
=-
=-
=-
=-
=-
=-
=-
DP>
fn2
Group is O(6)
enter rv1
21
Dimension = 64
Group is S(6)
      2\{51\} + 2\{42\} + 2\{41^2 \} + \{321\}
Dimension = 64
DP>
fn2
Group is O(6)
enter rv1
s21
Dimension = 280
Group is S(6)
  2\{\Delta;21\}_+ + 2\{\Delta;21\}_- + 8\{\Delta;2\}
  + 6\{\Delta;1\} + \{\Delta;0\}_+ + \{\Delta;0\}_-
Dimension = 280
```

As a third example we consider the use of the reduced notation introduced on page 25 for giving an $n-$independent treatment of the $O_n \downarrow S_n$ branching rules for tensor type representations $[\lambda]$ of O_n. Salam and Wybourne (*J. Phys. A:Math. Gen.* **22** ,3771 (1989)) have shown that the branching rule

can be succintly written as

$$[\lambda] \downarrow < 1 > \otimes \{\lambda/G\} \tag{6.1}$$

where we note the skew operation with the $S-$function series G described on page 29 and the use of the inner plethysm operation described on page 27. We propose to write a function acting in the SFNmode that inputs a λ as an svar and outputs a list of S_n irreducible representations in reduced notation. This can be accomplished by the following function:-

```
sfn
enter sv1
i_pl sk sv1,g
stop
```

Notice how the two operations 'sk' and 'i_pl' have been combined into a one line statement. Let us run the function for $\lambda = 321$.

```
DP>
fn1
Schur Function Mode
enter sv1
321
      <41> + 2<4> + <321> + 3<32> + 3<31^2 > + 9<31> + 6<3>
       + 3<2^2 1> + 7<2^2 > + <21^3 > + 9<21^2 > + 15<21>
       + 6<2> + 2<1^4 > + 6<1^3 > + 6<1^2 > + 2<1>
SFN>
```

For an application of this function to the nuclear symplectic shell model see:-

B. G. Wybourne, *The representation space of the nuclear symplectic* $Sp(6,R)$ *shell model*, J. Phys. A: Math. Gen. **25** 4389-98 (1992).

The above function is quite general, making no reference to a particular value of n. Had we wanted results for a specific value of n we could have followed the function with the command "**Make**WtOfSfnToN" indicating the value of the integer n.

As the fourth example we consider the writing of a function to evaluate signed sequences for the groups O_N and Sp_{2k}. In practical applications it is sometimes necessary to determine what sequence of non-standard labels will, upon application of the appropriate modification rules yield a given standard $[\lambda]$ or $[\lambda]^*$ of O_N or $< \lambda >$ of Sp_{2k}. (See, for example, D. J. Rowe, B. G. Wybourne and P. H. Butler, *Unitary representations, branching rules and matrix elements*

for the non-compact symplectic groups J. Phys. A: Math. Gen. **18** 939–953 (1985).) For the group O_{2k} the modification rules can be written in the form

$$[\lambda_1\lambda_2\ldots\lambda_k(C^o)] = [\lambda_1\lambda_2\ldots\lambda_k]^* \tag{6.2a}$$

$$[\lambda_1\lambda_2\ldots\lambda_k(C^e)] = [\lambda_1\lambda_2\ldots\lambda_k] \tag{6.2b}$$

with all other partitions having more than k parts vanishing. The series C^o and C^e are restricted to members of the C series with C^o containing only partitions of *odd* Frobenius rank and C^e of *even* rank.

The corresponding rules for O_{2k+1} are

$$[\lambda_1\lambda_2\ldots\lambda_k(G^o)] = [\lambda_1\lambda_2\ldots\lambda_k]^* \tag{6.3a}$$

$$[\lambda_1\lambda_2\ldots\lambda_k(G^e)] = [\lambda_1\lambda_2\ldots\lambda_k] \tag{6.3b}$$

with all other partitions having more than k parts vanishing. The series G^o and G^e are restricted to members of the G series parts with G^o containing only partitions of *odd* weight and G^e of *even* weight.

For the symplectic group Sp_{2k} the rule is

$$< \lambda_1\lambda_2\ldots\lambda_k(A) >=< \lambda_1\lambda_2\ldots\lambda_k > \tag{6.4}$$

with all other partitions having more than k parts vanishing and here the A series is used. This last case is the simplest and exposes most of the essential features that will arise in writing functions for the other signed sequences.

It is crucially important to note that in both Eq.(6.3) and (6.4) the partitions (λ) may have fewer than k vanishing parts and hence may involve the trailing zeroes referred to in Chapter One.

Since we are dealing with a single group that will stay constant we should work in the REPmode. This will be done by making the first line of our function the command 'rep'. The second line must set the group which we will here choose as 'gr sp10'. For reasons that will shortly become apparent, we will enter the standard tensor irreducible representation as an svar and hence will write the third line as 'enter svar 1'. So as to initially see every line of output let us make the fourth line 'sup false'. That completes the preamble.

The next step is to generate the terms in the appropriate series up to a prescribed cutoff in the weight and length of the partitions. This maybe done by use of the two commands '**Len**gth' and '**Ser**iesToIntWt' . For example we might use the command sequence 'len10 ser20,a' which would generate all the terms of the $A-$series up to weight 20 and length 10. These must then have the partition (λ) **INS**erted in front of them as in Eq. (6.4). If $\ell_\lambda < k$ then there will be trailing zeroes. This problem can be overcome if before issuing the **INS**ert command we **AT**tach the partition (1^k) to 'svar1'. This will ensure there are no trailing zeros and later we can remove the partition (1^k) using the command **rm_P**artitionFromSfn. Thus we have for Sp_{10} the command sequence

ins at svar1 1^5 len10ser20,a

We can now safely remove the partition 1^5 which will result in a list of non-standard $S-$functions which must be made standard by use of the command **std** resulting in a list of now standard $S-$functions which must be converted into irreducible representations using the **conv_s** command. Thus we could accomplish all of this paragraph by issuing the single command sequence

conv_s std rm_p1^5 ins at svar1 1^5 len10 ser 20,a

We can check the result by applying the modification rules to the output by again using the command **std** and we should obtain the inputted irreducible representation with a multiplicity equal to the number of terms in the outputted signed sequence.

Thus the final function could be written as:-

```
rep
gr sp10
enter sv1
sup false
conv_s std rm_p1^5 ins at svar1 1^5 len10 ser20,a
std last
stop
```

Running this function leads to the output:-

```
DP>
fn1
REP mode
Group is Sp(10)
enter sv1
1
   <421^13 > - <321^12 > + <2^2 1^11 > - <1^11 > + <1>
   5<1>
REP>
```

The above example illustrates the way in which quite complex sets of commands can be written as a single sequence. By now the user should be able to write simple functions with some facility and should be able to write functions for each of Eq. (6.2a) - (6.3b). (You will need to use commands such as **Rm_EvenR**kSfnsOnly.). See also **SIGNSEQ**uence.

■ 6.2 Advanced Tutorial 2 : Using the Rule Command

So far all the functions we have written involved the setting of, at most, a single

group. There are many cases where we might wish to consider direct product groups or to evaluate quite complex expressions say for example

$$[\lambda] \rightarrow \sum_{\zeta} < \{\{\lambda/C\} \circ \zeta\}/B > \times < \zeta/B > \tag{6.5}$$

which is the $O_{4mn} \rightarrow Sp_{2m} \times Sp_{2n}$ branching rule for tensor irreducible representations. As we shall see shortly such expressions can be evaluated readily using the **Rule** command. To start with using **Rule** let us consider the formula for evaluating the Kronecker product of two irreducible representations of Sp_{2k}

$$< \lambda > \times < \mu >= \sum_{\zeta} < \lambda/\zeta \cdot \mu/\zeta > \tag{6.6}$$

where the $\cdot$ signifies outer $S-$function multiplication and we have to sum over all $S-$functions ζ that will simultaneously skew with both λ and μ. We can anticipate that the outer $S-$function multiplication will be accomplished by the command **Cont**ract which will change the number of groups set. This suggests we should start by setting two identical symplectic groups. A possible function could read:-

```
enter group sp(n)*sp(n)
enter rv1
enter rv2
dim[rv1*rv2]
sup false
rule[rv1*rv2]sum sk1sk2
cont1,2 o last
std last
dim last
stop
```

Study each line carefully and see if you can determine its action. Note that the function runs in the DPmode and λ and μ are entered as rv1 and rv2 and then the dimension of their product calculated for checking purposes. The summation with respect to ζ is done using **Rule** with the 1 and 2 referring to the irreducible representations of groups 1 and 2. **Cont**ract is used after **Rule** to evaluate the outer $S-$function products. At that stage we cannot gaurantee that all of the resulting partitions will correspond to standard labelled irreducible representations of Sp_{2k} and hence we should apply **std** to the output. Finally we should make a dimensional check of the result. Now run the function:-

```
fn1
enter group sp(n)*sp(n)
2sp6sp6
Groups are   Sp(6) * Sp(6)
enter rv1
21
enter rv2
31
Dimension = 12096
 <21><31> + <2><3> + <2><21> + <1^2 ><3> + <1^2 ><21>
  + 2<1><2> + <1><1^2 > + <0><1>
Group is Sp(6)
  <52> + <51^2 > + <5> + <43> + 2<421> + <41^3 > + 3<41>
 + <3^2 1> + <32^2 > + <321^2 > + 3<32> + 3<31^2 > + 2<3>
 + 2<2^2 1> + <21^3 > + 3<21> + <1^3 > + <1>
  <52> + <51^2 > + <5> + <43> + 2<421> + 3<41> + <3^2 1>
 + <32^2 > + 3<32> + 3<31^2 > + 2<3> + 2<2^2 1> + 3<21>
 + <1^3 > + <1>
Dimension = 12096
DP>
```

Notice that in replying to the request to 'enter group sp(n)*sp(n)' the reply includes the number of groups but *not* the prefix **gr**oup. The first lot of output has given the result of the skew operations while the second lot of output has come from the contraction with the consequential reduction in the number of groups set. The action of the **std** command has modified the non-standard irreducible representations involving partitions into more than three parts.

Let us return to the evaluation of Eq. (6.5). To begin we need to set the initial group O_{4mn} and then enter the desired irreducible representation as an rvar. We should then compute its dimension for checking purposes. We are now finished with the group O_{4mn} and must change the group setting to $Sp_{2m} \times Sp_{2n}$. We now have set the direct product of two groups and our starting irreducible representation will be of the form $[rv1 * 0]$. The first task is to skew rv1 with the $C-$series. This can be done by the Rule command 'rule [rv1*0] sk1 with c'. This gives a number of irreducible representations of $Sp_{2m} \times Sp_{2n}$ which may be referred to as 'last'. We need to sum over all $S-$functions ζ that can form an inner product with the terms in $\{\lambda/C\}$ and at the same time associate the same value of ζ with the second group, Sp_{2n}. This may be accomplished by the Rule

command 'rule last sum i1 eq2'. At this stage we have calculated the terms in

$$\sum_{\zeta} < \{\{\lambda/C\} \circ \zeta\} > \times < \zeta > \tag{6.7}$$

It remains now only to skew the irreducible representations of each group separately with the compatible members of the $B-$series and then to modify the resulting list of irreducible representations of $Sp_{2m} \times Sp_{2n}$ and make a dimensional check. All the above can be encaspulated in the function given below:-

```
enter group o4mn
enter rv1
dim[rv1]
enter group sp2m*sp2n
rule[rv1*0]sk1with c
rule last sum i1eq2
rule last sk1with b
rule last sk2with b
sup false
std last
dim last
stop
```

Notice where we have inserted the 'sup false' command. We have agreed to suppress all output bar that of the last two commands. Had we omitted it the only output would be that of the last commmand. Had we placed 'sup false' after the third line all the intermediate steps of the calculation would have been sent to the screen. Let us now run the function:-

```
fn1
enter group o4mn
o32
Group is O(32)
enter rv1
21
Dimension = 10880
enter group sp2m*sp2n
2sp8sp4
Groups are   Sp(8) * Sp(4)
    <3><21> + <3><1> + <21><3> + <21><21> + 2<21><1>
    + <1^3 ><21> + <1^3 ><1> + <1><3> + 2<1><21> + 2<1><1>
Dimension = 10880
DP>
```

Notice the function was written so that we could choose the groups O_{4mn} and $Sp_{2m} * Sp_{2n}$. Had we wished to obtain results for many different irreducible representations for say $O_{32} \rightarrow Sp_8 \times Sp_4$ then we might have written the function as:-

```
group o32
enter rv1
dim[rv1]
group2sp8sp4
rule[rv1*0]sk1with c
rule last sum i1eq2
rule last sk1with b
rule last sk2with b
sup false
std last
dim last
stop
```

As a further example of writing functions, this time using both **Rule** and **Cont**ract, let us consider writing a function to evaluate the branching rule $SO_7 \downarrow SU_2 \times SU_2 \times SU_2$ first for tensor irreducible representations $[\lambda]$ and then for spin irreducible representations $[s;\lambda]$. We first note that the vector

irreducible representation [1] decomposes as

$$[1] \downarrow \{1\}\{1\}\{0\} + \{0\}\{0\}\{2\}$$

An arbitrary tensor irreducible representation $[\lambda]$ of SO_7 decomposes as

$$[\lambda] \downarrow \sum_{\rho,\tau} \{\lambda/C\rho\} \circ \{\tau\} \cdot \{\tau\} \cdot \{2\} \otimes \{\rho\} \tag{6.8}$$

The above result follows from using Eqs. (2.34) to evaluate the plethysm

$$(\{1\}\{1\}\{0\} + \{0\}\{0\}\{2\}) \otimes [\lambda]$$

and making the observation that under $SO_7 \uparrow SU_7$

$$[\lambda] \uparrow \{\lambda/C\}$$

We note from Eq. (6.8) that the operations skew, inner and plethysm must be used as well as the modification rules for SU_2. The following function may be constructed to do the job:-

```
gr so7
enter rv1
dim[rv1]
gr4su2su2su2su2
rule[rv1*0*2*0]sk1 with C
rule last sum sk1 eq4
cont3,4 pl last
std_o gr3,last
rule last sum i1eq2
std last
sup false
last
dim last
stop
```

This function deserves close attention as it illustrates a number of new features. First note that 4 SU_2 groups have been set, this is so that we can perform the necessary plethysm and then contract the last two SU_2 groups to a single SU_2 group. Second note the use of the operation **Eq**ual and **SUM** being used together. Third note that after performing the contraction the resulting SU_2 irreducible representations have been modified using **std_o**. This has been done to reduce the number of intermediate terms and hence memory requirements. The modification could have been left to the end using **std** but larger examples will readily exhaust the available heap space. The reduction in the number of

intermediate terms can be seen by moving the command 'supoutpt false' to just below the fourth line and entering rv1 as '221' and then repeating the example with the command line 'std_o gr3,last'.

To obtain the corresponding result for the spin irreducible representations $[\Delta;\lambda]$ of SO_7 we first note that

$$[\Delta;\lambda] = \Delta \cdot [\lambda/P] = \Delta \cdot \{\lambda/PC\} = \Delta \cdot \{\lambda/E\} \tag{6.9}$$

and that under $SO_7 \downarrow SU_2 \times SU_2 \times SU_2$

$$\Delta \downarrow \{0\}\{1\}\{1\} + \{0\}\{0\}\{2\} \tag{6.10}$$

Noting Eq. (6.8) we then obtain the branching rule

$$[\Delta;\lambda] \downarrow \Delta \cdot \sum_{\rho,\tau} \{\lambda/E\rho\} \circ \{\tau\} \cdot \{\tau\} \cdot \{2\} \otimes \{\rho\} \tag{6.11}$$

leading us to write the function as

```
gr so7
enter rv1
dim[rv1]
setr1ch_s rv1
gr4su2su2su2su2
rule[rv1*0*2*0]sk1 with e
rule last sum sk1eq4
cont3,4,pl,last
std_o gr3,last
rule last sum i1eq2
std last
p last,[0*1*1]+[1*0*1]
sup false
last
dim last
stop
```

In this case rv1 is entered as a spin irreducible representation and its dimension determined. The command line 'setr1ch_s rv1' removes the spin index and the function follows closely the previous case, apart from changing the skew series from C to E. Finally the whole result is multiplied by the result of Eq. (6.10).

■ 6.3 The U_1 trick in SCHUR

The unitary group in one dimension has one dimensional irreducible representations . Kronecker multiplication is accomplished by simple addition of the ir-

reducible representation labels. This simple observation allows the SCHUR user to employ a simple trick to get information that might be thought to be outside the scope of SCHUR. We illustrate this by an application to the calculation of the characters of the Hecke algebras $H_n(q)$ of type A_{n-1} as outlined by King and Wybourne (*J. Phys. A: Math.Gen.* **23** L1193–l1197 (1990). Basically they give a generalised power sum expansion

$$p_r(q;\mathrm{t}) = \sum_{\substack{a,b=0 \\ a+b+1=r}}^{r-1} (-1)^b q^a s_{(a+1,1^b)}(\mathrm{t}) \tag{6.12}$$

where q is an arbitrary but fixed complex parameter, $\mathbf{t} = (t_1, t_2, \ldots)$ an arbitrary set of indeterminates and the $S-$functions s_λ are all single hooks. For $\rho = (\rho_1, \rho_2, \ldots)$ they let

$$p_\rho(q;\mathbf{t}) = p_{\rho_1}(q;\mathbf{t}), p_{\rho_2}(q;\mathbf{t}), \ldots \tag{6.13}$$

and establish the relationship

$$p_\rho(q;\mathbf{t}) = \sum_\lambda \chi^\lambda_\rho(q) s_\lambda(\mathbf{t}) \tag{6.14}$$

where the $\chi^\lambda_\rho(q)$ are characters of $H_n(q)$ for the representation π_λ for the conjugacy class ρ.

As an example, consider the case of calculating the characters $\chi^\lambda_\rho(q)$ for the conjugacy class (321) for $H_6(q)$. We have from Eq. (6.12) that

$$\begin{aligned} p_3(q) &= q^2 s_3 - q s_{21} + s_{1^3} \\ p_2(q) &= q s_2 - s_{1^2} \\ p_1(q) &= s_1 \end{aligned} \tag{6.15}$$

where for typographical convenience we have suppressed the variables $\mathbf{t}$. It follows from Eq. (6.13) that

$$p_{(321)}(q) = (q^2 s_3 - q s_{21} + s_{1^3})(q s_2 - s_{1^2}) s_1 \tag{6.16}$$

Thus to calculate the desired characters we need to multiply out the right-hand-side of Eq. (6.16) and then gather together the polynomials in q associated with each s_λ. This may be done in SCHUR as follows.

1. Set the groups in the DPMode as $U_6 \times U_1$.
2. Set three variables var1, var2, and var3 for the three terms in Eq. (6.15) with the $S-$function as an irreducible representation of U_6 and its associated q exponent as the U_1 irreducible representation.
3. Now multiply the three vars together using the command 'p var1,p var2,var3'.
4. Each irreducible representation $\{a\}$ of U_1 is associated with a q exponent a.

   The whole procedure can be viewed as:-

```
DP>
group 2 u6u1
Groups are   U(6) * U(1)
DP>
setp1[3*2]-[21*1]+[1^3*0]
DP>
setp2[2*1]-[1^2*0]
DP>
setp3[1*0]
DP>
p var1,p var2,var3
      {6}{3} + 2{51}{3} - 2{51}{2} + 2{42}{3} - 3{42}{2}
       + {42}{1} + {41^2 }{3} - 4{41^2 }{2} + 2{41^2 }{1}
       + {3^2 }{3} - {3^2 }{2} + {3^2 }{1} + {321}{3}
       - 4{321}{2} + 4{321}{1} - {321}{0} - 2{31^3 }{2}
       + 4{31^3 }{1} - {31^3 }{0} - {2^3 }{2} + {2^3 }{1}
       - {2^3 }{0} - {2^2 1^2 }{2} + 3{2^2 1^2 }{1}
       - 2{2^2 1^2 }{0} + 2{21^4 }{1} - 2{21^4 }{0}
       - {1^6 }{0}
DP>
```

leading to the TEXoutput:-

```
sb_tex true
DP>
last
\{6\}\{3\} + 2\{51\}\{3\} - 2\{51\}\{2\} + 2\{42\}\{3\}
 - 3\{42\}\{2\} + \{42\}\{1\} + \{41^2 \}\{3\}
 - 4\{41^2 \}\{2\} + 2\{41^2 \}\{1\} + \{3^2 \}\{3\}
 - \{3^2 \}\{2\} + \{3^2 \}\{1\} + \{321\}\{3\}
 - 4\{321\}\{2\} + 4\{321\}\{1\} - \{321\}\{0\}
 - 2\{31^3 \}\{2\} + 4\{31^3 \}\{1\} - \{31^3 \}\{0\}
 - \{2^3 \}\{2\} + \{2^3 \}\{1\} - \{2^3 \}\{0\}
 - \{2^2 1^2 \}\{2\} + 3\{2^2 1^2 \}\{1\}
 - 2\{2^2 1^2 \}\{0\} + 2\{21^4 \}\{1\} - 2\{21^4 \}\{0\}
 - \{1^6 \}\{0\}
DP>
$$\eqalignno{
&\{6\}q^3 + \{51\}(2q^3-2q^2) + \{42\}(2q^3-3q^2+q)
 + \{41^2\}(q^3-4q^2+2q)\cr
& + \{3^2\}(q^3-q^2+q) + \{321\}(q^3-4q^2+4q-1)
 +\{31^3\}(-2q^2+4q-1)\cr
& + \{2^3\}(-q^2+q-1) + \{2^2 1^2\}(-q^2+3q-2)
 + \{21^4\}(2q-1)- \{1^6 \}\cr}$$
```

where the last segment was obtained by use of a text editor to produce a result for direct incorporation into TEX for publication purposes.

$$\{6\}q^3 + \{51\}(2q^3 - 2q^2) + \{42\}(2q^3 - 3q^2 + q) + \{41^2\}(q^3 - 4q^2 + 2q) + \{3^2\}(q^3 - q^2 + q) + \{321\}(q^3 - 4q^2 + 4q - 1) + \{31^3\}(-2q^2 + 4q - 1) + \{2^3\}(-q^2 + q - 1) + \{2^2 1^2\}(-q^2 + 3q - 2) + \{21^4\}(2q - 1) - \{1^6\}$$

■ **Exercise**

Establish the following character table for the Hecke algebra $H_4(q)$ of type A_3 and show that if $q \to 1$ the character table for the symmetric group $\mathcal{S}_4$ is recovered:-

	(1^4)	(21^2)	(2^2)	(31)	(4)
$\{4\}$	1	q	q^2	q^2	q^3
$\{31\}$	3	$2q-1$	q^2-2q	q^2-q	q^2
$\{2^2\}$	2	$q-1$	q^2+1	q	0
$\{21^2\}$	3	$q-2$	$-2q^2+1$	$-q+1$	q
$\{1^4\}$	1	-1	1	1	-1

6.4 The Final Test

As a final test for the user we suggest the following exercises. The person who completes this test without looking at the answers will indeed be qualified as a Master of SCHUR.

1. Use SCHUR to confirm the following three results in reduced notation:-

$$\begin{aligned} <2> &= <1>^2 - <1^2> - <1> - <0> \\ <21> &= <1^2><1> - <1^3> - <1>^2 + <0> \\ <3> &= <1>^3 + <1^3> - 2<1^2><1> - <1>^2 + <0> \end{aligned} \tag{6.17}$$

2. Show that for the group S_n the dimensions $f_n^{<\mu>}$ of the irreducible representations $<2>$, $<21>$ and $<3>$ are as follows:-

$$f_n^{<2>} = \frac{n(n-3)}{2} \tag{6.18a}$$

$$f_n^{<21>} = \frac{n(n-2)(n-4)}{3} \tag{6.18b}$$

$$f_n^{<3>} = \frac{n(n-1)(n-5)}{6} \tag{6.18c}$$

3. Show that if under $U_{\frac{n(n-3)}{2}} \rightarrow S_n \qquad \{1\} \rightarrow <2>$ then for an arbitrary irreducible representation $\{\lambda\}$ we have

$$\{\lambda\} \rightarrow \\ <1> \otimes \left[\sum_{\rho,\mu,\nu} (-1)^{\omega_\rho} \{\lambda/\rho \circ \mu\} \cdot \{\mu\} \cdot (\{1^2\} \otimes \{\tilde{\rho}/\nu\}) \cdot \{\nu/M\} \right] \tag{6.19}$$

4. Let $x = f_n^{<21>}$. Show that if under $U_x \rightarrow S_n \qquad \{1\} \rightarrow <21>$ then for an arbitrary irreducible representation $\{\lambda\}$ we have

$$\begin{aligned} &\{\lambda\} \rightarrow \\ &<1> \otimes \sum_{\rho,\mu,\nu,\tau} (-1)^{\omega_\rho} (\{1^2\} \otimes \{\lambda/M\rho \circ \mu\} \cdot \{\mu\}) \\ &\times \cdot (\{1^3\} \otimes \{\tilde{\rho}/\tau\}) \cdot \{\tau \circ \nu\} \cdot \{\nu\} \end{aligned} \tag{6.20}$$

5. Write a function to obtain the branching rule for Eq.(6.19) for $n = 8$ and obtain the decomposition for the $\{21\}$ irreducible representation for $U_{20} \rightarrow S_8$.
6. Write a function to obtain the branching rule for Eq.(6.20) for $n = 8$ and obtain the decomposition for the $\{21\}$ irreducible representation for $U_{64} \rightarrow S_8$.

   Possible answers to (5.) and (6.) are now given:-

Answer to 5. The function could be defined as:

```
gr u20
enter rv1
dim[rv1]
gr5u8u8u8u8u8
rule[rv1*0*1^2*0*0]sum sk1conj4
rule last ch_phase4
rule last sum i1eq2
cont1,2o,last
rule last sum sk3eq4
cont2,3pl last
cont1,2,o,last
rule last sk2,with m
cont1,2,o,last
gr s8
rule last i_pl1
sup false
rule last make1,8
dim last
stop
```

Running the above function gives:

```
DP>
->fn1
Group is U(20)
->21
Dimension = 2660
Groups are   U(8) * U(8) * U(8) * U(8) * U(8)
Groups are   U(8) * U(8) * U(8) * U(8)
Groups are   U(8) * U(8) * U(8)
Groups are   U(8) * U(8)
Group is U(8)
Group is S(8)
      2{71} + 5{62} + 4{61^2 } + 4{53} + 9{521} + 3{51^3 }
       + 2{4^2 } + 6{431} + 5{42^2 } + 5{421^2 } + {41^4 }
       + 2{3^2 2} + 3{3^2 1^2 } + 2{32^2 1} + {321^3 }
Dimension = 2660
```

Answer to 6. The function could be defined as:

```
gr u64
enter rv1
dim[rv1]
gr7u8u8u8u8u8u8u8
rule[1^2*rv1*0*1^3*0*0*0]sk2with m
rule last sum sk2conj5
rule last ch_phase5
rule last sum i2eq3
cont1,2pl last
cont1,2last
rule last sum sk3eq4
cont2,3pl last
cont1,2last
rule last sum i2eq3
cont1,2last
cont1,2last
gr s8
rule last i_pl1
sup false
rule last make1,8
dim last
stop
DP>
```

Running the above function gives the answer:

```
->fn1
Group is U(64)
->21
Dimension = 87360
Groups are   U(8) * U(8) * U(8) * U(8) * U(8) * U(8) * U(8)
Groups are   U(8) * U(8) * U(8) * U(8) * U(8) * U(8)
Groups are   U(8) * U(8) * U(8) * U(8) * U(8)
Groups are   U(8) * U(8) * U(8) * U(8)
Groups are   U(8) * U(8) * U(8)
Groups are   U(8) * U(8)
Group is U(8)
Group is S(8)
  2{8} + 19{71} + 53{62} + 54{61^2 } + 69{53} + 156{521}
 + 79{51^3 } + 33{4^2 } + 160{431} + 126{42^2 }
 + 195{421^2 } + 70{41^4 } + 91{3^2 2} + 118{3^2 1^2 }
 + 142{32^2 1} + 124{321^3 } + 37{31^5 } + 25{2^4 }
 + 51{2^3 1^2 } + 35{2^2 1^4 } + 10{21^6 } + {1^8 }
Dimension = 87360
```

People have now a-days, got a strange opinion that everything should be taught by lectures. Now, I cannot see that lectures can do so much good as reading the books from which the lectures are taken. I know nothing that can be best taught by lectures, except where experiments are to be shewn. You may teach chymistry by lectures. You might teach the making of shoes by lectures!
— Samuel Johnson 1766

7

Examples of SCHUR in Physics, Chemistry and Mathematics

■ Introduction

In this chapter we give some detailed examples of using SCHUR in the fields of physics, chemistry and mathematics. In physics we illustrate the use of SCHUR in studying group aspects of the Standard Model and its extensions in particle physics while for chemistry we look at some aspects of the electronic states of the nitrogen molecule, N_2. Finally we look at the problem of the asymptotic behaviour of certain group-subgroup decompositions and unimodal distributions. The examples in this chapter should also indicate some of the potential of SCHUR in pedagogical as well as research situations.

■ 7.1 The Simple SU_3 Quark Model of Baryons and Mesons

The simple quark model of the low energy baryons and mesons was based upon the group structure $SU_3 \supset U_1^Y \times SU_2^I$ where Y is the hypercharge and I the isospin. Using SCHUR in the REPmode with the group set as SU_3 we find that there are two irreducible representations of dimension 3 corresponding to the vector representation $\{1\}$ and its conjugate $\{1^2\}$. In the literature these are often designated as **3** and $\overline{\mathbf{3}}$. Now to find the subgroup content of these two irreducible representations. We return to the DPMode and thence to the BRMode and set the branching rule numbers to 16 1 2 and see that the groups are set as SU(3) to U(1) ∗ SU(2). Entering the first irreducible representation, $\{1\}$ gives the result

$$\{1\} \to \{1\}\{1\} + \{-2\}\{0\} \tag{7.1}$$

while entering $\{1^2\}$ gives

$$\{1^2\} \to \{2\}\{0\} + \{-1\}\{1\} \tag{7.2}$$

Now to determine the hypercharge Y and isospin I for each irreducible representation. The irreducible representations of the isospin group SU_2^I are labelled by a single part partitions, $\{m\}$. Recalling the local isomorphism between SO_3 and SU_2 with SU_2 being the covering group of SU_2 we obtain the usual isospin quantum number I by dividing m by 2. i.e. $\{m\} \sim [\frac{m}{2}]$. In normalising the U_1 irreducible representations for $SU_n \to U_1 \times SU_{n-1}$ SCHUR ensures that they come as integers and satisfy the usual tracelessness condition for a U_1 group that occurs in a direct product. The normalisation is fixed so that the weight of the label of the U_1 irreducible representation associated with the vector irreducible representation $\{1\}$ of SU_{n-1} is of unit magnitude.With these remarks in view, the hypercharge Y is one third of the U_1 weight reported by SCHUR. Thus in terms of the hypercharge Y and the isospin I we could rewrite Eqs (7.1) and (7.2) as

$$\{1\} \to (\tfrac{1}{3}, \tfrac{1}{2}) + (-\tfrac{2}{3}, 0) \tag{7.1$'$}$$

and

$$\{1^2\} \to (\tfrac{2}{3}, 0) + (-\tfrac{1}{3}, \tfrac{1}{2}) \tag{7.2$'$}$$

where the quantum number pairs Y and I are given in curved brackets as (Y, I). These quantum numbers are exactly those associated with the (u, d, s) quarks

and $(\overline{u}, \overline{d}, \overline{s})$ antiquarks respectively. The charges Q of these objects can be deduced from the relationship (in units of e)

$$Q = I_3 + \tfrac{Y}{2} \tag{7.3}$$

leading to

SU_3	quark	I	I_z	Y	Q
$\{1\}$	u	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{2}{3}$
	d		$-\frac{1}{2}$		$-\frac{1}{3}$
	s	0	0	$-\frac{2}{3}$	$-\frac{1}{3}$
$\{1^2\}$	$\bar{u}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{3}$	$-\frac{2}{3}$
	$\bar{d}$		$\frac{1}{2}$		$\frac{1}{3}$
	$\bar{s}$	0	0	$\frac{2}{3}$	$\frac{1}{3}$

Mesons are built from quark-antiquark $(q\overline{q})$ pairs. Returning to the REPmode with the group set as for SU_3 we find that

$$\{1\} \times \{1^2\} = \{21\} + \{0\} \tag{7.4}$$

The irreducible representation $\{21\}$ is found to be of dimension 8 and hence we obtain an octet and a singlet of mesons. The quantum numbers (Y,I) to be associated with members of the octet follow by determination of the $SU_3 \rightarrow U_1 \times SU_2$ branching rule in either the DPMode or BRMode to give

$$\{21\} \rightarrow \{3\}\{1\} + \{0\}\{2\} + \{0\}\{0\} + \{-3\}\{1\} \tag{7.5}$$

and hence the (Y, I) quantum numbers

$$\{21\} \rightarrow (1, \tfrac{1}{2}) + (0, 1) + (0, 0) + (-1, \tfrac{1}{2}) \tag{7.6}$$

The meson charges may be determined from Eq. (7.3) and it is left as an exercise to plot out the numbers (I_3, Y) to obtain the familiar display of the meson nonet (octet + singlet).

The baryons are built from triple products of quarks (qqq). Evaluating the triple product in the REPmode with the group set as for SU_3 and using the nested command "prod1,prod1,1" leads to

$$\{1\} \times \{1\} \times \{1\} = \{3\} + 2\{21\} + \{0\} \tag{7.7}$$

The irreducible representation $\{3\}$ is of dimension 10 and thus we have obtained the irreducible representations associated with a decuplet $(\{3\})$, two octets $(\{21\})$ and a singlet $(\{0\})$ of baryons. The (Y, I) quantum numbers associated with the baryon octets are exactly the same as those found for the meson octet found in Eq. (7.6). Those for the decuplet follow from the $SU_3 \rightarrow U_1 \times SU_2$ branching rule with SCHUR readily yielding

$$\{3\} \rightarrow \{3\}\{3\} + \{0\}\{2\} + \{-3\}\{1\} + \{-6\}\{0\} \tag{7.8}$$

and hence the (Y, I) quantum numbers

$$\{3\} \rightarrow (1, \tfrac{3}{2}) + (0, 1) + (-1, \tfrac{1}{2}) + (-2, 0) \tag{7.9}$$

Again it is left as an exercise to make the familiar (I_3, Y) plot for the baryon decuplet.

The above gives an introduction to using SCHUR as a tool for learning about the simple SU_3 model of the low energy mesons and baryons. The SCHUR user should now have enough skill to be able to look at models involving additional quarks such as, for example, the charmed quark (c) in SU_4.

7.2 Unification Models and QCD

Many attempts have been made to unify the weak and strong interactions. Here we use SCHUR to give the group content of some well-known examples. The simplest model is that associated with the group structure

$$SU_5 \supset U_1^{Y_{wk}} \times SU_2^{I_{wk}} \times SU_3^c \tag{7.10}$$

where the $U_1^{Y_{wk}} \times SU_2^{I_{wk}}$ group is that associated with electroweak theory and SU_3^c is the color group of quantum chromodynamics (QCD), the theory of strong interactions. For this study it is convenient to write the short SCHUR function

```
group su5
enter rv1
dim[rv1]
br8 2 3gr1[rv1]
stop
```

and to run the function for the SU_5 irreducible representations $\{1\}$, $\{1^4\}$, $\{1^2\}$, $\{1^3\}$ and $\{21^3\}$ to yield the following results:-

$$\mathbf{5}\ \{1\} \rightarrow \{3\}\{1\}\{0\} + \{-2\}\{0\}\{1\} \tag{7.11a}$$

$$\overline{\mathbf{5}}\ \{1^4\} \rightarrow \{2\}\{0\}\{1^2\} + \{-3\}\{1\}\{0\} \tag{7.11b}$$

$$\mathbf{10}\ \{1^2\} \rightarrow \{6\}\{0\}\{0\} + \{-1\}\{1\}\{1\} + \{-4\}\{0\}\{1^2\} \tag{7.11c}$$

$$\overline{\mathbf{10}}\{1^3\} \rightarrow \{4\}\{0\}\{1\} + \{-1\}\{1\}\{1^2\} + \{-6\}\{0\}\{0\} \tag{7.11d}$$

$$\mathbf{24}\ \{21^3\} \rightarrow \{5\}\{1\}\{1^2\} + \{0\}\{2\}\{0\} + \{0\}\{0\}\{21\} + \{0\}\{0\}\{0\} + \{-5\}\{1\}\{1\} \tag{7.11e}$$

The above results can be rewritten in terms of the quantum numbers (Y^{wk}, I^{wk}) and the dimensions of the SU_3^c irreducible representations as

$$\mathbf{5}\ \{1\} \rightarrow (1, \tfrac{1}{2}, \mathbf{1}^c) + (-\tfrac{2}{3}, 0, \mathbf{3}^c) \tag{7.12a}$$

$$\overline{\mathbf{5}}\ \{1^4\} \rightarrow (\tfrac{2}{3}, 0, \overline{\mathbf{3}}^c) + (-1, \tfrac{1}{2}, \mathbf{1}^c) \tag{7.12b}$$

$$\mathbf{10}\ \{1^2\} \rightarrow (2, 0, \mathbf{1}^c) + (\tfrac{1}{3}, \tfrac{1}{2}, \mathbf{3}^c) + (-\tfrac{4}{3}, 0, \overline{\mathbf{3}}^c) \tag{7.12c}$$

$$\overline{\mathbf{10}}\{1^3\} \rightarrow (\tfrac{4}{3}, 0, \mathbf{3}^c) + (-\tfrac{1}{3}, \tfrac{1}{2}, \overline{\mathbf{3}}^c) + (-2, 0, \mathbf{1}^c) \tag{7.12d}$$

$$\mathbf{24}\ \{21^3\} \rightarrow (\tfrac{5}{3}, \tfrac{1}{2}, \overline{\mathbf{3}}^c) + (0, 1, \mathbf{1}^c) + (0, 0, \mathbf{8}^c) + (0, 0, \mathbf{1}^c) + (-\tfrac{5}{3}, \tfrac{1}{2}, \mathbf{3}^c) \tag{7.12e}$$

We can associate a family of fermions with the 15 dimensional reducible representation $\{1^4\} + \{1^2\} \sim \overline{5} + \mathbf{10}$ of $SU_5 \supset U_1 \times SU_2 \times SU_3^c$ and a set of 24 bosons with the adjoint representation $\{21^3\} \sim \mathbf{24}$. The relevant (Y, I, I_3, Q) follow exactly as before to give for the 15 fermions:-

SU_5	fermion	I	I_z	Y	Q	color
$\bar{\mathbf{5}}$	ν_L	$\frac{1}{2}$	$\frac{1}{2}$	-1	0	$\mathbf{1}^c$
	e_L		$-\frac{1}{2}$		-1	
	d_L^c	0	0	$\frac{2}{3}$	$\frac{1}{3}$	$\bar{\mathbf{3}}^c$
$\mathbf{10}$	u_L	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{2}{3}$	$\mathbf{3}^c$
	d_L		$-\frac{1}{2}$		$-\frac{1}{3}$	
	u_L^c	0	0	$-\frac{4}{3}$	$-\frac{2}{3}$	$\bar{\mathbf{3}}^c$
	e_L^c	0	0	2	1	$\mathbf{1}^c$

These are precisely the quantum numbers of the first family of 15 fermions of the SU_5 Grand Unification Theory.

The quantum numbers for the 24 gauge bosons follow from Eq. (7.12e) in a similar manner . Thus $(0, 0, \mathbf{1}^c)$ yields the quantum numbers of the photon, $(0, 1, \mathbf{1}^c)$ those of the $W^{\pm}$ weakons, with their Z^0 partner coming from $(0, 0, \mathbf{1}^c)$ and $(0, 0, \mathbf{8}^c)$ leads to the quantum numbers of the eight gluons of QCD.

A disappointing feature of this model is the occurence of the fermions in a reducible representation. One might wish to search for a group having a single irreducible representation that can be spanned by an entire fermion family. Returning to SCHUR and looking at the dimensions of rank 5 Lie groups we find there is no suitable group with an appropriate 15 dimensional irreducible representation. Setting the group for SO_{10} we find that the two basic spin irreducible representations $\Delta_{\pm}$ are of dimension 16 and decompose under $SO_{10} \rightarrow U_1 \times SU_5$ (using branching rule 11 with $n = 10$) as

$$\Delta_+ \rightarrow \{3\}\{1^4\} + \{-1\}\{1^2\} + \{-5\}\{0\} \tag{7.13a}$$

$$\Delta_- \rightarrow \{5\}\{0\} + \{1\}\{1^3\} + \{-3\}\{1\} \tag{7.13b}$$

The decomposition of the Δ_+ irreducible representation of SO_{10} yields the $\overline{\mathbf{5}}+\mathbf{10}$ reducible representation of SU_5 but contains one additional fermion having the quantum numbers of a right-handed neutrino.

The above examples should suffice for the user to explore other structures in particle physics using SCHUR.

7.3 Electronic States of the N_2 Molecule

The following example illustrates a number of aspects of applying SCHUR to the problem of enumerating the electronic states of the nitrogen molecule, N_2, and at the same time should give the user additional insights into using SCHUR and the need, sometimes, to adopt particular strategies.

The nitrogen molecule has 14 electrons that we shall consider to be distributed among the ten atomic orbitals ($1s, 2s$ and $2p$). The atomic orbitals can be considered as forming a basis for the vector irreducible representation $\{1\}$ of SU_{10}. The two spin states ($m_s = \pm\frac{1}{2}$) give a basis for the vector irreducible representation $\{1\}$ of SU_2. The total wavefunction should be antisymmetric. For a single particle, combining the spin part of the wavefunction with the orbital part leads to a basis for the vector irreducible representation $\{1\}$ of the group U_{20}. The complete set of antisymmetric states formed by the 14 electrons in N_2 distributed over the ten atomic orbitals will span the $\{1^{14}\}$ irreducible representation of U_{20}. To find out how many states arise we set the group as for U_{20} and issue the command in the DPMode

```
dim[1^!14]
```

and are informed that the dimension of $\{1^{14}\}$ is 38 760. What are the total spins associated with these states? To answer that question issue the command

```
br8 2 10gr1[1^!14]
```

and SCHUR responds with

```
Groups are U(2)*U(10)

{10 4}{2^4 1^6} + {95}{2^5 1^4} + {86}{2^6 1^2} + {7^2}{2^7}
```

If we reset the groups with the command

```
gr2su2su10
```

and then

```
std last
```

we obtain the output

```
Groups are SU(2)*SU(10)

{6}{1^4} + {4}{2^5 1^4} + {2}{2^6 1^2} + {0}{2^7}
```

Recalling the isomorphism $SU_2 \sim SO_3$ we can obtain the spin S by dividing the one part label of the SU_2 irreducible representations by 2 alternatively we may issue the command

```
au gr1so3,last
```

giving the output

Groups are SO(3)*SU(10)

$$\{3\}\{1\hat{}4\} + \{2\}\{2\hat{}5\ 1\hat{}4\} + \{1\}\{2\hat{}6\ 1\hat{}2\ \} + \{0\}\{2\hat{}7\ \}$$

We can save this result for later use, if desired, by simply issuing the command

setp1last

We can recover it anytime later by issuing the command

v1

but remember to reset the groups back to SO(3)*SU(10) first if the groups have in the meantime been changed.

Computing the dimensions of the above SU_{10} irreducible representations we find there are 210 states with $S = 3$, 2310 with $S = 2$, 6930 with $S = 1$ and 4950 with $S = 0$. Thus the answer to our question is that there are 4950 singlet states.

Let us now assume that the the core orbitals $(1s, 2s)$ are separated from the valence orbitals $(2p)$. This means that we can consider the orbitals as spanning irreps of $SU_{10} \supset SU_2^c \times SU_8^v$. For the singlet states of N_2 we need the decomposition of the irreducible representation $\{2\hat{}7\}$ under $SU_{10} \supset SU_2^c \times SU_8^v$. To do this set the group for SU_{10} and then issue the command

br8 2 8gr1[2^7]

SCHUR responds with the output

Groups are U(1)*SU(2)*SU(8)

$$\{12\}\{0\}\{2\hat{}5\} + \{2\}\{1\}\{2\hat{}51\} + \{-8\}\{2\}\{2\hat{}6\} + \{-8\}\{0\}\{2\hat{}51\hat{}2\}$$
$$+\{-18\}\{1\}\{2\hat{}61\} + \{-28\}\{0\}\{2\hat{}7\} \tag{7.14}$$

What are we to make of the U_1 irreducible representations? Had we branched the vector irreducible representation $\{1\}$ we would have obtained the decomposition

$$\{1\} \to \{8\}\{1\}\{0\} + \{-2\}\{0\}\{1\} \tag{7.15}$$

Here the interpretation is clear the first irreducible representation is associated with a single electron in the core and none in the valence orbitals and the second *vice versa*. Let n, n_c and n_v be the total number of, the number of core, and number of valence electrons respectively. Noting the Eq. (7.14) and that

$$n = n_c + n_v \tag{7.16}$$

we readily conclude that if k is the U_1 eigenvalue labelling the U_1 irreducible representation then

$$n_c = \frac{2n + k}{10} \tag{7.17}$$

If we issue the command

uonediv10gr1uoneadd28gr1last

where last is the output given in Eq. (7.14) we obtain the output

$$\{4\}\{0\}\{2\hat{}5\} + \{3\}\{1\}\{2\hat{}51\} + \{2\}\{2\}\{2\hat{}6\}$$
$$+\{2\}\{0\}\{2^5 1\hat{}2\} + \{1\}\{1\}\{2\hat{}61\} + \{0\}\{0\}\{2\hat{}7\} \tag{7.18}$$

where now for each irreducible representation the first label gives the number of electrons n_c occupying core orbitals. The number of electrons n_v occupying valence orbitals follows trivially from Eq. (7.16).

The total spins S_c and S_v carried by the core and valence electrons can now be readily determined. Remembering that we are dealing with molecular singlet states we must necessarily have $S_c = S_v$. The first term in Eq. (7.18) has $S_c = 0$ since it coreesponds to total occupancy of the core. The second and fifth terms have $S_c = \frac{1}{2}$ corresponding to single hole in the core for the second and a single electron in the core in the fifth. The sixth term corresponds to an empty core and hence $S_c = 0$. The third term and fourth term have two electrons in the core and thus we could expect $S_c = 0$ or 1. But for the third term the core is orbital symmetric and hence the spin part must be antisymmetric leading to $S_c = 0$ whereas the opposite is the case for the fourth term and $S_c = 1$.

In general the spins may be quickly determined by noting that for n electrons in orbital states classified by some SU_N the irreducible representations are all of the form

$$\{\lambda\} = \{2^{\frac{n-2S}{2}} 1^{2S}\} \tag{7.19}$$

where the length of the partition satisfies

$$\ell_\lambda = \tfrac{n+2S}{2} \leq N \tag{7.20}$$

and if $\ell_\lambda = N$ then there is the SU_N equivalence

$$\{\lambda_1, \lambda_2, \ldots, \lambda_N\} \equiv \{\lambda_1 - \lambda_N, \lambda_2 - \lambda_N, \ldots, 0\} \tag{7.21}$$

Thus for 12 electrons in SU_8 we have possible irreducible representations of the form $\{2^{6-S}\ 1^{2S}\}$ which for $S = 1$ corresponds to $\{2^5\ 1^2\}$ and hence our assignment of $S_c = S_v = 1$ for the fourth term in Eq. (7.18).

Now consider the case of the singlet states of N_2 with the core fully occupied. That leaves 8 electrons in valence orbitals associated with the $\{2^5\}$ irreducible representation of SU_8. SCHUR gives the dimension of that irreducible representation as 1176 and thus we have 1176 singlet states associated with the two atoms which we will distinguish by the letters a and b. If we now perform the branching $SU_8 \to U_1 \times SU_4^a \times SU_4^b$ and then use the commands **uoned**iv and **uonea**dd, as before, we obtain the SCHUR output

```
DP>
br8 4 4gr1[2^5]
Groups are   U(1) * SU(4) * SU(4)
      {24 }{0}{2} + {16 }{1^3 }{21} + {8}{2^3 }{2^2 }
       + {8}{1^2 }{21^2 } + {0}{2^2 1}{2^2 1} + {0}{1}{1}
       + {-8}{2^2 }{2^3 } + {-8}{21^2 }{1^2 }
       + {-16  }{21}{1^3 } + {-24  }{2}{0}
DP>
uonediv8gr1uoneadd40gr1last
      {8}{0}{2} + {7}{1^3 }{21} + {6}{2^3 }{2^2 }
       + {6}{1^2 }{21^2 } + {5}{2^2 1}{2^2 1} + {5}{1}{1}
       + {4}{2^2 }{2^3 } + {4}{21^2 }{1^2 }
       + {3}{21}{1^3 } + {2}{2}{0}
```

where now the U_1 label gives the number of electrons in orbitals associated with atom a. The corresponding spins associated with the respective atoms again follow from Eqs. (7.19) - (7.21) to give the states

$$\{0\}_8^{a,s}\{2\}_2^{b,s} + \{1^3\}_7^{a,d}\{21\}_3^{b,d} + \{2^3\}_6^{a,s}\{2^2\}_4^{b,s} + \{1^2\}_6^{a,t}\{21^2\}_4^{b,t}$$
$$+ \{2^21\}_5^{a,d}\{2^21\}_5^{b,d} + \{1\}_5^{a,d}\{1\}_5^{b,d} + \{2^2\}_4^{a,s}\{2^3\}_6^{b,s} + \{21^2\}_4^{a,t}\{1^2\}_6^{b,t}$$
$$+ \{21\}_3^{a,d}\{1^3\}_7^{b,d} + \{2\}_2^{a,s}\{0\}_8^{b,s} \tag{7.22}$$

where the electron numbers are given as subscripts and the superscript letters $s,\ d,\ t$ designate singlet, doublet and triplet spin states respectively.

The preceding example gives a brief view of the application of SCHUR to electronic states of molecules and the user is encouraged to explore such topics further.

7.4 Plethysms and Asymptopia

The user of almost any software package quickly learns its limitations. Some problems take too long to execute while others quickly exhaust all the memory available to the user. Many of these problems cannot be overcome simply by the acquisition of a larger and faster computer. They, in principle, become combinatorially explosive and even a better algorithm will not significantly improve the situation. In these cases alternative strategies have to be devised. For large problems we may no longer be interested in individual numbers but rather in their asymptotic behaviour. The future is more likely to be found in asymptopia rather than in utopia. Plethysms form an example of a combinatorially explosive situation which we will now briefly explore, for a very restricted case, by way of several open-ended user exercises.

Suppose the rotation group SO_3 is embedded in the unitary group U_{2j+1} so that under $U_{2j+1} \Rightarrow SO_3$ the vector representation $\{1\} \rightarrow [j]$. An arbitrary irreducible representation $\{\lambda\}$ of U_{2j+1} decomposes into irreducible representations $[J]$ of SO_3 as

$$[j] \otimes \{\lambda\} = \sum_J c_J^{\{\lambda\}} [J] \tag{7.23}$$

where the coefficients $c_J^{\{\lambda\}}$ are non-negative integers. These coefficients may be calculated using branching rule 7. In evaluating these coefficients SCHUR uses a method involving polynomial division rather than the usual plethysm routines. It is worth noting the identity, for SO_3,

$$[j] \otimes \{1^n\} = [(2j - n + 1)/2] \otimes \{n\} \tag{7.24}$$

as the symmetric plethysms are evaluated much faster than the corresponding antisymmetric SO_3 plethysms.

Choose $j = 8$ and determine the decomposition of the U_{2j+1} irreducible representations $\{n\}$ for values of $n \leq 13$ (Going to $n \geq 14$ will create problems for users of 16-bit computers as then many of the coefficients exceed *MaxInt*). Observe the strong tendency to a unimodal distribution of the $c_J^{\{\lambda\}}$ with respect to J. Can we approximate the form of the distribution?

For sufficiently large j and w_λ we have

$$c_J^{\{\lambda\}} \sim A(J + \frac{1}{2}) \exp\left[-\frac{(J + \frac{1}{2})^2}{2\sigma(j,\lambda)^2}\right] \tag{7.25}$$

Replacing summations by integration leads to

$$\begin{aligned} \sum_J c_J^{\{\lambda\}} &\sim A \int_0^\infty (J + \frac{1}{2}) \exp\left[-\frac{(J + \frac{1}{2})^2}{2\sigma(j,\lambda)^2}\right] dJ \\ &= A\sigma(j,\lambda)^2 \end{aligned} \tag{7.26}$$

The left-hand-side of (7.26) is the sum of the coefficients as computed by the SCHUR command **countc**.

Likewise

$$\sum_J (2J+1)c_J^{\{\lambda\}} \sim A\sigma(j,\lambda)^3\sqrt{2\pi}$$
$$= Dim_\lambda \tag{7.27}$$

where Dim_λ is the dimension of the irreducible representation $\{\lambda\}$ of U_{2j+1} furthermore

$$\sum_J J(J+1)(2J+1)c_J^{\{\lambda\}} \sim A\sigma(j,\lambda)^3\sqrt{2\pi}(3\sigma(j,\lambda)^2 - \frac{1}{4})$$
$$= j(j+1)(2j+1)I_2^{\{\lambda\}} \tag{7.28}$$

where $I_2^{\{\lambda\}}$ is the second-order Dynkyn index for the irreducible representation $\{\lambda\}$ of U_{2j+1}. The factor $j(j+1)(2j+1)$ appearing in the right-hand-side of (7.28) arises from matching the normalisation of second-order Dynkin indexes for the two groups.

Let J_m denote the SO_3 irreducible representation $[J_m]$ associated with the maximal coefficient $c_{J_m}^{\{\lambda\}}$. We find

$$J_m \sim \sigma(j,\lambda) - \frac{1}{2} \qquad c_{J_m}^{\{\lambda\}} \sim A\sigma(j,\lambda)\exp(-\frac{1}{2}) \tag{7.29}$$

Use of the above results leads to

$$\sigma(j,\lambda) \sim \sqrt{\frac{j(j+1)(2j+1)I_2^{\{\lambda\}}}{3Dim_\lambda}} \tag{7.30}$$

Thus all the parameters defining the distribution can be directly calculated by SCHUR within the group SU_{2j+1}. But how closely does the distribution given in Eq. (7.25) approximate the exact calculations by SCHUR? That is left as an exercise for the user. Perhaps the user can obtain a better approximation and explore the many aspects of the asymptotic distribution of the coefficients generated by SCHUR.

If your experiment needs statistics you ought to have done a better experiment
— Ernest Rutherford *1851–1937*

8

Further Reading for users of SCHUR

8.1 Introduction

SCHUR incorporates a large amount of mathematics that is unfamiliar to many mathematicians and physicists. While many may wish to simply treat SCHUR as a black box that responds to commands and issues forth results others may wish to know more of the background mathematics contained in SCHUR and to explore the literature in greater detail. This chapter is for the latter class of SCHUR users. Historically many of the tools were forged at the end of the nineteenth and beginning of the twentieth centuries. Elie Cartan[10] classified the complex semisimple Lie algebras in his thesis of 1894 while Issia Schur[110] in his dissertation of 1901 discussed the properties of matrix groups. Meanwhile, the English cleric Alfred Young was starting his monumental work on the symmetric group. Young's papers[159] while rich in content are rather inpenetrable. Young's work was brought to the attention of physicists largely by Hermann Weyl[125] who himself had made major contributions to the character theory of Lie groups. Daniel Rutherford[101] drew to the attention of mathematicians the work of Young.

The 1930's saw considerable developments in the theory of the symmetric group and matrix groups in general especially in the hands of Francis Murnaghan[80] and Dudley Littlewood[60]. Both wrote significant books though it would be fair to say that their style of presentation probably appears obscure to modern readers. Littlewood placed considerable emphasis on the theory of symmetric functions which had already made its appearance in Isaac Newton's 1707 *Arithmetica Universalis*. Much of the theory of symmetric functions had been collected together and further developed by Major Percy MacMahon in his two volume work *Combinatory Analysis* of 1915. Littlewood's treatment of the characters of the classical continuous groups involved extensive use of what he termed "Schur functions" and abbreviated to $S-$functions. These functions had been introduced by Jacobi in 1841 and extensively exploited by Schur. Littlewood and Richardson[68] in 1934 were fortunate to guess the rule for multiplying Schur functions with formal proofs coming in the 1970's.

Littlewood[56−68] continued for some thirty years developing the theory of group characters and symmetric functions. Notable for SCHUR was his introduction of the plethysm of $S-$functions and the two papers[57,58] of 1944 in which the entire subject was developed in connection with invariant theory. The computation of plethysms still remains an incomplete problem in spite of numerous studies, some of which are listed in the references.

Since the time of Littlewood and Murnaghan there have been many further developments and the collected references, while representative, are far from complete. New applications are coming to note almost daily and only a selection of applications have been referenced. For a gentle introduction to the symmetric group the book by Sagan[103] is recommended followed by the more comprehensive books of Macdonald[71] and of James and Kerber[35]. For physicists Messiah[73] gives a gentle summary of Young tableaux and Young symmetrisers while Hamermesh[31] gives rather more detail. For papers giving specific appli-

cations of SCHUR the reader could note the references [23, [28], [45], [49], [51], [79], [100], [108]–[111], [119], [120], [147]–[150], [153-155] and [158].

■ 8.2 References

[1] G. E. Andrews, *The theory of partitions*, Addison-Wesley, Reading, MA. 1976.

[2] G. R. E. Black, R. C. King, and B. G. Wybourne, Kronecker products for compact semisimple Lie groups, *J. Phys. A: Math. Gen.* **16**, 1555–1589 (1983).

[3] G. R. E. Black and B. G. Wybourne, Branching rules and even-dimensional rotation groups SO_{2k}, *J. Phys. A:Math. Gen.* **16**, 2405–2421 (1983).

[4] P. H. Butler and R. C. King, Branching rules for $U(N) \supset U(M)$ and the evaluation of outer plethysms, *J. Math. Phys.* **14**, 741–745 (1973).

[5] P. H. Butler and B. G. Wybourne, Applications of S-functional analysis to continuous groups,*J. de Phys.* **30**, 795–802 (1969).

[6] P. H. Butler and B. G. Wybourne, Reduction of the Kronecker products for rotational groups, *J. de Phys.* **30**, 655–664 (1969).

[7] P. H. Butler and B. G. Wybourne, Tables of outer S-function plethysms, *Atomic Data* **3**, 133–151 (1971).

[8] C. Carré and J. -Y. Thibon, Plethysm and vertex operators, *Adv. in Appl. Math.*

[9] C. Carré and J. -Y. Thibon, Some formulas for plethysm coefficients, *LITP 91.63 Inst. Blaise-Pascal, Paris 1991.*

[10] E. Cartan, *Sur la structure des groupes de transformation finis et continus*, Thesis, Nony, Paris 1894.

[11] M. J. Carvalho, Symmetrised Kronecker products of the fundamental representation of $Sp(n, R)$, *J. Phys. A: Math. Gen.* **23**, 1909–1927 (1990).

[12] Y. M. Chen, A. M. Garsia and J. Remmel, Algorithms for plethysm, *Contemporary Mathematics* **34**, 109–153 (1984).

[13] J. G. Cleary and B. G. Wybourne, Statistical group theory and the distribution of angular momentum states, *J. Math. Phys.* **12**, 45–52 (1971).

[14] C. J. Cummins, Inverse branching rules, *J. Phys. A: Math. Gen.* **22**, L1055–L1060 (1989).

[15] C. J. Cummins, $su(n)$ and $sp(2n)$ WZW fusion rules, *J. Phys. A: Math. Gen.* **24**, 391–400 (1991).

[16] D. G. Duncan, Note on a formula by Todd, *J. London Math. Soc.* **27**, 235–236 (1952).

[17] D. G. Duncan, On D. E. Littlewood's algebra of $S-$functions, *Can. J. Math.* **4**, 504–512 (1952).

[18] D. G. Duncan, Note on the algebra of $S-$functions, *Can. J. Math.* **6**, 509–510 (1952).

[19] Ö. Eğecioğlu, Computation of outer products of Schur functions, *Computer Phys. Comm.* **28**, 183–187 (1982).

[20] Ö. Eğecioğlu and J. B. Remmel, Symmetric and antisymmetric outer plethysms of Schur functions, *Atomic Data and Nuclear Data Tables* **32**, 157–196 (1985).

[21] N. El Samra and R. C. King, Dimensions of irreducible representations of the classical Lie groups, *J. Phys. A: Math. Gen.* **12**, 2317–2328 (1979).

[22] H. K. Farahat, On Schur functions, *Proc. London Math. Soc. (3)* **8**, 621–630 (1958).

[23] R. J. Farmer, R. C. King and B. G. Wybourne, Spectrum-generating functions for strings and superstrings, *J. Phys. A: Math. Gen.* **21**, 3979–4007 (1988).

[24] H. O. Foulkes, Differential operators associated with $S-$functions, *Quart. J. Math.* **24**, 136–143 (1949).

[25] H. O. Foulkes, The new multiplication of $S-$functions, *J. London Math. Soc.* **26**, 132–139 (1951).

[26] H. O. Foulkes, Plethysm of $S-$functions, *Phil. Transac. Roy. Soc.* (London) *A* **246**, 555–591 (1954).

[27] J. S. Frame, G. de B. Robinson and R. M. Thrall, The hook graphs of S_n, *Can. J. Math.* **6** 316–324 (1954).

[28] S. A. Fulling, R. C. King, B. G. Wybourne and C. J. Cummins, Normal forms for tensor polynomials I. The Riemann tensor, *Class. Quantum Grav.* **9**, 1151–1197 (1992).

[29] A. M. Garsia and J. B. Remmel, Shuffles of permutations and the Kronecker product, *Graphs and Combinatorics* **1**, 217–263 (1985).

[30] A. M. Garsia and J. B. Remmel, Symmetric functions and raising operators, *Linear and Multilinear Algebra* **10**, 15–43 (1981)..

[31] K. Grudzinski and B. G. Wybourne,Computing Properties of the Non-Compact Groups Mp(2n) and Sp(2n,R) using SCHUR Proc. Third Intnl School on Theoretical Physics, (1995) pp 469-483 (World Scientific).

[32] K. Grudzinski and B. G. Wybourne,Symplectic models of n-particle systems *Rept. Math. Phys.* **38**, 251-266 (1996).

[33] K. Grudzinski and B. G. Wybourne,Plethysm for the noncompact group Sp(2n,R) and new S-function identities *J. Phys. A:Math. Gen.* **29**, 6631-6641 (1996).

[34] M. Hamermesh, *Group Theory and its Application to Physical Problems*, Addison–Wesley, Reading, Mass., 1962.

[35] M. G. Hirst and B. G. Wybourne, Statistical group theory and the distribution of angular momentum states:II , *J. Phys. A: Math. Gen.* **19**, 1545-1549 (1986).

[36] R. Howe, $(Gl_n, Gl_m)-$duality and symmetric plethysm, *Proc. Indian Acad. Sci. (Math. Sci.)* **97**, 85–109 (1987).

[37] E. M. Ibrahim, *Tables for plethysms of $S-$functions*, Royal Society of London, Depository of Unpublished Tables (19–).

[38] G. D. James and A. Kerber, *The Representation Theory of the Symmetric Group*, Addison–Wesley, Reading, Mass., 1981.

[39] V. F. R. Jones, Hecke algebra representations of braid groups and link polynomials, *Annals Math.* **126**, 335–388 (1987).

[40] R. C. King, Branching rules for classical Lie groups using tensor and spinor methods, *J. Phys. A: Math. Gen.* **8**, 429–449 (1975).

[41] R. C. King, Kronecker products of representations of exceptional Lie groups, *J. Phys. A: Math. Gen.* **14**, 77–83 (1981).

[42] R. C. King and A. H. A. Al-Qubanchi, Natural labelling schemes for simple roots and irreducible representations of exceptional Lie algebras, *J. Phys. A: Math. Gen.* **14**, 15–49 (1981).

[43] R. C. King and A. H. A. Al-Qubanchi, The Weyl groups and weight multiplicities of the exceptional groups, *J. Phys. A: Math. Gen.* **14**, 51–75 (1981).

[44] R. C. King and N. G. I. El-Sharkaway, Standard Young tableaux and weight multiplicities of the classical Lie groups, *J. Phys. A: Math. Gen.* **16**, 3153–3177 (1983).

[45] R. C. King, Luan Dehuai and B. G. Wybourne, Symmetrised powers of rotation group representations, *J. Phys. A: Math. Gen.* **14**, 2509–2538 (1981).

[46] R. C. King and B. G. Wybourne, Holomorphic discrete series and harmonic series unitary irreducible representations of non-compact Lie groups: $Sp(2n, R)$, $U(p, q)$, and $SO^*(2n)$, *J. Phys. A: Math. Gen.* **18**, 3113–3139 (1985).

[47] R. C. King and B. G. Wybourne, Characters of Hecke algebras $H_n(q)$ of type A_{n-1}, *J. Phys. A: Math. Gen.* **23**, L1193–L1197 (1991).

[48] R. C. King and B. G. Wybourne, Some noteworthy spin plethysms, *J. Phys. A:Math. Gen.* **15**, 1137–1141 (1982).

[49] R. C. King and B. G. Wybourne, Representations and traces of the Hecke algebras $H_n(q)$ of type A_{n-1}, *J. Math. Phys.* **33**, 4–14 (1992).

[50] R. C. King, B. G. Wybourne and M. Yang, Slinkies and the $S-$function

content of certain generating functions, *J. Phys. A: Math. Gen.* **22**, 4519–4535 (1989)

[51 R. C. King and B. G. Wybourne,The place of the adjoint representation in the Kronecker square of irreducible representations of simple Lie groups, *J. Phys. A:Math. Gen.* **29**, 5059-77 (1996).

[52] R. C. King and B. G. Wybourne, Kronecker powers of irreducible representations of $Sp(2n, R)$, *J. Phys. A:Math. Gen.* (Submitted 1998).

[53] D. E. Knuth, Permutations, matrices and generalised Young tableaux, *Pacific J. Math.* **34**, 709–727 (1970).

[54] K. Koike and I. Terada, Young-diagrammatic methods for the representation theory of the classical groups of type B_n, C_n and D_n, *J. Algebra* **107**, 466–511 (1987).

[55] A. Lascoux and P. Pragacz, $S-$function series, *J. Phys. A:Math. Gen.* **21**, 4105–4114 (1988).

[56] D. E. Littlewood, Polynomial concomitants and invariant matrices, *J. London Math. Soc.* **11**, 49–55 (1936).

[57] D. E. Littlewood, Invariant theory, tensors and group characters, *Phil. Transac. Roy. Soc.* (London) *A* **239**, 305–365 (1944).

[58] D. E. Littlewood, On invariant theory under restricted groups, *Phil. Transac. Roy. Soc.* (London) *A* **239**, 387–417 (1944).

[59] D. E. Littlewood, On the concomitants of spin tensors, *Proc. London Math. Soc. (2)* **49**, 307–327 (1947).

[60] D. E. Littlewood, *The Theory of Group Characters and Matrix Representations of Groups*, 2nd ed., Clarendon Press, Oxford, 1950.

[61] D. E. Littlewood, Modular representations of symmetric groups, *Proc. Roy. Soc.* (London) *A* **209**, 333-353 (1951).

[62] D. E. Littlewood, Invariant theory under orthogonal groups, *Proc. London Math. Soc. (2)* **50**, 349–379 (1948).

[63] D. E. Littlewood, Plethysm and the inner product of $S-$functions, *J. London Math. Soc.* **32**, 18 (1957).

[64] D. E. Littlewood, The inner plethysm of $S-$functions, *Can. J. Math.***10**, 1–16 (1958).

[65] D. E. Littlewood, Products and plethysms of characters with orthogonal, symplectic and symmetric groups, *Can. J. Math.***10**, 17 (1958).

[66] D. E. Littlewood, On certain symmetric functions, *Proc. London Math. Soc. (3)* **11**, 485–498 (1961).

[67] D. E. Littlewood, The Kronecker product of symmetric group representations, *J. London Math. Soc.* **31**, 89–93 (1956).

[68] D. E. Littlewood and A. R. Richardson, Group characters and algebra,

Phil. Transac. Roy. Soc. (London) *A* **233**, 99–141 (1934).

[69] Luan Dehuai and B. G. Wybourne, The symmetric group: branching rules, products and plethysms for spin representations, *J. Phys. A: Math. Gen.* **14**, 327–348 (1981).

[70] Luan Dehuai and B. G. Wybourne, The alternating group: branching rules, products and plethysms for ordinary and spin representations, *J. Phys. A: Math. Gen.* **14**, 1835–1848 (1981).

[71] I. G. Macdonald, *Symmetric Functions and Hall Polynomials*, Clarendon Press, Oxford, 1979.

[72] P. A. MacMahon, Symmetric functions and the theory of distributions, *Proc. London Math. Soc.* **19**, 220–256 (1889).

[73] A. Messiah, *Quantum Mechanics*, Vol. 2, North-Holland, Amsterdam, 1961, Appendix D.

[74] A. O. Morris, The spin representation of the symmetric group, *Proc. London Math. Soc. (3)* **12**, 55–76 (1962).

[75] A. O. Morris, The spin representation of the symmetric group, *Can. J. Math.* **17**, 543–549 (1965).

[76] A. O. Morris, A survey on Hall-Littlewood functions and their applications to representation theory, *in Combinatoire et représentation du groupe symétrique* (D. Foata, Ed.). Lecture Notes in Math. **579**, Springer-Verlag, Berlin, 1978.

[77] P. A. Morris, Applications of graph theory to $S-$function theory, *J. London Math. Soc.* **8**, 63–72 (1974).

[78] P. A. Morris and G. H. J. van Rees, On computing plethysms of Schur functions, *MATCH* **3**, 51–66 (1977).

[79] I. Morrison, P. W. Pieruschka and B. G. Wybourne, The interacting Boson model with the exceptional groups G_2 and E_6, *J. Math. Phys.* **32**, 356–372 (1991).

[80] F. D. Murnaghan, *The theory of group representations*, Johns Hopkins University Press, Baltimore, Md., 1938.

[81] F. D. Murnaghan, On the representations of the symmetric group, *Amer. J. Math.* **59**, 437–488 (1937).

[82] F. D. Murnaghan, The characters of the symmetric group, *Anais Acad. Brasil Ci.* **23**, 141–154 (1951).

[83] F. D. Murnaghan, A generalisation of Hermite's law of reciprocity, *Anais Acad. Brasil Ci.* **23**, 347–368 (1951).

[84] F. D. Murnaghan, On the analysis of representations of the linear group, *Proc. Natl. Acad. Sci. U.S.A.* **37**, 51–55 (1951).

[85] F. D. Murnaghan, On the analysis of $\{m\} \otimes \{1^k\}$ and $\{m\} \otimes \{k\}$, *Proc.*

Natl. Acad. Sci. U.S.A. **41**, 721–723 (1954).

[86] F. D. Murnaghan, *The unitary and rotation groups: Lectures in Applied Mathematics*, Spartan Books, Washington D.C. 1962.

[87] M. L. Nazarov, An orthogonal basis of irreducible projective representations of the symmetric group, *Functional Anal. Appl.* **22**, 66–68 (1988).

[88] M. L. Nazarov, Young's orthogonal form of irreducible projective representations of the symmetric group, *J. London Math. Soc. (2)* **42**, 437–451 (1990).

[89] M. J. Newell, On the representations of the orthogonal and symplectic groups, *Proc. Roy. Irish Acad.* **54A**, 143–152 (1951).

[90] M. J. Newell, Modification rules for orthogonal and symplectic groups, *Proc. Roy. Irish Acad.* **54A**, 153–163 (1951).

[91] M. J. Newell, A theorem on the plethysm of $S-$functions, *Quart. J. Math. Oxford Ser.* **II 2**, 161–166 (1951).

[92] J. J. C. Nimmo, Wronskian determinants, the KP hierachy and supersymmetric polynomials, *J. Phys. A: Math. Gen.* **22**, 3212–3221 (1989).

[93] J. J. C. Nimmo, Hall-Littlewood symmetric functions and the BKP equation, *J. Phys. A: Math. Gen.* **23**, 751–760 (1990).

[94] S. P. O. Plunkett, Plethysm of $S-$functions, *Can. J. Math.* **24**, 541–52 (1972).

[95] R. C. Read, The use of $S-$functions in combinatorial analysis, *Can. J. Math.* **20**, 808–841 (1968).

[96] J. B. Remmel, A formula for the Kronecker products of Schur functions of hook shapes, *J. Algebra* **120**, 100–118 (1989).

[97] J. B. Remmel, On Kronecker products of Schur functions of two row shapes,

[98] J. B. Remmel and R. Whitney, Multiplying Schur functions, *J. Algorithms* **5**, 471–487 (1984).

[99] G. de B. Robinson, *Representation theory of the symmetric group*, Edinburgh University Press, Edinburgh, 1961.

[100] D. J. Rowe, B. G. Wybourne and P. H. Butler, Unitary representations, branching rules and matrix elements for the non-compact symplectic groups, *J. Phys. A: Math. Gen.* **18**, 939–953 (1985).

[101] D. E. Rutherford, *Substitutional analysis*, Edinburgh University Press, Edinburgh, 1948.

[102] B. E. Sagan, Shifted tableaux, Schur $Q-$functions and a conjecture of R. Stanley, *J. Combin. Theory Ser. A* **45**, 62–103 (1987).

[103] B. E. Sagan, *The symmetric group*, Wadsworth & Brooks/Cole mathematics series, Pacific Grove, Calif. (1991).

[104] M. A. Salam and B. G. Wybourne, $Q-$functions and $O_n \to S_n$ branching rules for ordinary and spin irreps, *J. Phys. A: Math. Gen.* **22**, 3771–3778 (1989).

[105] M. A. Salam and B. G. Wybourne, Shifted tableaux, Schur's Q functions, and Kronecker products of S_n spin irreps, *J. Math. Phys.* **31**, 1310–1314 (1990).

[106] M. A. Salam and B. G. Wybourne, The $q-$deformation of symmetric functions and the symmetric group, *J. Phys. A: Math. Gen.* **24**, L317-L321 (1991).

[107] M. A. Salam and B. G. Wybourne, Vertex operators and the symmetric functions, *J. Phys. A: Math. Gen.* **25** , 2297–2310 (1992).

[108] T. Scharf. J-Y Thibon and B. G. Wybourne, Reduced notation, inner plethysms and the symmetric group, *J. Phys. A: Math. Gen.* **26** , 7461–7478 (1993).

[109] T. Scharf. J-Y Thibon and B. G. Wybourne, Powers of the Vandermonde determinant and the quantum Hall effect, *J. Phys. A: Math. Gen.* **27** , 4211–4219 (1994).

[110] T. Scharf. J-Y Thibon and B. G. Wybourne, Generating functions for stable branching coefficients of U(n)-¿S(n), O(n)-¿S(n) and O(n-1)-¿S(n), *J. Phys. A:Math. Gen.* **30**, 6963-75 (1997).

[111] I. Schur, Über eine klasse von matrizen, die sich einer gegeben matrix zuordnen lassen, *Dissertation, Berlin* (1901).

[112] I. Schur, Über die darstellung der symmetrischen und der alternierenden gruppe durch gebrochene lineare substitutionen, *J. Reine Angew. Math.* **139**, 155–250 (1911).

[113] P. R. Smith and B. G. Wybourne, Selection rules and the decomposition of the Kronecker squares of irreducible representations, *J. Math. Phys.* **8**, 2434–2440 (1967).

[114] P. R. Smith and B. G. Wybourne, Plethysm and the theory of complex spectra, *J. Math. Phys.* **9**, 1040–1051 (1968).

[115] R. P. Stanley, Theory and application of plane partitions I. II. *Stud. Appl. Math.* **50**, 167–188, 259–279 (1971).

[116] J. R. Stembridge, Shifted tableaux and the projective representations of symmetric groups, *Adv. Math.* **74**, 87–134 (1989).

[117] M. Stone, Vertex operators in the quantum Hall effect, *Int. J. Mod. Phys. B* **5**, 509–527 (1991).

[118] S. Sundaram, Orthogonal tableaux and an insertion scheme for $SO(2n+1)$, *J. Combin. Theory (A)* **53**, 239–256 (1990).

[119] Jean-Yves Thibon, Frederic Toumazet and Brian G Wybourne, Products and plethysms for the fundamental harmonic series representations of

U(p,q), *J. Phys. A:Math. Gen.* **30**, 4851-6 (1997).

[120] Jean-Yves Thibon, Frederic Toumazet and Brian G Wybourne, Symmetrised squares and cubes of the fundamental unirreps of Sp(2n,R), *J. Phys. A:Math. Gen.***31**, (Feb.) (1998).

[121] R. M. Thrall, On symmetrized Kronecker powers and the structure of the free Lie ring, *Amer. J. Math.* **64**, 371-388 (1942).

[122] J. A. Todd, A note on the algebra of $S-$functions, *Proc. Cambridge Philos. Soc.* **45**, 328–334 (1949).

[123] V. Vanagas, *Algebraic methods in nuclear theory*, (in Russian) Moscow 1971.

[124] V. Vanagas, *Algebraic foundations of the microscopic nuclear theory*, (in Russian) Nauka, Moscow 1988.

[125] H. Weyl, *The Classical Groups*, Princeton University Press, Princeton, 1939.

[126] B. G. Wybourne and M. J. Bowick, Basic properties of the exceptional Lie groups, *Aust. J. Phys.* **33**, 259–286 (1977).

[1] B. G. Wybourne, *Symmetry Principles and Atomic Spectroscopy*, Wiley, New York, 1970.

[127] B. G. Wybourne, Symmetry principles in atomic spectroscopy, *J. de Phys.* **31** *Colloque C* **4**, C4–C33 (1970).

[128] B. G. Wybourne, $SU_6 \times SU_3^c$ scalars in E_7 irreps, *J. Math. Phys.* **19**, 529–530 (1978).

[129] B. G. Wybourne, Aspects of the exceptional group E_8, *Aust. J. Phys.* **32**, 417–426 (1979).

[130] B. G. Wybourne, Enumeration of group invariant quartic polynomials in Higgs scalar fields, *Aust. J. Phys.* **33**, 941–949 (1980).

[131] B. G. Wybourne, The Littlewood-Richardson rule – The cornerstone for computing group properties, *Banach Center Publications* **26** (2) 475–482 (1990).

[132] B. G. Wybourne, Branching rules for $E_8 \downarrow SO_{16}$, *J. Phys. A: Math. Gen.* **17**, 1397–1402 (1989).

[133] B. G. Wybourne, Characters, dimensions and branching rules for covariant irreps of $U(N/M)$, *J. Phys. A: Math. Gen.* **17**, 1573–1578 (1989).

[134] B. G. Wybourne, Hermite's reciprocity law and the angular momentum states of equivalent particle configurations, *J. Math. Phys.* **10**, 467–471 (1969).

[135] B. G. Wybourne, Coefficients of fractional parentage and LL-Coupling, *J. de Phys.* **30**, 35–38 (1969).

[136] B. G. Wybourne, Symmetry classification of two-particle operators in atomic spectroscopy, *J. de Phys.* **30**, 39–46 (1969).

[137] B. G. Wybourne, *Classical Groups for Physicists*, J Wiley and Sons, New York 1973.

[138] B. G. Wybourne, The 'Gruppen Pest' yesterday, today and tomorrow,*Int. J. Quantum Chem. Symp. No. 7,* 35–43 (1973).

[139] B. G. Wybourne, Lie algebras in quantum chemistry:symmetrised orbitals, *Int. J. Quantum Chem.,* **7**, 1117–1137 (1973).

[140] B. G. Wybourne, Rank dependency of group properties, *Physica* **114A**, 350–360 (1982).

[141] B. G. Wybourne, Rank dependency of weight multiplicities for simple Lie groups, *J. Phys. A:Math. Gen.* **15**, 2687–2697 (1982).

[142] B. G. Wybourne, Group theory and microcomputers, *Asia-Pacific Physics News,* **2**, 9–10 (1987).

[143] B. G. Wybourne, Symmetrised powers of the fundamental irrep of E_6, *J. Math. Phys.* **31**, 2345 (1990).

[144] B. G. Wybourne, Group structures for the interacting Boson approximation in nuclei, pp , *Symmetries in Science* **V**, Plenum Press, New York 1990.

[145] B. G. Wybourne, A role for the exceptional groups E_6 and G_2 in the IBM nuclear models, *XVIII Int. Coll. Group Theor. Meth. in Physics Moscow. Lecture Notes in Physics* **382** 140–144 (1991).

[146] B. G. Wybourne and P. H. Butler, The configuration $(d+s)^N$ and the group R_6, *J. de Phys.* **30**, 181–186 (1969).

[147] B. G. Wybourne, The Eight-fold way of the electronic $f-$shell, *J. Phys. B: Atom. Mol. Opt. Phys.* **25**, 1683–1696 (1992).

[148] B. G. Wybourne, The representation space of the nuclear symplectic $Sp(6,R)$ shell-model, *J. Phys. A:Math. Gen.* **25**, 4389–98 (1992).

[149] B. G. Wybourne, Applications of S-functions to the quantum Hall effect and quantum dots, *Rept. Math. Phys.* **34**, 9–16 (1994).

[150] B. G. Wybourne, SCHUR, An interactive Program for Calculating Properties of Lie Groups and Symmetric Functions, *Euromath Bulletin* **2**, 145-159 (1996).

[151] Brian G. Wybourne, Norbert Flocke and Jacek Karwowski, Characters of Two-Row Representations of the Symmetric Group *Int. J. Quantum Chem.* **62**, 261-4 (1997).

[152] Brian G. Wybourne, Guesses - Hunches - Formulae - Discoveries *Adv. Quantum Chem.* **28**, 312-317 (1997).

[153] B. G. Wybourne, Plethysm and the Non-Compact Groups Sp(2n,R),

Seminaire Lotharingien de Combinatoire **B36g**, 5pp (1996).

[154] B. G. Wybourne, Plethysm in Physics and Chemistry Applications, *Proc. Fourth Intnl School on Theoretical Physics*, (1996) (World Scientific).

[155] B. G. Wybourne, Recent extensions and developments in SCHUR, *Proc. Fourth Intnl School on Theoretical Physics*, (1996) (World Scientific).

[156] M. Yang and B. G. Wybourne, Extended Poincaré supersymmetry, rotation groups and branching rules, *J. Phys. A: Math. Gen.* **19**, 2003–2017 (1986).

[157] M. Yang and B. G. Wybourne, New *s*-function series and non-compact Lie groups, *J. Phys. A: Math. Gen.* **19**, 3513–3525 (1986).

[158] M. Yang and B. G. Wybourne, Squares of S-functions of special shapes, *J. Phys. A: Math. Gen.* **28**, 7011–7017 (1995).

[159] A. Young, *The collected papers of Alfred Young 1873–1940*, University of Toronto Press, Toronto, 1978.

Additional information is available at the WEB page
http://www.phys.uni.torun.pl/∼bgw
A number of relevant recent papers and papers published or submitted since the publication of this manual are available from that site in down-loadble postscript format.

People in general do not willing read if they can having anything else to amuse them
— Samuel Johnson 1783

A

An Appendix in which every command in SCHUR is explained

■ Introduction

This appendix seeks to give a brief description of each SCHUR command. The commands are listed alphabetically. The format for explaining a command is as follows:

COMmand

Format:

Modes:

Description:

Examples:

The boldface letters of the command are the minimal letters required by SCHUR. To respond correctly, they may be entered as either lower case or upper case letters.

Format: gives a general format for entry of the command appropriate to the relevant modes. EXPR in a format line is any expression appropriate to the modes, e.g. it may be a list of Sfns, reps or Dreps, or even a set of nested commands.

Modes: lists the modes of SCHUR in which the command may be used (SFN, REP, DPM, BRM).

Description: gives a brief description of the SCHUR command.

Examples: here one or more simple examples of the command are given. User input is indicated by an arrow ⇒. All other entries represent responses from SCHUR. Throughout the examples the abbreviated forms of the commands are used. Occasionally, optional brackets are used to clarify heavily nested sequences of commands.

In order to shorten the length of commands a number of abbreviations have been used. These are given in the table below.

ch = change,	conv = convert,	e = elementary
fn = function	grN = group number,	h = homogeneous
i = inner,	int = integer	m = monomial
o = outer,	op = operator,	p = power sum
p = pfn,	q = qfn ,	rd = reduce
rk = rank,	rm = remove	rp = replace
s = sfn,	sb = setboolean,	std = standardise
wr = write,	wt = weight	

A complete list of all the commands in SCHUR is given in Table A.1.

Table A.1 All the commands in SCHUR.

ABSoluteValue	ADD	ALLskewSfn
ASSOCiate	ATtachPartitionToSfn	AUtoOrIsoMorphism
BRanch	BRMode	
CANcelDatFile	CASIMIRGnthTrace	CASimirNthordertrace
CH_CoeffsToOneForSfns	CH_LabelForOn	CH_PhaseOfSfns
CH_SpinIndex	CH_UoneReps	CLASS
COLumns	COMPare	COMPLement
CONJADD	CONJugateSfnList	CONSPLIT
CONTENT	CONTractGroups	CONTRAGredientRep
CONV_D_TO_Rep	CONV_D_TO_Sfn	CONV_R_TO_Sfn
CONV_S_TO_Rep	COUNTCoeffsInList	COUNTTermsInList
COVariant		
DEAD	DIMension	DPMode
D_TO_Plabel	DYNKINIndex	
END	ENTerVar	EQualSfnList
E_TO_FSymmFn	E_TO_HSymmFn	E_TO_MSymmFn
E_TO_SSymmFn	EXITmode	EXPandSfnList
FFPROD	FIRSTPart	FN
FPROD	FRACAHnotation	FROB
F_TO_ESymmFn	F_TO_HSymmFn	F_TO_MSymmFn
F_TO_SSymmFn	FUSion	
GENERIC	GENprod	GRoup
GWT		
HALLpolynomialProduct	HCLASS	HEAPstatus
HECKE	HELP	HSTD
HSTDList	H_TO_ESymmFn	H_TO_FSymmFn
H_TO_MSymmFn	H_TO_SSymmFn	
INDEXsequence	INSertPartitionIntoSfn	INTegerDivideCoeffs
INVerseseries	I_PLethysmRd	I_QfnProduct
I_sfnProduct	I_SFNQfnProduct	
KINSert	KMatrix	Kostka
LABel	LASTresult	LATticetest
LENgthOfPartitionsSelect	LICENSE	LINES

LOadFile	LOGfile	LSEQuence
MACMixedSeries	MACseries	MAKEwtOfSfnToN
MAXCoeffInList	MIXedTensorReps	M_TImesSfnProduct
M_TO_ESymmFn	M_TO_FSymmFn	M_TO_HSymmFn
M_TO_SSymmFn	MUlt_CoeffsByAnInt	MULT_List
MULT_Ntimes	MULT_PartsByAnInt	MULT_SelectInList
MULT_SPlitIntoTwoLists	MYlistOfSfns	
NLambda	NSKew	NSTDise
ONSCalar	O_PfnProduct	O_QfnProduct
O_Restrict	O_sfnProduct	
PARITYsequence	PAUSE	PLethysm
PLG	ProductKronecker	PROPertyOfRepList
P_TO_Dlabel	P_TO_SSymmFn	
QEXPandSpecialSeries	QQEXpandSpecialSeries	QQSEries
QSAME	QSERies	Q_TO_SsymmFn
RACAHnotation	RAISEInverseOp	RAISEop
RD_I_QfnProduct	RD_I_sfnProduct	RD_RAISEInverseOp
RD_RaiseOp	READFnFromDisk	REMark
REPmode	RETurn	RIEMANNList
RIEMANNPlethList	RIEMANNScalarsOrderN	RM_EVENPARTS
RM_EVENRkSfnsOnly	RM_EVENWtInList	RM_FirstPartOfSfn
RM_Group	RM_NMP	RM_ODDPARTS
RM_ODDRkSfnsOnly	RM_ODDWtInList	RM_PartitionFromSfn
RM_PARTSequalN	RM_RepeatedPartsSfns	RM_SOnEvenLabels
RM_UoneWtOverMax	RP_FirstPartBySpin	RP_RepOrSfnByWt
RP_SfnCoeffByInt	RSAMEwtSfnList	RULE
RVar		
SAMEwtSfns	SAVEsetVar	SB_Bell
SB_CONjecture	SB_CUT	SB_Digits
SB_DIMension	SB_LISToutput	SB_More
SB_POWerNotation	SB_PROGress	SB_Qfn
SB_RD_notation	SB_REVerseOrder	SB_TexOutPut
SCALARInner	SCHAR	SERIESTErmsThatSkew
SERiesToIntWt	SETFnVar	SETLIMit
SET_PWT	SETRVar	SETSVar
SETVarInDPmode	SFNmode	SIGNSEQuence
SK_Pfn	SK_Qfn	SK_sfn

SMON	SNchar	SPIN
SPLitIntoSpinAndTensor	SPONModify	SPRCH
SPREXtend	SPSTAR	SQuares
STARequivalent	STATusOfSchur	STD
STD_OneDprep	STD_Qfn	S_TO_ESymmFn
S_TO_FSymmFn	S_TO_HSymmFn	S_TO_MSymmFn
STOP	S_TO_PsymmFn	S_TO_QsymmFn
SUBtract	SUM	SUMSQuares
SUPpressOutputToScreen	SVar	SWAPgroups

TABleOfBranchingRules

UONEAddInteger	UONEDivInteger	UONETrace

VarForDpreps	VMult

WHATGroup	WITH	WRFNTODisk
WRfnToScreen	WSEQuence	WTofRepOrSfnSelect

YHooklengths	YOungDiagrams	YShapeSelect

Zero

■ **ABS**olutevalue
Format: Abs EXPR
Modes: SFN
Description: Makes all coefficients positive in EXPR.
Example: SFN>
⇒ `abs 3.21^3-2.321+42-1^5`
`{ 42} +2{ 321} +3{ 21^3 } +{ 1^5 }`
SFN>

■ **ADD**
Format: ADD EXPR1, EXPR2
Modes: DPM, REP, SFN
Description: Adds EXPR1 to EXPR2 to produce the resultant.
Example: SFN>
⇒ `add 3.21^3 -2.321 +42 -1^5, -3.21^3 -2.321 +42 -1^5`
`2{ 42} -4{ 321} -2{ 1^5 }`
SFN>

■ **ALL**skewsfn
Format: ALL EXPR
Modes: SFN
Description: Generates all Sfns which will skew with EXPR. There is no overcounting so if one Sfn skews with several members of EXPR it only appears once.
Example: SFN>
⇒ `all 3.321+2.21+42`
`{ 42} +{ 41} +{ 4} +{ 321} +{ 32} +{ 31^2 } +{ 31} +{ 3} +{ 2^2 1}`
`+{ 2^2 } +{ 21^2 } +{ 21} +{ 2} +{ 1^3 } +{ 1^2 } +{ 1} +{ 0}`
SFN>

■ **ASSOC**iate
Format: ASSOC EXPR (REP)
ASSOC GRNO EXPR (DPM)
Modes: REP, DPM
Description: ASSOCiate replaces every irrep in EXPR by its associate. It is assumed that the group has been set as Sp(2n,R) or O(n) - other settings will give an error message. See also SPONModify. Notice that some representations are self-associate.
Example: REP>
⇒ `gr spr8`
`Group is Sp(8,R)`
REP>
⇒ `assoc 3.2;53+2.2;511+2;5`
`3<2(53)> +<2(5)># +2<2(5)>`
REP>

```
⇒     sponm last
          3<2(53)> +3<2(5)>
          REP>
```

■ **AT**tachPartitionToSfn

Format: AT PARTITION, EXPR
Modes: SFN
Description: Adds the parts of PARTITION to the corresponding parts of each term in EXPR, preserving their multiplicities.
Example:

```
SFN>
⇒     at321, 6 +2.321 -3.21^5
      { 921} +2{ 642} -3{ 5321^3 }
          SFN>
```

■ **AU**toOrIsoMorphisms

Format: AU gr grN TARGET_GR,EXPR(DPM)
AU TARGET_GR,EXPR (REP)
Modes: DPM, REP
Description: Performs the automorphism or isomorphism current group -> TARGET_GR on EXPR. In particular for SO(8) -> SO(8), SO(6) -> SU(4), SO(5) -> Sp(4), SU(2) -> SO(3) -> Sp(2) and SO(2) -> U(1).
Example:

```
DP>
⇒     gr2so6sp4
      Groups are SO(6) * Sp(4)
      DP>
⇒     au gr1su4,[s1^3+ * 21]
      Groups are SU(4) * Sp(4)
      { 3}<21>
      DP>
⇒     REP
      REP mode
      Group is SU(4)
      REP>
⇒     gr so8
      Group is SO(8)
      REP>
⇒     au so8,1
      Group is SO(8)
      [s;0]-
      REP>
⇒     au so8,last
      Group is SO(8)
      [s;0]+
      REP>
```

```
⇒     au so8,last
      Group is SO(8)
      [1]
      REP>
```

Notes:

```
 Note the compulsory comma (,) after the name of the tar-
get group.
The isomorphisms SO(4) -> SU(2) * SU(2) -> SO(3) * SO(3)
-> Sp(2) * Sp(2)
result in a change in the number of groups set. The ac-
tion of such
isomorphisms can be evaluated in the DPMode using branch-
ing rule 15
for SO(4) -> SU(2) * SU(2).
```

■ **BR**anch

Format: BR RULE_NUMBERS gr grN EXPR

Modes: DPM

Description: Executes the branching rule defined by 'RULE_NUMBERS' on the group 'grN' of EXPR. Rule numbers are entered as rule number n, m, p, q taken from Table A.2.
When a rule is executed on grN the resulting groups are shifted to the end of the sequence of DPrep groups.

Example:

```
DP>
⇒     gr g2
      Group is G(2)
      DP>
⇒     br41gr1[52+2.31-3.1]
      Group is SO(3)
              [13 ] +[12 ] +[11 ] +[10 ] +2[9] +4[8] +4[7] +2[6]
      +4[5]
              +4[4] +[3] +3[2] +[1]
⇒     br41gr1[92]
      Group is SO(3)
            [25 ] +[24 ] +2[23 ] +2[22 ] +4[21 ] +5[20 ] +6[19
      ] +7[18 ]
             +9[17 ] +10[16 ] +12[15 ] +12[14 ] +14[13 ] +14[12
      ] +15[11 ]
             +15[10 ] +15[9] +14[8] +14[7] +12[6] +12[5] +9[4]
      +8[3]
             +5[2] +4[1] +[0]
      DP>
⇒     gr spr6
      Group is Sp(6,R)
      DP>
⇒     br36,6gr1[s0;1-s0;0]
```

```
                Group is U(3)
               -{ s;10 } -{ s;8} -{ s;6} -{ s;4} -{ s;2} -{ s;0}
DP>
```

Notes:

```
 01/26/2006 : all branching rules are not fully verified
```

■ **BRM**ode

Format: BRM
Modes: DPM
Description: Switches from the DPMode to the BRMode. Going into the BR-Mode
allows the user to pick a branching rule and to evaluate it for a number of representations taken one at a time, without multiplicity, without having to reset the rule. However, the result of a calculation cannot be processed further.
When this mode is called it first asks you to "enter branching & rule numbers". This should be done in the form: rule number,m,n,p,q
where m,n,p,q are any necessary group-subgroup parameters.
To change the rule enter STOP. To exit to the DPMode enter EXITmode. See also the helpfile TABleOfBranchingRules.
Example: DP>

```
⇒   brm
                Branch Mode
                enter branching & rule numbers>
⇒   1,6
                U(6) to O(6)
                BRM>
⇒   421
                [421] +[41] +[32] +[31^2 ] +[3] +[2^2 1] +2[21]
    +[1]
                BRM>
⇒   stop
                enter branching & rule numbers>
⇒   41
                G(2) to SO(3)
                BRM>
⇒   52
                [13 ] +[12 ] +[11 ] +[10 ] +2[9] +2[8] +2[7]
    +2[6] +2[5]
          +2[4] +2[3] +[2] +[1]
          BRM>
⇒   stop
          enter branching & rule numbers>
⇒   27,3,5
```

```
                SO(8) to SO(3) * SO(5)
                  BRM>
⇒       32
                        [3][2] +[3][1] +[3][0] +[2][3] +[2][21] +[2][2]
                +[2][1^2 ] +2[2][1] +[2][0] +[1][31] +[1][2^2 ] +[1][21]
                +2[1][2] +[1][1^2 ] +[1][1] +[1][0] +[0][32] +[0][3]
                +[0][21] +[0][1]
                BRM>
⇒       exit
                DPrep Mode (with function)
                        DP>
```

■ **CAN**celDatFile

Format: CAN INTEGER
Modes: DPM, REP, BRM
Description: Used to delete the *.DAT file called in connection with certain branching rules and products. Assists in the conservation of heap space.
Example:

```
DP>
⇒       gr e8
                        Group is E(8)
                  DP>
⇒       br50gr1[21^7]
                Group is SO(16)
                        [1^2] + [s;0]+
                        DP>
⇒       can 50
                        DP>
```

■ **CASIMIRG**eneralNthTrace

Format: CasimirG INTEGER REP1, REP2
Modes: REP
Description: Calculates the trace of the INTEGER-th order Casimir invariant for rep1 with respect to rep2. The command is available for all the compact semisimple Lie groups.
Example:

```
REP>
⇒       gr e8
                        Group is E(8)
                        REP>
⇒       casimirg 2 21^7,21
                        60**1*trace 2th order Casimir invariant is
        96
                        REP>
⇒       casimirg 3 21,21
                        60**2*trace 3th order Casimir invariant is
        -36000
```

```
                    REP>
```

Notes:

```
 The output actually gives the Casimir invariant for REP2
with respect
to REP1 since setting REP1=adjoint gives the same results
as CASIMIRN.
```

■ **CA**simirNthOrderTrace

Format: CA INTEGER EXPR
Modes: REP
Description: Computes the trace of the INTEGER-th order Casimir invariant with
respect to the adjoint representation of the set group. This command is available for all the compact semisimple Lie groups.
Example:

```
REP>
⇒   gr sp6
                    Group is Sp(6)
                    REP>
⇒   casimirn 2 2
                    16**1*trace 2th order Casimir invariant is
16
                    REP>
⇒   casimirn 3 2
                    16**2*trace 3th order Casimir invariant is
-64
                    REP>
```

■ **C**h_CoeffsToOneForSfns

Format: Ch_C EXPR
Modes: SFN
Description: Reduces all multiplicities in EXPR to + 1.
Example:

```
SFN>
⇒   ch_c 2.32+5.21-7.2
                    { 32} + { 21} + { 2}
                    SFN>
```

■ **CH_L**abelForOn

Format: Ch_L gr grN EXPR (DPM)
Ch_L EXPR (REP)
Modes: DPM, REP
Description: Used for swapping the (+) and (-) labels that arise in EXPR for the groups SO(2n). This operation corresponds to an SO(2n)->SO(2n)
automorphism that is NOT covered by the command AUtoOrIsoMorphisms.
Example:

```
DP>
⇒   gr so8
                    Group is SO(8)
```

```
                    DP>
⇒     [s1+] + [s0-]
                    [s;1]+ + [s;0]-
                    DP>
⇒     ch_l gr1last
                    [s;1]- + [s;0]+
                    DP>
⇒     rep
                    REP mode
                    Group is SO(8)
                    REP>
⇒     s1+ + s0- + 21^3- + 1^4+ + 1^3 + 2^4
                    [s;1]+ +[s;0]- +[21^3 ]- +[1^4 ]+ +[1^3 ] +[2^4
      ]
           REP>
⇒     ch_l last
           [2^4 ] +[21^3 ]+ +[s;1]- +[1^4 ]- +[1^3 ] +[s;0]+
                    REP>
```

■ **CH_P**haseOfSfns

Format: Ch_P EXPR
Modes: SFN
Description: Multiplies each coefficient in EXPR by (-1)^omega where omega is the weight of the associated partition in EXPR.
Example: `SFN>`

```
⇒     ch_p 321-2.21+3.2
           { 321} +2{ 21} +3{ 2}
                    SFN>
```

■ **CH_S**pinIndex

Format: Ch_S gr grN EXPR (DPM)
Ch_S EXPR (REP)
Modes: DPM, REP
Description: Ch_S deletes the spin index s in a rep [s;a] to give [a], and inserts a spin index s in any tensor rep [a] to give [s;a]. This may be useful in setting up functions, however please note that this command deletes any + or - labels.
Example: `DP>`

```
⇒     gr2so8su4
                    Groups are SO(8) * SU(4)
                    DP>
⇒     ch_s gr1 [s31+ * 21] + [21 * 321]
                    [31]{ 21} +[s;21]{ 321}
                    DP>
⇒     rep
```

```
                REP mode
                Group is SO(8)
                        REP>
    ⇒     gr so7
                        Group is SO(7)
                        REP>
    ⇒     ch_s 321+s31^2
                        [s;321] +[31^2]
                REP>
```

■ **CH_U**oneReps

Format: Ch_U INTEGER gr grN EXPR (DPM)
Ch_U INTEGER EXPR (REP)

Modes: DPM, REP

Description: This operation multiplies the weights of all U(1) reps in EXPR by INTEGER for the group specified by the grN.

Example:

```
DP>
    ⇒     gr2u1su5
                        Groups are U(1) * SU(5)
                        DP>
    ⇒     ch_u 3 gr1 [6 * 21] + [~2 * 21^2] + [~6 * 2^3 1]
                        { 18 }{ 21} + { -6}{ 21^2} + { -18 }{ 2^3 1}
                        DP>
```

■ **CLASS**

Format: CLASS EXPR

Modes: SFN

Description: EXPR should be a list of partitions without multiplicity, and the output of CLASS EXPR is the same list but now with multiplicities equal to the number of elements in the corresponding class of the symmetric group of the appropriate weight. For partitions of weight N>13 overflow can be expected. To obtain results for N > 13 you may determine the number of elements in a single class using the command HCLass as shown below.

Example:

```
SFN>
    ⇒     class 5 + 41 + 32 + 31^2 + 2^21 + 21^3 + 1^5
                 24{ 5} + 30{ 41} + 20{ 32} + 20{ 31^2 } + 15{ 2^2
          1} + 10{ 21^3 } + { 1^5 }
                        SFN>
    ⇒     class 32+42
                90{ 42} +20{ 32}
                SFN>
    ⇒     hclass!15 5
                        Number of elements in the class =
                        32438693442355200
```

```
SFN>
```

■ **COL**umns

Format: COL INTEGER
Modes: DPM, REP, SFN
Description: Used to set tables in TeX. INTEGER is the desired number of columns. The default value has been set at 5. See also SB_Tex and LINES.

■ **COMP**are

Format: Comp EXPR1,EXPR2
Modes: REP, SFN
Description: Compares EXPR1 and EXPR2, and creates a new EXPR in which the multiplicities are the products of the corresponding multiplicities in EXPR1 and EXPR2.
Example:

```
SFN>
```

⇒

```
comp 42 +31, 5 +4.42 +5.31 +2^2
4{ 42} + 5{ 31}
SFN>
```

⇒

```
comp 2.42 -3.31, 5 +4.42 +5.31 +2.2^2
8{ 42} -15{ 31}
SFN>
```

■ **COMPL**ement

Format: COMPLement INTEGER1, INTEGER2, EXPR
Modes: SFN
Description: INTEGER1 should be greater than or equal to greatest number of parts of any partition in EXPR, while INTEGER2 should be greater than or equal to the largest part of any partition in EXPR. (Otherwise an error message appears). Each partition in EXPR is then replaced by its complement with respect to INTEGER1^INTEGER2. Thus, given that the above conditions on
these integers are satisfied, then COMPL M,N,EXPR is equivalent to SKew M^N,EXPR, complete with multiplicities.
Example:

```
SFN>
```

⇒

```
compl 6, 4, 321 -2.2+31^5
-2{ 4^5 2} +{ 4^3 321} +{ 3^5 1}
SFN>
```

■ **CONJADD**

Format: CONJADD EXPR
Modes: SFN
Description: Acts on each term in EXPR by replacing each partition a by its conjugate a' if and only if a'>a with respect to the usual reverse lexicographic ordering, whereby for example

```
          41^2 > 31^3
Example:  SFN>
    ⇒     outer 21,21
                        { 42} + { 41^2 } + { 3^2 } + 2{ 321} + { 31^3
          } + { 2^3 }   + { 2^2 1^2 }
                        SFN>
    ⇒     conjadd last
                        2{ 42} + 2{ 41^2 } + 2{ 3^2 } + 2{ 321}
                        SFN>
```

■ **CONJ**ugateSfnList

Format: Conj EXPR
Modes: SFN
Description: Takes the conjugate of the S-functions in EXPR.
Example:

```
SFN>
    ⇒     conj 42+31
                        { 2^21^2} + { 21^2}
                        SFN>
    ⇒     conj last
                        { 42} + { 31}
                        SFN>
```

■ **CONSPLIT**

Format: CONSPLIT EXPR
Modes: REP
Description: This command only works with groups SU(n). It interprets EXPR as a list of irreps of SU(n) and splits it into three sets of terms. Self-contragredient irreps are stored as RVAR19. In the case of contragredient pairs of irreps the lowest weight partner is stored as RVAR18 and the higher weight partner as RVAR20.
Example:

```
REP>
    ⇒     gr su6
                         Group is SU(6)
                         REP>
    ⇒     p21^4,21^4
                         { 42^4 } + { 3^2 2^3 } + { 31^3 } + { 2^2
          1^2 } + 2{ 21^4 } + { 0}
                         REP>
    ⇒     consplit last
                         REP>
    ⇒     rv18
                         { 31^3 }
                         REP>
    ⇒     rv19
                         { 42^4 } + { 2^2 1^2 } + 2{ 21^4 } + { 0}
```

```
                    REP>
⇒     rv20
                    { 3^2 2^3 }
                    REP>
```

■ **CONTENT**

Format: CONTENT INTEGER EXPR
Modes: SFN
Description: If x is a cell in a Young frame with coordinates x = (i,j) then the 'content' of the cell is c(x) = j - i. If INTEGER = 0 and EXPR is a list of S-functions then the content of every cell of every Young frame is displayed. If INTEGER = n then the display is of the integers n + c(x).
Example:

```
SFN>
⇒     content 0 4321
      +--+--+--+--+
      : 0: 1: 2: 3:
      +--+--+--+--+
      :-1: 0: 1:
      +--+--+--+
      :-2:-1:
      +--+--+
      :-3:
      +--+
      SFN>
⇒     content 2 o21,21
                                  2.
      +-+-+-+-+ +-+-+-+-+ +-+-+-+ +-+-+-+ +--+--+--+ +-+-+ +-
      -+--+
      :2:3:4:5: :2:3:4:5: :2:3:4: :2:3:4: : 2: 3: 4: :2:3: : 2:
      3:
      +-+-+-+-+ +-+-+-+-+ +-+-+-+ +-+-+-+ +--+--+--+ +-+-+ +-
      -+--+
      :1:2:     :1:       :1:2:3: :1:2:   : 1:       :1:2: : 1:
      2:
      +-+-+     +-+       +-+-+-+ +-+-+   +--+       +-+-+ +-
      -+--+
                :0:               :0:     : 0:       :0:1: : 0:
                +-+               +-+     +--+       +-+-+ +-
      -+
                                          :-1:             :-
      1:
                                          +--+             +-
      -+
      SFN>
```

■ **CONT**ractGroups

Format: Cont INTEGER1, INTEGER2 OPERATION EXPR
Modes: DPM

Description: Operates with OPERATION on reps of the INTEGER1 (=n)-th and
INTEGER2 (=m)-th groups of the DPrep EXPR and assigns the result
to repm whilst removing repn. The set of groups are changed accordingly. The default operation is Kronecker product under the m-th group whilst the other five operations involve the partitions labelling reps m and n.
default rep = repm * repn under the m-th group.
Mix rep = { a;b}, a mixed tensor as in U_m.
O a = a.b, the Sfn outer product.
I a = a\circ b, the Sfn inner product.
Sk a = a/b, the Sfn skew.
Pleth a = a\otimes b, the Sfn plethysm.
These operations are especially useful in implementing new product or branching rules.

Example:

```
DP>
⇒   gr2su8su8
                Groups are SU(8) * SU(8)
                DP>
⇒   prod[1 * 1],[2 * 1^2]
                { 3}{ 21} + { 3}{ 1^3} + { 21}{ 21} + { 21}{
1^3}
                DP>
⇒   cont 1,2 last
                Group is SU(8)
                { 51} + 2{ 42} + 3{ 41^2} + { 3^2} + 4{ 321}
+ 3{ 31^3} + { 2^3}
     + 2{ 2^21^2} + { 21^4}
                DP>
⇒   gr2u8u8
                Groups are U(8) * U(8)
                DP>
⇒   [2 * 2]
                { 2}{ 2}
                DP>
⇒   cont 1,2 pleth,last
                Group is U(8)
                { 4} + { 2^2}
                DP>
```

■ **CONTRAG**redientRep

Format: Contrag gr grN EXPR (DPM)
Contrag EXPR (REP)

Modes: DPM, REP

Description: Replaces the representations in EXPR by their

contragredient representations for the U(1), SU(n), and E(6) groups. The command CH_LabelForOn may be used for SO(2k) where appropriate.

Example:
```
DP>
⇒   gr3e6su3u1
                Groups are E(6) * SU(3) * U(1)
                DP>
⇒   [1^2 * 1 * ~3]
                (1:1){ 1}{ -3}
                DP>
⇒   contrag gr1last
                (1:1^5){ 1}{ -3}
                DP>
⇒   contrag gr2last
                (1:1^5){ 1^2}{ -3}
                DP>
⇒   contrag gr3last
                (1:1^5){ 1^2}{ 3}
                DP>
⇒   rep
        REP mode
                Group is E(6)
                REP>
⇒   contrag 1^6
                (1:1)
          REP>
```

■ **CONV_D_TO_R**ep

Format: Conv_D_to_R EXPR

Modes: REP

Description: Converts a list of DPreps into a list of REPs. N.B. It is assumed that the DPreps involve just one group else an error message is given.

■ **CONV_D_TO_S**fn

Format: Conv_D_to_S EXPR

Modes: SFN

Description: Converts a list of DPreps into a list of S-functions. N.B. It is assumed that the DPreps involve just one group else an error message is given. Assumes the irreps are of type tensor and do not have attached labels or mixed notation. Useful for doing plethysms in the DPMode. The inverse of converting a list of S-functions into a list of DPreps is accomplished by [conv_s_to_r EXPR] where

EXPR can be a list of S-functions or an S-function expression that generates a list of S-functions. e.g. [conv_s_to_r 21+321] or [conv_s_to_r o21,21] etc.

■ **CONV_R_TO_S**fn

Format: Conv_R_to_S EXPR
Modes: SFN
Description: Converts a list of REP irreps into a list of S-functions.
N.B. Assumes the irreps are of tensor type and do not have attached labels or mixed notation.
Example: `REP>`

```
⇒    gr sp8
                    Group is Sp(8)
                    REP>
⇒    21 + 321 + 42
          <21> + <321> + <42>
                    REP>
⇒    sfn
          Schur Function Mode
                    SFN>
⇒    conv_r_to_s last
                    { 42} + { 321} + { 21}
                    SFN>
```

■ **CONV_S_TO_R**ep

Format: Conv_S_to_R OPERATION(s) EXPR
Modes: REP, DPM
Description: Converts an S-function EXPR to REP form. default = direct transfer a -> { a}
SPin EXPR adds a spin label to EXPR, a -> [s;a]
MIXedTensorReps EXPR1,EXPR2 forms a mixed tensor a,b -> { a;b}
LABel CHAR adds a one character label CHAR to EXPR as in SO(2k)
(e.g. a -> [a]+)
More than one operation may be used at a time. The hierachy is SPin, LABel, MIXedTensorReps.
The resulting irreps may not be in standard form in which case the command STD should be used. The command may be used for an
appropriately set group in either REP or DPMode. In the case of the latter this command may be applied to one or more groups.
Example: `REP>`

```
⇒    gr so6
     Group is SO(6)
```

```
        REP>
  ⇒     conv_s_to_r spin label - outer2,1
        [s;3]- + [s;21]-
        REP>
  ⇒     gr u5
        Group is U(5)
        REP>
  ⇒     Conv_s_to_r mixed 21,31
        { 21;31}
        REP>
  ⇒     dpm
        DPrep Mode (with function)
          Group is U(5)
        DP>
  ⇒     gr3 u5sp4so6
        Groups are   U(5) * Sp(4) * SO(6)
        DP>
  ⇒     [conv_s_to_r mixed 21,31*43*conv_s_to_r spin label - o2,1]
        { 21;31}<43>[s;3]- +{ 21;31}<43>[s;21]-
        DP>
```

■ **COUNTC**oeffsInList

Format: Countc EXPR
Modes: DPM, REP, SFN
Description: Computes the sum of the coefficients of the terms in EXPR and returns CoeffSum
Example:

```
SFN>
  ⇒     countc 42+41^2+3^2+2.321+31^3 +2^3 +2^2 1^2
                    CoeffSum = 8
                    SFN>
  ⇒     countc outer 21,21
                    CoeffSum = 8
                    SFN>
```

■ **COUNTT**ermsInList

Format: CountT EXPR
Modes: DPM, REP, SFN
Description: Counts the number of distinct terms in EXPR without regard to their multiplicity, and returns TermCount.
Example:

```
SFN>
  ⇒     outer21,21
                    { 42} + { 41^2} + { 3^2} + 2{ 321} + { 31^3}
        + { 2^3} + { 2^2 1^2}
                    SFN>
  ⇒     countt last
                    TermCount = 7
```

```
                    SFN>
⇒   countt outer4321,4321
                    TermCount = 206
                    SFN>
```

■ **COV**ariant

Format: COV EXPR
Modes: REP
Description: Converts a mixed tensor irrep of the group U(n) into an equivalent covariant irrep. The group must be set as U(n) else an error message is generated. The result is printed as { p;lambda} where the integer 'p' is to be interpreted as the exponent of the one-dimensional irrep whose basis is a totally antisymmetric n-th rank covariant tensor and 'lambda' is a standard covariant irrep of U(n).
Example:

```
REP>
⇒   gr u6
                    Group is U(6)
                    REP>
⇒   cov 21;321
                    { -2;54321}
                    REP>
```

■ **DEAD**

Format: DEAD qfn1,qfn2,qfn3
Modes: SFN
Description: qfn1,qfn2, qfn3 are Q-functions specified by partitions. DEAD is a Boolean operator that is FALSE if the Q-function qfn1 does not appear in the Q-function product qfn2 . qfn3. If DEAD is TRUE then qfn1 probably appears in the product, This is an implementaion of Theorem 2 in J. Math. Phys, (31) 1310 (1990).
Example:

```
SFN>
⇒   dead!10 654321,32,765431
                    dead =true
                    SFN>
⇒   dead 9875421,32,765431
                    dead =false
                    SFN>
```

■ **DIM**ensions

Format: Dim EXPR
Modes: DPM, REP
Description: Calculates the dimensions of EXPR. Dimensions are computed for all the compact semisimple Lie groups and the symmetric groups and for the representations associated with products

of those groups. Dimensions for the supergroups U(m/n) and OSp(m/n) can be calculated by first branching to their maximal subgroups U(m) * U(n) or O(m) * Sp(n).

Example: `DP>`

⇒ `gr3su8sp4so6`

```
                Groups are SU(8) * Sp(4) * SO(6)
                DP>
```

⇒ `dim[4321*42*s321+]`

```
                Dimension = 6035420160
          DP>
```

Notes:

`See also Sb_Dimensions`

■ **DPM**ode

Format: DPM

Modes: REP, SFN

Description: Invoking DPMode in the REPmode or SFNmode returns SCHUR to the DPMode. The same effect can be produced by the command EXITmode.

■ **D_TO_P**label

Format: D_to_P DYNKIN_LABEL

Modes: REP

Description: Converts the partition labels for the irreps in EXPR to Dynkin labels. The group must first be set to be one of the simple Lie groups: SU(n), SO(n), Sp(2n), E6, E7, E8, F4 or G2.

Example: `REP>`

⇒ `gr g2`

```
Group is G(2)
REP>
```

⇒ `d_to_p 12`

```
<<partition>>      (41)
REP>
```

⇒ `gr e8`

```
Group is E(8)
REP>
```

⇒ `d_to_p 021+2.10^61`

```
<<partition>>      (10 3^5 2) +2(51^7 )
REP>
```

■ **DYNKINI**ndex

Format: DynkinI EXPR

Modes: REP

Description: Computes the total second order Dynkin index for the irreps in EXPR. See also the entry for the command PROP.

The normalisation is chosen to give the dynkin index a value of 1 for the fundamental irrep of the group.

Example:
```
REP>
⇒   gr e8
                    Group is E(8)
            REP>
⇒   dynkini 21^7
                    2nd-dynkin = 1
                    REP>
⇒   dynkini 21
                    2nd-dynkin = 25
                    REP>
⇒   dynkini 21^7 +2.21
                    2nd-dynkin = 51
                    REP>
```

■ **END**

Format: End
Modes: All
Description: Ends the current session. Closes and saves any LOGfiles still open.

■ **ENT**erVar

Format: Ent OPERATION comment
Modes: DPM, REP, SFN
Description: Use of this line in a function allows the user to input data to the function while it is being executed. SVar N requires the user to input Sfn variable N and is available in the DPM, REP and SFN modes. RVar N requires the user to input rep variable N and is available in DPM and REP modes. Var N requires the user to input DPrep N and is available in the DPM mode. GRoup requires the user to enter the groups the function requires in the same format as for the GRoup command, (see GRoup). 'comment' is any text you wish that may be helpful to the user of the function.

■ **EQ**ualSfnList

Description: This command is used in conjunction with the command RULE and
is described under that heading.

■ **E_T**o_FsymmFn

Format: E_To_F EXPR

Modes: SFN
Description: Treats EXPR as a list of elementary symmetric functions e_{ \lambda} and transforms them into a list of forgotten symmetric functions f_{ \mu}.
Example: SFN>
⇒ `e_to_f 21`

```
                f_{ 3} + 2f_{ 21} + 3f_{ 1^3}
                SFN>
```

■ **E_T**o_HsymmFn
Format: E_To_H EXPR
Modes: SFN
Description: Treats EXPR as a list of elementary symmetric functions e_{ \lambda} and transforms them into a list of homogeneous symmetric functions h_{ \mu}.
Example: SFN>
⇒ `e_to_h 21`

```
                - h_{ 21} + h_{ 1^3}
                SFN>
```

■ **E_T**o_MsymmFn
Format: E_To_M EXPR
Modes: SFN
Description: Treats EXPR as a list of elementary symmetric functions e_{ \lambda} and transforms them into a list of monomial symmetric functions m_{ \mu}.
Example: SFN>
⇒ `e_to_m 21`

```
                m_{ 21}  + 3m_{ 1^3}
          SFN>
```

■ **E_TO_S**symmFn
Format: E_To_S EXPR
Modes: SFN
Description: Treats EXPR as a list of elementary symmetric functions e_{ \lambda} and transforms them into a list of Schur symmetric functions.
Example: SFN>
⇒ `e_to_s 21`

```
                { 21} + { 1^3}
          SFN>
```

■ **EXIT**mode
Format: EXIT
Modes: REP, SFN, BRM
Description: Returns the user to the DPMode from any other mode.

■ **EXP**andSfnList
Format: Exp EXPR
Modes: SFN
Description: Expands EXPR by expressing every term c{ a} as a sum of c terms
{ a} if c is positive, and as a sum of -c terms -{ a} if c is negative, so that in the final expression all coefficients are either +1 or -1.
Example: SFN>
⇒ exp 3.321 -2.21
{ 321} +{ 321} +{ 321} -{ 21} -{ 21}
SFN>

■ **FFPROD**
Format: FFPROD EXPR1,EXPR2
Modes: REP
Description: EXPR1 is a list of finite non-unitary irreps of Sp(2n,R) while EXPR2 is an infinite dimensional unitary irrep.
The action of FFPROD is to form the generic product of EXPR1 x EXPR2 to give a finite list of unitary irreps of Sp(2n,R).
The group must be set as Sp(2n,R) else an error message is generated. If any term in the intermediateproducts do not correspond to a generic product an error message is generated.
See also FPROD, GENERIC and COVariant.
Example: REP>
⇒ gr spr6
Group is Sp(6,R)
REP>
⇒ ffprod 0;21 ,6;21
<8(0)> +<7(3)> +2<7(21)> +3<7(1)> +<6(42)> +<6(4)>
+<6(3^2)> +4<6(31)> +3<6(2^2)> +3<6(2)> +3<6(1^2
)>
+<6(0)> +<5(52)> +2<5(43)> +2<5(41)> +4<5(32)>
+<5(3)> +2<5(21)> +<4(53)> +<4(4^2)> +<4(42)>
+<4(3^2)>
REP>

■ **FIRSTP**artsort
Format: FIRSTP INTEGER, EXPR
Modes: REP, SFN
Description: If INTEGER is positive then this command returns all terms in EXPR whose first part is less than or equal to INTEGER. If INTEGER is negative then the command returns all
terms in EXPR whose first part is exactly equal to INTEGER.

Example: SFN>
⇒ firstp 3,42+321+2^3+1^4
{ 321} + { 2^3} + { 1^4}
SFN>
⇒ firstp -3,last
{ 321}
SFN>

■ **FN**

Format: Fn INTEGER
Modes: DPM, REP, SFN
Description: INTEGER is N. Fn executes function N, N =1,...,50. The first command in function N must match the mode you start the function in. Usually only the result of the last command in function N is ported to the screen. The exceptions are the results of the PROP, DIM, GRoup and mode change commands. If you wish other intermediate results to appear on the screen use the command SUPpressOutputToScreen at the appropriate place in your function. N.B. Functions cannot call other functions, i.e. you may not have Fn N as a line in a function. An extensive account of the construction and use of functions is given in the Tutorial 5.1.

■ **FPROD**

Format: FPROD EXPR1,EXPR2
Modes: REP
Description: EXPR1 is a list of finite covariant irreps of U(n) entered in the covariant form p;a, 0;a or ~p;a, where p is a positive integer and ~p signifies -p.
EXPR2 is an irrep of the non-compact group Sp(2n,R).
The action of FPROD is to form the generic product of EXPR1 x EXPR2 to give a finite list of irreps of Sp(2n,R). The group must be set as Sp(2n,R) else an error message is generated. If any terms in the products do not correspond to a generic product an error message is generated.
Example: REP>
⇒ gr spr6
Group is Sp(6,R)
REP>
⇒ fprod ~1;21, 5;21
<4(42)> +<4(41^2)> +<4(3^2)> +2<4(321)>
+<4(2^3)>

```
                    REP>
⇒     fprod 0;1+~2;32, 4;21
           <4(31)> +<4(2^2 )> +<4(21^2 )> +<2(53)> +<2(521)>
           +<2(4^2 )> +2<2(431)> +<2(42^2 )> +<2(3^2 2)>
                    REP>
```

■ **FRACAH**notation
Format: FRACAH EXPR
Modes: REP
Description: Converts an EXPR of Racah G(2) irreps labelled in Racah's SO(3) basis into a list of G(2) irreps labelled in the SU(3) basis. See also RACAHnotation.
Example:

```
REP>
⇒     gr g2
                    Group is G(2)
                    REP>
⇒     prod31,21
                    (52) + (41) + (4) + 2(31) + (3) + (2) + (1)
                    REP>
⇒     racah last
             (4) + (32) + (31) + (3) + 2(21) + (2) + (1)
                    REP>
⇒     fracah last
           (52) +(41) +(4) +2(31) +(3) +(2) +(1)
                    REP>
```

■ **FROB**enius
Format: FROBenius INTEGER, EXPR
Modes: SFN
Description: If INTEGER is positive then this command returns all terms in EXPR which are of Frobenius rank INTEGER or less. If INTEGER is negative then the command returns all terms in EXPR which are of Frobenius rank INTEGER exactly.
Example:

```
SFN>
⇒     outer21,21
                 { 42} + { 41^2 } + { 3^2 } + 2{ 321} + { 31^3
} + { 2^3 } + { 2^2 1^2 }
                  SFN>
⇒     frob-2last
                  { 42} + { 3^2 } + 2{ 321} + { 2^3 } + { 2^2
1^2 }
                 SFN>
```

■ **F_T**o_EsymmFn
Format: F_To_E EXPR
Modes: SFN
Description: Treats EXPR as a list of forgotten symmetric functions

f_{ \lambda} and transforms them into a list of elementary symmetric functions e_{ \mu}.

Example: `SFN>`

⇒ `f_to_e 21`

```
                - 3e_{ 3} + 5e_{ 21} - 2e_{ 1^3}
                SFN>
```

■ **F_T**o_HsymmFn

Format: F_To_H EXPR

Modes: SFN

Description: Treats EXPR as a list of forgotten symmetric functions f_{ \lambda} and transforms them into a list of homogeneous symmetric functions h_{ \mu}.

Example: `SFN>`

⇒ `f_to_h 21`

```
                - 3h_{ 3} + h_{ 21}
                SFN>
```

■ **F_T**o_MsymmFn

Format: F_To_M EXPR

Modes: SFN

Description: Treats EXPR as a list of forgotten symmetric functions f_{ \lambda} and transforms them into a list of monomial symmetric functions m_{ \mu}.

Example: `SFN>`

⇒ `f_to_m 21`

```
                - 2m_{ 3} - m_{ 21}
                SFN>
```

■ **F_T**o_SsymmFn

Format: F_TO_S EXPR

Modes: SFN

Description: Treats EXPR as a list of forgotten symmetric functions f_{ \lambda} and transforms them into a list of $S-$functions { \mu}.

Example: `SFN>`

⇒ `f_to_s 21`

```
          -2{ 3} +{ 21}
                SFN>
```

■ **FUS**ion

Format: Fus INTEGER EXPR

Modes: REP

Description: The INTEGER, which should be positive, is the level of an affine representation, and EXPR a list of irreps of either SU(n) or Sp(n). In the case of SU(n) the group should be set as U(n) and the irreps given in mixed form. It is assumed that the irreps are

either standard or have been made standard with the operation STD. For further details see Cummins, J.Phys.A:Math.Gen. 24,391-400(1991).

Example: REP>

```
⇒   gr sp10
                    Group is Sp(10)
                    REP>
⇒   std532^6
                    - <532^3 >
                    REP>
⇒   fus 3 last
                    <3^2 2^3 >
                    REP>
⇒   gr u8
                    Group is U(8)
                    REP>
⇒   std 7632^31;982^41
                    { 7631;9821}
                    REP>
⇒   fus 10 last
                    { 43^2 1;6521}
                    REP>
```

■ **GENERIC**

Format: GENERIC REP1 and REP2

Modes: REP

Description: REP1 and REP2 should be standard irreps of Sp(2n,R). Then this command gives the output generic=true if the product of REP1 and REP2 is generic, otherwise the return is generic=false. For further details see F. Toumazet, in "Symmetry, Spectroscopy and SCHUR", Proc. Brian G. Wybourne Conference, UMK Publ, Torun, Poland, 2006 pp????".

Example: REP>

```
⇒   gr spr8
                    Group is Sp(8,R)
                    REP>
⇒   generic 2;1,5;62
         generic=true
                    REP>
```

■ **GEN**prod

Format: GEN INTEGER

Modes: SFN

Description: Generates the list of monomials corresponding to the expansion of the product over i from 1 to (INTEGER N - 1) of (x_i - x_N)^2.

A typical monomial x_1 x_3^3 is represented by { 103}. See also VMult and SMON.

Example: SFN>

⇒
```
gen 3
      { 2^2 } - 2{ 21^2 } + { 202} - 2{ 121} + 4{ 1^2 2}
- 2{ 103} + { 02^2 } - 2{ 013}
      + { 0^2 4}
              SFN>
```

■ **GR**oup

Format: Gr INTEGER GROUPNAMES (DPM)
Gr GROUPNAME (REP)

Modes: DPM, REP

Description: Sets the group(s) with respect to which all rep and DPrep calculations are done. INTEGER is the number of groups in a direct product and is not required if equals to 1. The maximum number of groups that may be set in DPM is 6. Table A.3 shows how the available groups are to be entered.
N.B. Not all the groups listed in Table A.3 are available for all the commands in SCHUR so check the appropriate commands.

Example: DP>

⇒
```
gr4e8f4g2e6
              Groups are E(8) * F(4) * G(2) * E(6)
              DP>
```

⇒
```
gr sumn4 6
              Group is SU(4/6)
              DP>
```

Notes:

Table A.3: Formats for entry of groups in SCHUR

Name of Group	code for group	Name of Group	code for group
U(n)	u n	S(n)	s n
SU(n)	su n	E(6)	e6
O(n)	o n	E(7)	e7
SO(n)	so n	E(8)	e8
Sp(n)	sp n	OSp(m/n)	osp m n
G(2)	g2	U(p/q)	umn p q
F(4)	f4	SU(p/q)	sumn p q
Sp(n,R)	spr n	Mp(n)	mp n
SO^*(2p)	son 2p		

■ **GWT**

Format: GWT INTEGER EXPR

Modes: SFN
Description: Takes the list of S-functions generated by EXPR and removes all those of weight less than INTEGER.
Example: SFN>

```
⇒   o21,3+5
                { 71} + { 62} + { 61^2 } + { 521} + { 51} +
{ 42} + { 41^2 } + { 321}
                SFN>
⇒   gwt7,last
                { 71} + { 62} + { 61^2 } + { 521}
                SFN>
```

■ **HALL**polynomialProduct
Format: Hall SFN, INTEGER
Modes: SFN
Description: Forms the product of a single Hall-Littlewood polynomial, for arbitrary variable t, corresponding to the partition associated with SFN with the Hall-Littlewood polynomial involving a single part INTEGER. The resulting coefficients are given as an array of integers a,b,c... displayed as (abc...). Each integer x corresponds to (1 - t^x).
The polynomial constructed from (abc...) must be divided by (1 - t). The implementation is based on Macdonald's "Symmetric Functions and Hall Polynomials" page 113 3.8.
An error is indicated if INTEGER is entered with more than one part
Example: SFN>

```
⇒   hall 32^21,3
                (1){ 62^2 1} + (1){ 5321} + (31){ 52^3} + (21){
52^2 1^2} + (1){ 432^2} + (21){ 4321^2}
     + (31){ 42^3 1} + (2){ 3^2 2^2 1}
     SFN>
```

Notes:

```
(31){ 52^3} -> (1 - t^3){ 52^3}
where for example { 52^3} is interpreted as the HL-polynomial
corresponding to the partition (52^3) etc.
```

■ **HCLASS**
Format: HCLASS EXPR
Modes: SFN
Description: Used in place of CLASS to record the number of elements of the class of S(N) specified by each partition in EXPR in those cases for which their might be an integer overflow.
Example: SFN>

```
⇒   class !15 5
```

```
 -543424512{ 15 5}
 SFN>
⇒ hclass !15 5
 Number of elements in the class =
 32438693442355200
 SFN>
```

■ **HEAP**status

Format: HEAPstatus

Modes: SFN, REP, DPM

Description: This command is mainly used by SCHUR code developpers. It displays count, rcount and pcount values: counters of dynamic allocations made respectively of sfn, ocharptr and prodtype structures.

Example:

```
SFN>
⇒ o21,2
        { 41} +{ 32} +{ 31^2 } +{ 2^2 1}
SFN>
⇒ heap
Count  = 4 Rcount = 0 Pcount = 0
SFN>
⇒ nskew6,!18 !12 6,6
23766{ 0}
SFN>
⇒ heap
Count  = 1 Rcount = 0 Pcount = 0
SFN>
⇒ rep
REP mode
REP>
⇒ gr spr8
Group is Sp(8,R)
  REP>
⇒ 2;0 + 2;2 + 2;2^2
        <2(0)> +<2(2)> +<2(2^2 )>
REP>
⇒ heap
 Count  = 1 Rcount = 3 Pcount = 0
REP>
⇒ sfn
Schur Function Mode
SFN>
⇒ last
23766{ 0}
SFN>
```

■ **HECKE**

Format: HECKE PARTITION
Modes: DPM
Description: Determines the character of the irrep of the Hecke algebra H_n(q), for q not a root of unity, in the class specified by a single PARTITION, entered without any enclosing brackets, whose parts give the cycle structure and whose weight is n. Prior to using HECKE the group must be set as U(n)*U(1), where n is the order of the Hecke algebra. (See pages 88-91 of the SCHUR manual for some additional details). In the output the partitions associated with U(n) correspond to the irrep labels of H_n(q), while the irrep of U(1) gives the exponents of q.
Example: DP>

```
⇒   gr2u6u1
                    Groups are U(6) * U(1)
                    DP>
⇒   Hecke 42
                    { 6}{ 4} + { 51}{ 4} - 2{ 51}{ 3} + { 42}{
4} - { 42}{ 3} + { 42}{ 2}
      - 2{ 41^2 }{ 3} + 2{ 41^2 }{ 2} - { 3^2 }{ 3} - {
321}{ 3} + 2{ 321}{ 2}
      - { 321}{ 1} + 2{ 31^3 }{ 2} - 2{ 31^3 }{ 1} - { 2^3
}{ 1} + { 2^2 1^2 }{ 2}
      - { 2^2 1^2 }{ 1} + { 2^2 1^2 }{ 0} - 2{ 21^4 }{ 1}
+ { 21^4 }{ 0}
      + { 1^6 }{ 0}
                    DP>
```

Notes:

```
        Thus we can deduce that the character of the ir-
rep { 321} and class
(42) of H_6(q) is (-q^3 + 2q^2 - q). Putting q = 1 gives
the
corresponding character of the symmetric group S(6), in
this case zero.
```

■ **HSTD**

Format: HSTD REP
Modes: REP
Description: HSTD is a Boolean that gives the result as "True" if rep is a highly standard irrep of the group Sp(2n,R) or SO*(2n). For all other groups an error mesage results.
Example: REP>

```
⇒   gr spr6
                     Group is Sp(6,R)
                     REP>
```

```
⇒     hstd 3;21
                        highlystandard =true
                        REP>
⇒     hstd 2;32
                        highlystandard =false
                        REP>
```

■ **HSTDL**ist

Format: HSTDL EXPR
Modes: REP
Description: HSTDList is a Boolean that gives the result as "True" if the list of all reps in EXPR contains only highly standard irreps of the group Sp(2n,R) or SO*(2n). For all other groups an error mesage results. If EXPR contains any non-highly standard irreps the result is "False". See also HSTD.
Example:

```
REP>
⇒     gr spr6
                        Group is Sp(6,R)
                        REP>
⇒     hstdl 3;21 + 4;22
                        highlystandard list =true
                        REP>
⇒     hstdl 3;21 + 4;22 + 2;32
                        highlystandard list =false
                        REP>
```

■ **H_To**_EsymmFn

Format: H_To_E EXPR
Modes: SFN
Description: Treats EXPR as a list of homogeneous symmetric functions h_{ \lambda} and transforms them into a list of elementary symmetric functions e_{ \mu}.
Example:

```
SFN>
⇒     h_to_e 21
                    - e_{ 21}  + e_{ 1^3}
          SFN>
```

■ **H_To**_FsymmFn

Format: H_To_F EXPR
Modes: SFN
Description: Treats EXPR as a list of homogeneous symmetric functions h_{ \lambda} and transforms them into a list of forgotten symmetric functions f_{ \mu}.
Example:

```
SFN>
⇒     h_to_f 21
                    f_{ 21}  + 3f_{ 1^3}
```

```
                    SFN>
```

■ **H_T**o_MsymmFn
Format: H_To_M EXPR
Modes: SFN
Description: Treats EXPR as a list of homogeneous symmetric functions h_{ \lambda} and transforms them into a list of monomial symmetric functions m_{ \mu}.
Example: SFN>
⇒ `h_to_m 21`

```
                    m_{ 3}  + 2m_{ 21}  + 3m_{ 1^3}
                    SFN>
```

■ **H_T**o_SsymmFn
Format: H_To_S EXPR
Modes: SFN
Description: Treats EXPR as a list of homogeneous symmetric functions h_{ \lambda} and transforms them into a list of Schur symmetric functions.
Example: SFN>
⇒ `h_to_s 21`

```
                    { 3} + { 21}
                    SFN>
```

■ **INDEX**sequence
Format: INDEX INTEGERS
Modes: SFN
Description: INDEX acts on a sequence of positive integers to produce a new sequence known as the Index of the Sequence. If the members of the sequence are a(r) r = 1,2 ...,N then the index i(r) is defined as the number of integers in the sequence lying to the left of a(r) which are equal to a(r), including a(r) itself, minus the number of integers lying to the left of a(r) which are equal to a(r) - 1.
Example: SFN>
⇒ `index111223324`

```
{ 123-2-1^2 0^2 -1}
                    SFN>
```

■ **INS**ertPartitionIntoSfn
Format: Ins PARTITION, EXPR
Modes: SFN
Description: Inserts the PARTITION together with trailing 0's if required, in front of every member of EXPR without standardising the new expression.
Example: SFN>
⇒ `ins 62,-43+2.321+21`

```
          -{ 6243} +2{ 62321} +{ 62^2 1}
          SFN>
⇒    std last
          { 63^3 } +{ 62^2 1}
                  SFN>
⇒    ins 6200,64
          { 620^2 64}
          SFN>
⇒    std last
          -{ 63^2 2^3 }
                  SFN>
```

■ **INT**egerDivideCoeffs

Format: Int INTEGER EXPR
Modes: DPM, REP, SFN
Description: Divides the coefficients in EXPR by the INTEGER. Gives an error message if the coefficients are not 0 mod INTEGER.
Example:

```
SFN>
⇒    !288.543 + !12.4321 + !48.321 - !72.21
                  288{ 543} + 12{ 4321} + 48{ 321} - 72{ 21}
                  SFN>
⇒    int 12 last
                  24{ 543} + { 4321} + 4{ 321} - 6{ 21}
                  SFN>
```

■ **INV**series

Format: INV INTEGER EXPR
Modes: DPM
Description: Used to compute the terms of the inverse of the series defined by EXPR up to weight INTEGER. The group is first set as appropriate to the irreps of the terms in EXPR. If the group is of the form U(1)*G then INTEGER refers to the weight of the irreps of U(1).
Example:

```
DP>
⇒    gr u12
     Group is U(12)
     DP>
⇒    [conv_s_to_r ser3,t]
     { 21} + { 1} + { 0}
     DP>
⇒    inv 3 last
     - { 3} - 3{ 21} + { 2} - { 1^3 } + { 1^2 } - { 1} + { 0}
     DP>
⇒    gr2u1u12
```

```
        Groups are   U(1) * U(12)
        DP>
   ⇒    mac a 6
        - { 3}{ 31^3 } - { 3}{ 2^3 } + { 2}{ 21^2 } - { 1}{ 1^2
        } + { 0}{ 0}
        DP>
   ⇒    setv1 last
        DP>
   ⇒    inv6 last
        { 3}{ 3^2 } + { 3}{ 2^2 1^2 } + { 3}{ 1^6 } + { 2}{ 2^2
        } + { 2}{ 1^4 }
        + { 1}{ 1^2 } + { 0}{ 0}
        DP>
   ⇒    wt 2, 6, p last, v1
        { 0}{ 0}
        DP>
   ⇒    gr2 u1 sp4
        Groups are   U(1) * Sp(4)
        DP>
   ⇒    mac c 4
        { 2}<31> -{ 1}<2> +{ 0}<0>
        DP>
```

■ **I_PL**ethysmRd

Format: I_Pl, EXPR
Modes: SFN
Description: Evaluates the plethysm <1> \otimes EXPR for S(N) in the reduced notation.
Example:

```
SFN>
   ⇒    i_pl 21
                <21> + <2> + <1^2 > + <1>
                SFN>
```

■ **I_Q**fnProduct

Format: I_Q EXPR1,EXPR2
Modes: SFN
Description: I_Q forms the inner product of the Q-functions in EXPR1 and EXPR2 to yield S-functions.
Example:

```
SFN>
   ⇒    i_q21+31,3+21+4
                2{ 31} + 2{ 2^2} + 2{ 21^2} + 4{ 21}
```

Notes:

```
 02/01/2006 This example seems to be wrong !
```

■ **I**_sfnProduct

Format: I EXPR1,EXPR2
Modes: SFN

Description: Calculates the sum of the inner products of all pairs of S-functions in EXPR1 and EXPR2 that have the same weight, including the product of their coefficients. It gives zero if there no such pairs having the same weight.

Example: `SFN>`

⇒ `i 31, 21^2`

```
                { 31} + { 2^2} + { 21^2} + { 1^4}
     SFN>
```

⇒ `i 31, 21`

```
     zero
     SFN>
```

⇒ `i 31-21, 2.21+21^2`

```
     { 31} -2{ 3} +{ 2^2 } +{ 21^2 } -2{ 21} +{ 1^4 } -
2{ 1^3 }
                SFN>
```

■ **I_SFNQ**fnProduct

Format: I_SfnQ EXPR1,EXPR2

Modes: SFN

Description: Calculates the sum of the inner products of pairs consisting of an S-function in EXPR1 and a Q-function in EXPR 2 that have the same weight, including the product of their coefficients. It gives zero if there no such pairs having the same weight.

Example: `SFN>`

⇒ `i_sfnq 321, 6 -5 +2.51`

```
     4Q_{ 6} +11Q_{ 51} +18Q_{ 42} +3Q_{ 321}
     SFN>
```

■ **KINS**ert

Format: KINS INTEGER gr grN EXPR (DPM)
KINS INTEGER EXPR (REP)

Modes: REP, DPM

Description: Use restricted to reps of the groups Sp(2n,R) and SO^*(2n). Replaces the integer k in <k(\lambda)> or [k(\lambda)] by INTEGER.
In the case of harmonic reps of Sp(2n,R) the spin index s is left unchanged. It may be changed using the command ch_spin.

Example: `DP>`

⇒ `gr spr8`

```
                Group is Sp(8,R)
                DP>
```

⇒ `[s2;21]`

```
                <s2(21)>
                DP>
```

⇒ `kins 3 gr1 last`

```
                    <s3(21)>
                    DP>
    ⇒     ch_spin gr1 last
                    <3(21)>
                    DP>
```

■ **KM**atrix

Format: KM INTEGER
Modes: SFN
Description: Computes the elements of the Kostka matrix and displays the result as a square array. The order of the rows and columns are those of reverse lexicographic ordering of the partitions of INTEGER. For example, in the case INTEGER=4 the order is 4, 31, 2^2, 21^2, 1^5.
Example:

```
SFN>
    ⇒ km 4
1 1 1 1 1
0 1 1 2 3
0 0 1 1 2
0 0 0 1 3
0 0 0 0 1
SFN>
```

■ **K**ostka

Format: Kostka SFN1, SFN2
Modes: SFN
Description: Computes the element of the Kostka matrix K_{ a,b} where a=SFN1 and b=SFN2 which are single S-functions.
Example:

```
SFN>
    ⇒ k4321,221^6
                Kostka matrix element =
     192
                SFN>
```

■ **LAB**el

Format: Conv_S_to_R OPERATION(s) EXPR
Modes: REP, DPM
Description: The command LABel is used in conjunction with CONV_S_TO_Rep.
The "LABel CHAR" OPERATION adds a one character label CHAR
to EXPR as in SO(2k) (e.g. a -> [a]+)
More than one operation may be used at a time.
The hierachy is SPin, LABel, MIXedTensorReps.
The resulting irreps may not be in standard form in which case the command STD should be used.
Example:

```
REP>
```

```
⇒     gr so6
                    Group is SO(6)
                    REP>
⇒     conv_s_to_r spin label - outer2,1
                    [s;3]- + [s;21]-
                    REP>
```

■ **LAST**result

Format: Last
Modes: DPM, REP, SFN
Description: LAST is the result of the last operation and is used as an expression.
Example:

```
SFN>
⇒     o 2,2
                    { 4} + { 31} + { 2^2}
                    SFN>
⇒     last
                    { 4} + { 31} + { 2^2}
                    SFN>
⇒     sk last,2
                    3{ 2} + { 1^2}
                    SFN>
```

Notes:

```
 Some commands, such as DIM, produce an output that is neither an EXPR,
nor a REP nor an SFN. In such cases, usually indicated by some text, such as
"dimension=124702", LAST may give zero or a previous EXPR, REP or SFN.
```

■ **LAT**ticetest

Format: LAT Sequence
Modes: SFN
Description: Sequence is a sequence of positive integers. LATTICE is a boolean that is TRUE if the sequence is a lattice permutation or FALSE otherwise
Example:

```
SFN>
⇒     lattice1111223324
                    lattice test = true
                          { 1^4 2^2 3^2 24}
                    SFN>
```

■ **LEN**gthOfPartitionsSelect

Format: Len grno,INTEGER, EXPR (DPM)
Len INTEGER, EXPR (REP, SFN)
Modes: DPM, REP, SFN
Description: In the DPMode if INTEGER is positive all terms in EXPR which

are of length INTEGER or less for grno are returned.
If INTEGER is negative then all terms in EXPR having a number of parts equal to the magnitude of INTEGER for grno are returned.
In the REP or SFN mode If INTEGER is positive then all terms in EXPR which are of length INTEGER or less are returned.
If INTEGER is negative then all terms in EXPR having a number of parts equal to the magnitude of INTEGER are returned.

Example:
```
SFN>
```
⇒
```
len 3,321+321^2 + 31 +2^6+2
             { 321} + { 31} + { 2}
             SFN>
```
⇒
```
len -3,last
             { 321}
             SFN>
```

■ **LINES**

Format: LINES INTEGER
Modes: DPM, REP, SFN
Description: Sets the number of lines for output. The default value has been set at 50.

■ **LO**adFile

Format: Lo RVar 'filename' (REP)
Lo SVar 'filename' (SFN)
Modes: REP, SFN
Description: Loads a previously saved file of RVar or Svar as chosen. See SAVEsetVar and SETSfnVar for additional details.

Example:
```
SFN>
```
⇒
```
setsv1 3+21+2
SFN>
```
⇒
```
save svar 'filename'
save[1]
SFN>
```
⇒
```
load svar 'filename'
load[1]
SFN>
```
⇒
```
sv1
{ 3} + { 21} + { 2}
SFN>
```

■ **LOG**file

Format: Log 'filename'

Modes: DPM, REP, SFN
Description: A logfile maybe setup at anytime by use of the command Log'filename'. This has the effect of directing current results to the file 'filename'. This file may be closed at anytime by issuing the command Log". The filename 'prn.' should direct the output to an attached printer. Exiting SCHUR will automatically close the current logfile. A logfile may be subsequently edited by any appropriate text editor.
Example:

```
DP>
⇒   log'testfile.tst'
                DP>
⇒   gr e8
                Group is E(8)
                DP>
⇒   dim[63]
                Dimension = 2903770000
                DP>
⇒   log''  (Closes logfile)
          DP>
```

■ **LSEQ**uence
Format: LSEQ Sequence
Modes: SFN
Description: LSEQ acts on a sequence of integers to produce the number of parts, or length, of the Sequence.
Example:

```
SFN>
⇒   lseq 1112234576
              length =      10
                     { 1^3 2^2 34576}
               SFN>
```

■ **MAC**series
Format: Macm ch INTEGER
Modes: DPM
Description: Used to generate both the q-dependent version of the Littlewood S-function series and the Macdonald denominator series expansion for the affine Kac-Moody algebras A_{ n-1}^{ (1)}. The group is first set as U(1)*U(n). The group U(1) is used to store the exponent w of q, which is the weight of the partition specifying each term in the SERIES which is to be entered as the single character f. INTEGER is the maximum weight of the irrep of U(1),
that is the maximum exponent of q, that is required. The output is a sum of mixed tensor irreps of U(N), { zeta';zeta}, where

zeta' is the conjugate of the partition zeta, with coefficient (-1)^w and U(1) label { w}, where w is the weight of the partition zeta.

Example: DP>

⇒
```
gr2 u1 u8
Groups are   U(1) * U(8)
DP>
```

⇒
```
macm f 4
{ 4}{ 4;1^4 } +{ 4}{ 31;21^2 } +{ 4}{ 2^2 ;2^2 } +{ 4}{
21^2 ;31} +{ 4}{ 1^4 ;4}
+{ 3}{ 3;1^3 } +{ 3}{ 21;21} +{ 3}{ 1^3 ;3} +{ 2}{ 2;1^2
} +{ 2}{ 1^2 ;2}
+{ 1}{ 1;1} +{ 0}{ 0;0}
DP>
```

■ **MAC**series

Format: Mac ch INTEGER

Modes: DPM

Description: Used to generate both the q-dependent version of the Littlewood S-function series and the Macdonald denominator series expansion for affine Kac-Moody algebras. The group is first set as U(1)*G, where G=U(n) for S-function series, and is any one of the classical groups for the affine case. The group U(1) is used to store the exponent of q, which is w/2 for the series a,b,c,d, and w in all other cases, where w is the weight of the partition specifying each irrep of G. The SERIES is entered as a single character a,b,...,z, and INTEGER is the maximum weight of the irrep of U(1), that is the maximum exponent of q, that is required.

Example: DP>

⇒
```
gr2 u1 u8
Groups are   U(1) * U(8)
DP>
```

⇒
```
mac g 6
{ 2}{ 321} +{ 2}{ 31^2 } -{ 1}{ 2^2 } -{ 1}{ 21} +{ 0}{
1} +{ 0}{ 0}
DP>
```

⇒
```
mac c 6
-{ 3}{ 41^2 } -{ 3}{ 3^2 } +{ 2}{ 31} -{ 1}{ 2} +{ 0}{ 0}
DP>
```

⇒
```
last
-{ 3}{ 41^2 } -{ 3}{ 3^2 } +{ 2}{ 31} -{ 1}{ 2} +{ 0}{ 0}
DP>
```

⇒
```
gr2 u1 sp4
Groups are   U(1) * Sp(4)
```

```
DP>
⇒   last
-{ 3}<41^2 > -{ 3}<3^2 > +{ 2}<31> -{ 1}<2> +{ 0}<0>
DP>
```

■ **MAKE**wtOfSfnToN

Format: Make INTEGER EXPR
Modes: SFN
Description: Takes each partition in EXPR and makes each one up to weight INTEGER by inserting an appropriate first part and then returns a
standardised list. Useful in going from reduced notation for S(N) to a specific value of INTEGER.
Example:

```
SFN>
⇒   make 10 421+32+31+2^3
{ 631} + { 532} + { 42^3}
SFN>
⇒   make 8 421+32+31+2^3
{ 431} +{ 3^2 2} -{ 32^2 1} +{ 2^4 }
SFN>
```

■ **MAXC**oeffInList

Format: Maxc, EXPR
Modes: DPM, REP, SFN
Description: Finds the largest multiplicity coefficient in the list produced by EXPR and echoes it to the screen.
Example:

```
SFN>
⇒   maxc 6.321 + 4.32 + !12.21
MaxCoeff = 12
SFN>
```

■ **MIX**edTensorReps

Format: [Conv_s_to_r mix EXPR1,EXPR2] (DPM)
Conv_s_to_r mix EXPR1,EXPR2 (REP)
Modes: DPM, REP
Description: This command is used in conjunction with the command CONV_S_TO_Rep and is used to combine two Sfn expressions to produce mixed tensor reps of SU(n) or U(n). Mixed tensor reps are of the form { a;b} and are always entered by placing a semicolon ';' between the partition entries for a and b.
Example:

```
DP>
⇒   gr2u6u4
Groups are U(6)*U(4)
DP>
⇒   [conv_s_to_r mix 32,421 * conv_s_to_r mix 1,21]
{ 32;421} { 1;21}
```

```
        DP>
  ⇒     repm
        REP mode
        Group is U(6)
        REP>
  ⇒     gr su6
        Group is SU(6)
        REP>
  ⇒     conv_s_to_r mix 32+2.21,43-2^3
        { 32;43} -{ 32;2^3 } +2{ 21;43} -2{ 21;2^3 }
        REP>
```

■ **M_TI**mesSfnProduct
Format: M_Ti EXPR1,EXPR2
Modes: SFN
Description: Evaluates the product of a list of monomial symmetric functions (EXPR1) and a list of S-functions (EXPR2) to produce a list of S-functions.
Example:

```
SFN>
  ⇒ m_ti 21-1,3+1^2
    { 51} +{ 42} -{ 41^2 } -{ 4} +{ 321} +{ 32} -2{ 31^3 } +{
    31^2 } -{ 31}
    -{ 2^2 1} -{ 21^3 } -{ 21} -2{ 1^5 } -{ 1^3 }
    SFN>
```

■ **M_T**o_EsymmFn
Format: M_To_E EXPR
Modes: SFN
Description: Treats EXPR as a list of monomial symmetric functions m_{ \lambda} and transforms them into a list of elementary symmetric functions e_{ \mu}.
Example:

```
SFN>
  ⇒ m_to_e 21
                - 3e_{ 3}  + e_{ 21}
       SFN>
```

■ **M_T**o_FsymmFn
Format: M_To_F EXPR
Modes: SFN
Description: Treats EXPR as a list of monomial symmetric functions m_{ \lambda} and transforms them into a list of forgotten symmetric functions f_{ \mu}.
Example:

```
SFN>
  ⇒ m_to_f 21
                - 2f_{ 3}  - f_{ 21}
                SFN>
```

■ **M_T**o_HsymmFn
Format: M_To_H EXPR
Modes: SFN
Description: Treats EXPR as a list of monomial symmetric functions m_{ \lambda} and transforms them into a list of homogeneous symmetric functions h_{ \mu}.
Example: SFN>
⇒ m_to_h 21
- 3h_{ 3} + 5h_{ 21} - 2h_{ 1^3}
SFN>

■ **M_T**o_SsymmFn
Format: M_To_S EXPR
Modes: SFN
Description: Treats EXPR as a list of monomial symmetric functions m_{ \lambda} and transforms them into a list of Schur symmetric functions.
Example: SFN>
⇒ m_to_s 21
{ 21} - 2{ 1^3}
SFN>

■ **MU**lt_CoeffByAnInt
Format: Mu INTEGER EXPR
Modes: DPM, REP, SFN
Description: Multiplies the coefficient of each term in EXPR by INTEGER, either N or -N.
Example: SFN>
⇒ mu 7 2.321-3.21
14{ 321} -21{ 21}
SFN>
⇒ mu -5 2.321-3.21
-10{ 321} +15{ 21}
SFN>

■ **MULT_L**ist
Format: MULT_L EXPR
Modes: SFN
Description: If EXPR generates a list of S-functions then the command MULT_List creates a table of the multiplicities and the number of terms having that multiplicity. See also MULT_Select.
Example: SFN>
⇒ setsv1 o 321,321
SFN>
⇒ mult_l sv1

```
Mult =1 Number of Terms =18
Mult =2 Number of Terms =7
Mult =3 Number of Terms =6
Mult =4 Number of Terms =3
SFN>
```
⇒
```
mult_s 4 sv1 t
4{ 5421} + 4{ 5321^2 } + 4{ 432^2 1}
SFN>
```

■ **MULT_N**times

Format: Mult_N INTEGER EXPR
Modes: REP, SFN
Description: Multiplies EXPR by itself INTEGER times.
Example:
```
SFN>
```
⇒
```
mult_n2,21+1
{ 42} + { 41^2} + { 3^2} + 2{ 321} + { 31^3} + 2{ 31} +
{ 2^3} + { 2^2 1^2}
+ 2{ 2^2} + 2{ 21^2} + { 2} + { 1^2}
SFN>
```

■ **MULT_P**artsByAnInt

Format: Mult_P INTEGER EXPR
Modes: SFN
Description: MULT_P multiplies every part of every partition appearing as a term in EXPR by INTEGER.
Example:
```
SFN>
```
⇒
```
mult_p 3 321+21
{ 963} + { 63}
SFN>
```

■ **MULT_S**electInList

Format: Mult_S INTEGER, EXPR [BOOLEAN]
Modes: DPM, REP, SFN
Description: Mult_S selects from the list generated by EXPR all items whose multiplicity coefficient is INTEGER or less.
If the optionnal BOOLEAN is True then it gives the items whose multiplicity coefficient is exactly the magnitude of Integer.
Example:
```
SFN>
```
⇒
```
setsv1 o 321,22-21
SFN>
```
⇒
```
mult_s -1 sv1
-{ 531} -{ 52^2 } -{ 521^2 } -{ 4^2 1} -2{ 432} -2{ 431^2
} -2{ 42^2 1}
-{ 421^3 } -{ 3^3 } -2{ 3^2 21} -{ 3^2 1^3 } -{ 32^3 } -
{ 32^2 1^2 }
SFN>
```
⇒
```
mult_s 2 sv1 t
```

```
2{ 4321}
SFN>
```

■ **MULT_SP**litIntoTwoLists

Format: Mult_Sp EXPR
Modes: SFN
Description: This command takes the EXPR and saves all the terms with odd coefficients as SVar17 and those with even coefficients as SVar18. Sometimes useful when working with memory restrictions.
Example:

```
SFN>
⇒ o 31,22-21
{ 53} +{ 521} -{ 52} -{ 51^2 } +{ 431} -{ 43} +{ 42^2 }
+{ 421^2 } -2{ 421}
-{ 41^3 } +{ 3^2 2} -{ 3^2 1} +{ 32^2 1} -{ 32^2 } -{ 321^2
}
SFN>
⇒ mult_sp last
SFN>
⇒ status
digits:true reverse:false more:false setlimit:12 pwt:200
logging:false
               10        20        30        40        50
 fns:--------------------------------------------------
svar:---------------78------------------------------ schur function
mode
SFN>
⇒ sv18
-2{ 421}
SFN>
⇒ sv17
{ 53} +{ 521} -{ 52} -{ 51^2 } +{ 431} -{ 43} +{ 42^2 }
+{ 421^2 } -{ 41^3 }
+{ 3^2 2} -{ 3^2 1} +{ 32^2 1} -{ 32^2 } -{ 321^2 }
SFN>
```

■ **MY**listOfSfns

Description: This command is used in conjunction with the command RULE and
is discussed under RULE.

■ **NL**ambda

Format: NLambda SFN
Modes: SFN
Description: For each SFN specified by a partition lambda, NL evaluates the parameter n(lambda) which is the sum of the numbers obtained by

inserting 0 into each box of the first row of the frame specified by lambda, 1 into each box of the second row, and so on until the last row.

Example: `SFN>`

⇒ `nlambda 43221`
`nlambda =  17`
`SFN>`

■ **NSK**ew

Format: Nsk INTEGER EXPR1, EXPR2

Modes: SFN

Description: INTEGER should be a positive integer N. Then NSKEW skews EXPR1 by EXPR2 N times.

Example: `SFN>`

⇒ `nskew 2 432,21`
`4{ 3} +8{ 21} +2{ 1^3 }`
`SFN>`

⇒ `nskew 3 432,21`
`8{ 0}`
`SFN>`

■ **NSTD**ise

Format: Nstd INTEGER EXPR

Modes: SFN

Description: EXPR is a list of S-functions. Nstdise applies the S-function standardisation rules to the first INTEGER parts of each partition in the EXPR list to produce a new list. INTEGER must be >1 else an error message is produced.

Example: `SFN>`

⇒ `nstd 3,1415`
`-{ 3215}`
`SFN>`

■ **ONSC**alar

Format: Onsc EXPR1, EXPR2

Modes: REP

Description: EXPR1 and EXPR2 are to be treated as lists of O(N) tensor irreps.
Onscalar compares the two lists and for any common pair of irreps forms the product of the multiplicities and adds them, and then writes to screen # scalars = . This command was introduced in the enumeration of Weyl scalars.

Example: `REP>`

⇒ `gr o5`
`Group is O(5)`

```
        REP>
   ⇒    p 21,21
        [42] + [41] #  + [4] + [3^2 ] + 2[32] #  + 3[31] + [3] #
        + 2[2^2 ] + 3[21] #  + 2[2] + 2[1^2 ] + [1] #  + [0]
        REP>
   ⇒    p 3,21
        [51] + [42] + [41] #  + [4] + [32] #  + 2[31] + [2^2 ] +
        [21] #
        + [2] + [1^2 ]
        REP>
   ⇒    onsc p21,21 , p3,21
        #  of scalars = 20
        REP>
```

■ **O_P**fnProduct

Format: O_P EXPR1, EXPR2
Modes: SFN
Description: Forms the outer product of two lists of P-functions.
Example:

```
SFN>
   ⇒    o_p 321,42
                    P_{ 741}  + P_{ 732}  + P_{ 651}  + 2P_{ 642}
        + P_{ 6321}  + P_{ 543}  + P_{ 5421}
                    SFN>
```

■ **O_Q**fnProduct

Format: O_Q EXPR1,EXPR2
Modes: SFN
Description: Forms the outer product of two lists of Q-functions.
Example:

```
SFN>
   ⇒    o_q 321+2.1,42-3.2
        4Q_{ 741} +4Q_{ 732} +4Q_{ 651} +8Q_{ 642} +2Q_{ 6321} +4Q_{
        543} +2Q_{ 5421}
        -6Q_{ 521} +4Q_{ 52} -6Q_{ 431} +4Q_{ 43} +2Q_{ 421} -12Q_{
        3} -6Q_{ 21}
        SFN>
```

■ **O_R**estrict

Format: O_Restrict INTEGER EXPR1,EXPR2
Modes: SFN
Description: Evaluates the S-function outer product of EXPR1 and EXPR2 using
the Littlewood-Richardson rule but with the maximum number of parts of each partition being restricted to INTEGER.
Example:

```
SFN>
   ⇒    o_r 3 32-2.21,21+3.1^2
        { 53} +{ 521} +{ 4^2 } +2{ 431} +3{ 43} +{ 42^2 } +3{ 421}
        -2{ 42}
```

```
-2{ 41^2 } +{ 3^2 2} +3{ 3^2 1} -2{ 3^2 } -4{ 321} -6{ 32}
-6{ 31^2 }
-2{ 2^3 } -6{ 2^2 1}
SFN>
```

■ **O**_sfnProduct

Format: O EXPR1,EXPR2
Modes: SFN
Description: Evaluates the S-function outer product of EXPR1 and EXPR2 using
the Littlewood-Richardson rule.
Example: SFN>

⇒
```
o 32-2.21, 21+3.1^2
{ 53} +{ 521} +{ 4^2 } +2{ 431} +3{ 43} +{ 42^2 } +{ 421^2
} +3{ 421}
-2{ 42} -2{ 41^2 } +{ 3^2 2} +{ 3^2 1^2 } +3{ 3^2 1} -2{
3^2 } +{ 32^2 1}
+3{ 321^2 } -4{ 321} -6{ 32} -2{ 31^3 } -6{ 31^2 } -2{ 2^3
} -2{ 2^2 1^2 }
-6{ 2^2 1} -6{ 21^3 }
SFN>
```

■ **PARITY**sequence

Format: PARITY SEQUENCE
Modes: SFN
Description: SEQUENCE is a sequence of positive integers. PARITY is EVEN if the sequence can be ordered as an increasing sequence by an even permutation otherwise PARITY is ODD.
Example: SFN>

⇒
```
parity1112233254
            parity of sequence is odd
            SFN>
```

■ **PAUSE**

Format: Pause
Modes: DPM, REP, SFN
Description: Halts operation until a key is pressed. Useful as an interrupt in functions.

■ **PL**ethysm

Format: Pl EXPR1, EXPR2
Modes: REP, SFN
Description: Calculates the plethysm of EXPR1 \otimes EXPR2.
In the SFN MODE EXPR1 and EXPR2 may be lists of S-functions while in the REP MODE EXPR1 may be tensor reps of U(n), SU(n), SO(2k+1), O(n) or G(2).
Plethysms may also be evaluated for single irreps of Sp(2n,R) or

SO^*(2n).
In that case first setlimit to say 12 and set_pwt to say 12
and terms to weight 12 will be computed. Higher values of
setlimit and set_pwt will yield higher weight terms but will
take longer to compute.

Example: `REP>`

⇒ `gr g2`
`Group is G(2)`
`REP>`

⇒ `pl 21,2`
`(42) + (2) + (0)`
`REP>`

⇒ `sfn`
`Schur Function Mode`
`SFN>`

⇒ `pl 21,2`
`{ 42} + { 321} + { 31^3} + { 2^3}`
`SFN>`

■ **PLG**

Format: PLG rep
Modes: REP
Description: The group must be set to be G2. Then PLG acts on a single REP
of
G2 to produce an EXPR consisting of a list of irreps of G2,
together with appropriate coefficients, such that the plethysm
(10) \otimes EXPR returns just REP.
Example:-REP>
-> gr g2
Group is G(2)
REP>
-> plg 21
(1^2) - (1)
REP>
-> pl 1,last
(21)
REP>

■ **P**roductKronecker

Format: P EXPR1, EXPR2
Modes: DPM, REP
Description: Calculates the product EXPR1 \times EXPR2 as a sum of
irreducible representations of the set groups. In DPMode
p [a*b],[c*d] is equal to [p a,c * p b,d], that is the direct
product of the product of the reps a and c of group 1 with

the product of the resp b and d of group 2.
The groups and classes of representations for which products are available is given in Table A.4 of the SCHUR manual.

Example:
```
DP>
```
⇒
```
gr2so5sp4
Groups are SO(5) * Sp(4)
DP>
```
⇒
```
p [2 * 1],[1 * 2]
[3]<3> + [3]<21> + [3]<1> + [21]<3> + [21]<21> + [21]<1>
+ [1]<3> + [1]<21> + [1]<1>
DP>
```
⇒
```
repm
REP mode
Group is SO(5)
REP>
```
⇒
```
gr su3
Group is SU(3)
REP>
```
⇒
```
p 32-2.21,1^2+3.1
{ 43} +3{ 42} +3{ 3^2 } -2{ 32} -5{ 31} -5{ 2^2 } +3{ 21}
-2{ 2} -2{ 1^2 } -6{ 1}
REP>
```

Notes:

Table A.4: Groups and classes of representations available for calculating Kronecker products in SCHUR.

Group	Classes of representations
Un	{ \mubar; 1}, { 1}
SUn	{ 1}
On	[1]#, [1], [s;1]#, [s;1]
SO{ 2k+1}	[1], [s;1]
SO{ 2k}	[1], [1]+/-, [s;1]+/-
Sp{ 2k}	<1>
Sp{ 2k}(R)	<s\kappar (1)>, <\kappa (1)>
SO*(2n)	[k (1)]
G2	(1)
F4	(1), (s;1)
E6	(1)
E7	(1)
E8	(1)
Sn	{ 1}, { s;1}, { s;1}+/-

■ **PROP**ertyOfRepList

Format: Prop EXPR

Modes: REP
Description: Computes dimensions, second order Casimir and Dynkin indices and
Dynkin labels for irreps for the set groups. The available groups are those listed in Table A.4 of the SCHUR manual.
Notice that the eigenvalue of the second order Casimir operator is only given in those cases for which EXPR is a single REP, with or without multiplicity.
Example:

```
REP>
⇒ gr e8
Group is E(8)
REP>
⇒ prop21^7
<dynkin label> (10000000)
dimension=248   60*2nd-casimir=60
2nd-dynkin = 1
REP>
⇒ prod21^7,21^7
(42^7) + (31^6) + (21^7) + (21) + (0)
REP>
⇒ prop last
<dynkin label> (20000000) + (01000000) + (10000000) + (00000010)
+ (00000000)
dimension = 61504          2nd-dynkin = 496
REP>
```

■ **P_TO_D**label
Format: P_to_D EXPR
Modes: REP
Description: Converts the Dynkin labels for the irreps in EXPR to partition labels. The group must first be set to be one of the simple Lie groups: SU(n), SO(n), Sp(2n), E6, E7, E8, F4 or G2.
Example:

```
REP>
⇒ gr e8
Group is E(8)
REP>
⇒ p_to_d 21^7
<dynkin label>(10000000)
REP>
⇒ p 21^7,21^7
(42^7 ) +(31^6 ) +(21^7 ) +(21) +(0)
REP>
⇒ p_to_d last
```

```
<dynkin label>(20000000) +(01000000) +(10000000) +(00000010)
+(00000000)
REP>
```

■ **P_T**o_SsymmFn

Format: P_To_S EXPR
Modes: SFN
Description: Expands a power sum symmetric function EXPR as a sum of S-functions.
Example:
```
SFN>
```
⇒
```
p_to_s 31
              { 4} - { 2^2} + { 1^4}
              SFN>
```

■ **QEXP**and

Format: Qexp EXPR
Modes: SFN
Description: Gives a polynomial expansion of the first SETLIMit terms in powers of q of the sum of all S-functions in EXPR with their variables specialised to (1,q,q^2,...)
Example:
```
SFN>
```
⇒
```
qexp 21
q +2q^2 +3q^3 +5q^4 +7q^5 +9q^6 +12q^7 +15q^8 +18q^9 +22q^10
+26q^11 +30q^12
{ 21}
SFN>
```

■ **QQEX**pandSpecialSeries

Format: Qqex EXPR
Modes: SFN
Description: Gives a polynomial expansion of the first SETLIMit terms in powers of q of the sum of all supersymmetric S-functions in EXPR with their variables specialised to (q,q^2,.../q,q^2,...)
Example:
```
SFN>
```
⇒
```
qqex 21-1
-2q -2q^2 +6q^4 +14q^5 +24q^6 +38q^7 +54q^8 +72q^9 +94q^10
+118q^11 +144q^12
{ 21} -{ 1}
SFN>
```

■ **QQSE**ries

Format: Qqse SERIES
Modes: SFN
Description: Gives a polynomial expansion of the first SETLIMit terms in powers of q of the sum of all supersymmetric S-functions in SERIES with their variables specialised to (q,q^2,.../q,q^2,...)". SERIES is any one of the S-function series A,B,C,D,E,F,G,H,L,M,P,Q.

Example: `SFN>`
⇒ `qqse b`
`1 +2q^2 +4q^3 +8q^4 +16q^5 +32q^6 +60q^7 +114q^8 +212q^9`
`+384q^10 +692q^11 +1232q^12`
`SFN>`

■ **QSAME**
Format: QSAME qfn1,qfn2
Modes: SFN
Description: qfn1,qfn2 are Q-functions specified by partitions. QSAME produces a list of Q-functions that could but do not necessarily arise in the Q-function product qfn1.qfn2. Most "dead" partitions have been removed from the list. See also DEAD.
Example: `SFN>`
⇒ `qsame 32,765431`
`Q_{ 10 85431} + Q_{ 10 76431} + Q_{ 10 75432} + Q_{ 10 654321}`
`+ Q_{ 986431} + Q_{ 985432}`
`+ Q_{ 976531} + Q_{ 976432} + Q_{ 9754321} + Q_{ 876541}`
`+ Q_{ 876532} + Q_{ 8764321}`
`SFN>`

■ **QSER**ies
Format: qser Series
Modes: SFN
Description: Gives a polynomial expansion of the first SETLIMit terms in powers of q of the sum of all S-functions in SERIES with their variables specialised to (q,q^2,...)". SERIES is any one of the S-function series A,B,C,D,E,F,G,H,L,M,P,Q.
Example: `SFN>`
⇒ `qser A`
`1 - q^3 - q^4 -2q^5 -2q^6 -2q^7 - q^8 + 2q^10 + 5q^11 +`
`7q^12`
`SFN>`

■ **Q_TO_S**symmFn
Format: Q_To_S EXPR
Modes: SFN
Description: Treats EXPR as a list of Q-functions and transforms them into a list of S-functions of type S(x,-1).
Example: `SFN>`
⇒ `q_to_s 31`
`- { 4} + { 31}`
`SFN>`

■ **RACAH**notation
Format: Racah EXPR
Modes: REP
Description: Converts an EXPR of G(2) irreps labelled in the SU(3) basis into

a list of G(2) irreps labelled in Racah's SO(3) basis.
N.B. This operation does not permit further operations on the EXPR. See FRACAHnotation for the inverse transformation.

Example:
```
REP>
```
⇒
```
gr g2
Group is G(2)
REP>
```
⇒
```
prod31,21
(52) + (41) + (4) + 2(31) + (3) + (2) + (1)
REP>
```
⇒
```
racah last
(4) + (32) + (31) + (3) + 2(21) + (2) + (1)
REP>
```

■ **RAISEI**nverseOp

Format: RaiseI s,EXPR (s = +/- 1)
Modes: SFN
Description: RaiseI is the inverse of the operator RaiseOp with s being the phase. The result is standardised as for S-functions. Formally RaiseI s gives \prod_{ i<j}1/(1 + sR_{ ij}).

Example:
```
SFN>
```
⇒
```
raisei 1,21
             { 3} + { 21}
             SFN>
```
⇒
```
raisei -1,21
             -{ 3} + { 21}
             SFN>
```

■ **RAISE**op

Format: Raise s, EXPR (s = +/- 1)
Modes: SFN
Description: Raise is the raising operator defined with the phase s. EXPR is a list of S-functions. If s = 1 then the raising operator acts on EXPR and then standardises the resultant as for Q-functions whereas for s = -1 the resultant is standardised as for S-functions.

Example:
```
SFN>
```
⇒
```
raise1,21
             { 3} + { 21}
             SFN>
```
⇒
```
raise1,2^2
             { 31}
             SFN>
```

```
    ⇒     raise-1,2^2
                        - { 31} + { 2^2}
                        SFN>
```

■ **RD_I_Q**fnProduct
Format: Rd_I_Q EXPR1, EXPR2
Modes: SFN
Description: Forms the reduced inner product of an S-function EXPR1 with a Q-function EXPR2 to yield a list of reduced Q-functions.
Example: SFN>

```
    ⇒     rd_i_q 1^2,2
                        2Q_<4>  + 2Q_<31>  + 6Q_<3>  + 3Q_<21>  +
          8Q_<2> + 4Q_<1>  + 2Q_<0>
                        SFN>
    ⇒     rd_i_q 2,1^2
                        zero
                        SFN>
```

■ **RD_I**_sfnProduct
Format: Rd_I EXPR1, EXPR2
Modes: SFN
Description: Forms the product of two lists of S(n) irreps in reduced notation.
Example: SFN>

```
    ⇒     rd_i 2,1
                        <3> + <21> + <2> + <1^2> + <1>
          SFN>
```

■ **RD_R**aiseOp
Format: Rd_R s, EXPR (s = +/- 1)
Modes: SFN
Description: As for RaiseOp except for reduced notation.
Example: SFN>

```
    ⇒     rd_r 1,21
                        <3> + <21> + 2<2> + <1>
                        SFN>
    ⇒     rd_r -1,21
                        - <3> + <21> - <1^2 > + <1>
                        SFN>
```

■ **RD_R**aiseOp
Format: Rd_R s, EXPR (s = +/- 1)
Modes: SFN
Description: As for RaiseOp except for reduced notation.
Example: SFN>

```
    ⇒     rd_r 1,21
```

```
                    <3> + <21> + 2<2> + <1>
                    SFN>
⇒   rd_r -1,21
                    - <3> + <21> - <1^2 > + <1>
                    SFN>
```

■ **READF**nFromDisk
Format: ReadF INTEGER 'filename'
Modes: DPM
Description: Reads function INTEGER (=N), N = 1... 20, from a disk file 'filename'. Whilst reading the function from disk a "=-" will appear for each line read.
Example: DP>

```
⇒   readf 1 'test.fn'
                    =-
                    =-
                    =-
                    .
                    .
                    =-
                    DP>
```

■ **REM**ark
Description: This command allows the insertion of remarks into logfiles and functions. If a Remark is encountered inside a function then the line is echoed if the function is written to screen with WRfnTo-Screen.
REMark may also be used to insert REMarks into a LOGfile.

■ **REP**mode
Format: Rep
Modes: DPM, SFN
Description: Allows the user to go from the DPMode and SFNmode into the REPmode.

■ **RET**urn
Format: RETurn
Modes: DPM, REP, SFN
Description: Used in functions to insert a blank line between sets of output data.

■ **R**iemannList
Format: RiemannL N
Modes: SFN

Description: Creates a list of S-functions corresponding to the partitions of the integer N with no parts = 1.
Example: `SFN>`
⇒ `riemannl 8`

```
                { 8} + { 62} + { 53} + { 4^2} + { 42^2} + {
3^2 2} + { 2^4}
                SFN>
```

■ **RIEMANNP**lethList
Format: RiemannP N
Modes: SFN
Description: A list of S-functions is formed as for RiemannL N and then the operation RiemannP is performed on that list. The operation RiemannP is related to a problem posed by S. A. Fulling in relationship to the determination of the independent scalars that can be formed by contraction of the Riemann tensor and its covariant derivatives.
Example: `SFN>`
⇒ `riemannp 6`

```
            { 6^2} + { 642} + 2{ 64} + { 62^3} + 2{ 62^2}
+ { 62} + { 4^3} + 2{ 4^2 2^2}
    + 2{ 4^2 2} + { 42^4} + 2{ 42^3} + { 2^6}
                SFN>
```

■ **RIEMANNS**calarsOrderN
Format: RiemannS SFN
Modes: SFN
Description: Takes a single S-function and replaces each part p by { p2}. If { p2} occurs k times then the plethysm { p2} \otimes { k} is formed. The resulting lists are combined via the Littlewood-Richardson rule and all partitions involving odd parts discarded. Thus { 42^2} becomes { 42}.({ 2^2} \otimes { 2}).
Example: `SFN>`
⇒ `riemanns 42^2`

```
                { 86} + 2{ 842} + { 82^3} + 2{ 6^2 2} + 2{
64^2} + 6{ 642^2} + 2{ 62^4}
     + 2{ 4^3 2} + 2{ 4^2 2^3} + { 42^5}
                SFN>
```

■ **RM_EVENPARTS**
Format: RM_EVENPARTS List
Modes: SFN
Description: Removes from List all S-functions involving at least one even part.
Example: `SFN>`
⇒ `o21,21`

```
                { 42} + { 41^2 } + { 3^2 } + 2{ 321} + { 31^3 } +
        { 2^3 } + { 2^2 1^2 }
                SFN>
    ⇒   rm_evenparts last
                        { 3^2 } + { 31^3 }
             SFN>
```

■ **RM_EVENR**kSfnsOnly
Format: Rm_EvenR EXPR
Modes: SFN
Description: Removes all partitions from EXPR whose Frobenius rank is even.
Example:
```
SFN>
    ⇒   rm_evenr { 1} + { 2} + { 21} + { 42}
                        { 21} + { 2} + { 1}
                        SFN>
```

■ **RM_EVENW**tInList
Format: Rm_EvenW EXPR
Modes: REP, SFN
Description: Removes all partitions from EXPR whose weight is even.
Example:
```
SFN>
    ⇒   rm_Evenw { 1} + { 2} + { 21} + { 42}
                        { 21} + { 1}
                        SFN>
```

■ **RM_F**irstPartOfSfn
Format: Rm_F EXPR
Modes: SFN
Description: Removes the first part of an S-function.
Used in the reduced notation for the
irreps of the symmetric group S(n).
Example:
```
SFN>
    ⇒   o 21,21
                        { 42} + { 41^2} + { 3^2} + 2{ 321} + { 31^3}
        + { 2^3} + { 2^2 1^2}
                        SFN>
    ⇒   rm_f last
                        <3> + <2^2> + <21^2> + 2<21> + <2> + <1^3>
        + <1^2>
                        SFN>
    ⇒   make6last
                        { 42} + { 41^2} + { 3^2} + 2{ 321} + { 31^3}
        + { 2^3} + { 2^2 1^2}
                        SFN>
```

■ **RM_G**roup
Format: Rm_G grN EXPR
Modes: DPM

Description: Eliminates the N-th group from DPrep EXPR.
This can be useful in branching chains where
groups may become redundant.
Example: DP>

```
⇒     gr2u1su5
                    Groups are U(1) * SU(5)
                    DP>
⇒     [1*31] + [2*2]
                    { 2}{ 2} + { 1}{ 31}
                    DP>
⇒     rm_g 1,last
                    Group is SU(5)
                    { 31} + { 2}
                    DP>
```

■ **RM_NMP**arts
Format: Rm_NMParts N, M, EXPR
Modes: REP, SFN
Description: Removes from EXPR all partitions with any part between N and M.
Example: REP>

```
⇒     gr U8
                    Group is U(8)
                    REP>
⇒     pl 0+2+4+6,3
                 { 18 } + { 16 2} + { 16 } + { 15 3} + { 15 1}
+ { 14 4} + { 14 2^2 }
       + 2{ 14 2} + 2{ 14 } + { 13 5} + { 13 41} + 2{ 13
3} + { 13 21} + 2{ 13 1}
       + 2{ 12 6} + { 12 42} + 3{ 12 4} + { 12 31} + { 12
2^2 } + 4{ 12 2}
       + 3{ 12 } + { 11 61} + { 11 52} + 2{ 11 5} + 2{ 11
41} + { 11 32}
       + 3{ 11 3} + 2{ 11 21} + 2{ 11 1} + { 10 8} + { 10
71} + { 10 62}
       + 3{ 10 6} + 2{ 10 51} + { 10 4^2 } + 2{ 10 42} +
5{ 10 4} + 2{ 10 31}
       + 2{ 10 2^2 } + 5{ 10 2} + { 10 1^2 } + 3{ 10 } +
{ 97} + { 963}
       + 2{ 961} + { 952} + 3{ 95} + { 943} + 3{ 941} +
{ 932} + 4{ 93}
       + 2{ 921} + 3{ 91} + { 8^2 2} + { 8^2 } + { 871}
+ { 864} + 2{ 862}
       + 4{ 86} + { 853} + 2{ 851} + { 84^2 } + 3{ 842}
+ 5{ 84}
       + 2{ 831} + 2{ 82^2 } + 5{ 82} + 3{ 8} + { 763} +
2{ 761} + { 752}
```

```
          + 2{ 75} + { 743} + 3{ 741} + { 732} + 3{ 73} + 2{
721} + 2{ 71}
          + { 6^3 } + { 6^2 4} + 2{ 6^2 2} + 3{ 6^2 } + { 651}
+ { 64^2 }
          + 2{ 642} + 4{ 64} + { 631} + 2{ 62^2 } + 4{ 62}
+ 3{ 6} + { 541}
          + { 53} + { 521} + { 51} + { 4^3 } + { 4^2 2} + 2{
4^2 } + { 42^2 }
          + 2{ 42} + 2{ 4} + { 2^3 } + { 2^2 } + { 2} + { 0}
            REP>
⇒     rm_nmp7,18last
            { 6^3 } + { 6^2 4} + 2{ 6^2 2} + 3{ 6^2 } + {
651} + { 64^2 }
          + 2{ 642} + 4{ 64} + { 631} + 2{ 62^2 } + 4{ 62}
+ 3{ 6} + { 541}
          + { 53} + { 521} + { 51} + { 4^3 } + { 4^2 2} + 2{
4^2 } + { 42^2 }
          + 2{ 42} + 2{ 4} + { 2^3 } + { 2^2 } + { 2} + { 0}
            REP>
```

■ **RM_ODDPARTS**

Format: RM_ODDPARTS List
Modes: SFN
Description: Removes from List all S-functions involving at least one odd part.
Example:

```
SFN>
⇒     o21,21
        { 42} + { 41^2 } + { 3^2 } + 2{ 321} + { 31^3 } +
{ 2^3 } + { 2^2 1^2 }
        SFN>
⇒     rm_oddparts last
        { 42} + { 2^3 }
       SFN>
```

■ **RM_ODDR**kSfnsOnly

Format: Rm_OddR EXPR
Modes: SFN
Description: Removes all partitions from EXPR whose Frobenius rank is odd.
Example:

```
SFN>
⇒     rm_oddr { 1} + { 2} + { 21} + { 42}
                { 42}
                SFN>
```

■ **RM_ODDW**tInList

Format: Rm_OddW EXPR
Modes: REP, SFN
Description: Removes all partitions from EXPR whose weight is odd.
Example:

```
SFN>
⇒     rm_oddw { 1} + { 2} + { 21} + { 42}
```

```
                    { 42} + { 2}
                    SFN>
```

■ **RM_P**artitionFromSfn

Format: Rm_P PARTITION, EXPR
Modes: SFN
Description: Removes the PARTITION from each member of the EXPR. The inverse of the command ATtachPartitionToSfn.
Example:

```
SFN>
⇒   rm_p 21,52
                    { 31}
                    SFN>
⇒   rm_p 21,41^2
                    { 201}
                    SFN>
⇒   std rm_p 21,41^2
                    zero
                    SFN>
```

■ **RM_PARTS**equalN

Format: Rm_Parts N, EXPR
Modes: REP, SFN
Description: Removes from EXPR all partitions with any part equal to N. If N = -1 then all partitions with any odd part are removed while if N = -2 then all partition with any even part are removed.
Example:

```
SFN>
⇒   rm_parts 2,4321+3^31+21+1^6
                    { 3^3 1} + { 1^6}
                    SFN>
⇒   rm_parts -1,42+321+421
                    { 42}
          SFN>
```

■ **R**m_RepeatedPartsSfns

Format: Rm_R EXPR
Modes: SFN
Description: Removes from the EXPR all S-functions involving repeated parts.
Example:

```
SFN>
⇒   321^3+2.321+21+2+ 1^2 + 0
          { 321^3 } +2{ 321} +{ 21} +{ 2} +{ 1^2 } +{ 0}
                    SFN>
⇒   rm_r last
                    2{ 321} + { 21} + { 2} + { 0}
                    SFN>
```

■ **RM_SO**nEvenLabel

Format: Rm_so EXPR
Modes: REP
Description: Removes the labels +/- from SO(2k) irreps . Used in Functions.

■ **RM_U**oneWtOverMax
Format: Rm_U INTEGER, grN, EXPR (DPM)
Rm_U INTEGER, EXPR (REP)
Modes: DPM, REP
Description: Removes from EXPR all representations involving a U(1) group rep whose integer weight exceeds INTEGER. Originally introduced in connection with the Macdonald identities.

■ **RP_F**irstPartBySpin
Format: Rp_F grN EXPR (DPM)
Rp_F EXPR (REP)
Modes: DPM, REP
Description: Removes the first part of each partition and replaces it by a spin index
Example: DP>
⇒ `gr2so5so7`
```
Groups are SO(5) * SO(7)
DP>
```
⇒ `rp_f gr1[21 * 321]`
```
[s;1][321]
DP>
```

■ **RP_R**epOrSfnByWt
Format: Rp_R group grN EXPR (DPM)
Rp_R EXPR (REP),(SFN)
Modes: DPM, REP, SFN
Description: This command replaces a rep or sfn by its weight. This command is especially useful with RULE when U(1) reps are involved.
Example: DP>
⇒ `gr2su8so8`
```
Groups are SU(8) * SO(8)
DP>
```
⇒ `branch1,8gr1[321*1]`
```
Groups are SO(8) * O(8)
[1][321] + [1][31] + [1][2^2] + [1][21^2] +
[1][2] + [1][1^2]
DP>
```
⇒ `rp_r gr2last`
```
[1][6] + 3[1][4] + 2[1][2]
```

```
                    DP>
```

■ **RP_S**fnCoeffByInt

Format: Rp_S INTEGER EXPR
Modes: SFN
Description: Returns EXPR with every coefficient replaced by the same INTEGER which must not be zero
Example:
```
SFN>
⇒   -4.4321+2.321+3.21
                    -4{ 4321} + 2{ 321} + 3{ 21}
                    SFN>
⇒   rp_s 12 last
                    12{ 4321} + 12{ 321} + 12{ 21}
                    SFN>
```

■ **RSAME**wtSfnList

Format: RSAME Sfn
Modes: SFN
Description: Creates a list of all S-functions which have the same weight as Sfn and in reverse-lexicographic order less than or equal to Sfn.
Example:
```
SFN>
⇒   rsame 321
                    { 321} + { 31^3 } + { 2^3 } + { 2^2 1^2 } +
{ 2 1^4 } + { 1^6 }
                    SFN>
```

■ **RULE**

Format: (1). Rule EXPR Sum Operations With
(2). Rule EXPR1 Sum MyList Operations With EXPR2
(3). Rule EXPR1 Operation With EXPR2
Modes: DPM
Description: RULE makes use of the DPrep structure by allowing S-function operations on the partitions labelling the representations of the component groups. This is the most complex, and also most powerful, command in SCHUR and is fully discussed in the Advanced Tutorial:Using Rule. Use of this command should not be attempted until the user has become familiar with most of the commands and structures in SCHUR.

■ **RV**ar

Format: Rv N
Modes: REP
Description: Contains the rep variable N, N=1...50, set with SETRepVar. Use as a rep EXPR when required.

■ **SAME**wtSfnList
Format: Same Sfn
Modes: SFN
Description: Creates a list of all S-functions which have the same weight as Sfn.
Example: `SFN>`
⇒ `same 4`
`{ 4} + { 31} + { 2^2} + { 21^2} + { 1^4}`
`SFN>`

■ **SAVE**setVar
Format: Save RVar INTEGER 'filename' (REP)
Save SVar INTEGER 'filename' (SFN)
Modes: REP, SFN
Description: If INTEGER is omitted then all non-zero xVars are saved to the file 'filename'. If INTEGER is included then only xVar INTEGER is saved. Saved files contain 'type' information distinguishing SVars from RVars so that entering
> save SVAR, 'test'
> Load RVAR, 'test'
will generate an error. The files are pure ASCII so they are available for external modification: the user can generate a LOADable file outside SCHUR or use a Saved file outside.
if SB_LISToutput is TRUE, the output is in a list format compatible to Maple.

■ **SB_B**ell
Description: When TRUE, BELL writes and ASCII 7 before getting the next command line in all modes except BRM. BELL is turned on by entering the command Sb_B TRUE

■ **SB_CONJ**ecture
Description: When FALSE, Kronecker products for the non-compact group Sp(2n,R)
are computed using the conjecture of King and Wybourne. When TRUE
the Kronecker products are calculated directly. The default value is set as TRUE.

■ **SB_CU**t
Format: Sb_Cu n
Modes: DPM, REP, SFN

Description: Gives an output in power notation whenever multiplicities of a part of a partition exceeds n and Sb_POW TRUE. The default value is set as n = 1.

Example:
```
SFN>
⇒   sb_pow false
                SFN>
⇒   32211111+321111111+222111111
                { 32211111} + { 321111111} + { 222111111}
                SFN>
⇒   sb_pow true
                SFN>
⇒   sb_cut 3
                SFN>
⇒   last
                { 3221^5} + { 321^7} + { 2221^6}
        SFN>
```

■ **SB_D**igits

Format: Sb_D [TRUE:FALSE]

Modes: DPM, REP, SFN

Description: Sb_D is a Boolean flag. When Sb_D is TRUE all numbers occurring in a partition (or a multiplicity) are assumed to be single digits. The default is set TRUE. When partition numbers are larger than 9 they are entered by prefixing them with a ! and terminating with a space. When Sb_D is FALSE every number must be followed by a space. Consider the inputting, in the SFNmode, of the partition (15 13 11 9 7 5) with a multiplicity of 137. If Sb_D is TRUE it could be entered as !137.!15!13!11 975 whereas if Sb_D was FALSE it could be entered as 137.15 13 11 9 7 5. In both cases the result will appear on the screen as 137{ 15 13 11 975}.

■ **SB_DIM**ension

Description: Sb_Dim is a Boolean whose default value has been set as TRUE which permits the command DIMension to calculate the dimensions of representations using the Weyl formulae. Setting Sb_Dim FALSE permits the command DIMension to calculate the corresponding dimensions using the hooklength formulae. The Boolean may be set in any of the three modes SFN, REP or DPM.

■ **SB_LIST**output

Format: sb_list TRUE:FALSE

Modes: REP, SFN

Description: Setting SB_LIST to TRUE forces variable saving (see SAVEset-Var)
to be in a Maple compatible format (see SchurToMaple functions in the dat directory). The default is FALSE.

■ **SB_M**ore
Format: Sb_M TRUE:FALSE
Modes: DPM, REP, SFN
Description: Sb_M is a Boolean which once set acts within all modes. If Sb_M FALSE is entered the screen freely scrolls while if Sb_M TRUE is entered the screen is filled and awaits a keypress to give more output. The default is set as FALSE.

■ **SB_P**owerNotation
Format: Sb_P TRUE:FALSE
Modes: DPM, REP, SFN
Description: Sb_P TRUE allows both the input and the output to be in power (or exponent) notation. Setting Sb_P FALSE gives output without exponent notation. The default setting is Sb_P TRUE. The setting persists in all modes including the BRMode until it is changed by the user.
See also Sb_CUt.
Example:

```
SFN>
⇒   o 21^2,1^2
                { 321} + { 31^3} + { 2^3} + { 2^2 1^2} + {
21^4}
                SFN>
⇒   sb_power false
                SFN>
⇒   last
                { 321} + { 3111} + { 222} + { 2211} + { 21111}
     SFN>
```

■ **SB_PROG**ress
Format: Sb_prog TRUE:FALSE
Modes: DPM, REP, SFN
Description: The default setting is Sb_prog TRUE and during a calculation the cursor rotates indicating the calculation is progressing. Setting Sb_prog FALSE turns off the display.

■ **SB_Q**fn
Format: Sb_Q TRUE:FALSE
Modes: SFN
Description: If Sb_Q is set TRUE then any partitions entered

in the SFNMode will be standardised as if they were Q-functions. The default value is FALSE.

■ **SB_RD**notation
Format: Sb_Rd TRUE:FALSE
Modes: SFN
Description: Putting Sb_Rd TRUE forces SCHUR to output reduced notation. The default setting is Sb_Rd FALSE. The user should always return Sb_Rd to its default setting to ensure correct results.

■ **SB_REV**erseOrder
Format: SB_REV TRUE:FALSE
Modes: DPM, REP, SFN
Description: Schur outputs lists of expressions that are normally sorted antilexicographically with respect to the alphabet 1,2,3,.... In the DPMode this will be for the first group in the list of product groups. Setting SB_REVerseOrder TRUE reverses this order. FALSE is the default setting.
Example:
```
SFN>
⇒ o21,3
                { 51} + { 42} + { 41^2} + { 321}
                SFN>
⇒ sb_rev true
                SFN>
⇒ last
                { 321} + { 41^2} + { 42} + { 51}
                SFN>
```

■ **SB_T**exOutput
Format: Sb_T TRUE:FALSE
Modes: DPM, REP, SFN
Description: Setting Sb_T TRUE forces output from all modes to be in TeX format. The default is Sb_T FALSE which gives normal output. Mainly for producing output in tabular form. See also COLumns and LINES as well as Appendix D Producing TeX Tables.

■ **SCALARI**nner
Format: SCALARI n,k
Modes: REP
Description: Computes the sum of the squares of the irreps of S(n) for all partitions up to length k. If k is set as

negative then the sum of squares of the irreps of S(n)
is computed for all partititons of length k.

Example: `REP>`

⇒ `gr S4`

```
                Group is S(4)
                REP>
```

⇒ `scalari4,3`

```
                4{ 4} + 2{ 31} + 3{ 2^2 } + 2{ 21^2 } + { 1^4
}
                REP>
```

⇒ `scalari4,-3`

```
                { 4} + { 31} + { 2^2 } + { 21^2 }
                REP>
```

■ **SCHAR**

Format: SCHAR REP, CLASS

Modes: REP

Description: REP is a single irrep of the symmetric group S(N) and
CLASS is a single class of the symmetric group S(N).
SCHAR returns the characteristic as an integer.

Example: `REP>`

⇒ `gr s56`

```
                Group is S(56)
                  REP>
```

⇒ `schar !48 8,86^35^24^32^4`

```
                  characteristic =  27
                  REP>
```

■ **SERIESTE**rmsThatSkew

Format: SeriesTe EXPR, Series

Modes: SFN

Description: Finds all the members of the SFN Series which can skew
with the SFN EXPR.

Example: `SFN>`

⇒ `serieste 321 + 21,m`

```
                { 3} + { 2} + { 1} + { 0}
                SFN>
```

■ **SER**iesToIntWt

Format: Ser N, Series

Modes: SFN

Description: Finds all the members of the SFN Series of weight <= N.

Example: `SFN>`

⇒ `ser 4,m`

```
                { 4} + { 3} + { 2} + { 1} + { 0}
                SFN>
```

■ **SETF**n

Format: Setf N
Modes: DPM
Description: Used to set function N, N = 1... 50, during a SCHUR session. It prompts you to enter each line with a "=-". STop signals completion. If you make a mistake there is no way of editing the function. Either retype the function or save it on disk with WRFNTODisk and use a text editor to correct it. If editing a function saved on disk remember to make the last line STop (with a carriage return). For an extensive discussion see the tutorial on writing functions.
Example:

```
DP>
⇒   setfn1
                    =-
⇒   rem evaluates the sfn outer product sv1.sv2
                    =-
⇒   sfn
                    =-
⇒   enter sv1
                    =-
⇒   enter sv2
                    =-
⇒   o sv1,sv2
                    =-
⇒   stop
                    DP>
⇒   fn1
                    Schur Function Mode
                    enter sv1
⇒   21
                    enter sv2
⇒   2^2
                    { 43} + { 421} + { 3^2 1} + { 32^2} + { 321^2}
+ { 2^3 1}
   SFN>
```

■ **SETLIM**it

Format: SETLIMit INTEGER
Modes: DPM, REP, SFN
Description: Sets the variable SETLIMIT which is used in non-compact group calculations. The default value has been set at 12.

■ **SET_PWT**

Format: SET_PWT INTEGER

Modes: REP, SFN
Description: Used in plethysm calculations. Computes only those plethysms of weight Integer or less. Default value set at 200.

- **SETRV**ar
  Format: SetRV N, EXPR
  Modes: REP
  Description: Defines Rep Variable N (RVar N, N = 1...50) to be EXPR. See also
  RVar command.

- **SETSV**ariable
  Format: Setsv N, EXPR
  Modes: SFN
  Description: Defines Sfn Variable N (SVar N, N = 1...50) to be EXPR. See also
  SVar command.

- **SETV**arInDPmode
  Format: SetV N, EXPR
  Modes: DPM
  Description: Defines DPrep variable N (Var N, N = 1...50) to be EXPR. See also Var command.

- **SFN**mode
  Format: sfn
  Modes: DPM, REP
  Description: Allows the user to shift from the DPMode or REPmode to the SFNmode.

- **SIGNSEQ**uence
  Format: signseq n, rep (REPmode)
  signseq k,n,char,boolean,sfn (SFNmode)
  Modes: REP,SFN
  Description: REP Used to construct a signed sequence of a Rep that involves up to n parts for the groups O(k), SO(k) or Sp(2k). Signed sequences may be constructed for any rep except for the tensor reps of SO(2k) having k non-zero parts.
  In the special case of the group U(k) the signed sequence is constructed as in page 3127 of
  J.Phys.A:Math. Gen.(18), 3113 (1985). The format for entry in this case is
  signseq Q,P, rep

where rep is a mixed tensor irrep of U(k) and Q and P limit
the parts of the contravariant and covariant tensors respectively.
SFN Used to construct a signed sequence of an S-function
that involves up to n parts which when treated as group
representations in the REP mode of O(k) or Sp(2k) all
modify to a single irrep of the group. Of particular
use in developing algorithms for the non-compact groups.
The character char designates the A, C or G series of
S-functions. Choosing the boolean TRUE results in a
signed sequence of S-functions whose maximal number of
parts is min(k,n) while setting the boolean FALSE restricts
the number of parts to n. It is assumed that sfn is a single
S-function. The S-function may be followed by the character
'#' if associated characters of O(k) are being considered.
In the case of Sp(2k) char ='A', while for O(k) if k is even
char = 'C' while for odd k char = 'G'.

Example:
```
REP>
```
⇒
```
gr o5
Group is O(5)
REP>
```
⇒
```
wt 16 signseq10,21
-[432^2 1^4 ] +[3^2 2^2 1^3 ] -[2^4 1] +[21]
REP>
```
⇒
```
wt 16 signseq10,21#
[5321^6 ] -[4321^5 ] +[3^2 21^4 ] -[2^3 1^2 ] +[21^2 ]
REP>
```

■ **SK_P**fn

Format: Sk_P EXPR1, EXPR2
Modes: SFN
Description: Forms the skew product of two lists of P-functions.
Example:
```
SFN>
```
⇒
```
sk_p 4321,32
             P_{ 41}
             SFN>
```

■ **SK_Q**fn

Format: Sk_Q EXPR1, EXPR2
Modes: SFN
Description: Forms the skew of the Q-functions in EXPR1 with those
in EXPR2.
Example:
```
SFN>
```
⇒
```
sk_q 4321,32
             Q_{ 41}
             SFN>
```

```
⇒     sk_q 4321,m
                  Q_{ 4321}  + Q_{ 432}  + Q_{ 431}  + Q_{ 421}
+ Q_{ 321}
      SFN>
```

■ **SK**_sfn

Format: (1). Sk EXPR1, EXPR2
(2). Sk EXPR, SERIES

Modes: SFN

Description: (1). Calculates the Sfn Skew EXPR1/EXPR2
(2). Calculates Sfn Skew EXPR/SERIES
SERIEs is a letter representing one of the series
A, B, C, D, E, F, G, H, L, M, P, Q, R, S, T, V, W, X, Y
which may be specified in upper or lower case.

Example:

```
SFN>
⇒     sk 321,2
{ 31} + { 2^2} + { 21^2}
SFN>
⇒     sk321,m
{ 321} + { 32} + { 31^2} + { 31} + { 2^2 1} + { 2^2} + {
21^2} + { 21}
SFN>
```

■ **SMON**

Format: SMON INTEGER, EXPR1, EXPR2

Modes: SFN

Description: EXPR1 is a list of S-functions in (INTEGER - 1) variables and EXPR2 is a list of monomials in INTEGER variables. The product of the two expressions is formed and the outputted expression is a set of standardised S-functions in INTEGER variables.
See also GENprod and VMult. Used in the expansion of the square of the Vandermonde determinant in INTEGER variables as a sum of S-functions. The example below corresponds to the expansion in three variables.

Example:

```
SFN>
⇒     gen 3
      { 2^2 } - 2{ 21^2 } + { 202} - 2{ 121} + 4{ 1^2 2}
- 2{ 103} + { 02^2 } - 2{ 013}
      + { 0^2 4}
            SFN>
⇒     smon3,2-3.11,last
      { 42} - 3{ 41^2 } - 3{ 3^2 } + 6{ 321} - 15{ 2^3 }
            SFN>
```

■ **SN**char

Format: SN INTEGER SFN

Modes: SFN
Description: SFN is a partition of N labelling a class of S(N). SNchar computes all non-zero characteristics for all irreps of S(N) indexed by partitions of N into not more than INTEGER parts.
Example: SFN>

```
⇒   sn2,9^38^47^365431
    { 99 } - { 97 2} + { 96 3} + { 95 4} + 3{ 92 7} + 4{ 91
    8} + { 90 9} - { 89 10 } + 3{ 88 11 }
    + 7{ 87 12 } + 3{ 86 13 } + 3{ 85 14 } + 12{ 84 15 } + 14{
    83 16 } + 6{ 82 17 } + 4{ 81 18 }
    + 11{ 80 19 } + 20{ 79 20 } + 16{ 78 21 } + 15{ 77 22 }
    + 26{ 76 23 } + 34{ 75 24 } + 21{ 74 25 }
    + 12{ 73 26 } + 29{ 72 27 } + 39{ 71 28 } + 36{ 70 29 }
    + 32{ 69 30 } + 42{ 68 31 } + 49{ 67 32 }
    + 38{ 66 33 } + 23{ 65 34 } + 37{ 64 35 } + 53{ 63 36 }
    + 46{ 62 37 } + 36{ 61 38 } + 41{ 60 39 }
    + 43{ 59 40 } + 32{ 58 41 } + 25{ 57 42 } + 22{ 56 43 }
    + 35{ 55 44 } + 31{ 54 45 } + 16{ 53 46 }
    + 8{ 52 47 } + 10{ 51 48 } + 5{ 50 49 }
    SFN>
```

■ **SP**in
Description: This command is only used in conjunction with the command CONV_sfnToRep and is discussed under that heading.

■ **SPL**itIntoSpinAndTensor
Format: Spl EXPR
Modes: REP
Description: Spl acts on a REP List and writes the tensor reps to RVar 1 and the spinor reps to RVar 2. This command is very useful in problems involving supersymmetry. Spl Lists may be recombined by use of ADD RV1,RV2.
Example: REP>

```
⇒   gr so6
                Group is SO(6)
                REP>
⇒   split 321+ + s21- + 2 + s1+
                REP>
⇒   rv1
                [321]+ + [2]
                REP>
⇒   rv2
                [s;21]- + [s;1]+
                REP>
```

■ **SPONM**odify
Format: SPONM EXPR

Modes: REP
Description: SPONModify acts on the Sp(2n,R) irreps contained in EXPR and applies the O(k) modification rule to produce a new list of Sp(2n,R) irreps with associate labels '#' where appropriate. NB. The group must be set as Sp(2n,R) - other group settings will produce an error message. See also ASSOCiate.
Example:

```
REP>
⇒   gr spr8
                Group is Sp(8,R)
                REP>
⇒   p1;11,1;11
      <2(10 2)> + <2(10 1^2 )> + <2(84)> + <2(82)> +
<2(81^2 )>
      + <2(6^2 )> + <2(64)> + <2(62)> + <2(61^2 )> +
<2(4^2 )>
      + <2(42)> + <2(41^2 )> + <2(2^2 )> + <2(21^2 )>
+ <2(1^4 )>
                REP>
⇒   sponm last
      <2(10 2)> + <2(10 )># + <2(84)> + <2(82)> + <2(8)>#
      + <2(6^2 )> + <2(64)> + <2(62)> + <2(6)># + <2(4^2
)>
      + <2(42)> + <2(4)># + <2(2^2 )> + <2(2)># + <2(0)>#
             REP>
```

■ **SPRCH**

Format: SPRCH EXPR
Modes: REP
Description: EXPR generates a list of irreps of the group Sp(2n,R). SPRCH acts on every member of the list to replace any irrep involving an n-part partition by an equivalent irrep involving fewer than n-parts while those involvingpartitions involving fewer than n-parts are left unchanged. The group must be set as Sp(2n,R), all other group settings produce an error message.
Example:

```
REP>
⇒   gr spr6
                Group is Sp(6,R)
                REP>
⇒   wt6p2;21,3;1
                <5(51)> + 2<5(42)> + 2<5(41^2 )> + <5(3^2
)> + 3<5(321)>
           + <5(31)> + <5(2^3 )> + <5(2^2 )> + <5(21^2
)>
                REP>
```

```
⇒     sprch last
                  <7(0)> + 2<6(3)> + 3<6(21)> + <6(1)> +
      <5(51)>
                + 2<5(42)> + <5(3^2 )> + <5(31)> + <5(2^2
      )>
                  REP>
```

■ **SPREX**tend

Format: SPREX EXPR1
Modes: REP
Description: Takes a set of Sp(2n,R) irreps and extends the partitions labelling the irreps to yield new standard irreps of Sp(2n,R) such that if the appropriate O(k) modification rules were applied the original irreps would be restored to within an associate label '#' which is suppressed. See also ASSOCiate and SPONModify. NB. The group must be set as Sp(2n,R) - other group settings will produce an error message.
Example:

```
REP>
⇒     gr spr8
                  Group is Sp(8,R)
                 REP>
⇒     2;0 + 2;2 + 2;2^2
                  <2(0)> + <2(2)> + <2(2^2)>
                 REP>
⇒     sprex last
                  <2(2^2)> + <2(21^2)> + <2(1^4)>
                 REP>
```

■ **SPSTAR**

Format: SPSTAR EXPR1 (REP)
SPSTAR gr GRNO EXPR1 (DPM)
Modes: REP, DPM
Description: Takes a set of Sp(2n,R) irreps and replaces each irrep by its "star" equivalent.
NB. The group must be set as Sp(2n,R) - other group settings will produce an error message.
Example:-REP>
->gr spr6
Group is Sp(6,R)
REP>
->p s;0,1;0
<s1(12)> + <s1(10)> + <s1(8)> + <s1(6)> + <s1(4)>
+ <s1(2)> + <s1(0)>
REP>
->spstar last
<s1(12 1)> + <s1(10 1)> + <s1(81)> + <s1(61)> + <s1(41)>

```
+ <s1(21)> + <s1(1^3 )>
REP>
```

■ **SQ**uares

Format: SQ EXPR
Modes: REP, SFN
Description: Sums the absolute squares of the coefficients in EXPR.
Example:

```
SFN>
⇒   o 21,41
                { 62} +{ 61^2 } +{ 53} +2{ 521} +{ 51^3 }
+{ 431} +{ 42^2 } +{ 421^2 }
      SFN>
⇒   sq last
      Sum of absolute squares of multiplicities =11
      SFN>
```

■ **STAR**equivalent

Format: STAR GRNO EXPR
Modes: DPM
Description: Forms the star equivalent of a list of irreps of the product group Sp(2n,R) x O(k). The list usually comes from the output of the group-subgroup decomposition Sp(2nk,R) -> Sp(2n,R) x O(k).
GRNO is the group number of the group Sp(2n,R). See also ASSOCiate and SPSTAR. NB. In this case we designate the infinite set of Sp(2nk,R) irreps of even weight by <s(0)> and those of odd weight by <s(1)>.
Example:

```
DP>
⇒   gr spr40
                 Group is Sp(40,R)
               DP>
⇒   wt2,6br38,8,5gr1[s;0]
               Groups are   Sp(8,R) * O(5)
      <s2(6)>[6] + <s2(51^3 )>[5]# + <s2(51)>[51] + <s2(42)>[42]
+ <s2(41^2 )>[41]#
      + <s2(4)>[4] + <s2(3^2 )>[3^2 ] + <s2(321)>[32]#
+ <s2(31^3 )>[3]# + <s2(31)>[31]
      + <s2(2^2 )>[2^2 ] + <s2(21^2 )>[21]# + <s2(2)>[2]
+ <s2(1^4 )>[1]#
      + <s2(1^2 )>[1^2 ] + <s2(0)>[0]
               DP>
⇒   star gr1 last
      <s2(61^3 )>[6]# + <s2(51^2 )>[51]# + <s2(5)>[5]
+ <s2(421)>[42]# + <s2(41^3 )>[4]#
      + <s2(41)>[41] + <s2(3^2 1)>[3^2 ]# + <s2(32)>[32]
+ <s2(31^2 )>[31]# + <s2(3)>[3]
```

```
      + <s2(2^2 1)>[2^2 ]# + <s2(21^3 )>[2]# + <s2(21)>[21]
+ <s2(1^3 )>[1^2 ]#
      + <s2(1)>[1]
              DP>
```

Notes:

```
 the result has been STDandardized.
```

■ **STAT**usOfSchur

Format: Stat

Modes: DPM, REP, SFN

Description: Determines the status of SCHUR by displaying information about the status configuration of SCHUR.
The format is:
digits:f more:f logging:f/'LOG filename'
10 20 30 40 50
FNS:12–6—-0———————————————-
VAR:123–8————1———————————
where f indicates the status of the flag, i.e. TRUE or FALSE.
If LOGGING is TRUE then the chosen name of the current logfile is displayed. The third line indicates the numbers of the functions that have been set. In the above example FNS 1,2,6 and 10 have been set. The forth line displays the VARS that have been set in the DPMode. In the REPmode the number of RVARS are displayed or in the SFNmode the number of SVARS are displayed. In each case the number of VARS, RVARS, or SVARS is given modulo 10 and hence in the above the VARS 1,2,3,8, and 21 have been set.

■ **STD**

Format: Std EXPR

Modes: DPM, REP, SFN

Description: Converts a list of DPreps, Reps or Sfns into a list of standardised DPreps, Reps or Sfns. The command is available for all compact semisimple Lie groups, for the tensor reps of U(m/n), SU(m/n) and OSp(m/n) and the spin and ordinary reps of S(n). See also STD_OneDprep and STD_Qfn.

Example:

```
DP>
⇒   gr2so5sp4
                Groups are SO(5) * Sp(4)
                DP>
⇒   std[31^3 * 31^3]
                -[3]<31>
                DP>
```

■ **STD_O**neDprep
Format: Std_O gr grN EXPR
Modes: DPM
Description: Standardises the irreps of the group grN leaving all other group irreps unmodified.
Example:
```
DP>
⇒  gr3su2su3so6
                Groups are   SU(2) * SU(3) * SO(6)
                DP>
⇒  [21*321*s21^4+]
                { 21}{ 321}[s;21^4 ]+
                DP>
⇒  std_o gr1 last
                 { 1}{ 321}[s;21^4 ]+
                DP>
⇒  std_o gr2 last
                { 1}{ 21}[s;21^4 ]+
                DP>
⇒  std_o gr3 last
                 - { 1}{ 21}[s;21]-
                DP>
```

■ **STD_Q**fn
Format: Std_Q EXPR
Modes: SFN
Description: Standardises Q-functions arising from EXPR.
Example:
```
SFN>
⇒  o21,21
                { 42} + { 41^2} + { 3^2} + 2{ 321} + { 31^3}
+ { 2^3} + { 2^21^2}
                SFN>
⇒  std_q last
                Q_{ 42} + 2Q_{ 321}
                SFN>
```

■ **S_TO_E**symmFn
Format: S_To_E EXPR
Modes: SFN
Description: Treats EXPR as a list of S-functions and transforms them into a list of elementary symmetric functions e_{ \mu}.
Example:
```
SFN>
⇒  s_to_e 21
                - e_{ 3}  + e_{ 21}
     SFN>
```

■ **S_TO_F**symmFn

Format: S_To_F EXPR
Modes: SFN
Description: Treats EXPR as a list of S-functions and transforms them into a list of forgotten symmetric functions f_{ \mu}.
Example: `SFN>`
⇒ `s_to_f 21`

```
f_{ 21} + 2f_{ 1^3}
SFN>
```

■ **S_TO_H**symmFn
Format: S_To_H EXPR
Modes: SFN
Description: Treats EXPR as a list of S-functions and transforms them into a list of homogeneous symmetric functions h_{ \mu}.
Example: `SFN>`
⇒ `s_to_h 21`

```
-h_{ 3}  + h_{ 21}
SFN>
```

■ **S_TO_M**symmFn
Format: S_To_M EXPR
Modes: SFN
Description: Treats EXPR as a list of S-functions and transforms them into a list of monomial symmetric functions m_{ \mu}.
Example: `SFN>`
⇒ `s_to_m 21`

```
m_{ 21}  + 2m_{ 1^3}
SFN>
```

■ **S**top
Format: stop
Modes: BRM, DPM
Description: In the BRMode Stop causes a current sequence of branchings to be terminated and a request to the user to set a new branching rule. In setting functions Stop is always used to mark the end of a function.

■ **S_TO_P**symmFn
Format: S_To_P EXPR
Modes: SFN
Description: Treats EXPR as a list of S-functions and transforms them into a list of powersum symmetric functions p_{ \mu}. The normalisation
has been chosen to give integer coefficients. The actual coefficients may be obtained by dividing by n! where n is the weight of the partitions in EXPR. It is assumed that all the

partitions in EXPR are of the same weight. Transformations from other symmetric functions to power sum symmetric functions may be found by combining sequences of commands. Thus if EXPR is a set of elementary symmetric functions then the command sequence E_TO_S, S_TO_P, EXPR will yield the desired result.

Example: `SFN>`

⇒ `s_to_p 31`

```
                - 6p_{ 4} - 3p_{ 2^2 } + 6p_{ 21^2 } + 3p_{
1^4 }
                SFN>
```

■ **S_TO_Q**symmFn

Format: S_To_Q EXPR

Modes: SFN

Description: Treats EXPR as a list of S-functions of type S(x,-1) and transforms them into a list of Q-functions.

Example: `SFN>`

⇒ `s_to_q 31`

```
                 Q_{ 4} + Q_{ 31}
                SFN>
```

■ **SUB**tract

Format: Sub EXPR1, EXPR2

Modes: DPM, REP, SFN

Description: Subtracts EXPR2 from EXPR1.

Example: `SFN>`

⇒ `setsv1 321+3+21`

```
     SFN>
```

⇒ `setsv2 4321+32-21`

```
     SFN>
```

⇒ `sv1`

```
                { 321} + { 3} + { 21}
                SFN>
```

⇒ `sv2`

```
                { 4321} + { 32} - { 21}
                SFN>
```

⇒ `sub sv1,sv2`

```
                - { 4321} + { 321} - { 32} + { 3} + 2{ 21}
                SFN>
```

■ **SUM**

Description: This command only arises in using the command RULE and is discussed under that heading.

■ **SUMSQ**uares

Format: SUMSQ EXPR

Modes: SFN
Description: EXPR is a list of S-functions. The action of SUMSQuares is to sum the Littlewood-Richardson squares of every S-function in EXPR.
Example: `SFN>`
⇒ `sumsq { 2} + { 1}`
`{ 4} + { 31} + { 2^2} + { 2} + { 1^2}`
`SFN>`

■ **SUP**pressOutPutToScreen
Format: Sup TRUE:FALSE
Modes: DPM, REP, SFN
Description: This line in a function allows the user to turn the output on or off during the execution of a function. Normally only the last line of a function is outputted. SUP FALSE turns it off. Placed at the beginning of a function it results in the output of the result of each line of the function. Useful for debugging functions. At the completion of a function SUP is automatically returned to TRUE.

■ **SV**ar
Format: Sv N
Modes: SFN
Description: Contains the Sfn variable N, (N = 1...50), set with SETSfnVar. Used as an Sfn when required.

■ **SWAP**groups
Format: SWAP grno a,grno b,Expr
Modes: DPM
Description: Interchanges the order of the groups a and b and appropriately reorders Expr.
Example: `DP>`
⇒ `gr3su3sp4so5`
`Groups are SU(3) * Sp(4) * SO(5)`
`DP>`
⇒ `[1*1*1]`
`{ 1}<1>[1]`
`DP>`
⇒ `p last,last`

```
      { 2}<2>[2] + { 2}<2>[1^2 ] + { 2}<2>[0] + { 2}<1^2
>[2]
      + { 2}<1^2 >[1^2 ] + { 2}<1^2 >[0] + { 2}<0>[2]
+ { 2}<0>[1^2 ]
      + { 2}<0>[0] + { 1^2 }<2>[2] + { 1^2 }<2>[1^2 ]
+ { 1^2 }<2>[0]
```

```
      + { 1^2 }<1^2 >[2] + { 1^2 }<1^2 >[1^2 ] + { 1^2
}<1^2 >[0]
      + { 1^2 }<0>[2] + { 1^2 }<0>[1^2 ] + { 1^2 }<0>[0]
      DP>
⇒     swap1,3last
      Groups are   SO(5) * Sp(4) * SU(3)
      [2]<2>{ 2} + [2]<2>{ 1^2 } + [2]<1^2 >{ 2} + [2]<1^2
>{ 1^2 }
      + [2]<0>{ 2} + [2]<0>{ 1^2 } + [1^2 ]<2>{ 2} +
[1^2 ]<2>{ 1^2 }
      + [1^2 ]<1^2 >{ 2} + [1^2 ]<1^2 >{ 1^2 } + [1^2
]<0>{ 2}
      + [1^2 ]<0>{ 1^2 } + [0]<2>{ 2} + [0]<2>{ 1^2 }
+ [0]<1^2 >{ 2}
      + [0]<1^2 >{ 1^2 } + [0]<0>{ 2} + [0]<0>{ 1^2 }
      DP>
```

■ **TAB**leOfBranchingRules

Description: Table A.2 : Table of branching rules

Rule No.	Group		Subgroup Rule and number(s) in BRM
1 :-	U(n)	->	O(n) 1,n
2 :-	U(n)	->	Sp(n) 2,n
3 :-	U(n)	->	U(n-1) 3,n
4 :-	U(m+n)	->	U(m) x U(n) 4,m,n
5 :-	U(mn)	->	U(m) x U(n) 5,m,n
6 :-	U(2k)	->	U(k) 6,2k
7 :-	U(n)	->	SO(3) 7,n
8 :-	SU(m+n)	->	U(1) x SU(m) x SU(n) 8,m,n
9 :-	Sp(2k)	->	SO(3) 9,2k
10 :-	Sp(2k)	->	U(1) x SU(k) 10,2k
11 :-	Sp(2k)	->	SU(2) x SO(k) 11,2k
12 :-	Sp(2k)	->	U(2k) 12,2k
13 :-	Sp(2k)	->	U(k) 13,2k
14 :-	Sp(2m+2n)	->	Sp(2m) x Sp(2n) 14,2m,2n
15 :-	Sp(2mn)	->	Sp(2m) x O(n) 15,2m,n
16 :-	S(m+n)	->	S(m) x S(n) 16,m,n
17 :-	S(n)	->	A(n) 17,n
18 :-	O(n)	->	S(n) 18,n
19 :-	O(n)	->	S(n+1) 19,n
20 :-	O(n)	->	U(n) 20,n
21 :-	O(2k) or O(2k+1)	->	U(k) 21,2k (or 2k+1)
22:-	O(m+n)	->	O(m) x O(n) 22,m,n
23 :-	O(mn)	->	O(m) x O(n) 23,m,n
24 :-	O(4mn)	->	Sp(2m) x Sp(2n) 24,2m,2n
25 :-	SO(2k+1)	->	SO(3) 25,2k+1
26 :-	SO(2k) or SO(2k+1)	->	U(1) x SU(k) 26,2k (or 2k+1)

27 :- SO(m+n) -> SO(m) x SO(n) 27,m,n
28 :- SO(4) -> SU(2) x SU(2) 28
29 :- SO(4) -> SO(3) 29
30 :- SO(7) -> SO(3) 30
31 :- SO(7) -> G(2) 31
32 :- SU(m/n) -> U(1) x SU(m) x SU(n) 32,m,n
33 :- SU(m+n/p+q) -> U(1) x SU(m/p) x SU(n/q) 33,m,p,n,q
34 :- U(mn+pq/mq+np) -> U(m/p) x U(n/q) 34,m,p,n,q
35 :- OSp(m/n) -> O(m) x Sp(n) 35,m,n (n even)
36 :- Sp(2n,R) -> U(n) 36,2n
37 :- Sp(2n,R) -> Sp(2,R) x O(n) 37,2n
38 :- Sp(2nk,R) -> Sp(2n,R) x O(k) 38,2n,k
39 :- Mp(2nk) -> Sp(2n,R) x O(k) 39,2n,k
40 :- G(2) -> SU(3) 40
41 :- G(2) -> SO(3) 41
42 :- G(2) -> SO(7) 42
43 :- F(4) -> SO(9) 43
44 :- E(6) -> SU(2) x SU(6) 44
45 :- E(6) -> U(1) x SO(10) 45
46 :- E(6) -> G(2) 46
47 :- E(7) -> SU(8) 47
48 :- E(7) -> U(1) x E(6) 48
49 :- E(8) -> SU(9) 49
50 :- E(8) -> SO(16)) 50
51 :- E(8) -> SU(2) x E(7) 51
52 :- E(8) -> SU(3) x E(6) 52
53 :- SU(27) -> E(6) 53
54 :- SU(56) -> E(7) 54
55 :- SU(248) -> E(8) 55
56 :- E(6) -> F(4) 56
57 :- F(4) -> SO(3) x G(2) 57
58 :- E(6) -> SU(3) x G(2) 58
59 :- SO(26) -> F(4) 59
60 :- E(8) -> F(4) x G(2) 60
61 :- SO^*(2n) -> U(n) 61,2n
62 :- U(n) -> S(n) 62,n
63 :- S(8) -> L168 63

■ **UONEA**ddInteger
Format: UOneA INTEGER gr grN EXPR (DPM)
UOneA INTEGER EXPR (REP)
Modes: DPM, REP
Description: Adds the INTEGER on to the first part of each partition in EXPR.

While specifically introduced for the group U(1) it may be used for any group.

Example:
```
DP>
⇒ gr2u1su5
Groups are U(1) * SU(5)
DP>
⇒ [~6*21]
{ -6}{ 21}
DP>
⇒ uoneadd8gr1last
{ 2}{ 21}
DP>
```

■ **UONED**ivInteger

Format: UOneD INTEGER gr grN EXPR (DPM)
UOneD INTEGER EXPR (REP)

Modes: DPM, REP

Description: Divides the INTEGER into the first part of each partition in EXPR. An error message is generated if the division is not exact. While specifically introduced for the group U(1) it may be used for any group.

Example:
```
DP>
⇒ gr2u1su5
                Groups are U(1) * SU(5)
                DP>
⇒ [~6*21]
                { -6}{ 21}
                DP>
⇒ uoned3gr1last
                { -2}{ 21}
                DP>
```

■ **UONET**race

Format: UONET EXPR

Modes: DPM

Description: UONETrace evaluates the trace when one of the set Groups is U(1). This is useful in checking branching rules leading to the appearance of a U(1) group. UONETrace multiplies the dimension of each rep in EXPR by its corresponding U(1) weight and should yield a null result.

Example:
```
DP>
⇒ gr so8
                Group is SO(8)
                DP>
⇒ branch26,8gr1[s1+]
```

```
                    Groups are   U(1) * SU(4)
                    { 6}{ 1} + { 2}{ 21} + { 2}{ 1^3} + { -2}{
        2^2 1} + { -2}{ 1}
             + { -6}{ 1^3}
                    DP>
    ⇒   uonet last
                    Utrace = 0
             DP>
```

■ **V**arForDpreps
Format: V N
Modes: DPM
Description: Contains the DPrep variable N, (N = 1...50), set with SETVarInDPmode. Used as the DPrep EXPR when required.

■ **VM**ult
Format: VM INTEGER, EXPR1, EXPR2
Modes: SFN
Description: EXPR1 and EXPR2 are two lists of monomials in INTEGER variables. Their product is output as a list of monomials.
The two monomials (x_1 - x_3) and (x_2 - x_3) are represented as { 1} - { 001} and { 01} - { 001} and their product
x_1x_2 -x_2x_3 - x_1x_3 + x_3^2
is represented as
{ 11} - { 011} - { 101} + { 002}
See also GENprod and SMON
Example: `SFN>`

```
    ⇒   vm3,1-001,01-001
                    { 1^2 } - { 101} - { 01^2 } + { 0^2 2}
                    SFN>
```

■ **WHATG**roup
Format: WhatG
Modes: DPM, REP
Description: This command returns the current setting of the groups.

■ **WITH**
Description: This command is only used in conjunction with the command RULE
and is discussed under that heading.

■ **WRFNTOD**isk
Format: Wrfntod INTEGER 'filename'
Modes: DPM
Description: Writes the function INTEGER (=N), (N = 1...50) to a disk file

'filename'. This file can be read from disk with ReadF to create a function.

■ **WR**fnToScreen
Format: Wr N
Modes: DPM
Description: Displays the contents of function N, (N = 1...50) on the screen.

■ **WSEQ**uence
Format: WSEQ Sequence
Modes: SFN
Description: WSEQ acts on a sequence of integers to produce the sum of the parts, or weight, of the Sequence.
Example: `SFN>`
⇒ `wseq 1112234576`

```
                weight =    32
                       { 1^3 2^2 34576}
                 SFN>
```

■ **WT**ofRepOrSfnSelect
Format: Wt grno, INTEGER, EXPR (DPM)
Wt INTEGER, EXPR (REP, SFN)
Modes: DPM, REP, SFN
Description: In the DPMode if INTEGER is positive then this command returns all terms in EXPR which are of weight INTEGER or less for the grno.
If INTEGER is negative then the command returns all terms in EXPR which are of weight INTEGER for the grno exactly.
In the REP or SFN modes if INTEGER is positive then this command returns all terms in EXPR which are of weight INTEGER or less.
If INTEGER is negative then the command returns all terms in EXPR which are of weight INTEGER exactly.
Example: `SFN>`
⇒ `wt6 outer21^2,21+2+1`

```
                { 41^2} + { 321} + { 31^3} + { 31^2} + { 2^2
1^2} + { 2^2 1} + { 21^3}
     SFN>
```

■ **YH**ooklengths
Format: YH EXPR
Modes: SFN
Description: Writes to screen the hooklengths of every cell of every

Young diagram of every partition in EXPR.

Example: `SFN>`

⇒ `yh o21,21`

```
                                2.
+-+-+-+-+ +-+-+-+-+ +-+-+-+ +-+-+-+ +-+-+-+ +-+-+ +-+-+
:5:4:2:1: :6:3:2:1: :4:3:2: :5:3:1: :6:2:1: :4:3: :5:2:
+-+-+-+-+ +-+-+-+-+ +-+-+-+ +-+-+-+ +-+-+-+ +-+-+ +-+-+
:2:1:     :2:       :3:2:1: :3:1:   :3:     :3:2: :4:1:
+-+-+     +-+       +-+-+-+ +-+-+   +-+     +-+-+ +-+-+
          :1:               :1:     :2:     :2:1: :2:
          +-+               +-+     +-+     +-+-+ +-+
                                    :1:           :1:
                                    +-+           +-+
```

`SFN>`

■ **YO**ungDiagrams

Format: YO EXPR

Modes: SFN

Description: Draws on the screen the frames corresponding to the Sfns in EXPR. Multiplicities appear above each frame except for multiplicity 1. The empty frame corresponding to `{ 0}` appears as a dot. Each cell of a frame is represented by a circle.

Example: `SFN>`

⇒ `42+41^2+3^2+2.321+31^3+2^3+2^2 1^2`

```
             { 42} + { 41^2} + { 3^2} + 2{ 321} + { 31^3}
+ { 2^3} + { 2^2 1^2}
```

`SFN>`

⇒ `yo last`

```
                              2
            OOOO  OOOO  OOO  OOO  OOO  OO  OO
            OO    O     OOO  OO   O    OO  OO
                  O          O    O    OO  O
                                  O        O
```

`SFN>`

■ **YS**hapeSelect

Format: YS letter EXPR

Modes: SFN

Description: letter belongs to the character set ['c','d','r','s','t'].
Choosing 'c' gives the column partitions in EXPR.
Choosing 'd' gives the double hook partitions in EXPR.
Choosing 'r' gives the single row partitions in EXPR.
Choosing 's' gives the single hook partitions in EXPR.
Choosing 't' gives the two row partitions in EXPR.
Any other choice of letters gives an error message.

Example: `SFN>`

```
⇒     ys s 42+41^2+3^2+2.321+31^3+2^3+2^2 1^2
                { 41^2} + { 31^3}
                SFN>
```

■ **Z**ero

Description: Returns a null expression. Can be used in initialisation of stored variables

B

The SCHUR Help Files

B.1 Help Files

Most of the information given in Appendix A can be obtained online by inputting the command ?filename where the minimal filename is the set of boldface uppercase letters associated with the command name as indicated in the table below. The user may input the filename in upper, lower, or mixed case using letters up to the full command name. Thus the commands ?abs, ?AbS, ?AbSoLuTeV will all result in a display on the screen of the helpfile associated with the command ABSoluteValue. The helpfiles are pure ASCII files and hence the user may add helpfiles of his/her own provided these are placed in the \HELP directory and their filenames do not conflict with existing filenames.

Table B.1 Helpfiles in SCHUR.

ABSoluteValue	ADD	ALLskewSfn
ASSOCiate	ATtachPartitionToSfn	AUtoOrIsoMorphism
BRanch	BRMode	
CANcelDatFile	CASIMIRGnthTrace	CASimirNthordertrace
CH_CoeffsToOneForSfns	CH_LabelForOn	CH_PhaseOfSfns
CH_SpinIndex	CH_UoneReps	CLASS
COLumns	COMPare	COMPLement
CONJADD	CONJugateSfnList	CONSPLIT
CONTENT	CONTractGroups	CONTRAGredientRep
CONV_D_TO_Rep	CONV_D_TO_Sfn	CONV_R_TO_Sfn
CONV_S_TO_Rep	COUNTCoeffsInList	COUNTTermsInList
COVariant		
DEAD	DIMension	DPMode
D_TO_Plabel	DYNKINIndex	
END	ENTerVar	EQualSfnList
E_TO_FSymmFn	E_TO_HSymmFn	E_TO_MSymmFn
E_TO_SSymmFn	EXITmode	EXPandSfnList
FFPROD	FIRSTPart	FN
FPROD	FRACAHnotation	FROB
F_TO_ESymmFn	F_TO_HSymmFn	F_TO_MSymmFn
F_TO_SSymmFn	FUSion	
GENERIC	GENprod	GRoup
GWT		
HALLpolynomialProduct	HCLASS	HEAPstatus
HECKE	HELP	HSTD

HSTDList H_TO_ESymmFn H_TO_FSymmFn
H_TO_MSymmFn H_TO_SSymmFn

INDEXsequence INSertPartitionIntoSfn INTegerDivideCoeffs
INVerseseries I_PLethysmRd I_QfnProduct
I_sfnProduct I_SFNQfnProduct

KINSert KMatrix Kostka

LABel LASTresult LATticetest
LENgthOfPartitionsSelect LICENSE LINES
LOadFile LOGfile LSEQuence

MACMixedSeries MACseries MAKEwtOfSfnToN
MAXCoeffInList MIXedTensorReps M_TImesSfnProduct
M_TO_ESymmFn M_TO_FSymmFn M_TO_HSymmFn
M_TO_SSymmFn MUlt_CoeffsByAnInt MULT_List
MULT_Ntimes MULT_PartsByAnInt MULT_SelectInList
MULT_SPlitIntoTwoLists MYlistOfSfns

NLambda NSKew NSTDise

ONSCalar O_PfnProduct O_QfnProduct
O_Restrict O_sfnProduct

PARITYsequence PAUSE PLethysm
PLG ProductKronecker PROPertyOfRepList
P_TO_Dlabel P_TO_SSymmFn

QEXPandSpecialSeries QQEXpandSpecialSeries QQSEries
QSAME QSERies Q_TO_SsymmFn

RACAHnotation RAISEInverseOp RAISEop
RD_I_QfnProduct RD_I_sfnProduct RD_RAISEInverseOp
RD_RaiseOp READFnFromDisk REMark
REPmode RETurn RIEMANNList
RIEMANNPlethList RIEMANNScalarsOrderN RM_EVENPARTS
RM_EVENRkSfnsOnly RM_EVENWtInList RM_FirstPartOfSfn
RM_Group RM_NMP RM_ODDPARTS
RM_ODDRkSfnsOnly RM_ODDWtInList RM_PartitionFromSfn
RM_PARTSequalN RM_RepeatedPartsSfns RM_SOnEvenLabels
RM_UoneWtOverMax RP_FirstPartBySpin RP_RepOrSfnByWt
RP_SfnCoeffByInt RSAMEwtSfnList RULE
RVar

SAMEwtSfns	SAVEsetVar	SB_Bell
SB_CONjecture	SB_CUT	SB_Digits
SB_DIMension	SB_LISToutput	SB_More
SB_POWerNotation	SB_PROGress	SB_Qfn
SB_RD_notation	SB_REVerseOrder	SB_TexOutPut
SCALARInner	SCHAR	SERIESTErmsThatSkew
SERiesToIntWt	SETFnVar	SETLIMit
SET_PWT	SETRVar	SETSVar
SETVarInDPmode	SFNmode	SIGNSEQuence
SK_Pfn	SK_Qfn	SK_sfn
SMON	SNchar	SPIN
SPLitIntoSpinAndTensor	SPONModify	SPRCH
SPREXtend	SPSTAR	SQuares
STARequivalent	STATusOfSchur	STD
STD_OneDprep	STD_Qfn	S_TO_ESymmFn
S_TO_FSymmFn	S_TO_HSymmFn	S_TO_MSymmFn
STOP	S_TO_PsymmFn	S_TO_QsymmFn
SUBtract	SUM	SUMSQuares
SUPpressOutputToScreen	SVar	SWAPgroups

TABleOfBranchingRules

UONEAddInteger	UONEDivInteger	UONETrace

VarForDpreps	VMult	

WHATGroup	WITH	WRFNTODisk
WRfnToScreen	WSEQuence	WTofRepOrSfnSelect

YHooklengths	YOungDiagrams	YShapeSelect

Zero

■ B.2 Function Files

Included with the package SCHURis a complete set of files for every function described in this manual. These files have the extension .fn and are prefixed by p# where # is the page number in the manual where the given function is described. The user may call upon any of these function files by issuing the command "readfn1'filename'" (e.g. readfn1'p83.fn') and then issuing the command "fn1" to run the function.

He pūreirei whakamatatanga

(Ancient Māori proverb)

C

Practical Details

Introduction

In this appendix we outline some practical details in running SCHUR.

Setting up directories

Before installing SCHUR three directory structures should be created on your system. These should be named SCHUR, SCHUR\HELP and SCHUR\DAT. The executable file SCHUR.EXE should be placed in the directory SCHUR while the files *.DAT are placed in the directory SCHUR\DAT. The HELP files are then placed in the directory SCHUR\HELP. SCHUR normally expects to find any userdefined functions of your creation in the directory SCHUR. If you choose to place them in a separate directory such as SCHUR\UFN then when you use the commands relating to functions you will need to give the path e.g. **readfn1'\UFN\myfn.fn'**. A similar situation holds for loading and saving with the commands **save** and **load**.

Limitations and set dimensions

SCHUR has preset values for a number of quantities. In particular

Maximum number of groups that may be set = 8.

Maximum number of functions that may be set = 10.

Maximum number of svar that may be set = 20.

Maximum number of rvar that may be set = 20.

Maximum number of var that may be set = 20.

Maximum size of a part of a partition =127.

Maximum size of a coefficient in an expression = MaxInt.

Maximum length for the dimension of a representation = 400 digits.

The above settings may be modified upon request in very special cases but most users are likely to find the preset values as optimum values.

Error messages and runtime errors

Considerable effort has been devoted to the avoidance of fatal runtime errors. Many errors, especially those associated with input errors, are trapped and reported to the screen. The error messages are usually self-explanatory. Results that appear beneath and error message should be ignored. Errors associated with exhaustion of memory are not trapped and as such will result in a fatal runtime error.

No man is wise enough to think of all the ideas that can occur to a fool
—Rudolph Peierls *Bird of Passage, Princeton 1985*

D

PRODUCING TEX TABLES

Introduction

In many cases the output from Schur is quite voluminous and without appropriate procedures the production of tabular information can be a very tedious process requiring much editing of saved logfiles. However SCHUR can produce tabular output that can almost immediately compiled for printing as a TEX file. This is accomplished by issuing the following commands:-

- **Sb_T**exOutput

  Format: **Sb_T** TRUE:FALSE
  Modes: DPM, REP, SFN
  Description: Setting **Sb_T** TRUE forces output from all modes to be in TEX format. The default is **Sb_T** FALSE which gives normal output.

- **Col**umns

  Format: **Col**umns INTEGER
  Modes: DPM, REP, SFN
  Description: **Col**umns allows the user to determine the number of columns in which to tabulate information. The default value is 5. The choice of INTEGER depends on the multiplicities and width of partitions involved.

- **Lines**

  Format: **Lines** INTEGER
  Modes: DPM, REP, SFN
  Description: **Lines** allows the user to determine the number of lines to be placed on a given page. The default value is 50. Every INTEGER lines printed leaves two blank lines.

- **Making a TEX table**

To produce a TEX file the user first sets SCHUR for producing TEX output by issuing the command **Sb_T**ex TRUE and then, if desired, uses the commands **Col**umns and **Lines** to set the number of columns and lines per page. All commands will now automatically produce TEX output. This output may be collected into a **Log**file as discussed on page 153. A text editor can then be used to remove any extraneous information and to add any desired textual information. Any page of the TEX output can now be converted into a TEX box for printing by attaching, for example, the line

```
\setbox1 = \hbox{\vbox{\settabs3\columns
```

to the start of the TEX page and at the bottom of the page the lines

```
}}
$$\box1$$
\vfill\eject
```

where it has been assumed the user has set the number of columns at 3.

As an example consider the following logfile created in SCHUR

```
DP>
->sb_tex true
DP>
->columns3
DP>
->gr u4
Group is U(4)
DP>
->p[321],[321]
\+$\{642\}$&$ + \ \{641^2\}$&$ + \ \{63^2\}$\cr
\+$ + \ 2\{6321\}$&$ + \ \{62^3\}$&$ + \ \{5^22\}$\cr
\+$ + \ \{5^21^2\}$&$ + \ 2\{543\}$&$ + \ 4\{5421\}$\cr
\+$ + \ 3\{53^21\}$&$ + \ 3\{532^2\}$&$ + \ \{4^3\}$\cr
\+$ + \ 3\{4^231\}$&$ + \ 2\{4^22^2\}$&$ + \ 3\{43^22\}$\cr
\+$ + \ \{3^4\}$\cr
DP>
->end
```

Using a text editor we can produce a TEXbox as

```
\setbox1 = \hbox{\vbox{\settabs3\columns
\+$\{642\}$&$ + \ \{641^2\}$&$ + \ \{63^2\}$\cr
\+$ + \ 2\{6321\}$&$ + \ \{62^3\}$&$ + \ \{5^22\}$\cr
\+$ + \ \{5^21^2\}$&$ + \ 2\{543\}$&$ + \ 4\{5421\}$\cr
\+$ + \ 3\{53^21\}$&$ + \ 3\{532^2\}$&$ + \ \{4^3\}$\cr
\+$ + \ 3\{4^231\}$&$ + \ 2\{4^22^2\}$&$ + \ 3\{43^22\}$\cr
\+$ + \ \{3^4\}$\cr
}}
$$\box1$$
```

Which when TEXcompiled yields

$$
\begin{array}{lll}
\{642\} & + \{641^2\} & + \{63^2\} \\
+ 2\{6321\} & + \{62^3\} & + \{5^22\} \\
+ \{5^21^2\} & + 2\{543\} & + 4\{5421\} \\
+ 3\{53^21\} & + 3\{532^2\} & + \{4^3\} \\
+ 3\{4^231\} & + 2\{4^22^2\} & + 3\{43^22\} \\
+ \{3^4\} & &
\end{array}
$$

I prefer the open landscape under a clear sky with its depth of perspective, where the wealth of sharply defined nearby details gradually fades away towards the horizon.
— H. Weyl, Classical Groups 1938

INDEX to the SCHUR user's Manual