Bayreuther Mathematische Schriften **43** (1994), 11-35

Die Speziestheorie und die Implementation ihrer Zykelindexreihen in Scratchpad II

von

Joachim Hornegger

Roberto Pirastu

Zusammenfassung

Die Speziestheorie erlaubt, kombinatorisch anschauliche Überlegungen direkt auf erzeugende Funktionen zu übertragen. Damit kann oft die Abzählung komplexer Strukturen durch geschickte Zerlegungen auf bereits Bekanntes zurückgeführt und damit erheblich vereinfacht werden. Die vorliegende Arbeit beinhaltet eine knappe Einführung in die von A. Joyal begründete Speziestheorie und die Beschreibung einer konkreten Implementierung von einigen grundlegenden Konstrukten. Als Programmierumgebung dient dazu das Computer-Algebra-System Scratchpad II. [1] [2]

1 Einführung

Zu den typischen Fragestellungen der Kombinatorik gehören Abzählprobleme. Man definiert Klassen von Objekten oder Strukturen, die besonderen Eigenschaften genügen, und stellt dann u.a. die natürliche Frage: Wieviele solche Strukturen existieren? Ferner interessiert man sich beispielsweise für die (Isomorphie-)Typen oder auch für Strukturen, die weitere Eigenschaften (z.B. eine gewisse Symmetrie) besitzen. Bei der Lösung der Abzählprobleme versucht man oft, die zu untersuchenden Strukturen mittels passender Konstruktoren in Zusammenhang mit "einfacheren" Strukturen zu bringen,

[1] Das Computer-Algebra-System Scratchpad II stand dem Lehrstuhl für Formale Sprachen und Automatentheorie der Universität Erlangen-Nürnberg im Rahmen eines Studienvertrags "Erprobung von Scratchpad II" mit der Firma IBM zur Verfügung.

[2] Scratchpad II wird seit Dezember 1991 unter dem Namen Axiom von der Firma NaG, Oxford, vertrieben.

für welche die entsprechenden Abzählungen leichter durchgeführt werden können.

Die im wesentlichen von A. Joyal ([Jo]) eingeführte Theorie der Spezies stellt einen passenden Rahmen zur Verfügung, in dem man eine große Anzahl solcher Zählprobleme lösen kann.

Bei der Behandlung von Zählproblemen sind formale Potenzreihen bekanntlich ein wichtiges Hilfsmittel. Die Theorie der Spezies stellt insbesondere Operationen und Regeln bereit, die den oben erwähnten kombinatorischen Zusammenhang zwischen Strukturen auf die entsprechenden Abzählungen und somit auch auf die dazugehörenden formalen Potenzreihen übertragen. Die auftretenden Berechnungen auf Potenzreihen sind aber i.a. sehr aufwendig und rein technischer Natur. Deshalb wurde eine Implementierung realisiert, die solche Berechnungen weitgehend übernimmt.

Zur Verwaltung von formalen Potenzreihen eignen sich besonders solche Computer-Algebra-Systeme, die unbeschränkte Folgen von Objekten zulassen. Die zu beschreibende Implementierung wurde als Erweiterung des Systems Scratchpad II im Rahmen einer von V. Strehl betreuten Studienarbeit realisiert. Sie wurde zuerst auf dem Rechner IBM 3090 des RRZE unter VM/CMS fertiggestellt und dann auf eine IBM RISC 6000 unter AIX übertragen.

Im Abschnitt 2 dieser Arbeit werden wir einige grundlegende Definitionen und Begriffe der Theorie der Spezies erläutern, indem wir ein konkretes Beispiel untersuchen. Für dieses Beispiel werden dann im Abschnitt 3 durch die Einführung einiger Operationen auf Spezies Methoden vorgestellt, um das dazugehörige Abzählproblem zu vereinfachen. Der wichtige Begriff der Zykelindexreihe wird im Abschnitt 4 eingeführt in Zusammenhang mit den vorgestellten Operationen auf Spezies.

Der ganze Abschnitt 5 beschreibt die Grundideen der Implementierung auf Scratchpad II. Zuerst wird der für die Behandlung von Potenzreihen sehr nützliche Begriff des Streams in 5.1 vorgestellt. In 5.2 wird die Verwendung solcher Gebilde zur Darstellung von Reihen in einer Variable besprochen, um dann in 5.3 den Ansatz zur Implementierung der Zykelindexreihen vorzustellen. 5.4 enthält eine kurze Beschreibung der Objekte, die der Benutzer des Systems behandeln kann.

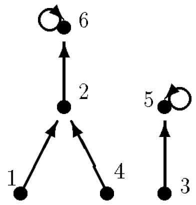
Die Struktur und die Hierarchie der erstellten Programme ist im Abschnitt 6 graphisch dargestellt. Schließlich wird im Abschnitt 7 eine konkrete Sitzung beschrieben. Hierbei wird das im Abschnitt 1 angegebene Beispiel behandelt und die entsprechenden Abzählprobleme algorithmisch gelöst.

2 Grundlegende Begriffe

Zur Verdeutlichung der wichtigsten Konzepte der Theorie der Spezies werden wir uns eine konkrete Klasse von Strukturen vornehmen, um die dazugehörigen Abzählprobleme zu lösen. Dieses Beispiel wird uns in der ganzen Arbeit begleiten.
Sei A eine endliche Menge; wir betrachten die Menge $F[A]$ aller Endofunktionen auf A mit der Eigenschaft, daß jedes Element entweder keinen oder genau zwei Vorgänger hat:

$$F[A] := \{f : A \rightarrow A \ ; \ \forall a \in A : \#f^{-1}(a) \in \{0, 2\}\}. \tag{1}$$

Für die Menge $A = \{1, 2, 3, 4, 5, 6\}$ wäre die durch folgenden Graphen beschriebene Funktion ein Beispiel für ein Element aus $F[A]$.



Wir sprechen von der **Spezies** F, die im wesentlichen diese Zuordnung $A \Rightarrow F[A]$ beschreibt. Die Elemente aus $F[A]$ für ein beliebiges A nennen wir dann F-**Strukturen**. Wegen der Funktorialität der Spezies (siehe [Jo]) ist die Natur der Elemente aus A nicht wesentlich und man kann sich auf den Fall $A = [\underline{n}] := \{1, \ldots, n\}$ beschränken. Die erste Frage, die bei der Betrachtung der F-Strukturen auftritt, wird natürlich sein, ob und wie man deren Anzahl $\#F[\underline{n}]$ bestimmen kann. In einfacheren Fällen werden wir in der Lage sein, eine geschlossene Formel für solch eine Anzahl anzugeben, wie z.B. für die Spezies der Permutationen:

$$S[A] := \{f : A \rightarrow A \ ; \ f \text{ ist bijektiv } \}$$

Hier ist $\#S[\underline{n}] = n!$. In anderen Fällen wird das nicht möglich sein. Oft kennt man aber für die gesuchten Zahlen eine Rekurrenz-Relation, wie z.B. für die Spezies der Partitionen:

$$P[A] := \{\tau : \tau \text{ ist eine Partition von } A\}$$

Für $\#P[\,\underline{n}\,]$ gilt nämlich:

$$B_n = \#P[\,\underline{n}\,] = \sum_{k\geq 1} \binom{n-1}{k-1} \#P[\,\underline{n-k}\,]$$

wobei B_n die n-te Bell-Zahl bezeichnet. Diese numerische Information über die Strukturen einer Spezies stellt man als formale Potenzreihe dar. Für eine beliebige Spezies G ist das die (exponentielle) erzeugende Funktion:

$$G(x) = \sum_{n\geq 0} \#G[\,\underline{n}\,]\ \frac{x^n}{n!}$$

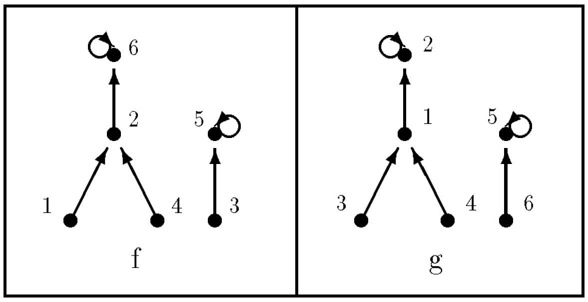
$G(x)$ heißt dann die **Kardinalität** (der numerierten Strukturen) der Spezies F. Für die angegebenen Beispiele ist bekannt oder leicht nachzurechnen, daß

$$S(x) = \sum_{n\geq 0} n!\ \frac{x^n}{n!} = \frac{1}{1-x}$$

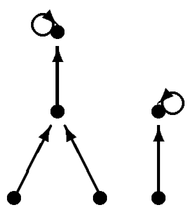
und

$$P(x) = \sum_{n\geq 0} B_n\ \frac{x^n}{n!} = e^{e^x} - 1$$

Außer der Anzahl der F-Strukturen interessiert man sich auch für die Anzahl der (Isomorphie-)Typen der F-Strukturen. Zu einem Isomorphie-Typ faßt man alle Strukturen aus $F[\,\underline{n}\,]$ zusammen, die sich nur durch eine Umbenennung der zugrundeliegenden Knoten unterscheiden. In unserem Beispiel würde das heißen, daß folgende Funktionen f und g aus $F[\ \underline{6}\ ]$ denselben Typ besitzen.



In der Tat entsteht g aus f durch Anwendung der Permutation (1362)(4)(5) und wir können den dazugehörenden Isomorphie-Typ durch den unmarkierten Funktionsgraphen veranschaulichen



In der Sprache der Permutationsgruppen würde man sagen, daß man die Bahnen $O(F[\underline{n}])$ folgender Gruppenoperation der symmetrischen Gruppe $SYM(n)$ auf n Elementen untersucht:

$$SYM(n) \times F[\underline{n}] \longrightarrow F[\underline{n}]\ ,\ (\sigma, f) \longmapsto F[\sigma]f \qquad (2)$$

wobei $F[\sigma]f$ die F-Struktur beschreibt, die durch Umbenennung der Knoten aus f gewonnen wird. Die gesuchte Anzahl von Isomorphie-Typen ist dann genau $\#O(F[\underline{n}])$. Auch diese numerische Information wird als formale Potenzreihe durch folgende (gewöhnliche) erzeugende Funktion dargestellt.

$$\tilde{F}(x) = \sum_{n \geq 0} \#O(F[\underline{n}])\ x^n \qquad (3)$$

und wir sprechen dann von der **typerzeugenden Funktion** der Spezies F. Die Reihe $\tilde{F}(x)$ wird als gewöhnliche erzeugende Funktion definiert aber unten werden wir eine andere Interpretation als exponentielle erzeugende Funktion angeben.
Bei der Spezies S der Permutationen kann man sich einen Isomorphie-Typ als Partition auf die zugrundeliegende Menge vorstellen, die einfach die Zyklen-Struktur der Permutation beschreibt. Wenn wir also mit $p(n)$ die Anzahl der Partitionen auf $[\underline{n}]$ bezeichnen, haben wir bekanntlich

$$\tilde{S}(x) = \sum_{n \geq 0} p(n)\ x^n = \prod_{k \geq 1} \frac{1}{1 - x^k}$$

Für die Spezies P der Partitionen gilt auch

$$\tilde{P}(x) = \sum_{n \geq 0} p(n)\ x^n = \prod_{k \geq 1} \frac{1}{1 - x^k}$$

In diesem Zusammenhang ist eine Betrachtung der in (2) beschriebenen Gruppenoperation von besonderem Interesse. Für die Länge der Bahn $O(s)$

einer F-Struktur s gilt nach dem Lemma von Burnside-Cauchy-Frobenius

$$\#O(s) = \frac{\#SYM(\underline{n})}{\#(SYM(\underline{n}))_s}$$

wobei $(SYM(\underline{n}))_s = \{\sigma \in SYM(\underline{n}); F[\sigma]s = s\}$ die Menge der Permutationen ist, die s invariant lassen. Danach

$$\#O(F[\,\underline{n}\,]) = \sum_{s\,\in\, F[\,\underline{n}\,]} \frac{1}{\#O(s)} = \frac{1}{n!} \sum_{s\,\in\, F[\,\underline{n}\,]} \#(SYM(\underline{n}))_s$$

Wenn wir dann die zu F **assoziierte Spezies** $\tilde{F}$ definieren durch

$$\tilde{F}[A] = \{(s,\sigma)\ ;\ s \in F[A] \wedge F[\sigma]s = s\}$$

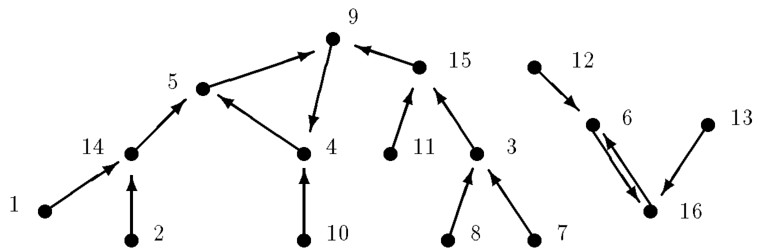
können wir folgendes beobachten

$$\#O(F[\,\underline{n}\,]) = \frac{\#\tilde{F}[\,\underline{n}\,]}{n!}.$$

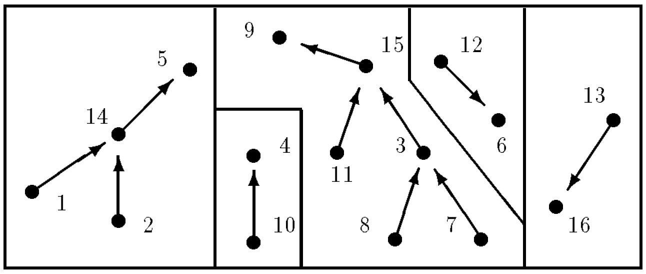
Damit kann man die in (3) definierte typerzeugende Funktion der Spezies F als die übliche Kardinalität der assoziierten Spezies $\tilde{F}$ ansehen.
Bis jetzt wurden nur Definitionen und Begriffe vorgestellt, die helfen sollen, die Problemstellung in der Sprache der Spezies zu formulieren. Sie geben aber kein Verfahren an, um das Problem zu lösen. Wir werden im nächsten Abschnitt Möglichkeiten kennenlernen, kombinatorische Überlegungen bzgl. des Aufbaus der Strukturen in die Theorie der Spezies einzubetten. Damit wird die Berechnung der erzeugenden Funktionen zu einer Spezies auf die Behandlung "einfacherer" Spezies reduziert.

3 Operationen auf Spezies

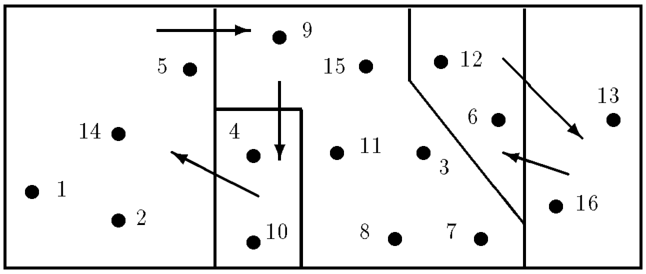
Betrachten wir wieder die anfangs definierten F-Strukturen, z.B. folgende Endofunktion aus $F[\,\underline{16}\,]$.



Wir können beobachten, daß diese Struktur auch folgendermaßen interpretiert werden kann: Zuerst betrachten wir nur die Kanten, die zu keinem Zyklus im Graphen gehören. Die baumartigen Zusammenhangskomponenten, die dadurch entstehen, nennen wir F-**Kontraktionen**. Ihre Form ist i.a. die eines Knotens, an dem ein binärer Baum hängt. Wir bekommen also eine Versammlung "einfacherer" Strukturen: Die Menge $\{1, \ldots, 16\}$ wird zerlegt in $\{\{1, 2, 5, 14\}, \{4, 10\}, \{3, 7, 8, 9, 11, 15\}, \{6, 12\}, \{13, 16\}\}$, und auf jede Menge der Zerlegung wird eine F-Kontraktion gelegt. Das können wir durch folgendes Bild verdeutlichen.



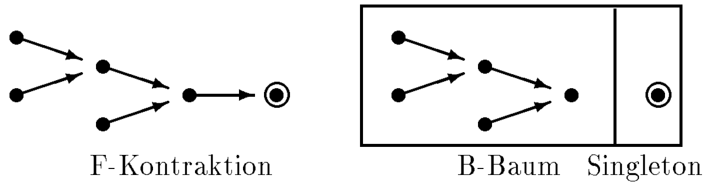
Die Teile der Zerlegung werden dann permutiert, d.h. wir legen eine Permutation auf die Menge der entstandenen F-Kontraktionen (die vorhin gelöschten Kanten).



Jede F-Struktur läßt sich auf diese Weise eindeutig zerlegen und jede so aufgebaute Struktur (Partition der zugrundeliegenden Menge, beliebige F-Kontraktion auf die einzelnen Mengen und beliebige Permutation auf alle Mengen der Partition) kann als F-Struktur betrachtet werden. Solch ein Zusammenhang wird in der Theorie der Spezies durch die Operation der **Substitution** beschrieben. Wenn wir annehmen, daß wir die F-Kontraktionen durch eine Spezies $Fkon$ schon beschreiben konnten, können wir symbolisch schreiben

$$F = S \circ Fkon \tag{4}$$

Betrachten wir jetzt die $Fkon$-Strukturen und versuchen, auch für sie eine geeignete Zerlegung zu finden. Wie schon beobachtet, kann jede F-Kontraktion auf n Knoten folgendermaßen interpretiert werden. Die Wurzel wird ausgezeichnet und auf die Menge der übrigen Knoten wird ein binärer Baum gelegt.



Dieser Sachverhalt entspricht einem **Produkt** von Spezies. Wenn wir mit $Btree$ bzw. mit X die Spezies der binären Bäume bzw. des isolierten Knotens bezeichnen, folgt aus obiger Beobachtung folgende Zerlegung

$$Fkon = X * Btree \tag{5}$$

Eine weitere Zerlegung für die $Btree$-Strukturen kann leicht plausibel gemacht werden

$$Btree = X + X * (E_2 \circ Btree) \tag{6}$$

wobei E_2 die Spezies der 2-elementigen Mengen beschreibt, also $E_2(x) = x^2/2$. Die Zerlegung erklärt sich dadurch, daß eine $Btree$-Struktur entweder aus einem einzigen Knoten besteht oder aus einem Knoten (Wurzel), der mit zwei weiteren $Btree$-Strukturen verbunden ist. Hierbei ist die Ordnung dieser zwei Unterbäume unwichtig. Die bis jetzt angegebenen Zerlegungen haben uns noch nicht dazu verholfen, etwa die Reihen $F(x)$ oder $\tilde{F}(x)$ zu berechnen. In der Tat können aber die angewandten Operationen auf die erzeugenden Funktionen der betrachteten Spezies zurückgeführt werden. So gilt z.B. für beliebige Spezies G und H

$$\begin{aligned} (G+H)(x) &= G(x) + H(x) \\ (G*H)(x) &= G(x) * H(x) \\ (G \circ H)(x) &= G(H(x)) \end{aligned} \tag{7}$$

Damit kann man in unserem Beispiel die Kardinalität der Spezies F leicht durch die Beziehung $F(x) = S(Fkon(x))$ berechnen, vorausgesetzt, man kennt die Kardinalitäten für S und $Fkon$. Um $Fkon(x)$ zu bestimmen, brauchen wir $Btree(x)$. Aus (6) gilt für die Kardinalität

$$Btree(x) = x + x * (\frac{x^2}{2} \circ Btree(x)) = x + x * (\frac{Btree(x)^2}{2}) \tag{8}$$

Daraus ergibt sich

$$Btree(x) = \frac{1 - \sqrt{1 - 2x^2}}{x} \tag{9}$$

und somit

$$Fkon(x) = 1 - \sqrt{1 - 2x^2} \tag{10}$$

Somit sind wir in der Lage, die gesuchte $F(x)$ zu berechnen. Wegen (4) gilt:

$$F(x) = S(Fkon(x)) = \frac{1}{1 - (1 - \sqrt{1 - 2x^2})} = \frac{1}{\sqrt{1 - 2x^2}} \tag{11}$$

Für die typerzeugenden Funktionen werden soweit möglich entsprechende Beziehungen wie in (7) aufgestellt. Es gibt aber Fälle, die eine Übertragung nicht zulassen, denn es gilt z.B. für die Substitution

$$(\widetilde{G \circ H})(x) \neq \tilde{G}(\tilde{H}(x)) \tag{12}$$

Da scheitert anscheinend unser Ansatz, durch Zerlegungen das Problem auf einfachere zu reduzieren. Um diese Schwäche zu umgehen, definiert man eine weitere formale Potenzreihe, die sowohl die Informationen aus der Kardinalität als auch die aus der typerzeugenden Funktion beinhaltet und sich bzgl. der Operationen "gutartig" verhält. Die zugrundeliegenden Ideen werden im nächsten Abschnitt vorgestellt.

4 Die Zykelindexreihe

Nach der Betrachtung der in (2) definierten Gruppenoperation und der zu F assoziierten Spezies tritt auf natürliche Weise eine andere Frage auf. Man versucht zu jeder Permutation σ und zu einer Spezies F abzuzählen, wieviele F-Strukturen unter der Wirkung von σ invariant bleiben. Diese Anzahl bezeichnen wir folgendermaßen

$$Fix_n(F[\sigma]) = \#\{f \in F[\underline{n}] \ ; \ F[\sigma]f = f\}$$

Diese Fragestellung findet in der Abzähltheorie von Pólya (siehe [PR]) eine ausführliche Behandlung. Nun kann man sich leicht überlegen, daß diese

Anzahl für zwei Permutationen gleich ist, falls sie konjugiert zueinander sind, d.h. wenn sie den gleichen Zykeltyp besitzen. Diese Information "beinhaltet" gewissermaßen auch die bisher betrachteten Anzahlen, denn offensichtlich gilt

$$Fix_n(F[id_n]) = \#\{f \in F[\underline{n}]\ ;\ F[id_n]f = f\} = \#F[\underline{n}]$$

und

$$\sum_{\sigma \in SYM(\underline{n})} Fix_n(F[\sigma]) = \#\tilde{F}[\underline{n}]$$

Zur Erfassung dieser Daten definieren wir eine neue formale Potenzreihe. Die **Zykelindexreihe** der Spezies F wird definiert durch

$$Z_F(x_1, x_2, x_3, \ldots) = \sum_{n \geq 0} \frac{1}{n!} \sum_{\sigma \in S_n} Fix_n(F[\sigma]) \cdot x_1^{\sigma_1} x_2^{\sigma_2} x_3^{\sigma_3} \cdots, \tag{13}$$

wobei $(\sigma_1, \sigma_2, \sigma_3, \ldots)$ den Zykeltyp der Permutation σ bezeichnet.
Es handelt sich dabei um eine Potenzreihe in einer unbeschränkten Anzahl von Variablen. Die Zusammenhänge mit den schon bekannten erzeugenden Funktionen einer Spezies können folgendermaßen beschrieben werden, wie man durch einfaches Einsetzen in (13) überprüfen kann.

$$F(x) = Z_F(x, 0, 0, \ldots) \tag{14}$$
$$\widetilde{F}(x) = Z_F(x, x^2, x^3, \ldots) \tag{15}$$

Beispiel:
Für die Spezies *Permutation* gilt:

$$Z_S(x_1, x_2, x_3, \ldots) = \prod_{i \geq 1} \frac{1}{1 - x_i}$$

und folglich

$$\begin{aligned} S(x) &= Z_S(x, 0, 0, \ldots) = \frac{1}{1-x} \\ \widetilde{S}(x) &= Z_S(x, x^2, x^3, \ldots) = \prod_{i \geq 0} \frac{1}{1 - x^i} \end{aligned}$$

Auch für die Zykelindexreihen werden Beziehungen wie in (7) aufgestellt. Es zeigt sich, daß auch die Substitution von Spezies einer Operation auf

Zykelindexreihen entspricht. Wenn wir diese Operation, die **plethystische Substitution** genannt wird, folgendermaßen definieren:

$$Z_G(x_1,x_2,x_3,\ldots) \circ Z_H(x_1,x_2,x_3,\ldots) := Z_G \left(\begin{array}{l} Z_H(x_1,x_2,x_3,\ldots), \\ Z_H(x_2,x_4,x_6,\ldots), \\ Z_H(x_3,x_6,x_9,\ldots),\ldots \end{array} \right)$$

dann gilt

$$Z_{G\circ H}(x_1,x_2,x_3\ldots) = Z_G(x_1,x_2,x_3,\ldots) \circ Z_H(x_1,x_2,x_3,\ldots) \tag{16}$$

Mit (16) und (14) kann man also das Problem lösen, das bei den typerzeugenden Funktionen in (12) aufgetreten war. Wir können nämlich $(\widetilde{G \circ H})(x)$ angeben durch

$$(\widetilde{G \circ H})(x) = Z_G(\tilde{H}(x), \tilde{H}(x)^2, \ldots) \tag{17}$$

Bis jetzt sind wir also in der Lage, die Zykelindexreihe einer Spezies N (und somit auch die dazugehörenden Kardinalitäten) zu berechnen, wenn wir eine Zerlegung von N in schon bekannte Spezies kennen. Oft kann man aber für N nur eine indirekte Zerlegung angeben. In unserem ursprünglichen Problem mußten wir z.B. aus (6) folgende Gleichung für die Kardinalität $Btree(x)$ lösen

$$Btree(x) = x * (1 + E_2(x)) \circ Btree(x)$$

Wegen der einfachen Gestalt von $(1 + E_2)(x)$ konnte man ohne Mühe die Lösung angeben. Für die Zykelindexreihen gibt es Formeln, die aus Gleichungen der Form $M = X * (N \circ M)$ die Reihe $Z_M(x_1, x_2, \ldots)$ für allgemeine Spezies N bestimmen.
Ferner lassen sich für zahlreiche Spezies Beziehungen der Form $L = M \circ N$ finden, wobei die Spezies L und M bekannt sind. In diesem Fall benötigt man Inversionsformeln, um aus den Zykelindexreihen der Spezies L und M die Reihe $Z_N(x_1, x_2, x_3, \ldots)$ zu bestimmen. Eine allgemeine Inversionsformel wurde von Gilbert Labelle gefunden. Jedoch sind Berechnungen nach dieser Formel sehr aufwendig, so daß spezielle Inversionsformeln für verschiedene Spezies M wesentlich effizientere Algorithmen nach sich ziehen. Für $M = S$ gilt z.B.:

$$Z_N(x_1, x_2, \ldots) = 1 - \prod_{k \geq 1} (Z_L(x_k, x_{2k}, \ldots))^{-\mu(k)}.$$

Weitere Formeln, sowie die Operationen auf Spezies und Zykelindexreihen, werden in [HP] ausführlich besprochen.
Diese knappe Einführung wollte nur einige Möglichkeiten der Theorie der Spezies vorstellen, kombinatorische Überlegungen und Zerlegungen für die Abzählung von Strukturen zu verwenden. Dabei ist der wesentliche Begriff die Zykelindexreihe, die alle von uns gesuchten Informationen enthält.

5 Aufgabenstellung

Unser Ziel war die Bereitstellung eines Domains (Datenstruktur) *Spezies* im System Scratchpad II. Nach der Architektur von Scratchpad II entspricht dieses der Definition eines geeigneten Domains und eventuell nötiger Packages, die dem Anwender die Behandlung von Spezies als neue Datenstruktur ermöglichen. Die ganze Information, die in der Datenstruktur enthalten sein soll, besteht aus der Zerlegung der Spezies und aus ihrer Zykelindexreihe. Die Abzählungen der Strukturen erfolgt also nicht durch Erzeugen der Strukturmengen, sondern durch die Bestimmung der zugehörigen Zykelindexreihe, die eventuell aus einer bekannten Zerlegung berechnet wird. Damit liegt der Schwerpunkt der Implementation auf der Realisierung von Zykelindexreihen.
Solche Reihen waren im System nicht verfügbar und mußten neu implementiert werden. Wir werden im folgenden zuerst die Möglichkeit untersuchen, ein unbeschränktes Gebilde wie eine formale Potenzreihe in einer Variablen im System zu verwalten. Anschließend werden wir daraus einen Lösungsansatz für die Zykelindexreihen entwickeln.

5.1 Der Stream

Scratchpad II basiert auf der Programmiersprache LISP. Deswegen steht zur Darstellung von Potenzreihen eine sehr nützliche Datenstruktur zur Verfügung: der Stream. Dieser unterscheidet sich von den üblichen Listen durch die Eigenschaft, keine fest vorgegebene Länge zu besitzen. Zu jedem Zeitpunkt ist nur eine endliche Anzahl von Elementen abgespeichert und man kann den Stream wie eine Liste behandeln. Die Länge dieser Liste ist aber theoretisch unbeschränkt und die Sprache wird bei einem Zugriff auf noch nicht berechnete Elemente der Liste die Liste entsprechend erweitern. Die Definition eines Streams erfolgt durch die Angabe einer Berechnungsvorschrift für die einzelnen Komponenten. Die Definition

```
potwo (n:  I) : Stream I ==
   delay cons ( 2 ** n, potwo ( n+1 ))
```

liefert zum Beispiel eine Funktion, die einen Stream erzeugt, der als n-tes Element die Zahl 2^n enthält. Die Funktion `cons` ist von LISP übernommen und sorgt für das Einfügen eines Elements am Listenanfang. Das `delay` bewirkt, daß der Aufruf des nächsten `potwo (n+1)` nur dann erfolgt, wenn das Ergebnis benötigt wird. Der Aufruf kann durchaus auch von mehreren Parametern abhängen. Durch

```
pt := potwo (0)
(1)   [ 1, 2, 4, 8, 16, ...]
Type:  ST I
```

wird dann der Stream einer Variable zugewiesen. Der Inhalt wird als Liste dargestellt, soweit die Elemente berechnet wurden (hier bis $n = 4$). Die weiteren Elemente werden nur auf Anforderung berechnet und abgespeichert. Beim Aufruf

```
pt.7
(2)   128
Type:  I
```

wurden also die neuen Elemente für $n = 5, 6, 7$ nach der in der Definition angegebenen Vorschrift ausgerechnet. Somit bleibt der Speicherverbrauch auf das Nötige beschränkt, unter Gewährleistung der potentiellen Unbeschränktheit der Liste.

5.2 Darstellung der Reihen aus $R[[x]]$

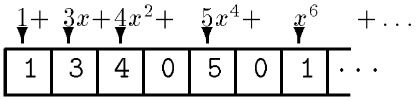
Jetzt liegt es auf der Hand, wie der Stream zur Darstellung einer formalen Potenzreihe aus $R[[x]]$ Verwendung finden kann: man speichert die Koeffizienten in einem Stream ab. Der i-te Koeffizient entspricht dabei der i-ten Stream-Komponente. Wir untersuchen zur Verdeutlichung die Implementation der Reihen in einer Variable. Einige Aspekte werden ausfürlich dargestellt, um später die Analogie zur Implementation der Zykelindexreihe deutlich zu machen.
Habe also die gewünschte Reihe die Form

$$F(x) = \sum_{n \geq 0} f(n) x^n,$$

wobei $f : N \longrightarrow R$ eine Funktion ist, die die Koeffizienten in Abhängigkeit von n und eventuell von anderen externen Parametern bestimmt. Dieser Reihe ordnen wir dann einen Stream zu, der folgendermaßen definiert wird:

```
r( n:  I) : ST R ==
   delay cons ( f(n), r(n+1))
```

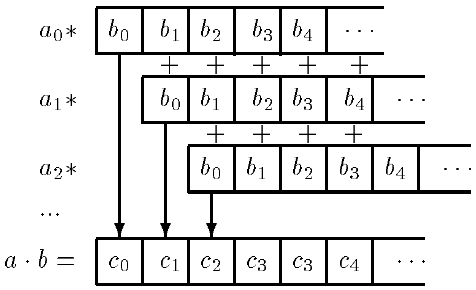
Für eine beliebig vorgegebene Reihe bedeutet das anschaulich:



Die Reihen aus $R[[x]]$ werden dann als `Stream R` intern dargestellt. Wir untersuchen jetzt exemplarisch die Implementation der Multiplikation zwischen Reihen.
Wie üblich, sagen wir, daß eine unendliche Folge von formalen Potenzreihen $(g_n(x))_{n\geq 0}$ **summierbar** ist, falls für jedes m endlich viele $g_n(x)$ eine Ordnung kleiner als m haben.
Die Tatsache, daß eine Folge $(g_n(x))_{n\geq 0}$ summierbar ist, bewirkt, daß jeder Koeffizient der Monome in der Reihe $\sum_{n\geq 0} g_n(x)$ durch eine **endliche** Summe berechnet werden kann. Wir möchten jetzt zwei formale Potenzreihen $a(x) = \sum_{n\geq 0} a_n x^n$ und $b(x) = \sum_{n\geq 0} b_n x^n$ miteinander multiplizieren. Man beachte, daß

$$\left(\sum_{n\geq 0} a_n x^n\right) \cdot \left(\sum_{n\geq 0} b_n x^n\right) = \sum_{n\geq 0} \left(a_n \cdot \sum_{i\geq 0} b_i x^{i+n}\right).$$

Da $ord(a_n \sum_{i\geq 0} b_i x^{i+n}) \geq n$ für jedes n, ist die Folge $(a_n \sum_{i\geq 0} b_i x^{i+n})_{n\geq 0}$ offensichtlich summierbar. Die zugrundeliegende Idee soll noch durch nachfolgendes Bild veranschaulicht werden:



In Scratchpad II

```
  prod (a:ST R, b:ST R, s:ST R, n:I): ST R ==
    delay cons (s.1, prod (a,b, (rest s) + a.n * b, n+1)
a * b == prod (a,b, a.1 * b, 0)
```

Wir benutzen hier den Stream `s` als Zwischenergebnis, um die im Bild dargestellte unendliche Summe der Streams `a.n * b` zu speichern. Diese Implementation hat einen wesentlichen Nachteil, der sich allerdings bei normalem Gebrauch kaum zeigt: bei der Berechnung jeder Komponente wird ein neuer Stream erzeugt und als Zwischenergebnis weitergegeben. Das heißt, daß beim Aufruf der n-ten Komponente des Streams im Speicher n Streams erzeugt werden, was einen hohen Speicherplatzverbrauch nach sich zieht.
Man kann sich leicht vorstellen, daß in unseren Berechnungen so oft die Multiplikation vorkommt, daß man gezwungen wird, einen weniger Speicher verbrauchenden Algorithmus zu implementieren.

5.3 Darstellung der Zykelindexreihen

Man sucht nach einer Möglichkeit, Zykelindikatormonome auf die Komponenten eines Streams so abzubilden, daß das Bestimmen der Koeffizienten eines beliebigen Monoms und die Definition verschiedener Operationen auf die Reihe effizient möglich ist.
Dazu ist es zunächst einmal notwendig, die Monome aus $R[[x_1, x_2, \ldots]]$ mit einer totalen Ordnung $\sqsubset$ zu versehen (die im Falle einer Variable offensichtlich war). Das geschieht mit Rücksicht auf die Bedeutung der Exponenten des Monoms in der kombinatorischen Interpretation der Zykelindexreihen.
Für ein beliebiges Zykelindikatormonom

$$m = k_m x_1^{\beta_1} x_2^{\beta_2} x_3^{\beta_3} \ldots$$

definieren wir den **Grad** von m durch

$$\Gamma(m) = \sum_{i \geq 1} i \cdot \beta_i.$$

$\Gamma(m)$ ist also die Anzahl der Punkte, auf den die Permutationen definiert sind, deren Zykeltyp durch die Exponenten $\beta_1, \beta_2, \ldots$ gegeben ist. Für diesen Grad gilt offensichtlich: $\Gamma(m_1 \cdot m_2) = \Gamma(m_1) + \Gamma(m_2)$ für alle $m_1, m_2 \neq 0$.
Man ordnet dann die Monome zuerst nach ihrem Grad. Monome mit gleichem Grad werden nach der inversen lexikographischen Ordnung auf den Exponenten geordnet.
Beispiel: $\Gamma(x_1^4 x_3^2) = 4 \cdot 1 + 2 \cdot 3 = 10$, $x_1^4 x_3^2 \sqsubset x_1 x_{10}$ und $x_1^5 x_2 x_3 \sqsubset x_1^4 x_3^2$.

Die Zykelindexreihen werden auf eine Darstellung von Reihen in einer Variable mittels der Abbildung

$$\sum_{n\geq 0}\frac{1}{n!}\sum_{\sigma\in S_n} Fix_n(F[\sigma])\cdot \underline{\mathbf{x}}^{\mathbf{cyc}(\sigma)}$$

$$\Downarrow$$

$$\sum_{n\geq 0}\left(\frac{1}{n!}\sum_{\sigma\in S_n} Fix_n(F[\sigma])\cdot \underline{\mathbf{x}}^{\mathbf{cyc}(\sigma)}\right)\gamma^n$$

transformiert.
Hier wird also eine Reihe aus $R[[x_1, x_2, \ldots]]$ auf eine Reihe aus $A[[\gamma]]$ *reduziert*, wobei jetzt $A = R[x_1, x_2, \ldots]$. Da für jeden Grad nur eine endliche Anzahl von Monomen existiert, sind die Komponenten des Streams endliche Gebilde. Zur Darstellung einer Zykelindexreihe benutzen wir folglich einen Stream, dessen Komponenten wiederum Listen sind, die alle Monome mit gleichem Grad enthalten. Innerhalb der Liste werden die Monome nach der angegebenen Ordnung abgespeichert. Wir können das anschaulich durch ein Bild für ein bekanntes Beispiel verdeutlichen.

$$Z_S(\underline{\mathbf{x}}) = 1 + x_1 + x_1^2 + x_2 + x_1^3 + x_1x_2 + x_3 + \ldots$$

$$\begin{array}{ll} Stream: & \begin{array}{|c|c|c|c|c}\hline 1 & x_1 & x_1^2, x_2 & x_1^3, x_1x_2, x_3 & \ldots \\ \hline \end{array} \\ Grad: & \quad 0 \quad 1 \qquad 2 \qquad\qquad 3 \end{array}$$

Das Verhalten der Ordnung bzgl. den üblichen Operationen wie der Addition und Multiplikation erlaubt, ähnliche Gedanken wie für den Fall in einer Variable anzuwenden. Durch die Struktur als Stream von Listen ist es auch möglich, die Anwendung der Rekursion weitgehend zu vermeiden, da für Listen iterative Algorithmen verwendbar sind. Das bedeutet i.a. einen niedrigeren Speicherverbrauch.

5.4 Domain Spezies

Es steht noch aus, für die einzelnen Komponenten der Repräsentation zu untersuchen, inwieweit sie vom System zur Verfügung gestellt werden. Man stellt zunächst fest, daß es keine Implementation von Zykelindikatormonomen gibt, die eine potentiell unbeschränkte (und durch die Anwendung von

Operationen wie die Multiplikation sich möglicherweise ändernde) Anzahl von Variablen haben. Danach muß die Verwaltung endlicher Listen von Monomen mit gleichem Grad ermöglicht werden. Für diese Listen müssen dann auch verschiedene Operationen definiert werden, die bei der Verarbeitung der Streams gebraucht werden, also z.B. die Multiplikation und die Addition, sowie die Ableitung oder verschiedene Vergleiche. Erst dann kann man die Darstellung als Stream verwenden und endlich eine Spezies als Zusammenfassung ihrer Zykelindexreihe und ihrer Zerlegung definieren. Die interne Darstellung einer Spezies hat dann die Form:

```
Rep:= Record (zyk:  ZykelIndexReihe R, zerl:  String)
```

Direkt auf Objekte des Typs `Spezies` können folgende Funktionen angewendet werden:

```
cycser      :  $           -> ZIR
```

Der Aufruf `cycser (f)` liefert die Zykelindexreihe der Spezies `f`.

```
cardnum     :  ($, I)      -> R
cardtyp     :  ($, I)      -> R
gfnum       :  $           -> ERZ
gftyp       :  $           -> ERZ
```

Durch `cardnum (f,n)` bzw. `cardtyp(f,n)` bekommt man die Anzahl der numerierten Strukturen bzw. der Isomorphie-Typen der Spezies `f` auf `n` Elementen. `gfnum(f)` bzw. `gftyp(f)` liefern die Kardinalität bzw. die typerzeugende Funktion von `f`.

```
"*"         :  ($, $)      -> $
"**"        :  ($, I)      -> $
"+"         :  ($, $)      -> $
deriv       :  $           -> $
point       :  $           -> $
cart        :  ($, $)      -> $
con         :  ($, NNI)    -> $
subst       :  ($, $)      -> $
norm        :  $           -> $
create      :  (ZIR, S)    -> $
```

Die Funktionen entsprechen einigen auf Spezies definierten Operationen, wie dem Produkt, der Summe, der Ableitung, der Punktierung, dem kartesischen Produkt, der Konzentrierung, der Substitution und der Normierung.

Nur wenige wurden im 3. Abschnitt vorgestellt. Die Funktion `create` bietet die Möglichkeit, eine Spezies aus ihrer Zykelindexreihe und ihrem Namen zu definieren.
Ferner wurden Funktionen definiert, die auf Zykelindexreihen wirken. Sie befinden sich im Package `RecursiveNumericOfSpezies`.

```
Rtree        :   ZIR          -> ZIR
Gcatalantree :   ZIR          -> ZIR
MEN          :   ZIR          -> ZIR
MSN          :   ZIR          -> ZIR
MCN          :   ZIR          -> ZIR
```

Die Funktion `Rtree` bestimmt aus der Zykelindexreihe der Spezies R die Zykelindexreihe der Spezies A, die folgender Gleichung genügt

$$A = X \cdot (R \circ A)$$

Die Funktion `Gcatalantree` bestimmt die Zykelindexreihe der Spezies A, für die gilt

$$A = X + (G \circ A)$$

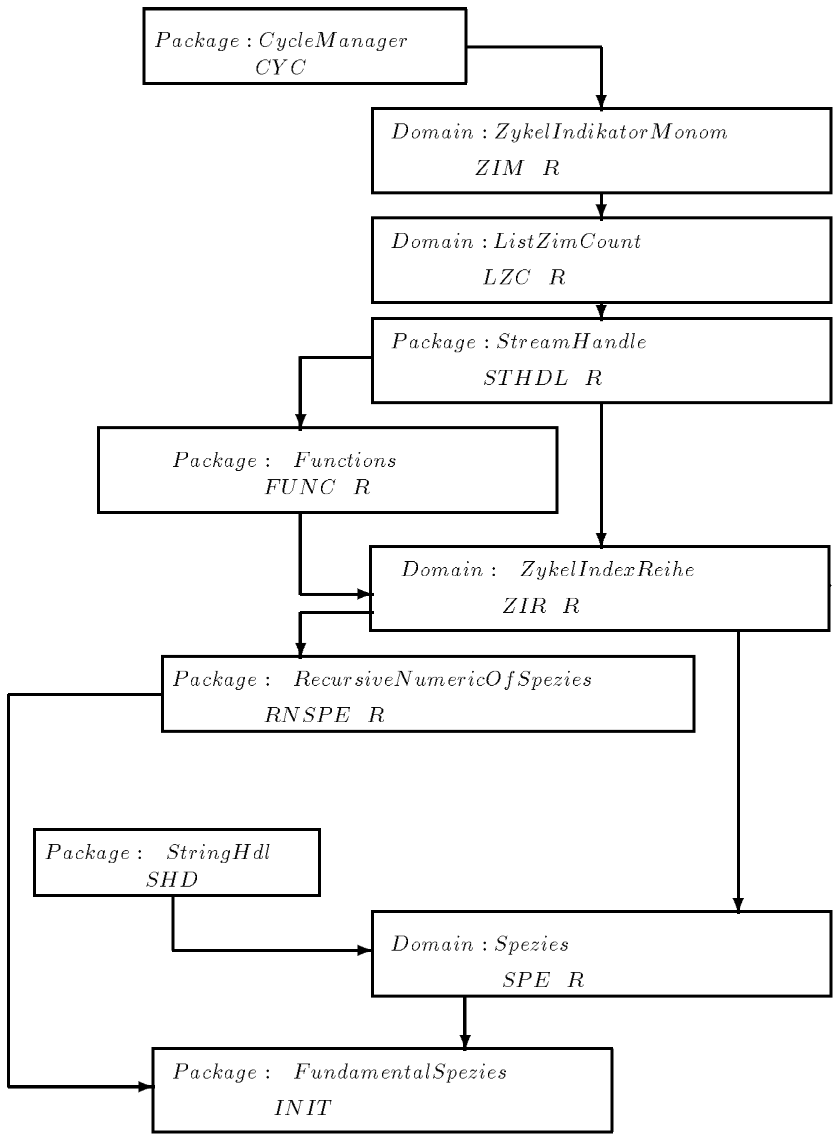
und die restlichen drei Funktionen lösen die Gleichungen

$$M = E \circ N \ , \ M = S \circ N \ , \ M = C \circ N$$

nach der Zykelindexreihe der Spezies N auf (hierbei ist C die Spezies der zyklischen Permutationen).

6 Software-Struktur

Nachdem wir im Abschnitt 5 einige Aspekte der Implementierung vorgestellt haben, möchten wir den gesamten Aufbau der Programme verdeutlichen. Wie schon gesagt, wird der Benutzer im wesentlichen mit Objekten vom Domain `Spezies` im System arbeiten. Dazu gehören aber einige andere Domains: Im Domain `ZIR` werden Zykelindikatormonome zuerst implementiert, in `LZC` werden dann endliche Listen solcher Monome verwaltet und in `ZIR` werden Zykelindexreihen als Datenstruktur definiert und mit Operationen versehen. Das Domain `Spezies` baut dann auf `ZIR` auf, um den Datentyp für Spezies zu definieren. Diese Zusammenhänge werden durch folgendes Bild veranschaulicht (ein Pfeil deutet die Abhängigkeit von Domains an).


Package : CycleManager
CYC
Domain : ZykelIndikatorMonom
ZIM R
Domain : ListZimCount
LZC R
Package : StreamHandle
STHDL R
Package : Functions
FUNC R
Domain : ZykelIndexReihe
ZIR R
Package : RecursiveNumericOfSpezies
RNSPE R
Package : StringHdl
SHD
Domain : Spezies
SPE R
Package : FundamentalSpezies
INIT

Außer Domains werden auch Packages definiert, die einige spezielle Funktionen beinhalten. Die Packages `CYC` und `SHD` stellen nur Hilfsfunktionen bereit, die für den Benutzer unwichtig sind. In den Packages `STHDL` und `FUNC` werden auf der Ebene der Streams die Funktionen definiert, die im Domain `ZIR` für Zykelindexreihen verwendet werden. Im Package `INIT` werden einige konkrete Spezies definiert und dem Benutzer zur Vefügung gestellt. Diese werden im folgendem Abschnitt vorgestellt.

7 Eine Beispielsitzung

Innerhalb dieses abschließenden Abschnitts soll anhand einer Beispielsitzung der praktische Umgang mit dem System erklärt werden. Im Package `INIT` befinden sich die vordefinierten Spezies. Durch Eingabe von

```
)show INIT
```

erscheinen auf dem Bildschirm die einzelnen Signaturen.

```
Graph        :   -> SPE RN    -- Graph
OGraph       :   -> SPE RN    -- Gerichteter Graph
LinOr        :   -> SPE RN    -- Lineare Ordnung
Zero         :   -> SPE RN    -- Leere Spezies
Emenge       :   -> SPE RN    -- Menge E
Emenget      :   -> SPEP      -- Gewichtete Menge
Teilmenge2   :   -> SPE RN    -- Zweielem.  Teilmenge
Teilmenge    :   -> SPE RN    -- Teilmenge
Teilmenget   :   -> SPEP      -- Gewichtete Teilmenge
SinTo        :   -> SPE RN    -- Singleton
Perm         :   -> SPE RN    -- Permutation
One          :   -> SPE RN    -- Leere Menge
CycPerm      :   -> SPE RN    -- Zyklische Permutation
IdEndo       :   -> SPE RN    -- Idempotente Endofunktion
RootTree     :   -> SPE RN    -- Wurzelbaum
BinTree      :   -> SPE RN    -- Binaerer Baum
StabEndo     :   -> SPE RN    -- Stabile Endofunktion
WirbelTier   :   -> SPE RN    -- Wirbeltier
EndoFix      :   -> SPE RN    -- Endofunktion mit Fixpunkt
PermOFix     :   -> SPE RN    -- Permutation ohne Fixpunkt
CatTree      :   -> SPE RN    -- Catalan-Baum
```

Wir möchten jetzt eine Sitzung beschreiben, wodurch die in (1) als Beispiel definierte Spezies F behandelt wird.
Dazu betrachten wir die angegebenen Zerlegungen und fangen mit der Bestimmung der Zykelindexreihe der Spezies $Btree$ an. Zuerst benötigen wir dazu die Spezies E_2 der zweielementigen Mengen. Als schon definierte Spezies haben wir die Uniforme Spezies zur Verfügung. Davon weisen wir die Zykelindexreihe der Variable `e` zu.

```
e:= cycser Emenge()
```

$$1 + x_1 + \frac{1}{2}\, {x_1}^2 + \frac{1}{2}\, x_2 + \frac{1}{6}\, {x_1}^3 + \frac{1}{2}\, x_1\, x_2 + \frac{1}{3}\, x_3 + \ldots \tag{1}$$

`Type: ZykelIndexReihe Fraction Integer`

Die Zykelindexreihe der Spezies E_2 besteht nur aus den Termen von `e`, die den Grad 2 besitzen.

```
e2:=con(e,2)
```

$$\frac{1}{2}\, {x_1}^2 + \frac{1}{2}\, x_2 + \ldots \tag{2}$$

`Type: ZykelIndexReihe Fraction Integer`

Jetzt können wir die Funktion `Rtree` aus dem Package `RNSPE` anwenden, welche die Lösung der Gleichung $Btree = X * ((1 + E_2) \circ Btree)$ (auf der Ebene der Zykelindexreihen) bestimmt.

```
btree:= Rtree (1+e2)
```

$$x_1 + \frac{1}{2}\, {x_1}^3 + \frac{1}{2}\, x_1\, x_2 + \ldots \tag{3}$$

`Type: ZykelIndexReihe Fraction Integer`

Wir können dann die Spezies $Btree$ definieren, die durch die Reihe $btree$ beschrieben ist. Die Spezies bekommt auch einen passenden Namen.

```
Btree:= create (btree, "Btree")
```

(4) "Btree"

`Type: Spezies Fraction Integer`

Jetzt, da wir die Spezies $Btree$ zur Verfügung haben, können die F-Kontraktionen durch die Zerlegung $Fkon = X * Btree$ folgendermaßen definiert werden

```
Fkon:= SinTo() * Btree
```

(5) "Singleton * Btree"

Type: Spezies Fraction Integer

und somit die gewünschte Spezies F.

```
F := subst(Perm(), Fkon)
```

(6) "(Perm o Singleton * Btree)"

Type: Spezies Fraction Integer

Man bemerkt, daß die eigentliche Ausgabe einer Spezies nur ihr Name ist, der die Zerlegung angibt, die der Definition zugrundeliegt.
Um die Zykelindexreihe der Spezies F zu bekommen (etwa bis zu den Termen vom Grad 5) verwenden wir

```
)set stream calc 6
cycser F
```

(7) $$1 + {x_1}^2 + \frac{3}{2}\,{x_1}^4 + \frac{1}{2}\,{x_1}^2\,x_2 + {x_2}^2 + \ldots$$

Type: ZykelIndexReihe Fraction Integer

und ähnlich für die Kardinalität der Spezies

```
gfnum F
```

(8) $$1 + x^2 + \frac{3}{2}\,x^4 + \frac{5}{2}\,x^6 + O\left(x^7\right)$$

Type: UnivariateTaylorSeries(Fraction Integer,x,0)

Die Gradschranke kann jederzeit verändert werden, um weitere Werte ausrechnen zu lassen.
Betrachten wir z.B. die Terme vom Grad 6

```
)set stream calc 7
con(cycser F,6)
```

(9) $$\frac{5}{2}\,{x_1}^6 + \frac{3}{2}\,{x_1}^4\,x_2 + {x_1}^2\,{x_2}^2 + {x_3}^2 + \ldots$$

Type: ZykelIndexReihe Fraction Integer

Daraus kann man leicht z.B. die Information ablesen, wieviele F-Strukturen auf 6 Elementen existieren. Das kann man auch direkt durch folgenden Befehl abfragen.

```
cardnum (F,6)
```

(10) 1800

`Type: Fraction Integer`

Die Anzahl der unnumerierten Strukturen ergibt sich aus obigen Termen durch die Summe $\frac{5}{2} + \frac{3}{2} + 1 + 1 = 6$.

```
cardtyp (F,6)
```

(11) 6

`Type: Fraction Integer`

Aus (9) kann man mit der Definition der Zykelindexreihen ((13) auf Seite 20) weitere Informationen gewinnen. Die Monome in (9) besagen, daß nur Permutationen vom Typ (6), $(4,1)$, $(2,2)$, $(0,0,2)$ als Automorphismen von F-Strukturen auf 6 Elementen auftreten. Ferner bedeutet der Koeffizient im Monom $\frac{3}{2}\, {x_1}^4\, x_2$, daß genau $(1^4 4! 2^1 1!)\frac{3}{2} = 72$ Strukturen aus $F[\,\underline{6}\,]$ von einer Permutation vom Typ $(4,1)$ nicht verändert werden.
Damit haben wir die vorgestellten Abzählprobleme für die betrachteten Endofunktionen algorithmisch gelöst.

Literatur

[De] **H. Décoste**, *Séries indicatrices d'espèces pondérées et q-analogues*, Ph.D., Université de Montréal, **1989**.

[HP] **J. Hornegger, R.Pirastu**, *Die Theorie der Spezies und die Implementation ihrer Zykelindexreihen auf dem Computer-Algebra-System Scratchpad II*, Studienarbeit, IMMD I der Universität Erlangen-Nürnberg, **1991**.

[Jo] **A. Joyal**, *Une théorie de series formelles*, Adv. in Math. 42, **1983**, pp. 1-82.

[La1] **G. Labelle**, *Some New Combinatorial Methods in the Theory of Species*, Lecture Notes in Mathematics No.1234, Springer Verlag, New York/Berlin, **1986**, pp. 192-206.

[La2] **J. Labelle**, *Application diverses de la théorie combinatoire des espèces de structures*, Ann. sc. math. Québec, Vol. 7, No. 1, **1983**, pp. 59-94.

[Ni] **A. Nickel**, *Abzählung und Konstruktion diskreter Strukturen und die Theorie der Spezies*, Diplomarbeit, Lehrstuhl II für Mathematik, Universität Bayreuth, **1989**.

[PR] **G. Pólya, R. C. Read**, *Combinatorial Enumeration of Groups, Graphs, and Chemical Compounds*, Springer Verlag, **1987**.

[Su] **R. Sutor**, *The Scratchpad Computer Algebra System Interactive Environment Users Guide*, New York, **1989**.

[UG] *Scratchpad II User's Guide, (IBM).*

ADRESSE DER AUTOREN:

Joachim Hornegger
Gartenstr. 39
8551 Wiesenthau
hornegge@informatik.uni-erlangen.de

Roberto Pirastu
Badstr. 33
8520 Erlangen
ropirast@immd1.informatik.uni-erlangen.de