Université du Québec à Montréal

A HOLONOMIC SYSTEMS APPROACH TO ALGEBRAIC COMBINATORICS
UNE APPROCHE HOLONOME À LA COMBINATOIRE ALGÉBRIQUE

THÈSE

PRESÉNTÉE

COMME EXIGENCE PARTIELLE

DU DOCTORAT EN MATHÉMATIQUES

PAR

MARNI JULIE MISHNA
BMath, University of Waterloo, 1998.
MSc., Simon Fraser University, 2000.

NOVEMBRE 2003

This thesis is dedicated to the memory of a wonderful woman, my mother, Vicki Munn.

Acknowledgements

I offer the following people and organizations heartfelt appreciation for their contributions to this work and their support during the period of my thesis.

The two research teams which I had the pleasure of being part: LaCIM (Université du Québec à Montréal) and Projet Algorithmes (Inria, Rocquencourt, France). Both are thriving incubators of combinatorics with warm, human aspects, and regular coffee. I thank also the students and postdocs of both groups, such as Marianne Durand, Cédric Lamathe, and Ludovic Meunier, for their humour and camaraderie;

A special separate mention must be made for *les assistantes extraordinaires* Lise Tourigny (LaCIM) and Virginie Collette (Inria) and Manon Gauthier (UQAM). You keep these mathematical machines well oiled and running flawlessly;

The National Science and Engineering Research Council, and the *Institut des Sciences Mathématiques* for generous financial support;

Peter Borwein (Simon Fraser University) and Gilbert Labelle (LaCIM) for accepting positions on the jury and providing important commentary on the text and presentation;

Math princesses on all continents, especially Karen Meagher (University of Ottawa) and Sylvie Corteel (Versailles); You are my *agences immobilières*, my sisters in craft and most of all, inspiring. Additionally, Ms. Meagher carefully read many sections and provided indications on how to make the text accessible to readers other than myself;

Chef extraordinaire Philippe Flajolet (Inria), for teaching me many essential lessons I could not have learned elsewhere, like: analysis is enjoyable when there is a combinatorial pretext, and that *chameaux blatèrent*;

François Bergeron (LaCIM), my director, for consistently offering of excellent advice, direction and ideas. His careful reading of the text, and the numerous suggestions

that followed, are also greatly appreciated;

Drill sergeants Frédéric Chyzak and Bruno Salvy (Inria), for intensive training regimens, countless fascinating discussions and opportunities, and boundless patience towards a non-*polytechnicien* trying to survive in their midsts. Additional thanks to M. Salvy for his active role on the jury;

Cedric Chauve for unfaltering support regarding all aspects of my work, and indeed life: technical, linguistic, emotional, and clean kitchen floor-ical;

Finally, thanks to all of the other quirky combinatorial characters that populate this community. Thank you for your good ideas, curious questions, engaging talks and providing the foundation for an intricate and rewarding domain of study.

❤Marni Mishna

Montreal, Canada, November 2003.

Table of Contents

List of Figures

List of Tables

Index of Notation

Notation	Description	Defined on page
$\mathcal{A}_n$, $\mathcal{A}_x$	the Weyl algebra	49, 56
$[a, b]$	bracket operator $ab - ba$	49
$a \cdot x$	left module action of a on x	
$x.a$	right module action of a on x	55
$a \cdot_\sigma x$	(σ-)twisted module action of a on x	55
$\mathcal{B}_k$	the Bernstein filtration	51
D_q	q-Derivative	68
$\delta_{x,y}$	Kronecker notation $\delta_{x,y} = 1$ if $x = y; 0$ otherwise	
e_λ	elementary symmetric function	26
$\mathcal{E}(t)$	$\sum_\lambda e_n t^n$	27
ex	exponential specialization	35
ex_q	exponential specialization, q-analog	36
$\mathcal{F}$	the Fourier transform of a module	55
$\langle f, g\rangle$, $\langle f, g\rangle_x$	the scalar product of symmetric functions	29
Γ_{F}	the asymmetry cycle index of F	122
$gr^{\mathcal{F}}\mathcal{R}$	graded algebra of $\mathcal{R}$ associated with the filtration $\mathcal{F}$	51
h_λ	complete homogeneous symmetric function	26
$\mathcal{H}(t)$	$\sum_\lambda h_n t^n$	27
$H_M(n)$	the Hilbert polynomial	53
H_q	q-shift operator	68

Notation	Description	Defined on page
$\mathbb{K}$	a field of characteristic 0	19
$\mathbb{K}[\![p]\!]$	ring of symmetric power series	27
$l(\lambda)$	length of a partition	25
$\lambda \vdash n$	a partition of n	25
λ'	conjugate partition	25
$\lambda \cup \nu$	union of partitions	30
Λ	ring of symmetric functions	27
Λ_n	ring of homogeneous symmetric functions	27
$\mathbf{m}_i(\lambda)$	number of parts equal to i in the partition λ	25
M_σ	the twisting of module M by σ	55
$\binom{n}{k}_q$	q-binomial	36
$[n]$	the set $\{1, 2, \ldots, n\}$	123
$n!_q$	modified q-factorial	36
$\mathbb{K}[\partial; \sigma, \delta]$	Ore algebra generated by ∂	64
$\mathbb{P}$	the set of all partitions	25
p_λ	power symmetric function	26
ps	(stable) principal specialization	37
ps_n	principal specialization	37
ps_n^1	specialized principal specialization	37
$p(x) \xrightarrow{q(x)} r(x)$	reduction modulo a polynomial	57
$(q)_n$	q-factorial	36
$(q\|a)_n$	q-Pochhammer	36
r_n, R_n	reduction specialization	35
$\mathbb{S}_n$	the symmetric group of order n	25
S_x	shift operator	64
ω	symmetric function involution	33
z_λ	norm of p_λ	28
Z_{F}	cycle index series	117

Résumé - Abstract

La théorie des systèmes holonomes s'avère être un outil important pour prouver automatiquement des identités combinatoires. Le produit scalaire de fonctions symétriques fournit un cadre utile à la formulation de nombreux problèmes en combinatoire. Ce travail utilise conjointement ces deux approches pour décrire des algorithmes de calcul, sous certaines conditions, du produit scalaire de fonctions symétriques, basés sur les systèmes holonomes.

Ces algorithmes sont valables dans des conditions plus générales que certains travaux précédents. Nous prouvons la correction et la terminaison de ces algorithmes. Des modifications mineures de ces algorithmes permettent de calculer certaines généralisations du produit scalaire, par exemple pour les fonctions symétriques de MacMahon et un q-analogue apparaissant dans l'étude des polynômes de Macdonald. De plus un algorithme général, paramétré par l'adjonction associée au produit scalaire, est décrit.

Ces algorithmes utilisent les bases de Gröbner dans les algèbres de Weyl et exploitent des conditions similaires à celles impliquées dans les algorithmes effectifs d'intégration pour les fonctions D-finies.

Ce travail est divisé en trois parties : la première fournit les bases nécessaires sur les fonctions symétriques et l'holonomie; la seconde définit et prouve plusieurs algorithmes de calcul du produit scalaire symétrique et une généralisation; la partie finale fournit des exemples combinatoires.

Mots Clés:
Fonctions symétriques; systèmes holonomes; énumeration combinatoire; combinatoire automatique

---

The theory of holonomic systems has proven a valuable tool for automatic proofs of combinatorial identities. The scalar product of symmetric functions provides a useful way to phrase many problems in algebraic combinatorics. This work brings together these two ideas to describe algorithms for computing the scalar product of two symmetric series under certain conditions, using some techniques from holonomic systems.

The algorithms here operate under more general conditions than previous work. The correctness and termination of the algorithms is proven. Small modifications of the algorithms yield techniques for calculating generalizations of scalar product, for example from MacMahon symmetric functions and a q-analog arising in the study of Macdonald polynomials, additionally, a general algorithm, parameterized by the adjoint of the scalar product, is given.

The algorithms use Gröbner bases in Weyl algebras, and exploit conditions similar to those involved in effective integration algorithms for D-finite functions.

The work is divided into three parts: the first provides the required background on symmetric functions and holonomy; The second defines and proves several algorithms for computing the symmetric scalar product as well as a generalization; the final part provides some typical combinatorial examples.

KEYWORDS:
Symmetric functions; holonomic systems; enumeration; automatic combinatorics

Une Approche Holonome à la Combinatoire Algébrique

Percy A. MacMahon, il y a plus d'un siècle de cela, fut un des premiers à utiliser un opérateur différentiel agissant sur les fonctions symétriques dans le cadre de problèmes d'énumération, mais sans pouvoir exploiter toute la puissance de cet outil. Nos travaux prennent leur source dans cette première tentative incomplète et s'appuient sur les immenses progrès qu'ont connus depuis l'utilisation combinatoire des séries formelles et fonctions symétriques, ainsi que le calcul formel. Plus précisément, les travaux présentés dans cette thèse se situent à la confluence du calcul formel et de la combinatoire algébrique, via l'étude d'algorithmes permettant de calculer, dans un cadre adapté à une large classe de problèmes combinatoire, le *produit scalaire de séries symétriques*. Cette approche avait été en partie esquissée dans des travaux de Goulden, Jackson et Reilly [37] et de Gessel [34], mais dans des cas particuliers et limités. Nos résultats s'appliquent dans un cadre plus général que les travaux sus-cités et permettent de traiter une large classe de problèmes. Par exemple, la notion de q-paramètre s'intègre naturellement dans le cadre que nous avons développé. Nous fournissons dans cette thèse un ensemble d'algorithmes originaux, analysés rigoureusement (en termes de correction et de terminaison), ainsi que plusieurs exemples non triviaux illustrant leur utilité dans la résolution de problèmes d'énumération. Le code des principaux algorithmes, ainsi que plusieurs exemples d'utilisations, regroupés dans des feuilles de calcul Maple, sont disponibles sur la page Web de l'auteur, `http://www.lacim.uqam.ca/~mishna`.

Le socle sur lequel repose ce travail est le codage et l'étude de familles d'objets combinatoires à l'aide de séries formelles, comme par exemple la fonction génératrice exponentielle ou la série indicatrice de cycles de Pólya. Ce paradigme, l'un des plus importants en combinatoire actuellement, permet d'utiliser de nombreux outils issus de l'algèbre, de l'analyse ou du calcul formel pour améliorer notre connaissance des objets combinatoires ainsi encodés.

Un autre intérêt majeur de la représentation de classes d'objets combinatoires par des séries formelles réside dans la possibilité de disposer d'une représentation finie pour une classe infinie d'objets, que ce soit par l'intermédiaire d'une forme close explicite

pour la série ou d'un système d'équations définissant celle-ci. Cette propriété ouvre la porte à l'utilisation de programmes de calcul formel pour la manipulation effective de ces représentations finies. Ce sont des considérations de cet ordre qui ont permis que l'ordinateur devienne un outil efficace pour la recherche mathématique, et donc pour la combinatoire.

De nombreuses familles d'objets combinatoires peuvent être décrites implicitement par des équations fonctionnelles algébriques. Plusieurs systèmes théoriques, exploitant justement cette propriété, ont été mis au point au cours des dernières décennies. Ces systèmes sont en général accompagnés de bibliothèques de calcul formel de manipulation d'équations fonctionnelles. On peut penser aux structures décomposables [27, 28], aux systèmes ECO de l'école florentine [3], aux grammaires d'objets [23, 26] ou à la théorie des espèces [7, 44].

Il est cependant clair que l'on ne peut se limiter à la manipulation d'équations fonctionnelles : de nombreux problèmes combinatoires font en effet intervenir des séries formelles non algébriques. Dans cette thèse nous nous intéressons à une classe de fonctions plus générale avec d'intéressantes propriétés de fermeture qui sont compatibles avec une manipulation algorithmique effective.

Séries formelles D-finies.

Une série formelle en une variable est dite *différentiellement finie*, ou simplement D-finie, si elle est solution d'une équation différentielle linéaire à coefficients polynomiaux. Cette équation différentielle s'avère être une structure de données efficace pour exprimer nombre d'informations sur la série solution et il existe de nombreux algorithmes permettant de manipuler de telles équations différentielles. En particulier, la famille des séries formelles D-finie à une variable est fermée pour la somme, le produit, le produit de Hadamard et la transformée de Borel. De plus, ces opérations peuvent se réaliser avec des algorithmes effectifs [75, §6.4]. On dira que ce sont des propriétés de fermeture *effectives*.

D'autre part, la suite des coefficients d'une série D-finie satisfait une récurrence linéaire. Ceci permet donc de calculer efficacement de nombreux termes de cette suite. On peut aussi noter que la connaissance du fait qu'une série formelle soit D-finite permet de déduire certaines propriétés sur la nature du comportement asymptotique de ses

coefficients. Ce comportement peut être automatiquement calculé à partir de l'équation différentielle vérifiée par la série. La connaissance d'une équation différentielle satisfaite par une série D-finie est donc de première importance, autant théoriquement qu'en pratique.

Naturellement, on peut étendre la plupart des remarques et propriétés précédentes au cas des séries formelles à plusieurs variables. Là encore il existe un corpus algorithmique et logiciel permettant de manipuler ces séries. C'est le cas, par exemple, de la bibliothèque Maple `Mgfun` de Chyzak.

Le contexte naturel pour ces généralisations est celui des anneaux d'opérateurs linéaires. En particulier, les algèbres de Ore s'imposent comme cadre pour une telle approche. On peut, dans ce contexte, étendre la notion de D-finitude à des classes de fonction caractérisées en termes d'opérateurs de Ore. Ces fonctions sont appelées ∂-finies, et encore une fois il existe des bibliothèques Maple permettant de les manipuler. Voir `Ore_Algebra` et `Holonomy`, de Chyzak [17].

Produit scalaire de fonctions symétriques.

Dans un autre ordre d'idée, Gessel [34] a posé les fondations d'une théorie de la D-finitude pour les fonctions symétriques. Encore une fois, on a ici un ensemble intéressant de propriétés de fermeture. Son travail a été en partie motivé par l'observation que le produit scalaire établit un lien direct entre séries formelles symétriques et fonctions génératrices d'objets combinatoires.

Ainsi Gessel présente un ensemble de conditions qui permettent de déterminer que certaines séries formelles sont D-finies via un calcul de produit scalaire faisant intervenir des séries de fonctions symétriques. Cependant, ses calculs ne se font pas de façon effectives. Le désir de rendre effectifs, et donc automatisables, les calculs en question est l'une des motivations de cette thèse.

Nous décrivons dans ce travail une collection d'algorithmes effectifs permettant de calculer un système d'équations différentielles vérifié par le produit scalaire de fonctions symétriques. Un de nos algorithmes est apparenté à une technique utilisée dans un cas particulier par Goulden, Jackson et Reilly pour calculer un produit scalaire. Nous formalisons cette méthode à l'aide de bases de Gröbner et de systèmes holonomes, ce qui

nous permet d'en déduire un algorithme, HAMMOND, qui termine et dont nous prouvons la correction.

De plus, nous résolvons le problème du calcul du produit scalaire sous certaines conditions de finitude. Nous décrivons deux algorithmes pour ce problème, qui sont tous deux facilement généralisables à d'autres produits scalaires.

À titre d'examples nous montrons comment généraliser les problèmes d'énumération de graphes k-réguliers, dans le contexte plus large de la théorie des espèces. Ces généralisations mettent évidence des conditions d'uniformité sur certaines familles de structures, et mènent à la preuve de la D-finitude des séries génératrices correspondantes. Il apparaît ainsi que la théorie des espèces fournit un outil de caractérisation des structures combinatoires admettant une fonction génératrice D-finie.

Nous montrons aussi comment nos algorithmes peuvent être facilement modifiés pour manipuler d'autres fonctions symétriques comme celles de MacMahon, ou des q-variantes liées aux polynômes de MacDonald.

Notre approche repose sur l'utilisation de systèmes holonomes du fait des relations profondes entre D-finitude et holonomie. Les systèmes holonomes s'avèrent d'un grand intérêt pour fournir un cadre unifié au sein duquel nous pouvons décrire et analyser rigoureusement nos algorithmes. Ils sont aussi à la base de notre généralisation aux q-séries.

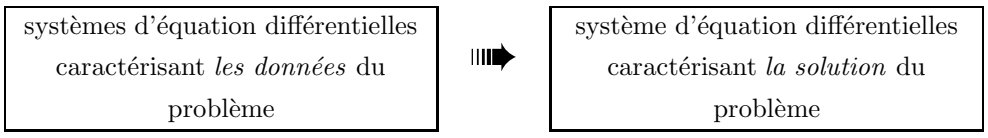
Plan du mémoire.

Dans une première partie, nous établissons le cadre théorique sur lequel nous nous appuyons pour mettre au point et analyser nos algorithmes. Nous commençons en rappelant plusieurs résultats classiques sur les séries différentiellement finies et les fonctions symétriques. Nous poursuivons par une brève présentation du calcul des bases de Gröbner et des systèmes holonomes, qui sont au cœur de nos algorithmes. Nous introduisons en particulier les bases de Gröbner pour certaines algèbres de Weyl comme un outil de calcul effectif dans ces algèbres. Nous concluons cette première partie par un chapitre sur les algèbres de Ore et les q-séries.

La description et l'analyse de nos algorithmes de calcul du produit scalaire de fonctions symétriques se situe dans la seconde partie de ce mémoire, plus particulièrement dans le

chapitre 5. Le chapitre 6 complète cette description en détaillant comment modifier les algorithmes du chapitre 5 pour calculer des produits plus généreaux. Cette possibilité d'adapter notre méthode à de nombreuses généralisations des fonctions symétriques en est un des points forts.

La troisième partie est dédiée à l'illustration de notre méthode. Elle ne présente essentiellement que des résultats de nature combinatoire obtenus à l'aide de nos algorithmes. En fait, cette partie peut être lue indépendamment des deux premières par un lecteur désirant seulement utiliser nos algorithmes. Le principe est ici qu'on établit une correspondance:



Le chapitre 7 contient quelques exemples classiques, comme les graphes réguliers et le recouvrement d'ensembles, en les plaçant dans le contexte de la théorie des espèces pour illustrer leur forte similitude ainsi que le vaste potentiel de généralisation. Le dernier chapitre contient des applications de l'utilisation d'algèbres de Ore, décrites au chapitre 4, pour plusieurs problèmes d'énumération combinatoire.

Introduction

Captain Percy A. MacMahon, in his early treatise on combinatory analysis [53], (but see also [54, 55]) introduced an operator, called the *Hammond operator*, which proved useful in the set up of many enumerative problems, such as counting Latin squares and other integer matrices with restrictions. This operator is essentially a differential operator acting on symmetric functions. He lacked, however, suitable machinery to manipulate the differential equations or even to fully illustrate the symmetry of his operator. His methodology did not catch on.

In the decades that followed, there has been impressive progress made on formal power series, algebraic combinatorics, symmetric functions, and Gröbner bases. Sufficiently many tools now exist to give his method its due.

This thesis describes a method to effectively calculate a quantity called *the scalar product of symmetric series*, under certain conditions, including those relevant to many of the problems considered by MacMahon. Earlier work, notably by Jackson, Goulden and Reilly [36, 37] and Gessel [34] initiated this approach, but limited the scope to calculating particular scalar products, precisely, the scalar product $\langle f, g \rangle$ of a symmetric series f and a particular g, fixed by each respective author. Here we provide a more general solution, and offer several examples to illustrate the variety of problems that fall into this category. Included are questions in graph enumeration, set coverings, orthogonal polynomials, and some variants.

To introduce the topic, we review some of the relevant progress in combinatorics and computer algebra made since MacMahon's time.

Analytic representations of combinatorial structures

One of the most important paradigms in discrete math closely identifies a combinatorial class with an analytic representations or, rather, an encoding. This is a good mathematical strategy; it offers results from analysis, algebra, and symbolic computation as tools towards the determination of combinatorial results.

In particular, much emphasis has been placed on formal series representations of classes: there are straightforward enumerative series such as the exponential generating function and more complicated series such as the cycle index series of Pólya, which encapsulates detailed structural and inventory information.

The relative complexity of combinatorial families

Confronted with such a formal representation of a class, the mathematician's inner analyst is immediately compelled to ask about the nature of the series. Is it rational? algebraic? Where are the poles? Does it satisfy a differential equation? These questions are made even more interesting when one can rephrase them in a combinatorial context. For example, how can one characterize the combinatorial structures whose exponential generating function are algebraic? The answers can, in a way, quantify the complexity of the structure.

Consider a simple analogy from computer science language theory. There is a strict chain of containment for formal languages over a finite alphabet: finite languages are contained in the set of regular languages which are in turn contained in context free languages. The formal sum of the words in a language is a formal series in the non commuting variables of the alphabet. In this case, the sum associated to a finite language is a polynomial; the series associated to a regular language is rational and finally, the series associated to a context-free language is algebraic. Not surprisingly, as the combinatorial object becomes more complex, so does the corresponding analytic class.

Equations and automatic combinatorics

There is another major function served by encoding combinatorial families by equations and series: the possibility of finite representation. It can be cumbersome to manipulate abstract infinite families of combinatorial objects, whereas formal power series can often be described in compact terms, for example, the geometric series $\sum_n X^n = \frac{1}{1-X}$. Computers have become an increasingly important tool in mathematics and in particular, in combinatorics. In this case one exploits finite representations of series and sets. This can take the form of a closed form of a power series, using elementary or special functions, or of other natural possibilities such as finite equations, either algebraic or differential, that the function might satisfy.

Many interesting combinatorial classes can be described with functional equations. Several competing combinatorial description systems have arisen in the past few decades which are based on the ability to translate smoothly between a combinatorial description and a set of defining equations. These are generally accompanied by a corresponding computer algebra package to manipulate the functional equations. For example, systems such as decomposable structures [27, 28], ECO systems [3], object grammars [23, 26], or species theory [7, 44], each provide natural ways to construct families of combinatorial objects with systems of equations. By construction, the objects' generating functions satisfy certain conditions, and are essentially automatically computed.

It is clear that for many problems of current interest, the class of algebraic functions is insufficient. This thesis considers a class of functions larger than algebraic, which also has numerous closure properties which can be calculated and generalized. Differential equations play an important role in this investigation.

D-finite power series

A power series in one variable is called differentiably finite, or simply D-finite, when it is the solution of a linear differential equation with polynomial coefficients. This differential equation is a convenient data structure for expressing information related to the series and many algorithms can operate directly using the defining differential equation. In particular, univariate D-finite power series are closed under sum, product, Hadamard product, and Borel transform, and algorithms computing the corresponding differential equations are known (see for instance [75, §6.4]). These properties are thus considered *effective* closure properties, since they are computable.

Moreover, the coefficient sequence of a univariate D-finite power series satisfies a linear recurrence, which makes it possible to compute many terms of the sequence efficiently and to transition between the differential equation of the series and the linear recurrence of its coefficients. These closure properties can thus be implemented in computer algebra systems [56, 70]. Also, the mere knowledge that a series is D-finite gives information concerning its asymptotic behavior, and much of the asymptotic behaviour can be computed starting from the defining differential equation. Thus, whether it be for algorithmic or theoretical reasons, it is often important to know if a given series is D-finite, and it is useful to compute the corresponding differential equation when

possible.

Some typical examples of combinatorial objects whose generating function are not D-finite include: certain families of walks in the plane [12], classes of planar animals [63], and linear chord diagrams with many crossings [45].

The property of satisfying a differential equation can be generalized to suit many situations. It can be extended to classes of power series in several variables in such a way that the class still enjoys many closure properties. Here again, we find that many key algorithms have been implemented in computer algebra systems, for example in Chyzak's `Mgfun` package [17].

This is even further generalized in the context of linear operators. For example, Ore operators are a family of linear operators which possess certain properties which make them suitable for this approach. In particular, we can generalize the notion of D-finite to include functions which satisfy particular equations of Ore operators. These functions are called ∂-finite and they enjoy many of the same closure properties as D-finite functions [16] and a corresponding implementation in Maple exists, namely Chyzak's `Ore_Algebra` and `Holonomy` packages [17].

The scalar product of symmetric functions

In another direction, Gessel, in [34], has laid a foundation of a theory of D-finiteness for symmetric functions, and again we find a significant collection of closure properties. His motivation for studying symmetric D-finite series is that some operations, notably a scalar product, yield a direct connection between symmetric power series and generating functions of combinatorial objects.

Gessel's work presented conditions under which one could determine whether certain power series were D-finite, but did not describe any algorithms to make these closure properties effective. It is here that the original work of this thesis begins.

This work provides a collection of effective algorithms to compute a system of differential equations satisfied by the scalar product of symmetric functions. One of the algorithms presented here bears some connection to a technique used by Goulden, Jackson and Reilly in [37] to compute a restrictive case of the scalar product. Their method is formalized here with the aid of Gröbner bases and holonomic system. Our formalization

yields an algorithm, HAMMOND which terminates and is provably correct.

In addition, we solve the general problem of computing scalar products of D-finite functions under a finiteness condition. Two algorithms are given to compute this and both can easily be modified to handle other kinds of scalar products, provided the adjoint to multiplication is effective.

The combinatorial examples which Gessel [33, 34], and Goulden, Jackson and Reilly [36, 37] present as fruits of their work can be extended and complemented with the aid of the algorithms given here. Using k-regular graphs as a running example, we show, in the context of species theory, how uniformity conditions on certain families of structures lead to D-finite generating functions. We discover that species theory provides a useful starting point for characterizing combinatorial structures with D-finite generating functions.

Several applications are generalized with the aid of slight modifications of the main algorithms. These modifications allow us to handle a variation on the notion of symmetric functions called MacMahon symmetric functions, and a q-analogue using a modified scalar product of Macdonald.

We have a "holonomic systems approach" because of the deep connections between D-finite functions and holonomy. Holonomic systems are invaluable to provide a unified setting in which we can rigorously describe and prove the termination of the algorithms presented here. Furthermore, it offers a natural setting in which to generalize the work into a q-analogue.

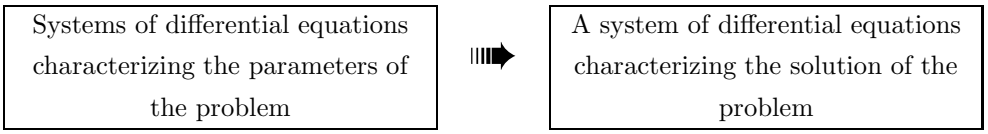
Detailed plan of the work at hand

This work divides itself naturally into three major parts.

The first part is the development of the theory required. In the first chapter we recall many classical facts about D-finiteness and the second chapter treats symmetric functions. This is followed by a brief presentation of holonomic systems and Gröbner bases, which will be the engine of the algorithms of Part 2. This discussion includes a definition of Weyl algebras, the context of the calculations, and Gröbner bases for these Weyl algebras, as a tool for computation in these algebras. The first part concludes with a chapter on Ore algebras and a q-series analogue extension of the discussion.

The second part introduces new algorithms for the effective computation of the scalar product of symmetric series. The fifth chapter concludes with proofs of correctness and termination. The guarantee of termination is a bi-product of the fact that we work with holonomic modules. This part concludes with the sixth chapter, which describes how the algorithm can be modified to compute similar scalar products (that is, bilinear symmetric forms). This gives algorithms for one of the usual scalar product of functions $\langle f|g\rangle = \int fg$, for various q-analogues, and for scalar products of MacMahon symmetric functions. This adaptability of our methodology to many generalizations of symmetric functions does much to validate our approach.

It is in the third part that we reap the rewards of the first two parts. The results are all combinatorial in nature. In fact, the third part can be read in isolation if the reader is willing to treat the algorithms of Part 2 as a black box:



Chapter 7 considers some classic examples, such as regular graphs and set covers, and places them in the context of species theory to illustrate how this is but one special instance of an infinity of such. We also consider a generalization of involutions related to Young tableaux. The final chapter considers some applications of Ore algebras, described in Chapter 4, to combinatorial enumeration.

Here is a sampling of some specific combinatorial problems that are well suited to an examination from a holonomic point of view, and are considered in this Part 3.

Problem 1. *Automatically determine the number of 4-regular simple graphs of size n in time linear in n.*

Problem 2. *Automatically determine a formula for the number of bounded standard Young tableaux of height* 4*.*

Problem 3. *Automatically determine the D_q-equation satisfied by the generating function of all plane partitions.*

Problem 4. *Automatically determine a generating function for symmetric function characters indexed by hook partitions.*

Problem 5. *Describe a unified approach for effective calculation of scalar products of functions.*

We conclude with some questions that arose in the course of this study but remain unanswered.

Summary of contributions of this thesis

The subject of this thesis sits at the intersection of computer algebra and algebraic combinatorics passing through the vehicle of algorithms on symmetric functions. The new results include a collection of new algorithms, complete with proofs of correctness and termination. These algorithms have been implemented, and tested in Maple. The code of the major algorithms, along with several key examples collected in Maple worksheets, are available on the author's web page, `http://www.lacim.uqam.ca/~mishna`.

More generally, we give a uniform approach to a wide variety of combinatorial problems. Although some of our examples fall under the umbrella of the methods of [36] and [34], the results we give are more general and can be applied to a more general set of situations. In particular, q-parameters enter our setup in a more natural way. Accommodating q-parameters is accomplished in part by using recent work of Chyzak and Salvy [16] on ∂-finite ideals for more general operator algebras.

The initial ideas for the central algorithms of Part 2 were first presented at the 14^{th} Formal Power Series and Algebraic Combinatorics conference in Melbourne, Australia [14], and the main proofs together with additional applications, appeared in the full article [15].

The author has also considered other aspects of computer algebra and combinatorics in [57], specifically, an extension of a decomposable structures applied to algorithm analysis. This work includes routines for the `combstruct` package, which is part of [17], and which are incorporated into the Maple library (versions 8 and greater). This work is not presented in this text.

Part 1

Theoretical Foundations

Summary of this part

In this part we describe the evolution of the notion of D-finite functions and their connections to combinatorics via symmetric functions. In particular, we determine closure operations for D-finite functions and describe consequences in the ring of symmetric functions. The second chapter describes the setting of holonomic functions; here we encounter the context in which we can describe and prove the algorithms which make the closure properties effective. The final chapter of this section generalizes some of the concepts to the wider context of Ore algebras with the aim of determining suitable q-analogues to the work of the earlier chapters. The background is mostly classical, although the discussion of q-analogues of symmetric functions in the context of ∂-finite functions is new.

Chapter I

D-finite functions

1.1 D-finite functions

One motivation for encoding combinatorial objects by formal series is to identify combinatorial operations with algebraic manipulations. Ideally, this process is automatic, and delegated to a computer algebra program. However, this is only feasible if the series has some sort of finite representation; for example, a functional equation that it satisfies. Here, we consider functions that satisfy a linear differential equation with polynomial coefficients. Throughout, $\mathbb{K}$ is a field of characteristic 0, typically $\mathbb{C}$ or $\mathbb{C}(t)$.

Definition 1.1 ***D-finite function.*** A function $f(x) \in \mathbb{K}[\![x]\!]$ is *D-finite* if it satisfies an equation of the form

$$P_d(x)f^{(d)}(x) + \ldots + P_1(x)f'(x) + P_0(x)f(x) = 0 \tag{1.1}$$

where $f^{(n)}(x) = \frac{d^n}{dx^n}f(x)$ is the n^{th} derivative of f, d is finite and $P_j(x) \in \mathbb{K}[x]$ for all j.

This is equivalent to the following condition

Proposition 1.1 (Alternate characterization of D-finiteness). *The function $f(x)$ is D-finite if $f(x)$ and all of its derivatives $f^{(n)}(x)$ span a finite dimensional vector subspace over the rational functions in x.*

> *Example.* The derivatives of $\sin(x)$ are either of the form $\pm\sin(x)$ or $\pm\cos(x)$. Thus, the vector space $\bigoplus_n \mathbb{K} \otimes \sin^{(n)}(x)$ is two dimensional and $\sin(x)$ is D-finite. D-finiteness is shown more simply by the equation $\sin''(x) + \sin(x) = 0$, which is of form Eq. (1.1).

1.1.1 Closure properties

Differential equations, and the functions which satisfy them, are well studied objects. As we shall see, the class of D-finite functions is strictly larger than the set of algebraic functions (it contains e^x, for example), yet possesses closure properties which are useful from both a combinatorial point of view and from an effective calculation point of view, making them well worthy of study. Stanley [74, 75, §6.4] popularized the study of D-finite functions and their closure properties under this view. The notion was further developed and then generalized to the multivariable case separately by Lipshitz [49] and Zeilberger [87, 88], where new uses emerged, notably the *creative telescoping* algorithm for producing automatic proofs of identities [89]. In general to say that an operation *preserves D-finiteness*, implies that given D-finite functions as arguments, the operation results in another D-finite function. It is even more interesting when this operation is effective. The following theorem introduces some basic properties.

Theorem 1.2 (Classic D-finite closure properties I).

1. *The set $\mathcal{D}$ of D-finite functions forms a sub-algebra of $\mathbb{K}[\![x]\!]$.*
2. *If $f(x) \in \mathbb{K}[\![x]\!]$ is D-finite and $g(x) \in \mathbb{K}[\![x]\!]$ is algebraic, then the composition $f(g(x))$ is also D-finite.*
3. *If $f(x)$ is algebraic, then $f(x)$ is D-finite.*

Here we only present the proof of Part 1 of Theorem 1.2 to illustrate a typical argument.

Proof. (Stanley). For any $f \in \mathbb{K}[\![x]\!]$, denote by V_f the vector space over $\mathbb{K}(x)$ spanned by the derivatives of f: $V_f = \bigoplus_n \mathbb{K}(x) \otimes f^{(n)}(x)$. For any $f, g \in \mathcal{D}$, and $\alpha, \beta \in \mathbb{K}$ set $u = \alpha f + \beta g$. Since $u' = \alpha f' + \beta g'$, and in general $u^{(n)} = \alpha f^{(n)} + \beta g^{(n)}$, then the set of derivatives of u are contained in $V_f + V_g$. Then, taking dimensions over $\mathbb{K}[\![x]\!]$,

$$\dim V_u \leq \dim (V_f + V_g) \leq \dim V_f + \dim V_g.$$

Both $\dim V_f$ and $\dim V_g$ are finite since f and g are D-finite, giving that $\dim V_u$ is finite. Thus, u is D-finite.

Next, we show that the product of two D-finite functions is D-finite. First, consider $V = \mathbb{K}[\![x]\!]$ as a vector space over $\mathbb{K}(x)$. Suppose that $u = fg$. Recall that $u' = f'g + fg'$.

There is a unique linear transformation $\phi : V_f \otimes_{\mathbb{K}(x)} V_g \to V$ satisfying $\phi(f^{(n)} \otimes g^{(k)}) = f^{(n)}g^{(k)}$. The derivatives of u are clearly contained in the image of ϕ, and thus if we compare their respective dimensions

$$\dim V_u \leq \dim V_f \otimes V_g \leq (\dim V_f)\,(\dim V_g) \leq \infty.$$

Once again we conclude that $\dim V_u$ is finite, giving the D-finiteness of u. ✪

The following consequence of Part 2 of Theorem 1.2 warrants explicit mention, as it represents a common situation in the forthcoming applications.

Corollary 1.3. *If $p(x)$ is a polynomial then $\exp(p(x))$ is D-finite with respect to the x variables.*

Frequently one can determine that a function is *not* D-finite by applying the following result.

Proposition 1.4. *Any D-finite function $f \in \mathbb{C}[\![x]\!]$ which is analytic on an open ball in $\mathbb{K}$ has a finite number of singularities in $\mathbb{K}$.*

This can be proved by establishing that any singularity of f must be a zero of one of the coefficient polynomials P_d of Eq. (1.1). These zeros are clearly finite in number.

For example, $1/\sin(x)$ is not D-finite. Thus, the composition of D-finite functions is not always D-finite, as both $1/x$ and $\sin x$ are D-finite. Another noteworthy example of a function which is not D-finite is $\exp(\exp(x))$. It is worthwhile to ask if the algebraicity requirement in Part (2) of Theorem 1.2 is the most general condition that applies in all cases. The answer, from the following discussion, is yes.

1.1.2 Composition of functions

There are a number of known results which help describe which compositions of functions are D-finite.

Harris and Sibuya in [42] (and in a result later generalized by Sperber and by Singer [71, 73]) proved that if both g and $1/g$ are D-finite, then g is of the form $A\exp(B)$, where A and B are both algebraic.

We can deduce the D-finiteness of the composition $f(g)$ of any rational f, and any g of the form $g = A\exp(B)$ as above algebraic. One way to phrase this result is to say that functions of this form $A\exp(B)$, are contained in the *D-finite composition closure* of the rational functions.

This problem can be posed for classes other than the rationals. For example, the algebraic functions are contained in their own D-finite composition closure since the composition of two algebraic functions is algebraic, and thus D-finite. It should be possible to answer questions of the following type using analytic arguments, or the characterization by holonomy developed in the next chapter.

Problem 6. *Given a family of functions $\mathcal{F}$, determine sets of functions g such that $f(g)$ is D-finite for all members f of the family $\mathcal{F}$.*

The following variants of the problem may also be interesting.

Problem 7. *Given a D-finite function f determine the set of functions g such that $f(g)$ is D-finite.*

Problem 8. *Given a D-finite function g, determine the set of functions g such that $f(g)$ is D-finite when f is.*

There are several other useful closure properties which generalize to the multivariable case, hence we consider them in further depth in Section 1.3.

1.1.3 Effective closure properties

Note, the proof of Theorem 1.2 offers no direct indication of how to translate the differential equations satisfied by $f(x)$ and $g(x)$ into one satisfied by $(f+g)(x)$ or $fg(x)$. An *effective closure property* refers to a closure property in combination with an explicit procedure for the computation of the differential equation satisfied by the resulting function. This topic will be considered in closer detail in Section 3.4.

1.2 P-recursive functions

There are other important properties of D-finite functions. A D-finite function in one variable has coefficients that satisfy a finite recurrence.

Definition 1.2 ***P-recursive function.*** A function $p : \mathbb{N} \to \mathbb{K}$ is said to be *P-recursive* if it satisfies a homogeneous linear recurrence with polynomial coefficients in n. In this case, we also say that the sequence given by $(p(n))_n$ is P-recursive.

A series $F(x) = \sum_{n\geq 0} p(n)x^n$ is D-finite if and only if $p(n)$ is P-recursive. One way to view this correspondence is a map from one sort of finite description (a differential equation) to another, (a recurrence). Similarly, other connections between notions of finite description are interesting and useful. More precisely, given two rings, each with their own proper notion of D-finite, (or, more generally, finite description), we search for ring homomorphisms which map D-finite elements to other D-finite elements.

In the next section we define what it means to be D-finite in the ring $\mathbb{K}[\![x_1, \ldots, x_n]\!]$ and in later sections we consider the ring of symmetric functions. In the final chapter of this part we present a very generalized notion of D-finite, called ∂-finite, which includes both "classical" D-finiteness and P-recursiveness as a special case.

1.3 D-finite functions in multiple variables

D-finite power series in n variables, $n \geq 1$ are defined using a natural generalization of the univariate version, stated in terms of vector spaces.

Definition 1.3 ***D-finite functions of multiple variables.*** A formal power series $f \in \mathbb{K}[\![x_1, \ldots, x_n]\!]$ is *D-finite* in $x_1, \ldots, x_n$ (or *with respect to* $x_1, \ldots, x_n$) when the set of all partial derivatives, $\partial^{i_1+\cdots+i_n} f/\partial x_1^{i_1} \cdots \partial x_n^{i_n}$, spans a finite-dimensional vector space over the field $K(x_1, \ldots, x_n)$.

Any set of differential equations which illustrates this property is called a *D-finite description* of a function.

Many of the properties of univariate power series carry over to this case. The set of D-finite power series is a $\mathbb{K}$-subalgebra of $\mathbb{K}[\![x_1, \ldots, x_n]\!]$ for the usual product of series. Furthermore, algebraic functions are D-finite. The following theorem summarizes the main closure properties of this family of series.

Theorem 1.5 (Classic D-finite closure properties II).

1. *If f is D-finite with respect to $x_1, \ldots, x_n$ then for any subset $\{x_{i_1}, \ldots, x_{i_k}\}$ of*

those variables, the specialization $F|_{x_{i_1}=\cdots=x_{i_k}=0}$ *is D-finite with respect to the remaining variables;*

2. *If* $f(x_1,\ldots,x_n)$ *is D-finite, and for each* $1 \leq i \leq n$, $g_i(y_1,\ldots,y_m)$ *is an algebraic function of* $\mathbf{y} = y_1,\ldots,y_m$, *then whenever the substitution* $f(g_1(\mathbf{y}),\ldots,g_n(\mathbf{y}))$ *is well-defined as a power series, it is D-finite with respect to* $\mathbf{y}$*;*

3. *If* $f(x_1,\ldots,x_n)$ *is D-finite, then* $\int_0^{x_n} f(x_1,\ldots,x_{n-1},t)\,dt$ *is a D-finite function with respect to* $x_1,\ldots,x_n$*;*

4. *If* f *and* g *are D-finite in the variables* $x_1,\ldots,x_{m+n}$, *then the Hadamard product*[1] $f \times g$ *with respect to the variables* $x_1,\ldots,x_n$ *is D-finite in* $x_1,\ldots,x_{m+n}$.

These properties are now classical. The first two are elementary, and can be proved with basic vector space arguments as in the univariate case. Property (3) is considered in [88]. Property (4) relies on more delicate properties of dimension and is due to Lipshitz [48].

Later sections describe how to make these closure properties effective.

[1]Recall that if $u^\alpha = u_1^{\alpha_1}\cdots u_k^{\alpha_k}$, then the Hadamard product of two series is

$$\sum_{\alpha\in\mathbb{N}^k} a_\alpha \mathbf{u}^\alpha \times \sum_{\beta\in\mathbb{N}^k} b_\beta \mathbf{u}^\beta = \sum_{\alpha\in\mathbb{N}^k} a_\alpha b_\alpha \mathbf{u}^\alpha.$$

Chapter II

Symmetric functions

The notation pertaining to symmetric functions herein follows [52]. Here are some of the basic definitions.

A *partition of* n, denoted $\lambda \vdash n$, is a weakly decreasing sequence of positive integers $\lambda = (\lambda_1, \ldots, \lambda_k)$, whose sum $\lambda_1 + \cdots + \lambda_k$ is equal to n. Denote the set of all partitions $\mathbb{P}$. Each λ_i is a *part*. The *length* of a partition, $l(\lambda)$, is the number of parts. The *conjugate partition* of λ is the partition λ' defined by $\lambda'_i = \text{card}\{j : \lambda_j \geq i\}$. An alternative notation for the partition λ is $\lambda = 1^{\mathbf{m}_1} \cdots k^{\mathbf{m}_k}$, which means that i occurs $\mathbf{m}_i = \mathbf{m}_i(\lambda)$ times in λ, for $i = 1, 2, \ldots, k$. These definitions are best visualized with the aid of a diagram. The *Young diagram* of a partition is a subarray where each part corresponds to a row of blocks. The number of blocks in row i is λ_i. The conjugate partition is the partition determined by flipping the diagram of λ along the principal diagonal, as in the following figure, illustrating the partition $\lambda = (5, 4, 1, 1)$ of 11 (equivalent to $1^2\,4\,5$ in the alternate notation), and its conjugate $\lambda' = (4, 2, 2, 2, 1)$.

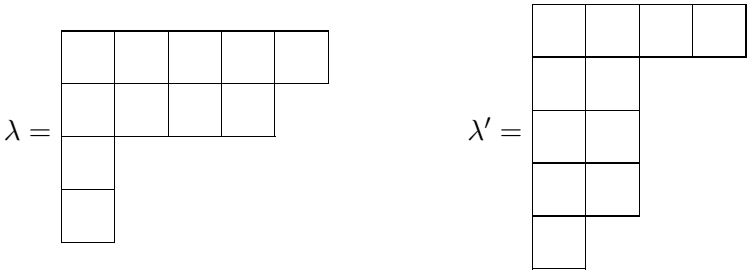


Figure **2.1** Young diagrams of a partition and its conjugate

A polynomial $f(x_1, x_2, \ldots, x_n)$ is said to be *symmetric* if for any permutation of n, $\sigma \in$

$\mathbb{S}_n$,

$$f(x_1, x_2, \ldots, x_n) = f(x_{\sigma(1)}, x_{\sigma(2)}, \ldots, x_{\sigma(n)}).$$

Denote by Λ_n the vector space over $\mathbb{K}$ of symmetric polynomials in n variables. The *monomial symmetric polynomial,* $m_\alpha \in \Lambda_n$, is defined for a given $\alpha = (\alpha_1, \ldots, \alpha_k) \in \mathbb{N}^k$, $k < n$, by the sum over all permutations $\sigma \in \mathbb{S}_n$ of distinct monomials $x_{\sigma(1)}^{\alpha_1} x_{\sigma(2)}^{\alpha_2} \cdots x_{\sigma(k)}^{\alpha_k}$.

The set of monomial polynomials m_λ where λ is a partition form a vector space basis of Λ_n. We generalize this basis to an infinite number of variables by defining $m_\lambda(x_1, x_2, \ldots)$ as the formal series,

$$m_\lambda(x_1, x_2, \ldots) = \sum_{\mathbf{n} \in N^{l(\lambda)}}^{*} x_{n_1}^{\lambda_1} \ldots x_{n_{l(\lambda)}}^{\lambda_{l(\lambda)}},$$

where the star indicates that the sum is over all $l(\lambda)$-tuples of positive integers which yield distinct monomials. Accordingly, we define Λ as the vector space generated by the basis of all m_λ, with λ a partition of any integer. This is derived more formally in [52] as an inverse limit of vector spaces.

There are other families of symmetric functions that can be defined with the aid of the monomial polynomial: the *power* symmetric functions,

$$p_n = m_{(n)} = x_1^n + x_2^n + \ldots,$$

the *elementary* symmetric functions,

$$e_n = m_{(1^n)} = x_1 x_2 \cdots x_n + x_2 x_3 \cdots x_{n+1} + \cdots$$

and the *complete homogeneous* symmetric functions,

$$h_n = \sum_{\lambda \vdash n} m_\lambda,$$

where the sum is over all partitions of n. We write $p_\alpha = p_{\alpha_1} \ldots p_{\alpha_k}$, and define e_α and h_α similarly. Using the usual notion of monomial degree, we can impose a grading on this vector space, $\Lambda = \bigoplus_k \Lambda^k$, where Λ^k is the vector space of homogeneous symmetric functions of degree k in the symmetric variables x_i.

There are several known bases of Λ^k, described as sets indexed by partitions λ of k. The principal among them are: the monomial (m_λ), the elementary (e_λ), the complete

homogeneous (h_λ), and the power (p_λ). There is another basis, the Schur basis (s_λ) which is arguably the most important, and Section 2.0.1 describes it in detail.

Since Λ is clearly closed under multiplication, it can alternatively be viewed as a polynomial ring generated by $p = p_1, p_2, \ldots$. In order to distinguish this point of view from the previous, we write $\mathbb{K}[p]$. This is isomorphic to Λ as it is a different point of view with respect to the same set of functions. Similarly, the h_n and e_n form ring bases and these will be referred to as the h and e bases respectively.

Generating series of symmetric functions live in larger rings of symmetric series, such as $\mathbb{K}[\![p]\!] = \mathbb{K}[\![p_1, p_2, \ldots]\!]$, and $\mathbb{K}[t][\![p]\!] = \mathbb{K}[t][\![p_1, p_2, \ldots]\!]$. For example, in $\mathbb{K}[t][\![p]\!]$, we have the generating series of complete homogeneous and elementary functions:

$$\mathcal{H}(t) = \sum_{n\geq 1} h_n t^n = \exp\left(\sum_m p_m \frac{t^m}{m}\right),$$
$$\mathcal{E}(t) = \sum_{n\geq 1} e_n t^n = \exp\left(\sum_m (-1)^m p_m \frac{t^m}{m}\right).$$

Often we will refer to $\mathcal{H}(1)$ as simply $\mathcal{H}$ and $\mathcal{E}(1)$ as $\mathcal{E}$.

2.0.1 Schur functions

Owing to their link to representations of the symmetric group, Schur functions form the most interesting basis of the symmetric functions. There are several different ways to define a Schur function indexed by a partition λ. Here, we write them as a determinant of an $l(\lambda) \times l(\lambda)$ matrix of complete homogeneous functions,

$$s_\lambda = \det\left([h_{\lambda_i - i + j}]_{1\leq i,j\leq l(\lambda)}\right). \tag{2.1}$$

This is generalizable to *skew-schur* functions

$$s_{\lambda/\mu} = \det(h_{\lambda_i - \mu_j - i + j}). \tag{2.2}$$

Here, $h_{-n} = h_{|n|}$, and $h_0 = 1$. For example, $s_n = h_n$ and $s_{1^2} = h_1^2 - h_2 = e_2$, (and in general, $s_{1^n} = e_n$). The Schur functions play an important role in defining the link between the irreducible representations of the symmetric group and symmetric functions. In fact, the irreducible representations $\mathcal{V}_\lambda$ of $\mathbb{S}_n$ are indexed by λ, partitions of n, and their characters $\chi(\mathcal{V}_\lambda)$ are directly linked to Schur functions via the *Frobenius map*. For more details on this interesting link, see [68].

2.1 Operations on symmetric functions

2.1.1 The Kronecker product of symmetric functions

It is clear that in the ring of symmetric functions that the "usual" polynomial multiplication serves as a product. The connection coefficients $c^{\nu}_{\mu\lambda}$ of products of Schur functions:

$$s_\lambda s_\mu = \sum_\nu c^{\nu}_{\mu\lambda} s_\nu,$$

are quite interesting. The $c^{\nu}_{\mu\lambda}$ are non-negative integers known as the Littlewood-Richardson numbers and although they have many combinatorial interpretations, and algorithms for their computation (see [75, §7.15], for example), none of them are particularly explicit.

There is a second interesting product, which arises in computing characters of the symmetric group. We follow Macdonald and call it the *Kronecker product* of symmetric functions and denote it by $*$. It was first described by Redfield as the *cap product* of symmetric functions and was rediscovered by Littlewood [50]. This product can be defined in representation theory terms using induced characters of representations of the symmetric group, however here we use the following relation to the power symmetric functions, and extend linearly:

$$p_\lambda * p_\mu = \delta_{\lambda\mu} z_\lambda p_\lambda, \tag{2.3}$$

where $\delta_{\lambda\mu} = 1$ if $\lambda = \mu$ and 0 otherwise and we define the normalization constant

$$z_\lambda = 1^{\mathbf{m}_1}\mathbf{m}_1! \cdots k^{\mathbf{m}_k}\mathbf{m}_k!.$$

Calculating the connection coefficients $\gamma^{(\rho)}_{\lambda,\mu}$ for this product

$$s_\lambda * s_\mu = \sum_\rho \gamma^{(\rho)}_{\lambda,\mu} s_\rho$$

is also challenging, and quite interesting. There are some combinatorial interpretations which have yielded results when λ, ρ and μ are of a particular form, for example, the work of Goupil and Schaeffer [38] or Rosas [65].

There is a direct correspondence with irreducible representations,

$$\chi(\mathcal{V}_\lambda \otimes \mathcal{V}_\mu) = \sum_\rho \gamma^{(\rho)}_{\lambda,\mu}\, \chi(\mathcal{V}_\rho).$$

2.1.2 Scalar product and coefficient extraction

The ring of symmetric series is endowed with a scalar product defined as a symmetric bilinear form such that the bases (h_λ) and (m_λ) are dual to each other:

$$\langle m_\lambda, h_\mu \rangle = \delta_{\lambda\mu}. \tag{2.4}$$

At times we may emphasize which variables are annihilated by the scalar product by a subscript. For example, $\langle f(p_1, t), g(p_1) \rangle_p$ is a function of t. MacMahon [53] describes actions which closely resemble the scalar product, or more accurately the adjunction relative to it. Section 2.2.2 describes an operator acting on symmetric function which he uses in much the same way that we use the scalar product. It is Redfield [64] who first formulates it in this way and makes the important observation that it is symmetric. Subsequently, this product is rediscovered, and initially (incorrectly) attributed to Hall [40].

The constant $z_\lambda = 1^{\mathbf{m}_1}\mathbf{m}_1! \dots k^{\mathbf{m}_k}\mathbf{m}_k!$, plays the role of the square of a norm of p_λ in the following important formula:

$$\langle p_\lambda, p_\mu \rangle = \delta_{\lambda\mu} z_\lambda. \tag{2.5}$$

The Schur basis is an orthonormal symmetric function basis under this scalar product. In fact, Schur functions can be defined as the result of applying the Gramm-Schmidt process for orthogonalizing a basis, applied to the monomial basis with the partitions ordered lexicographically[1].

2.1.3 Plethysm

Plethysm is a way to compose symmetric functions. It can be defined on the power sum ring basis and extended to all of $\mathbb{K}[\![p]\!]$. It is defined by $p_n[p_m] = p_{nm}$ and extended by

$$(fg)[h] = f[h]\, g[h], \quad (f+g)[h] = f[h] + g[h] \quad \text{and} \quad p_n[g] = g[p_n]. \tag{2.6}$$

To clarify this action, observe that

$$f(p_1, p_2, \dots, p_k, \dots)[p_n] = f(p_{1n}, p_{2n}, \dots, p_{kn}, \dots). \tag{2.7}$$

[1] In such an ordering, $1^n < 1^{n-1}2 < \cdots < n$.

It can also be defined on the level of Λ :

$$p_k\left[f(x_1, x_2, \ldots)\right] = f(x_1^k, x_2^k, \ldots). \tag{2.8}$$

For more details see Macdonald [52, p. 77].

2.2 Differential operators for symmetric functions

The subject of differential operators arises naturally in the study of the scalar product of symmetric functions.

2.2.1 The adjoint of multiplication

The *adjoint*, $\perp$, of multiplication with respect to the symmetric scalar product is an endomorphism of the ring of symmetric series $\perp\colon \mathbb{K}[\![p]\!] \to \mathbb{K}[\![p]\!]$ which is defined by the following relation

$$\left\langle G, F^{\perp} H \right\rangle = \langle FG, H \rangle$$

for all $F, G, H \in \mathbb{K}[\![p]\!]$. We can describe this action on the power, complete homogeneous and Schur bases in a natural way. These are derived in [52, §I 5.], and we summarize them here.

Schur functions satisfy the relation $\langle s_\lambda, s_\mu s_\nu \rangle = \left\langle s_{\lambda/\mu}, s_\nu \right\rangle$ thus,

$$s_\mu^{\perp} s_\lambda = s_{\lambda/\mu}. \tag{2.9}$$

The action on orthonormal bases (h_λ) and (m_λ) can be determined similarly. Create a new partition $\mu = \lambda \cup \nu$ by ordering the parts $\lambda_1, \ldots, \lambda_{l(\lambda)}, \nu_1, \ldots, \nu_{l(\nu)}$ into a partition μ. Then we have $h_\lambda^{\perp} m_\mu = 0$ unless $\mu = \lambda \cup \nu$ for some partition ν. One implication of this, is that $h_n^{\perp} m_\mu = 0$ unless μ has at least one part equal to n. If so, then $h_n^{\perp} m_\mu = m_\nu$ where ν is the μ with exactly one n part removed. It is this removal action that led MacMahon to call $h_n^{\perp}$ the *obliterating operator* [53, §26].

The adjoint of multiplication by a power symmetric function is easiest to describe in terms of a differential operator: $p_n^{\perp} = n\frac{\partial}{\partial p_n}$. As the p_n form a ring basis of $\mathbb{K}[p]$ over $\mathbb{K}$, this gives a way to describe the adjoint of multiplication by any symmetric function. If $F \in \mathbb{K}[\![p]\!]$ is given as $F(p_1, p_2, \ldots)$, then $F^{\perp}$ is the differential operator $F(\frac{\partial}{\partial_{p_1}}, 2\frac{\partial}{\partial_{p_2}}, \ldots)$, a linear differential operator with coefficients in $\mathbb{K}$.

2.2.2 Hammond operators

Although we will largely focus on the power differential operators, historically some attention has been paid to the complete homogeneous and elementary differential operators. Hammond introduced a family of such operators in the 19^{th} century [41]. MacMahon made extensive use of them as well [53]. The *Hammond operator* H_n was originally defined in a way that is equivalent in modern notation to $h_n^\perp$. MacMahon describes how to associate differentiation with a combinatorial notion of obliteration and applies it to symmetric functions to count matrices with positive integer entries and limitation on the row and column sums. We describe these ideas in Part 3.

2.2.3 The Heisenberg Lie algebra

It is an interesting aside to note other formalisms in which these operators have been studied. Define the family of operators $\pi_n : \mathbb{K}[\![p]\!] \to \mathbb{K}[\![p]\!]$ for $n \in \mathbb{Z}$ as follows: when n is positive, π_n is multiplication by p_n; when n is negative, $\pi_n = p_{|n|}^\perp$; and π_0 is the identity. The algebra is formed by the linear span of these elements and has the following bracket operator:

$$[\pi_m, \pi_n] = n\pi_0\delta_{m+n,0}.$$

Jing [43] considers various classes of classical symmetric operators from the viewpoint of vertex operators.

2.3 D-finite symmetric series

Gessel defines D-finiteness for series in an infinite number of variables by generalizing property Theorem 1.5(1). A series $F \in \mathbb{K}[\![x_1, x_2, \dots]\!]$ is said to be *D-finite with respect to the* x_i if the specialization of all but a finite choice S of variables to 0 is D-finite for any choice of S. In this case, all of the properties in Theorem 1.5, except (2), hold in the infinite multivariate case. Proposition 2.1 gives an analogue for (2), a result for algebraic substitution in the infinite case.

This definition is tailored to symmetric series $\mathbb{K}[\![p]\!]$ by considering the power sum basis.

Definition 2.1 ***D-finite symmetric series.*** A symmetric series in $\mathbb{K}[t][\![p]\!]$ is said to be *D-finite* when it is D-finite with respect to $p_1, p_2, \dots$ and the t variables.

S	generating series
$\{(n)\}_n$	$\mathcal{H} = \exp(\sum_n p_n/n)$
$\{(1^n)\}_n$	$\mathcal{E} = \exp(\sum_n (-1)^{n+1} p_n/n)$
all partitions	$\mathcal{S} = \mathcal{H}\left[e_1 + e_2\right]$
λ all parts even	$\mathcal{S}\mathcal{E}(-1)$
λ' all parts even	$\mathcal{S}\mathcal{H}(-1)$

TABLE **2.1** Generating series for $\sum_{\lambda \in S} s_\lambda$ for different families of partitions.

Example. Two simple examples of D-finite series are

$$\mathcal{H} = \exp\left(\sum_n p_n/n\right) \qquad \text{and} \qquad \mathcal{E} = \exp\left(\sum_n (-1)^{n+1} p_n/n\right).$$

In each case a specialization of all but a finite number of p_n to 0 results in a function of the form exp(polynomial), which is clearly D-finite. These and other other examples from Section 2.6 are given in Table 2.1.

Other definitions of D-finite are possible, in particular with respect to other bases. The power sum basis is a useful choice since we have a natural differentiation connected to the scalar product and furthermore, as an application one can connect to the standard D-finite definition via the scalar product, as we shall see in Theorem 2.7. A different choice of basis leads to a different set of D-finite functions, as the next example illustrates.

Example. The series $y = \exp(\sum_n h_n)$ is clearly D-finite with respect to the h basis, just as $\sum_n h_n = \exp(\sum p_i/k)$ is D-finite with respect to the power sum basis. However, if we re-write each h_n in the power sum basis, we see that $y = \exp(\exp(p_i/k))$ and, as $\exp(\exp(p_1))$ is not D-finite considering the example on page 21, neither is y D-finite with respect to the power sum basis.

2.4 Closure properties of D-finite symmetric series

Many symmetric function operators are closed under D-finiteness. In this section we explore plethysm, the Kronecker product, and the scalar product. The usual product of polynomials is also D-finite.

2.4.1 Algebraic substitution

The change of basis map from the h basis to the power sum basis in the earlier discussion illustrates that when an infinite number of variables are involved, some care must be taken in the study of D-finiteness with respect to substitution. Here is a more restrictive version of Theorem 1.5(2) suitable for the case of an infinite number of variables.

Proposition 2.1 (D-finite algebraic substitution). *Let x and y respectively denote the (possibly infinite) sets of variables $x = x_1, x_2, \ldots$ and $y = y_1, y_2, \ldots$. Suppose $f(x)$ is a D-finite function with respect to the x variables. Suppose, $x_i = g_i(y)$ for a finite number of i and furthermore:*

1. *Each g_i is an algebraic function of y;*
2. *For each k there exists an N_k such that g_i is not a function of y_k for $i > N_k$;*
3. *The substitution $F(g_1, g_2, \ldots)$ is defined as a power series;*

Then $f(g_1, g_2, \ldots)$ is a D-finite function of y.

Proof. Fix $n \in \mathbb{N}$ and let r_n be the specialization $y_i = 0$ for $i > n$. If $N = \max_{1 \leq i \leq n}\{N_i\}$, then $r_n(f(g_1, g_2, \ldots)) = r_n f(g_1, \ldots g_N)$. As a function in a finite number of variables $F(y_1, \ldots, y_N)$ is D-finite, and the substitution $y_i = g_i(t_1, \ldots, t_n, 0, \ldots)$ is finite and algebraic. The result follows by an application of Theorem 1.5(2). ✪

An example of a ring morphism which satisfies these three properties, commonly denoted ω, sends $h_i \mapsto e_i$ and preserves D-finiteness since $\omega(p_\lambda) = \mathrm{sgn}(\lambda) p_\lambda$ with $\mathrm{sgn}(\lambda) = (-1)^{n-l(\lambda)}$. Since this morphism is equivalent to the algebraic substitution, $p_n \mapsto (-1)^{n-1} p_n$, in which each p_i is used exactly once, it is D-finite preserving. Notably, it has the following effect on the Schur functions: $\omega(s_\lambda) = s_{\lambda'}$, where λ' is the conjugate partition to λ.

Plethysm is also clearly a composition that satisfies these conditions. In fact, many of the most interesting symmetric series can be written as a plethysm of two symmetric series. Gessel gives the following result for preservation of D-finiteness under plethysm in a simple case, essentially restating Corollary 1.3. However, we give the proof to illustrate a typical argument to show an operation preserves D-finiteness. It is followed by the argument for the more general case, using Theorem 2.1.

Proposition 2.2 (Plethysm and D-finiteness I; Gessel). *If g is a polynomial in the p_i's, then $\mathcal{H}[g]$ and $\mathcal{E}[g]$ are D-finite.*

Proof. Define $R_n : \mathbb{K}[\![p]\!] \to \mathbb{K}[\![p_1 \ldots, p_n]\!]$ as the ring homomorphism which maps p_m to 0 for $m > n$. To establish that $\mathcal{H}[g]$ is D-finite we show that for any n, $R_n(\mathcal{H}[g])$ is D-finite in the variables $p_1, p_2, \ldots, p_n$.

Remark, that if $g = g(p_1, \ldots, p_N)$,

$$\begin{aligned} R_n(\mathcal{H}[g]) &= R_n\left(\exp\left(\sum_{k\geq 1}\frac{p_k}{k}[g]\right)\right) \\ &= R_n\left(\exp\left(\sum_{k\geq 1}\frac{g(p_k,\ldots,p_{kN})}{k}\right)\right) \\ &= \exp\left(\sum_{k=1}^{n} R_n\left(g(p_k,\ldots,p_{kN})/k\right)\right), \end{aligned}$$

which is D-finite by Proposition 1.5. The D-finiteness of $\mathcal{E}[g]$ is proven similarly. ✪

Theorem 2.3 (Plethysm and D-finiteness II). *Let f be any D-finite symmetric series and g any symmetric series which is algebraic. Then the plethysm $f[g]$ is a D-finite symmetric series.*

Proof. The plethysm $f[g]$ is algebraic substitution $p_n = p_n[g]$ in f. The hypotheses of Proposition 2.1 applies. Each of these substitutions is an algebraic function and, as the plethysm will always be defined as a power series, we can conclude that $f[g]$ is D-finite. ✪

2.5 Symmetric function specializations

As we have seen, a symmetric function can be viewed to be either an element of Λ or $\mathbb{K}[p]$. In either case, the variables are independent and we can therefore consider homomorphisms to other rings, defined by the action on either the symmetric variables $x_1, x_2, \ldots$, or the ring basis p. Let $\mathcal{R}$ be a commutative $\mathbb{K}$-algebra with identity. A *specialization* of Λ is a ring homomorphism $\phi : \Lambda \to \mathcal{R}$. Similarly, a specialization of the

power sum basis is a homomorphism from $\mathbb{K}[p]$ to $\mathcal{R}$, or $\mathbb{K}[\![p]\!]$ to $\mathcal{R}$. We are interested in homomorphisms that are D-finite preserving. The notation here is from [75, §7.8].

By Theorem 1.5(1), we have that the *exponential specialization*

$$\mathrm{ex} : \mathbb{K}[p] \to \mathbb{K}[\![t]\!]$$

which sends p_1 to t and p_n to 0 when $n > 1$, maps a D-finite symmetric series to a D-finite univariate power series, and is thus D-finite preserving. This action, and a generalization of it, will be treated in finer detail in the context of the scalar product of symmetric functions, in the next section.

A second simple example is the *reduction specialization*, $r_n : \Lambda \to \Lambda_n$ defined by specializing variables to zero:

$$r_n(f(x_1, x_2, \ldots)) := f(x_1, \ldots, x_n, 0, 0, \ldots).$$

This is often implied by the notation $f(x_1, \ldots, x_n)$. Note, we have that each of the following sets,

$$\{r_n(m_\lambda)\}, \{r_n(e_\lambda)\}, \{r_n(p_{\lambda'})\}, \text{ and } \{r_n(h_\lambda)\},$$

where λ ranges over partitions of length less than n form $\mathbb{K}$-bases of Λ_n. Under this specialization the set $\{r_n(p_\lambda)\}$ is not a basis for Λ_n, and thus we cannot take it for granted that this map automatically preserves D-finiteness in the smaller ring. In fact, r_1 does *not* preserve D-finiteness. If we write it as a specialization of the p_i variables, we see that it maps p_n to x_1^n, and hence the D-finite symmetric function $\sum_\lambda p_\lambda$ is mapped to

$$r_1\left(\sum_\lambda p_\lambda\right) = r_1\left(\prod_n \frac{1}{1-p_n}\right) = \prod_n \frac{1}{1-x_1^n},$$

which is not a D-finite function in x_1 since it possesses an infinite number of singularities, contrary to the characterization in Proposition 1.4. However, if we consider the subset of Λ which is also D-finite with respect to the x_i variables, then r_n is a D-finite preserving homomorphism by the definition of D-finiteness.

2.5.1 Some q-specializations

A particularly interesting case, the *q-specializations*, occurs when $\mathcal{R} = \mathbb{K}(q)[\![t]\!]$. Here, q is a formal parameter, assumed not to be root of 1. These are investigated in further detail in Section 4.4.

First we introduce some basic notation. Define the *q-factorial* and *q-binomial* respectively as

$$(q)_n = (1-q)(1-q^2)\cdots(1-q^n) \qquad \binom{n}{k}_q = \frac{(q)_n}{(q)_k(q)_{n-k}}. \tag{2.10}$$

More generally, we have the q-Pochhammer function with respect to a variable a

$$(q|a)_n = (1-a)(1-aq)\cdots(1-aq^{n-1}),$$

with $(q|q)_n = (q)_n$. Finally, define a second q-analogue of the factorial,

$$n!_q = (1+q)(1+q+q^2)\cdots(1+q+\ldots+q^{n-1}).$$

There is a natural q-analogue of the exponential specialization. Define

$$\mathrm{ex}_q : \Lambda \to \mathbb{K}(q)[\![t]\!]$$

as the symmetric function specialization which sends x_n to $tq^{n-1}(1-q)$, for $1 \leq i$. This is called the *q-analogue of the exponential specialization* in [75, §7.8]. Its effect on the power basis is follows:

$$\mathrm{ex}_q\left(p_n\right) = \mathrm{ex}_q\left(x_1^n + x_2^n + \ldots\right) = \frac{(1-q)^n t^n}{(1-q^n)}.$$

The ring morphism ex_q is expressed in the other symmetric bases as the following:

$$\mathrm{ex}_q\left(h_n\right) = t^n/n!_q \quad \text{and} \quad \mathrm{ex}_q\left(e_n\right) = q^{\binom{n}{2}} t^n/n!_q. \tag{2.11}$$

This is called a q-analogue of ex because

$$\lim_{q\to 1} \mathrm{ex}_q\left(F(t)\right) = \mathrm{ex}(F)(t).$$

Later, we determine how this specialization fits into the discussion of D-finite preserving operations, although we first need a notion of D-finite in the ring $\mathbb{K}(q)[\![t]\!]$. We address this question in Chapter 4 after having developed suitable machinery.

Now we describe a second q-specialization. The *principal specialization* is defined as a Λ specialization by $\mathrm{ps}_n(x_i) = q^{i-1}$ if $i \leq n$ and 0 otherwise. Defined as a $\mathbb{K}[p]$ specialization, it maps p_k to $\frac{1-q^{k(n+1)}}{1-q^k}$. Under this map, we have

$$\mathrm{ps}_n(h_k) = \binom{n+k-1}{k}_q \quad \text{and} \quad \mathrm{ps}_n(e_k) = q^{k(k-1)/2}\binom{n}{k}_q$$

If we let $n \to \infty$, we have a limiting value, ps. Under this map,

$$\mathrm{ps}(p_k) = 1/(1-q^k), \quad \mathrm{ps}(h_k) = 1/(q)_k \quad \text{and} \quad \mathrm{ps}(e_k) = q^{k(k-1)/2}/(q)_k.$$

These are all obtained by simple combinatorial reasoning (see [75, §7.8]). We note that for $F \in \Lambda^n$,

$$\mathrm{ex}_q(F) = (1-q)^n t^n \,\mathrm{ps}(F).$$

To conclude this discussion, we define a specialization of a specialization, ps_n^1,

$$\mathrm{ps}_n^1(F) = \lim_{q \to 1} \mathrm{ps}_n(F).$$

For $F \in \Lambda$, this is also denoted $F(1^n)$. If we treat n as a variable, the action of this maps $p_k(x_1, x_2, \ldots) = x_1^k + x_2^k + \ldots + x_n^k + \ldots$ to

$$\mathrm{ps}_n(p_k(1, 1, \ldots, 1, 0, \ldots)) = \underbrace{1 + 1 + \ldots + 1}_{n \text{ times}} + 0 + 0 \ldots = n$$

for all k. If we view n as a variable, this will be a function in n. This may not be D-finite preserving even in cases when the resulting series in n makes sense, for example, $\exp(\sum_k p_n^k/k!)$ is a D-finite symmetric function, whereas the image under this specialization, $\exp(\exp(n))$ is not.

2.5.2 A refined notion of D-finite symmetric series

Each of the above specializations has a simple description in terms of the x_i (or symmetric) variables, that is when viewed as a function in Λ. Clearly many of the specializations of x are D-finite preserving *with respect to the x variables*. This suggests the following problem.

Problem 9. *Characterize D-finite symmetric series which, when viewed as elements of Λ, are D-finite with respect to the symmetric variables, $x_1, x_2, \ldots$.*

In simple cases, we have some positive results.

Proposition 2.4. *The symmetric series $\mathcal{E}$ and $\mathcal{H}$ are D-finite with respect to the x_i variables.*

Proof. Consider

$$\begin{aligned} r_n(\mathcal{H}) &= \exp\left(\sum_k \frac{1}{k}(x_1^k + x_2^k + \ldots + x_n^k)\right) \\ &= \prod_{i=1}^{n} \exp(\sum_k \frac{1}{k} x_i^k) \\ &= \prod_{i=1}^{n} \exp(\log \frac{1}{1-x_i}) \\ &= \prod_{i=1}^{n} \frac{1}{1-x_i}, \end{aligned}$$

which is a finite product, and hence D-finite. ✪

This proof generalizes to prove an analogue of Proposition 2.2.

Proposition 2.5. *The plethysm $\mathcal{H}[g]$ of $\mathcal{H} = \sum_n h_n$ with a polynomial in the p_i, g, such that $g(0) = 0$, is D-finite as a function of the x_i variables.*

Proof. The idea, as in the previous example, is to simplify the arguments of the exponential to a logarithm which then simplifies the expression to a finite product of rational functions, which is clearly D-finite.

Write g in the monomial basis as the *finite sum* $\sum_\lambda c_\lambda m_\lambda$. Recall that

$$p_k[g] = \sum_\lambda c_\lambda m_\lambda(x_1^k, x_2^k, \ldots),$$

and that $r_n(m_\lambda) = 0$ when $l(\lambda) > n$.

Using this, we expand $\mathcal{H}[g]$:

$$\begin{aligned} r_n(\mathcal{H}[g]) &= \exp\left(\sum_k \frac{1}{k} r_n(p_k[g])\right) \\ &= \exp\left(\sum_k \frac{1}{k} \sum_\lambda c_\lambda x_1^{k\lambda_1} x_2^{k\lambda_2} \cdots x_n^{k\lambda_n})\right) \\ &= \exp\left(\sum_\lambda c_\lambda \sum_k \frac{1}{k} x_1^{k\lambda_1} x_2^{k\lambda_2} \cdots x_n^{k\lambda_n}\right) \\ &= \prod_\lambda \exp\left(c_\lambda \log(1 - (\mathbf{x}^\lambda))^{-1}\right) \\ &= \prod_\lambda \frac{1}{(1-(\mathbf{x}^\lambda))^{c_\lambda}}, \end{aligned}$$

which is a finite product of rational functions, and is consequently D-finite. The D-finiteness $\mathcal{E}[g]$ is shown similarly. ✪

2.5.3 The Kronecker and scalar products

The specialization ex from the last section is a specific case of a larger closure result. Theorem 1.5(4) has the following very important consequence.

Proposition 2.6 (Kronecker product and D-finiteness; Gessel). *Let f and g be D-finite symmetric series in $\mathbb{K}[\![p]\!]$. Then the Kronecker product $f * g$ is D-finite.*

Proof. Suppose that $f = \sum_\lambda c_\lambda p_\lambda$, and $g = \sum_\lambda a_\lambda p_\lambda$ for $c_\lambda, a_\lambda \in \mathbb{K}$. Then the Kronecker product can be written as a Hadamard product:

$$f * g = \left(\sum_\lambda c_\lambda p_\lambda \times \sum_\lambda a_\lambda p_\lambda \right) \times \sum z_\lambda p_\lambda.$$

Note that

$$\sum_{\lambda \in \mathbb{P}} z_\lambda p_\lambda = \lim_{N \to \infty} \prod_{n=1}^{N} A(np_n) \qquad \text{with} \qquad A(x) = \sum_{n=0}^{\infty} n! x^n$$

is clearly D-finite when all but a finite number of the p_n are set to 0. A double application of the closure of D-finiteness under Hadamard product implies that $f * g$ is D-finite. ✪

The following result is a direct consequence of this.

Theorem 2.7 (The scalar product and D-finiteness; Gessel). *Let f and g in Λ_t be D-finite symmetric series, and suppose that g involves only finitely many of the p_i's. Then $\langle f, g \rangle$ is D-finite with respect to the t_i variables provided it is well defined as a power series.*

Proof. By Theorem 2.6, $f * g$ is D-finite. In the case when g contains only a finite number of p_i, say $p_1, \ldots, p_n$, the algebraic substitution $p_i = 1, 1 \leq i \leq n$ is finite, and thus by Theorem 2.1, if $f * g\big|_{p_i=1} = \langle f, g \rangle$ is well defined as a power series then it is D-finite. ✪

For any finite set of integers $\mathcal{S}$, this theorem gives the D-finiteness of the scalar product $\left\langle f, \left(1 - t\sum_{n\in\mathcal{S}} h_n\right)^{-1}\right\rangle$, which can also be described in terms of coefficient extraction as in the next corollary.

Corollary 2.8 (Gessel). *Let f be a D-finite symmetric function and let $\mathcal{S}$ be a finite set of integers. Define $S_n \in \mathbb{Z}$ as follows: S_n is the sum over all n-tuples $(s_1, \ldots, s_n) \in \mathcal{S}^n$ of the coefficient $x_1^{s_1} x_2^{s_2} \cdots x_n^{s_n}$ in f. Then $s(t) = \sum_n S_n t^n$ is D-finite.*

One of the principal contributions of this work is a collection of algorithms, presented in Part 2, which makes Theorem 2.7 effective. We remark that the condition of using only a finite number of p_i variables can not be omitted, since given a sequence c_n which is not P-recursive, we can construct

$$\left\langle \sum p_n c_n/n, \sum_n p_n t^n \right\rangle = \sum_n c_n t^n,$$

which is not D-finite, yet it is the scalar product of two D-finite symmetric series.

On the other hand, it is also not a necessary condition, since

$$\langle \mathcal{H}, \mathcal{H}(t)\rangle = \frac{1}{1-t},$$

is D-finite and $\mathcal{H}$ uses an infinite number of p_i.

There are other scalar products for symmetric functions that are particularly relevant to the development of important symmetric functions, such as Macdonald polynomials. They are treated in Chapter 6.

2.6 A collection of D-finite symmetric series

For future reference we describe a collection of symmetric series. Gessel [34] defines $c_k = \sum_{n=0}^{\infty} h_n h_{n+k}$; This is D-finite as it is the Hadamard product of two D-finite symmetric series, namely $\sum_{n=0}^{\infty} h_n t^n$ and $\sum_{n=0}^{\infty} h_n t^{n+k}$, evaluated at $t = 1$. This can be used to establish the D-finiteness of $\sum_n s_{n,n}$, since $s_{n,n} = h_n^2 - h_{n-1}h_{n+1}$.

Next, we also use some classic results in symmetric series to deduce that $\sum_{\lambda\in\mathbb{P}} s_\lambda$ and $\sum_\lambda s_{\lambda/\mu}$ (for fixed μ) are D-finite. Since $\sum_{\lambda\in\mathbb{P}} s_\lambda = \mathcal{H}[e_1 + e_2]$ and $\sum_\lambda s_{\lambda/\mu} = h[e_1 + e_2]\sum_\mu s_{\nu/\mu}$, the D-finiteness follows from the earlier discussion on plethysm. These last two examples, under the restriction that the sums are limited to partitions

with at most k parts are also D-finite, but this requires a more detailed argument using a determinantal formula.

2.7 Generalizing symmetric functions

Generalizing symmetric functions to accommodate multiple variable sets presents several options. The most straightforward of which uses disjoint variable sets, (in the simplest case say $x = x_1, x_2, \ldots$ and $y = y_1, y_2, \ldots$), and functions independently symmetric in the $x_i's$ and the $y_i's$. In this case, a symmetric function can be written in the form

$$\sum_{\lambda,\mu} c_{\lambda\mu} p_\lambda(x) p_\lambda(y),$$

where λ and μ are partitions and $p_\lambda(x) = p_\lambda(x_1, x_2, \ldots)$ and $p_\mu(y) = p_\mu(y_1, y_2, \ldots)$. A function is D-finite if it is D-finite with respect to the $p_n(x)$ and the $p_n(y)$. This case introduces new variables but still largely resembles the case of the (infinite) multivariate. However Gessel gives some interesting applications to permutations with longest increasing subsequence of a fixed, given length in [34, §7]. For example, he uses that the scalar product can be defined in this case by the relation

$$\langle f(x,y), g(x,y)\rangle_{x,y} = \langle f(x), g(x)\rangle_x \langle f(y), g(y)\rangle_y .$$

On the other hand, one can consider a slightly modified definition and solve a larger collection of interesting problems, including Latin rectangles.

2.7.1 MacMahon symmetric functions

A second generalization considers functions of the following flavor:

$$L(x,y) = \prod_{i \neq j} (1 + x_i y_j), \tag{2.12}$$

where the product is over all pairs of distinct positive integers. MacMahon introduces in [53] a family of functions possessing the key property of this nearly symmetric example. Here we give the definition for just two sets of variables for the sake of simplifying notation, but the general definition is straightforward.

Definition 2.2 ***MacMahon symmetric function.*** A function

$$f(x_1, x_2, \ldots; y_1, y_2, \ldots) \in \mathbb{K}[\![x, y]\!]$$

is a *MacMahon symmetric function* if the coefficient of $x_1^{\alpha_1}x_2^{\alpha_2}\cdots y_1^{\beta_1}y_2^{\beta_2}\cdots$ is equal to the coefficient of $x_{i_1}^{\alpha_1}x_{i_2}^{\alpha_2}\cdots y_{i_1}^{\beta_1}y_{i_2}^{\beta_2}\cdots$ for any finite set of *distinct* integers $\{i_1, i_2, \ldots\}$.

MacMahon used these functions and a suitably generalized notion of Hammond operators (defined here in Section 2.2.2) to determine some enumerative formulas for Latin rectangles and other related combinatorial objects. Unfortunately, his presentation lacked a requisite elegance (due in part to the youth of linear algebra at the time) for this method to become popular. Fortunately Gessel recognized [33] how this could be reformulated and fit into a theory of D-finiteness of MacMahon symmetric functions. A different direction is taken by Doubilet [25], and subsequently Rosas [65] with combinatorial interpretations of the functions, and some of their operators. In particular, Rosas shows in [66] that they are the generating functions for the orbits of sets of functions indexed by partitions under the diagonal action of the Young subgroup of a symmetric group. This gives a description of the change of bases matrix between the different bases. She describes in [67] the action of the principal specialization on several of the bases.

Remarkably, the algorithms developed in Part 2 for the usual symmetric functions will also work for these functions with only a slight modification. Using these algorithmic results we can revisit some of the original examples of MacMahon as part of our unified approach to scalar products and symmetric functions.

There is a rather complete parallel theory developed for MacMahon symmetric functions. Indeed they generalize well in a natural way to a class of *non-commutative* symmetric functions. However, here we discuss only the basic notions and properties. Complete definitions and developments are provided in [66].

Define a *bipartite number* (a, b) as a pair from $\mathbb{N}^2 \setminus \{(0, 0)\}$.

Definition 2.3 ***Bipartite partition.*** A *bipartite partition* of $(a, b) \in \mathbb{N}^2 \setminus \{(0, 0)\}$ is a set of bipartite numbers $\pi = \{(a_i, b_i)\}$ whose pointwise sum is (a, b). That is, $\sum_i a_i = a$ and $\sum_j b_j = b$. These are generally written as the unordered list $(a_1, b_1)(a_2, b_2)\cdots$.

For example, $\{(1, 1)(1, 0)(1, 0)\}$ is a bipartite partition of $(3, 1)$. This can also be written $\{(1, 1)(1, 0)(1, 0)\} \vdash (3, 1)$.

Bipartite numbers and partitions generalize in the obvious way to $k-$ary partitions of

integer vectors of length k (k-ary numbers).

We can define analogues to most of the common bases using these partitions. The *monomial MacMahon symmetric function* associated to a bipartite partition π is the sum over all distinct monomials of the form:

$$x_{i_1}^{a_1} y_{i_1}^{b_1} x_{i_2}^{a_2} y_{i_2}^{b_2} \cdots .$$

For example, $m_{(2,3)(1,1)} = \sum_{i \neq j} x_i^2 y_i^3 x_j y_j$ and $m_{(1,1)(1,1)} = \sum_{i \leq j} x_i y_i x_j y_j$. The *power symmetric functions* are defined on bipartite numbers as

$$p_{(a,b)} = \sum_i x_i^a y_i^b = m_{(a,b)},$$

and extended to a bipartite partition $\pi = \{(a_i, b_i)\}$ multiplicatively:

$$p_\pi = \prod_i p_{(a_i,b_i)}.$$

We define e_π *the elementary Macmahon functions* and h_π, *the complete homogeneous MacMahon functions* using the following products:

$$1 + \sum_{a,b \in \mathbb{N}^2} e_{(a,b)} s^a t^b = \prod_i (1 + x_i s + y_i t)$$

so that $e_{(a,b)} = m_{(1,0)^a (0,1)^b}$ and

$$1 + \sum_{a,b \in \mathbb{N}^2} h_{(a,b)} s^a t^b = \prod_i \frac{1}{1 - x_i s - y_i t}.$$

These bases are also extend multiplicatively to bipartite partitions. Further, for any of these four types, we can describe a basis for the vector space of all MacMahon symmetric functions with total x-degree a and total y-degree b by indexing over all bipartite partitions of (a, b). This is the vector space of *bi-homogeneous degree* (a, b).

2.7.2 The scalar product

The scalar product is defined in a manner analogous to the usual symmetric functions. The formulas appear to be almost identical to Eq. (2.4) and Eq. (2.5):

$$\langle h_\pi, m_\mu \rangle = \delta_{\pi,\mu}.$$

MacMahon proves symmetry of this operation. Using this, one can likewise deduce that

$$\langle p_\pi, p_\mu \rangle = \delta_{\pi,\mu} z_\pi, \tag{2.13}$$

where

$$z_\pi = \prod_{(a_i,b_j)} \mathbf{m}_{(a_i,b_j)}(\pi) \left(\frac{a_i! b_j!}{(a_i + b_j - 1)!} \right)^{\mathbf{m}_{a_i,b_j}(\pi)}$$

Here π is the bipartite partition $\{(a_i, b_j)\}$, and $\mathbf{m}_{(a_i,b_j)}(\pi)$ is the multiplicity of (a_i, b_j) in π.

Remark that when $\pi \vdash (a, 0)$, (that is, $b_j = 0$ for all j) this reduces to the usual symmetric function scalar product and z_π.

Definition 2.4 ***D-finite MacMahon symmetric function.*** A MacMahon symmetric function is *D-finite* if it is D-finite with respect to the $p_{(a,b)}$.

For example, one can show that $L(x, y)$ of Eq. (2.12) is also equal to

$$L(x, y) = \exp\left(\sum_{j=1}^{\infty} \frac{(-1)^{j-1}}{j} \left(p_{(j,0)} p_{(0,j)} - p_{(j,j)} \right) \right).$$

An analogue to Theorem 2.7 is also true.

Theorem 2.9 (MacMahon scalar product and D-finiteness). *If $f = \sum f_\pi p_\pi$ and $g = \sum g_\pi p_\pi$ are two D-finite MacMahon symmetric functions such that g uses only a finite number of p_π, then $\langle f, g \rangle$ is D-finite.*

Proof. This can be proved in a fashion analogous to the usual symmetric functions. First remark that

$$\sum_\pi z_\pi p_\pi = \prod_{(a,b)\in\mathbb{N}^2} A\left(\frac{a!b!}{(a+b-1)!} p_{(a,b)} \right), \quad \text{where} \quad A(x) = \sum_{n=0}^{\infty} n! x^n.$$

By Theorem 1.5(4), we have that the Hadamard product

$$\left(\sum_\pi f_\pi p_\pi \times \sum_\pi g_\pi p_\pi \right) \times \sum_\pi z_\pi p_\pi,$$

is D-finite, and so is the specialization $p_{(a,b)} = 1$. Together this implies that the scalar product of D-finite MacMahon symmetric functions is D-finite. ✪

In Part 2. we make this effective, and in Part 3 we give some combinatorial applications of this theory, notably the enumeration of Latin rectangles.

There are several ring morphisms from MacMahon symmetric functions that should preserve D-finiteness, such as specializations to one set of variables, as well as other diagonal-like operators.

Chapter III

An introduction to holonomy

3.1 Algebraic properties of differential operators

Once it is known that an operation preserves D-finiteness, it is very natural to ask how to determine the differential system satisfied by the resulting function.

The algebra in which our manipulations occur is generated by two types of elements: One is the differential operator ∂_x, which is differentiating a function with respect to x; the second is x, a multiplication operator which is multiplication on the left by x. Each time we wish to indicate an operator we use $\cdot$. That is, $x \cdot f(x) = xf(x)$ and $\partial_x \cdot f(x) = \frac{df(x)}{dx}$. Under this view, a differential equation, say $f''(x)+x^2f'(x)+f(x) = 0$, corresponds to the polynomial operator $\partial_x^2 + x^2\partial_x + 1 \cdot f(x) = 0$. Thus, any differential expression is viewed as a non-commuting expression in ∂_x and x.

In fact, the essential aspects of the differential operator approach are reduced to defining a suitable commutation relation between multiplying by x and differentiating with respect to x. We can then work within the algebra of such differential operator expressions, and manipulate differential equations. This algebra is known as the *Weyl algebra*, and is detailed in Section 3.2.

If a function $f(x)$ satisfies a linear differential equation with polynomial coefficients, it will satisfy many differential equations of this form (for example, differentiating the differential equations yields new differential equations of higher order). In fact, the set of these differential equations form a left ideal in the Weyl algebra. The theory of holonomy links properties of these ideals of operators with the property of being D-finite. This gives us an alternate characterization of D-finite.

The study of holonomic systems was initiated by Bernstein [8] in order to answer a question of Gel'fand about whether a certain function of a complex variable could be extended to a meromorphic function defined in the whole complex plane. The study of these systems has since branched into several different directions. For a complete picture of developments related to the algebraic study of differential equations, consult the introduction of [20]. Here, we limit our discussion to aspects of holonomic systems pertaining to effective computation of D-finite closure properties.

Galligo made a key contribution to this topic in [29] with the assertion that a non-commutative version of Buchberger's algorithm applied to the Weyl algebra yields Gröbner bases of left ideals. This result is an important element of Takayama's work on the effective integration of holonomic functions [80]. Zeilberger illustrated how this effective integration can be utilized to find differential equations satisfied by certain special functions, and how to verify certain families of identities automatically, using computer algebra [88]. These studies were also propelled by a renaissance of interest in hypergeometric functions, which began around the time of Apéry's proof of the irrationality of $\zeta(3)$ [82]. The problem of integration of a holonomic function is of interest for the purpose of this thesis since it shares many algorithmic properties with the computation of the scalar product of symmetric functions, the central problem of this work. The algorithms we describe here therefore bear resemblance to some known algorithms for integration.

Indeed, the 1990s marked a very active period for the development of computer algebra tools for the treatment of holonomic systems. A small selection of available packages for the major computer algebra systems includes: packages for general manipulation of holonomic systems such as `KAN` [79], `gfun` and `Holonomy` [16, 70] for Maple, and `D-module` for Macaulay II [47]; packages treating the hypergeometric case for summation and integration such as the `Ekhad` packages [86] for Maple, and the work of the RISC group for a Mathematica version [61].

One parallel development is a treatment for a general class of linear operators, called the Ore operators. We consider this in the next chapter.

A useful tool for computation is Gröbner bases, modified to suit a non-commutative setting. We provide a small summary of some vocabulary and basic results in the final section of this chapter.

The goal of this brief introduction to holonomic systems is to provide sufficient background to ensure that the motivation, and correctness of the algorithms introduced in the next part is clear.

3.2 The Weyl algebra of differential operators

As we have already outlined, differential equations can be efficiently manipulated in a suitable algebra. We call the algebra generated by differential operators, and multiplication by variables, the Weyl algebra. The book of Coutinho [20] is an excellent introduction to this topic, and the theorems referenced in this chapter indexed with the letter C refer to theorems from this book. (Ex. Thm. C.2.1)

Definition 3.1 ***The Weyl Algebra*** $\mathcal{A}_n$**.** The *Weyl algebra* $\mathcal{A}_n$ of dimension n is the associative $\mathbb{K}$-algebra

$$\mathcal{A}_n = \mathbb{K}\big\langle x_1, \ldots, x_n, \partial_{x_1}, \ldots, \partial_{x_n}; [\partial_{x_i}, x_j] = \delta_{i,j},\ 1 \leq i, j \leq n \big\rangle,$$

where the bracket $[a, b]$ denotes $ab - ba$ and $\delta_{i,j}$ is the Kronecker notation. This algebra can be identified with the algebra of linear differential operators with coefficients that are polynomial in $x = x_1, \ldots, x_n$. Related to this is $\mathcal{A}_n(x)$, the algebra of linear differential operators with coefficients in $\mathbb{K}(x)$.

The algebra $\mathcal{A}_n$ has a natural action on the ring of formal power series $\mathbb{K}[\![x_1, \ldots, x_n]\!]$:

$$\partial_i \cdot f = \frac{\partial f}{\partial x_i}, \quad x_i \cdot f = x_i f. \tag{3.1}$$

If there exists a polynomial $P \in \mathcal{A}_n$ such that $f \in \mathbb{K}[\![x_1, \ldots, x_n]\!]$ satisfies the differential equation $P \cdot f = 0$, then $QP \cdot f = Q \cdot (P \cdot f) = Q \cdot 0 = 0$ for any $Q \in \mathcal{A}_n$.In fact, if there is a system of $P_i \in \mathcal{A}_n$, $1 \leq i \leq k$ that satisfy $P_i \cdot f = 0$, then f also satisfies $(\sum_{i=1}^{k} Q_i P_i) \cdot f = 0$. Consequently, the set of elements of $\mathcal{A}_n$ which annihilate f in this manner form a left ideal. This ideal is denoted by $\mathcal{I}_f$ and is called the *annihilating ideal* of f. Note that this is not a two sided ideal, and in fact $\mathcal{A}_n$ contains only trivial two sided ideals (Thm. C.3.1).

Annihilating ideals of D-finite functions warrant a special label.

Definition 3.2 ***D-finite ideal.*** A left ideal $\mathcal{I}$ of $\mathcal{A}_n$ is D-finite if $\mathcal{A}_n/\mathcal{I}$ is finite dimensional over $\mathbb{K}$(x).

The name comes from the following observation. When $\mathcal{I}$ is the annihilating ideal of a function f, then the quotient $\mathcal{A}_n/\mathcal{I}$ is isomorphic to the $\mathcal{A}_n$-module $\mathcal{A}_n f$. This module is generated by partial derivatives of f, and thus is finite dimensional if and only if f is D-finite. This bears repeating. An annihilating ideal $\mathcal{I}_f$ is a D-finite ideal if and only if f is D-finite function.

A very important feature of this algebra is that each element can be written uniquely as a polynomial in non-commuting variables:

$$F = \sum_{(\alpha,\beta)\in\mathbb{N}^{2n}} c_{\alpha\beta}\, x^\alpha \partial^\beta,$$

where $x^\alpha = x_1^{\alpha_1}\cdots x_n^{\alpha_n}$, and $\partial^\beta = \partial_1^{\beta_1}\cdots\partial_n^{\beta_n}$. This form is obtained by repeated application of the relation

$$\partial_i x_i = x_i \partial_i + 1.$$

The fact that there is such a standard form is extremely useful. Essentially, it implies that there is a natural vector space isomorphism between the Weyl algebra and the polynomial algebra $\mathbb{K}[x_1,\ldots,x_n,y_1,\ldots y_n]$, specifically the map sending $x^\alpha\partial^\beta$ to $x^\alpha y^\beta$. This form greatly simplifies manipulation within the algebra and renders the computations amenable to computer algebra treatment.

3.2.1 Gradations and filtrations

Another important feature of $\mathcal{A}_n$-modules is the existence of a natural algebra filtration and an associated grading. The definition of holonomy presented here is phrased in terms of gradations, as are many of the related algorithms.

Definition 3.3 ***graded ring.*** A $\mathbb{K}$-algebra $\mathcal{R}$ is *graded* if there are $\mathbb{K}$-vector subspaces $G_i, i\in\mathbb{N}$ such that

$$G_i \cdot G_j \subseteq G_{i+j} \quad \text{and} \quad \mathcal{R} = \bigoplus_{i\in\mathbb{N}} G_i.$$

Each G_i is called a *homogeneous component*, of *degree* i.

> *Example.* The ring of polynomials $\mathbb{K}[x_1,\ldots,x_n]$, graded with degree function $\deg(x^\alpha) = |\alpha| = \alpha_1 + \ldots + \alpha_n$. In this ring, a homogeneous component G_i is the vector space generated by monomials of degree i.

The Weyl algebra resembles the polynomial ring, however it does not admit a grading under the usual polynomial degree. The element $\partial_1 x_1$ appears to be homogeneous of degree 2, yet it is also equal to $x_1\partial_1 + 1$, which is not homogeneous. Instead, we consider a filtration of this ring.

Definition 3.4 ***filtered algebra.*** Let $\mathcal{R}$ be a $\mathbb{K}$-algebra. A family $\mathcal{F} = (V_i)$ of increasing $\mathbb{K}$-vector spaces $\mathbb{K} = V_0 \subseteq V_1 \subseteq \cdots \subseteq \mathcal{R}$ is a *filtration* of $\mathcal{R}$ if

$$V_i \cdot V_j \subseteq V_{i+j} \quad \text{and} \quad \mathcal{R} = \bigcup_{i\geq 0} V_i.$$

An algebra admitting such a filtration is a *filtered algebra*.

One example of a filtration of $\mathcal{A}_n$, the *Bernstein filtration*, $(\mathcal{B}_k)$, filters according to the maximal degree of an element in standard form. This filtration has the useful feature that each $\mathcal{B}_k$ is a vector space of finite dimension, with basis $x^\alpha\partial^\beta$, satisfying $|\alpha| + |\beta| \leq k$.

Given a filtration $\mathcal{F}$ for a ring $\mathcal{R}$, we construct an associated graded ring, $gr^{\mathcal{F}}\mathcal{R}$, called the *graded algebra of R associated with the filtration $\mathcal{F}$,*

$$gr^{\mathcal{F}}\mathcal{R} = \bigoplus_{i\geq 0} (V_i/V_{i-1})\,. \tag{3.2}$$

The multiplication in this algebra can be defined using the canonical projection of vector spaces, $\sigma_k : V_k \to V_k/V_{k-1}$ where $\sigma_k(a)$ is non-zero only when $a \notin V_{k-1}$. Thus, the multiplication is given by $\sigma_m(a)\sigma_n(b) := \sigma_{n+m}(ab)$ for $\sigma_m(a)\sigma \in gr^{\mathcal{F}}\mathcal{R}$. It is straightforward to verify that this multiplication is compatible with the algebra and the obvious filtration.

Surprisingly, the graded algebra of $\mathcal{A}_n$ associated with the Bernstein filtration is isomorphic to the polynomial ring over $\mathbb{K}$ in $2n$ variables (Theorem C.3.1).

Next, we provide analogous definitions for modules of filtered rings. Suppose $\mathcal{R}$ is a $\mathbb{K}$-algebra furnished with either a gradation (G_i) or a filtration (V_i), and that M is a left $\mathcal{R}$-module.

A *module gradation of M* is a family (Γ_i) of vector subspaces of M such that

$$G_i \cdot \Gamma_j \subseteq \Gamma_{i+j} \quad \text{and} \quad \bigoplus_{i\geq 0} \Gamma_i = M.$$

A *filtration of* M is an increasing family of subspaces (Φ_i) of M satisfying

$$f_i \cdot \Phi_j \subseteq \Phi_{i+j} \quad \text{and} \quad \bigcup_{i \geq 0} \Phi_j = M.$$

As in the case of an algebra, to each module admitting a filtration, we associate a graded module.

Definition 3.5 ***associated graded module.*** Suppose $\mathcal{R}$ is a $\mathbb{K}$-algebra with a filtration $\mathcal{F}$, and M a filtered left $\mathcal{R}$-module with filtration Φ. The $gr^{\mathcal{F}}\mathcal{R}$-left module $gr^{\Phi}M$,

$$gr^{\Phi}M = \bigoplus_n \left(\Phi_{i+1}/\Phi_i\right).$$

is the *associated graded module to* M.

Again we use projections to complete the definition, and here one is used to define the module action. Define μ to be the canonical projection $\mu_j : \Phi \longrightarrow \Phi_k/\Phi_{k-1}$. Then, using the same projection σ_m as defined earlier, we define the action $\sigma_k(a)\cdot\mu_i(u) = \mu_{i+k}(a\cdot u)$, and extend linearly. This defines a module action on $gr^{\Phi}M$.

A filtration of a module is said to be *good* if Φ_n is a finite dimensional vector space for all n.

It might be desirable to index a filtration by a totally ordered monoid different from that of the integers. We could, for example, refine a polynomial algebra filtration by degree, into a filtration of degree sequence of the leading monomial. The above definitions can be suitably modified to handle this more general situation.

3.2.2 Hilbert polynomial

Here, our examples use only good filtrations. In this case, there is an interesting polynomial associated to the filtration.

Theorem 3.1 (The Hilbert polynomial; Hilbert). *Suppose M is a finitely generated left $\mathbb{K}[u_1, \ldots, u_r]$-module with grading $\Gamma = (\Gamma_i)$. Then there exists rational numbers $c_0, \ldots, c_d$ for $d \leq r$ with $c_d \neq 0$ such that*

$$H_M(n) := \sum_{i \leq n} \dim_{\mathbb{K}} \Gamma_i = \frac{c_d}{d!} n^d + \cdots + c_0$$

for sufficiently large n. The polynomial $H_M(n)$is called the Hilbert polynomial *of M.*

The degree of this polynomial is an important invariant of the module called the *Hilbert dimension* of the module. The leading coefficient c_d is equally important, it is the *multiplicity* of the module.

A useful result for our purposes is the case of the associated graded module of a good filtration Φ of a $\mathcal{A}_n$-module M. The Hilbert polynomial is equal to

$$H_M(n) = \dim_{\mathbb{K}} \Phi_n = \dim_{\mathbb{K}} \mathcal{B}_n / \left(M \cap \mathcal{B}_n\right). \tag{3.3}$$

3.3 Holonomic modules

Bernstein's inequality states that the Hilbert dimension of a non-trivial $\mathcal{A}_n$-module is greater than or equal to n [9, Thm. 1.3]. The finitely generated modules with this dimension exactly form a special class. First we give some basic definitions, and then we consider the intuition behind the definitions.

Definition 3.6 ***holonomic module.*** A finitely generated left $\mathcal{A}_n$-module is *holonomic* if it is either trivial, or if it has Hilbert dimension n.

Example. Perhaps the simplest example of a non-trivial holonomic $\mathcal{A}_d$-module is $M = \mathbb{K}[x_1, \ldots, x_d]$, under the usual action. Filtered by total degree into $(M_n)_n$ which is equivalent to $M_n = (M \cap \mathcal{B}_n)$ where $(\mathcal{B}_n)$ is the Bernstein filtration of $\mathcal{A}_d$, we have by Eq. (3.3) that

$$\begin{aligned} H_M(n) &= \dim_{\mathbb{K}} \left(\mathcal{B}_n / \left(M \cap \mathcal{B}_n\right)\right) = \dim_{\mathbb{K}} \left(\mathcal{B}_n / M_n\right) \\ &= \dim_{\mathbb{K}} \mathcal{B}_n \cap \mathbb{K}[\![\partial_1, \ldots, \partial_d]\!] \\ &= \binom{n+d}{n} = n^d + \text{ lower order terms.} \end{aligned}$$

Thus, the Hilbert dimension is d, and the module is holonomic.

We shall say an ideal is a *holonomic ideal* if, when viewed as a $\mathcal{A}_n$-module, it is a holonomic module.

If $f \in \mathbb{K}[\![x]\!]$ is contained in a holonomic module, then it is a *holonomic function*. This definition is motivated by the fact that f is holonomic when its annihilating ideal, $\mathcal{I}_f$, is a holonomic ideal. That is, $\mathcal{A}_n f \simeq \mathcal{A}_n / \mathcal{I}_f$ is a holonomic $\mathcal{A}_n$-module. Thus, f is holonomic when $\mathcal{A}_n / \mathcal{I}_f$ is a holonomic module.

Example. In the univariate case, a function $f(x)$ is holonomic if

$$\dim_{\mathbb{K}} \mathcal{B}_n \cdot f(x) = \dim_{\mathbb{K}} \bigoplus_{j+k\leq n} \left(\mathbb{K} \otimes x^j f^{(k)}\right) = O(n).$$

Consider the function $\sin(x)$. Observe that

$$\bigoplus_{j+k\leq n} \mathbb{K} \otimes x^j \sin^{(k)}(x) = (\mathbb{K}[x])_n \sin(x) \oplus (\mathbb{K}[x])_n \cos(x)$$

with $(\mathbb{K}[x])_k$ the set of polynomials of degree at most k. The dimension of this space over $\mathbb{K}$ is $2n$, which is of order n, and thus $\sin(x)$ is holonomic.

More generally, the intuition is as follows. The Bernstein filtration $(\mathcal{B}_k)$ of $\mathcal{A}_d$ is composed of a sequence of finite dimensional vector spaces each of dimension $\binom{d+2n}{2n} = O(n^{2d})$, which corresponds to Hilbert dimension $2d$. For any $\mathcal{A}_d$-module M, generated by $y_i \in \mathbb{K}[\![x_1, \ldots, x_d]\!]$ such that $M = \bigoplus_i \mathcal{A}_d y_i$, we have that M is a holonomic module if, for n large, the space $\bigoplus_i \mathcal{B}_n y_i$ has dimension $O(n^d)$.

3.3.1 D-finite functions are holonomic

Holonomic systems are of interest to us because D-finite functions are also holonomic. Recall that $f \in \mathbb{K}[\![x]\!]$ is D-finite if and only if its annihilating ideal is D-finite.

Proposition 3.2 (Holonomy and D-finiteness; Bernstein, Takayama). *A function f is holonomic if and only if it is D-finite.*

The converse direction is the more difficult [81], however, the general idea of this should be clear from the above discussion. This is the characterization of D-finite functions that allow us to determine numerous effective closure properties. To develop this topic in depth, we first develop a second theoretical tool, Gröbner bases for the Weyl algebra.

Very often we are interested in $\mathcal{A}_n(x) \simeq \mathbb{K}(x) \otimes \mathcal{A}_n$. Holonomy is not impeded by this extension.

Theorem 3.3 (Holonomy and $\mathbb{K}(x)$; Bernstein, Kashiwara). *Suppose $\mathcal{I}$ is a left $\mathcal{A}_n$-ideal. Then $\mathcal{A}_n/\mathcal{I}$ is a holonomic module if and only if $\mathbb{K}(x) \otimes \mathcal{I}$ is a D-finite ideal of $\mathbb{K}(x) \otimes \mathcal{A}_n$. Otherwise stated, any ideal $\mathcal{J} \subseteq \mathcal{A}_n(x)$ is D-finite if and only if the module $\mathcal{A}_n/\mathcal{J} \cap \mathcal{A}$ is D-finite.*

3.3.2 Some closure properties

We now list some basic properties of holonomic modules that will be useful in the next part. These results are found in [20, 10, Ch. V], for example.

Submodules and quotients of holonomic modules are holonomic. The Weyl algebra is Noetherian, that is every decreasing chain of ideals is finite. Holonomic $\mathcal{A}_n$-modules are cyclic, that is, generated by a single element.

In general, *twisting* a left $\mathcal{R}$-module M by an $\mathcal{R}$-automorphism σ results in a new $\mathcal{R}$-module M^σ. The underlying space is the same, but we define a different action for elements of $\mathcal{R}$. Suppose that the action of $r \in \mathcal{R}$ on $m \in M$ is given by $r \cdot m$. For any automorphism σ of $\mathcal{R}$, the *twisted module* of M by σ, M^σ, is equal to M as a group, however, the action of r on m in M^σ is defined as $\sigma(r) \cdot m$. We will denote this twisted action in the following way:

$$r \cdot_\sigma m := \sigma(r) \cdot m. \tag{3.4}$$

Sometimes, for notational convenience, we use a twisting action to denote a *right* module action. Thus, we may denote the right action $m \,.\, r$ with $r \cdot_\sigma m$.

$r \cdot_\sigma m = m.r$. Throughout we denote a right action by ".", (as in $x.a$, for a acting on x to the right).

One particular classic twist action closely resembles the scalar product adjoint.

Definition 3.7 ***Fourier transform of a module.*** The $\mathcal{A}_n$-automorphism $\mathcal{F}$ defined by

$$\mathcal{F}(x_i) = \partial_i \text{ and } \mathcal{F}(\partial_i) = -x_i$$

for $1 \leq i \leq n$ is the called *Fourier transform* of an $\mathcal{A}_n$-module.

The name comes from the fact that it sends linear differential operators with complex coefficients to polynomials.

This automorphism preserves the Bernstein filtration. In fact, for any finitely generated left $\mathcal{A}_n$-module M, M and $M^\mathcal{F}$ have the same multiplicity and Hilbert dimension. This implies the following useful result.

Proposition 3.4 (Fourier transform of holonomic modules). *Holonomic $\mathcal{A}_n$-modules are closed under Fourier transform. That is, if M is a holonomic module, so*

is $M^{\mathcal{F}}$.

In general, any twisting that preserves the Bernstein filtration will also preserve holonomy by Eq. (3.3).

3.3.3 Products of modules

In order to consider the tensor product of Weyl algebras, we must, for clarity's sake, develop notation to handle multiple variable sets. Denote by $\mathcal{A}_x$ the Weyl algebra over indeterminates $x = x_1, \ldots x_n$ and by $\mathcal{A}_y$ the Weyl algebra over indeterminates $y = y_1, \ldots, y_m$. The Weyl algebra over the union of these variable sets shall be denoted $\mathcal{A}_{x,y}$.

Now, suppose M is an $\mathcal{A}_x$-module and that N is a $\mathcal{A}_y$-module. Recall that the tensor product of M and N, is a $\mathbb{K}$-vector space, linearly generated by the set of $m \otimes n$ where m ranges over all generators of M, and n ranges over all generators of N, and where scalar multiplication satisfies $k(a \otimes b) = (ka) \otimes b = a \otimes (kb)$.

We define an $\mathcal{A}_{x,y}$ action on $M \otimes N$ as follows. First, we write $\mathcal{A}_{x,y}$ as the external product $\mathcal{A}_x \otimes \mathcal{A}_y$. Next, for all $(p, q) \in \mathcal{A}_x \otimes \mathcal{A}_y$ and $(u, v) \in M \otimes N$ define

$$(p, q) \cdot (u, v) = (p \cdot u, q \cdot v),$$

and extend bilinearly.

One deduces the following from simple properties of dimension of tensor products.

Proposition 3.5. *Let M be a holonomic $\mathcal{A}_x$ module and N and holonomic $\mathcal{A}_y$ module. Then $M \otimes N$ is a holonomic $\mathcal{A}_{x,y}$-module.*

3.4 Effective properties using Gröbner bases

Gröbner bases serve here as the primary tool for making several closure properties of holonomic functions effective. This section is a short detour to recall many of the basic definitions and classic results for commutative algebras as presented in the excellent reference [21], as well as the extension of this theory to non-commutative cases. The reader already familiar with basic facts about Gröbner bases could easily skip to the final section of this chapter.

3.4.1 Canonical bases for ideals

In order to manipulate ideals, it is convenient to be able to describe a canonical basis. Also, given a set of generators for an ideal in a polynomial algebra $\mathbb{K}[x]$, it is of interest to be able to determine if a polynomial $p(x)$ is contained in this ideal. Both of these problems can be solved using Gröbner bases. The basic idea is to "reduce" elements with respect to an ordering. Basis generators can be reduced to give a "least" or canonical basis. This solves the first problem. If a given element, when "reduced" by the basis element, reduces to 0, then one can deduce that it is a member of the ideal. This reduction can be done with an algorithm known as Buchberger's algorithm and we give some of the fundamental elements in it.

The idea of reduction of a polynomial is rooted in the Euclidean algorithm for integers, which generalizes in a straightforward manner to reduction modulo principal ideals: Given two polynomials $p(x), q(x) \in \mathbb{K}[x]$, $p(x)$ is *reduced modulo* $q(x)$ (and also the ideal generated by $q(x)$) to its remainder (of smaller total degree) $r(x)$ of the polynomial division $p(x)/q(x)$. That is, an element written $p(x) = q(x)t(x) + r(x)$, where $\deg(r(x)) < \deg(p(x))$, reduces to $r(x)$. This shall be denoted $p(x) \xrightarrow{q(x)} r(x)$. This $r(x)$ is called the *normal form* of $p(x)$.

In the more general setting of Noetherian rings, where ideals are generated by a finite number of elements, a similar set-up exists, but requires some work to establish.

First, recall the notion of a *monomial ordering*. All of the multivariate monomial orderings reduce to "degree" in the univariate case. A *monomial order* is a total ordering, $\preceq$, of monomials that satisfies two properties:

- if $a \preceq b$ then $sa \preceq ab$ for any s in the ring;
- and, for any monomial a we have $1 \preceq a$.

Two useful monomial orderings are the *lexicographical ordering*, briefly, Lex , and *total degree*, denoted DegLex. Assuming an ordering $x_1 \preceq x_2 \preceq \dots$ of the variables, Lex compares two monomials $x^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \dots x_k^{\alpha_k}$ and $x^\beta = x_1^{\beta_1} x_2^{\beta_2} \dots x_l^{\beta_l}$ with the rule that $x^\alpha \preceq x^\beta$ if and only if $\alpha_i = \beta_i$ for i from 1 to some $n-1$, and $\alpha_n > \beta_n$. Total degree first compares the total degree of the monomials and then breaks ties with lexicographical ordering. In the applications we consider we shall use a third ordering, the *elimination*

ordering, $\preceq_{\text{Elim}}$. Here the variables are divided into two sets, those to be eliminated, $x = x_1, \ldots, x_n$, and the remaining variables, say $y = y_1, \ldots, y_m$. In this ordering, $x^\alpha y^\beta \preceq_{\text{Elim}} x^{\alpha'} y^{\beta'}$ if $x^\alpha \preceq_{\text{DegLex}} x^{\alpha'}$, with the tie $\alpha = \alpha'$ broken by the comparison $y^\beta \preceq_{\text{DegLex}} y^{\beta'}$.

To illustrate these orderings, here are the smallest terms of $\{x, y, z\}^*$, ordered alphabetically, according to the different term orderings:

Lex	$1 \preceq x \preceq x^2 \preceq x^3 \ldots\ldots \preceq y \preceq xy \preceq x^2y \preceq \ldots \preceq y^2 \preceq \ldots \preceq xyz \preceq x^2yz$
DegLex	$1 \preceq x \preceq y \preceq z \preceq x^2 \preceq xy \preceq xz \preceq y^2 \preceq \ldots$
Elim(z)	$1 \preceq x \preceq y \preceq x^2 \preceq xy \preceq y^2 \preceq \ldots \preceq z \preceq xz \preceq yz \preceq \ldots$

The *leading monomial* of a polynomial is the monomial which is greatest with respect to a particular monomial ordering. The function which produces the leading monomial of a polynomial p is $\text{lm}(p)$. If $x \preceq y \preceq z$, then

$$\text{lm}_{\text{Lex}}(x^3 + z^2 + x^2y^2) = z^2 \quad \text{lm}_{\text{DegLex}}(x^3 + z^2 + x^2y^2) = x^2y^2.$$

If we use the elimination ordering, with elimination set x, then $\text{lm}(x^3 + z^2 + x^2y^2) = x^3$.

The reduction algorithm uses the same basic idea as the univariate case: we apply a division algorithm to determine a remainder which is smaller with respect to the chosen monomial ordering, and proceed recursively with reducing this remainder.

3.4.2 Gröbner bases

The observation which motivates the next definition states that if $p(x)$ can be reduced by $q(x)$, then it must be that one of its terms is a multiple of the leading term of $q(x)$.

Definition 3.8 ***(commutative) Gröbner basis.*** For an ideal $\mathcal{I}$ of the commutative ring $\mathbb{K}[x_1, x_2, \ldots, x_n]$ and any monomial ordering $\preceq$ of $x = x_1, \ldots, x_n$, a *Gröbner basis* is a subset $\mathcal{G}$ of $\mathcal{I}$ with the property that the ideal generated by the set of leading monomials of $\mathcal{G}$ is equal to the ideal generated by the leading monomials of $\mathcal{I}$. Concisely,

$$\langle \text{lm}(p) | p \in \mathcal{G} \rangle = \langle \text{lm}(q) | q \in \mathcal{I} \rangle.$$

The interest in such a generating set of an ideal comes from the following key properties:

- Every non-empty ideal possesses a finite Gröbner basis;

- Every Gröbner basis also generates the complete ideal.

What we previously referred to as Buchberger's algorithm takes as input a generating set of an ideal and does a finite set of comparisons and reductions to construct Gröbner basis of this ideal. The reduction step compares leading terms of polynomials and reduces them using a *syzygy*, that is, a well-chosen combination of the two elements whose leading term is smaller than the initial two elements with respect to the monomial ordering. For more details, the reader is referred to [21].

The non-commutative setting

Thus far we have limited our view to the commutative case. In order to treat Weyl algebras we require a natural extension of this theory. The work of Mora on the cases of free monoids is a thorough investigation. Galligo [29] considered the Weyl algebra case, and Chyzak unified these two approaches to suit the context of Ore algebras of linear operators. In order to prevent too great a diversion in this direction, we restrict our discussion to a reassurance to the reader that we have suitable analogues of all of the elements required from the commutative case, notably, Gröbner bases and Buchberger's algorithm. This is by no means a trivial assertion, and the interested reader is encouraged to consult the development and proofs in [13, 60].

As remarked upon in the discussion of the commutative case, an important property of an algebra is the existence of some version of the division algorithm, which allows one to compute normal forms. In many cases this is accomplished with Buchberger's algorithm using monomial orderings. Essentially, one first constructs a Gröbner basis (canonical with respect to the monomial ordering) and then "reduces" the polynomial using this Gröbner basis.

There are certain filtration properties of the Weyl algebra which make this particular non-commutative version feasible. The existence of a normal form for elements in this algebra, the proximity of this normal form to the (commutative) polynomial algebra and the compatibility between the monomial ordering and the algebra operations are all essential ingredients.

The following example provides some insight into what these computations resemble. It also illustrates a step that forms a part of the algorithms.

Example. The symmetric series given by $G(t, p_1, p_2) = \exp(t/2(p_1^2 + p_2))$ satisfies three differential equations which can be determined by differentiating the function in turn by t, p_1, and p_2:

$$\begin{aligned}\frac{\partial G}{\partial t}(t,p_1,p_2) - (p_1^2/2 - p_2/2)G(t,p_1,p_2) &= 0\\ \frac{\partial G}{\partial p_1}(t,p_1,p_2) - (tp_1^2)G(t,p_1,p_2) &= 0\\ \frac{\partial G}{\partial p_2}(t,p_1,p_2) - (-t/2)G(t,p_1,p_2) &= 0.\end{aligned}$$

Thus, the annihilating ideal $I_G \subset \mathcal{A}_{t,p_1,p_2}$ contains

$$\{\underline{p_1^2} - 2\partial_t + p_2, \underline{tp_1} - \partial_{p_1}, \underline{2\partial_{p_2}} + t\}.$$

In fact, this is a Gröbner basis with respect to the LexDeg($t < p_1 < p_2 < \partial_t, < \partial_{p_1} < \partial_{p_2}$) monomial ordering. We can reduce a polynomial from $\mathcal{A}_{t,p_1,p_2}$ with respect to this ordering.

Consider the polynomial $tp_1\partial_{p_1} - t$. To reduce this modulo this ideal we remark that the leading term is a multiple of the leading term of one of the elements of the basis, namely $\underline{tp_1} - \partial_{p_1}$. Thus, we can reduce the leading monomial by taking the well chosen multiple of this element (to make a syzygy, in fact):

$$tp_1\partial_{p_1} - t \xrightarrow{\partial_{p_1}(\underline{tp_1} - \partial_{p_1})} -\underline{\partial_{p_1}^2}.$$

Remark, $\partial_{p_1}(tp_1 - \partial_{p_1}) = t\partial_{p_1}p_1 - \partial_{p_1}^2 = tp_1\partial_{p_1} - t - \partial_{p_1}^2$ by the commutation rule of p_1 and ∂_{p_1}. The leading term of the reduced polynomial is not a multiple of a leading term of any element of the basis, thus, it is completely reduced. Since it is not 0, we conclude that it is not in the ideal.

3.4.3 Effective Integration

We are now sufficiently equipped to return to the discussion of effective, holonomy preserving operations. We immediately focus our energy on integration, since it most closely resembles the symmetric function scalar product, and algorithms for its effective calculation are known.

Here we follow [20, Ch. 10] to give an indication of how to use Theorem 3.6 to make integration effective for D-finite functions.

Suppose we are given a D-finite function $f(x, y) \in \mathbb{R}[\![x, y]\!]$ satisfying

$$\lim_{y \to \pm\infty} x^a y^b \partial_x^c \partial_y^d \cdot f(x,y) = 0, \tag{3.5}$$

for all $a, b, c, d \in \mathbb{N}$. More precisely, say we have a D-finite description of $f(x, y)$, that is, the differential system it satisfies. We can use holonomic systems to determine a differential equation satisfied by $F(x) = \int_{-\infty}^{\infty} f(x, y)\, dy$ in the following way.

Further, suppose we have some way to compute a non-trivial $D \in \mathcal{A}_x$, which decomposes into a sum of two (non-trivial) operators,

$$D = S + \partial_y T,$$

with $S \in I_F$ and $T \in \mathcal{A}_{x,y}$. In this case,

$$D \cdot f(x, y) = (S + \partial_y T) \cdot f(x, y) = 0 + \partial_y T \cdot f(x, y). \tag{3.6}$$

Thus, integrating the leftmost and rightmost sides of Eq. (3.6), and applying the fundamental theorem of calculus, we have

$$\int_{-\infty}^{\infty} D \cdot f(x, y)\, dy = \int_{-\infty}^{\infty} \partial_y \cdot (T \cdot f)\; dy = [T \cdot f]_{-\infty}^{\infty} = 0,$$

by Eq. (3.5). Now, since $D \in \mathcal{A}_x$ is not a function of y nor ∂_y, it commutes, as an operation, with integration by y:

$$\int_{-\infty}^{\infty} D \cdot f(x, y)\, dy = D \cdot \int_{-\infty}^{\infty} f(x, y)\, dy = D \cdot F(x) = 0.$$

This gives us $D \cdot F(x) = 0$, that is, a non-trivial differential equation satisfied by $F(x)$.

Such an element D lives in the vector space

$$(I_f + \partial_y \mathcal{A}_{x,y}) \cap \mathcal{A}_x. \tag{3.7}$$

This is a *left* ideal plus a *right* ideal intersected with a particular sub-algebra. The fact that this intersection is non-trivial follows from Theorem 3.6 (in the next section) combined with some closure properties of holonomic modules.

Takayama [80] has developed an algorithm to find elements precisely like D, thus making integration effective. Iterating over k, he generates bases of $(\mathcal{B}_k \cap I_f)$ and $(\mathcal{B}_k \cap \partial_y \mathcal{A}_{x,y})$[1]. The sum of these contains a non-trivial holonomic module, we use an elimination ordering to determine a Gröbner basis of a non-trivial (sub)ideal contained in the intersection given by Eq. (3.7). Success is guaranteed by Theorem 3.6.

[1] $\mathcal{B}_k$ is the Bernstein filtration defined on page 51.

In Part 2, when we consider an effective scalar product, the problem will be of a similar nature. Remark that the key step involved eliminating variables in a sum of two ideals, one a left ideal and the other a right. Essentially, this elimination succeeds because of a holonomic module which is contained in the sum, to which we apply Theorem 3.6.

3.5 Holonomy and elimination

The holonomy is used in effective integration because it guarantees that the submodule formed by the intersection of a holonomic module and reduced variable set was non-trivial. This was important in the above integration to find the element D which contained only x and ∂_x. The formal statement of this property of holonomic modules is as follows.

Theorem 3.6 (Holonomy and elimination; Bernstein). *Let $x = x_1, \ldots, x_r$, and suppose that $\mathcal{I}$ is a left ideal of $\mathcal{A}_n$ such that $\mathcal{A}_n/\mathcal{I}$ is a holonomic $\mathcal{A}_n$-module. Then the subalgebra S of $\mathcal{A}_n$ generated by any of the $r+1$ of the $2n$ elements taken from the generators of $\mathcal{A}_n$ has a non-trivial intersection with $\mathcal{I}$. In particular, this is true for*

$$S = \{x_r, \partial_1, \partial_2, \ldots, \partial_r\},$$

implying that it is possible to simultaneously eliminate all polynomials from $\mathcal{I}$ which contain $x_1, x_2, \ldots x_{r-1}$, and still have a non-trivial ideal remaining.

Proof. Define $S_n = S \cap \mathcal{B}_n$, a filtration of S, contained in the Bernstein filtration. The sequence of dimensions of S_n over $\mathbb{K}$ is of asymptotic order $O(r+1)$ since S is generated by $r+1$ elements.

At the same time, the module $A_r/\mathcal{I}$ is filtered by, say, $I_n = \mathcal{B}_n/(\mathcal{B}_n \cap \mathcal{I})$. The sequence of dimensions of $I_n = O(r)$ by virtue of the holonomicity of $\mathcal{I}$.

Both of these filtrations exist as a sequences of subspaces $(\mathcal{B}_n)$, that is $I_n \cup S_n \subseteq B_n$. Suppose I_n and S_n are disjoint for all n. Then, $\dim_{\mathbb{K}} I_n \cup S_n = O(2r+1)$, which, for n large enough is too large, since it is contained in B_n, which has dimension $O(2r)$. Thus, there must be a non-trivial intersection of $I_n \cup S_n \subseteq B_n$ eventually, giving a S and $\mathcal{I}$. ✪

Chapter IV

Non-commutative algebras of linear operators

Just as Weyl algebras provide suitable algebraic machinery to manipulate differential equations, it is possible to define a similar algebraic structure for other families of linear operators. The property of Weyl algebras, owing to Galligo [29], which enables effective closure properties of D-finite ideals is the existence of Gröbner bases. This is also true in the more general setting of Ore algebras, which are introduced in this chapter. Chyzak and Salvy, in [16], define ∂-finite functions, and generalize many of the important results of the previous chapters to the context of linear operators of Ore type. This makes it possible to enfold, in a common theoretical framework, effective computations such as summation, multiplication, certain specializations and integration. Their work capitalizes on the fact that division algorithms are available in *skew polynomial rings*, thus making it possible to generalize Buchberger's algorithm. One important contribution of Chyzak's thesis is an effective implementation in Maple of these closure properties, in [17]. The setting is very general, and the system is well suited to mixed problem types.

The connection to our investigation of symmetric functions and D-finiteness stems from the suitability of Ore algebras for computation of q-specializations of symmetric functions. In particular, this set up is well-suited to describe effective maps from the ring of symmetric functions to $\mathbb{K}(q)[\![t]\!]$ which yield ∂-finite functions. These are examined in the final section of this chapter.

4.1 Ore algebras

The commutation rule central to the definition of Weyl algebras comes from Leibnitz' rule for differentiating a product:

$$(f(x)g(x))' = f'(x)g(x) + f(x)g'(x).$$

This is specific instance of a *Skew polynomial ring*, a ring generated by operators denoted ∂ and x which act on some other ring of functions, for example, $\mathbb{K}[\![x]\!]$ or $\mathbb{K}[n]$.

Definition 4.1 ***Skew polynomial ring.*** A skew polynomial ring $A[\partial; \sigma, \delta]$, is defined for any integral domain A; where ∂ as an A-endomorphism satisfying a relation of the type

$$\partial \cdot f = \delta(f) + \sigma(f)\partial;$$

with σ any A-endomorphism, and δ a linear function satisfying

$$\delta \cdot (fg) = (\sigma \cdot f)\,(\delta \cdot g) + (\delta \cdot f)\,g$$

for all f and g. In this case δ is said to be a *σ-derivation.*

In the differential case "∂" is differentiation and "x" is multiplication by x acting on differentiable functions. Thus, we set $\sigma(x) = x$ and $\delta = \frac{d}{dx}$. A second example sets "∂" to the *shift* operator S_n, which sends $f(n)$ to $f(n+1)$, and "x" is multiplication by n, and these operator act on integer functions. In this case, $\sigma(f) = 0$ and $\delta = S_n$.

In these cases, it is sufficient to describe σ and δ of the above equation to determine how the two operations interact. In the cases where σ and δ are constants, the operations are called *Ore operators*. In the case when A is a field we call it an *Ore algebra*. It is also possible to describe Ore algebras of several operators. This general case shall be denoted

$$\mathbb{O}_n = \mathbb{K}[\partial_1; \sigma_1, \delta_1][\partial_2; \sigma_2, \delta_2] \cdots [\partial_n; \sigma_n, \delta_n] \tag{4.1}$$

When the context is clear, we may also denote this by $\mathbb{K}[\partial_1, \ldots, \partial_n]$. Table 4.1 describes some Ore operators.

The key properties of skew polynomial rings of interest here include the fact that this algebra permits division algorithms, and that we have the necessary machinery to describe an analogue of Buchberger's algorithm.

Operator	$(\partial \cdot f)(x)$	$(x \cdot f)(x)$	$(\partial \cdot fg)(x)$
Differentiation, $\frac{d}{dx}$	$f'(x)$	$xf(x)$	$f(x)(\partial \cdot g)(x) + (\partial \cdot f)(x)g(x)$
Shift, S_x	$f(x+1)$	$xf(x)$	$f(x+1)(\partial \cdot g)(x)$
Difference, Δ	$f(x+1) - f(x)$	$xf(x)$	$f(x+1)(\partial \cdot g)(x) + (\partial \cdot f)(x)g(x)$
q-Dilatation, $H_{x;q}$	$f(qx)$	$xf(x)$	$f(qx)(\partial \cdot g)(x)$
q-Differentiation, D_q	$\frac{f(qx)-f(x)}{(q-1)x}$	$xf(x)$	$f(qx)(\partial \cdot g)(x) + (\partial \cdot f)(x)g(x)$
q-shift, $S_{x;q}$	$f(x+1)$	$q^x f(x)$	$f(x+1)(\partial \cdot g)(x)$

TABLE **4.1** Ore operators and their Leibnitz rules

4.2 A generalization of D-finite: ∂-finite

One strong motivation for studying D-finite functions is that, from a computer algebra perspective, they can be represented by a finite amount of information well suited for algebraic manipulation. This allows *automatic* verification of identities of a certain nature. Algebraically speaking, their annihilating ideals are defined by a finite number of relations. We generalize this aspect of a D-finite ideal to Ore algebras as follows.

Definition 4.2 ***∂-finite ideal.*** Let $\mathbb{O}$ be a an Ore algebra over a field $\mathbb{K}$. A left ideal $\mathcal{I}$ of $\mathbb{O}$ is *∂-finite* if $\mathbb{O}/\mathcal{I}$ is a finite dimensional vector space over $\mathbb{K}$.

The functions (or series, or sequences) upon which these operators act, which are annihilated by a ∂-finite idea are called ∂-finite. Thus, to be ∂-finite with respect to the Ore algebra $\mathbb{K}(x)[\partial_x; 1, \partial_x]$ of operators acting on $\mathbb{K}[\![x]\!]$ corresponds the usual notion of D-finiteness and, likewise P-recursiveness is equivalent to ∂-finiteness with respect to $\mathbb{K}[n][S_n; 0, S_n]$, acting on the ring of sequences.

In the case of the Weyl algebra there is a direct correspondence between D-finite functions and holonomic functions. It is therefore natural to ask if a similar quality holds for ∂-finite functions, but to date, no such general quality is known.

4.3 Closure properties of ∂-finite functions

Remarkably, in the rather general setting of Ore algebras it is possible to describe (effective) closure properties. Thus, many of important characteristics of the Weyl algebra case remain true. The next theorem summarizes the major closure properties

of the set of ∂-finite ideals.

Theorem 4.1 (Closure properties of ∂-finite functions; Chyzak and Salvy). *Suppose f and g are ∂-finite functions with respect to $\mathbb{O}_n$. Then*

1. *The function $f+g$ is a ∂-finite functions with respect to $\mathbb{O}_n$;*
2. *The function fg is a ∂-finite functions with respect to $\mathbb{O}_n$;*
3. *Given any $P \in \mathbb{O}_n$, f satisfies an equation $\sum_{i=0}^{k} \left(a_i P^i\right) \cdot F = 0$ with $k \leq \dim \mathbb{O}/\operatorname{ann} f$.*

There is also a result for specializations:

Proposition 4.2 (Specializations of ∂-finite functions; Chyzak and Salvy). *Let $x = x_1, \ldots, x_n$ and $y = y_1, \ldots, y_m$. If $f(x, y)$ is ∂-finite with respect to*

$$K(x,y)[\partial_x; \sigma_x, \delta_x][\partial_y; \sigma_y, \delta_y],$$

then for any $a \in \mathbb{K}^m$, the specialization $f(x, a)$ is ∂-finite with respect to

$$K(x)[\partial_x; \sigma_x, \delta_x].$$

4.3.1 Effective closure properties

The characterization of ∂-finite ideals by *rectangular systems* is useful for identification purposes. However, in general they can be computationally intensive to determine.

Definition 4.3 ***rectangular system.*** A system of polynomials

$$P_i(x_1, \ldots, x_n, \partial_1, \ldots, \partial_n), \quad 1 \leq i \leq n$$

of an Ore algebra is said to be *rectangular* when each ∂_i is involved in exactly one of its elements. That is, we can rearrange the indices to write

$$P_i(x_1, \ldots, x_n, \partial_1, \partial_2, \ldots, \partial_n) = Q_i(x_1, \ldots, x_n, \partial_i), 1 \leq i \leq n.$$

For example, the following differential system of $\mathcal{A}_3$, $\{\partial_1^2 x_2 + 3, \partial_2^2 + 2\partial_2, \partial_3 - 1\}$, is rectangular. The following result is a straightforward consequence of linear dependence.

Proposition 4.3. *An ideal of an Ore algebra is ∂-finite if and only if it contains a rectangular system.*

4.3.2 Summation

One of the most useful closure properties of holonomic functions is closure under integration. This property generalizes in Ore algebras as the "anti-derivatives" of Ore operators.

Consider the Ore algebra $\mathbb{O} = \mathbb{K}(x_i)[\partial_x; \sigma_x, \delta_x]$, of operators acting upon an algebra $\mathcal{F}$ of functions. We assume the existence of an indefinite operator ∂^{-1} and a definite operator ∂^{-1}, for some boundary. For example, in the differential case $\partial = \frac{d}{dx}$, ∂^{-1} corresponds to integration (modulo some analytic conditions) and in the shift case $\partial = S_n - 1$, we have that $\partial^{-1} = \sum_{-\infty}^{n-1}$. Further, we assume that they commute with the ∂_j of $\mathbb{O}$ whenever $j \neq i$ and that they satisfy $\partial^{-1}\partial = \partial\partial^{-1} = 1$ in the indefinite case and $\partial^{-1}\partial = \partial\partial^{-1} = 0$ in the definite case. This latter requirement is frequently a constraint on the functions from $\mathcal{F}$.

Effective versions of the anti-derivative can be determined using a modified strategy to that presented for integration in Section 3.4.3. In particular, we determine an annihilating ideal contained in the sum of a left ideal and a right ideal. The success of our approach relies in part on the following result, which generalizes Theorem 3.6.

Theorem 4.4 (Ore Algebras and Elimination; Chyzak). *Given Ore algebra $\mathbb{O}$, suppose that J is a left ideal of $\mathbb{O}$ and consider the subalgebra S generated by a family*

$$\{x_{i_1}, \ldots, x_{i_u}, \partial_{j_1}, \ldots, \partial_{j_v}\}$$

of $u + v$ indeterminates taken from the set of generators of $\mathbb{O}$. Then, if the dimension d of $\mathbb{O}/J$ is such that $d > u + v$, then the intersection $S \cap J$ is non-trivial. That is, it is possible to simultaneously eliminate at least $r + s - d - 1$ indeterminates of the ideal J.

One application of this theory here concerns the effective summation of ∂-finite functions, as we shall see in the following section.

4.4 Some ∂-finite preserving q-specializations

The q-specializations of symmetric functions ex_q and ps introduced in Section 2.5.1 describe generating functions of many combinatorial objects, for example plane partitions [75, §7]. Now, q-series appear as refinements of general results, and D_q-equations

Generating series	q-Differential equation satisfied
$\mathrm{ex}_q(\mathcal{H}) = \sum_n \frac{x^n}{(q)_n}$	$D_q - 1$
$\mathrm{ex}_q(\mathcal{E}) = \sum_n \frac{x^n}{(q)_n} q^{\binom{n}{2}}$	$D_q - H$
$\mathrm{ex}_q(\mathcal{H}[p_k])$	$D_q - x^{k-1}(1-q)^{k-1}$
$\mathrm{ex}_q(\sum_n h_{n,n})$	$xq{D_q}^2 + \left(-x^2q^3 + x^2q + 1\right) D_q + x^3q(q-1)^2 - x(q+1)^2$

TABLE **4.2** Symmetric series under ex_q

are refinements of differential equations, in the sense that D_q becomes a derivation as q tends to 1:

$$\lim_{q\to 1}\left(D_q \cdot F(x;q)\right) = \frac{d}{dx}\left(\lim_{q\to 1} F(x;q)\right).$$

Here, we use Ore algebras to determine D_q-equations of q-specializations of symmetric functions.

4.4.1 Two Ore algebras of interest

Two Ore operators of particular interest to this discussion are the q-differentiation (D_q) and q-dilation (H_q):

$$D_q \cdot F(x;q) = \frac{F(x;q) - F(qx;q)}{(1-q)x} \quad \text{and} \quad H_q \cdot F(x;q) = F(qx;q). \tag{4.2}$$

We note that $D_q \cdot \mathrm{ex}_q(\mathcal{H}) = \mathcal{H}$ and $H_q \cdot \mathrm{ex}_q(\mathcal{E}) = \mathcal{E}$[1]. Thus, if we define corresponding Ore algebras $\mathbb{O}_D$ and $\mathbb{O}_H$ generated by these operations over $\mathbb{K}$, we have already found a ∂-finite element for each. This suggests the following problem.

Problem 10. *Characterize the elements of $\mathbb{K}[\![p]\!]$ whose image under ex_q are ∂-finite with respect to either $\mathbb{O}_D$ or $\mathbb{O}_H$.*

A partial answer can be obtained by applying the effective summation described in the previous section to families of symmetric polynomials, and a flavour of the corresponding results appear in Table 4.2. These were calculated automatically with the aid of the `Holonomy` package of Chyzak [17].

We obtain a second family of examples using some basic plethysms. The following lemma is inspired by the q-exponential formula that Gessel describes in [30].

[1] Recall $\mathcal{H} = \sum_n h_n$ and $\mathcal{E} = \sum_n e_n$

Lemma 4.5. *The following D_q-equation is valid for any k:*

$$\left(D_q - x^{k-1}(1-q)^{k-1}\right) \cdot \mathrm{ex}_q(\mathcal{H}[p_k]) = 0 \tag{4.3}$$

Proof. By definition, $\mathcal{H}[p_k] = \sum_\lambda \frac{p_{k\lambda}}{z_\lambda}$ and thus, when we apply ex_q

$$\begin{aligned}\mathrm{ex}_q\left(H[p_k]\right) &= \sum_{n\geq 0} x^{kn}(1-q)^{kn} \sum_{\lambda\vdash n} \frac{1}{\prod_i (1-q^{k\lambda_i})} \\ &= \sum_{n\geq 0} x^{kn}(1-q)^{kn} \frac{1}{(q^k)_n}.\end{aligned}$$

From which we deduce

$$x^{k-1}(1-q)^{k-1}\,\mathrm{ex}_q\left(H[p_k]\right) = \sum_{n\geq 1} \frac{x^{kn-1}(1-q^{kn-1})}{(q^k)_{n-1}}.$$

Now,

$$\begin{aligned}D_q \cdot \mathrm{ex}_q\left(H[p_k]\right) &= \sum_{n\geq 0} \frac{x^{nk-1}(1-q)^{nk-1}(1-q^{nk})}{(q^k)_n} \\ &= \sum_{n\geq 0} \frac{x^{nk-1}(1-q)^{nk-1}}{(q^k)_{n-1}}.\end{aligned}$$

The $n=0$ term in the bottom sum vanishes since when $n=0$, $q^{nk}-1=0$, and thus these two series are equal. ✪

In general the problem of determining D_q-equations of specialization of $\mathcal{H}[p_k^n]$ for $n>1$ is much harder.

This is certainly an interesting direction to explore. Indeed, one can also ask the same question about other specializations, and about other Ore algebras and arrive at the following definition and subsequent general question.

Definition 4.4 ***∂-finite preserving q-specialization.*** Let $\mathbb{O}$ be an Ore algebra generated by Ore operators that act on $\mathbb{K}(q)[\![x]\!]$. A homomorphism $\phi : \mathbb{K}[\![p]\!] \to \mathbb{K}(q)[\![x]\!]$ is a *∂-finite preserving q-specialization for* $\mathbb{O}$ if for any D-finite symmetric series $F \in \mathbb{K}[\![p]\!]$, $\phi(F)(x;q)$ is a ∂-finite function of O.

Problem 11. *Determine a suitable Ore algebra $\mathbb{O}$ of operators acting on $\mathbb{K}(q)[\![t]\!]$ and a homomorphism $\phi : \mathbb{K}[\![p]\!] \to \mathbb{K}(q)[\![x]\!]$ that is ∂-finite preserving.*

4.5 Application: Enumeration of plane partitions

To close this section, we describe a combinatorial application of this collection of techniques towards the enumeration of plane partitions.

A *plane partition* is a two dimensional subarray of positive integers such that they are (weakly) decreasing horizontally from left to right, and from top to bottom. For example,

$$\begin{array}{cccc} 6 & 4 & 4 & 1 \\ 6 & 3 & 2 & \\ 1 & 1 & & \end{array} .$$

Remark that the shape of a plane partition is a partition, in this case (4,3,1). The weight of the plane partition is the sum of the elements in the subarray, in this case 28. Define the generating series $P_\lambda(q)$ defined as the sum over all plane partitions p of shape λ,

$$P_\lambda(q) = \sum q^{weight(p)}.$$

The fact that relates plane partitions to the discussion here, is the surprising result that

$$P_\lambda(q) = \mathrm{ps}\, s_\lambda.$$

Thus, using the summation technique for Ore algebras, we can determine the D_q equations satisfied by generating series of plane partitions.

Part 2

Algorithms

Summary of this part

The goal of this part is to provide effective algorithms closure properties described in the previous part. The inputs and outputs of the algorithms should be understandable to those who have bypassed the description of holonomy in the previous part, however, the description of the algorithms and certainly the proof that they offer what they promise requires holonomy. Applications of these algorithms are provided in the next part.

The algorithms are defined on the following pages:

Algorithm Name	**What it computes**	**Location**
SCALAR_DE	$\langle f(p), g(p,t) \rangle$	Page 80
HAMMOND	$\langle f(p), \sum_n h_k^n t^n \rangle$	Page 85
SCALAR_DE2	$\langle f(p,t), g(p,t) \rangle$	Page 89
ITENSOR_DE	$f(p) * g(p)$	Page 109

Chapter V

An effective scalar product

This chapter focuses on calculating the scalar product of symmetric functions, introduced in Section 2.1.2. We first briefly recall some classic techniques for computation and then introduce our solution using holonomic systems. This results in three different algorithms, which are presented and then analysed.

5.1 Existing techniques for computing the scalar product

Scalar product calculations arise in several applications. It is thus useful to have an efficient, or at least effective, way to compute them. Many algorithms to compute scalar products rely on rewriting all functions considered in some orthogonal basis (see Eq. (1.2.4) and Eq. (1.2.5). This is the usual method to compute the scalar product of symmetric functions in the main symmetric function computer algebra packages `ACE` [83], `SCHUR` [85], `SF` [77], and `Symmetrica` [46]. The major drawback of this technique is that the computations can become intensive for symmetric functions of degree as low as 10, and are unsuited to computations with series.

However, many generating series of symmetric functions have a nice closed form (see Table 2.1, for example), it is thus reasonable to try to determine scalar products in terms of generating series. This is much more efficient, when it is possible, and it allows computations of scalar products of higher degree. In particular, if the generating function is D-finite, the coefficients are P-recursive. Thus they are computable in a number of arithmetic operations linear in n in terms of simple polynomial algebra computations.

This is precisely the approach presented for a special case by Goulden and Jackson in in [36, 37]. They outline, via two well-chosen examples, a method for computing

$\left\langle f, \sum_\lambda h_\lambda t^\lambda \right\rangle$ and $\langle f, \sum_\lambda h_{1^n} t^n \rangle$. They call the former the *Hammond series* of f, a nod to the indirect use of Hammond operators.

Gessel, in [34] contributes techniques to compute scalar products of the form

$$\left\langle f, \sum_m h_1^n h_2^m t^m \right\rangle \quad \text{and} \quad \left\langle f, \sum_m h_1^n h_3^m t^m \right\rangle$$

for fixed n. The explicit formulas he gives require that f be expressed as a formal sum in the power sum basis of symmetric series. He indicates that other cases would be rather cumbersome to pursue in this manner.

Here we consider the general automatic computation of $\langle f, g \rangle$, for D-finite g provided f satisfies a natural condition. More accurately, we give an algorithm to compute a differential equation satisfied by $\langle f, g \rangle$. In that sense, the algorithms presented in this chapter are similar to effective integration of holonomic functions. We also prove correctness and termination of the algorithms.

5.2 Calculating $\langle f, g \rangle$ by holonomic systems

In this section we introduce a basic algorithm to compute the differential equation satisfied by the scalar product of two D-finite symmetric series (under some hypotheses), one of which depends on the extra variables (typically denoted using t) that "survive" the scalar product computation. Hence, we get a result which is a series in these variables. When the number of additional variables is 1, the output is a single differential equation for which existing computer algebra algorithms might find a closed-form solution. In most cases however, no such solution exists and we are content with a differential equation from which useful information can be extracted. Once the ideas are clear on how to proceed, we describe a succinct formulation of the Hammond series algorithm. The third algorithm generalizes the first, and allows the t variables to appear in both symmetric series.

The basic tool in use here is non-commutative Gröbner bases in extensions of Weyl algebras. We work primary with two Weyl algebras $\mathcal{A}_{p,t}(t)$, the algebra of differential operators polynomial in p but possibly rational in t; and $\mathcal{A}_t(t)$, the restriction of $\mathcal{A}_t$ to $\mathbb{K}(t)$.

For the algorithm, we work in an extension

$$\mathcal{A}_{p,t}(t) = \mathbb{K}(t) \otimes_{\mathbb{K}[t]} \mathcal{A}_{p,t}$$

of the Weyl algebra in which the coefficients of the differential operators are still polynomial in p but rational in t. Suppose f and g are D-finite symmetric series from $\mathbb{K}[t][\![p]\!]$ as per the hypotheses of Theorem 2.7. In particular, they both satisfy systems of linear differential equations with coefficients polynomials from $\mathbb{K}(t)[p]$. We can write these equations as elements of $\mathcal{A}_{p,t}(t)$ acting on f and g. The set $\mathcal{I}_g = \operatorname{ann}_{\mathcal{A}_{p,t}(t)} f$ (resp. $\mathcal{I}_g$) of all operators of $\mathcal{A}_{p,t}(t)$ annihilating f (resp. g) is then a *left* ideal of $\mathcal{A}_{p,t}(t)$. Given as input Gröbner bases for $\mathcal{I}_f$ and $\mathcal{I}_g$, our algorithm outputs non-zero elements of the annihilating left ideal $\operatorname{ann}_{\mathcal{A}_t(t)} \langle f, g \rangle$.

Example. Suppose

$$f(p_1, p_2) = \exp\big((p_1^2 - p_2)/2 - p_2^2/4\big) \quad \text{and} \quad g(p_1, p_2, t) = \exp\big(t(p_1^2 + p_2)/2\big).$$

Differentiating f with respect to p_1 yields a differential equation

$$\frac{\partial f(p_1, p_2)}{\partial p_1} - p_1 f(p_1, p_2) = 0.$$

Differentiating f and g with respect to all variables in this manner gives generators for their annihilating $\mathcal{A}_{p,t}$-ideals:

$$\mathcal{I}_f = \langle p_2 + 2\partial_{p_2} + 1, p_1 - \partial_{p_1} \rangle \quad \text{and} \quad \mathcal{I}_g = \left\langle p_1^2 + p_2 - 2\partial_t, 2\partial_{p_2} - t, \partial_{p_1} - t p_1 \right\rangle.$$

These ideals are used by the algorithms to compute the following element of $\mathcal{I}_{\langle f,g \rangle}$:

$$2(1-t)\partial_t - t^2.$$

To calculate such a result, elements of $\mathcal{I}_f$ and $\mathcal{I}_g$ are combined with the aid of the adjunction map $\perp$. Recall this is defined for an operator $P \in \mathcal{A}_{p,t}$ by $\langle P \cdot f, g \rangle = \left\langle f, P^\perp \cdot g \right\rangle$. Observe that the adjunction map is an involution as well as an algebra anti-automorphism. For any $\mathcal{A}_p$-ideal $\mathcal{I}$, define

$$\mathcal{I}^\perp = \{m^\perp : m \in I\}.$$

Note that, although adjunction extends to $\mathcal{A}_p(t)$ by setting $t_i^\perp = t_i$, no adjoint for the ∂_{t_i} can be defined in any consistent way.

We now proceed to outline the algorithm for the simple case. From this point on we elect to have $f \in \mathbb{K}[\![p]\!]$, i.e., f independent of the variables t. The condition on f that it does not involve t implies that $\partial_{t_i} \cdot f = 0$ for i from 1 to k. We can use this fact to simplify our calculations. In this case, we will take $J_f = \operatorname{ann}_{\mathcal{A}_p} f$. Note that $J_f = \mathcal{I}_f^{\perp} \cap \mathcal{A}_p$.

This allows us to determine the action of combinations of $P \in \mathcal{J}_f^{\perp}\mathcal{A}_t(t)$ and $Q \in \mathcal{I}_g$. For example, given $S \in \mathcal{A}_p$, $T \in \mathcal{A}_{p,t}$,

$$\Big\langle f, (P^{\perp}S + TQ) \cdot g \Big\rangle = \Big\langle S^{\perp}P \cdot f, g \Big\rangle + \Big\langle f, TQ \cdot g \Big\rangle = 0.$$

Furthermore, if we can find a combination such that $P^{\perp}S + TQ = R \in \mathcal{A}_t$, and is non-zero, we have $0 = \langle f, R \cdot g \rangle = R \cdot \langle f, g \rangle$.

Thus, we conduct our search for an element of $\operatorname{ann}_{\mathcal{A}_t} \langle f, g \rangle$ by determining a non-zero element of $(\mathcal{J}_f^{\perp}\mathcal{A}_t(t) + \mathcal{I}_g) \cap \mathcal{A}_t$. We shall prove in Section 5.6.1 that such an element exists. Basically, the goal of our algorithms is to compute sufficiently many non-zero elements of $(\mathcal{J}_f^{\perp}\mathcal{A}_t(t)+\mathcal{I}_g)\cap\mathcal{A}_t$ so as to generate a D-finite description of the scalar product. We now give an example of what this set up looks like, followed by a formalization of the ideas.

Example. We continue the above example to illustrate how to find such a combination. We remark that the generators we determined for $\mathcal{I}_f$ and $\mathcal{I}_g$ are in fact Gröbner bases with respect to the DegRevLex ordering, where monomials are first compared by degree, with ties broken by reverse lexicographical ordering. The variables are ordered $t < \partial_t < p_1 < p_2 < \partial_{p_1} < \partial_{p_2}$.

To generate an element of $\mathcal{J}_f^{\perp}\mathcal{A}_t(t) + \mathcal{I}_g$ we begin with a monomial, say $p_1\partial_{p_1}$, and reduce it modulo the two ideals. In order to create the desired syzygy once we apply adjunction, however, we must apply $\perp$ to the monomial and the order of the variables before reducing with respect to $\mathcal{J}_f$. This has no effect on this particular monomial, however we reverse the ordering of the p_i and ∂_{p_1}. The reduction resembles:

$$p_1\partial_{p_1} \xrightarrow{\mathcal{J}_f} \partial_{p_1}^2 \qquad p_1\partial_{p_1} \xrightarrow{\mathcal{I}_g} tp_1^2 \xrightarrow{\mathcal{I}_g} -tp_2 + 2\partial_t.$$

By this reduction we have that $p_1\partial_{p_1} - \partial_{p_1}^2 \in \mathcal{J}_f$ and $p_1\partial_{p_1} + tp_2 - 2t\partial_t \in \mathcal{I}_g$. We use adjunction to combine them in the following way:

$$\begin{aligned} 0 &= \langle 0, g \rangle + \langle f, 0 \rangle = \left\langle \left(p_1\partial_{p_1} - \partial_{p_1}^2\right) \cdot f, g \right\rangle + \left\langle f, \left(p_1\partial_{p_1} + tp_1^2\right) \cdot g \right\rangle \\ &= \left\langle f, \left((p_1\partial_{p_1} - \partial_{p_1}^2)^{\perp} + p_1\partial_{p_1} - tp_2 + 2t\partial_t\right) \cdot g \right\rangle \\ &= \left\langle f, \left(-p_1^2 + tp_2 - 2t\partial_t\right) \cdot g \right\rangle. \end{aligned}$$

> We have thus generated, $-p_1^2+tp_2-2t\partial_t$, an element of $\mathcal{J}_f^{\perp}\mathcal{A}_t(t)+\mathcal{I}_g$. By considering other monomials we can generate other elements of this set. We describe below how we can take linear combinations of such elements over $\mathbb{K}(t)$ to find an element in $\mathcal{A}_t$.

Remark the similarity to the integration situation. Here $\mathcal{I}_g$ is a left $\mathcal{A}_{p,t}(t)$ ideal, $\mathcal{J}_f^{\perp}\mathcal{A}_t(t)$ is *right* $\mathcal{A}_{p,t}$-ideal and objects of the form $p+q$, $p \in \mathcal{J}_f^{\perp}\mathcal{A}_t(t)$ and $q \in \mathcal{I}_g$ does not form an ideal. We generate elements of $\mathcal{J}_f^{\perp}\mathcal{A}_t(t)$ and $\mathcal{I}_g$, and reduce them in this vector space.

The structure of the sum $J_f^{\perp}\mathcal{A}_t(t)+\mathcal{I}_g$ that we use is that of a vector space over $\mathbb{K}(t)$. The idea is to use $K(t)$-linear algebra with the vector space structure to eliminate both the the ∂_{p_i}'s and the p_i's. Roughly speaking, we perform Gaussian elimination to remove the monomials involving the p_i's and the ∂_p's.

The main loop of the algorithm considers monomials of increasing degree with respect to any monomial ordering on $p, \partial_p, \partial_t$, though we typically choose DegRevLex($t \preceq \partial_t \preceq p \preceq \partial_p$). We reduce each monomial α with respect to (the Gröbner bases for) $\mathcal{J}_f^{\perp}\mathcal{A}_t(t)$ and $\mathcal{I}_g$. Note that the chosen monomial ordering is the same for both $\mathcal{I}_g$ and $\mathcal{J}_f^{\perp}\mathcal{A}_t(t)$. As a variant calculation, the remainder of the reduction of a monomial α with respect to $\mathcal{J}_f^{\perp}\mathcal{A}_t(t)$ can be viewed as the adjoint of the remainder of the reduction of $\alpha^{\perp}$ with respect to $\mathcal{I}_f$. However, to reflect the fact that adjunction modifies the variables, when reducing with respect to $\mathcal{I}_f$ we need to use a different order, specifically, the ordering $\preceq_{\perp}$ defined by $\beta_1 \preceq_{\perp} \beta_2$ if and only if $\beta_1^{\perp} \preceq \beta_2^{\perp}$. Notice, here this order is simply DegRevLex($\partial_p \preceq p$).

We now state the algorithm more formally as Algorithm 1, followed by an example in the next section. After this example, we describe the modifications necessary to handle specific cases more efficiently, and how to treat the general case. The proofs that these algorithms work and terminate are delayed to Section 5.6.

Notice, if $m = 1$, as will be the case in our examples, there is only one variable t, and the dimension condition in 3(d) is simplified to

> (d) If B contains an element $P \neq 0$ from $\mathcal{A}_t$, break and return P.

The remainder of the reduction with respect to the Gröbner basis $\mathcal{G}_g$ is a multivariate

ALGORITHM: SCALAR_DE

INPUT: *Symmetric functions* $f \in \mathbb{K}[\![p]\!]$ *and* $g \in \mathbb{K}[t][\![p]\!]$*, both D-finite in* p, t*, respectively given by a system of linear differential operators of* $\mathcal{A}_p$ *and* $\mathcal{A}_{p,t}(t)$*.*

OUTPUT: *A system of differential equations satisfied by* $\langle f, g\rangle$*, which describes it as D-finite.*

1. Determine a Gröbner basis $\mathcal{G}_g$ for the left ideal $\operatorname{ann}_{\mathcal{A}_{p,t}(t)} g$ with respect to any monomial ordering $\preceq$, as well as a Gröbner basis $\mathcal{G}_{f^\perp}$ for the right ideal $\operatorname{ann}_{\mathcal{A}_p} f^\perp$ with respect to the monomial ordering induced by $\preceq$ on $\mathcal{A}_p$;
2. $B := \{\}$;
3. Iterate through each monomial α in $p, \partial_p, \partial_t$ in the increasing order given by $\preceq$;
   (a) Write $\alpha = \beta\gamma$ with $\beta \in \mathcal{A}_p$ and $\gamma \in \mathbb{K}[\partial_t]$;
   (b) $\alpha_f := \big(\beta - (\beta \operatorname{red}_{\preceq} \mathcal{G}_{f^\perp})\big)\gamma$;
   (c) $\alpha_g := \alpha - (\alpha \operatorname{red}_{\preceq} \mathcal{G}_g)$;
   (d) Introduce α_f and α_g as two new elements into B and reduce so as to eliminate p, ∂_p;
   (e) Compute the dimension of the ideal generated by $B \cap \mathcal{A}_t(t)$. If this dimension is 0, break and output $B \cap \mathcal{A}_t(t)$.

Algorithm 1 A basic algorithm for an effective scalar product

analogue of the remainder of the division. It is such that for any α, $\alpha_{\mathcal{G}} = \alpha - (\alpha \operatorname{red} \mathcal{G})$ belongs to the ideal generated by $\mathcal{G}$. A similar statement holds for $\mathcal{G}_f$.

For this description we have assumed that Gröbner bases can be computed for both left and right ideals. If they can only be computed on one side, say for left ideals, then the operators α_f can be obtained as follows: first, determine the monomial ordering $\preceq_\perp$ induced by adjunction on $\mathcal{A}_p$ viewed as a left structure from the ordering $\preceq$ on $\mathcal{A}_p$ viewed as a right structure; then, replace the Gröbner basis $\mathcal{G}_{f^\perp}$ with the Gröbner basis $\mathcal{G}_f$ for the left ideal $\operatorname{ann}_{\mathcal{A}_p} f$ with respect to $\preceq_\perp$; α_f is then computed as $\alpha - (\alpha^\perp \operatorname{red}_{\preceq_\perp} \mathcal{G}_f^\perp)$. This way we get $\mathcal{G}_{f^\perp} = (\mathcal{G}_f)^\perp$.

The introduction into the basis B performs a Gaussian reduction of α with respect to the elements already in B and returns the new value of B. In practice, B can be handled (not inefficiently) by a computation of Gröbner basis over a module with respect to a monomial order that eliminates the p_i's and ∂_{p_i}'s.

Finally, some classical technique can be applied at Step 3(b) to avoid the repetition of reduction for the same β.

5.3 Enumerating k-regular graphs

The enumeration of regular graphs is a basic, interesting question of graph theory which has been considered in several investigations [19, 62, 37, 34]. Nonetheless, it is an instructive example of our approach and appears as the simplest case of a whole family of examples. We treat the general case in Chapter 7. The set of all simple graphs labelled with integers from $\mathbb{N} \setminus \{0\}$ can be encoded in the product:

$$G(x) = \sum_{G \in \mathcal{G}} \prod_{(i,j) \in E(G)} x_i x_j = \prod_{i<j} (1 + x_i x_j), \tag{5.1}$$

as each edge $(i, j) \in E(G)$ is either in the graph or not. We can similarly encode graphs with multiple edges (multigraphs) by

$$M(x) = \prod_{i<j} \frac{1}{(1 - x_i x_j)}.$$

Clearly both of these are symmetric functions, and in fact, $G = \mathcal{E}[e_2]$ and $M = \mathcal{H}[e_2]$. Section 7.1 will discuss how to determine these equivalences. Both are easily rewritten

in terms of the p_i's:

$$G = \exp\left(\sum_n (-1)^n (p_n^2 - p_{2n})/2n\right) \quad \text{and} \quad M = \exp\left(\sum_n \left(p_n^2 - p_{2n}\right)/2n\right) \tag{5.2}$$

In any given term, the degree of x_k gives the valency of node k. So, for example, the coefficient $g_n = [x_1 \dots x_n]G$ gives the number of 1-regular graphs, or perfect matchings on the complete graph on n vertices, and in general the coefficient $g_n^{(k)} = [x_1^k \dots x_n^k]G$, also given as $[m_{k^n}]G$, gives the number of k-regular graphs on n vertices. By virtue of (2.4), coefficient extraction amounts to a scalar product, and the generating function $G_k(t)$ of k-regular graphs is given by

$$G_k(t) := \sum_n g_n^{(k)} \frac{t^n}{n!} = \langle G, H_k \rangle \,, \tag{5.3}$$

where

$$H_k(t) := \sum_n h_{k^n} \frac{t^n}{n!} = \sum_n \frac{(h_k t)^n}{n!} = \exp(h_k t).$$

Now, as $h_k = \sum_{\lambda \vdash k} p_\lambda / z_\lambda$ (where the sum is over all partitions λ of k), the exponential generating function $H_k(t)$ is also $\exp\left(t \sum_{\lambda \vdash n} p_\lambda / z_\lambda\right)$, an exponential in a finite number of p_i's. By Theorem 1.5(3), this is D-finite. Further, as a result of scalar product property (2.5), we can rewrite Eq. (5.3) as

$$G_k(t) = \left\langle \exp\left(\sum_{n \text{ even},\ n \le k} (-1)^{n/2} \frac{p_n^2}{2n} + \frac{p_n}{n} + \sum_{n \text{ odd},\ n \le k} \frac{p_n^2}{2n} \right), \exp\left(t \sum_{\lambda \vdash k} \frac{p_\lambda}{z_\lambda} \right) \right\rangle \tag{5.4}$$

and now by Theorem 2.7 this generating function $G_k(t)$ is D-finite.

The generating function for 2-regular graphs, according to Eq. (5.4), is given by

$$G_2(t) = \left\langle \exp\left((p_1^2 - p_2)/2 - p_2^2/4\right), \exp\left(t(p_1^2 + p_2)/2\right) \right\rangle .$$

As we saw in our previous example, Algorithm 1 calculates that $G_2(t)$ satisfies the differential equation

$$2(1-t)G_2'(t) - t^2 G_2(t) = 0,$$

which is easily solved to find $G_2(t) = e^{-\frac{1}{4}t(t+2)}/\sqrt{1-t}$.

Table A.1 summarizes the results by the same algorithm for $k = 2, 3, 4$. These match with the results in [37].

5.3.1 Efficient enumeration of k-regular graphs

An efficient procedure for the enumeration of k-regular graphs derives immediately from the differential equations for the generating series of k-regular graphs collected in Table A.1. Indeed, one simply needs to convert the differential equation for $G_k(t)$ into a recurrence relation for its coefficients $g_n^{[k]}$ and to determine sufficiently many starting values $g_0^{[k]}$, $g_1^{[k]}$, ..., from which unrolling the recurrence enables one to compute $g_n^{[k]}$ for any n efficiently.

Implementations are available to help with this approach. For example, the Maple package `gfun` by Salvy and Zimmerman [70] contains commands dedicated to the conversion step and the iterative calculations based on a linear recurrence. Computations in the case $k = 4$ result in a recurrence relation of order 15 already published by Read and Wormald [62] and can be found as a formula accompanying sequence number A005815 in Sloane's encyclopedia of integer sequences [72]. From this recurrence relation and initial terms, it is then a matter of seconds to compute the exact integer values for hundreds of terms in the sequence.

It should be stressed that this method proves much more efficient than the direct computation of the scalar product based on a term wise expansion and application of formula (2.5). For example, Stembridge's implementation in the package SF for symmetric function manipulation in Maple [77] already requires several minutes to compute the $g_n^{[4]}$ for n up to 15, and becomes unsuitable to handle the symmetric functions that would be necessary to obtain $g_{20}^{[4]}$. Far from showing any weakness of SF's general approach, this illustrates the computational progress provided by our techniques in the specific setting of differentiably finite series.

5.4 Hammond series

In the example above it turned out that, apart from monomials of degree 1, we needed only to examine the monomials p_1^2 and $p_1\partial_{p_1}$ to reach the solution. In general, depending on the monomial order, the algorithm might well consider many monomials before it adds the ones that eliminate the p_i's and ∂_{p_i}'s. The problem becomes far more serious as the number of monomials increases. It turns out that, in some cases when the scalar product is of the type $\langle f, H^{(k)}(t)\rangle$, it is possible to modify the approach and eliminate

the p_i's and the ∂_{p_i}'s in a manner using the *Hammond series*[1] of Goulden, Jackson and Reilly in [37]. Here we offer a concise description of their approach and give an explicit algorithm for their technique.

For $f \in \mathbb{K}[\![p_1, p_2, \dots]\!]$, the Hammond series of f, is defined as

$$\mathbb{H}(f)(t_1, \dots, t_n) = \left\langle f, \sum_{\lambda} h_\lambda t^\lambda / m(\lambda)! \right\rangle,$$

where the sum is over all partitions λ and if $\lambda = 1^{\mathbf{m}_1} \cdots k^{\mathbf{m}_k}$ then $t^\lambda = t_1^{\mathbf{m}_1} \cdots t_k^{\mathbf{m}_k}$ and $\mathbf{m}(\lambda)! = \mathbf{m}_1! \mathbf{m}_2! \cdots \mathbf{m}_k!$.

Observe that the generating function for k-regular graphs is

$$G_k(t) = \mathbb{H}(G)(0, \dots, 0, t, 0, \dots)$$

where the t occurs in position k. This is true for any generating function which takes the form $\langle f, H^{(k)}(t) \rangle$, for some f.

The H-series theorem from [37] is useful here. It states that $\mathbb{H}(\partial_{p_n} \cdot f)$ and $\mathbb{H}(p_n \cdot f)$ can be expressed as polynomials in the $\partial_{t_i} \mathbb{H}(f)$'s. In terms of Gröbner bases, this corresponds to introducing the additional variables $t_1, \dots, t_k$ (instead of $t = t_k$ alone) and working with the generating series $\mathcal{H}_k(t_1, \dots, t_k)$ of the $h_\lambda z_\lambda^{-1}$ over partitions whose largest part is k, instead of the univariate $H^{(k)}(t)$. The H-series theorem therefore implies that for an appropriate monomial order, there is a Gröbner basis of the set $I_{\mathcal{H}_k}$ of all operators of $\mathcal{A}_{p,t}$ annihilating $\mathcal{H}_k$, with elements of the form

$$p_i - P_i(\mathbf{t}, \mathbf{d_t}), \quad \partial_{p_i} - Q_i(\mathbf{t}, \mathbf{d_t}), \qquad i = 1, \dots, k. \tag{5.5}$$

The modified algorithm is shown in Algorithm 2.

After Step (3), all the p_i's and ∂_{p_i}'s have been eliminated and R_0 contains a set of generators of a D-finite $\mathcal{A}_t(t)$-ideal annihilating $\langle f, \mathcal{H}_k \rangle$. Then, in order to obtain differential equations satisfied by the specialization at $t_1 = \cdots = t_{k-1} = 0$, Step (4) proceeds in order by eliminating differentiation with respect to t_i and then setting $t_i = 0$ in the remaining operators.

---

[1] also referred to as the *Gamma series* or the *H-series*.

ALGORITHM: HAMMOND
INPUT: *An integer k, and $f \in \mathbb{K}[\![p_1, \ldots, p_n]\!]$.*
OUTPUT: *A differential equation satisfied by $\mathbb{H}(f)(0, \ldots, 0, t)$ where t is in position k*

1. Compute $\mathcal{G}_f$, a Gröbner basis for the left ideal J_f annihilating f in $\mathcal{A}_p$;
2. Compute $\mathcal{G}_{\mathcal{H}_k}$, a Gröbner basis of the form (5.5);
3. For each $U \in \mathcal{G}_f$, compute $r_U \in \mathcal{A}_t$ as the reduction of $U^\perp$ by $\mathcal{G}_{\mathcal{H}_k}$ for an order which eliminates p, ∂_p. Let R_0 be the set of r_U's;
4. for i from 1 to $k-1$ eliminate ∂_{p_i} from R_{i-1} and set $t_i = 0$ in the resulting polynomials; call R_i the new set;
5. Return R_{k-1}.

Algorithm 2 An algorithm to compute the Hammond series of a symmetric series

Note that the Gröbner basis of Step (2) can be precomputed for the required k's (although most of the time is actually spent in Step (4)).

In order to compute the elimination in Step (4), one should not compute a Gröbner basis for an elimination order, since this would in particular perform the unnecessary computation of a Gröbner basis of the eliminated ideal. Instead, one can modify the main loop in the Gröbner basis computation so that it stops as soon as sufficient elimination has been performed or revert to skew elimination by the non-commutative version of the division algorithm as described in [16].

This calculation is comparatively rapid since the size of the basis is greatly reduced. Further, it reduces as it progresses, on account of setting variables to 0. We can compute the case of 4-regular graphs in a second, in place of a couple of minutes using the general algorithm. The 5-regular expression requires significantly more computation time, and the memory limitations on our machines prevented us from being able to compute it.

As a variant calculation for Step (3), one could compute r_U by simply replacing each monomial of U of the form $p_1^{\alpha_1} \ldots p_n^{\alpha_n} \partial_{p_1}^{\beta_1} \ldots \partial_{p_n}^{\beta_n}$ with the product

$$Q_n^{\beta_n} \ldots Q_1^{\beta_1} P_n^{\alpha_n} \cdots P_1^{\alpha_1}.$$

5.4.1 Proof of Termination and Correctness

Termination of HAMMOND is obvious. On the other hand, the full proof of correctness requires a technical result proved in Section 5.6. Essentially, we need the result to show that $\mathcal{H}_k(t-1,\ldots,t_k)$ is D-finite, from which we can deduce correctness. The following corollary articulates a property of D-finite functions in the simple language of symmetric functions and D-finite descriptions, and is a direct consequence of Proposition 5.6 that will be proved independently.

Corollary 5.1. *Let $f \in \mathbb{K}[\![p_1,\ldots,p_n]\!]$ and $g \in \mathbb{K}[t_1,\ldots,t_k][\![p_1,\ldots,p_n]\!]$ be D-finite symmetric series with corresponding D-finite descriptions $J_f \subset \mathcal{A}_p$ and $\mathcal{I}_g \subset \mathcal{A}_{p,t}(p,t)$. Under these conditions, the vector space $\left(J_f^{\perp}\mathcal{A}_t(t) + \mathcal{I}_g\right) \cap \mathcal{A}_t(t)$ is non-trivial and contains a D-finite description of $\langle f, g\rangle$.*

Proposition 5.2. *Algorithm 2 terminates and is correct.*

Proof. First, we remark that for fixed k,

$$\mathcal{H}_k(t_1,\ldots,t_k) = \exp\left(\sum_{j=1}^{k} h_j t_j\right)$$

is a D-finite symmetric series by Theorem 1.5 since each h_j is a finite combination of $p_1, \ldots, p_j$. Thus, $f = H(f)(t_1,\ldots,t_k) = \langle \mathcal{H}_k(t_1,\ldots,t_k), f\rangle$ is a D-finite function of $t_1, \ldots, t_k$, by Theorem 2.7.

We proceed by proving the following invariant of the main loop: the set R_{i-1} generates a D-finite description of $\mathcal{H}(f)(0,\ldots,0,t_i,t_{i+1},\ldots,t_k)$. This establishes the result since it implies that R_{k-1} contains a D-finite description of $\mathcal{H}(f)(0,\ldots,0,t_k)$, in this case, a single differential equation. This is precisely what the algorithm claims to determine.

To prove the base case of this invariant, note that R_0 contains the generators of $\left(J_f^{\perp}\mathcal{A}_t(t) + \mathcal{I}_{\mathcal{H}_k}\right) \cap \mathcal{A}_t(t)$. We appeal to Corollary 5.1, to conclude that R_0 contains a D-finite description of $\mathcal{H}(f)(t_1,\ldots,t_k)$.

The general case is proven with the known result [16] that given a D-finite description of a function $f(x_1,x_2,\ldots,x_n)$, one can compute the D-finite description of the specialization $f(x_1,\ldots,x_{n-1},0)$. This can be done, for example, by first eliminating ∂_{x_n}, removing factors of x_n in the remaining polynomials, and finally, setting $x_n = 0$ in the equations. This is precisely the process outlined in Algorithm 2. ✪

5.5 The general situation of the scalar product of symmetric functions

So far, we have limited the scope of the algorithms to pairs of D-finite symmetric functions where only one of the two functions contains the variables $t_1, \ldots, t_k$. While this is sufficient in many applications, it is possible to modify SCALAR_DE in order to accommodate the t_i's in both functions and thus make the full power of Theorem 2.7 effective. While no additional ideas are to be used, the description of the algorithm is more technical.

SCALAR_DE manipulates monomials α and reduces them modulo the ideals $\mathcal{I}_f$ and $\mathcal{I}_g$ in order to determine equations of the form

$$\Big\langle f, \big(\alpha - (\alpha \operatorname{red}_{\preceq} \mathcal{J}_f^{\perp} \mathcal{A}_t(t))\big) \cdot g \Big\rangle = 0 \quad \text{and} \quad \big\langle f, \big(\alpha - (\alpha \operatorname{red}_{\preceq} \mathcal{I}_g)\big) \cdot g \big\rangle = 0, \tag{5.6}$$

where, by hypothesis, on the left, α does not involve any of the ∂_{t_i}'s. What makes the situation of SCALAR_DE and the left-hand identity in (5.6) simple is the assumption that f does not depend on t, making the action of $\mathcal{A}_t$ on $\langle f, g\rangle$ act on the right-hand argument only. The difficulty in generalizing lies in that, in general, the action of ∂_{t_i} on f may be non-trivial and must be considered in the differentiation rule for scalar products,

$$\partial_{t_i} \cdot \langle f, g\rangle = \langle \partial_{t_i} \cdot f, g\rangle + \langle f, \partial_{t_i} \cdot g\rangle\,, \tag{5.7}$$

which itself stems from the differentiation rule for usual products on the level of coefficients.

The idea is to manipulate operators in *three* sets of ∂_{t_i}'s: one which acts on the full scalar product $\langle f, g\rangle$, and one for each of its components, acting directly on the component. To facilitate the description of this situation, we denote the former by ∂_{t_i}, the one acting on the left component by ∂_{l_i}, and the one acting on the right component ∂_{r_i}. Using this notation, we wish to view Eq. (5.7) as

$$\partial_{t_i} = \partial_{l_i} + \partial_{r_i}. \tag{5.8}$$

We thus modify SCALAR_DE by enlarging the family of monomials over which we iterate, and use Eq. (5.8) to eliminate the ∂_{l_i}'s before beginning Gaussian elimination. Here, we iterate over monomials $\alpha\partial_l^{\beta}\partial_r^{\gamma}$ of the free commutative monoid $\{p, \partial_p, \partial_l, \partial_r\}^*$ with $\alpha \in \{p, \partial_p\}^*$ to examine the following generalizations of Eq. (5.6):

$$\Big\langle \big(\alpha^{\perp}\partial_t^{\beta} - (\alpha^{\perp}\partial_t^{\beta} \operatorname{red} \mathcal{G}_F)\big) \cdot F, \partial_r^{\gamma} \cdot g \Big\rangle = 0 \tag{5.9}$$

and

$$\left\langle \partial_t^\beta \cdot f, \left(\alpha\partial_t^\gamma - (\alpha\partial_t^\gamma \operatorname{red} \mathcal{G}_g)\right)\cdot g \right\rangle = 0,$$

or, in operator notation,

$$\left(\alpha^\perp\partial_l^\beta - (\alpha^\perp\partial_l^\beta \operatorname{red} \mathcal{G}_f)\right)\partial_r^\gamma \cdot \langle f, g\rangle = 0 \quad \text{and} \quad \partial_l^\beta\left(\alpha\partial_r^\gamma - (\alpha\partial_r^\gamma \operatorname{red} \mathcal{G}_g)\right)\cdot \langle f, g\rangle = 0.$$

Making use of Eq. (5.8) and applying adjunction to the first equation in Eq. (5.9), we get a linear combination of terms of the form $\partial_t^{\beta'} \cdot \langle f, \alpha' \cdot g\rangle$ with coefficients in $\mathbb{K}[t]$, where $\beta' \in \mathbb{N}^k$, and $\alpha' \in \mathcal{A}_{p,t}(t)$. The algorithm proceeds as before by performing Gaussian elimination over $\mathbb{K}(t)$ to eliminate p, ∂_p, and ∂_r. In our implementation, the monomial order $\preceq$ is DegRevLex($p < \partial_p < \partial_l < \partial_r$). The algorithm is summarized in Algorithm 3.

5.6 Termination and Correctness

The common goal of the algorithms present thus far, is to find a system of differential equations satisfied by $\langle f, g\rangle$, a task equivalent to finding a non-zero element in $\mathcal{A}_t$ which annihilates $\langle f, g\rangle$. In this section we present the proofs of correctness for Algorithms 1 and 3. This is the principal results of Theorem 5.3. Although Algorithm 1 is a specialization of Algorithm 3, parts of the proof would become artificially more involved if restricted to the simple case. We thus treat both algorithms simultaneously.

5.6.1 Sketch of the proof

The discussion at the beginning of Section 5 has illustrated how to manipulate the annihilators of f and g to determine a combination

$$P^\perp S + TQ \in \mathcal{A}_t \qquad \text{with} \qquad P \in \mathcal{I}_f^\perp, Q \in \mathcal{I}_g, S \in \mathcal{A}_p(t), T \in \mathcal{A}_{p,t}(t),$$

which annihilates $\langle f, g\rangle$. Not all of the elements in $\operatorname{ann}_{\mathcal{A}_t} \langle f, g\rangle$ are of this form, however, as the following simple example illustrates. If $f = p_1 - p_2$ and $g = p_1 + p_2/2$, then $\langle f, g\rangle = 1 - 1 = 0$ and thus $1 \in \operatorname{ann}_{\mathcal{A}_t} \langle f, g\rangle$, but 1 can not be written as a combination of the form $P^\perp S + TQ$ for these f and g. Nonetheless, we show that the annihilating elements that can be written this way form a non-trivial subideal of $\operatorname{ann}_{\mathcal{A}_t} \langle f, g\rangle$, generated by the algorithms we describe.

The adjunction properties of scalar products are naturally accommodated by tensor products. Specifically, the proof below centers around a certain $\mathcal{A}_t$-module S whose

ALGORITHM: SCALAR_DE2
INPUT: $f \in \mathbb{K}[t][\![p]\!]$ *and* $g \in \mathbb{K}[t][\![p]\!]$, *both D-finite in* p, t.
OUTPUT: *A system of differential equations satisfied by* $\langle f, g\rangle$, *which describes it as D-finite.*

1. Determine a Gröbner basis $\mathcal{G}_g$ for the left ideal $\operatorname{ann}_{\mathcal{A}_{p,t}(t)} g$ with respect to any monomial ordering $\preceq$, as well as a Gröbner basis/ $\mathcal{G}_{g^\perp}$ for the right ideal $\operatorname{ann}_{\mathcal{A}_p} g^\perp$ with respect to the same ordering;
2. $B := \{\}$;
3. Iterate through each monomial α in $p, \partial_p, \partial_l, \partial_r$ with respect to any ordering which, after setting $\partial_l = \partial_t, \partial_r = 1$ or $\partial_r = \partial_t, \partial_l = 1$, respectively, induces the ordering $\preceq$;
   (a) $\alpha_l := \alpha|_{\partial_l=\partial_t,\partial_r=1}$;
   (b) $\alpha_f := \alpha_l - (\alpha_l \operatorname{red}_\preceq \mathcal{G}_{f^\perp})$;
   (c) $\alpha_r := \alpha|_{\partial_r=\partial_t,\partial_l=1}$;
   (d) $\alpha_g := \alpha_r - (\alpha_r \operatorname{red}_\preceq \mathcal{G}_g)$;
   (e) Introduce $\alpha_f|_{\partial_l=\partial_t-\partial_r} \times \alpha|_{p=\partial_p=\partial_l=1}$ and $\alpha_g \times \alpha|_{p=\partial_p=\partial_r=1}$ into B and reduce so as to eliminate $p, \partial_p, \partial_r$;
   (f) Compute the dimension of the ideal generated by $B \cap \mathcal{A}_t(t)$. If this dimension is 0, break and output $B \cap \mathcal{A}_t(t)$.

Algorithm 3 A general algorithm for the scalar product of symmetric functions

elements are tensors, twisted by an action resembling the scalar product adjunction. For example,

$$(i^{-1}p_i \cdot u) \otimes v = (u.\partial_{p_i}) \otimes v = u \otimes (\partial_{p_i} \cdot v),$$

which corresponds to the equivalence $\langle (i^{-1}p_i) \cdot f, g \rangle = \langle f, \partial_i \cdot g \rangle$. (See also Eq. (5.10–5.13) below.) On the other hand, the ∂_{l_i} and ∂_{r_i} that are involved in the description of Algorithm 3 really are the operators $\partial_{t_i} \otimes 1$ and $1 \otimes \partial_{t_i}$ respectively, acting on S, where 1's denote identity maps.

The module S can be expressed in terms of the ideal $\mathrm{ann}_{\mathcal{A}_t}(f^{\perp} \otimes g)$, itself contained in $\mathrm{ann}_{\mathcal{A}_t} \langle f, g \rangle$. The former ideal is non-trivial and in fact, is sufficient to describe the scalar product as holonomic, a property which implies D-finiteness. We demonstrate that the algorithms calculate a Gröbner basis for $\mathrm{ann}_{\mathcal{A}_t(t)}(f^{\perp} \otimes g)$, in other words a ∂-finite description of the scalar product $\langle f, g \rangle$.

The main result is summarized by the following theorem.

Theorem 5.3 (Termination of Algorithms 1 and 3). *Suppose f and g are symmetric series subject to the conditions of Algorithm 1 (resp. Algorithm 3). Then, Algorithm 1 (resp. Algorithm 3) determines, in finite time, a Gröbner basis for a non-zero ∂-finite ideal contained in* $\mathrm{ann}_{\mathcal{A}_t(t)} \langle f, g \rangle$.

The discussion so far has not relied on the explicit description of the scalar product. Rather, remark that Algorithms 1 and 3 are essentially parameterized by the adjunction property of the scalar product of symmetric functions, and can easily be redefined and adapted to other adjunctions. It suits our needs for the proof to consider adjoints for the usual scalar product of functions, $\langle f | g \rangle := \int f(x)g(x)\,dx$. (To avoid confusion, we notationally distinguish $\langle f | g \rangle$ from $\langle f, g \rangle$.)

Indeed, guided by existing results concerning the preservation of holonomy under operations involving the usual scalar product, we link the symmetric case to the usual one with a map from one adjunction to the other. This reduction also demonstrates how algorithms analogous to Algorithms 1 and 3 for other scalar products could be shown to terminate with the correct output. (See Section 6.)

To raise this comparison to the level of intuition, we could identify $\langle f, g \rangle$ with the

integral

$$\int_{\mathbb{R}^n} \mathcal{L}\big(q \mapsto f(q_1, 2q_2, \ldots, nq_n)\big)(p)\, G(p)\, dp_1 \ldots dp_n,$$

where $\mathcal{L}$ is the modified Laplace transform $\mathcal{L}(f)(p) = \int_{\mathbb{R}^n} f(q) e^{-(p_1 q_1 + \cdots + p_n q_n)}\, dq$, and which satisfies

$$\mathcal{L}\big(q \mapsto q_i f(q)\big)(p) = -(\partial_{p_i} \circ \mathcal{L})(f)(p).$$

Notice, for example:

$$\begin{aligned}
\left\langle i^{-1} p_i \cdot f, g \right\rangle &= \int_{\mathbb{R}^n} \mathcal{L}\big(q \mapsto q_i f(q_1, \ldots, nq_n)\big)(p)\, g(p)\, dp_1 \ldots dp_n \\
&= -\int_{\mathbb{R}^n} (\partial_{p_i} \circ \mathcal{L})(f)(p)\, (\partial_{q_i} \cdot g)(p)\, dp_1 \ldots dp_n \\
&= \int_{\mathbb{R}^n} \mathcal{L}\big(q \mapsto f(q_1, \ldots, nq_n)\big)(p)\, (\partial_{q_i} \cdot g)(p)\, dp_1 \ldots dp_n \\
&= \left\langle f, \partial_{p_i} \cdot g \right\rangle .
\end{aligned} \tag{5.10}$$

Formally, we must work on the level of abstract modules, however. This avoids situations where the integral is not convergent or the Laplace transform is not defined as a function.

To prove Theorem 5.3, we show Corollary 5.7 below which states that $\operatorname{ann}_{\mathcal{A}_t}\left(f^{\perp} \otimes g\right)$ is a non-zero subideal of $\operatorname{ann}_{\mathcal{A}_t}\langle f, g\rangle$ such that $\mathcal{A}_t / \operatorname{ann}_{\mathcal{A}_t}\left(f^{\perp} \otimes g\right)$ is a holonomic module. This is done in multiple stages. First, in Section 5.6.2, we define S, the algebraic structure in which our calculations take place, and prove that it is holonomic by reducing the problem to the usual scalar product analogue, where similar results are known. This analogue is detailed in Section 5.6.3. Next, in Section 5.6.4 we express S as a quotient. Corollary 5.7 follows from this discussion. Finally, to conclude that the algorithm terminates, we relate S directly to the algorithm and prove in Section 5.6.5 that all of the generators are determined in finite time. Together, these results prove Theorem 5.3 and thus the correctness and termination of Algorithms 1 and 3.

5.6.2 The scalar product of symmetric functions

We now formally define the $\mathcal{A}_t$-module S. Begin with $U = \mathcal{A}_{p,t} \cdot f$ and $V = \mathcal{A}_{p,t} \cdot g$, two holonomic $\mathcal{A}_{p,t}$-modules. We shall denote by $U^{\perp}$ the module of adjoints of U: as $\mathbb{K}$-vector spaces, $U = U^{\perp}$, and a right $\mathcal{A}_p[t]$-action is defined on $U^{\perp}$ by $u \cdot P = P^{\perp} \cdot u$ for any $u \in U^{\perp}$ and $P \in \mathcal{A}_{p,t}$, where the last operation is taken for the left structure of U. Set S as the tensor product $U^{\perp} \otimes_{\mathcal{A}_p[t]} V$, which makes it a $\mathbb{K}[t]$-module. This has

the desirable effect of encoding the scalar product adjunction relations: for all $u \in U$ and all $v \in V$,

$$(\partial_{p_i} \cdot u) \otimes v = (u.\partial_{p_i}^{\perp}) \otimes v = (u \cdot i^{-1} p_i) \otimes v = u \otimes (i^{-1} p_i \cdot v), \tag{5.11}$$

$$(p_i \cdot u) \otimes v = (u \cdot p_i^{\perp}) \otimes v = (u.i\partial_{p_i}) \otimes v = u \otimes (i\partial_{p_i} \cdot v), \tag{5.12}$$

$$t_i \cdot (u \otimes v) = (t_i.u) \otimes v = (u \cdot t_i) \otimes v = u \otimes (t_i \cdot v). \tag{5.13}$$

To endow S with a $\mathcal{A}_t$-module structure, let ∂_{t_i} act on a pure tensor $u \otimes v$ by

$$\partial_{t_i} \cdot (u \otimes v) = (\partial_{t_i} \cdot u) \otimes v + u \otimes (\partial_{t_i} \cdot v), \tag{5.14}$$

and extend to S by $\mathbb{K}$-linearity. In other words, $\partial_{t_i} = \partial_{l_i} + \partial_{r_i}$ after defining $\partial_{l_i} = \partial_{t_i} \otimes 1$ and $\partial_{r_i} = 1 \otimes \partial_{t_i}$, where 1's are identity maps.

Armed with this definition and Theorem 5.4 (formally stated and proven independently in Section 5.6.3), we prove that S is holonomic. Theorem 5.4 is an analogous result for the usual scalar product, its corresponding adjunction, and the corresponding adjoint module $M^\star$ of a module M. It states that for holonomic M and N, $M^\star \otimes_{\mathcal{A}_p[t]} N$ is a holonomic $\mathcal{A}_t$-module under the same action (5.14) under ∂_{t_i}. We appeal to this theorem with an appropriate choice for M and N.

To determine the relationship between the two scalar products and make our choice for M and N, we compare both adjunction operations. In the symmetric case, adjunction is defined as the anti-automorphism $\perp$ which maps p_i to $i\partial_{p_i}$ and ∂_{p_i} to $i^{-1}p_i$, for all i, and the usual scalar product adjunction is defined as the anti-automorphism $\star$ which maps ∂_{p_i} to $-\partial_{p_i}$, and leaves the p_i variables unchanged. One way to connect both adjunctions is to factor $\perp$ into the composition of three algebra morphisms:

1. The automorphism τ mapping (p_i, ∂_i) to $(ip_i, i^{-1}\partial_i)$. This corresponds to the dilation which maps a function f to $p \mapsto f(p_1, 2p_2, \ldots, np_n)$;

2. The Fourier transform automorphism $\mathcal{F}$ mapping (p_i, ∂_i) to $(-\partial_i, p_i)$ introduced in Section 3.3.2. Informally speaking, this corresponds to mapping a function f to its Laplace transform $\mathcal{L}(f)$;

3. The anti-automorphism $\star$ mapping (p_i, ∂_i) to $(p_i, -\partial_i)$.

The important property to note is that each of these three maps preserves holonomy since they preserve total degree, hence are filtration-preserving bijections. A direct calculation on p_i and ∂_i verifies that $\perp = \star \circ \mathcal{F} \circ \tau$, so that the composite $\perp$ also is a holonomy-preserving linear bijection. Thus, we appeal to Theorem 5.4 with $M = U^{\mathcal{F}\circ\tau}$ and $N = V$, which are both holonomic. One concludes that

$$S = U^{\perp} \otimes_{\mathcal{A}_p[t]} V = \left(U^{\mathcal{F}\circ\tau}\right)^{\star} \otimes_{\mathcal{A}_p[t]} V = M^{\star} \otimes_{\mathcal{A}_p[t]} N \tag{5.15}$$

is a holonomic $\mathcal{A}_t$-module. After we have deduced the quotient structure of S in Section 5.6.4, this information is used to prove that $\operatorname{ann}_{\mathcal{A}_t}(f^{\perp} \otimes g)$ is non-trivial and that the quotient module $\mathcal{A}_t / \operatorname{ann}_{\mathcal{A}_t}(f^{\perp} \otimes g)$ is holonomic, a fact we use to show that the algorithms terminate.

5.6.3 Preservation of holonomy under the usual scalar product

In the previous section, we reduced the proof of the holonomy of $S = U^{\perp} \otimes_{\mathcal{A}_p[t]} V$ to an analogous result in terms of the usual scalar product, to be proven in this section: the holonomy of $T = M^{\star} \otimes_{\mathcal{A}_p[t]} N$ for holonomic modules M and N.

The following notion will be used in the proof: *the integral of a $\mathcal{A}_{p,t}$-module P* is defined as

$$\int P = \int P \, dp_1 \ldots dp_n = P \,/\, \Big(\sum_i \partial_{p_i} \cdot P\Big).$$

It is the image of composed maps: the Fourier transform $\mathcal{F}$, the inverse image π_* under the projection π from $\mathbb{K}^{n+m}$ to $\mathbb{K}^n$ defined by $\pi(p,t) = t$, and the inverse Fourier transform. Specifically we have,

$$\int P = \mathcal{F}^{-1} \pi_* \mathcal{F}(P).$$

These maps preserve holonomy (see [10, Th. 3.3.4] or [20, Th. 18.2.2 and Sec. 20.3]), so that the integral of a holonomic $\mathcal{A}_{p,t}$-module is a holonomic $\mathcal{A}_t$-module. (See also [10, Th. 3.1.8].)

The module T fits naturally in between an existing holonomy-preserving surjection from the $\mathcal{A}_t$-module $\int M \otimes_{\mathbb{K}[p,t]} N$ to the space $\langle M|N\rangle$. Factoring this map to pass through $T = M^{\star} \otimes_{\mathcal{A}_p[t]} N$ yields:

$$\int M \otimes_{\mathbb{K}[p,t]} N \xrightarrow{\phi} M^{\star} \otimes_{\mathcal{A}_p[t]} N \xrightarrow{\psi} \langle M|N\rangle \,, \tag{5.16}$$

where ψ maps $m \otimes n$ to $\langle m|n\rangle$, and ϕ is a natural $\mathcal{A}_t$-linear surjection that we are about to define in the course of the next theorem, as well as the integral module to the left of it. After proving that the first module in (5.16) is holonomic, the surjectivity of ϕ implies the holonomy of T.

Theorem 5.4 (Holonomy of $M^\star \otimes_{\mathcal{A}_p[t]} N$). *Suppose that M and N are two holonomic $\mathcal{A}_{p,t}$-modules, and define T as $M^\star \otimes_{\mathcal{A}_p[t]} N$. Then, T is a holonomic $\mathcal{A}_t$-module under the action of ∂_{t_i} given by*

$$\partial_{t_i} \cdot (m \otimes n) = (\partial_{t_i} \cdot m) \otimes n + m \otimes (\partial_{t_i} \cdot n).$$

Proof. We first focus our attention on $\int M \otimes_{\mathbb{K}[p,t]} N$ in (5.16). Consider the $\mathcal{A}_{p,t}$-module $P := M \otimes_{\mathbb{K}[p,t]} N$, with action of ∂_{p_i} defined by

$$\partial_{p_i} \cdot (m \otimes n) = (\partial_{p_i} \cdot m) \otimes n + m \otimes (\partial_{p_i} \cdot n),$$

and action of ∂_{t_i} defined similarly.

We can also write this as the inverse image $\iota^* (M \otimes_{\mathbb{K}} N)$, where ι is the map from $\mathbb{K}^{m+n}$ to $\mathbb{K}^{(n+m)+(n+m)}$ which sends (p,t) to (p,t,p,t). The advantage of the second presentation is that the holonomy of P is obtained from the holonomic closure under inverse image under embeddings (see [10, Th. 3.2.3] or [20, Sec. 15.3 and Ex. 15.4.5]) and the holonomic closure under tensor product over $\mathbb{K}$ by Cor. C.13.4.2. Therefore, $\int P$ is holonomic.

Next we define a $\mathcal{A}_t$-linear surjection to T. Define a map from $M \times N$ to T which sends (m,n) to $m \otimes n$. This map is $\mathbb{K}[p,t]$-balanced, $\mathbb{K}[p,t]$-bilinear, and surjective. By the universality of the tensor product, this induces a surjective map $\phi : M \otimes_{\mathbb{K}[p,t]} N \twoheadrightarrow T$. Consider the action of ∂_{p_i} on the tensor $m \otimes n$,

$$\begin{aligned}
\phi\big(\partial_{p_i} \cdot (m \otimes n)\big) &= \phi\big((\partial_{p_i} \cdot m) \otimes n + m \otimes (\partial_{p_i} \cdot n)\big) \\
&= (\partial_{p_i} \cdot m) \otimes n + m \otimes (\partial_{p_i} \cdot n) \\
&= m \otimes (-\partial_{p_i} \cdot n) + m \otimes (\partial_{p_i} \cdot n) = 0.
\end{aligned}$$

That is, $\sum_i \partial_{p_i} \cdot P \subset \ker \phi$, and thus ϕ also induces a well-defined surjective map from $\int P$ to T. Any good filtration of $\int P$ will induce a good filtration for T (see [10, Prop. 1.11] or [20, Lemma 7.5.1]). Thus, T is finitely generated with dimension bounded by that of $\int P$. Therefore, T is holonomic. ✪

5.6.4 The quotient structure of S

The next task is to express the $S = U^\perp \otimes_{\mathcal{A}_p[t]} V$. This requires modules over, and ideals of $\mathcal{A}_{p,t}$ rather than $\mathcal{A}_{p,t}(t)$. We therefore complete our notation and introduce the annihilators $I_f = \operatorname{ann}_{\mathcal{A}_{p,t}} f$ and $I_g = \operatorname{ann}_{\mathcal{A}_{p,t}} g$, which to be used in place of $\mathcal{I}_f = \operatorname{ann}_{\mathcal{A}_{p,t}(t)} f$ and $\mathcal{I}_g = \operatorname{ann}_{\mathcal{A}_{p,t}(t)} g$, respectively. Note that $I_f = \mathcal{I}_f \cap \mathcal{A}_{p,t}$ and $\mathcal{I}_f = \mathbb{K}(t) \otimes_{\mathbb{K}[t]} I_f$, and similarly for g. Last, although adjunction has not been defined for ∂_t, we use the notation $\mathcal{A}_{p,t}^\perp$ to denote $\mathcal{A}_{p,t}$ endowed with both structures of $\mathcal{A}_t$-module on the left and $\mathcal{A}_p[t]$-module on the right.

Proposition 5.5. *The $\mathcal{A}_t$-module $S = \mathcal{A}_{p,t} \cdot f^\perp \otimes_{\mathcal{A}_p[t]} \mathcal{A}_{p,t} \cdot g$ is isomorphic to*

$$(\mathcal{A}_{p,t}^\perp \otimes_{\mathcal{A}_p[t]} \mathcal{A}_{p,t}) / (I_f^\perp \otimes_{\mathcal{A}_p[t]} \mathcal{A}_{p,t} + \mathcal{A}_{p,t}^\perp \otimes_{\mathcal{A}_p[t]} I_g).$$

Proof. The $\mathcal{A}_t$-module S is also a module over $\mathcal{A}_{p,t}^\perp \otimes_{\mathcal{A}_{p[t]}} \mathcal{A}_{p,t}$, generated by $f^\perp \otimes g$. Consider the two exact sequences of respectively right and left $\mathcal{A}_p[t]$-modules

$$\begin{array}{ccccccccc} 0 & \longrightarrow & I_f^\perp & \xrightarrow{\rho} & \mathcal{A}_{p,t}^\perp & \xrightarrow{\alpha} & U^\perp & \longrightarrow & 0, \\ 0 & \longrightarrow & I_g & \xrightarrow{\sigma} & \mathcal{A}_{p,t} & \xrightarrow{\beta} & V & \longrightarrow & 0, \end{array}$$

where $\alpha(P) = f^\perp.P$, $\beta(Q) = Q \cdot g$, and ρ and σ are inclusions. (Here, $f = f^\perp$, but we write $f^\perp$ when viewed as an element of $U^\perp$, f when viewed as in U.) We combine them to make a third exact sequence:

$$\begin{array}{ccccccc} \ker(\alpha \otimes \beta) & \longrightarrow & \mathcal{A}_{p,t}^\perp \otimes_{\mathcal{A}_p[t]} \mathcal{A}_{p,t} & \xrightarrow{\alpha \otimes \beta} & S & \longrightarrow & 0, \\ & & P \otimes Q & \longmapsto & (f^\perp.P) \otimes (Q \cdot g) & & \end{array} \tag{5.17}$$

where, by [11, II.59, Proposition 6],

$$\ker(\alpha \otimes \beta) = \operatorname{im}(\rho \otimes 1_{\mathcal{A}_{p,t}}) + \operatorname{im}(1_{\mathcal{A}_{p,t}^\perp} \otimes \sigma) = I_f^\perp \otimes_{\mathcal{A}_p[t]} \mathcal{A}_{p,t} + \mathcal{A}_{p,t}^\perp \otimes_{\mathcal{A}_p[t]} I_g$$

as $\mathbb{K}[t]$-modules. We conclude that as $\mathbb{K}[t]$-modules, then, as $\mathcal{A}_t$-modules,

$$S \simeq (\mathcal{A}_{p,t}^\perp \otimes_{\mathcal{A}_p[t]} \mathcal{A}_{p,t}) / \ker(\alpha \otimes \beta) \simeq (\mathcal{A}_{p,t}^\perp \otimes_{\mathcal{A}_p[t]} \mathcal{A}_{p,t}) / (I_f^\perp \otimes_{\mathcal{A}_p[t]} \mathcal{A}_{p,t} + \mathcal{A}_{p,t}^\perp \otimes_{\mathcal{A}_p[t]} I_g).$$

✪

To be more explicit, note that this isomorphism maps the class of $1 \otimes 1$ in the quotient to $f^\perp \otimes g \in S$. Remark also that, as $\mathcal{A}_t$-modules,

$$\begin{aligned}\ker(\alpha \otimes \beta) &= \big\{P \otimes Q \in \mathcal{A}_{p,t}^\perp \otimes \mathcal{A}_{p,t} : (\alpha \otimes \beta)(P \otimes Q) = 0\big\} \\ &= \big\{P \otimes Q \in \mathcal{A}_{p,t}^\perp \otimes \mathcal{A}_{p,t} : (f^\perp \cdot P) \otimes (Q \cdot g) = 0\big\} \\ &= \big\{P \otimes Q \in \mathcal{A}_{p,t}^\perp \otimes \mathcal{A}_{p,t} : (P \otimes Q) \cdot (f^\perp \otimes g) = 0\big\} \\ &= \operatorname{ann}_{\mathcal{A}_{p,t}^\perp \otimes_{\mathcal{A}_p[t]} \mathcal{A}_{p,t}} (f^\perp \otimes g),\end{aligned}$$

so that we also have

$$\operatorname{ann}_{\mathcal{A}_{p,t}^\perp \otimes_{\mathcal{A}_p[t]} \mathcal{A}_{p,t}} (f^\perp \otimes g) = \ker(\alpha \otimes \beta) = I_f^\perp \otimes_{\mathcal{A}_p[t]} \mathcal{A}_{p,t} + \mathcal{A}_{p,t}^\perp \otimes_{\mathcal{A}_p[t]} I_g. \tag{5.18}$$

Proposition 5.6. *The $\mathcal{A}_t$-module $S' = \mathcal{A}_t \cdot (f^\perp \otimes g)$ is a submodule of S, isomorphic to*

$$\mathcal{A}_t' \,/\, \big((I_f^\perp \otimes_{\mathcal{A}_p[t]} \mathcal{A}_{p,t} + \mathcal{A}_{p,t}^\perp \otimes_{\mathcal{A}_p[t]} I_g) \cap \mathcal{A}_t'\big),$$

where $\mathcal{A}_t' \simeq \mathcal{A}_t$ is the smallest $\mathbb{K}$-subalgebra of $\mathcal{A}_{p,t}^\perp \otimes_{\mathcal{A}_p[t]} \mathcal{A}_{p,t}$ generated by $\mathbb{K}[t]$, $1 \otimes \partial_{t_1} + \partial_{t_1} \otimes 1$, …, $1 \otimes \partial_{t_k} + \partial_{t_k} \otimes 1$. In the simplified situation when $I_f = \partial_t \mathcal{A}_{p,t} + \mathcal{A}_t J_f$ for $J_f = \operatorname{ann}_{\mathcal{A}_p} f$, S' is isomorphic to

$$\mathcal{A}_t \,/\, \big((\mathcal{A}_t J_f^\perp + I_g) \cap \mathcal{A}_t\big).$$

We first prove this proposition, and then in the next section we discuss how to connect the above description of S' directly to the algorithm and how to apply it to show that the algorithms terminate.

Proof. The annihilator of $f^\perp \otimes g$ in $\mathcal{A}_t' \cdot (f^\perp \otimes g)$ is

$$\operatorname{ann}_{\mathcal{A}_t'} (f^\perp \otimes g) = \operatorname{ann}_{\mathcal{A}_{p,t}^\perp \otimes_{\mathcal{A}_p[t]} \mathcal{A}_{p,t}} (f^\perp \otimes g) \cap \mathcal{A}_t'.$$

In view of the action of $\mathcal{A}_t$ on S' through the isomorphism between $\mathcal{A}_t$ and $\mathcal{A}_t'$, we thus have that S' is isomorphic to

$$\mathcal{A}_t / \operatorname{ann}_{\mathcal{A}_t} (f^\perp \otimes g) \simeq \mathcal{A}_t' / \operatorname{ann}_{\mathcal{A}_t'} (f^\perp \otimes g) = \mathcal{A}_t' / \big(\operatorname{ann}_{\mathcal{A}_{p,t}^\perp \otimes_{\mathcal{A}_p[t]} \mathcal{A}_{p,t}} (f^\perp \otimes g) \cap \mathcal{A}_t'\big).$$

Owing to (5.18), this proves the general quotient expression for S' in the proposition statement.

Now, to prove the formula in the simpler case, observe that when $I_f = \partial_t \mathcal{A}_{p,t} + \mathcal{A}_t J_f$,

$$\begin{aligned} I_f^\perp \otimes_{\mathcal{A}_{p[t]}} \mathcal{A}_{p,t} &= \partial_t \mathcal{A}_{p,t}^\perp \otimes_{\mathcal{A}_{p[t]}} \mathcal{A}_{p,t} + \mathcal{A}_t J_f^\perp \otimes_{\mathcal{A}_{p[t]}} \mathcal{A}_{p,t} \\ &= \partial_t \mathcal{A}_t \otimes_{\mathbb{K}[t]} \mathcal{A}_{p,t} + \mathcal{A}_t \otimes_{\mathbb{K}[t]} \mathcal{A}_t J_f^\perp \end{aligned}$$

while $\mathcal{A}_{p,t}^\perp \otimes_{\mathcal{A}_p[t]} I_g = \mathcal{A}_t \otimes_{\mathbb{K}[t]} I_g$, whence the relation

$$\ker(\alpha \otimes \beta) = \partial_t \mathcal{A}_t \otimes_{\mathbb{K}[t]} \mathcal{A}_{p,t} + \mathcal{A}_t \otimes_{\mathbb{K}[t]} (\mathcal{A}_t J_f^\perp + I_g).$$

Since $\mathcal{A}_{p,t}^\perp \otimes_{\mathcal{A}_p[t]} \mathcal{A}_{p,t} = \mathcal{A}_t \otimes_{\mathbb{K}[t]} \mathcal{A}_{p,t}$, we have

$$\begin{aligned} S &\simeq (\mathcal{A}_t \otimes_{\mathbb{K}[t]} \mathcal{A}_{p,t}) / \ker(\alpha \otimes \beta) \\ &\simeq (\mathbb{K}[t] \otimes_{\mathbb{K}[t]} \mathcal{A}_{p,t}) / \big(\mathbb{K}[t] \otimes_{\mathbb{K}[t]} (\mathcal{A}_t J_f^\perp + I_g)\big) \\ &\simeq \mathcal{A}_{p,t} / (\mathcal{A}_t J_f^\perp + I_g). \end{aligned}$$

Following these isomorphisms, $\mathcal{A}'_t$ can be identified as the copy of $\mathcal{A}_t$ included in $\mathcal{A}_{p,t}$ in the last quotient above. Therefore, the submodule S' of S is isomorphic to the quotient announced in the proposition statement. ✪

Corollary 5.7. *The ideal* $\operatorname{ann}_{\mathcal{A}_t}(f^\perp \otimes g)$ *is:*

1. *Isomorphic to* $(I_f^\perp \otimes_{\mathcal{A}_p[t]} \mathcal{A}_{p,t} + \mathcal{A}_{p,t}^\perp \otimes_{\mathcal{A}_p[t]} I_g) \cap \mathcal{A}'_t$ *as a* $\mathcal{A}_t$*-module;*
2. *A non-trivial ideal contained in* $\operatorname{ann}_{\mathcal{A}_t} \langle f, g \rangle$ *and such that* $\mathcal{A}_t / \operatorname{ann}_{\mathcal{A}_t}(f^\perp \otimes g) \simeq S'$ *is holonomic.*

Proof. From (5.18),

$$\begin{aligned} \operatorname{ann}_{\mathcal{A}'_t}(f^\perp \otimes g) &= \Big(\operatorname{ann}_{\mathcal{A}_{p,t}^\perp \otimes_{\mathcal{A}_p[t]} \mathcal{A}_{p,t}}(f^\perp \otimes g)\Big) \cap \mathcal{A}'_t \\ &= \Big(I_f^\perp \otimes_{\mathcal{A}_p[t]} \mathcal{A}_{p,t} + \mathcal{A}_{p,t}^\perp \otimes_{\mathcal{A}_p[t]} I_g\Big) \cap \mathcal{A}'_t \end{aligned} \tag{5.19}$$

and we have shown (1) in the corollary statement. The $\mathcal{A}_t$-module $S' \simeq \mathcal{A}_t / \operatorname{ann}_{\mathcal{A}_t}(f^\perp \otimes g)$ is a holonomic $\mathcal{A}_t$-module, as a submodule of the holonomic module S. Now as $\mathcal{A}_t$ is not holonomic, thus $\operatorname{ann}_{\mathcal{A}_t}(f^\perp \otimes g)$ must be non-trivial by a simple dimension argument. Finally, we recall that this non-trivial ideal is contained in $\operatorname{ann}_{\mathcal{A}_t} \langle f, g \rangle$, since there is a surjection from S' to $\mathcal{A}_t / \operatorname{ann}_{\mathcal{A}_t} \langle f, g \rangle$ given by $\psi : (u \otimes v) \mapsto \langle u, v \rangle$. This proves (2) in the corollary statement. ✪

5.6.5 Termination

We now link the modules S and S' to the algorithms and prove their termination. The termination of Algorithm 3 is more technical to prove than that of Algorithm 1 since ∂_{t_i} can act separately on f and g. Thus, for ease of presentation, we consider Algorithms 1 and 3 in turn, to show that they eventually generate a Gröbner basis for $\mathrm{ann}_{\mathcal{A}_t(t)}(f^\perp \otimes g)$.

Termination of SCALAR_DE

The basic idea of Algorithm 1 is to compute filtrations of $\mathcal{I}_f$ and $\mathcal{I}_g$ incrementally and to recombine them at each step. The algorithm terminates when condition (3e) in the algorithm description is satisfied. We show that the algorithm will satisfy this condition by eventually producing a Gröbner basis for $\mathrm{ann}_{\mathcal{A}_t(t)}(f^\perp \otimes g)$. This subideal describes $f^\perp \otimes g$ and $\langle f, g\rangle$ as D-finite.

Proof. (Theorem 5.3, Algorithm 1) Algorithm 1 places a constraint on f that allows us to take advantage of the simpler $\mathcal{A}_t$-structure of $U = \mathcal{A}_{p,t} \cdot f$: when each $\partial_{t_i} \cdot f$ is 0, we have $U = \mathbb{K}[t] \otimes_{\mathbb{K}} (\mathcal{A}_p \cdot f)$ and $I_f = \partial_t \mathcal{A}_{p,t} + \mathcal{A}_t J_f$. Taking the intersection with $\mathcal{A}'_t$ is then far more transparent: from the previous section, we obtain the following simplification of Eq. (5.19):

$$\mathrm{ann}_{\mathcal{A}_t}(f^\perp \otimes g) = \left(J_f^\perp \mathcal{A}_t + I_g\right) \cap \mathcal{A}_t. \tag{5.20}$$

Consider the monoid of monomials generated by $p, \partial_p, \partial_t$, ordered by the monomial order $\preceq$ specified by the algorithm, and denote by $\mathcal{V}_\alpha$ the filtration $\bigoplus_{\beta \preceq \alpha} \mathbb{K}(t)\beta$. After Step (3d) in the main loop of Algorithm 1, with α as loop index, B generates

$$L_\alpha = \left(J_f^\perp \mathcal{A}_t(t) \cap \mathcal{V}_\alpha\right) + \left(\mathcal{I}_g \cap \mathcal{V}_\alpha\right).$$

By our choice of the elimination term order, $B \cap \mathcal{A}_t(t)$ consists in generators of the intersection $L_\alpha \cap \mathcal{A}_t(t)$.

Next we show that for each β, $\left(J_f^\perp \mathcal{A}_t(t) + \mathcal{I}_g\right) \cap \mathcal{V}_\beta$ is in L_α for some α. Since $\mathcal{V}_\beta$ is finite-dimensional, so must be the intersection under consideration. Let us introduce a basis $b_1, \ldots, b_d$ of it; each b_i can be written in the form $u_i + v_i$ for $f_i \in \mathcal{I}_f^\perp \left(= J_f^\perp \mathcal{A}_t(t)\right)$

and $g_i \in \mathcal{I}_g$, so that the intersection

$$\left(J_f^{\perp}\mathcal{A}_t(t) + \mathcal{I}_g\right) \cap \mathcal{V}_\beta = \bigoplus_{i=1}^{d} \mathbb{K}(t)(f_i + g_i)$$

is a subset of

$$\sum_{i=1}^{d} \mathbb{K}(t) f_i + \sum_{i=1}^{d} \mathbb{K}(t) g_i \subset \left(\mathcal{A}_t(t) J_f^{\perp} \cap \mathcal{V}_\alpha\right) + \left(\mathcal{I}_g \cap \mathcal{V}_\alpha\right) = L_\alpha$$

provided $\alpha = \max\{\max_i \deg f_i, \max_i \deg g_i\}$.

Assume that Algorithm 1 fails to terminate on some input f and g. Since $\operatorname{ann}_{\mathcal{A}_t(t)}(f^{\perp} \otimes g)$ is finitely generated by noetherianity of $\mathcal{A}_t(t)$, we can choose a finite set of generators for it, and set β to their maximal leading monomial. Consequently, the chosen generators are in

$$\operatorname{ann}_{\mathcal{A}_t(t)}(f^{\perp} \otimes g) \cap \mathcal{V}_\beta \simeq \left(\mathcal{A}_t(t) J_f^{\perp} + \mathcal{I}_g\right) \cap \mathcal{A}_t(t) \cap \mathcal{V}_\beta.$$

By the reasoning above, the latter is a subspace of L_α for some α. When the loop index reaches α, $\operatorname{ann}_{\mathcal{A}_t(t)}(f^{\perp} \otimes g)$ is a subideal of the ideal generated in $\mathcal{A}_t(t)$ by $B \cap \mathcal{A}_t(t)$. Since, by Corollary 5.7, $\mathcal{A}_t / \operatorname{ann}_{\mathcal{A}_t}(f^{\perp} \otimes g)$ is a holonomic module, $\operatorname{ann}_{\mathcal{A}_t(t)}(f^{\perp} \otimes g)$ is of dimension 0, and condition (3e) is satisfied. The algorithm terminates, a contradiction to our assumption. ✪

A limitation of the algorithm is that we cannot predict in advance how many monomials must be tested, and hence cannot estimate the running time. Also note that the proof has used the fact that Algorithm 1 loops over α in the same order $\preceq$ as the one used for reductions.

Termination of SCALAR_DE2

The termination of Algorithm 3 can be proved similarly, but we must use greater care when treating the ∂_{t_i}.

Proof. (Theorem 5.3, Algorithm 3) Since there is no adjoint action for ∂_{t_i}, we consider occurrences of ∂_{t_i} in the left argument of the scalar product differently from those on the right side. This is modelled in S by tensoring over $\mathcal{A}_p[t]$, where ∂_t is absent and thus, $\partial_{t_i} \otimes 1$ differs from $1 \otimes \partial_{t_i}$. Both still obey the same commutation law with t_i as ∂_{t_i}. Denote the former by ∂_{l_i} and the latter by ∂_{r_i}.

Having distinguished these two cases, we rewrite several of the important elements from the previous proof using this new notation. For example,

$$\mathcal{A}_{p,t}^{\perp} \otimes_{\mathcal{A}_p[t]} \mathcal{A}_{p,t} = \mathbb{K}\langle p, t, \partial_p, \partial_l, \partial_r; [\partial_{p_i}, p_j] = [\partial_{l_i}, t_j] = [\partial_{r_i}, t_j] = \delta_{i,j}, \\ [p_i, p_j] = [p_i, t_j] = [t_i, t_j] = [\partial_{l_i}, p_j] = [\partial_{r_i}, p_j] = [\partial_{p_i}, t_j] = 0\rangle,$$

and its subalgebra $\mathcal{A}'_t$ is generated by $\mathbb{K}[t]$, $\partial_{l_1} + \partial_{r_1}, \ldots, \partial_{l_k} + \partial_{r_k}$. We can also rewrite $I_f^{\perp} \otimes_{\mathcal{A}_p[t]} \mathcal{A}_{p,t} + \mathcal{A}_{p,t}^{\perp} \otimes_{\mathcal{A}_p[t]} I_g$ in the form $I_f^{\perp}\big|_{\partial_t=\partial_l}\mathbb{K}[\partial_r] + \mathbb{K}[\partial_l] I_g\big|_{\partial_t=\partial_r}$. Recall from the algorithm description that the ordering of the monomials in $p, \partial_p, \partial_l, \partial_r$ chosen at Step (3) is subject to constraints that relate it with the ordering $\preceq$ of monomials in $p, \partial_p, \partial_t$; it is thus again denoted by $\preceq$.

Algorithm 3 actually computes with coefficients that are rational functions in t, and so with elements of $\mathcal{I}_f^{\perp}\big|_{\partial_t=\partial_l}\mathbb{K}[\partial_r] + \mathbb{K}[\partial_l]\mathcal{I}_g\big|_{\partial_t=\partial_r}$. After the algorithm has introduced (variants of) α_f and α_g at Step (3e) with loop index α, the set B contains generators of the vector space obtained after setting $\partial_l = \partial_t - \partial_r$ in

$$\left(\mathcal{I}_f^{\perp}\big|_{\partial_t=\partial_l}\mathbb{K}[\partial_r]\right) \cap \mathcal{U}_\alpha + \left(\mathbb{K}[\partial_l]\mathcal{I}_g\big|_{\partial_t=\partial_r}\right) \cap \mathcal{U}_\alpha,$$

where $\mathcal{U}_\alpha$ denotes the filtration $\bigoplus_{\beta \preceq \alpha} \mathbb{K}(t)\beta$ for α, β ranging over the monomials in the variables $p, \partial_p, \partial_r, \partial_l$. We use this fact to conclude termination.

Let $\mathcal{V}'_\beta$ be the image of the $\mathcal{V}_\beta$ of the previous section, under the same transformation which takes $\mathcal{A}_t(t)$ to $\mathcal{A}'_t(t)$, that is,

$$\mathcal{V}'_\beta = \bigoplus_{p^a \partial_p^b \partial_t^c \preceq \beta} \mathbb{K}(t) p^a \partial_p^b \left(\partial_l + \partial_r\right)^c.$$

For each β, there is β' such that $\mathcal{V}'_\beta \subset \mathcal{U}_{\beta'}$. By noetherianity of $\mathcal{A}_t(t)$, we have that $\mathrm{ann}_{\mathcal{A}_t(t)}(f^{\perp} \otimes g)$ is finitely generated. Choose a finite set of generators and set β to their maximal leading monomial. The generators are thus contained in $\mathrm{ann}_{\mathcal{A}_t(t)}(f^{\perp} \otimes g) \cap \mathcal{V}_\beta$, which is isomorphic to $\mathrm{ann}_{\mathcal{A}'_t(t)}(f^{\perp} \otimes g) \cap \mathcal{V}'_\beta$, itself a subset of $\mathrm{ann}_{\mathcal{A}'_t(t)}(f^{\perp} \otimes g) \cap \mathcal{U}_{\beta'}$ for some β'. By (5.19) the latter is also $\mathcal{X} \cap \mathcal{U}_{\beta'}$ for

$$\mathcal{X} = \mathcal{I}_f^{\perp} \otimes_{\mathcal{A}_p(t)} \mathcal{A}_{p,t}(t) + \mathcal{A}_{p,t}(t)^{\perp} \otimes_{\mathcal{A}_p(t)} \mathcal{I}_g.$$

The intersection $\mathcal{X} \cap \mathcal{U}_{\beta'}$ is finite-dimensional, since $\mathcal{U}_{\beta'}$ is so; suppose it has for basis $b_1, \ldots, b_d$, with each b_i of the form $b_i = f_i \otimes r_i + l_i \otimes g_i$, where $f_i \in \mathcal{I}_f^{\perp}\big|_{\partial_t=\partial_l}$, $g_i \in \mathcal{I}_g\big|_{\partial_t=\partial_r}$,

$r_i \in \mathbb{K}[\partial_r]$, and $l_i \in \mathbb{K}[\partial_l]$. Then,

$$\mathcal{X} \cap \mathcal{V}'_\beta \subset \bigoplus_{i=1}^{d} \mathbb{K}(t)(f_i r_i + l_i g_i).$$

Set $\alpha' = \max\{\max_i \deg f_i r_i, \max_i \deg l_i g_i\}$, where here deg extracts the leading monomial. This implies that the generators of the intersection $\mathcal{X} \cap \mathcal{U}_{\beta'}$ are contained in the space

$$\left(\mathcal{I}_f^{\perp}\big|_{\partial_t=\partial_l} \mathbb{K}[\partial_r]\right) \cap \mathcal{U}_{\alpha'} + \left(\mathbb{K}[\partial_l]\mathcal{I}_g\big|_{\partial_t=\partial_r}\right) \cap \mathcal{U}_{\alpha'}.$$

By our earlier loop invariant, the same generators, after setting $\partial_l = \partial_t - \partial_r$, are contained in the space spanned by B when the loop index is set to α'. Thus, it suffices to run the algorithm until $\alpha = \alpha'$ and the generators of $\mathrm{ann}_{\mathcal{A}_t}(f^{\perp} \otimes g)$ will be contained in B. At this point the termination conditions are satisfied, and the algorithm terminates. ✪

Chapter VI

Related algorithms

6.1 Computing other scalar products

The work in the previous chapter used the fact that the usual scalar product of functions, $\langle f|g\rangle = \int fg$, preserves D-finiteness. Further, the proof indirectly described an effective algorithm for its calculation; an algorithm more direct than simply the composition of effective algorithms for integration and the product. In fact, we remarked at the time that the algorithms given are easily modified to accommodate other scalar products, provided that the adjoint to multiplication satisfies a particular (natural) property.

We now state explicitly the generalization of Algorithm 1 (and by extension, Algorithm 3), to other scalar products. First, we give the preservation of D-finiteness.

Theorem 6.1 (General D-finite scalar products). *Suppose* $x = x_1, x_2, \ldots$, $y = y_1, y_2, \ldots$ *and*

$$\phi : \mathbb{K}[\![x, y]\!] \times \mathbb{K}[\![x, y]\!] \to \mathbb{K}[\![y]\!]$$

is a symmetric, bilinear form with a degree preserving $\mathbb{K}(x, y)$*-automorphism* $\#$ *as multiplication adjunction.*

1. *If* $f \in \mathbb{K}[\![x, y]\!]$ *and* $g \in \mathbb{K}[\![x, y]\!]$ *are D-finite with respect to the* x *and* y *variables, one of which requires only a finite number of the* x_i*'s, then* $\phi(f, g)$ *is D-finite with respect to the* y *variables.*

2. *Furthermore, a D-finite description of* $\phi(f, g)$ *is contained in* $\left((I_f)^{\#} + I_g\right) \cap \mathcal{A}_y$.

Proof. Begin, as before, setting $U = \mathcal{A}_{x,y} \cdot f$ and $V = \mathcal{A}_{x,y} \cdot g$, and denote by $U^{\#}$ the

adjoint module of U. Set S as the tensor product $U^{\#} \otimes_{\mathcal{A}_p[t]} V$, a $\mathbb{K}[t]$-module which encodes this scalar product.

Now, any filtration preserving $\mathbb{K}(x, y)$-automorphism can be rearranged into an action that twists the module $U^{\#}$, into a module $M^{\star}$, with M holonomic. By Theorem 5.4, we have that $S = M^{\star} \otimes_{\mathcal{A}_p[t]} N$ is holonomic.

Both of the results now follow from Corollary 5.7. ✪

Thus, we modify Algorithm 1 (or 3) by essentially doing a text substitution of $\perp$ by $\#$. The modified version of Algorithm 1, GEN_SCALAR_DE, follows.

6.1.1 Macdonald polynomials

Macdonald polynomials are a generalization of Schur functions, which incorporate two additional variable parameters, generally denoted q and t. These functions generalize the orthogonality property of Schur functions with respect to a modified scalar product. This approach can also be used to define Hall polynomials, a slightly simpler generalization which can in fact be obtained from Macdonald polynomials. The book of Macdonald [52] provides a thorough description of Macdonald and Hall polynomials, although [51] specifically develops the analogy between Schur, Hall and Macdonald polynomials using modified scalar products.

The scalar products in each of these three cases preserve D-finiteness, as they satisfy the conditions of Theorem 6.1, as we shall see.

The first, which can be used to define Hall polynomials, is a symmetric, bilinear map,

$$\langle , \rangle_t : \mathbb{K}[\![p]\!] \to \mathbb{K}(t)[\![p]\!],$$

defined by the relation

$$\langle p_\lambda, p_\mu \rangle_t = \delta_{\mu\lambda} \prod \left(\frac{k}{1-t^k} \right)^{\mathbf{m}_\lambda(k)} \mathbf{m}_\lambda(k)!. \tag{6.1}$$

Denote the value of $\langle p_\lambda, p_\lambda \rangle_t$ by $z_\lambda(t)$. In this case, the adjoint $*$ to multiplication is straightforward to compute,

$$p_n^* = \frac{n}{1-t^n} \partial_{p_n}, \qquad \partial_{p_n}^* = \frac{1-t^n}{n} p_n, \qquad (p_n \partial_{p_n})^* = \partial_{p_n} p_n. \tag{6.2}$$

ALGORITHM: GEN_SCALAR_DE

INPUT: *Symmetric functions* $f \in \mathbb{K}[\![p]\!]$ *and* $g \in \mathbb{K}[t][\![p]\!]$*, both D-finite in* p, t*, respectively given by a system of linear differential operators of* $\mathcal{A}_p$ *and* $\mathcal{A}_{p,t}(t)$*.*

The adjoint function $\#$ *of a symmetric, bilinear function* $\phi : \mathbb{K}[t][\![p]\!]\mathbb{K}[t][\![p]\!] \to \mathbb{K}[\![t]\!]$ *such that* $\#$ *is degree preserving.*

OUTPUT: *A system of differential equations satisfied by* $\phi(f, g)$*, which describes it as D-finite.*

1. Determine a Gröbner basis $\mathcal{G}_g$ for the left ideal $\operatorname{ann}_{\mathcal{A}_{p,t}(t)} G$ with respect to any monomial ordering $\preceq$, as well as a Gröbner basis $\mathcal{G}_{f^\#}$ for the right ideal $\operatorname{ann}_{\mathcal{A}_p} f^\#$ with respect to the monomial ordering induced by $\preceq$ on $\mathcal{A}_p$;
2. $B := \{\}$;
3. Iterate through each monomial α in $p, \partial_p, \partial_t$ in the increasing order given by $\preceq$;
   (a) Write $\alpha = \beta\gamma$ with $\beta \in \mathcal{A}_p$ and $\gamma \in \mathbb{K}[\partial_t]$;
   (b) $\alpha_f := \big(\beta - (\beta \operatorname{red}_\preceq \mathcal{G}_{f^\#})\big)\gamma$;
   (c) $\alpha_g := \alpha - (\alpha \operatorname{red}_\preceq \mathcal{G}_g)$;
   (d) Introduce α_f and α_g as two new elements into B and reduce so as to eliminate p, ∂_p;
   (e) Compute the dimension of the ideal generated by $B \cap \mathcal{A}_t(t)$. If this dimension is 0, break and output $B \cap \mathcal{A}_t(t)$.

Algorithm 4 An algorithm for effective general scalar products

Remark that $*$ is a degree preserving bijection. Theorem 6.1 implies that a function $\langle F(p,x), G(p)\rangle_t$ is D-finite with respect to x over $\mathbb{K}(t)$.

Example. The function $\langle \mathcal{H}(x), \mathcal{H}[e_2]\rangle_t$ satisfies a differential equation in x with coefficients from $\mathbb{K}(t)$. This differential equation can be computed by Algorithm 4.

One interest of this map, apart from defining Hall polynomials, is that it yields a familiar q-specialization:

$$\left\langle f((1-q)p_1, (1-q)^2 p_2, \ldots), \sum h_n x^n \right\rangle_t \Big|_{t=q} = \mathrm{ex}_q(f).$$

A second generalized scalar product is used to compute the Mcdonald symmetric functions. It is defined on the p_λ basis by

$$\langle p_\lambda, p_\mu\rangle_{q,t} = z_\lambda(q,t),$$

with

$$z_\lambda(q,t) = \prod \left(\frac{k(1-q^k)}{1-t^k}\right)^{\mathbf{m}_\lambda(k)} \mathbf{m}_\lambda(k)!.$$

The adjoint to multiplication by p_λ is determined in a similar fashion:

$$p_n^* = \frac{n(1-q^n)}{1-t^n}\partial_{p_n}, \qquad \partial_{p_n}^* = \frac{1-t^n}{n(1-q^n)} p_n, \qquad (p_n\partial_{p_n})^* = \partial_{p_n} p_n. \tag{6.3}$$

Again, this adjunction satisfies the conditions of Theorem 6.1.

Some coefficient extraction problems can be set up using the complete and monomial symmetric functions. In the latter case, we have that

$$\langle m_\lambda, h_\mu\rangle_{q,t} = \delta_{\lambda\mu} \prod_{\lambda_i\in\lambda} \frac{(1-q^{\lambda_i})}{(1-t^{\lambda_i})}.$$

6.2 MacMahon symmetric functions

The scalar product defined for the MacMahon symmetric functions in Section 2.7 is another natural candidate for this approach. Let $\lambda = \{(a_i, b_i)^{\mathbf{m}_i}\}$, and $\mu = \{(c_i, d_i)^{n_i}\}$. Then the scalar product satisfies

$$\langle p_\lambda, p_\mu\rangle = \delta_{\lambda,\mu} \prod_{(a_i,b_i)} \mathbf{m}_i! \left(\frac{a_i! b_i!}{(a_i+b_i-1)!}\right)^{\mathbf{m}_i},$$

where $\mathbf{m}_i$ is the number of occurrences of (a_i, b_i) in λ.

The adjoint of multiplication by $p_{(a,b)}$, is given by

$$p_{(a,b)}^{\perp} = \left(\frac{a_i! b_j!}{(a_i + b_j - 1)!} \right) \partial_{p_{(a,b)}}.$$

This satisfies the conditions given by Theorem 6.1, and thus we can describe an effective algorithm to compute this. We use this algorithm in Part 3 to enumerate $k \times n$-Latin rectangles amongst other applications.

6.3 Computing the Kronecker product

Recall we earlier defined the Kronecker product of symmetric functions by the action on p_λ given by

$$p_\lambda * p_\mu = \langle p_\lambda(xy), p_\mu(x) \rangle_y \,, \tag{6.4}$$

and then extended linearly. Proposition 2.6 stated that under some conditions, the Kronecker product of two D-finite symmetric functions is again D-finite. As we saw in Section 2.1.1, this product arises in many different domains such as physics and representation theory. In this section we show how to modify SCALAR_DE to give an algorithm to compute the Kronecker product of D-finite functions.

6.3.1 Existing techniques

Since this problem is of interest in a variety of different contexts, it is not surprising that different techniques have been developed to treat this problem.

Historically, the quantity of interest has been the coefficient of s_μ in $s_\lambda * s_\nu$, where λ, μ and ν are all partitions of n. This gives the multiplicity of a character of the representation. Stein and Zemach [76] note (in 1993) that computing this for $n = 16$ (denoted by them to be $S(16)$) took over 16 hours in 1954, whereas they are able to compute $S(20)$ in just under a minute on a Cray supercomputer using their SYMPACK routines. They muse that "perhaps the next 35 years will see an equivalent improvement".

To compute these values using computer algebra systems, the packages which compute the scalar product of symmetric functions, mentioned in the last chapter, also contain procedures to calculate the tensor product of symmetric functions. For example the

Maple package of Stembridge dedicated to symmetric function computations, `SF` [77], computes, the Kronecker product of two symmetric functions of small degree. His algorithm expands the symmetric function into the power sum basis and then applies (2.3) to a pairwise comparison of terms.

6.3.2 A slight modification of SCALAR_DE

The definition given for the Kronecker product in Eq. (2.3) suggests that an algorithm to compute the scalar product may be modified to compute the Kronecker product. Indeed this is the case. The basic idea is to "tag" instances of p_i with a shadow variable t_i in one of the symmetric functions. We compute the scalar product with SCALAR_DE, and the resulting function of t_i undergoes the substitution $t_i = p_i$ to return it to a symmetric function.

That is, we write Eq. (2.3) as $p_\lambda * p_\mu = \left\langle p_\lambda t^\lambda, p_\mu \right\rangle \big|_{t_i \mapsto p_i}$, and in general we have

$$f(p_1, \ldots, p_n) * g(p_1, \ldots, p_n) = \langle f(p_1(xy), \ldots, p_n(xy)), g(p_1(x), \ldots, p_n(x)) \rangle_x \,.$$

6.3.3 Solving problems using ITENSOR_DE

Many interesting problems which use this operation require an infinite number of p_n, and are thus at first glance seemingly unsuitable for direct application of our algorithms. However, applying our algorithms for several truncations of a combinatorial problem can serve as a means to generate information upon which reasonable conjectures can be formulated. For example, Eq. (6.6) below was initially conjectured after a clear pattern emerged from a sequence of appeals to Algorithm 5. For each of these, we render the problem applicable by setting most p_n's to 0. In some cases, notably symmetric series arising from plethysms, there is sufficient symmetry and structure which can be exploited to verify these guesses by applying one of Algorithm 4 to well chosen subproblems. That is, in certain cases, such as the example that follows, the Kronecker product of two functions each with an infinite number of p_n variables can be reduced to a finite number of symbolic calculations.

For example, if two symmetric series F and G can be expressed respectively in the form

$$F(p_1, p_2, \ldots) = \prod_{n \geq 1} f_n(p_n) \qquad \text{and} \qquad G(p_1, p_2, \ldots) = \prod_{n \geq 1} g_n(p_n),$$

ALGORITHM: ITENSOR_DE

INPUT: *Symmetric functions $f \in \mathbb{K}[\![p]\!]$ and $g \in \mathbb{K}[t][\![p]\!]$, both D-finite in p, t, respectively given by a system of linear differential operators of $\mathcal{A}_p$ and $\mathcal{A}_{p,t}(t)$.*

OUTPUT: *A system of differential equations satisfied by $f * g$, which describes it as D-finite.*

1. Call $\mathcal{G}$ the system defining G and set $\mathcal{G}' = \{t_1\partial_{t_1} - p_1\partial_{p_1}, \ldots, t_n\partial_{t_n} - p_n\partial_{p_n}\}$;
   (a) For each element in $\mathcal{G}$, replace p_i with $t_i p_i$, ∂_{p_i} with $t_i^{-1}\partial_{p_i}$ and add to $\mathcal{G}'$;
   (b) For each element in $\mathcal{G}$, replace p_i with $t_i p_i$, ∂_{p_i} with $p_i^{-1}\partial_{t_i}$, clear denominators, and add to $\mathcal{G}'$;
2. Follow the steps of Algorithm 1 on the input system for F and the modified system $\mathcal{G}'$ for G;
3. In the output of Algorithm 1 make the substitution $t_i = p_i$ and $\partial_{t_i} = \partial_{p_i}$ and return this value.

Algorithm 5 An algorithm for an effective Kronecker product

for functions f_n, g_n, then one can easily deduce that

$$F * G = \prod_{n\geq 1} f_n(p_n) * g_n(p_n). \tag{6.5}$$

Remark that series which arise as plethyms of the form $h[u]$ or $e[u]$, where u can be written as a sum $\sum_n u_n(p_n)$, for some functions u_n, are precisely of this form. For example, we can use this fact to compute the Kronecker product of the sum of all Schur functions

$$F(p_1, p_2, \ldots) = \sum_\lambda s_\lambda = h[p_1 + 1/2p_1^2 - 1/2p_2] = \exp\left(\sum_i \frac{p_i^2}{2i} + \frac{p_{2i-1}}{2i-1}\right),$$

and itself. Due to the patterns present, we can reduce the calculation of the entire product to two symbolic calculations. More precisely, in order to determine a system of differential equations satisfied by $G = F * F$ we consider only the even and odd cases, and set

$$f_{2n} = \exp(p_{2n}^2/4n) \quad \text{and} \quad f_{2n-1} = \exp((p_{2n-1}^2/2 + p_{2n-1})/(2n-1)).$$

All of the functions $g_{2n} = f_{2n} * f_{2n}$ are obtained from a single computation by our Algorithm 4, adapted to handle a formal parameter. This modification is of the same nature of that described in Section 9.1. Here we introduce the scalar product given by the adjunction formula $p^\diamond = n\partial$ for a *formal parameter* n from the field K. Thus computing $\exp(p^2/4n) * \exp(p^2/4n)$ with this variant algorithm results in a first-order operator in p and ∂, which, once interpreted back in terms of p_n becomes:

$$(1 - p_n^2)\frac{\partial g_n(p_n)}{\partial p_n} + p_n g_n(p_n) = 0, \qquad \text{for even } n.$$

A second calculation for $g_{2n-1} = f_{2n-1} * f_{2n-1}$ results in:

$$n(1+p_n)(1-p_n)^2\frac{\partial g_n(p_n)}{\partial p_n} - \left(1 + (n+1)p_n - np_n^2\right) g_n(p_n) = 0, \qquad \text{for odd } n.$$

These linear equations are satisfied respectively by the functions

$$g_{2n} = \left(1 - p_{2n}^2\right)^{-1/2} \qquad \text{and} g_{2n-1} = \exp\left(\frac{p_{2n-1}}{(2n-1)(1-p_{2n-1})}\right)\left(1 - p_{2n-1}^2\right)^{-1/2}.$$

Applying Eq. (6.5) above, we get the following result.

Proposition 6.2. *The Kronecker product of the sum of the Schur functions with itself is*

$$\left(\sum_\lambda s_\lambda\right) * \left(\sum_\lambda s_\lambda\right) = \exp\left(\sum_{n\geq 1} \frac{p_{2n-1}}{(2n-1)(1-p_{2n-1})}\right)\left(\prod_{n\geq 1}\left(1 - p_n^2\right)\right)^{-1/2}. \tag{6.6}$$

Part 3

Combinatorial applications

Summary of this part

This part illustrates some combinatorial problems that can be formulated as a scalar product computation. The express aim is to automatise the solutions of these problems. Included among these are: a generalization of regular graph enumeration using the theory of species; A generalization of involutions related to Young tableaux with repeated entries; MacMahon symmetric functions are used to determine enumerative results on $k \times n$-Latin squares.

Chapter VII

Coefficient extraction and generating functions

One interesting combinatorial motivation for studying scalar products of symmetric functions is the large number of enumeration problems that can be expressed as such scalar product computations. As we mentioned earlier, a typical example is the differential equation satisfied by the generating function for k-regular graphs, seen in Chapter 5. It is but one of the simplest of a family of combinatorial problems described in this chapter using *species theory*. Some of the examples in this section have appeared previously in various sources: see for example [33, 34, 36, §4.3], and [75, Ch. 7]; They illustrate how algorithms of Part 2 allow us to determine the solutions automatically, in a unified manner. Thus, classical results are obtained as output from our algorithms.

These examples are all special instances of a notion of D-finiteness for species of structures that still has to be rigourously investigated. This notion of D-finiteness is clear in the instances we consider, but a general framework[1] requires technical development which would distract us from our current purpose. In each example the essential arguments for D-finiteness are clear and to make this obvious we exhibit explicit linear differential equations satisfied by the species under consideration.

The first set of examples illustrates a systematic process for enumerating structures which can be described as sets of objects, these objects being subject to certain regularity constraints. As we shall see, these structures are encoded by symmetric series, and generating functions for some regular sub-families are extracted using scalar product computation. Section 5.3 illustrates how labelled graphs can be encoded using mono-

[1] involving the notion of virtual polynomial species, i.e. with finite support

mials. A scalar product computation

$$\sum_n g_n t^n/n! = \left\langle \exp\left(\sum_n (-1)^n (p_n^2 - p_{2n})/2n\right), \exp\left(h_k t\right)\right\rangle, \tag{7.1}$$

determines the generating series for k-regular graphs. (That is, g_n is the number of labelled k-regular graphs on n vertices.)

We explain in the next section how to set up such problems as a scalar product calculation.

7.1 Theory of species

Species theory (in the sense of [7, 44]) offers a formalism for defining and manipulating combinatorial structures which provides natural connections between combinatorial operations on structures, such as union and product; and analytic operations on corresponding encoding series, like addition and multiplication. An important connection to our work here is that the series we consider are D-finite symmetric series, and many of the natural combinatorial actions preserve D-finiteness on the level of these series. In this section we only outline a notion of D-finiteness for species. Strict conditions are not presented in view of the need for the lengthy development of theoretical tools that would be disproportionately time consuming compared to our needs.

A description of a *species of structures* F contains two ingredients. The first describes how to produce, for any given set U, a finite set $\mathsf{F}[U]$. Intuitively, $\mathsf{F}[U]$ is the set of structures of species F constructed using elements from U. The second ingredient for a species describes how structures in $\mathsf{F}[U]$ can be naturally translated into structures in $\mathsf{F}[V]$, along any explicit bijection from U to V. The strict sense of "naturally" is made precise in [7]. Practically speaking, a species could be specified using any of the traditional languages of set theory, algorithms, diagrams, or any other means that makes $\mathsf{F}[U]$ clear given U.

Basic examples of species include: the species of sets $\mathsf{E}[U] = \{U\}$; the species characteristic of sets of cardinality k are defined as

$$\mathsf{E}_k[U] = U, \text{ if } |U| = k \text{ and } \{\} \text{ otherwise;}$$

the species $\mathsf{G}[U]$ of graphs with vertex set U; and the species of permutations, $\mathsf{P}[U] = \mathbb{S}_U$.

Now, we have not explicitly given the translation rule along bijections since it is clear in these examples. This is really only necessary when it is not obvious.

7.1.1 The cycle index series

Associated to each species is a *cycle index series*, (in the sense of Pólya). This series makes automatic many notions linked to Pólya theory. In particular the enumeration of structures up to isomorphism.

Define p_n as the power sum symmetric function. To each species F we associate a symmetric series $Z_{\mathsf{F}}(p_1, p_2, \ldots)$, which is defined as follows.

Definition 7.1 ***cycle index series.*** For any species F define its *cycle index series* Z_{F} as the series in $\mathbb{C}[\![p_1, p_2, \ldots]\!]$:

$$Z_{\mathsf{F}}(p_1, p_2, \ldots) := \sum_n \sum_{\lambda \vdash n} \operatorname{fix} \mathsf{F}[\lambda] \frac{p_1^{\mathbf{m}_1} p_2^{\mathbf{m}_2} \cdots p_k^{\mathbf{m}_k}}{z_\lambda} \tag{7.2}$$

where the value of $\operatorname{fix} \mathsf{F}[\lambda]$ is the number of structures of F which remain fixed under some labelling permutation of type[2] λ, and $\mathbf{m}_k$ gives the number of parts of λ equal to k.

For example, $\operatorname{fix} \mathsf{E}[\lambda] = 1$, (since any permutation of the elements of U does not change U). Thus,

$$Z_{\mathsf{E}}(p_1, p_2, \ldots) = \sum_n \sum_{\lambda \vdash n} \frac{p_1^{\mathbf{m}_1} p_2^{\mathbf{m}_2} \cdots p_k^{\mathbf{m}_k}}{z_\lambda} = \exp\left(\sum_n p_n / n \right).$$

The cycle index series embodies the essence of Pólya Theory, and the enumeration of configurations up to isomorphism. It appears as a set version of the Frobenius characteristic of the character of a representation of the symmetric group. It turns out that this gives us a natural way to determine generating series for combinatorial families, such as in Eq. (7.1), of structures that satisfy regularity conditions. We develop this further in Section 7.1.4.

[2] A permutation of type $(1^{m_1}, 2^{m_2}, \ldots)$ has m_1 fixed points, m_2 cycles of length 2, etc.

7.1.2 Combinatorial operations

There are several combinatorial analogues to the usual operations on series, and we shall see that these operations translate well into operations on cycle index. For example, for any set U, the sum of two species, $\mathsf{F}_1 + \mathsf{F}_2$ can be defined as the disjoint union of $\mathsf{F}_1[U]$ and $\mathsf{F}_2[U]$. Correspondingly,

$$Z_{\mathsf{F}_1+\mathsf{F}_2}(p_1, p_2, \ldots) = Z_{\mathsf{F}_1}(p_1, p_2, \ldots) + Z_{\mathsf{F}_2}(p_1, p_2, \ldots). \tag{7.3}$$

Here all sums stand for disjoint union. The product of two species is defined as

$$(\mathsf{F}_1 \cdot \mathsf{F}_2)[U] := \sum_{V+W=U} \mathsf{F}_1[V] \times \mathsf{F}_2[W].$$

Correspondingly,

$$Z_{\mathsf{F}_1\cdot\mathsf{F}_2}(p_1, p_2, \ldots) = Z_{\mathsf{F}_1}(p_1, p_2, \ldots) Z_{\mathsf{F}_2}(p_1, p_2, \ldots), \tag{7.4}$$

the usual product in $\mathbb{C}[\![p_1, p_2, \ldots]\!]$.

Another useful operation is substitution. The substitution of two species $\mathsf{F}_1 \circ \mathsf{F}_2$, denoted $(\mathsf{F}_1 \circ \mathsf{F}_2)[U]$ is formally defined as

$$(\mathsf{F}_1 \circ \mathsf{F}_2)[U] := \sum_{\pi \in \mathrm{Part}[U]} \mathsf{F}_1[\pi] \times \prod_{\beta \in \pi} \mathsf{F}_2[\beta],$$

where $\mathrm{Part}[U]$ stands for the set of partitions of U. Elements of π are the blocks of the partition. Now, instead of unravelling the above definition, we explain with the aid of an example. The substitution $(\mathsf{E} \circ \mathsf{E}_2)[U]$ is equal to set of partitions of U into blocks of size 2. A typical element of $\mathsf{E} \circ \mathsf{E}_2[\{1,2,3,4\}]$ would be $\{\{1,2\},\{3,4\}\}$. Another is $\mathsf{E} \circ \mathsf{C}$, sets of cycles, which is equivalent to permutations. The effect on the cycle index series is best described using symmetric function plethysm $p_n[g]$ as defined on page 29

$$Z_{\mathsf{F}_1\circ\mathsf{F}_2}(p_1, p_2, p_3, \ldots) = Z_{\mathsf{F}_1}[Z_{\mathsf{F}_2}] = Z_{\mathsf{F}_1}(p_1[Z_{\mathsf{F}_2}], p_2[Z_{\mathsf{F}_2}], \ldots). \tag{7.5}$$

There are other kinds of combinatorial operations, and the reader is pointed towards [7] for details.

7.1.3 D-finite species

We would like to be able to obtain many examples of D-finite symmetric series from combinatorial considerations. The notion outlined here, and the subsequent discussion,

rely on some terminology from species which we shall not define, we only use it to illustrate that there are subtleties surrounding the exact details of the definition. A *polynomial species with finite support* essentially has a finite cycle index. The *derivative of a species* F, is itself a species denoted F' is formulated as

$$\mathsf{F}'[U] := \mathsf{F}[U + \{*\}],$$

where "$*$" is an element added to U. For example, for $k > 2$, $\mathsf{E}_k'[U] = \mathsf{E}_{k-1}[U]$. All the usual analytic properties of the derivative apply in the realm of species. This allows for a formal, combinatorial notion of a differential equation.

Echoing the definition of D-finite function in one variable we could simply say that a species F is D-finite if it satisfies a linear differential equation with coefficients polynomial species, but we need to consider these coefficients to be "virtual" species in the sense of [7].

To avoid this, we can also define them as follows.

Definition 7.2 ***D-finite species.*** A species F is said to be *D-finite* if and only if it satisfies an equation of the form

$$\mathsf{S}_n\mathsf{F}^{(n)} + \ldots + \mathsf{S}_0\mathsf{F} = \mathsf{R}_n\mathsf{F}^{(n)} + \ldots + \mathsf{R}_0\mathsf{F} \tag{7.6}$$

for some polynomial species (in the usual sense[3]) S_k and R_k, for $0 \leq k \leq n$. We assume a condition of non-triviality, $\mathsf{S}_k \neq \mathsf{R}_k$.

Now, one might think that this takes care of the notion of D-finiteness for species, but there is still the problem of describing explicit closure properties in this setup. Let us just say that this can be done, and instead let us exhibit the relevant equation of the form Eq. (7.6) for our examples.

Example. The species of lists is defined

$$\mathsf{L}[U] = \{(a_1, a_2, \ldots, a_n) : a_i \in U, n \in \mathbb{N}\}.$$

The species X is the more common notation for E_1. Lists satisfy the differential equation $\mathsf{L}' = \mathsf{X}\mathsf{L}' + \mathsf{L}$, thus lists are a D-finite species.

[3]Species which can be written as polynomials of *molecular species*.

Example. The species of sets E satisfies $\mathsf{E}' = \mathsf{E}$, thus E is a D-finite species. Furthermore, the species $\mathsf{G} = \mathsf{E} \circ \mathsf{E}_2$ satisfies

$$\mathsf{G}' = (\mathsf{E}' \circ \mathsf{E}_2) \cdot \mathsf{E}_2' = \mathsf{X} \cdot \mathsf{G},$$

by applying the chain rule. Thus, G is D-finite. In fact, any substitution of a polynomial species F into E is D-finite since $(\mathsf{E} \circ \mathsf{F})' = \mathsf{F}' \cdot (\mathsf{E} \circ \mathsf{F})$, and derivatives of polynomial species are again polynomial species. This is the case for some of our examples here.

Ideally, the following should hold (but it remains to prove them).

1. If F and G are D-finite species, then so are $\mathsf{F} + \mathsf{G}$, and $\mathsf{F} \cdot \mathsf{G}$, F';
2. If G is a polynomial species, (in particular, its cycle index series is a polynomial), then $\mathsf{F} \circ \mathsf{G}$ is a D-finite species;
3. If F satisfies an "algebraic equation" of species, including for instance, equations of the form $\mathsf{F} = \mathsf{X}\mathsf{P}(\mathsf{X}, \mathsf{F})$ for polynomial species P, then F is a D-finite species. (This is the case for many families of trees, for example).

Proving these results should be essentially similar to the function case, however, particular attention is required when subtraction intervenes. Essentially, we separate equations into positive and negative parts and manipulate the equations in such a way as to avoid division. These delicate details should pose no problem, however they are not treated in this discussion.

One observation that is useful, is that the set of all D-finite species F such that Z_{F} is a D-finite symmetric series is closed under the above properties and has some useful applications. One sub-family of this set is treated next. Ideally, we would like to be able to characterize all species which have D-finite generating functions as D-finite species. To do this propertly we have to invoke generalizations of derivatives of species.

We now focus our attention on the case of $\mathsf{S} = \mathsf{E} \circ \mathsf{F}$ for polynomial F. Certainly for all species of this form we have that the cycle index series $Z_{\mathsf{S}}(p_1, p_2, \ldots)$ is a D-finite symmetric series since it is the plethysm of $\mathcal{H}$ with a polynomial. This property yields some interesting applications when other results on D-finite function are reinterpreted in this context.

The *exponential generating series* of a species F is the sum $\mathsf{F}(t) = \sum_n |\mathsf{F}[n]|t^n/n!$, where $|\mathsf{F}[n]|$ is the number of structures of type F on a set of size n. The *ordinary generating function*, $\widetilde{\mathsf{F}}(t)$, is the sum $\mathsf{F}(t) = \sum_n \mathrm{Orb}(\mathsf{F}[n])t^n$, where $\mathrm{Orb}(\mathsf{F}[n])$ is the number structures of F on a set of size n distinct up to relabelling. Also recall the notation $[x^n]f(x)$ refers to the coefficient of $[x^n]$ in the expansion of $f(x)$. This definition extends likewise to monomials.

This next proposition illustrates how the combinatorial property of D-finiteness can yield a number of D-finite series. This is useful since in many cases it is easier, or preferable, to prove D-finiteness on the combinatorial level.

Proposition 7.1. *Suppose* F *is a D-finite species such that* Z_{F} *is a D-finite symmetric series and let* $p_n = x_1^n + x_2^n + \ldots$.

1. *The exponential generating function* $\mathsf{F}(t)$ *is D-finite with respect to* t.
2. *If the cycle index* $Z_{\mathsf{F}}(p_1, p_2, \ldots)$ *is D-finite with respect to the* x_i *variables, then the ordinary generating function* $\widetilde{\mathsf{F}}(t)$ *is D-finite with respect to* t.
3. *For fixed* k, *the series* $\sum_n \left([x_1^k \cdots x_n^k]Z_{\mathsf{F}}\right) t^n$ *is D-finite with respect to* t.

Proof. The first two parts are proven using two basic results about cycle index series:

$$\mathsf{F}(t) = Z_{\mathsf{F}}(p_1, p_2, \ldots)|_{p_n=\delta_{1n}t} \quad \text{and} \quad \widetilde{\mathsf{F}}(t) = Z_{\mathsf{F}}(p_1, p_2, \ldots)|_{p_n=t^n}\,.$$

Recall from the discussion on specializations in Section 2.5 that the first preserves D-finiteness for any n, while a sufficient condition on the second specialization requires that $Z_{\mathsf{F}}(p_1, p_2, \ldots)$ be D-finite with respect to the x_i-variables, (when viewed as a series in $\mathbb{C}[\![x_1, x_2, \ldots]\!]$).

The third item of the proposition is proved by remarking that $\sum_n \left([x_1^k \cdots x_n^k]Z_{\mathsf{F}}\right) t^n = \langle \mathbb{Z}_{\mathsf{F}}, \exp(th_k)\rangle$, which is D-finite by Proposition 2.7. ✪

7.1.4 Defining combinatorial families

We offer now a brief combinatorial interpretation of the specialization $Z_{\mathsf{F}}(p_1, p_2, \ldots)$. To illustrate the idea, we consider the impact on E_k. The general theory is developed as *multi-sort species* in [2], or as *symmetric species* by Bergeron [5, 6], and more generally in Pólya theory, see for example, [22].

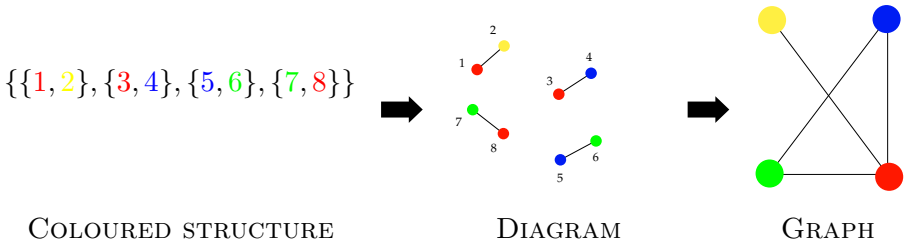


FIGURE **7.1** A correspondence between a coloured set partition and a graph

Note that

$$Z_{\mathsf{E}_k}(p_1, p_2, \ldots) = \sum_{\lambda \vdash k} p_\lambda / z_\lambda = h_k,$$

the complete homogeneous symmetric function and consequently $Z_{\mathsf{E}}(p_1, p_2, \ldots,) = \mathcal{H}$.

The combinatorial interpretation, as prescribed by Pólya Theory, is that this cycle index series considers all distinct (non-isomorphic) colourings of the elements, which we explain with the aid of E. In general, Z_{F} counts isomorphism classes of coloured F-structures. The species E_2 over the set $\{a, b\}$ is the set with one element $\{\{a, b\}\}$. All possible "colourings" of this set, say by positive integers, gives $S = \{\{i, j\} : i \leq j\}$. Observe that colouring a by i, and b by j, is considered "isomorphic to" colouring a by j and b by i. We encode each $\{i, j\}$ in S by the monomial $t_i t_j$. This includes the case when $i = j$, which we encode by t_i^2. Note, if $p_n = t_1^n + t_2^n + \ldots$, we have $Z_{\mathsf{E}_2}(p_1, p_2, \ldots) = h_2 = \sum_{i \leq j} t_i t_j$ which is precisely the sum over all coloured configurations of S.

Using the fact that $\mathsf{E} \circ \mathsf{E}_2$ corresponds to the set of pairs, one deduces from the general notion of cycle index series that $Z_{\mathsf{E} \circ \mathsf{E}_2}$ counts the isomorphism class of sets of pairs (edges) of elements with colours (vertices). This correspondence is illustrated in Figure 7.1. These isomorphism classes can be bijectively encoded as multigraphs on the set of colours.

For many applications, like regular graphs, we would like to count colourings without repetition. There is a notion of series, comparable to the cycle index, which takes into account this kind of restriction: the *asymmetry index series*, denoted Γ_{F}, of a species F as introduced by Labelle [7]. The series Γ behaves analytically in much the same way as the cycle index series, notably, substitution (in almost all cases) is reflected by plethysm, etc. This compatibility of Γ with operations allows one to reduce computations of said

Γ to basic species such as given in the table in Appendix C.

In a fashion similar to the cycle index series, Γ_{F} arises through the enumeration of colourings of asymmetric F-structures. In this case,

$$\Gamma_{\mathsf{E}_k} = e_k \quad \text{and} \quad \Gamma_{\mathsf{E}} = \mathcal{E}.$$

Taking the same species $\mathsf{E} \circ \mathsf{E}_2$ as above, and using the asymmetry index series with a similar argument, we get that $\Gamma_{\mathsf{E}\circ\mathsf{E}_2} = \mathcal{E}[e_2]$ encodes simple graphs without loops on the set of colours precisely as is determined by Eq. (5.4).

This gives us a way to have direct access to monomial encodings of combinatorial objects, as symmetric functions expressed in common bases, like the power sum basis. In fact, one can show that graphs with loops are encoded by $\mathcal{E}[h_2]$, and graphs with multiple edges, but no loops are given by $\mathcal{H}[e_2]$. Appendix C presents some series in the power sum basis that can be composed to determine the encodings of different combinatorial structure. In the next section we consider sets of sets.

7.1.5 Set Covers

Using this framework we can examine other species of structures built up from smaller objects. These species will be "D-finite" and both Z_{F} and Γ_{F} will give rise to interesting combinatorial objects. To begin we treat sets of finite sets.

Definition 7.3 ***k*-cover of a set.** A collection of sets $\mathcal{B} = \{B_1, \ldots, B_k\}$ *covers* $[n] = \{1, 2, \ldots, n\}$ if $\bigcup_{i=1}^{k} B_i = [n]$. A cover is *restrictive* if all of the B_i are distinct. Here[4], a k-cover of $[n]$ is a cover in which any given element of $[n]$ occurs in exactly k subsets.

One can deduce with combinatorial reasoning that the number of distinct covers for a set of n elements is

$$\frac{1}{2}\sum_{k=0}^{n}(-1)^k \binom{n}{k} 2^{2^n-k}.$$

Devitt and Jackson [24] give a generating function for the number of k-covers of $[n]$ by r subsets, a notion introduced in [18]. Further, they prove that the number of arithmetic operations required to actually calculate the number of k-covers of an n set by their

[4]Some sources use the term k-cover to refer to the covers with exactly k subsets.

method is bounded by $cn^k \log n$. Results for fixed k, specifically $k = 2, 3$ were treated in [18] and [4] respectively.

We can derive direct enumeration results in a similar manner using HAMMOND.

A j-set is a set of cardinality j. Remark that a k-regular graph on n vertices is a restrictive k-cover of $[n]$ into 2-sets. In general, calculating the generating function for restrictive k-covers of $[n]$ into j-sets can be expressed as

$$\left\langle \Gamma_{\mathsf{E}\circ\mathsf{E}_j}(p_1, p_2, \ldots), \sum_n h_k^n t^n \right\rangle = \left\langle \mathcal{E}[e_j], \sum_n h_k^n t^n \right\rangle.$$

To determine k-covers with mixed-cardinality sets, say both i and j, we calculate $\left\langle \Gamma_{\mathsf{E}\circ(\mathsf{E}_i+\mathsf{E}_j)}(p_1, p_2, \ldots), \sum_n h_k^n t^n \right\rangle = \langle \mathcal{E}[e_i + e_j], \sum_n h_k^n t^n \rangle$.

This yields the following simple consequence of Theorem 7.1.

Corollary 7.2. *Let S be a finite set of integers. For fixed n, the generating function for k-covers of sets by sets with cardinality an integer from S is D-finite.*

For example, we express the problem of counting distinct restrictive 2-covers of a set of cardinality n by sets of cardinality less than 5 as a scalar product. Denote the generating function of such set covers, by $S(t)$. We have,

$$S(t) = \left\langle \mathcal{E}[e_1 + e_2 + e_3 + e_4], \sum_n h_k^n t^n \right\rangle. \tag{7.7}$$

This problem is perfectly suited to HAMMOND. We can determine this differential equation, and the initial terms of the counting sequence:

$$1, 0, 1, 8, 80, 1037, 17200, 350682, 8544641, 243758420, 8010360039.$$

Cycle covers

We modify this notion slightly to consider another related problem, well suited to this paradigm. Define a *restrictive cycle cover* as a covering of $[n]$ by distinct cycles. Again, it is k-regular if every element occurs in exactly k cycles. A 3-regular cycle cover of [5], for example, is $\{(135)(2453)(14)(1254)(23)\}$. Notice that this is distinct from the cover $\{(153)(2534)(14)(1542)(23)\}$.

Remark, when $k = 1$ the total number of 1-regular cycle covers is simply the number of permutations. In the case of a restrictive cycle cover, this limits the size of each cycle in the permutation. For example, the number of permutations in which each cycle is of length less than 3 is

$$\left\langle \Gamma_{\mathsf{E}\circ(\mathsf{C}_1+\mathsf{C}_2+\mathsf{C}_3)}(p_1, p_1, \ldots), \sum_n h_1^n t^n \right\rangle = \left\langle \mathcal{E}[p_1 + p_2^2/2 + p_1^3/3], \exp(p_1 t) \right\rangle .$$

This is counted by the sequence

$$1, 1, 2, 6, 18, 66, 276, 1212, 5916, 31068, 171576, 1014696.$$

7.1.6 Parameterized solutions

Capitalizing on the symbolic nature of the algorithms, we can add additional formal variables to determine solutions of parameterized problems, or, rather, problems with "weighted" parameters.

For example, we can use parameters to describe objects which are, in some sense, between two objects. For example, $\mathcal{E}[e_2]$ encodes simple graphs whereas $\mathcal{E}[h_2]$ encodes graphs with loops. The series $\mathcal{E}[ah_2 + (1-a)e_2]$ uses a variable a, which when between 0 and 1 gives a continuous interpolation from the series encoding simple graphs to the index series encoding graphs allowing loops. We can use the scalar product algorithm to determine the differential equation satisfied by two-regular graphs with this extra parameter governing the "simplicity" of the graph. That is, when the parameter is set to 1 we have count graphs with loops. When it is set to 0 we have the counting sequence for graphs without loops. For other values, in particular for values between 0 and 1, this can be viewed as a random variable, though an explicit combinatorial description is less clear.

The function $G_a(t) = \langle \mathcal{E}\left[ah_2 + (1-a)e_2\right], \sum h_2^n t^n \rangle$ satisfies the differential equation

$$\left(-2\,a - t^2 + 2\,at\right) G_a\left(t\right) + \left(-2\,t + 2\right) \frac{d}{dt} G_a\left(t\right) = 0$$

(determined by SCALAR_DE or HAMMOND), has solution

$$G_a(t) = \frac{e^{(-1/4\,t + a - 1/2)t}}{\sqrt{t-1}}.$$

The initial terms in the counting sequence are

$$1, a, a^2, 1 + a^3, 4\,a + a^4 + 3, 10\,a^2 + 15\,a + a^5 + 12.$$

1	1	1	2	3
2	2	3	3	4
4	4	5		
5	5			

FIGURE **7.2** A 3-uniform Young tableau

7.2 Generalized involutions and regular tableaux

Another family of combinatorial objects whose generating function can be resolved with this method is a certain class of Young tableaux, namely k-uniform Young tableaux. A *Young tableaux* is a Young diagram with the entries of the array filled with positive integers. A standard Young tableaux of shape $\lambda \vdash n$ satisfies the condition that the integers from 1 to n fill the boxes in a manner which is strictly increasing from top to bottom and weakly increasing along the rows from left to right.

Standard Young tableaux are in direct correspondence with many different combinatorial objects. For example, Stanley [75] has studied the link between standard tableaux and paths in Young's lattice, the lattice of partitions ordered by inclusion of diagrams. This link was then generalized to tableaux with repeated entries (see [35]). Gessel remarks that such paths have arisen in the work of Sundaram and the combinatorics of representations of symplectic groups [78].

The *weight* of a tableau is $\mu = (\mu_1, \ldots, \mu_k)$ where μ_1 is the number of 1's, μ_2 is the number of 2's, etc. in the tableau entries. Here we consider Young tableaux where each entry appears k times that is, tableaux with weight $\mu = (k, k, \cdots, k)$, which are column strictly increasing and row weakly increasing. Such a tableau will be referred to here as *k-uniform*. Figure 7.2 illustrates a 3-uniform tableau. Two observations from [52] are essential. First, $[x_1^{\mu_1} \cdots x_k^{\mu_k}]s_\lambda$ is the number of (column strictly increasing, row weakly increasing) tableaux with weight μ. Secondly,

$$\sum_\lambda s_\lambda = \mathcal{H}\left[e_1 + e_2\right] = \exp\left(\sum_i p_i^2/2i + \sum_{i \text{ odd}} p_i/i\right),$$

which is D-finite.

Define $y_n^{(k)}$ to be the number of k-uniform tableaux of size kn, and let Y_k be the gener-

ating series of these numbers:

$$Y_k(t) = \sum_n y_n^{(k)} t^k.$$

The previous two observations imply

$$Y_k(t) = \left\langle \exp\left(\sum_{i=1}^{k} p_i^2/2i + \sum_{i \text{ odd}}^{k} p_i/i \right), \sum_n h_{k^n} t^n \right\rangle.$$

This problem is well suited to our methods since again we treat an exponential of a polynomial in the p_i's.

Calculating the equations for $k = 1, 2, 3, 4$ is rapid with either Algorithm 1 or Algorithm 2. The case $k = 5$ exhausted the memory on the machine after two weeks of enthusiastic calculation. The resulting differential equations are listed in Table A.3. For $k = 1, 2$ these results accord with known results, for example, [39, 75], and are the entries A000085 and A000985 respectively in the Sloane encyclopedia of integer sequences [72]. The first few values of $y_n^{(k)}$ are summarized in Appendix B. For $k = 3, 4$ these appear to be new.

7.3 Orthogonal polynomials

Next we consider some problems that are not of "regular structure"-format. Orthogonal polynomials can be described as solutions of differential equations, making them ideal candidates for manipulation in the context of holonomy.

The *associated Laguerre polynomials*, $L_n^{(k)}(x)$ satisfy a differential equation, and many recurrence properties, see Andrews [1] as a reference. When these polynomials are evaluated for certain choices of x the value $L_n^{(k)}(x)$ has a combinatorial description.

A sequence of integers from 1 to n is said to have *increasing support* if it contains $1, 2, \ldots, n$ as a (not necessarily consecutive) subsequence. Thus, 1213 has increasing support while 1312 does not. Goulden and Jackson [36] observed that the number $I_\lambda(n)$ of sequences $a_1, a_2, a_3, \ldots$ with increasing support whose elements form the multiset $\{a_1, a_2, a_3, \ldots\} = \{1^{\lambda_1+1} 2^{\lambda_2+1} \cdots n^{\lambda_n+1}\}$ can be expressed as a scalar product,

$$I_\lambda(n) = \left\langle h_\lambda, (1-p_1)^{-(n+1)} \exp\left(\sum_k (-1)^k \frac{p_k}{k(1-p_1)^k} \right) \right\rangle. \tag{7.8}$$

Gessel deduces from the expression (7.8) that $I_\lambda(n) = n! L_n^{(k)}(1)$ when $\lambda = 1^k$. That is, he counts the number of sequences with increasing support of the multi-set

$$\{1^2, \ldots, k^2, (k+1), \ldots, n\}.$$

It is possible to describe the generating function of $L_n^{(k)}(1)$ using generating functions of this expression.

Proposition 7.3. *The generating function of the associated Laguerre polynomials evaluated at 1, $\mathcal{L}(s,t) := \sum_{n,k} L_n^{(k)}(1) s^k t^n$ is D-finite in the s and t variables.*

Proof. The generating series can be expressed as a scalar product:

$$\begin{aligned}\mathcal{L}(s,t) &= \left\langle \sum_n h_1^n s^n/n!, \sum_n t^n (1-p_1)^{-(n+1)} \exp\left(\sum_k (-1)^k \frac{p_k}{k(1-p_1)^k}\right)\right\rangle \\ &= \left\langle \exp(p_1 s), \exp\left(\frac{-p_1 x}{1-p_1}\right)\frac{1}{1-p_1-t}\right\rangle.\end{aligned}$$

Remark that both input functions are clearly D-finite and involve only p_1, thus by Theorem 2.7, the function $\mathcal{L}(s,t)$ is D-finite with respect to t and s. ✪

This calculation is an ideal candidate for SCALAR_DE2. We can also calculate results for other sequences as well. Consider $\lambda = 1^r 2^s$. The generating function for this is given by:

$$I_{1^r 2^s}(n) = \left\langle \exp\left(t(h_1+h_2)\right), (1-p_1)^{-(n+1)} \exp\left(\frac{-p_1}{1-p_1} + \frac{p_2}{2(1-p_1)^2}\right)\right\rangle. \tag{7.9}$$

This is also D-finite, however, more computationally complex.

We can take a different approach to determine some other interesting facts. We calculate the diagonal of $\mathcal{L}(s,t)$ which is also D-finite. This calculation can profit from an explicit description of $L_n^{(n)}$. Set

$$l(n) := L_n^{(n)} = \sum_{j=0}^{n} \frac{1}{j!}\binom{2n}{n-j}(-x)^j.$$

The coefficient of the diagonal $\sum_n l(n) s^n t^n$, be determined using the summation methods described in the Ore Algebra chapter. We determine automatically a recurrence satisfied by the $l(n)$ is

$$n l(n+2) + (-n^3 - 7n^2 - 9n - 2) l(n+1) + 2(2n+1)(n+1)^3 l(n) = 0$$

The initial terms of the sequence are:

$$1, 1, 5, 47, 641, 11389, 248749, 6439075.$$

7.4 Applications of MacMahon symmetric functions

This section details some combinatorial problems that can be expressed as a scalar product of two MacMahon symmetric functions, as described in Section 2.7. These examples all use D-finite MacMahon symmetric functions, and hence the algorithms of the previous part apply.

7.4.1 Latin rectangles

Latin squares, and their cousin, Latin rectangles, are classic combinatorial objects, originally introduced by Euler. MacMahon symmetric functions can be used to formulate generating functions of Latin rectangles, and to determine enumerative and asymptotic results.

Definition 7.4 ***Latin rectangle.*** A $k \times n$ *Latin rectangle* is a $k \times n$ array of integers such that each row is a permutation of $[n]$ but no element appears twice in the same column.

For example, a $2 \times n$ Latin rectangle is a derangement. Gessel sets up different combinatorial models of Latin rectangles in [31, 32, 33]. Here, we present the $2 \times n$ case to illustrate the general argument.

One can show that the number of $2 \times n$ Latin rectangles, $r_2(n)$, is equal to

$$r_2(n) = [x_1, \cdots, x_n, y_1, \cdots, y_n] \left(\sum_{i \neq j} x_i y_j \right)^n .$$

In general [31] we have that $k \times n$ are counted by $\left\langle h^n_{(1^k)}, e^n_{(1^k)} \right\rangle$. We have that $e^n_{(1,1)} = \left(\sum_{i \neq j} x_i y_j \right)^n$ and extracting $x_1, \cdots, x_n, y_1, \cdots, y_n$ is thus equivalent to the scalar product $\left\langle h^n_{(1,1)}, e^n_{(1,1)} \right\rangle$. Gessel [33] illustrates how to develop an asymptotic expression for $r_2(n)$ as n goes towards infinity using the largest terms in the development of the power.

However, this problem is also well suited to a holonomic systems approach using generating series. We convert the expressions to the power bases, with the aid of the conversion formulas from the appendices of [25]. More specifically, using $\mu(\hat{0},\sigma)$, the Möbius functions for generalization of Young's lattice to bipartite partitions we convert $h_{(1^k)}$ and $e_{(1^k)}$ to the power basis with the following formulas:

$$h_\pi = \sum_{\pi\leq\lambda} \left|\mu(\hat{0},\sigma)\right| p_\sigma \qquad \text{and} \qquad e_\pi = \sum_{\sigma\leq\pi} \mu(\hat{0},\sigma) p_\sigma. \tag{7.10}$$

The generating series of $2\times n$ Latin squares is the generating series

$$\begin{aligned} R_2(t) &= \sum_n r_n(n) t^n \\ &= \left\langle \sum_n \left(p_{(1,0)(0,1)} + p_{(1,1)}\right)^n t^n, \sum_n \left(p_{(1,0)(0,1)} - p_{(1,1)}\right)^n \right\rangle \\ &= \left\langle \frac{1}{1-\left(p_{(1,0)(0,1)} + p_{(1,1)}\right)t}, \frac{1}{1-\left(p_{(1,0)(0,1)} + p_{(1,1)}\right)t} \right\rangle. \end{aligned}$$

This calculation is well suited for Algorithm 4 modified for the scalar product of MacMahon symmetric functions.

7.4.2 Equivalence classes of words

A second example using MacMahon symmetric functions, which is amenable to these techniques is the determination of equivalence classes of words that satisfy some commutation relations. Let m be monoid freely generated by $a_1, a_2, \ldots, a_n,\ b_1, b_2, \ldots, b_n$ subject to the commutation relation $a_i b_i = b_i a_i$. Classic theory of Cartier-Foata implies that the number of equivalence classes of words in the monoid with α_i occurrences of a_i and β_i occurrences of b_i is equal to the coefficient:

$$\left[\mathbf{u}^\alpha \mathbf{v}^\beta\right] \left(1 - x_1 - x_2 - \ldots - x_n - y_1 - y_2 - \ldots - y_n + x_1 y_1 + \ldots x_n y_n\right)^{-1}.$$

Since $1 - x_1 - x_2 - \ldots - x_n - y_1 - y_2 - \ldots - y_n + x_1 y_1 + \ldots x_n y_n = 1 - p_{(0,1)} - p_{(1,0)} + p_{(1,1)}$, this is equivalent to determining the scalar product

$$\left\langle h_\pi, \left(1 - p_{(0,1)} - p_{(1,0)} + p_{(1,1)}\right)^{-1} \right\rangle.$$

Thus, for families of π we can determine generating functions of these words. Consider for example the sequence of bi-partitions $(\pi_n)_n = ((1,1)^n)_n$. In this case we have $\sum_n h^n_{(1,1)} t^n/n! = \exp\left(p_{(1,0)(0,1)} + p_{(1,1)}\right)$.

Conclusion

Many enumerative problems of combinatorics can be phrased as a scalar product calculation. The main contribution of this thesis is the presentation of a general framework from which we can derive a collection of new algorithms which compute a differential system of equations satisfied by scalar products. Conceptually these algorithms take their inspiration from effective integration of D-finite functions, and they rely on properties of holonomic systems for their proof of correctness and termination.

Equally important is the revelation that a number of diverse problems and families of problems have solutions which reduce to a scalar product calculation, all amenable to the same algorithm, and an automatic solution. Our algorithms can be tailored to different kinds of symmetric functions, such as the MacMahon symmetric function or various q-parameterized problems. Thus, we can include amongst our examples several classical problems dating to MacMahon, such as latin rectangle enumeration.

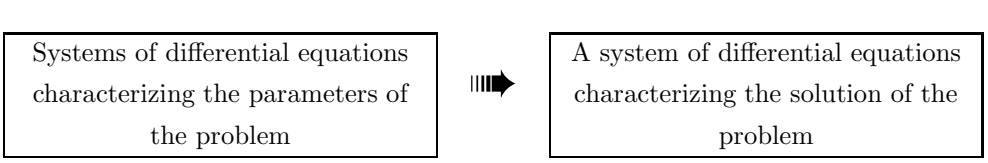
We thus contribute to the growing body of automatic combinatorics, which lies at the confluence of combinatorics and symbolic algebra, and whose purpose is to yield automatic results, for example in enumeration, asymptotics, or identity proving. Here we offer techniques to automatically calculate, directly from a combinatorial description, differential operators for sub-families of objects subject to certain regularity constraints.

All of the algorithms described are implemented in Maple and are available for public distribution. They manipulate differential equations using Gröbner basis calculations in a Weyl Algebra setting.

Several directions for future work have become apparent in the course of this study.

Future applications

Several enumeration problems, untreated here, fall into our general setup:



These span a large spectrum from the study of series arising in the enumeration of bounded height tableaux, to examples coming from the study of free Lie algebras.

Future research

The algorithms presented here determine generators for a D-finite sub-ideal of the annihilators of a series in some operator ring. A solution determining generators for the whole annihilating ideal would have wide ranging implications including the problem of effective integration and determination of solution spaces of certain differential equations. The general problem is stated as follows.

> *For some left $\mathcal{A}_{x,t}$-ideal $\mathcal{I}$ and some right $\mathcal{A}_{x,t}$ − ideal $\mathcal{J}$ determine the generators of $(\mathcal{I}+\mathcal{J})\cap\mathcal{A}_x$.*

An approach consisting of a different filtration using the weighted bases of [69] seems promising.

The asymptotic analysis of differential equations is yet another interesting direction to follow. In particular, a followup project could consist of meshing existing asymptotic tools with the output of our algorithms here.

Our work on ∂-finite preserving q-specializations represents just a glimpse of a potentially rich study. It should be possible to characterize symmetric series which yield ∂-finite q-series for certain specializations.

Finally, we outline a potential definition of D-finite species which could be a useful starting point for characterizing combinatorial structures with D-finite generating series. In particular an analysis of combinatorial equations defining tree-like structures is in order. Such a study could also be useful for treating some longstanding open problems, such as the D-finiteness of generating functions of k-regular graphs with a specified set of forbidden subgraphs. Wormald [84] determined that the generating function of 3-regular graphs without triangles is D-finite. Gessel notes [34] that determining the

D-finiteness of generating functions for permutations with forbidden subsequences is a difficult problem.

Appendix A

Differential Equations

The data in the section are presented in the following format. The differential equation satisfied by the exponential generating series $f(t) = \sum f_n \frac{t^n}{n!}$ of an object parameterized by k is

$$\phi_0 f(t) + \phi_1 f'(t) + \phi_2 f''(t) = 0.$$

$k = 2$	ϕ_0	$-t^2$
	ϕ_1	$-2t + 2$
	ϕ_2	0
$k = 3$	ϕ_0	$t^3(2t^2 + t^4 - 2)^2$
	ϕ_1	$-3(t^{10} + 6t^8 + 3t^6 - 6t^4 - 26t^2 + 8)$
	ϕ_2	$-9t^3(2t^2 + t^4 - 2)$
$k = 4$	ϕ_0	$-t^4(t^5 + 2t^4 + 2t^2 + 8t - 4)^2$
	ϕ_1	$-4(t^{13} + 4t^{12} - 16t^{10} - 10t^9 - 36t^8 - 220t^7 - 348t^6$ $-48t^5 + 200t^4 - 336t^3 - 240t^2 + 416t - 96)$
	ϕ_2	$16t^2(t-1)^2(t^5 + 2t^4 + 2t^2 + 8t - 4)(t + 2)^2$

Table **A.1** Differential equations: k-regular (simple) graphs

$k=1$	ϕ_0	t
	ϕ_1	-1
$k=2$	ϕ_0	$\left(t^2-2\,t\right)$
	ϕ_1	$(-2\,t+2)$
$k=3$	ϕ_0	$\left(t^{11}-2\,t^9+44\,t^3+16\,t^5-24\,t\right)$
	ϕ_1	$\left(-3\,t^{10}-3\,t^6-6\,t^8-54\,t^2+24\right)$
	ϕ_2	$\left(9\,t^7-18\,t^3\right)$

TABLE **A.2** Differential equations: k-regular multi-graphs

	ϕ_0	$-(t-1)$
$k=1$	ϕ_1	1
	ϕ_2	0
	ϕ_0	$t^2(t-2)$
$k=2$	ϕ_1	$-2(t-1)^2$
	ϕ_2	0
	ϕ_0	$(t^{11}+t^{10}-6t^9-4t^8+11t^7-15t^6+8t^5-2t^3+12t^2-24t-24)$
$k=3$	ϕ_1	$-3t(t^{10}-2t^8+2t^6-6t^5+8t^4+2t^3+8t^2+16t-8)$
	ϕ_2	$9t^3(-t^2-2+t+t^4)$
	ϕ_0	$\delta(t)$
$k=4$	ϕ_1	$-4\gamma(t)$
	ϕ_2	$16t^2(t-2)(t+1)^2\beta(t)$
	ϕ_3	$-64t^4(t-2)^2(t+1)^4\alpha(t)$

TABLE **A.3** Differential equations: Tableaux of weight k^n, $k=1..4$

(note: $\alpha(t),\beta(t),\gamma(t),\delta(t)$ are given in the next table)

$\alpha(t)$	$t^{14} - t^{13} - 5t^{12} - 7t^{11} + 6t^{10} + 35t^9 + 39t^7 - 50t^6 - 162t^5 - 92t^4$
	$+228t^3 + 424t^2 + 248t + 48$
$\beta(t)$	$t^{29} - 3t^{28} - 16t^{27} + 24t^{26} + 147t^{25} + 14t^{24} - 770t^{23} - 666t^{22} + 1416t^{21}$
	$+3567t^{20} - 916t^{19} - 16598t^{18} + 17766t^{17} + 40678t^{16} - 102556t^{15} - 53272t^{14}$
	$+390656t^{13} + 364080t^{12} - 707936t^{11} - 1406336t^{10} - 552544t^9$
	$+1397664t^8 + 2020864t^7 + 176256t^6 - 916864t^5 + 304896t^4 + 1283328t^3$
	$+877056t^2 + 253440t + 27648$
$\gamma(t)$	$t^{28} - t^{27} - 14t^{26} - 20t^{25} + 111t^{24} + 278t^{23} - 196t^{22} - 1216t^{21} - 1384t^{20} + 2765t^{19}$
	$+3170t^{18} - 3400t^{17} + 12140t^{16} + 15588t^{15} - 70280t^{14} - 108946t^{13} + 121796t^{12}$
	$+349056t^{11} + 116992t^{10} - 481704t^9 - 706320t^8 + 3040t^7 + 581184t^6 + 158688t^5$
	$-297408t^4 - 173952t^3 + 22272t^2 + 35712t + 6912$
$\delta(t)$	$2t^{21} - 3t^{20} - 17t^{19} - 2t^{18} + 74t^{17} + 105t^{16} - 108t^{15} - 172t^{14} - 252t^{13} + 432t^{12}$
	$-667t^{11} + 1500t^{10} + 7336t^9 - 3772t^8 - 23056t^7 - 20584t^6 + 15504t^5 + 38160t^4$
	$+17904t^3 - 4512t^2 - 5568t - 1152$

Table **A.4** Polynomials related to the differential equation satisfied by $Y_4(t)$

Appendix B

Counting sequences

The sequences here were all generated using the differential equations in the previous appendix. The EIS number accompanying the sequences refers to its entry in the *Sloane encyclopedia of integer sequences* [72].

Table **B.1** Counting sequence: k-regular graphs

k	EIS	
2	A001205	1, 0, 0, 1, 3, 12, 70, 465, 3507, 30016, 286884, 3026655, 34944085, 438263364, 5933502822, 86248951243, 1339751921865, 22148051088480, 388246725873208, 7193423109763089
3	A002829	1, 0, 0, 0, 1, 0, 70, 19355, 0, 11188082, 0, 11555272575, 0
4	A005815	1, 0, 0, 0, 0, 1, 15, 465, 19355, 1024380, 66462606, 5188453830, 480413921130, 52113376310985, 6551246596501035

Table **B.2** Counting sequence: k-regular multi-graphs

k	EIS	
1		1, 0, 1, 0, 3, 0, 15, 0, 105, 0, 945, 0, 10395, 0, 135135, 0
2	A002137	1, 0, 1, 1, 6, 22, 130, 822, 6202, 52552, 499194, 5238370, 60222844, 752587764, 10157945044, 147267180508
3		1, 0, 1, 0, 10, 0, 760, 0, 190050, 0, 103050570, 0, 102359800620, 0, 168076482974400,
4		1, 0, 1, 1, 15, 158, 3355, 93708, 3535448, 170816680, 10307577384 759439940230, 67095584693434, 7001532238614324, 851997581131397870, 119582892039683711842

TABLE **B.3** Counting sequence: k-covers by sets of cardinality one and two

k	EIS
1	1, 1, 2, 4, 10, 26, 76, 232, 764, 2620, 9496, 35696, 140152, 568504, 2390480, 10349536
2	1, 0, 1, 4, 18, 112, 820, 6912, 66178, 708256, 8372754, 108306280, 1521077404, 23041655136, 374385141832, 6493515450688
3	1, 0, 0, 1, 10, 112, 1760, 35150, 848932, 24243520, 805036704, 30649435140, 1322299270600, 64008728200384, 3447361661136640, 205070807479444088
4	1, 0, 0, 0, 1, 26, 820, 35150, 1944530, 133948836, 11234051976, 1127512146540, 133475706272700, 18406586045919060, 2925154024273348296, 530686776655470875076

TABLE **B.4** Counting sequence: Tableaux of weight k^n

k	EIS	
1	A000085	1, 1, 2, 4, 10, 26, 76, 232, 764, 2620, 9496, 35696, 140152, 568504
2	A000985	1, 1, 3, 11, 56, 348, 2578, 22054, 213798, 2313638, 27627434
3		1, 1, 4, 23, 214, 2698, 44288, 902962, 22262244
4		1, 1, 5, 42, 641, 14751, 478711, 20758650, 1158207312

Appendix C

Some counting series of small species

Object	Series	Value	Object	Series	Value
2-sets	Γ_{E_2}	$e_2 = p_1^2/2 - p_2/2$	2-multisets	Z_{E_2}	$h_2 = p_1^2/2 + p_2/2$
3-sets	Γ_{E_3}	e_3	3-multisets	Z_{E_3}	h_3
4-sets	Γ_{E_4}	e_4	4-multisets	Z_{E_4}	h_4
k-sets	Γ_{E_k}	e_k	k-multisets	Z_{E_k}	h_k
3-cycles	Z_{C_3}	$p_1^3/3 + p_3/3$	triples	Z_{X^3}	p_1^3
4-cycles	Z_{C_4}	$p_1^4/4 + p_2^2/12 + p_4/12$	4-arrays	Z_{X^4}	p_1^4
5-cycles	Z_{C_5}	$p_1^5/5 + p_5/30$	5-arrays	Z_{X^5}	p_1^5
k-cycles	Z_{C_k}	$\sum_{cd=k} \phi(d) p_d^c / k!$	k-arrays	Z_{X^k}	p_i^k

Table **C.1** Cycle index series of small species

Appendix D

The ScalarProduct Maple package

D.1 Introduction and help pages

This chapter is a quick guide to the Maple package which provides the functions described in this thesis. It is available at `http://www.labri.fr/~mishna`.

Requirements

This package relies on a few other packages. The basic functionality does not require SF of Stembridge, however, it is useful for describing functions. The `Ore_Algebra` and `Groebner` packages are required, however. Groebner is part of Maple in versions 7 and higher. Both are part of the algolib library. Download the code and save it, for example as SP.mpl. To use the file The file can then be read into your Maple session, for Maple versions 7 and higher. To access the commands, execute the following commands.

```
> read ("SP.mpl"):
```

From here, you can either execute

```
> with(ScalarProduct):
```

and have access to all of the functions; Or access them individually, for example,

```
> ScalarProduct[itensor_de](exp(pn/n), exp(pn/n), f);
```

scalar_de - *Determines a differential equation satisfied by the scalar product of two symmetric functions*

Calling Sequence

```
scalar_de(F, G, vlist, fname, adj, adj_consts)
scalar_de(Fsys, Gsys, vlist, fname, adj, adj_conts)
```

Parameters

F, G - D-finite symmetric functions

$Fsys, Gsys$- D-finite descriptions of functions

vlist - a list of variables that survive the scalar product. The end result is a function in these variables,

$fname$ - a name to be used for the output system. If this is set to 'TRUE', the function returns its result in operator notation.

adj - (optional) adjunction to the scalar product. The default is the symmetric adjoint which sends p_n to $n\frac{d}{dp_n}$. Any named constants it contains must be passed in adj_consts.

Description

- scalar determines a system of differential equations satisfied by $\langle F, G\rangle$, the scalar product (of symmetric functions) of F and G.
- Symbolic scalar products can be calculated by using the variables pn where n is any symbol.
- For the time being, only G can be a function of the *vlist* variables. A version of the algorithm in the case where both are functions of the *vlist* variables is known, and is in the implementation phase.

Examples

```
# the calculation for the 2-regular graph generating series
>scalar_de(exp(-1/2*p2+1/2*p1^2-1/4*p2^2),exp((p2/2+p1^2)*t),[t],f(t));
```

$$[t^2 f(t) + (2t - 2)\frac{d}{dt} f(t)]$$

hammond - *Determines a differential equation satisfied by the scalar product of a function and* $\sum(h_k^n t^n)$

Calling Sequence

```
hammond(F, kmax, fname)
```

Parameters

F- A D-finite function using a finite number of p_n variables

$kmax$ - the largest n that F contains

$fname$ - a name and variable for the output function.

Description

- hammond computes the differential equation satisfied by the *Hammond Series* or *Gamma series* of F. [36, 37]. This is equivalent to the scalar product $\langle F, \sum_n h_k^n t^n \rangle$.
- This procedure uses special properties of $(\sum_n h_k^n t^n)$ to offer a potentially more efficient algorithm.

Examples

```
# A second way to calculate the differential equation satisfied by the
# generating series of 2-regular graphs
#
>hammond(exp(-1/2*p2+1/2*p1^2-1/4*p2^2), 2, f(t));
```

$$[t^2 f(t) + (2t-2)\frac{d}{dt} f(t)]$$

itensor_de - *Determines a differential equation satisfied by the Kronecker product of two symmetric functions*

Calling Sequence

```
itensor_de(F, G, fname, adj, adj_const)
itensor_de(F, Gsys, fname, adj, adj_const)
```

Parameters

F, G - D-finite symmetric functions

$Gsys$ - D-finite descriptions of the function G

$fname$ - a name to be used for the output system. If this is set to 'TRUE', the function returns its result in operator notation.

adj - (optional) adjunction to the scalar product. The default is the symmetric adjoint which sends p_n to $n\frac{d}{p_n}$.

adj_const - (optional) named constants which appear in adj.

Description

- This function determines a differential equation satisfied by the Kronecker product of symmetric functions. This product has many monikers, including the cup product, the internal product and the tensor product of symmetric functions.

- This product arises in the study of the tensor product of characters of representations of the symmetric group.
- The result can be output as a differential operator if an optional TRUE flag is added at the end of the input. The advantage of this is that the output can then be directly used as input to ITENSOR_DE or SCALAR_DE.
- For the time being, F must be given as a function, not as a system. This should be corrected in a future version.

Examples

```
> itensor_de(exp(pn/n),exp(pn/n), f);dsolve([op(%),f(0)=1], f(pn));
```

$$[f(pn) - n\frac{d}{dpn}f(pn)]$$

$$f(pn) = \exp(pn/n)$$

Auxiliary functions

- `define_system(F, vars, fon)` returns the D-finite description of F, a function of the variables vars, that is a system of differential equations that it satisfies, which contains sufficient information to prove that the input function is D-finite. The output is expressed as a function using fon.
- `truncate(f, k)` sets $p_n = 0$ in f for $n > k$.
- `diffeq_to_op(sys, f)` converts a differential system (a set or list) for the function f to operator notation.
- `seriesH(f, k)` (requires SF) the symmetric function plethysm of the series $H = \sum h_n t^n$ and f, $(H[f])$, truncated at k. (as in truncate above)
- `seriesE(f, k)` (requires SF) the symmetric function plethysm of the series $E = \sum e_n t^n$ and f, $(E[f])$, truncated at k.
- `hammond_series(k)` the series $\sum h[\lambda]t^{|\lambda|}$ truncated at k.

D.2 Sample Session

Here we illstrate some of the problems that were encountered in the earlier sections and how their solution can be determined using an implementaion of the algorithms. First we read in the code.

```
read("maple/lib/src/SF.mpl"):
read("maple/lib/src/SP.mpl");
with(ScalarProduct):
```

Graph enumeration

Given a differential equation, we can either try to solve it, or do a series expansion on the initial term. We can use tools in gfun to develop the first few terms of different counting sequences.

```
detoseq:= proc(de, f, N)
local P;
   P:=gfun[rectoproc](
         gfun[diffeqtorec]({op(de), op(0,f)(0)=1}, f, a(n)),
         a(n));
   seq(P(i)*i!, i=0..N);
end:
```

```
> graph:=n->seriesE(e2, n):
> graph(3); # some examples
```

$$\exp\left(-1/2p2 + 1/2p1^2 - 1/4p2^2 + 1/6p3^2\right)$$

```
>graph3:=hammond(graph(3), 3, f(t));
```

$$(-t^1 1 - 4t^3 - 4t^9 + 8t^5)f(t) + (18t^8 - 18t^4 + 3t^1 0 - 78t^2 + 24 + 9t^6)\frac{d}{dt}f(t) + (-18t^3 + 18t^5 + 9t^7)\frac{d^2}{dt^2}f(t)$$

```
>detoseq(graph3,f(t), 15);
```

$$1, 0, 0, 0, 1, 0, 70, 0, 19355, 0, 11180820, 0, 11555272575, 0, 19506631814670, 0$$

The following example corresponds to the problem in Section 7.1.6.

```
>sgraph:= n->seriesE(alpha*e2 + (1-alpha)*h2, n):
>sgraph3:=hammond(sgraph(3),3, f(t), {alpha});
```

$$\left(60\,t^5\alpha^2 - 12\,t^7\alpha^2 + 20\,t^7\alpha - 8\,t^9\alpha - 8\,t^7\alpha^3 - 24\,t\alpha^2 - 24\,t - 20\,t^5 - 72\,t^3\alpha\right.$$
$$\left. + 4\,t^9 + 52\,t^3 - 24\,t^3\alpha^2 + 48\,t\alpha - t^{11} - 16\,t^5\alpha - 16\,t^5\alpha^4 + 40\,t^3\alpha^3\right) f\left(t\right)$$
$$+ \left(-18\,t^3 + 9\,t^7 + 18\,t^5\alpha\right)\frac{d^2}{dt^2} f\left(t\right) +$$
$$\left(- 54\,t^2 + 36\,t^6\alpha^2 - 12\,t^6\alpha - 18\,t^4\alpha + 24 + 3\,t^{10} - \right.$$
$$\left. 24\,t^2\alpha + 24\,t^8\alpha - 15\,t^6 - 6\,t^8\right)\frac{d}{dt} f\left(t\right)$$

```
>detoseq(sgraph3, f(t), 5);
```

$$1, 0, 1 - 2\,\alpha + \alpha^2, 0, 30\,\alpha^2 - 16\,\alpha^3 - 24\,\alpha + 8 + 3\,\alpha^4, 0$$

Kronecker product

This next example illustrates how to calculate $\sum s_\lambda * \sum s_\lambda$. See the discussion in Section 6.3.3. First we calculate the even entries.

```
>itensor_de(exp(pn^2/2/n),exp(pn^2/2/n), f);
```

$$\left[-pn\, f\left(pn\right) + \left(1 - pn^2\right)\frac{d}{dpn} f\left(pn\right)\right]$$

```
>dsolve([op(%), f(0)=1], f(pn));
```

$$f\left(pn\right) = \frac{i}{\sqrt{pn - 1}\sqrt{pn + 1}}$$

The odd elements are calculated:

```
>itensor_de(exp(pn^2/2/n+pn/n),exp(pn^2/2/n+pn/n), f);
```

$$\left[\left(1 + pn\, n - pn^2 n + pn\right) f\left(pn\right) + \left(-pn^3 n + pn^2 n + pn\, n - n\right)\frac{d}{dpn} f\left(pn\right)\right]$$

```
>simplify(dsolve([op(%),f(0)=1], f(pn)),exp);
```

$$ie^{-\frac{pn}{n(pn-1)}}\frac{1}{\sqrt{pn + 1}}\frac{1}{\sqrt{pn - 1}}$$

Together these give Proposition 6.2.

References

[1] Andrews, G. E., Askey, R., and Roy, R. *Special functions*, vol. 71 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 1999.

[2] Auger, P., Labelle, G., and Leroux, P. Combinatorial addition formulas and applications. *Adv. in Appl. Math. 28*, 3-4 (2002), 302–342.

[3] Barcucci, E., Del Lungo, A., Pergola, E., and Pinzani, R. ECO: a methodology for the enumeration of combinatorial objects. *J. Differ. Equations Appl. 5*, 4-5 (1999), 435–490.

[4] Bender, E. A. Partitions of multisets. *Discrete Math. 9* (1974), 301–311.

[5] Bergeron, F. Une combinatoire du pléthysme. *J. Combin. Theory Ser. A 46*, 2 (1987), 291–305.

[6] Bergeron, F. A combinatorial outlook on symmetric functions. *J. Combin. Theory Ser. A 50*, 2 (1989), 226–234.

[7] Bergeron, F., Labelle, G., and Leroux, P. *Combinatorial species and tree-like structures*. Cambridge University Press, Cambridge, 1998.

[8] Bernšteĭn, J. N. Modules over a ring of differential operators. An investigation of the fundamental solutions of equations with constant coefficients. *Funkcional. Anal. i Priložen. 5*, 2 (1971), 1–16.

[9] Bernšteĭn, J. N. Analytic continuation of generalized functions with respect to a parameter. *Funkcional. Anal. i Priložen. 6*, 4 (1972), 26–40.

[10] Borel, A., Ed. *Algebraic D-modules*. Academic Press Inc., Boston, MA, 1987.

[11] Bourbaki, N. *Éléments de mathématique. Algèbre. Chapitres 1 à 3*. Hermann, Paris, 1970.

[12] Bousquet-Mélou, M., and Petkovšek, M. Walks confined in a quadrant are not always D-finite. *Theoret. Comput. Sci. 307* (2003), 257–276.

[13] Chyzak, F. *Fonctions holonomes en calcul formel*. PhD thesis, École polytechnique, 1998. INRIA, TU 0531. 227 pages.

[14] Chyzak, F., Mishna, M., and Salvy, B. Effective D-finite symmetric functions. In *FPSAC'02* (2002). extended abstract.

[15] Chyzak, F., Mishna, M., and Salvy, B. Effective scalar products of D-finite symmetric series. submitted for publication. Available as preprint math.CO/0310132, 2003.

[16] Chyzak, F., and Salvy, B. Non-commutative elimination in Ore algebras proves multivariate identities. *J. Symbolic Comput. 26*, 2 (1998), 187–227.

[17] Chyzak, F., Salvy, B., and Zimmerman, P. The algolib library for Maple, including `gfun`, `mgfun`, `holonomy` and `ore_algebra` packages.
Available at: `http://algo.inria.fr/software`.

[18] Comtet, L. Birecouvrements et birevêtements d'un ensemble fini. *Studia Sci. Math. Hungar. 3* (1968), 137–152.

[19] Comtet, L. *Advanced combinatorics*, enlarged ed. D. Reidel Publishing Co., Dordrecht, 1974. The art of finite and infinite expansions.

[20] Coutinho, S. C. *A primer of algebraic D-modules*. Cambridge University Press, Cambridge, 1995.

[21] Cox, D., Little, J., and O'Shea, D. *Ideals, varieties, and algorithms*, second ed. Undergraduate Texts in Mathematics. Springer-Verlag, New York, 1997. An introduction to computational algebraic geometry and commutative algebra.

[22] de Bruijn, N. G. Pólya's theory of counting. In *Applied Combinatorial Mathematics*, E. Beckenbach, Ed. John Wiley and Sons, New York, 1964, pp. 144–184. Chapter 5.

[23] Delest, M. P., and Duchon, P. Exploration de paramètres inconnus par des Q-grammaires. In *SCFA'99, Barcelona* (1999).

[24] Devitt, J. S., and Jackson, D. M. The enumeration of covers of a finite set. *J. London Math. Soc. (2) 25*, 1 (1982), 1–6.

[25] Doubilet, P. On the foundations of combinatorial theory. VII. Symmetric functions through the theory of distribution and occupancy. *Studies in Appl. Math. 51* (1972), 377–396. *Reprinted in* "Gian-Carlo Rota in Combinatorics", J. P. Kung Ed., Birkhäuser, (1995), 403–422.

[26] Dutour, I., and Fédou, J.-M. Object grammars and random generation. *Discrete Mathematics and Theoretical Computer Science 2* (1998), 49–63.

[27] Flajolet, P., and Salvy, B. Computer algebra libraries for combinatorial structures. *J. of Symbolic Computation 20* (1995), 653–671.

[28] Flajolet, P., Zimmerman, P., and Van Cutsem, B. A calculus for the random generation of labelled combinatorial structures. *Theoretical Comput. Sci. 132*, 1-2 (1994), 1–35.

[29] Galligo, A. Some algorithmic questions on ideals of differential operators. In *EUROCAL '85, Vol. 2 (Linz, 1985)*, vol. 204 of *Lecture Notes in Comput. Sci.* Springer, Berlin, 1985, pp. 413–421.

[30] Gessel, I. M. A q-analog of the exponential formula. *Discrete Math. 40*, 1 (1982), 69–80.

[31] Gessel, I. M. Counting three-line Latin rectangles. In *Combinatoire énumérative (Montreal, Que., 1985/Quebec, Que., 1985)*, vol. 1234 of *Lecture Notes in Math.* Springer, Berlin, 1986, pp. 106–111.

[32] Gessel, I. M. Counting Latin rectangles. *Bull. Amer. Math. Soc. (N.S.) 16*, 1 (1987), 79–82.

[33] Gessel, I. M. Enumerative applications of symmetric functions. In *Proceedings of Séminaire Lotharingien de Combinatoire* (1987), vol. B17a. 17pp.

[34] Gessel, I. M. Symmetric functions and P-recursiveness. *J. Combin. Theory Ser. A 53*, 2 (1990), 257–285.

[35] Gessel, I. M. Counting paths in Young's lattice. *J. Statist. Plann. Inference 34*, 1 (1993), 125–134.

[36] Goulden, I. P., and Jackson, D. M. *Combinatorial enumeration*. Wiley-Interscience Series in Discrete Mathematics. John Wiley & Sons Inc., New York, 1983.

[37] Goulden, I. P., Jackson, D. M., and Reilly, J. W. The Hammond series of a symmetric function and its application to P-recursiveness. *SIAM J. Algebraic Discrete Methods 4*, 2 (1983), 179–193.

[38] Goupil, A., and Schaeffer, G. Factoring n-cycles and counting maps of given genus. *European J. Combin. 19*, 7 (1998), 819–834.

[39] Gupta, H. Enumeration of symmetric matrices. *Duke Math. J. 35* (1968), 653–659.

[40] Hall, P. The algebra of partitions. In *Proceedings of the 4th Canadian Math. Congress* (1959), pp. 147–159.

[41] Hammond, J. On the use of certain differtial operators in the theory of equations. *Proc. London Math. Soc. 14* (1883), 119–129.

[42] Harris, Jr., W. A., and Sibuya, Y. The reciprocals of solutions of linear ordinary differential equations. *Adv. in Math. 58*, 2 (1985), 119–132.

[43] Jing, N. Vertex operators and generalized symmetric functions. In *Proceedings of the Conference on Quantum Topology (Manhattan, KS, 1993)* (River Edge, NJ, 1994), World Sci. Publishing, pp. 111–126.

[44] Joyal, A. Une théorie combinatoire des séries formelles. *Adv. in Math. 42*, 1 (1981), 1–82.

[45] Klazar, M. Non-p-recursiveness of numbers of matchings or linear chord diagrams with many crossings. *Adv. Appl. Math 30*, 1 (2003), 126–136.

[46] Lehrstuhl Mathematik II. `SYMMETRICA`–A collection of routines in c for computations in representation theory.
Available at: `http://www.mathe2.uni-bayreuth.de/axel/symneu_engl.html`.

[47] Leykin, A., and Tsai, H. D-Module package for Macaulay2.
Available at: `http://www.math.uiuc.edu/Macaulay2`.

[48] Lipshitz, L. The diagonal of a D-finite power series is D-finite. *J. Algebra 113*, 2 (1988), 373–378.

[49] Lipshitz, L. D-finite power series. *J. Algebra 122*, 2 (1989), 353–373.

[50] Littlewood, D. E. The kronecker product of symmetric group representations. *J. London Math Soc. 31* (1956), 89–93.

[51] Macdonald, I. G. Schur functions: theme and variations. In *Séminaire Lotharingien de Combinatoire (Saint-Nabor, 1992)*, vol. 498 of *Publ. Inst. Rech. Math. Av.* Univ. Louis Pasteur, Strasbourg, 1992, pp. 5–39.

[52] Macdonald, I. G. *Symmetric functions and Hall polynomials*, second ed. The Clarendon Press Oxford University Press, New York, 1995.

[53] MacMahon, P. A. *Combinatory analysis*. Two volumes (bound as one). Chelsea Publishing Co., New York, 1960.

[54] MacMahon, P. A. *Collected papers. Vol. I.* MIT Press, Cambridge, Mass., 1978. Combinatorics, Mathematicians of Our Time, Edited and with a preface by George E. Andrews.

[55] MacMahon, P. A. *Collected papers. Vol. II*, vol. 24 of *Mathematicians of Our Time*. MIT Press, Cambridge, MA, 1986. Number theory, invariants and applications, Edited and with a preface by George E. Andrews.

[56] Mallinger, C. Algortihmic manipulations and transformations of univariate holonomic functions and sequences. Master's thesis, RISC, Johannes Kepler Universität Linz, Austria, Aug. 1996.

[57] Mishna, M. J. Attribute grammars and automatic complexity analysis. *Advances in Applied Mathematics 30*, 1-2 (2003), 189–207.

[58] Mora, F. Gröbner bases for non-commutative polynomial rings. In *Algebraic algorithms and error-correcting codes* (Berlin, 1985), J. Calmet, Ed., vol. 229 of *lncs*, Springer Verlag, pp. 353–362. Conference proceedings of AAECC-3, Grenoble, France.

[59] Mora, T. Seven variations on standard bases. Preprint 45, Università di Genova, Dipartimento di Matematica, Mar. 1988.

[60] Mora, T. An introduction to commutative and noncommutative Gröbner bases. *Theoretical Comput. Sci. 134* (1994), 131–173.

[61] Paule, P., and Riese, A. Software for hypergeometric summation and q-hypergeometric summation.
Available at: `http://www.risc.uni-linz.ac.at/research/combinat/risc/software/`.

[62] Read, R. C., and Wormald, N. C. Number of labeled 4-regular graphs. *J. Graph Theory 4*, 2 (1980), 203–212.

[63] Rechnitzer, A. Haruspicy and anisotropic generating functions. *Advances in Applied Mathematics 30* (2003), 228–257.

[64] Redfield, J. H. The theory of group reduced distributions. *American Journal of Mathematics 49* (1927), 433–455.

[65] Rosas, M. H. The Kronecker product of Schur functions indexed by two-row shapes or hook shapes. *J. Algebraic Combin. 14*, 2 (2001), 153–173.

[66] Rosas, M. H. MacMahon symmetric functions, the partition lattice, and Young subgroups. *J. Combin. Theory Ser. A 96*, 2 (2001), 326–340.

[67] Rosas, M. H. Specializations of MacMahon symmetric functions and the polynomial algebra. *Discrete Math. 246*, 1-3 (2002), 285–293. Formal power series and algebraic combinatorics (Barcelona, 1999).

[68] Sagan, B. E. *The symmetric group*, second ed., vol. 203 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2001.

[69] Saito, M., Sturmfels, B., and Takayama, N. *Gröbner deformations of hypergeometric differential equations*, vol. 6 of *Algorithms and Computation in Mathematics*. Springer-Verlag, Berlin, 2000.

[70] Salvy, B., and Zimmermann, P. Gfun: a Maple package for the manipulation of generating and holonomic functions in one variable. *ACM Transactions on Mathematical Software 20*, 2 (1994), 163–177.

[71] Singer, M. F. Algebraic relations among solutions of linear differential equations. *Trans. Amer. Math. Soc. 295*, 2 (1986), 753–763.

[72] Sloane, N. The on-line encyclopedia of integer sequence.
Available at: `http://www.research.att.com/~njas/sequences/`.

[73] Sperber, S. On solutions of differential equations which satisfy certain algebraic relations. *Pacific J. Math. 124*, 1 (1986), 249–256.

[74] Stanley, R. P. Differentiably finite power series. *European J. Combin. 1*, 2 (1980), 175–188.

[75] Stanley, R. P. *Enumerative combinatorics. Vol. 2.* Cambridge University Press, Cambridge, 1999.

[76] Stein, P. R., and Zemach, C. Symmetric function algebra on a computer. *Adv. in Appl. Math. 14*, 4 (1993), 430–454.

[77] Stembridge, J. R. A Maple package for symmetric functions. *J. Symbolic Comput. 20*, 5-6 (1995), 755–768.
Available at: `http://www.math.lsa.umich.edu/~jrs/maple.html`.

[78] Sundaram, S. Orthogonal tableaux and an insertion algorithm for $\mathrm{SO}(2n+1)$. *J. Combin. Theory Ser. A 53*, 2 (1990), 239–256.

[79] Takayama, N. `kan/sm1`, a system for computation in algebraic analysis.
Available at: `http://www.math.sci.kobe-u.ac.jp/KAN/index.html`.

[80] Takayama, N. An algorithm of constructing the integral of a module– an infinite dimensional analog of Gröbner basis. In *Proceedings of ISSAC'90, Kyoto* (1990), ACM, pp. 206–211.

[81] Takayama, N. An approach to the zero recognition problem by Buchberger algorithm. *J. Symbolic Comput. 14*, 2-3 (1992), 265–282.

[82] van der Poorten, A. A proof that Euler missed...Apéry's proof of the irrationality of $\zeta(3)$. *Math. Intelligencer 1*, 4 (1979), 195–203. An informal report.

[83] Veigneau, S. ACE, an Algebraic Combinatorics Environment for the computer algebra system MAPLE: User's Reference Manual, Version 3.0. Report 98–11, IGM, 1998.
Available at: `http://phalanstere.univ-mlv.fr/~ace/ACE/3.0/index.html`.

[84] Wormald, Nick, C. The number of labelled cubic graphs with no triangles. In *Twelfth Southeastern Conference on Combinatorics, Graph theory and computing, Vol. II* (1981), 33, pp. 359–378.

[85] Wybourne, B. G. `SCHUR` group theory software–An interactive program for calculating properties of lie groups and symmetric functions.
Available at: `http://smc.vnet.net/schur.html`.

[86] Zeilberger, D. `Ekhad` and `qekhad` packages for Maple.
Available at: `http://www.math.rutgers.edu/~zeilberg/programs.html`.

[87] Zeilberger, D. A fast algorithm for proving terminating hypergeometric identities. *Discrete Math. 80*, 2 (1990), 207–211.

[88] Zeilberger, D. A holonomic systems approach to special functions identities. *J. Comput. Appl. Math. 32*, 3 (1990), 321–368.

[89] Zeilberger, D. The method of creative telescoping. *J. Symbolic Comput. 11*, 3 (1991), 195–204.

Index