Oktober 2014

Computeralgebra Rundbrief

> Ausgabe 55

- Hintertüren und Schwächen im kryptographischen Standard SP 800-90A
- GTPack - ein Mathematica Paket für Gruppentheorie in der Festkörperphysik
- The GAP package SingularInterface
- 3D-Grafik in der Schule mit Computeralgebra

www.fachgruppe-computeralgebra.de

Impressum

Der Computeralgebra-Rundbrief wird herausgegeben von der Fachgruppe Computeralgebra der GI in Kooperation mit der DMV und der GAMM (verantwortlicher Redakteur: Prof. Dr. Michael Cuntz, car@mathematik.de)

Der Computeralgebra-Rundbrief erscheint halbjährlich, Redaktionsschluss 15.02. und 15.09. ISSN 0933-5994. Mitglieder der Fachgruppe Computeralgebra erhalten je ein Exemplar dieses Rundbriefs im Rahmen ihrer Mitgliedschaft. Fachgruppe Computeralgebra im Internet: http://www.fachgruppe-computeralgebra.de.

Konferenzankündigungen, Mitteilungen, einzurichtende Links, Manuskripte und Anzeigenwünsche bitte an den verantwortlichen Redakteur.

GI (Gesellschaft für Informatik e.V.)
Wissenschaftszentrum
Ahrstr. 45
53175 Bonn
Telefon 0228-302-145
Telefax 0228-302-167
gs@gi-ev.de
http://www.gi-ev.de

DMV (Deutsche Mathematiker-Vereinigung e.V.)
Mohrenstraße 39
10117 Berlin
Telefon 030-20377-306
Telefax 030-20377-307
dmv@wias-berlin.de
http://www.dmv.mathematik.de

GAMM (Gesellschaft für Angewandte Mathematik und Mechanik e.V.)
Technische Universität Dresden
Institut für Statik und Dynamik der Tragwerke
01062 Dresden
Telefon 0351-463-33448
Telefax 0351-463-37086
GAMM@mailbox.tu-dresden.de
http://www.gamm-ev.de

Computeralgebra-Rundbrief

Nr. 55 Oktober 2014

Inhaltsverzeichnis

Mitteilungen der Sprecher

Liebe Mitglieder der Fachgruppe Computeralgebra,

die Herbstsitzung der Fachgruppenleitung fand am 26. September 2014 an der Leibniz Universität Hannover statt.

Zu den Neuerungen innerhalb der Fachgruppe zählt der Rundbrief 55, den Sie in den Händen halten. Neu daran ist aber nicht nur der Inhalt, sondern, wie auch schon beim Rundbrief 54, die Herstellung. Seit dem Frühjahr geben wir den Rundbrief bei einer neuen Druckerei in Auftrag. Der Vorteil liegt bei gleichbleibender Qualität in einer deutlichen Kostenersparnis. Wir danken Thomas Hahn, der den Wechsel energisch vorangetrieben hat. Dies gibt uns einen größeren Spielraum bei der Förderung junger Mitglieder und anderen Aktivitäten.

Die Umstellung der Mitgliederverwaltung von der DMV auf die GI, über die schon im Rundbrief 54 berichtet wurde, ist nun erfolgreich abgeschlossen. Die Fachgruppenleitung dankt den Geschäftsstellen der DMV und der GI ganz herzlich für ihren Einsatz und die reibungslose Durchführung.

Die Mitgliederzahl der Fachgruppe liegt zur Zeit etwas unter 400. Damit sie bald über 400 liegt, wäre es hilfreich, wenn Sie potentielle Interessenten auf die Mitgliedschaft in der Fachgruppe aufmerksam machen könnten. Die Ermöglichung einer kostenfreien, befristeten Mitgliedschaft für Studierende und Doktoranden wurde auf dem letzten Fachgruppenleitungstreffen angedacht. Über die weitere Entwicklung halten wir sie auf dem Laufenden.

Innerhalb der GI gibt es zwei Neuigkeiten. Thomas Wilke (Christian-Albrechts-Universität zu Kiel) wurde Anfang des Jahres zum Sprecher des Fachbereichs Grundlagen der Informatik der GI, dem unsere Fachgruppe angehört, gewählt. Herr Wilke nahm an der Computeralgebra-Tagung im Mai teil, wo ihn Mitglieder der Fachgruppe kennenlernen konnten. Auf Initiative von Herrn Wilke findet Anfang 2015 ein Treffen der Sprecher der Fachgruppen im Fachbereich Grundlagen der Informatik zwecks Vernetzung der Fachgruppen statt.

In der GI wird aktuell auch die Einrichtung einer neuen Fachgruppe „NumSim“ diskutiert. Diese Fachgruppe soll sich mit den Themen Numerische Simulation und High Performance Computing beschäftigen. Die Arbeitsschwerpunkte können unter anderem die Entwicklung und Parallelisierung numerischer Algorithmen und die Anwendung neuer Architekturen sowie Hardware sein. Weitere Informationen erhalten Sie von Patrick Diehl (me@diehlpk.de) oder unter der Mailingliste

https://mail.gi-ev.de/mailman/listinfo/numsim

Zum Schluß bitten wir wie immer alle Mitglieder der Fachgruppe, die Rundbrief-Redaktion mit Informationen, Themenvorschlägen, Beiträgen, Berichte über Promotionen und Habilitationen, Hinweisen auf Bücher, auf Programmpakete und auf Tagungen etc. zu unterstützen.

Da das vorliegende Heft relativ umfangreich ist, wollen wir diese Mitteilungen nicht weiter in die Länge ziehen und Sie herzlich zum Weiterblättern einladen.

Florian Heß *Gregor Kemper*

Tagungen der Fachgruppe

Tagung in Kassel, 2014

Tagung der Fachgruppe Computeralgebra in Kassel, 15. – 17.05.2014

`http://www.fachgruppe-computeralgebra.de/tagung-kassel-2014/`

Von 15. bis 17. Mai 2014 fand die sechste Tagung der Fachgruppe in Kassel statt. Auch diese Ausgabe der erfolgreichen Konferenzreihe war mit fast 50 Teilnehmern gut besucht.
Es gab fünf einstündige Hauptvorträge: Severin Neumann (Passau), der Gewinner des Nachwuchspreises von 2012, eröffnete die Tagung mit einem Vortrag über parallele und verteilte Algorithmen zur Berechnung von Gröbnerbasen. Jens Zumbrägel (Dresden) berichtete über neue Algorithmen für das diskrete Logarithmus-Problem in Körpern kleiner Charakteristik, und Bettina Eick (Braunschweig) gab einen Überblick über den Stand der Forschung in der algorithmischen Gruppentheorie. Am Abschlusstag zeigte Jan Steffen Müller (Oldenburg) einen Querschnitt durch die Theorie diophantischer Gleichungen und diskutierte den Einsatz von Computeralgebra und geometrischen Methoden bei Lösbarkeitsfragen. Raymond Hemmecke (München) referierte über algebraische Methoden in der ganzzahligen Optimierung.

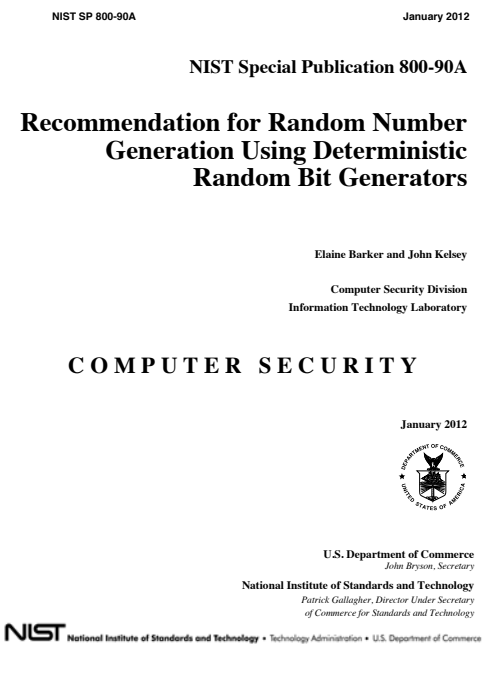
Das Programm wurde durch 20 eingereichte Beiträge, zwei Software-Präsentationen und eine Diskussionsrunde über die Fachgruppe und ihre aktuellen Aktivitäten vervollständigt. Alle Vortragenden nahmen Rücksicht auf die heterogene Zusammensetzung des Publikums und begannen ihre Präsentationen mit einer allgemeinverständlichen Einführung in die Thematik. Diese weitgehende Zugänglichkeit der Beiträge auch für Nicht-Spezialisten wurde allgemein begrüßt.
Beim gemeinsamen Abendessen, das am Freitag im Ristorante La Piazza stattfand, diskutierten die anwesenden Mitglieder der Fachgruppenleitung – unterstützt durch Werner Bley – über die Vortragsbeiträge, um den Nachwuchs-Preisträger auszuwählen, was aufgrund der hohen Qualität zahlreicher Präsentationen schwerfiel. Schließlich setzte sich Christian Eder (Kaiserslautern) mit seinem schönen Vortrag über signatur-basierte Gröbnerbasis-Algorithmen durch. Herr Eder hat 2012 bei Gerhard Pfister promoviert und danach im Team von Jean-Charles Faugère gearbeitet. Wie gewohnt war mit der Auszeichnung ein Preisgeld von 500 Euro verbunden und die Einladung, bei der nächsten Tagung der Fachgruppe einen Hauptvortrag zu halten.
Die Fachgruppe dankt dem Fachbereich Grundlagen der Informatik der GI für das Stiften des Preisgeldes sowie den Sponsoren Additive, Maplesoft und Springer-Verlag für ihre großzügige Unterstützung. Besonders herzlicher Dank gebührt Wolfram Koepf und seinem Team für die exzellente Organisation der Tagung.

Eva Zerz (Aachen)

Hintertüren und Schwächen im kryptographischen Standard SP 800-90A[1]

M. Fischlin
(TU Darmstadt)

marc.fischlin@cryptoplexity.de



NIST SP 800-90A January 2012

NIST Special Publication 800-90A

Recommendation for Random Number Generation Using Deterministic Random Bit Generators

Elaine Barker and John Kelsey

Computer Security Division
Information Technology Laboratory

COMPUTER SECURITY

January 2012

U.S. Department of Commerce
John Bryson, Secretary

National Institute of Standards and Technology
Patrick Gallagher, Director Under Secretary of Commerce for Standards and Technology

NIST National Institute of Standards and Technology • Technology Administration • U.S. Department of Commerce

ii

Die durch Edward Snowden angestoßenen Enthüllungen liefern seit Juni 2013 immer mehr bemerkenswerte Details darüber, welche Methoden Geheimdienste zur Sammlung und Verwertung von personenbezogenen Daten anwenden. Im vorliegenden Artikel diskutieren wir speziell den Fall des kryptographischen Standards SP 800-90, bei dem der amerikanische Geheimdienst NSA (National Security Agency) gemäß eines Artikels der New York Times vorsätzlich Schwächen eingebaut hatte. Wir erläutern anhand der mathematischen Details, wie diese Hintertür aussieht und wie die Schwächen einzuordnen sind.

Kryptographische Standards

Technische Standards dienen der Vereinheitlichung von Normen und Verfahren und sollen dadurch die Interoperabilität fördern. Auch in der Kryptographie gibt es zahlreiche Standards und Standardisierungsorganisationen. Eine wichtige Institution ist hier neben der International Organization for Standardization (ISO) und der International Electrotechnical Commission (IEC) das amerikanische National Institute of Standards and Technology (NIST). NIST setzt mit seiner Reihe „Special Publications (800 series)“ Standards speziell für Themen zur Informationssicherheit (siehe [10]). So beschreiben beispielsweise die Standards SP 800-56A bis SP 800-56C, wie zwei Teilnehmer basierend auf dem Diskreten Logarithmus- oder dem Faktorisierungsproblem einen gemeinsamen kryptographischen Schlüssel aushandeln können.

Die von NIST zunächst nur für die Vereinigten Staaten entwickelten Standards etablieren sich auch oft zu internationalen de-facto-Standards und werden teilweise in ISO/IEC-Standards übernommen. Dabei ist bemerkenswert, dass die Standards im Bereich der IT-Sicherheit oft ein weiteres Attribut zugewiesen bekommen, nämlich, dass standardisierte Verfahren auch besondere Sicherheitsgarantien mit sich bringen. Die Erfahrung hat allerdings gezeigt, dass diese Annahme im Allgemeinen zu optimistisch ist. Dies gilt um so mehr, als wir heute wissen, dass Geheimdienste absichtlich schwache Standards vorgeschlagen haben. Wir betrachten hier mit dem Standard SP 800-90 der NIST einen solchen Fall.

Schwächen im Standard SP 800-90

Der Standard SP 800-90 beschäftigt sich mit der Erzeugung von sogenannten Pseudozufallszahlen, also quasi zufälligen Werten. Dieses Konzept wird ausführlicher

[1] Dieser Artikel erschien in den DMV-Mitteilungen 2014-0012, S. 18-22. Wiederabdruck mit freundlicher Genehmigung der Deutschen Mathematiker-Vereinigung.

in Abschnitt diskutiert. Der NIST-Standard SP 800-90 bzw. der ergänzende Standard SP 800-90A diskutiert Möglichkeiten, wie man solche Pseudozufallswerte auf vermeintliche sichere Weise genererieren kann. Teile des NIST-Standards finden sich auch im ISO/IEC Standard 18031:2011 [8] wieder.

Was war passiert?
Schon kurz nach Veröffentlichung der ersten Version des Standards SP 800-90 im Jahr 2006 – die Version SP 800-90A ergänzte den ursprünglichen Standard und die aktuelle Version datiert auf Januar 2012 – wurde Kritik am Standard öffentlich. Wissenschaftliche Arbeiten [15, 2, 17] aus den Jahren 2006 und 2007 zeigten, dass eines der vorgeschlagenen Verfahren, der Pseudozufallsgenerator Dual_EC_DRBG, nicht die gewohnten Sicherheitsstandards erfüllte. Schon damals ließen Kommentatoren wie Bruce Schneier anklingen, dass es sich dabei um absichtlich implementierte Schwächen handeln könnte [14]. Tatsächlich bestätigt der aktuelle Standard SP 800-90A die Mitarbeit der NSA an der Erstellung, wobei die Zusammenarbeit mit den IT-Sicherheitsexperten der NSA an solchen Standards *per se* nicht ungewöhnlich ist. Erst im Jahr 2013 wurde im Rahmen der Enthüllungen um Edward Snowden durch einen Artikel der New York Times [12] bestätigt, dass die NSA Schwächen absichtlich eingebaut hat.

Inzwischen rät NIST selbst explizit von der Verwendung des Generators ab [11]. In einigen kommerziellen Produkten wurde der Generator implementiert, aber bis auf zwei Ausnahmen, dem BSAFE-Toolkit und dem Data Protection Manager der Firma RSA, nicht als der standardmäßig verwendete Generator eingesetzt, sondern nur als Option [3]. Der Generator sollte natürlich nicht mehr verwendet werden. RSA selbst empfiehlt inzwischen, in dem betroffenen Produkt auf einen anderen Generator umzusteigen [4]. Bemerkenswert ist, dass beide Empfehlungen, sowohl die von NIST als auch die von RSA, erst nach der Veröffentlichung des Artikels der New York Times im September 2013 herausgegeben wurden, obwohl die ersten Kritiken an der kryptographischen Sicherheit des Dual_EC_DRBG bereits 2006 und 2007 veröffentlicht wurden.

Hintergrund: Pseudozufallsgeneratoren
Zufallswerte sind essenziell für die Sicherheit kryptographischer Verfahren. Sie treten an zahlreichen Stellen in Verfahren auf, angefangen mit der Erzeugung der geheimen Schlüssel, über die Wahl von Initialisierungsvektoren für Verschlüsselungen, bis hin zur Generierung von unvorhersagbaren Fragen in Challenge-Response-Verfahren zur Authentisierung. Klassische Rechner können aber in der Regel keine „echten“ Zufallswerte erzeugen, da sie deterministisch arbeiten, also ihre Ergebnisse eindeutig vorherbestimmt sind. Die folgende Aussage, die dem Mathematiker und Informatiker John von Neumann zugeordnet wird, fasst dieses Dilemma pointiert zusammen:

> Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin. (John von Neumann)

Der Entropie-Begriff von Shannon [16], den er in seiner wegweisenden Arbeit von 1948 über Informationstheorie einführte, bestätigt diese Sichtweise. Für eine (diskrete) Zufallsvariable X ist die Shannon-Entropie definiert als

$$H(X) := -\sum_x \text{Prob}[X = x] \cdot \log_2 \text{Prob}[X = x],$$

wobei x die möglichen Werte von X durchläuft. Intuitiv gibt die Shannon-Entropie an, wieviele Bits man durchschnittlich benötigt, um den Ausgang einer Stichprobe von X zu kommunizieren. Die Shannon-Entropie ist genau dann maximal, wenn X uniform verteilt ist.

Für jede Zufallsvariable und jede mathematische Funktion C (die der Leser hier als einen deterministischen Computer ansehen kann) gilt

$$H(X) \geq H(C(X)),$$

wobei $C(X)$ die Zufallsvariable beschreibt, die zuerst eine Stichprobe von X wählt und dann C anwendet. Folglich kann kein (deterministischer) Computer im Sinne der Shannon-Entropie „zusätzlichen Zufall“ erzeugen, auch wenn man ihm „etwas Zufall“ in Form von X zur Verfügung stellt.

Die Kryptographie löst das Dilemma, indem sie die Anforderung an die Zufallswerte abschwächt, und lediglich *pseudozufällige* Werte verlangt. Pseudozufällige Werte erreichen zwar keine maximale Shannon-Entropie, sehen aber dennoch für alle praktischen Zwecke wie echt zufällig aus. In der Kryptographie wird dies formal in die Theorie sogenannter *Pseudozufallsgeneratoren* eingebettet. Ein Pseudozufallsgenerator G ist ein deterministischer (und effizienter) Algorithmus, der eine kurze, echt zufällige Eingabe – üblicherweise als binäre Zeichenkette r aus der Menge $\{0,1\}^*$ aller endlichen Bitfolgen kodiert – in eine längere Ausgabe transformiert, sodass diese Ausgabe „wie zufällig“ aussieht. Letzteres wird mit Hilfe des Begriffs der Ununterscheidbarkeit formalisiert, soll hier aber nicht weiter ausgeführt werden. Eine Einführung gibt das Buch von Goldreich [6].

In der Notation der Entropie verlangt man, dass ein Pseudozufallsgenerator maximale *Pseudo-Entropie* [7] besitzt. Eine Zufallsvariable X hat Pseudo-Entropie

$$H^{\text{HILL}}(X) \geq k,$$

wenn sie ununterscheidbar (im obigen Sinne) von einer Zufallsvariablen Y mit Shannon-Entropie $H(Y) \geq k$ ist. Folglich hat ein Pseudozufallsgenerator, der für n-Bit-Eingaben längere $\ell(n)$-Bit-Ausgaben erzeugt, Pseudo-Entropie $\ell(n)$, während die Shannon-Entropie maximal n sein kann. Für die meisten kryptographischen Anwendungen genügt nachweislich eine hohe Pseudo-Entropie, um Sicherheit zu gewährleisten.

Ob sichere Pseudozufallsgeneratoren existieren, lässt sich mit gegenwärtigen Methoden nicht nachweisen. Man kann solche Generatoren aber unter sehr schwachen kryptographischen Annahmen ableiten [7],

unter anderem auch unter gängigen Sicherheitsannahmen wie der Schwierigkeit, die diskrete Exponentiation effizient zu invertieren (dem sogenannten Diskreten-Logarithmus-Problem). Daher gilt es im Augenblick als vernünftig, die Existenz solcher Generatoren anzunehmen. Gleichzeitig muss man aber besondere Vorsicht walten lassen, wenn man einen Generator entwickelt.

Wie setzen moderne Computer nun solche Pseudozufallsgeneratoren ein? Üblicherweise wird der Computer mit einem kurzen, möglichst zufälligen Wert initialisiert, beispielsweise durch Messungen von rauschanfälligen Komponenten während des Starts, oder durch „zufällige“ Eingaben des Anwenders. Bei Bedarf erzeugt der Generator aus seinem Startwert hinreichend viele Pseudozufallsbits, gibt einen Teil als Ausgabe aus, und aktualisiert gleichzeitig seinen Startwert mit einem anderen Teil der generierten Bits. Die tatsächlichen Verfahren variieren dieses Schema auf vielfältige Weise.

Der Generator Dual_EC_DRBG

Der Standard SP 800-90 beschreibt verschiedene, vermeintlich sichere Pseudozufallsgeneratoren. Eine Variante ist der Generator Dual_EC_DRBG, der auf elliptischen Kurven beruht.

Elliptische Kurven werden wegen ihrer guten Sicherheitseigenschaften – kurze Schlüssel und Resistenz gegen bekannte Diskrete Logarithmus-Verfahren – verstärkt in der Kryptographie eingesetzt. Im vorliegenden Fall wurden im Standard eine Primzahl p sowie die Konstanten a und b spezifiziert (z. B. ist $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} + 1$), die eine elliptische Kurve definieren. Die Kurvenpunkte

$$E_{a,b}(\mathbb{F}_p) = \big\{(x, y) \in \mathbb{F}_p^2 \mid y^2 = x^3 + ax + b\big\},$$

bilden zusammen mit einer entsprechend definierten Addition eine Gruppe, die für die Werte hier sogar von primer Ordnung ist. Die x-Koordinaten von Kurvenpunkten können nach Wahl von p mit 256 Bits dargestellt werden. Der Standard spezifiziert zusätzlich zwei Punkte $P = (x_P, y_P)$ und $Q = (x_Q, y_Q)$, die beide verschieden vom „Punkt im Unendlichen“ $\mathcal{O}$, dem neutralen Element der Gruppe, sind. Der Punkt P ist ein Erzeuger der additiven Gruppe.

Der Dual_EC_DRBG wird mit einem zufälligen Startwert s_0 aus $\mathbb{F}_p$ initialisiert. Bei Bedarf generiert das Verfahren 240 Pseudozufallsbits, indem es ausgehend vom aktuellen Wert $s_i \in \mathbb{F}_p$ den Kurvenpunkt s_iP berechnet und s_{i+1} als die x-Koordinate x-coord(s_iP) des Punktes für die nächste Iteration speichert. Ebenso berechnet das Verfahren den Kurvenpunkt $s_{i+1}Q$ und gibt nun die untersten 240 Bits der 256 Bits der x-Koordinate des Punktes (in einer üblichen Binärdarstellung) als Pseudozufallsbits aus. Benötigt man mehr Bits, wiederholt man das Verfahren hinreichend oft.

Das Design des Generators beruht auf dem sogenannten *Diffie-Hellman-Entscheidungsproblem*. Beim bekannten Schlüsselaustauschverfahren von Diffie und Hellman [5] erzeugen die zwei Teilnehmer Alice und Bob einen gemeinsamen geheimen Schlüssel in einer elliptischen Kurve mit Erzeuger P, indem Alice für zufälliges q einen Punkt $Q = qP$ berechnet und Q an Bob sendet, und Bob $R = rP$ für zufälliges r berechnet, und mit R antwortet. Alice kann nun lokal $S = qR = (qr)P$ berechnen, und Bob ebenfalls $S = rQ = r(qP) = (qr)P$, sodass S der gemeinsame Schlüssel wird.

Das klassische *Diffie-Hellman-Berechnungsproblem*, auf dessen Sicherheit das Diffie-Hellman-Verfahren beruht, besagt nun, dass es für einen Angreifer schwierig ist, ebenfalls den Schlüssel S aus den ausgetauschten Daten P, Q, R zu berechnen. Das DH-Entscheidungsproblem fordert nun sogar, dass man den Schlüssel S nicht einmal von einem unabhängig gewählten Kurvenpunkt unterscheiden kann: Gegeben vier Kurvenpunkte (P, Q, R, S) entscheide man, ob S ein unabhängiger und uniform verteilter Punkt ist, oder ob $S = rQ$ für $R = rP$ gilt. Beide Probleme, das Berechnungsproblem und das Entscheidungsproblem, gelten in der Kryptographie als schwierig, und das DH-Verfahren somit als sicher. Weitere Informationen zu den beiden Problemen findet man in [1].

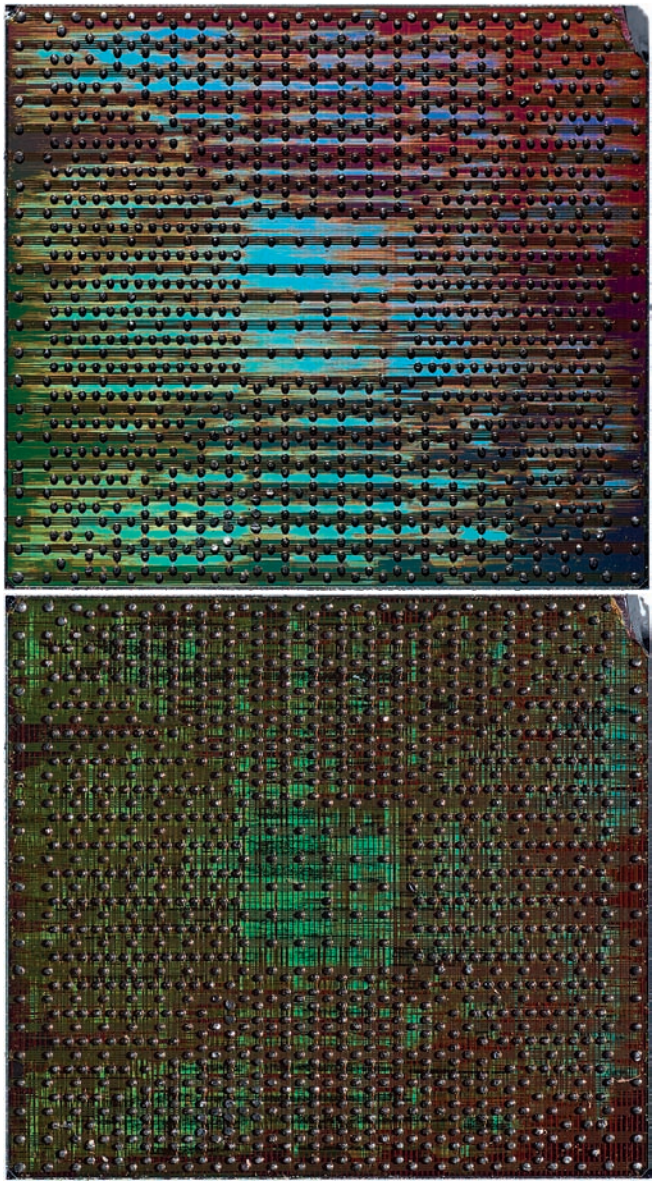
Im Fall des Generators Dual_EC_DRBG spielt $s_{i+1}Q$ die Rolle der Schlüssels S, der wie zufällig aussieht, selbst wenn man P, Q und sogar den in der nächsten Iteration des Generators berechneten Wert $R = s_{i+1}P$ kennen würde. In diesem Sinne ist das *grundsätzliche* Design-Prinzip des Generators plausibel, auch wenn jede Iteration durch zwei Multiplikationen in der elliptischen Kurve wesentlich langsamer als andere bekannte Generatoren ist. Dass der Generator dennoch Schwächen hat, liegt an den Details, wie wir im folgenden Abschnitt diskutieren.

Schwächen im Generator Dual_EC_DRBG

Bereits kurz nach Erscheinen des Standards haben die ersten Analysen [15, 2] gezeigt, dass die Ausgaben des Dual_EC_DRBG nicht die übliche kryptographische Sicherheit bieten. Diese Analysen nutzen aus, dass die Punkte der elliptischen Kurve bei Projektion auf die untersten 240 Bits der x-Koordinate eben nicht uniform sind: Ein nicht zu vernachlässigender Anteil der Menge der Zeichenketten aus 240 Bits kann nicht von solchen Punkten getroffen werden. Dieser Schritt, die einfache Projektion auf die untersten Bits, schleust also Schwächen ein, obwohl die Kurvenpunkte uniform verteilt sind. Es ist bemerkenswert, dass hier schon vor Veröffentlichung des Standards bessere Verfahren zur Glättung der Verteilung bekannt waren (beispielsweise in [7]).

Die Autoren von [15, 2] zeigen, dass sie wegen der einfachen Projektion auf die Bits die Ausgabe des Dual_EC_DRBG mit einem Vorteil von ca. 0,1 % gegenüber echt zufälligen Werten erkennen können. Indem sie mehrere Ausgaben untersuchen, lässt sich dieser Vorteil sogar auf einige Prozentpunkte steigern. Anders ausgedrückt: Die Pseudo-Entropie des Dual_EC_DRBG ist nicht maximal. Zwar gelten Generatoren mit solchen verzerrten Ausgaben in der Kryptographie nicht als sicher, dennoch war nicht bekannt, wie man diese Schwäche des Dual_EC_DRBG unmittelbar hätte ausnutzen können, um beispielsweise große Tei-

le eines geheimen Schlüssels verlässlich zu berechnen. Wie dies gehen könnte, zeigte die Arbeit von Shumov und Ferguson [17]. Die beiden Autoren bestätigen, dass man bei geschickter Wahl der Punkte P und Q – was für die Entwickler des Standards durchaus möglich gewesen wäre – die Ausgaben des Generators mit geringem Aufwand vorhersagen kann.



Die deterministische Chip-Sicht: Ein G5 in unterschiedlich einfallendem Licht (Sammlung Günter M. Ziegler. Foto: Christoph Eyrich)

Der Angriff von Shumov und Ferguson basiert auf folgender Beobachtung. Da P ein Erzeuger, $Q \neq \mathcal{O}$ und die Ordnung der elliptischen Kurve prim ist, gibt es ein ganzzahliges e mit $P = eQ$. Shumov und Ferguson gehen nun davon aus, dass der Angreifer den Wert e kennt. Der Angreifer betrachtet dann eine Ausgabe $r \in \{0,1\}^{240}$ einer Iteration des Generators, die er beispielsweise als Teil eines Initialisierungsvektors einer Verschlüsselung oder als Frage in einer Challenge-Response-Authentisierung lernen kann. Er berechnet alle möglichen $2^{16} = 65.536$ Bitfolgen der Länge 256, die auf r enden. Für jeden dieser Werte testet der Angreifer, ob diese Bitfolge der x-Koordinate eines Punktes auf der elliptischen Kurve entspricht, indem er prüft, ob $z = x^3 + ax + b \bmod p$ ein quadratischer Rest in $\mathbb{F}_p$ ist. Jedes solche x gibt maximal zwei mögliche Werte für y. Der Gesamtaufwand dafür ist so gering, dass er mit jedem herkömmlichen Rechner zu bewerkstelligen ist.

Der Angreifer erhält so eine Menge von maximal $2 \cdot 2^{16}$ Kurvenpunkten $R_1, R_2, R_3, \ldots$, wobei der „richtige“ Punkt $s_{i+1}Q$ darin enthalten sein muss. Mit Hilfe der Kenntnis von e kann der Angreifer dann die Punkte $eR_1, eR_2, eR_3, \ldots$ berechnen, sodass darunter auch der Punkt $s_{i+1}P = s_{i+1}(eQ) = e(s_{i+1}Q)$ sein muss. Folglich kennt der Angreifer eine beschränkte Menge von Punkten, deren x-Koordinate auch den übernächsten Zustandswert $s_{i+2} = \text{x-coord}(s_{i+1}P)$ des Generators enthält. Würde er den Punkt eindeutig kennen, könnte der Angreifer alle folgenden Ausgaben des Generators dann exakt vorhersagen! Dazu muss er aber nun lediglich einige weitere Ausgaben des Generators beobachten, um die Menge der potenziellen Kandidaten auf ein Element zu reduzieren. Folglich ist der Generator bei Kenntnis von e mit $P = eQ$ vollkommen unsicher.

Der Angriff von Shumov und Ferguson ist nicht überraschend, wenn man bedenkt, dass das DH-Berechnungsproblem und auch das DH-Entscheidungsproblem bei bekanntem e mit $P = Qe$ beide leicht zu lösen sind: Gegeben $P, Q = qP, R$ und eben e, das multiplikative Inverse zu q modulo der bekannten Gruppenordnung, kann man leicht den Schlüssel $S = rQ = (rq)P$ für $R = rP$ berechnen, indem man einfach q aus e berechnet und $S = qR$ bildet. In diesem Sinne „spielt“ der Angreifer dann quasi die Rolle von Alice im DH-Schlüsselaustausch. Nur wenn Q nachweislich so bestimmt worden wäre, dass niemand effizient e berechen kann, wäre die Sicherheit des Dual_EC_DRBG gewährleistet. Dass dies nicht der Fall war, hat die Veröffentlichung der New York Times gezeigt.

Wie geht es weiter?

Mehr als fünf Jahre nach Bekanntwerden der Schwächen, aber erst nach Bestätigung durch die New York Times, dass die NSA die Spezifikation beeinflusst hat, raten inzwischen sowohl NIST als auch Firmen wie RSA von der Verwendung des Generators Dual_EC_DRBG ab. Man kann daher hoffen, dass in Zukunft keine Angriffe über diese Schwachstelle erfolgen werden.

In einem im Dezember 2013 erschienenen Artikel der Nachrichtenagentur Reuters [9] wurde behauptet, dass die Firma RSA für die Verwendung des schwachen Generators Dual_EC_DRBG in ihrem BSAFE-Produkt von der NSA bezahlt wurde. Die Firma bestritt diese Vorwürfe umgehend [13]. Eine Klärung steht noch aus.

NIST hat inzwischen den Standard SP 800-90A auf den Status eines „Draft“ zurückgesetzt und im Zeitraum von September 2013 bis November 2013 um Kommentierung gebeten. Das weitere Vorgehen der NIST ist zum Zeitpunkt, zu dem dieser Artikel verfasst wurde, nicht bekannt.

Fazit

Die Rolle der NSA bei der Erstellung des Standards SP 800-90, als auch die weiteren bekannt gewordenen Maßnahmen der Geheimdienste, sind natürlich äußerst bedenklich. Im Fall des Dual_EC_DRBG wurden die eingebauten Schwächen unmittelbar entdeckt. Dennoch blieb der Generator Teil des Standards und wurde auch in Produkten verwendet. Diese Tatsache zeigt – unabhängig davon, ob Standards bewusst oder unbewusst eingebaute Schwächen beinhalten – das Problem heutiger Sicherheitsstandards. Sie beruhen in der Regel zwar auf bekannten Sicherheitsprinzipien, fallen aber in ihren Sicherheitsbetrachtungen gegenüber Analysen, wie sie in akademischen Veröffentlichungen heute üblich sind, stark ab. Solche wissenschaftlichen Betrachtungen schließen in der Regel sogenannte Reduktionsbeweise oder ausführliche kryptanalytische Techniken ein, während die Sicherheitskriterien von Standards im Allgemeinen vage bleiben. Trotzdem genießen standardisierte Verfahren einen verblüffenden Vertrauensvorschuss bezüglich ihrer Sicherheit, und werden bedenkenlos in Produkten verwendet. Hier sind allerdings nicht nur die Entwickler von Standards und die Firmen gefordert, sondern auch Kryptographen, um gemeinsam die Sicherheitsgarantien von kryptographischen Standards zu verbessern.

Literatur

[1] Dan Boneh. The decision Diffie-Hellman problem. volume 1423 of *Lecture Notes in Computer Science*, pages 48–63. Springer, 1998.

[2] Daniel R. L. Brown and Kristian Gjøsteen. A security analysis of the NIST SP 800-90 elliptic curve random number generator. In *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 466–481. Springer, 2007.

[3] Computer Emergency Response Team (CERT). Dual_EC_DRBG output using untrusted curve constants may be predictable. Vulnerability Note VU#274923, November 2013.

[4] Computer Emergency Response Team (CERT). Dual_EC_DRBG output using untrusted curve constants may be predictable. RSA Security, Inc. Information to Vulnerability Note VU#274923, September 2013.

[5] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[6] Oded Goldreich. *Pseudorandom Generators: A Primer*. ULECT series. AMS, 2010.

[7] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[8] International Organization for Standardization. Iso/iec 18031:2011: Information technology – security techniques – random bit generation, 2011.

[9] Joseph Menn. Exclusive: Secret contract tied NSA and security industry pioneer. Reuters, Dezember 2013.

[10] National Institute of Standards and Technology. Special publications (800 series). `http://csrc.nist.gov/publications/PubsSPs.html`, 2013.

[11] National Institute of Standards and Technology. Supplemental ITL bulletin for September 2013, September 2013.

[12] Nicole Perlroth. Government announces steps to restore confidence on encryption standards. New York Times, September 2013.

[13] RSA, The Security Division of EMC. RSA response to media claims regarding NSA relationship. `https://blogs.rsa.com/news-media-2/rsa-response/`, Dezember 2013.

[14] Bruce Schneier. The strange story of Dual_EC_DRBG. `www.schneier.com/blog`, November 2007.

[15] Berry Schoenmakers and Andrey Sidorenko. Cryptanalysis of the dual elliptic curve pseudorandom generator. *IACR Cryptology ePrint Archive*, 190, 2006.

[16] Claude Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948.

[17] Dan Shumow and Niels Ferguson. On the possibility of a back door in the NIST SP 800-90 Dual EC PRNG. Crypto Rump Session, 2007.

GTPack - ein *Mathematica* Paket für Gruppentheorie in der Festkörperphysik

M. Geilhufe, W. Hergert
(Max-Planck-Institut für Mikrostrukturphysik, Martin-Luther-Universität Halle-Wittenberg)

geilhufe@mpi-halle.mpg.de
wolfram.hergert@physik.uni-halle.de

Einleitung

Die Anwendung gruppentheoretischer Algorithmen in der Festkörperphysik führt oft zu Rechnungen, die sich leicht automatisieren lassen. Dies ist auf rein numerischen Wege möglich und so teilweise in *ab initio* Programmen zur Berechnung der elektronischen Struktur von Festkörpern integriert [1]. Computeralgebrasysteme haben den Vorteil, dass sich gruppentheoretische Überlegungen mit einer analytischen Behandlung physikalischer Modellsysteme verbinden lassen. Die Entwicklung von **GTPack** zielt darauf ab, ein Paket zu programmieren, welches insbesondere die Behandlung von Problemen der Elektronenstruktur von Festkörpern oder der Photonik erlaubt. Die Implementierung sollte dabei soweit als möglich der in der Physik verwendeten Literatur und Notation [2, 3, 4] folgen. Der Schwerpunkt der Entwicklung liegt daher etwas anders als bei großen Paketen wie CrystGAP oder CARAT [5, 6]. Da im Rahmen des Physikstudiums in Halle begleitende *Mathematica*-Kurse zur theoretischen Physik angeboten werden, haben wir uns entschieden, *Mathematica* als Grundlage von **GTPack** zu wählen.

Grundlagen

Im Rahmen der Quantentheorie des Festkörpers ist es das Ziel, geeignete Approximationen für die Lösung der Vielteilchen-Schrödinger-Gleichung zu finden. Im Festkörper wechselwirken ca. 10^{13} Elektronen über die Coulombwechselwirkung. Die Schrödingergleichung als Grundgleichung der Quantentheorie kann für ein solches Ensemble weder analytisch noch numerisch gelöst werden. Näherungen gestatten es, das Problem auf die Bewegung eines Elektrons in einem effektiven Potential aller anderen Teilchen zurückzuführen. Das effektive Potential wird dabei wesentlich durch die Anordnung der Atome im Festkörper bestimmt. Damit übertragen sich die Symmetrieeigenschaften des Kristalls auf den Lösungsraum der effektiven Schrödingergleichung. Physikalische Messgrößen wie Leitfähigkeit oder Magnetisierung lassen sich aus der Lösung berechnen.

Aufgrund der hohen Symmetrie von Kristallen, stellt die Verwendung gruppentheoretischer Methoden zum einen eine Vereinfachung des numerischen Aufwandes dar (z. B. durch Entwicklung der Lösung in einen Satz symmetrieangepasster Basisfunktionen) zum anderen erlaubt sie die Vorhersage von bestimmten Eigenschaften der Lösung (z. B. Entartung von Energieeigenwerten, d.h. verschiedene quantenmechanische Zustände gehören zur gleichen Energie) ohne numerische Auswertung.

GTPack besitzt einen modularen Aufbau und ist in verschiedene Pakete untergliedert. Die Basispakete *Basic.m* und *RepresentationTheory.m* umfassen grundlegende Algorithmen der Gruppen- und Darstellungstheorie. Dabei werden alle Ausgaben algorithmisch berechnet, **GTPack** greift nicht auf Tabellen zurück. Beispielsweise werden Charaktertafeln nach der Methode von Burnside berechnet [7], Matrizen irreduzibler Darstellungen können nach dem Algorithmus von Flodmark und Blokker [8] konstruiert werden oder Clebsch-Gordan-Koeffizienten nach der Methode von Van den Broek und Cornwell [9].

Das Paket *Install.m* enthält die nötigen Module um Punktgruppen zu generieren. Hierbei ist es wichtig zu erwähnen, dass alle Gruppenelemente sowohl symbolisch (z. B. C_{3z} als dreizählige Drehung um die z-Achse), als Matrix, als Quaternion oder als Satz von Euler-Winkeln dargestellt werden können. Je nach Anwendung können Drehmatrizen im $\mathbb{R}^3$ oder $\mathbb{R}^2$, Spin-Drehmatrizen aus der Gruppe $U(2)$ (für die Verwendung von Doppelgruppen) oder Permutationsmatrizen gewählt werden. Generatoren der 32 kristallographischen Punktgruppen sind in **GTPack** gespeichert und erleichtern das Installieren einer Punktgruppe.

In *Symbols.m* werden die Definitionen von Symbolen, zum Beispiel die Bezeichungen der Punktgruppen und Punktgruppenelemente, wie sie in der Physik üblich sind, zusammengefasst. Eine Reihe von Hilfsfunktionen befinden sich in *Auxiliary.m*. Hierbei haben wir unter anderem grundlegende Befehle der Quaternionen-Algebra eingeführt, sowie kartesische und reelle Darstellungen der Kugelflächenfunktionen implementiert. Die Pakete *Lattice.m* und *CrystalStructure.m* stellen Algorithmen zur Bereitstellung von Strukturinformationen im direkten und reziproken Raum zur Verfügung, wie sie unter anderem für die Be-

rechnung von elektronischen Bandstrukturen benötigt werden. Die Pakete *TightBinding.m*, *ElectronicStructure.m*, *Pseudopotential.m* sind Anwendungspakete die sich mit der Diskussion der elektronischen Struktur von Festkörpern beschäftigen, während *Photonics.m* die Untersuchung der Eigenschaften photonischer Kristalle gestattet. Die Datensätze `TB_Handbook.parm` und `PseudoPotential.parm` enthalten Parametersätze, die das Aufstellen von Modellen zur Berechnung der elektronischen Struktur erleichtern. `GTPack.struc` stellt Raumgruppeninformationen bereit. Die Datensätze können nach den Erfordernissen des Nutzers erweitert werden.

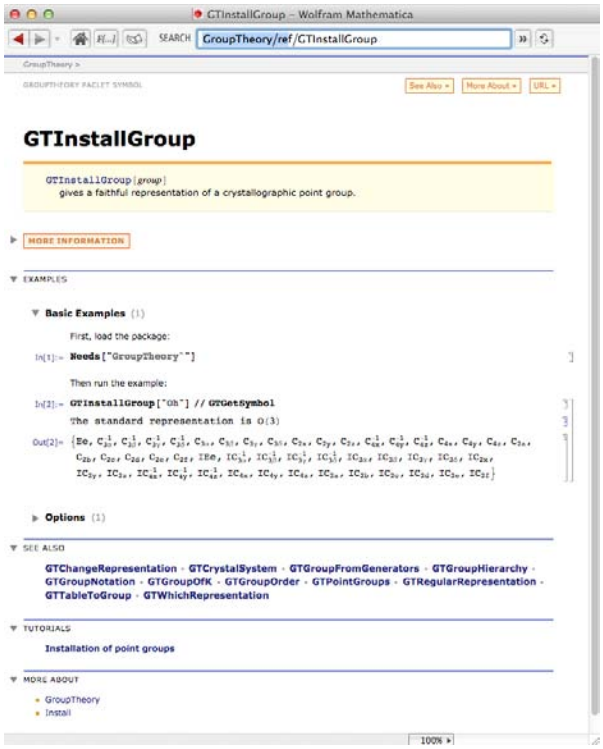


Abbildung 1: *Referenzseite zum Befehl* ***GTInstallGroup GTPack***

GTPack verfügt über ein Hilfesystem, welches in das *Documentation Center* von *Mathematica* integriert ist. Wesentliche Befehle des Paketes, die Punkt- und Doppelgruppen sowie alle Elemente der Punkt- und Doppelgruppen sind von einer Palette abrufbar. Die Referenzseiten, Leitseiten und Tutorials sind im *Mathematica*-Stil abgefasst (siehe Abb. 1)

Beispiele

Die Arbeit mit dem Paket soll an zwei einfachen Beispielen erläutert werden. Dabei wird noch einmal deutlich, dass alle Befehlsnamen direkten Bezug zur Aufgabe haben. Zur Unterscheidung von den Standardbefehlen von *Mathematica* tragen Befehle des Pakets den Präfix **GT** und Optionen den Präfix **GO**.

Kristallfeldaufspaltung

Abb. 2 zeigt ein Atom in unterschiedlicher kristallographischer Umgebung. Die oktaedrische Koordination entspricht einer Situation, wie sie z. B. im Bariumtitanat ($BaTiO_3$) auftritt.

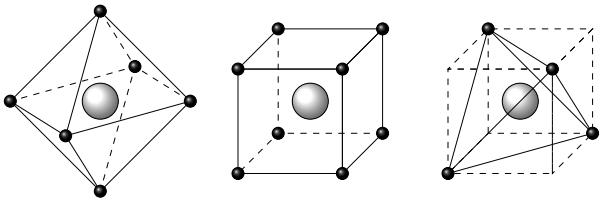


Abbildung 2: *Ein Zentralatom innerhalb eines oktaedrischen, kubischen und eines tetraedrischen Kristallfeldes.*

Aufgabe ist es nun festzustellen, wie die Entartung von Energienivaus des Zentralatoms durch das elektrostatische Feld der es umgebenden Atome aufgehoben wird. Dabei beschränken wir uns auf ein einziges Elektron und vernachlässigen Schwingungen der Atome (zeitunabhängiges Potential). Es sei $\hat{\mathsf{H}}_0$ der zum Zentralatom gehörige Hamilton-Operator. Die Schrödinger-Gleichung des atomaren Problems lautet in diesem Fall

$$\hat{\mathsf{H}}_0 \psi_{lm}(\vec{r}) = E_0^l \psi_{lm}(\vec{r}). \tag{1}$$

ψ_{lm} ist die Wellenfunktion des Elektrons im $2l+1$-fach entartetem Energieniveau E_l ($-l \leq m \leq l$). Angenommen die Lösung von Gleichung (1) sei bekannt, so können wir im Rahmen der linearen Störungstheorie die Lösung für den Hamilton-Operator $\hat{\mathsf{H}} = \hat{\mathsf{H}}_0 + V_{\mathrm{Kr}}$ ermitteln. Dabei ist V_{Kr} das Kristallfeldpotential, welches durch die das Zentralatom umgebenden Atome zustande kommt.

Das Kristallfeldpotential lässt sich in reelle Kugelflächenfunktionen S_l^m entwickeln,

$$V_{\mathrm{Kr}}(r,\theta,\phi) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} A_l^m r^l S_l^m(\theta,\phi). \tag{2}$$

Aufgrund der Symmetrie des Systems kann Gleichung (2) stark vereinfacht werden. Angenommen es gibt g Transformationen T einer Gruppe $\mathcal{G}$, welche den Hamiltonoperator $\hat{\mathsf{H}} = \hat{\mathsf{H}}_0 + V_{\mathrm{kr}}$ invariant lassen. Das heißt, nach Anwendung des Projektionsoperators der Einsdarstellung Γ_1 auf das Kristallfeldpotential muss gelten

$$\hat{\mathcal{P}}^{\Gamma_1} V_{\mathrm{Kr}} = \frac{1}{g} \sum_{T \in \mathcal{G}} \hat{\mathsf{P}}(T) V_{\mathrm{Kr}} = V_{\mathrm{Kr}}. \tag{3}$$

Da die Kugelflächenfunktion S_l^m zu bestimmten l eine Basis in einem $2l+1$-dimensionalen Vektorraum bilden, muss gelten

$$\hat{\mathsf{P}}(T) S_l^m = \sum_{m'} S_l^{m'} D_{m'm}^l(T). \tag{4}$$

Wir können somit ein lineares Gleichungssystem für die Koeffizienten A_l^m formulieren,

$$A_l^m = \frac{1}{g} \sum_{T \in \mathcal{G}} \sum_{m'} A_l^{m'} D_{m'm}^l(T). \tag{5}$$

Mit Hilfe des Befehls **GTCrystalField** in **GTPack** ist es möglich, dieses Gleichungssystem für beliebige

Symmetrie zu lösen und das Kristallfeldpotential zu formulieren. Alternativ zur Entwicklung nach reellen Kugelflächenfunktionen, können komplexe Kugelflächenfunktionen verwendet werden, oder das Ergebnis in sogenannten Operatoräquivalenten [10] ausgedrückt werden.

```
vcr =
 GTCrystalField[Ohgroup, 4, GOHarmonics → "Real"] /.
  Y[l_, m_] -> GTTesseralHarmonicY[l, m, θ, ϕ]
```

$$\frac{A[0,0]}{2\sqrt{\pi}} + \frac{1}{7} r^4 A[4,0] \left(\frac{21\left(3 - 30\,\mathrm{Cos}[\theta]^2 + 35\,\mathrm{Cos}[\theta]^4\right)}{16\sqrt{\pi}} + \frac{105\left(-1 + \mathrm{Cos}[\theta]^2\right)^2 \mathrm{Cos}[4\phi]}{16\sqrt{\pi}} \right)$$

```
A00 = Table[GTCrystalFieldParameter[0, 0,
    Rlist[[i]], qlist[[i]]], {i, 1, 3}];

A40 = Table[GTCrystalFieldParameter[4, 0,
    Rlist[[i]], qlist[[i]]], {i, 1, 3}];
```

Die folgende Tabelle beinhaltet die Werte für die Kristallfeldparameter A_0^0 und A_4^0.

```
TableForm[{A00, A40},
 TableHeadings →
  {{"A₀⁰", "A₄⁰"}, {"Oktaeder", "Würfel",
    "Tetraeder"}}]
```

	Oktaeder	Würfel	Tetraeder
A_0^0	$\frac{12\sqrt{\pi}\,q}{R}$	$\frac{16\sqrt{\pi}\,q}{R}$	$\frac{8\sqrt{\pi}\,q}{R}$
A_4^0	$\frac{7\sqrt{\pi}\,q}{3R^5}$	$-\frac{56\sqrt{\pi}\,q}{27R^5}$	$-\frac{28\sqrt{\pi}\,q}{27R^5}$

```
Amat =
  Table[
   Integrate[Sin[θ] GTTesseralHarmonicY[2, m1, θ, ϕ]
     vcr GTTesseralHarmonicY[2, m2, θ, ϕ],
    {θ, 0, π}, {ϕ, 0, 2π}], {m1, -2, 2}, {m2, -2, 2}];

rule = {q r^4/R^5 → 6 Dq[oct], q/R → ϵ[oct] / 6};
```

Eigenwerte des Oktaeders:

```
Expand[Eigenvalues[Amat]] /.
 A[l_, m_] -> GTCrystalFieldParameter[l, m,
   Rlist[[1]], qlist[[1]]]
 /. rule

{-4 Dq[oct] + ϵ[oct],
 -4 Dq[oct] + ϵ[oct], -4 Dq[oct] + ϵ[oct],
 6 Dq[oct] + ϵ[oct], 6 Dq[oct] + ϵ[oct]}
```

Abbildung 3: *Berechnung der Kristallfeldaufspaltung mit* ***GTPack****.*

Die Punktgruppe für Oktaeder und Würfel in Abb. 2 ist O_h, die Punktgruppe des Tetraeders T_d. Das Kristallfeld hat für die Punktgruppen O_h und T_d die Form

$$V_{cr} = A_0^0 S_0^0(\theta,\phi) + A_4^0 r^4 \left(S_4^0(\theta,\phi) + \sqrt{\frac{5}{7}} S_4^4(\theta,\phi) \right)$$

In linearer Näherung kann die Änderung des $2l+1$-fach entarteten Energieeigenwerte E_0^l über die Eigenwerte der Matrix $a^l_{m'm}$ berechnet werden,

$$a^l_{m'm} = \int \mathrm{d}\vec{\varphi}\ \ _{lm'}(\vec{r}) V_{cr} \phi_{lm}(\vec{r}). \qquad (6)$$

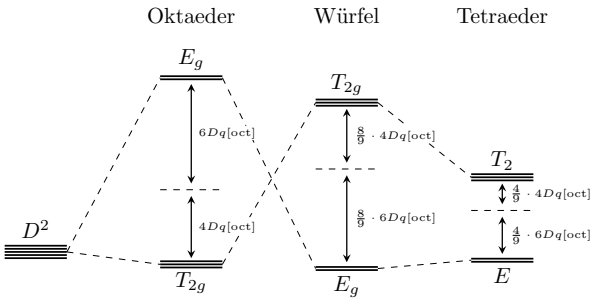


Abbildung 4: *Aufspaltung eines d-Niveaus innerhalb eines Kristallfeldes mit oktaedrischer, kubischer und tetraedrischer Symmetrie.*

Das Ergebnis für d-Elektronen ($l = 2$) ist in Abb. 4 dargestellt. Dabei haben wir für die Berechnung der Kristallfeldparameter A_l^m ein einfaches Punktladungsmodell verwendet, wie es in dem Befehl **GTCrystalFieldParameters** implementiert ist. Die Ladungen der Atome sind in der Liste `qlist` und deren Positionen in der Liste `Rlist` gegeben. Der Übersichtlichkeit halber haben wir den Parameter $Dq[oct]$ eingeführt [4].

```
<< GroupTheory`

Ohgroup = GTInstallGroup["Oh"];

The standard representation has changed to O(3)

charactertable = GTCharacterTable[Ohgroup,
    GOIrepNotation → "Mulliken"];
```

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}
A_{1g}	1	1	1	1	1	1	1	1	1	1
T_{1g}	3	-1	-1	-1	1	-1	1	0	0	3
T_{2g}	3	-1	-1	1	-1	1	-1	0	0	3
T_{1u}	3	-1	1	-1	-1	1	1	0	0	-3
T_{2u}	3	-1	1	1	1	-1	-1	0	0	-3
A_{1u}	1	1	-1	-1	1	1	-1	-1	1	-1
E_u	2	2	-2	0	0	0	0	1	-1	-2
A_{2u}	1	1	-1	1	-1	-1	1	-1	1	-1
A_{2g}	1	1	1	-1	-1	-1	-1	1	1	1
E_g	2	2	2	0	0	0	0	-1	-1	2

```
χp = GTAngularMomentumChars[charactertable[[1]],
    1];
χd = GTAngularMomentumChars[charactertable[[1]],
    2];
χf = GTAngularMomentumChars[charactertable[[1]],
    3];

GTIrep[χp, charactertable];

T1u

GTIrep[χd, charactertable];

T2g⊕Eg

GTIrep[χf, charactertable];

T1u⊕T2u⊕A1u
```

Abbildung 5: *Qualitative Analyse der Aufspaltung von Energieniveaus in kubischen Kristallfeldern.*

Alternativ lässt sich eine qualitative Überlegung des Problems anstellen. Nach Installation der Gruppe O_h mit **GTInstallGroup** wird die Charaktertafel mit **GTCharacterTable** erzeugt und dabei die Bezeichnung der irreduziblen Darstellungen nach Mulliken per Option ausgewählt. Die Wellenfunktionen des

Zentralatoms sind ohne das Kristallfeld Basisfunktionen zu irreduziblen Darstellungen der Gruppe O(3). Diese irreduziblen Darstellungen sind allerdings reduzibel in kubischer Symmetrie (O_h). Mit **GTAngularMomentumChars** werden die Charaktere der $(2l+1)$-dimensionalen Darstellungen für die s, p, d Wellenfunktionen mit den Drehimpulsquantenzahlen $l = 1, 2, 3$ bestimmt. Die Darstellungsmatrizen entsprechen gerade den Matrizen $D^l_{m'm}$ aus der vorhergehenden Diskussion. Die Ausreduktion erfolgt mit **GTIrep**. Die fünfdimensionale Darstellung (auch fünffache Entartung des Energieniveaus) der d-Elektronen spaltet somit in eine zweidimensionale und eine dreidimensionale Darstellung ($T_{2g} \oplus E_g$) auf. Die siebendimensionale Darstellung der f-Elektronen zerfällt in zwei dreidimensionale Darstellungen und eine eindimensionale Darstellung. Der Output von **GTIrep** entspricht dabei der in der Physik üblichen Mullikennotation.

Elektronenstruktur von Graphen

Im Jahre 2010 erhielten Gleim und Novoselov den Nobelpreis für ihre Untersuchungen an Graphen, einer zweidimensionalen Kohlenstoffmodifikation mit bienenwabenartiger Struktur. Spezielle elektronische Eigenschaften dieses Materials können in einem Tight-Binding Modell [11] gut erklärt werden. In diesem Modell sind die Elektronen stark an den Atomen lokalisiert. Durch diese Annahme kann die Schrödinger-Gleichung in Matrix-Form formuliert werden. Matrizen für solche Tight-Binding Modelle lassen sich mit **GTPack** automatisch erzeugen.

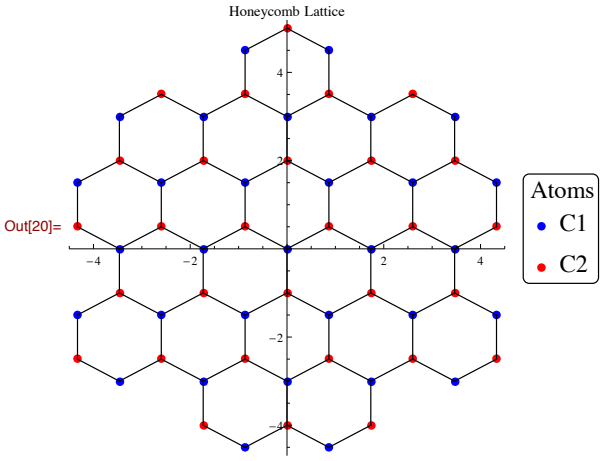


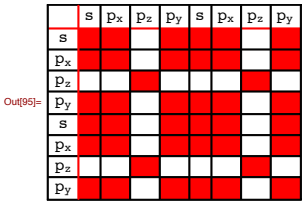
Abbildung 6: *Struktur von Graphen. Die 2 Kohlenstoffatome in der Basis sind mit C1 und C2 bezeichnet*

Abb. 6 zeigt die Struktur von Graphen. Mit den Befehlen **GTCluster, GTShells** (s. Abb. 7) wird die Struktur aufgebaut und es werden die Nachbarschaftsverhältnisse analysiert. Die Liste `hcp` enthält dabei die Strukturinformationen, die aus der Datenbank `GTPack.struc` geladen werden. Ausgehend von der Strukturinformation und Informationen über die pro Atom zu berücksichtigenden Orbitale (ham[[1,1]] in Abb. 7) kann man die Schrödinger-Gleichung in Matrixform gewinnen.

```
cl = GTCluster[hcp, 6, GOLattice → {a -> 1}];
bas = hcp[[7]] /. {a → 1};
shells = {2, 2};

85 atoms

lat = GTShells[cl, bas, shells, GOVerbose → True,
   GOTbLattice → {{"C1,C1", {2}}, {"C2,C2", {2}},
     {"C1,C2", {1}}},
   GOPosition → "Relative"];
```

Basis	dist	Atoms	dist	Atoms
C1	1	{{C2, 3}}	$\sqrt{3}$	{{C1, 6}}
C2	1	{{C1, 3}}	$\sqrt{3}$	{{C2, 6}}

```
In[94]:= bas = {{"C1", {0, 1}}, {"C2", {0, 1}}};
         hamc = GTTbHamiltonian[bas, lat];

In[95]:= GTHamiltonianPlot[hamc, bas]
```




```
In[110]:= ham = {{hamc[[3, 3]], hamc[[3, 7]]},
            {hamc[[7, 3]], hamc[[7, 7]]}};
          ham[[1, 1]]

Out[111]= 2 (2 Cos[3 π η] Cos[√3 π ξ] + Cos[2 √3 π ξ]) (ppπ)_1^{C1,C1} + (pp0)^{C1}
```

Abbildung 7: *Aufstellung des tight-binding Hamiltonians für Graphen.*

GTHamiltonianPlot zeigt die Struktur der entstehenden Matrix aus der die Energieeigenwerte berechnet werden. Man erkennt, dass die p_z-Orbitale entkoppeln und diese Energiewerte aus einem 2×2-Eigenwertproblem berechnet werden können. Dieses Eigenwertproblem liefert 2 Energiebänder $E(\mathbf{k})$, die in Abb. 8 dargestellt sind. In analoger Weise können die anderen 6 Bänder generiert werden.

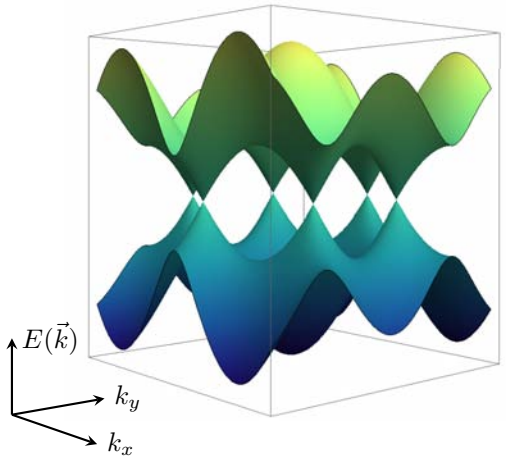


Abbildung 8: *Bandstruktur $E(\mathbf{k})$ von Graphen in Abhängigkeit vom Vektor $\mathbf{k} = (k_x, k_y)$ in der* BRILLOUIN *Zone. Die Bänder zeigen lineares Verhalten an den 6 Dirac-Punkten.*

Zusammenfassung

GTPack stellt ein Paket dar, welches gruppentheoretische Algorithmen mit der Analyse von Modellsystemen der Festkörperphysik verbindet. Damit kann die Anwendung von Gruppentheorie in der Physik in der Lehre

besser demonstriert werden. Weiterhin können Ergebnisse, die mit komplexen *ab initio* Codes, basierend auf der Dichtefunktionaltheorie, erzielt werden, analysiert und interpretiert werden.

Literatur

[1] J. Hafner, Materials simulations using VASP—a quantum perspective to materials science. *Computer Physics Communications*, 177:6–13, 2007.

[2] J.F. Cornwell, Group Theory in Physics. *Academic Press*, 1984.

[3] M. S. Dresselhaus, G. Dresselhaus und A. Jorio, Group Theory - Application to the Physics of Condensed Matter. *Springer-Verlag*, 2008.

[4] W. Haberditzl, Quantenchemie. 4. Komplexverbindungen. *Hüthig*, 1979.

[5] The GAP Group, GAP – Groups, Algorithms, and Programming, Version 4.7.4. 2014.

[6] J. Opgenorth, W. Plesken und T. Schulz, Crystallographic Algorithms and Tables. *Acta Crystallographica Section A*, 54:517–531, 1998.

[7] M. El-Batanouny und F. Wooten, Symmetry and Condensed Matter Physics: A Computational Approach. *Cambridge University Press*, 2008.

[8] S. Flodmark und E. Blokker, A computer program for calculation of irreducible representations of finite groups. *International Journal of Quantum Chemistry*, 1:703–711, 1967.

[9] P. M. van Den Broek und J. F. Cornwell, Clebsch-Gordan coefficients of symmetry groups. *physica status solidi (b)*, 90:211–224, 1978.

[10] H. A. Buckmaster, R. Chatterjee, und Y. H. Shing, The application of tensor operators in the analysis of EPR and ENDOR spectra. *physica status solidi (a)*, 13:9–50, 1972.

[11] A. V. Podolskiy und P. Vogl, Compact expression for the angular dependence of tight-binding Hamiltonian matrix elements. *Phys. Rev. B*, 69:23301, 2004.

Superstringtheorie und elliptische Kurven

J. Keitel
(Max-Planck-Institut für Physik)

jkeitel@mpp.mpg.de

Zusammenfassung

Seit der Entdeckung der Stringtheorie werden durch diese regelmäßig Querverbindungen zwischen Themengebieten der reinen Mathematik und physikalischen Fragestellungen entdeckt. In diesem Beitrag diskutieren wir Aspekte elliptischer Kurven, die relevant für das Studium von Stringkompaktifizierungen sind.

Physikalische Motivation

Trotz des überwältigenden Erfolgs des Standardmodells in der Beschreibung der elektroschwachen und der starken Wechselwirkung existiert bis heute keine experimentell überprüfte Theorie, die das Standardmodell mit Einsteins allgemeiner Relativitätstheorie vereint. Dies ist unter anderem der großen Disparanz der relevanten physikalischen Skalen geschuldet: Quanteneffekte in Gravitationstheorien werden relevant in Prozessen, die bei Energien nahe der Planckenergie von $E_{Planck} \approx 10^{19}$ GeV stattfinden, während moderne Teilchenbeschleuniger Schwerpunktenergien von $E_{LHC} \approx 10^4$ GeV erreichen. Dennoch gibt es einen Kandidaten, der Eichtheorien, die Bausteine des Standardmodells, mit der allgemeinen Relevativitätstheorie vereint: Die Superstringtheorie.

Im Rahmen der Superstringtheorie wird das Konzept des nulldimensionalen Punktteilchens ersetzt durch ausgedehnte eindimensionale Objekte, sogenannte *Strings*. Bei Energien, die deutlich kleiner sind als die Stringskala, reproduzieren solche Strings dann das Teilchenspektrum von Eichtheorien und Gravitation. Aus internen Konsistenzgründen müssen diese Strings jedoch in einer Welt mit zehn Raumzeitdimensionen existieren. Um dies in Verbindung mit unseren Beobachtungen zu bringen, ist es daher notwendig, sechs die-

ser Dimensionen mathematisch *kompakt* zu machen und ihnen eine Größe zu geben, die bisher experimentell noch nicht beobachtet worden sein kann. Da die Superstringtheorie in zehn Dimensionen nur einen dimensionsbehafteten Parameter aufweist, ist es die Geometrie der kompakten Extradimensionen, die physikalische Observablen in vier Dimensionen, wie zum Beispiel Naturkonstanten, festlegt. Eine entscheidende Aufgabe im Studium solcher *Stringkompaktifizierungen* ist es daher, Verbindungen zwischen geometrischen Größen und physikalischen Observablen herzustellen.

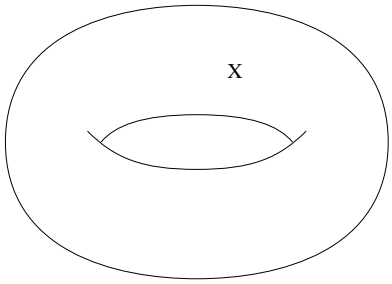


Abbildung 1: *Visualisierung einer elliptischen Kurve über $\mathbb{C}$: Ein 2-Torus mit einem markierten Punkt.*

Im Folgenden beschäftigen wir uns mit einer bestimmten Klasse von Stringkompaktifizierungen, sogenannten *F-Theorie-Kompaktifizierungen* [1]. Diese fügen den sechs kompakten Dimensionen noch zwei weitere unphysikalische Dimensionen in Form einer *elliptischen Kurve* hinzu. Intuitiv entspricht eine elliptische Kurve über den komplexen Zahlen einem 2-Torus mit einem markierten Punkt, siehe Abbildung 1. Unser Ziel ist es, die *Mordell-Weil-Gruppe* der elliptischen Kurve zu bestimmen, da diese die Anzahl der $U(1)$-Eichgruppen, also Wechselwirkungen, die der elektromagnetischen Kraft ähnlich sind, bestimmt. Bevor wir unser Vorgehen genau beschreiben, führen wir jedoch kurz ein paar wohlbekannte Eigenschaften elliptischer Kurven ein. Alle unsere Rechnungen wurden mittels Sage [2] durchgeführt und wir werden daher im Laufe dieses Artikels immer wieder Bezug darauf nehmen.

Elliptische Kurven

Per Definition ist eine elliptische Kurve über einem Körper K eine glatte projektive Kurve von Geschlecht 1 mit einem Punkt $\mathcal{O}$, dessen Koordinaten in K liegen. Solange $\text{char}(K) \notin \{ 2, 3\}$ lässt sich jede solche Kurve in *Weierstrassform*

$$\mathcal{E} : \; y^2 z = x^3 + f x z^2 + g z^3 \tag{7}$$

darstellen. Hierbei sind $(x : y : z)$ die homogenen Koordinaten des projektiven Raumes $\mathbb{CP}^2$ und f, g liegen in K. Elliptische Kurven haben die besondere Eigenschaft, dass ihre Punkte mit Koeffizienten in K eine abelsche Gruppe bilden, die Mordell-Weil-Gruppe $\text{MW}(\mathcal{E})$.

Sage [2] enthält Funktionen, um elliptische Kurven einfach zu manipulieren. Eine elliptische Kurve über K lässt sich durch

```
sage: E = EllipticCurve(K, [f, g])
```

erstellen. Gegeben $\mathcal{E}$ kann man nun Punkte auf E durch Angabe ihrer Koeffizienten definieren

```
sage: pt = E(pt_x, pt_y, pt_z)
```

und diese Punkte dann bezüglich der Gruppenwirkung addieren. Sage wählt automatisch den Punkt bei unendlich, $(0 : 1 : 0)$, als neutrales Element der Gruppenstruktur:

```
sage: zero = E(0, 1, 0)
sage: zero + zero
(0 : 1 : 0)
```

Für elliptische Kurven, deren Darstellung in Weierstrassform bekannt ist, kann man weiterhin herausfinden, welches der Rang der Untergruppe der Mordell-Weil-Gruppe ist, die eine gegebene Menge von Punkten erzeugt.

Elliptische Faserungen

Für unsere F-Theorie-Kompaktifizierungen betrachten wir Mannigfaltigkeiten M_8, die eine Faserung

$$\mathcal{E} \to M_8 \xrightarrow{\pi} M_6 \tag{8}$$

aufweisen, wobei $\mathcal{E}$ eine elliptische Kurve definiert und M_6 die *Basismannigfaltigkeit* ist, auf der die sechs Extradimensionen des Superstrings kompaktifiziert werden. Bildlich gesehen wird also an jeden Punkt in M_6 eine elliptische Kurve $\mathcal{E}$ gefügt, deren komplexe Struktur über M_6 variieren kann. Weiterhin fordern wir, dass M_8 eine Calabi-Yau-Mannigfaltigkeit ist und daher $c_1(TM_8) = 0$ gilt. Damit $\mathcal{E}$ eine elliptische Kurve ist, muss M_8 einen globalen Schnitt besitzen, der das neutrale Element der Faser bestimmt. Besitzt M_8 weitere Schnitte, so definieren diese zusätzliche rationale Punkte in der Faser, die die Mordell-Weil-Gruppe $\text{MW}(\mathcal{E})$ erzeugen.

Im Folgenden interessieren wir uns für Calabi-Yau-Mannigfaltigkeiten $M_8 \subset Y_{10}$, die als Hyperflächen in torischen Varietäten Y_{10} konstruiert werden können. Die torischen Varietäten Y_{10} weisen dann selbst eine Faserung

$$Y_4 \to Y_{10} \to M_6 \tag{9}$$

auf. Insbesondere ist die elliptische Kurve $\mathcal{E} \subset Y_4$ eine Hyperfläche in der Mannigfaltigkeit Y_4, die selbst eine torische Varietät ist.

Elliptische Kurven in torischen Varietäten

Als Nächstes betrachten wir daher elliptische Kurven, die als Hyperflächen in torischen Varietäten dargestellt werden. Eine Einführung in torische Geometrie findet sich beispielsweise in [3, 4]. Um gewisse Regularitätseigenschaften der elliptischen Faserungen zu garantieren, betrachten wir nur torische Varietäten, die Gorenstein und Fano sind [5]. Bis auf Isomorphismen stehen diese in eindeutiger Korrespondenz zu reflexiven Polytopen. Als Polytop bezeichnen wir hier die konvexe Hülle

endlich vieler Punkte. Weiterhin ist ein n-dimensionales Polytop P im Gitter $\mathbb{Z}^n$ reflexiv, wenn das dazu duale Polygon

$$P^* = \{y \in \mathbb{R}^n | \langle y, x\rangle \geq -1 \; \forall x \in P\} \qquad (10)$$

wieder ein Gitterpolytop ist, also nur Eckpunkte in $\mathbb{Z}^n$ hat. Da für zwei Polytope P und P'

$$\mathrm{vol}(P) < \mathrm{vol}(P') \leftrightarrow \mathrm{vol}(P^*) > \mathrm{vol}(P'^*) \qquad (11)$$

gilt und das Volumen eines Gitterpolytops durch das Gitter nach unten beschränkt ist, gibt es in jeder Dimension eine obere Schranke für das Volumen reflexiver Gitterpolytope. Insbesondere bedeutet dies, dass es in jeder Dimension nur endlich viele reflexive Polytope gibt.

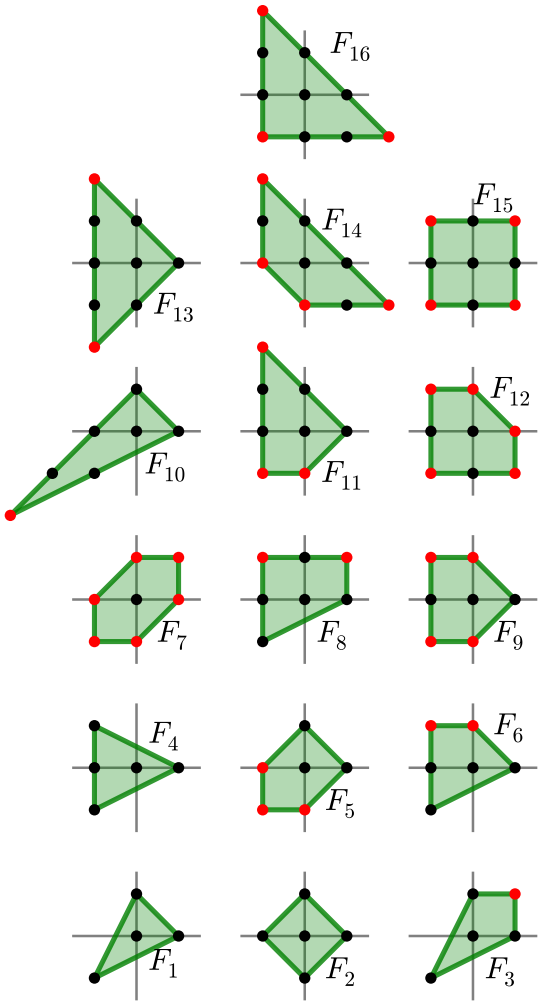


Abbildung 2: *Die 16 reflexiven Polygone. Gitterpunkte, deren zugehörige Divisoren eine generische elliptische Kurve einmal schneiden, sind rot markiert.*

In zwei Dimensionen existieren bis auf Gitterisomorphismen 16 verschiedene reflexive Polygone und in Sage lassen sich diese und die dazu gehörigen torischen Flächen leicht konstruieren [2, 6]. Beispielsweise erstellt

```
sage: F5 = ReflexivePolytope(2, 5)
sage: X5 = CPRFanoToricVariety(
....:   Delta_polar=F5)
sage: X0
2-d CPR-Fano toric variety
covered by 3 affine patches
sage: X5.anticanonical_hypersurface(
....:   monomial_points='all')
Closed subscheme of 2-d CPR-Fano
toric variety covered by 5 affine
patches defined by:
  a1*z0^2*z1^2*z4^3
  + a5*z0^2*z1*z2*z4^2
  + a7*z0*z1^2*z3*z4^2
  + a4*z0^2*z2^2*z4
  + a6*z0*z1*z2*z3*z4
  + a0*z1^2*z3^2*z4
  + a3*z0*z2^2*z3
  + a2*z1*z2*z3^2
```

die generische elliptische Kurve in der torischen Varietät dP_2, die in unserer Notation $\mathbb{CP}^2$ aufgeblasen an zwei Punkten entspricht. Divisoren von Y_4, die $\mathcal{E}$ einmal schneiden, werden zu Schnitten der Faserung M_8. In dem Fall von dP_2 sind das die Divisoren $V(z_0)$, $V(z_1)$ and $V(z_4)$. Für jeden solchen Divisor lässt sich der Schnittpunkt mit $\mathcal{E}$ ausrechnen. Zum Beispiel gilt

$$\mathcal{E} \cap V(z_0): \quad \left(0:1:1:1:-\frac{a_2}{a_0}\right). \qquad (12)$$

Für eine generische Wahl von Koeffizienten a_i gilt $a_i \neq 0$ und der Ausdruck ist daher wohldefiniert.

Um nun die Gruppe zu bestimmen, die von den Schnittpunkten von $V(z_0)$, $V(z_1)$ und $V(z_4)$ mit $\mathcal{E}$ erzeugt wird, ist es hilfreich, die Kurve in Weierstrassform (7) zu bringen. Obwohl dies allgemein immer möglich ist, kann es in der Praxis kompliziert sein, die entsprechende Transformation explizit zu bestimmen. In dem Fall der 16 torischen Gorenstein-Fano-Varietäten mit komplexer Dimension 2 lässt sich dieses Problem jedoch lösen [7]. Zunächst beobachtet man, dass die 16 Hyperflächengleichungen Spezialfälle von Hyperflächengleichungen in $\mathbb{CP}^2$, $\mathbb{CP}_{112}$ und $\mathbb{CP}^1 \times \mathbb{CP}^1$ sind. Insbesondere lässt sich das obige Beispiel in $\mathbb{CP}^2$ einbetten nachdem man die Aufblasung, die einen von $\mathbb{CP}^2$ zu dP_2 führt, rückgängig macht. Dies geschieht durch die Abbildung

$$(z_0 : z_1 : z_2 : z_3 : z_4) \mapsto (1 : 1 : z_2 : z_3 : z_4)\,, \qquad (13)$$

die die obige Gleichung abbildet auf

$$\begin{aligned} p = {} & a_1 z_4^3 + a_5 z_2 z_4^2 + a_7 z_3 z_4^2 + a_4 z_2^2 z_4 \\ & + a_6 z_2 z_3 z_4 + a_0 z_3^2 z_4 + a_3 z_2^2 z_3 + a_2 z_2 z_3^2\,. \end{aligned} \qquad (14)$$

Dies definiert offensichtlich eine nicht-generische Kubik in $\mathbb{CP}^2$.

Da für Hyperflächen in $\mathbb{CP}^2$, $\mathbb{CP}_{112}$ und $\mathbb{CP}^1 \times \mathbb{CP}^1$ Transformationen in Weierstrassform in der Literatur bekannt sind, sind wir nun in der Lage, die Weierstrassform beliebiger elliptisch gefaserter Hyperflächen auszurechnen. Insbesondere ist diese Funktionalität bereits in Sage implementiert:

```
sage: WeierstrassForm(p)
```

gibt eine Liste mit den Weierstrasskoeffizienten f und g zurück. Weiterhin lässt sich durch

```
sage: WeierstrassForm(p,
....:   transformation=True)
```

auch die explizite Abbildung der homogenen Koordinaten $(z_0 : \dots z_n)$ auf Punkte in Weierstrassform berechnen. Ist die elliptische Kurve in Weierstrassform samt der durch die Schnitte erzeugten Punkte gegeben, lässt sich die von den Punkten aufgespannte Gruppe ermitteln.

Es ist wichtig anzumerken, dass es für bestimmte Faserungen zusätzliche Schnitte geben kann, die nicht durch Divisoren der obigen Form definiert sind. Mit dem hier vorgestellten Ansatz ermittelt man jedoch eine Untergruppe der Mordell-Weil-Gruppe, in [8] die *torische* Mordell-Weil-Gruppe genannt, die *jede* Faserung mit der entsprechenden generischen Faser hat.

Ergebnisse und Ausblick

Mittels der in den vorherigen Abschnitten diskutierten Methode lassen sich die Untergruppen der Mordell-Weil-Gruppen, die von torischen Divisoren erzeugt werden, für generische elliptische Fasern in den 16 zweidimensionalen Gorenstein-Fano-Varietäten bestimmen. Es finden sich [8]

$$\mathbb{Z}\,,\mathbb{Z}\oplus\mathbb{Z}\,,\mathbb{Z}\oplus\mathbb{Z}\oplus\mathbb{Z}\,, \tag{15}$$

sowie folgende Gruppen mit Torsion:

$$\mathbb{Z}_2\,,\mathbb{Z}_3\,,\mathbb{Z}\oplus\mathbb{Z}_2\,. \tag{16}$$

Weiterhin gibt es zwei Varietäten, in denen die Mordell-Weil-Gruppe der generischen elliptischen Kurve trivial ist und drei weitere Fälle, in denen die generische Faser zwar eine Kurve von Geschlecht 1 ist, aber die Faserung keine globalen Schnitte besitzt.

Abgesehen von den verschiedenen physikalischen Szenarien, die in derartigen Stringkompaktifizierungen untersucht werden können, gibt es eine offensichtliche Verallgemeinerung der hier besprochenen Arbeit: Anstatt elliptische Kurven zu untersuchen, die Hyperflächen in torischen Varietäten sind, kann man allgemeiner elliptische Kurven betrachten, die *vollständige Durchschnitte* in höherdimensionalen torischen Räumen sind. Da die Anzahl reflexiver Polytope rapide mit der Dimension anwächst (es gibt 4319 dreidimensionale und $\sim 5\cdot 10^8$ vierdimensionale reflexive Polytope), erfordert eine solche Verallgemeinerung die Automatisierung der hier vorgestellten Rechnungen. Weiterhin stellt sich heraus, dass der hier vorgestellte Ansatz zum Ermitteln der Weierstrassform in höheren Dimensionen nicht mehr alle Fälle abdeckt und daher einen neuen Algorithmus erforderlich macht [9].

Literatur

[1] C. Vafa, *Evidence for F theory*, *Nucl.Phys.* **B469** (1996) 403–418, [hep-th/9602022].

[2] W. Stein *et. al.*, *Sage Mathematics Software (Version 6.3)*. The Sage Development Team, 2014. http://www.sagemath.org.

[3] W. Fulton, *Introduction to toric varieties*. No. 131. Princeton University Press, 1993.

[4] D. A. Cox, J. B. Little, and H. K. Schenck, *Toric varieties*. American Mathematical Soc., 2011.

[5] V. V. Batyrev, *Dual polyhedra and mirror symmetry for calabi–yau hypersurfaces in toric varieties*, in *J. Alg. Geom*, Citeseer, 1994.

[6] V. Braun and A. Novoseltsev, *Toric varieties module of Sage*. The Sage Development Team, 2014. http://www.sagemath.org.

[7] V. Braun, *Toric Elliptic Fibrations and F-Theory Compactifications*, *JHEP* **1301** (2013) 016, [arXiv:1110.4883].

[8] V. Braun, T. W. Grimm, and J. Keitel, *Geometric Engineering in Toric F-Theory and GUTs with U(1) Gauge Factors*, *JHEP* **1312** (2013) 069, [arXiv:1306.0577].

[9] V. Braun, T. W. Grimm, and J. Keitel, *Work in progress*.

3D-Grafik in der Schule mit Computeralgebra

J.-H. Müller, U. Schürmann
(Rivius-Gymnasium der Stadt Attendorn, Westfälische Wilhelms-Universität Münster)

jan.mueller@math.uni-dortmund.de
schuermann.uwe@uni-muenster.de

Zusammenfassung

Dreidimensionale Computergrafik, wie sie in Animationsfilmen oder Computerspielen Einsatz findet, ist heute für viele Schülerinnen und Schüler ein alltägliches Phänomen. Dabei beruht sie u.a. auf grundlegenden Konzepten der Analytischen Geometrie, was sie für den Mathematikunterricht in der Sekundarstufe II interessant macht.
Im Artikel wird anhand von ersten unterrichtlichen Erfahrungen gezeigt, wie dreidimensionale Computergrafik mit einem Computeralgebrasystem (CAS) gewinnbringend für Analytische Geometrie genutzt werden kann.

Vorbemerkungen und technische Voraussetzungen

Eine Gefahr von anwendungsorientiertem Mathematikunterricht besteht darin, dass die Bearbeitung interessanter Probleme in realistischen Kontexten mitunter viel kontextspezifisches Wissen voraussetzt und zusätzliche mathematische Verfahren verlangt, die ansonsten im Unterricht nicht thematisiert würden. Ein realitätsbezogener Mathematikunterricht erfordert demnach von der Lehrkraft die geplanten Inhalte didaktisch so zu reduzieren, dass das Verhältnis von echtem Realitätsbezug und unterrichtlichen Rahmenbedingungen wie curricularen Vorgaben, Zeitkontingent und mathematischen Fähigkeiten der Lerngruppe gewahrt bleibt.

Deshalb werden im Folgenden nur solche Unterrichtsbeispiele vorgestellt, die auch Lehrpersonen und Lernende ohne explizite Kenntnisse von dreidimensionaler Computergrafik oder gar Programmierung nutzen können. Die Beispiele können am Computer von den Lernenden selbst visualisiert werden, wodurch deutlich wird, dass es sich bei den vorgestellten Beispielen um echte Anwendungen aus der Computergrafik handelt.

Die im Artikel vorgestellten Unterrichtsbeispiele beruhen auf dem CAS Maxima[1]. Es ist kostenlos und kann auf verschiedenen Betriebssystemen genutzt werden. Somit bietet es den Vorteil, dass Schülerinnen und Schüler Unterrichtsmaterialien auf dem heimischen Computer nutzen können. Durch einfache Maxima-Dateivorlagen, wie sie auch in diesem Artikel zu finden sind, kann zudem der Aufwand der Bedienung des Programms für die Lerngruppe deutlich verringert werden. Im Folgenden wird Maxima hauptsächlich als dreidimensionales Zeichenwerkzeug genutzt. Darüber hinaus sollte Maxima natürlich auch in seiner eigentlichen Funktion als CAS Verwendung finden, z. B. wenn Schnittprobleme zwischen Objekten thematisiert werden und dazu lineare Gleichungssysteme gelöst werden müssen.

Unterrichtsbeispiele

Abbildung 1: *Haus in zwei Verschiedenen Ansichten*

Beschreibung von Objekten

Die grundlegende Idee der 3D-Grafik ist die mathematische Beschreibung realer Objekte im Raum. Eine erste Übung könnte deshalb darin bestehen, Lernende zunächst geradlinig begrenzte reale Objekte mit Koordinaten beschreiben und anschließend zeichnen zu lassen. Abbildung 1 zeigt ein Haus in zwei verschiedenen Ansichten. Hierbei müssen die Lernenden Abschätzungen zu Längen- und Größenverhältnissen anstellen und entscheiden, welche Teilaspekte eines Objektes zunächst

[1] Download: http://maxima.sourceforge.net/. Eine Einführung in das Erstellen von Grafiken mit Maxima findet sich z. B. hier: http://www.austromath.at/daten/maxima/zusatz/Grafiken_mit_Maxima.pdf (Zugriff am 18.02.14).

außer Acht gelassen werden können. Auf diese Weise gewinnen sie dreidimensionale Koordinaten, die als Eckpunkte des Hauses gezeichnet werden. Um ein besseres Abbild des Hauses zu erhalten, müssen anschließend Punkte miteinander verbunden und Flächen ggf. eingefärbt werden.

Abbildung 2 zeigt einen ersten Versuch mithilfe von Maxima die Seitenwand des Hauses, die in Abbildung 1 im linken Bild zu sehen ist, mit Punkten zu beschreiben. Im Maxima-Code wird zunächst mit load(draw) ein Grafik-Paket geladen, mit dem Grafiken flexibler gestaltet werden können als mit den Standardbefehlen von Maxima. Mit dem Befehl wxdraw3d wird eine 3D-Grafik als Zusammenstellung aus verschiedenen 3D-Grafikobjekten erstellt. Im vorliegenden Fall sind dies erstens eine Reihe von Punkten für die Seitenwand, zweitens eine Reihe von Punkten für das große Fenster und drittens eine Reihe von Punkten für das kleine Fenster (jeweils mit geschätzten Maßzahlen in Meter). Es werden weiterhin verschiedene Optionen angegeben. So sorgt z. B. POINTS_JOINED=TRUE dafür, dass die Punkte miteinander verbunden werden. Um tatsächlich ein geschlossenes Vieleck zu erzeugen muss der erste Punkt am Ende noch einmal angegeben werden.

```
load(draw)$
Seitenwand:points([[0,0,0],[4,0,0],[4,0,6],[0,0,6],[0,0,0]])$
GrossesFenster:points([[2.5,0,0],[3.5,0,0],[3.5,0,2],[2.5,0,2],[2.5,0,0]])$
KleinesFenster:points([[0.5,0,4],[0.8,0,4],[0.8,0,4.3],[0.5,0,4.3],[0.5,0,4]])$
wxdraw3d(
xyplane=0,color=red,point_type=7,points_joined=true,
xrange=[0,6],yrange=[0,6],zrange=[0,6],
Seitenwand,GrossesFenster,KleinesFenster);
```

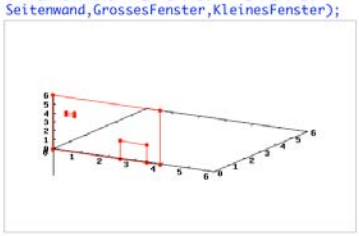


Abbildung 2: *Seitenwand des Hauses durch Punkte*

Die Wände des Hauses können natürlich ebenso gut mithilfe von Ausschnitten aus Ebenen in Parameterform beschrieben werden, wie Abbildung 3 mithilfe des Befehls PARAMETRIC_SURFACE zeigt.

Für die einzelnen Flächen werde die Parameter s und t entsprechend eingeschränkt: s liegt im gegebenen Beispiel zwischen 0 und 4 und t zwischen 0 und 6. Für die x-, y- und z-Koordinaten der Fläche werden Terme angegeben. Im Unterricht kann es z. B. gewinnbringend sein, die Lerngruppe mit der Syntax aus Abbildung 3 zu konfrontieren und einen Zusammenhang etwa zu folgender Darstellung der zur Seitenwand gehörenden Ebene in Parameterform herstellen zu lassen:

$$e_1 \;:\; x = \begin{pmatrix}0\\0\\0\end{pmatrix} + s\cdot\begin{pmatrix}1\\0\\0\end{pmatrix} + t\cdot\begin{pmatrix}0\\0\\1\end{pmatrix};$$

$$0 \le s \le 4,\;\; 0 \le t \le 6.$$

Ebenso gut kann den Schülerinnen und Schülern die Art der mathematischen Modellierung der Seitenwand hingegen völlig freigestellt werden. Erfahrungen aus dem Unterricht zeigen, dass es für Schülerinnen und Schüler anspruchsvoll ist, geeignete dreidimensionale mathematische Beschreibungen zu finden und diese in die Syntax von Maxima zu übersetzen. Der Einsatz von Arbeitsvorlagen und Hinweisen zur Syntax ist deshalb zu empfehlen. Die beiden gezeigten Beispiele stehen natürlich nur paradigmatisch für eine Vielzahl an Möglichkeiten, wie die gestellte Aufgabe gelöst werden kann. Uns erscheint es deshalb wichtig, Lernenden Hilfen aber keine Lösungen anhand zu geben. Der Mehrwert dieses Ansatzes entsteht durch einen anschließenden Austausch im Unterricht über die gewählten Methoden und der Abwägung ihrer jeweiligen Vor- und Nachteile.

```
load(draw)$
Seitenwand:parametric_surface(s,0,t,s,0,4,t,0,6)$
GrossesFenster:parametric_surface(s,0,t,s,2.5,3.5,t,0,2)$
KleinesFenster:parametric_surface(s,0,t,s,0.5,0.8,t,4,4.3)$
wxdraw3d(
xyplane=0,color=red,
xrange=[0,6],yrange=[0,6],zrange=[0,6],
Seitenwand,GrossesFenster,KleinesFenster
);
```

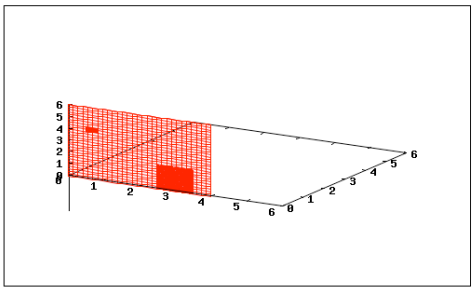


Abbildung 3: *Seitenwand des Hauses durch Ebenen*

Licht und Schatten

Um das geometrisches Objekt auf unserem Bildschirm möglichst realistisch aussehen zu lassen, ist es noch ein weiter Weg. Zu diskutieren wäre, was noch alles getan werden muss, um eine möglichst realistische Darstellung zu erreichen. Neben der Erweiterung unserer Seitenwand auf ein komplettes räumliches Modell bieten sich eine Vielzahl an Fragestellungen an, die eine kontextuelle Einbettung der üblichen curricularen Vorgaben der Analytischen Geometrie ermöglichen. Die folgenden Fragestellungen zeigen, dass unser Beispiel hierfür geeignet ist.

3D in 2D

Unabhängig von unserem Beispiel zeigte sich in Unterrichtssituationen, wie konsterniert viele Schülerinnen und Schüler auf die Frage reagieren, wie die dreidimensionale Wirkung von Objekten bei der Darstellung auf einer eigentlich zweidimensionalen Bildschirmfläche zustande kommt. In diesem Zusammenhang können sich Bezüge zum Kunstunterricht ergeben. So führt beispielsweise die Zentralperspektive mittels eines zu definierenden Fluchtpunktes auf das Erstellen von Geradengleichungen im Raum und das Ermitteln von Schnittpunkten mit einer zu definierenden „Bildschirmebene“. Und im Zusammenhang mit der Parallelperspektive kann zudem die Eigenschaft linearer Abhängigkeit

von Richtungsvektoren der Geradengleichungen thematisiert werden.

Schattenwurf
Nach der Thematisierung von Geraden- und Ebenengleichungen werden im Unterricht üblicherweise Schnittprobleme behandelt. Definiert man eine Lichtquelle im Raum, so kann analysiert werden, welchen Schatten das Haus in der Ebene wirft. In diesem Kontext ergeben sich Schnittprobleme auf natürliche Art und Weise. Wegen der hierfür zu lösenden Gleichungssysteme wird das CAS zunehmend bedeutsam.

Einfärbung der Seitenwände
Da die Lichtquelle die Seitenwände des Hauses unterschiedlich direkt beleuchtet, ist es ebenso naheliegend zu fragen, wie die Seitenwände (etwa in Graustufen) eingefärbt werden sollten. Qualitativ ist klar, dass eine Seitenwand, die der Lichtquelle zugewandt ist heller einzufärben ist als eine Seitenwand, die der Lichtquelle weniger stark zugewandt ist. Als Maß der „Zugewandtheit“ können Winkel genutzt werden, die z. B. zwischen Normalenvektoren zu einer Seitenwand und einer Verbindung der Seitenwand zur Lichtquelle ermittelt werden.

Dynamik
Ein Haus möchte man sich natürlich von allen Seiten anschauen, vergrößern, verkleinern oder in die eine oder andere Richtung verschieben können (etwa so, wie Architekten am Computer vorgehen, wenn sie ein Gebäude entwerfen). Anhand dieser Problemstellung können Matrizen und die Matrix-Vektor-Multiplikation eingeführt werden. Mit den Matrizeneinträgen können Schülerinnen und Schüler experimentieren und bekommen die Wirkung jeder Veränderung vom Programm veranschaulicht.

Ausblick und Grenzen des Ansatzes

Wie sich gezeigt hat, lassen sich einige interessante Anwendungen aus der Computergrafik mithilfe eines CAS im Unterricht realisieren. Die hier vorgestellten Beispiele sind durchaus erweiterbar, z. B. indem der Kontext Computergrafik ergänzt wird durch rechenminimierende Verfahren oder mathematischen Problemen, wie sie sich bei der Programmierung von Computerspielen stellen (vgl. Schürmann 2014 b). Es können somit viele Begriffe und Verfahren der Analytischen Geometrie anwendungsorientiert eingeführt bzw. vertieft werden. Ebenso lohnt sich ein Blick auf digitale Bildbearbeitung (viele Anregungen bietet der ISTRON-Band 9) oder das RGB-Farbmodell, welches für die Darstellung von Farben auf Computerbildschirmen verwendet wird. Auch dieser Kontext kann im Sinne der Analytischen Geometrie gedeutet werden (vgl. Schürmann 2014 a). Beide Kontexte, Computergrafik und Farbmodelle, bauen eine Brücke zum Kunstunterricht, wo perspektivische Zeichnungen und Farbmodelle ebenfalls von Bedeutung sind. Ein fächerverbindender Unterricht wird damit möglich.

Literatur

[1] Meyer, Jörg und Oldenburg, Reinhard (Hrsg.) (2006): Materialien für einen realitätsbezogenen Mathematikunterricht. *Istron-Schriftenreihe Band 9: Computeranwendungen*, Berlin, Hildesheim, Franzbecker-Verlag.

[2] Schürmann, Uwe (2014 a): Abbildungsmatrizen im Kontext von Farbtransformationen in Vektorgrafiken. *PM* 55/56, S. 43 – 47.

[3] Schürmann, Uwe (2014 b): 3D-Computerspiele und Analytische Geometrie. Maaß, Jürgen und Hans-Stefan Siller (Hrsg): Neue Materialien für einen realitätsbezogenen Mathematikunterricht 2, *ISTRON-Schriftenreihe*, S. 115 - 130.

The SYMBOLICDATA Project – from Data Store to Computer Algebra Social Network

H.-G. Gräbe, S. Johanning, A. Nareike
(Universität Leipzig)

graebe@informatik.uni-leipzig.de
nareike@informatik.uni-leipzig.de
simonjohanning@googlemail.com

Introduction

A powerful digital research infrastructure becomes increasingly important in today's networked and interlinked world. This includes digital support for dissemination of new papers, the refereeing process, conference submissions, and scientific communication within communities. Services such as MathSciNet, arXiv.org, EasyChair.org, or bibsonomy.org have been established and their usefulness is acknowledged by the larger scientific community. For mathematics in the large the vision of a *21st Century Global Library for Mathematics Research* (GDML) [2] matured during the last years.

Smaller academic communities, e. g. the Computer Algebra (CA) community, are challenged to organize their *intracommunity* communication infrastructure in a similar way. Infrastructural efforts are rarely acknowledged by the reputational processes of science, however, and are hence left to the casual engagement of volunteers unless led by leading-edge scientists of the community.

Open Source culture offers plenty of experience of how to substitute centrally organized projects by decentralized networked structures and indeed the new focus of SYMBOLICDATA version 3 is that of an intercommunity project, providing not only reliable access to data for testing and benchmarking purposes but also technical support for interlinking between different CA subcommunities.

In this paper we explain the technological basics of an RDF based semantic web of mathematics and discuss social interlinking to the *vision in large* of an emerging GDML.

RDF Basics

We refer to [6] for an overview of the goal, aims, concepts, infrastructure and history of the SYMBOLICDATA project, in particular for a more detailed motivation to use RDF concepts and Linked Data principles. The use of such principles and notations for the metadata of our data store is one of the core advances with version 3 of SYMBOLICDATA. In this section we give a brief overview of such concepts.

RDF is a data model which stores information as *triples*

s p o .

which are considered a *sentence* with subject s, predicate p and object o. Subjects and predicates are URIs (Uniform Resource Identifiers) while objects (or 'values') can be a URI or a literal (a string in quotes). There are shortcut notations of RDF, e. g. RDF-XML, JSON, or Turtle, and plenty of tools and parsers for the different formats.

As an example we consider the resource at

```
http://symbolicdata.org/XMLResources/
  IntPS/Czapor-86c.xml
```

that represents the XML record about the polynomial system

$$x^2 + y\,z\,a + x\,d + g$$
$$y^2 + x\,z\,b + y\,e + h$$
$$z^2 + x\,y\,c + z\,f + k$$

known as the *Czapor-86c* example. Such a polynomial system can be interpreted differently as polynomial ideal in different polynomial rings (see below).

We are going to store properties of the interpretation of that polynomial system as ideal

$$I \subset \mathbb{Q}[x, y, z, a, b, c, d, e, f, g, h, k]$$

(an *ideal configuration*, associated with that record) as new RDF subject. First, a new URI has to be assigned to such a new subject (using a sound naming scheme, not discussed here):

```
<http://symbolicdata.org/Data/Ideal/
  Czapor-86c.Flat>
```

or `sdideal:Czapor-86c.Flat` for short.

Now we can assign metainformation to that subject. Some of these properties can be extracted directly from the XML resource, others have to be calculated. Here is the record in Turtle notation:

```
sdideal:Czapor-86c.Flat a sd:Ideal ;
rdfs:comment "Flat variant of Czapor-86c" ;
   sd:hasDegreeList "3,3,3" ;
   sd:hasLengthsList "4,4,4" ;
   sd:relatedPolynomialSystem
      sdpol:Czapor-86c ;
   sd:hasVariables
      "x,y,z,a,b,c,d,e,f,g,h,k" .
```

This record contains 8 triples in Turtle shortcut notation. Since all triples share the same subject (the first line), Turtle compacts the notation by separating property–value pairs to the same subject with a semicolon. The `sd:`, `sdp:`, `sdideal:`, `sdpol:` and `sdxml:` prefixes are abbreviations for name-space prefixes[1].

The configuration above refers to the Polynomial System `sdpol:Czapor-86c`, described by the RDF record

```
sdpol:Czapor-86c
   a sd:IntegerPolynomialSystem ;
   sd:createdAt "1999-03-26" ;
   sd:createdBy sdp:Graebe_HG ;
   sd:relatedXMLResource
      sdxml:Czapor-86c.xml .
```

In other words, the *Czapor-86c.Flat* configuration is derived from the 'true' *Czapor-86c* example, the ideal

$$I' \subset S' = \mathbb{Q}(a,b,c,d,e,f,g,h,k)[x,y,z]$$

generated by the same polynomials in a different polynomial ring S'. Geometrically the latter represents a complete intersection of three (generic affine) quadrics over the field of rational functions in the given parameters. There is also a record on that configuration:

```
sdideal:Czapor-86c a sd:Ideal ;
   sd:createdAt "1999-03-26" ;
   sd:createdBy sdp:Graebe_HG ;
   sd:hasDegreeList "2,2,2" ;
   sd:hasLengthsList "4,4,4" ;
   sd:hasDegree "8"^^xsd:integer ;
   sd:hasDimension "0"^^xsd:integer ;
   sd:hasParameters "a,b,c,d,e,f,g,h,k" ;
   sd:parameterize sdideal:Czapor-86c.Flat ;
   sd:hasVariables "x,y,z" .
```

Czapor-86c is derived from the *Czapor-86c.Flat* configuration by parameterization with respect to the given parameters (see below for details). Note that the degree lists of *Czapor-86c.Flat* and *Czapor-86c* are different. The record contains some more metainformation: S'/I' is zero dimensional and has degree 8.

To operate on RDF data the databases (called *RDF Graphs*) have to be uploaded into an *RDF triple store*. SYMBOLICDATA uses the Virtuoso triple store [18] that provides also a SPARQL endpoint [14] to query the data, see [6] or our wiki [16] for more background information.

Modelling Polynomial Systems

RDF provides language tools to express metainformation in an interoperably searchable way that have to be applied in a *domain-specific way* to model topics from CA subcommunities. We explain such aspects of SYMBOLICDATA modelling again on the topic of Polynomial Systems. Similar considerations are required to model any other part of the SYMBOLICDATA database and the SYMBOLICDATA wiki [16] gives details about modelling other data (Free Algebras, G-Algebras, Geometry Proof Schemes etc.).

Polynomial Systems XML resources store lists of polynomials in distributive normal form with integer coefficients together with a complete list of variables (and, for modular systems, the modular base domain $GF(p)$). Hence even modular polynomial systems can be semantically considered as set $F = \{f_1, \ldots, f_s\}$ of polynomials in $S = \mathbb{Z}[x_1, \ldots, x_n]$ in the indeterminates $x_1, \ldots, x_n$ listed in the record.

Polynomial Systems Solving considers polynomial systems in different contexts. We have already shown how to construct the *Czapor-86c* ideal from the *Czapor-86c.Flat* resource. This is an example for a standard kind of interpretation where we divide indeterminates $x_1, \ldots, x_n$ into disjoint subsets $u_1, \ldots, u_k$ and $z_1, \ldots, z_m$ and consider the ideal

$$I' \subseteq S' = R(u_1, \ldots, u_k)[z_1, \ldots, z_m]$$

generated by the images of $f_1, \ldots, f_s$ in S' over the base coefficient field R. Here $u_1, \ldots, u_k$ are considered as parameters and $z_1, \ldots, z_m$ as variables. R is usually the field $\mathbb{Q}$ of rationals or a modular field $GF(p)$ (other settings are possible). S has the universal property that the canonical map on the indeterminates extends to a ring homomorphism $S \to S'$ in a unique way. We call such an interpretation of a Polynomial System resource as ideal generators in a polynomial rings S' *(ideal) configuration*. (Such configurations can be derived not only from Polynomial System resources but from other configurations as well.)

While SYMBOLICDATA version 2 would store each new configuration as a new resource, version 3 foresees (polynomial-time) transformations to express their relation to the basic resource.

For example, the `sd:homogenize` transformation derives a new configuration by homogenization (with respect to standard grading). Given the configuration F in S' and a new variable h we generate the homogenized polynomials $F^h = \{f_1^h, \ldots, f_s^h\}$ in

$$S'' = R(u_1, \ldots, u_k)[z_1, \ldots, z_m, h]$$

and the ideal I'' generated by F^h in S''. There is a natural ring homomorphism $\phi : S'' \to S'$ mapping $h \to 1$ and the polynomials $f_1^h, \ldots, f_s^h$ are called the *pull-back polynomials* of $f_1, \ldots, f_s$ with respect to ϕ. Note that the pull-back ideal $\phi^{-1}(I')$ contains the pull-back polynomials but is not necessarily generated by them.

The transformations `sd:flatten`, `sd:parameterize` and `sd:substitute` are defined by similar concepts. Compared to former SYMBOLICDATA versions the only restriction is that semantic-aware tools (that 'know' what a polynomial is) are required to generate configurations from given basic ones. We believe

[1] Note that for more clarity we use a sloppy notation here that is not fully compliant with the RDF notational standards.

this is not a real restriction since for serious computations on Polynomial Systems semantic-aware tools are required in any case. Such tools also have to provide a Polynomial Systems parser to input the basic XML examples. With your favorite CA software being aware of polynomial semantics it should be easy to implement the transformation modes required to obtain the different configurations. As a proof-of-concept, A. Nareike compiled the *sdsage package* [10] to be integrated with the Sagemath system [12].

Navigation within the Polynomial Systems Data

Special semantic knowledge is also required for navigation and identification of data. This topic is particularly important for intercommunity communication since one cannot expect researchers to be familiar with common practices of the different subcommunity. For a case-in-point let us again turn to Polynomial Systems Solving.

It is one of the challenges to check whether a Polynomial System configuration obtained from an external source is contained in the database, since the 'same' configuration may be given by polynomials with different variable sets and in different term orders. Thus for navigational purposes *fingerprints* of Polynomial System configurations are required that are independent of variable names and term orders. For a polynomial $0 \neq f \in S' = R(u_1, \ldots, u_k)[z_1, \ldots, z_m]$, invariants may be derived from the set $T(f)$ of terms. Every such polynomial has a distributive normal representation

$$f = \sum_{\alpha \in \mathbb{N}^m} c_\alpha \cdot z^\alpha, \quad \begin{array}{l} c_\alpha \in R(u_1, \ldots, u_k), \\ z^\alpha = z_1^{\alpha_1} \cdot \ldots \cdot z_m^{\alpha_m}, \end{array}$$

and $T(f) = \{z^\alpha : c_\alpha \neq 0\}$ is independent of the term order (but not of the variable names). There are two invariants that are well-defined for f regardless of variable names and orders – the number $|T(f)|$ of terms (the *length* of the polynomial f) and the pattern of the total degrees $(\deg(z^\alpha) : c_\alpha \neq 0)$ of the terms in $T(f)$. In particular, for $0 \neq f$ the maximum degree $\deg(f) = \max(\deg(z^\alpha) : c_\alpha \neq 0)$ is well-defined.

We use ordered lists of polynomial lengths and of maximum degrees as fingerprints of configurations and provide them precompiled as part of the metadata for a given configuration. Such a fingerprint can easily be computed by almost all semantic-aware tools. While configurations with different fingerprints are definitely distinct, there can be different examples with the same fingerprint. Although such fingerprints could be refined there was no need so far, since the examples with equal fingerprints are rare and can easily be inspected by hand.

Knowledge Frames and Social Framing

The SYMBOLICDATA project can be seen as part of the worldwide efforts towards a *World Digital Mathematical Library* (WDML). The most viable contributions on that way are the EuDML project [3], funded by the European Union during 2010–2013, and the WDML project [19], triggered by the US National Research Council and heavily supported by the IMU – the International Mathematical Union – and the Alfred P. Sloan Foundation. There are other activities on the way in that direction, in particular by Wolfram Research Inc., developing *Wolfram Alpha* as one of the most mature online resources of structured mathematical knowledge.

One of the big challenges within these projects is the question "How to structure and represent mathematical knowledge?" Such a question is not only (and probably not even in the first plan) a question about mathematics but also about mathematicians, their common efforts, social relations, and the way how they organise common work. Minsky's well established concept of *knowledge frames* as "artificial intelligence data structure used to divide knowledge into substructures by representing 'stereotyped situations'" [4] suggests that such "stereotyped situations", in our case different research contexts with complex social structures and interrelations, play an important role in the way how knowledge structures emerge and evolve. A. and M. Kohlhase [8] emphasized the importance of such practices for mathematicians and formalized them in the (slightly different) notions of *theory graphs* and *framing* (the latter considered as 'reframing' in a Minsky inspired terminology [7]).

Computer Algebra at the border of mathematics and computer science played an important and special role during any computer triggered technological breakthrough in mathematics in the last decades. It is a first class challenge to the CA community to organize the changes towards a GDML for their own community and to contribute to the changes at large.

The setting of (not only) the SYMBOLICDATA Project to address the needs of the CA community and their severe subcommunities – considered as a "major social frame" with many intimately related "smaller social frames" – is a first class playground

- to study and explore own cooperate contexts and practices under the special aspects and formalization requirements of modern semantic web technologies,
- to develop and communicate best practices along the IMU recommendations [11],
- and to advance towards a better understanding of their requirements and consequences,

since we have researchers well educated both in mathematics and computer science and the scientific CA community is small enough to allow for social relations in a less institutionalized way.

Towards CASN — a Computer Algebra Social Network

Such a challenge meets other visions of the SYMBOLICDATA Project, in particluar to collect not only benchmark and testing data but also valuable background in-

formation about the records in the database, e. g. information about papers, people, history, systems, concerned with the examples in our collection. RDF is particularly suited for this, for it provides both the concept of typeless URIs within a typed world to point to resources of different types in a uniform way and it allows linking to foreign URIs in other databases to build up a semantic network with many nodes where the node at `symbolicdata.org` is only one in the multitude of nodes of such a (distributed) Computer Algebra Social Network.

There are plenty of activities towards such an 'e-science world', in particular by the *Association of German Libraries* [5], by the *Zentralblatt Mathematik* [13], by the MKM community [9] and others.

As explained in the introduction it is a great challenge to small scientific communities to adopt such developments for its own scientific communication processes and to join forces with other scientific communities to get own requirements publicly recognised. A first step in such a direction is a more detailed description of ongoing scientific processes using standard RDF terminology.

SYMBOLICDATA started such efforts within the CA scientific community, collecting and presenting

- information about scientific activities of people – as of Sept. 2014 the SYMBOLICDATA People database contains `foaf:Person` entries of 708 people from the CA scientific community that can be explored via the SYMBOLICDATA SPARQL endpoint [14] – it is mirrored at [15],
- information about upcoming conferences – the information is extracted via SPARQL query to [15] from `http://symbolicdata.org/casn/UpcomingConferences/` and displayed in the Wordpress based site of the German Fachgruppe [20],
- information about past conferences and conference reports with references to the SYMBOLICDATA People database about speakers and organizers,
- information about German CA working groups and the SPP 1489 projects with references to the SYMBOLICDATA People database,
- keyword enriched information about scientific publications in the CA-Rundbrief of the German Fachgruppe using the `dcterms` ontology – the information is displayed in the Wordpress based site of the German Fachgruppe [20],
- and semantic annotations to news in the blog of the German Fachgruppe as instances of RDF type `sioc:BlogPost` and `bibo:Document` attached to the blog post URL.

A particularly interesting project was started in cooperation with *Zentralblatt Mathematik* (ZBMath) towards a better author disambiguation. During the last years ZBMath spent much effort for better author disambiguation primarily using language processing technology to retrieve data tracks of people from their stock of abstracts, see [13]. The SYMBOLICDATA references within our People database offer additional insight into people activities (and – if properly recognised by the community – allows for active support and influence on a process that touches both vivid personal and scientific interests of the community), and we started to align SYMBOLICDATA People URIs with ZBMath URIs in the *ZBMath Person Matches* table at `http://symbolicdata.org/Data/ZBMathPeople/`.

Despite the importance of the ongoing 'Big Data' harvesting processes for the moment this and other activities suffer from little attention of a broader CA audience. In our talk at the CICM-14 conference [6] we asked: "How turn passive users (listed in the SYMBOLICDATA People database) into active ones?" A first hurdle is authentication. Nowadays there are three different ways to solve that problem:

1. The 'classical one': Set up a local database with user/password; users have to administer plenty of such application/user/password records (of course best using RDF technology).
2. The 'Google one': Use the (OAuth based) authentication service of one of the 'big players'. Your advantage: one password fits all, but who knows what the guys are tracking ...
3. The WebID approach: Use your browser certificate and a FOAF profile managed by your own to provide access.

   You register with SYMBOLICDATA your FOAF profile, we will take the challenge stored there and offer it mixed up with our challenge to your browser. If the browser (with your certificate, imported into the browser from your source) returns the correct answer we know that's you sitting in front of it.

We started to set up such an infrastructure along the FOAF standards as 'proof of concept':

- The SYMBOLICDATA People database contains 'lightweight' FOAF profiles.
- For any person interested to join actively the CASN we generate a `foaf:PersonalProfileDocument` that relates this 'lightweight' FOAF profile with your own FOAF profile as `foaf:primaryTopic`.
- For a limited number of people from the German Fachgruppe (yet as passive users) we created such profiles and use them to display information about the different boards of the German Fachgruppe at their website [20].
- You can take over your own profile at any convenient web place under your control and extend it to meet the authentication requirements and tell us about that place.

  We change your PersonalProfileDocument and you are ready actively to join the CASN process.

The vision of a Computer Algebra Social Network goes far beyond that:

- Maintain at your local site up-to-date information about your working group and its people in a consistent RDF format as e. g. the AKSW team [1] does at `http://aksw.org/Team.html`.
- Maintain your FOAF profile at a personal page containing all important public up-to-date information about your activities in a consistent RDF format as e. g. the AKSW team member Natanael Arndt does at `http://aksw.org/NatanaelArndt.html`.
- Organize a regular harvesting process within the scientific community for such information to feed common pages at sites as e. g. `http://www.computeralgebra.de`, and to substitute part of the information centrally stored at SYMBOLICDATA today by decentrally managed one.
- Set up and run within the scientific community a semantic-aware Facebook-like Social Network and contribute to it about all topics around Computer Algebra using tools that express your contributions in an RDF-based syntax.

The last point sounds quite visionary but it is in no way utopic. The AKSW team provides a first prototype of a tool that realizes the challenging concept of a *Distributed Semantic Social Network* [17] to be running, in contrast to Facebook, on a multitude of independent nodes all over the world. Since the concept works also with a single node and can be extended later on, we set up such a node at `symbolicdata.org` for testing, see our CASN wiki page [16] for more information. Even if all this is very pre-alpha yet, the future is already on the way. Don't miss the train.

References

[1] The Agile Knowledge Engineering and Semantic Web Group at Leipzig University. `http://aksw.org/About.html` [2014-02-19]

[2] Developing a 21st Century Global Library for Mathematics Research. Report of the Committee on Planning a Global Library of the Mathematical Sciences. The National Academies Press 2014. `http://www.nap.edu/catalog.php?record_id=18619` [2014-09-24]

[3] EuDML – the European Digital Mathematical Library. `https://eudml.org/` [2014-09-23]

[4] Frame at Wikipedia. `http://en.wikipedia.org/wiki/Frame_(artificial_intelligence)` [2014-09-23]

[5] Gemeinsame Normdatei (GND). Informationsseite der Deutschen Nationalbibliothek. `http://www.dnb.de/DE/Standardisierung/GND/gnd_node.html` [2014-09-13]

[6] H.-G. Gräbe, A. Nareike, S. Johanning. The SymbolicData Project — Towards a Computer Algebra Social Network. In: Workshop and Work in Progress Papers at CICM 2014. CEUR-WS.org vol. 1186. `http://ceur-ws.org/Vol-1186/#paper-21` [2014-09-13]

[7] S. Kaufman, M. Elliott, D. Shmueli: Frames, Framing and Reframing. In: G. and H. Burgess (eds.): Beyond Intractability. Conflict Information Consortium, University of Colorado, Boulder. Posted: September 2003. `http://www.beyondintractability.org/essay/framing` [2014-09-23]

[8] A. and M. Kohlhase: Spreadsheet Interaction with Frames: Exploring a Mathematical Practice. In: Intelligent Computer Mathematics, Lecture Notes in Computer Science Volume 5625, 2009, pp 341–356.

[9] Mathematical Knowledge Management. The MKM interest group. `http://www.mkm-ig.org/` [2014-09-13]

[10] A. Nareike. The SYMBOLICDATA sdsage package. `http://symbolicdata.org/wiki/PolynomialSystems.Sage` [2014-02-28]

[11] J. Pitman, C. Lynch: Planning a 21st Century Global Library for Mathematics Research. Notices of the AMS, August 2014.

[12] Sage – a free open-source mathematics software system. `http://www.sagemath.org` [2014-02-19]

[13] U. Schöneberg, W. Sperber. POS Tagging and its Applications for Mathematics. In: Intelligent Computer Mathematics. Lecture Notes in Computer Science, Volume 8543, 2014, pp 213–223.

[14] The SYMBOLICDATA SPARQL Endpoint. `http://symbolicdata.org:8890/sparql` [2014-02-19]

[15] The CASN SPARQL Endpoint. `http://symbolicdata.org:8891/sparql` [2014-09-13]

[16] The SYMBOLICDATA Project Wiki. `http://wiki.symbolicdata.org` [2014-09-13]

[17] S. Tramp et al.: An Architecture of a Distributed Semantic Social Network. In: Semantic Web Journal 2012, Special Issue on The Personal and Social Semantic Web.`http://www.semantic-web-journal.net/sites/default/files/swj201_4.pdf` [2014-09-23]

[18] Virtuoso Open-Source Edition. `http://virtuoso.openlinksw.com/` [2014-02-19]

[19] World Digital Mathematics Library (WDML). `http://www.mathunion.org/ceic/wdml/` [2014-09-23]

[20] Website of Fachgruppe Computeralgebra `http://www.fachgruppe-computeralgebra.de/` [2014-03-06]

Shared Memory Concurrency for GAP

Reimer Behrends
(TU Kaiserslautern)

behrends@mathematik.uni-kl.de

The GAP system [3], as it is introduced on the GAP Web site, is an open-source system for computational discrete algebra, with particular emphasis on computational group theory. It provides a programming language, an extensive library of functions implementing algebraic algorithms written in the GAP language as well as large data libraries of algebraic objects. The kernel of the system is implemented in C, and the library is implemented in the GAP language. Both the kernel and the library were originally sequential and did not support parallelism.

In the 4-year long EPSRC project "HPC-GAP: High Performance Computational Algebra and Discrete Mathematics" (`http://www-circa.mcs.st-and.ac.uk/hpcgap.php`), started in September 2009, a team of researchers at the University of St Andrews reengineered the GAP system to allow parallel programming in it using both shared and distributed memory approaches. In this article, we report on the results of the HPC-GAP project with respect to parallelizing execution on multicore systems with shared memory. The concurrency model that we employ aims at making concurrency facilities available to GAP users, while preserving the existing codebase (hundreds of thousands of lines of code and data) with as few changes as possible. To this end, the model preserves the appearance of sequentiality on a per-thread basis by containing each thread within its own data space, while making concurrent interaction possible through selective sharing of data and by allowing the migration of objects between data spaces.

Shared Memory: Regions

The fundamental concept through which HPC-GAP regulates concurrent interaction is that of a *region*. A region is a data space in memory; each GAP object belongs to precisely one region. Access to GAP objects is controlled through ownership of regions: in order to access objects within a region, a thread must first obtain exclusive or shared ownership of the region. As a rule, when a thread has exclusive ownership, it can read and modify objects within the region; when it has shared ownership, it can only read the objects. Violating these access rules results in a runtime error, thus disallowing access without ownership entirely.

This approach effectively makes data races – situations where two threads concurrently modify the same object or where one thread modifies an object while another reads it – impossible, eliminating what is typically the major source of software defects in multi-threaded systems. In this regard, our design differs from most mainstream programming languages, which tend to not enforce this property (see, e.g., [2]).

Each thread has an associated *thread-local region*, of which it always has exclusive ownership. In particular, objects in a thread's thread-local region can never be accessed by other threads.

When using only a single thread, the thread's behavior is essentially indistinguishable from a sequential system to the end user (though underlying libraries may employ concurrency). This design intentionally hides the complexities of parallelism from users who do not have the necessary expertise in parallel programming. They can safely continue to use the system as though it were sequential code while still benefiting from any parallelized libraries and packages that others have written. By having their actions compartmentalized within the main thread's region, their code will not interfere with any parallelized code and the parallelized code will not interfere with their code.

In addition, code that requires concurrent interaction can also create one or more *shared regions*. In order to obtain exclusive or shared ownership of a shared region, a thread must first acquire the read-write lock associated with the region (a read lock for shared ownership, a write lock for exclusive ownership). Shared regions are typically created through the `ShareObj(obj)` function, which creates a new shared region and moves `obj` inside.

Threads can claim shared or exclusive ownership to a region through a new control structure, the `atomic` statement. An atomic statement takes one or more arguments, optionally preceded by a `readonly` or `readwrite` descriptor. The thread then gains shared (`readonly`) or exclusive (`readwrite`) ownership for the duration of the `atomic` statement. Example:

```
atomic readonly obj, readwrite obj2 do
  # Thread has shared ownership of the region
  # containing <obj> and exclusive ownership
  # of the region containing <obj2>. Ownership
  # of both regions will be relinquished when
  # control flow reaches the "od" keyword at
  # the end of the code block.
od;
```

For convenience, constant data (esp. large tables) can be stored in a special *read-only region*. All threads have permanent shared ownership of the read-only region, meaning that they can access the data in it as though it were a shared region for which they hold a read lock, but without the extra overhead and inconvenience of having to aquire one each time. The `MakeReadOnly(obj)` primitive puts `obj` in the read-only region.

Finally, HPC-GAP knows a *public region*, to which all threads have constant read and write access. This is the one exception to the rule that one has to have exclusive ownership in order to modify objects within a region. As a consequence, only objects that expose exclusively atomic operations to GAP (i.e., operations that cannot create data races) can be stored in the public region. The public region therefore only holds certain special kinds of objects (such as those needed by the various HPC-GAP synchronization primitives) as well as certain types of immutable objects. Functions, for example, are always contained in the public region[1], so that the same function can be called by multiple threads.

Objects can be migrated or copied between regions. Migration, unlike copying, is an inexpensive operation in that it simply changes the region membership descriptor of an object. Migration can be performed through the `MigrateObj(obj, target)` function, which migrates objects between arbitrary regions, or the `AdoptObj(obj)` function, which migrates an object to the current thread-local region.

Multi-threading

In order to use multiple threads, GAP programmers are encouraged to use the task library, which provides convenient abstractions over low-level thread management[2]. The task library uses the concept of futures [3] similar to mainstream programming languages such as C# and Java.

The primary interface provided consists of the `RunTask()` and `TaskResult()` functions:

```
task := RunTask(f, x1, ..., xn);
result := TaskResult(task);
```

`RunTask()` takes a function f and zero or more values x_i as its arguments. It then evaluates $f(x_1, \ldots, x_n)$ asynchronously in a different thread. `RunTask()` finishes immediately (i.e. while f is still being executed in parallel), returning a task descriptor as a result. The `TaskResult()` function takes a task descriptor as its sole argument, waits until that task has completed and returns the result of the underlying evaluation of $f(x_1, \ldots, x_n)$.

The task library provides additional primitives (such as waiting for the first of a set of tasks to finish, having the execution of tasks triggered by a condition, or the cancellation of tasks). But `RunTask()` and `TaskResult()` are the core primitives in that they allow the concurrent execution of arbitrary computations with an interface that is only slightly more complex than a regular function call.

Note that the evaluation of $f(x_1, \ldots, x_n)$ is performed in a separate thread; in order to make this possible, any thread-local arguments are implicitly copied to the new thread; likewise, the result (if thread-local) is copied back to the thread-local region of the thread performing the `TaskResult()` call.

Example: Parallel Matrix Multiplication

Figure 1 uses the standard parallel matrix multiplication algorithm to illustrate the above concepts.

```
 1 ParMatrixMultiplyRow := function(m1, m2, i)
 2   local result, j, k, n, s;
 3   result := [];
 4   atomic readonly m1, readonly m2 do
 5     n := Length(m1);
 6     for j in [1..n] do
 7       s := 0;
 8       for k in [1..n] do
 9         s := s + m1[i][k] * m2[k][j];
10       od;
11       result[j] := s;
12     od;
13   od;
14   return result;
15 end;
16
17 ParMatrixMultiply := function(m1, m2)
18   local tasks, result;
19   ShareObj(m1);
20   ShareObj(m2);
21   atomic readonly m1, readonly m2 do
22     tasks :=
23       List([1..Length(m1)],
24         i -> RunTask(ParMatrixMultiplyRow,
25                      m1, m2, i));
26     result := List(tasks, TaskResult);
27   od;
28   atomic readwrite m1, readwrite m2 do
29     AdoptObj(m1);
30     AdoptObj(m2);
31   od;
32   return result;
33 end;
```

Figure 1: *Parallel matrix multiplication.*

The `ParMatrixMultiply()` function in lines 17–33 performs the actual multiplication. It takes two square matrices as its arguments. The `ShareObj()` primitives in lines 19–20 create new shared regions for these matrices and migrate them inside. The `atomic` construct in lines 21–27 then acquires read locks for the two shared regions, granting the current thread shared ownership. The substance of the work occurs in lines 22–25, where one task is started for each row vector in `m1`. This is done through the standard `List()` function, which takes two arguments, a list and a function, and applies the function to each element in the list. This function performs a `RunTask()` call; thus, the result of `List()` is a list of task descriptors. The second `List()` invocation in line 26 then maps each of the task descriptors to their results, returning a list of the

[1] Obviously, this holds only for the function's *code*, which never changes, not any *data* that it operates on.

[2] Of course, low-level thread and concurrency primitives are also available for advanced concurrency constructs.

row vectors of the product of the matrices. In lines 28–31, the two arguments are migrated back to the current thread's thread-local region; the `AdoptObj()` calls effectively undo the `Shareobj()` calls in lines 19–20.

The actual multiplication operation for each row vector takes place in the `ParMatrixMultiplyRow` function in lines 1–15. This is the basic matrix multiplication algorithm for a given row vector. The only difference compared to the sequential version is the `atomic` statement in lines 4–13, which gives the task temporary shared ownership of the matrix regions. Note that this is necessary even though the main thread had already done the same: the tasks have no knowledge of what the main thread is currently doing and thus have to also claim shared ownership on their own. Because all tasks require only read access to the matrices, shared ownership is sufficient and all tasks can execute in parallel (as long as the hardware permits).

Conclusion

A public beta release of HPC-GAP is planned for the near future. If you would like access to the current pre-release version, please contact the author or the GAP Group.

References

[1] GAP – Groups, Algorithms, and Programming, Version 4.7.5; 2014 `http://www.gap-system.org`

[2] Hansen, P. B. Java's Insecure Parallelism. *SIGPLAN Notices*, v.34 n.4, p.38–45, April 1999.

[3] Friedman, D. P. and Wise, D. S. The Impact of Applicative Programming on Multiprocessing. *1976 International Conference on Parallel Processing*, p.263–272.

The GAP package `SingularInterface`

M. Barakat, M. Horn, F. Lübeck, O. Motsak, M. Neunhöffer, H. Schönemann
(TU Kaiserslautern, JLU Gießen, RWTH Aachen University, TU Kaiserslautern, triAGENS GmbH, TU Kaiserslautern)

`barakat@mathematik.uni-kl.de, max.horn@math.uni-giessen.de, frank.luebeck@math.rwth-aachen.de, motsak@mathematik.uni-kl.de, max@9hoeffer.de, hannes@mathematik.uni-kl.de`

What is `SingularInterface`?

The GAP package `SingularInterface` is a highly efficient and robust unidirectional low-level interface to SINGULAR [2, 3]. It is the outcome of an intensive collaboration between core developers of both systems.

The goal of this interface is to map all of SINGULAR's powerful functionality into GAP. To achieve this it automatically wraps *all* SINGULAR datatypes and exports *all* of SINGULAR's interface procedures to GAP.[1] Furthermore, all procedures of any contributed library can be loaded on demand.[2]

This package is a rather "faithful" image of SINGULAR; it does not make an extensive attempt for a better integration of SINGULAR into the GAP ecosystem. This is intentionally left to other packages, which are free to realize this in different ways.

The development of `SingularInterface` has reached a β-phase and is already actively used in some research projects. We hope to attract more users in the near future, whose feedback will be crucial for a successful further development.

How to get it?
To download and install `SingularInterface` please follow the instructions on

`http://gap-system.github.io/SingularInterface/`

If you are reading this article, say, more than one year in the future, and have a recent GAP installation, then hopefully you already have a working version of this package.

To check that the package has been successfully installed, start GAP and type:

```
gap> LoadPackage( "SingularInterface" );
true
```

To see all imported procedures type:

```
gap> SI_<press TAB twice>
```

The SINGULAR library "`standard.lib`" is loaded by default. To see all imported SINGULAR library procedures type:

```
gap> SIL_<press TAB twice>
```

To load any other library, e.g. "`matrix.lib`", type:

```
gap> SI_LIB( "matrix.lib" );
true
```

[1] With the prefix "`SI_`" prepended to their names.
[2] They appear in GAP with the prefix "`SIL_`" prepended to their names.

Goals

The motivation behind developing `SingularInterface` is the increasing interest of various research projects in combining the strength of both systems: GAP users get access to SINGULAR's polynomial arithmetic and highly optimized Gröbner basis engine. SINGULAR users gain a second front end language for this engine – in addition to the current SINGULAR language – with an advanced object model primarily designed for modeling higher mathematical structures, as well as access to GAP as an expert system for group and representation theory.

An example

Here is a short example in SINGULAR 4.0.1 demonstrating some basic procedures. On the left you see the SINGULAR code, on the right the corresponding GAP code. SINGULAR uses "=" for assignments and suppresses any output while GAP uses ":=" for assignments and triggers the so-called `View`-method, which gives a very brief description of the object (unless suppressed by a trailing ";;"). Basically, SINGULAR's `print` procedure is mapped to the so-called `Display`-method in GAP. The current version of `SingularInterface` is `2014.09.23`.

Start by loading `SingularInterface` in GAP.

```
> // standard.lib is automatically preloaded
> // this example needs no further libraries
```

```
gap> LoadPackage( "SingularInterface" );
true
```

Define the ring $R := \mathbb{Q}[x_0, x_1, x_2, x_3]$ (with the monomial ordering `degrevlex`):

```
> ring R=0,(x0,x1,x2,x3),dp;

> short=0;

> option(redTail);
```

```
gap> R := SI_ring( 0, "x0..3", [["dp",4]] );
<singular ring, 4 indeterminates>
gap> ## short=0 is the default, disable by:
gap> ## Singular( "short=1" );
gap> SI_option( "redTail" );
true
```

Define the polynomial $(x_1 + x_3)^2$:

```
> poly p=(x1+x3)^2; p;
x1^2+2*x1*x3+x3^2
```

```
gap> AssignGeneratorVariables( R );
#I  Assigned the global variables
#I   [ x0, x1, x2, x3 ]
gap> p := (x1+x3)^2;
x1^2+2*x1*x3+x3^2
gap> IsSingularPoly( p );
true
```

Define the ideal $I := \langle x_0^2 - x_1x_3, x_0x_1 - x_2x_3 \rangle \lhd R$:

```
> ideal I=x0^2-x1*x3,x0*x1-x2*x3;

> print(I);
x0^2-x1*x3,
x0*x1-x2*x3
```

```
gap> I := SI_ideal([x0^2-x1*x3, x0*x1-x2*x3]);
<singular ideal, 2 gens>
gap> Display( I );
x0^2-x1*x3,
x0*x1-x2*x3
```

The corresponding matrix i:

```
> def i=matrix(I);

> print(i);
x0^2-x1*x3,x0*x1-x2*x3
```

```
gap> i := SI_matrix( I );
<singular matrix, 1x2>
gap> Display( i );
x0^2-x1*x3,x0*x1-x2*x3
```

The sum $I + I$ means the sum of ideals:

```
> J=I+I;

> print(J);
x0^2-x1*x3,
x0*x1-x2*x3
```

```
gap> J := I + I;
<singular ideal, 2 gens>
gap> Display( J );
x0^2-x1*x3,
x0*x1-x2*x3
```

Whereas $i + i$ means the sum of matrices:

```
> print(i+i);
2*x0^2-2*x1*x3,2*x0*x1-2*x2*x3
```

```
gap> Display( i + i );
2*x0^2-2*x1*x3,2*x0*x1-2*x2*x3
```

The squared ideal $I^2 \lhd R$:

```
> def I2=I^2;

> print(I2);
x0^4-2*x0^2*x1*x3+x1^2*x3^2,
x0^3*x1-x0*x1^2*x3-x0^2*x2*x3+x1*x2*x3^2,
x0^2*x1^2-2*x0*x1*x2*x3+x2^2*x3^2
```

```
gap> I2 := I^2;
<singular ideal, 3 gens>
gap> Display( I2 );
x0^4-2*x0^2*x1*x3+x1^2*x3^2,
x0^3*x1-x0*x1^2*x3-x0^2*x2*x3+x1*x2*x3^2,
x0^2*x1^2-2*x0*x1*x2*x3+x2^2*x3^2
```

The Gröbner basis of the ideal I is returned as a new different (but mathematically equal) ideal G:

```
> def G=std(I);

> print(G);
x0*x1-x2*x3
x0^2-x1*x3
x1^2*x3-x0*x2*x3
```

```
gap> G := SI_std( I );
<singular ideal, 3 gens>
gap> Display( G );
x0*x1-x2*x3,
x0^2-x1*x3,
x1^2*x3-x0*x2*x3
```

The syzygies of the generators of G are the columns of the SINGULAR datatype `module`[3]:

```
> def S=syz(G); S;
S[1]=x0*gen(1)-x1*gen(2)-gen(3)
S[2]=x1*x3*gen(1)-x2*x3*gen(2)-x0*gen(3)
> print(S);
x0, x1*x3,
-x1,-x2*x3,
-1, -x0
```

```
gap> S := SI_syz( G );
<singular module, 2 vectors in free module of
 rank 3>
gap> Display( S );
x0, x1*x3,
-x1,-x2*x3,
-1, -x0
```

To access the second column of `S` use:

```
> S[2];
x1*x3*gen(1)-x2*x3*gen(2)-x0*gen(3)
> print(S[2]);
[x1*x3,-x2*x3,-x0]
```

```
gap> S[2];
<singular vector, 3 entries>
gap> Display( S[2] );
[x1*x3,-x2*x3,-x0]
```

To access the first entry of the second column of `S` use:

```
> S[2][1];
x1*x3
> p-S[2][1];
x1^2+x1*x3+x3^2
```

```
gap> S[2][1];
x1*x3
gap> p - S[2][1];
x1^2+x1*x3+x3^2
```

To create a matrix use:

```
> matrix m[3][2]=x0,x3, x1,x2, x3,x0;

> print(m);
x0,x3,
x1,x2,
x3,x0
```

```
gap> m := SI_matrix(R,3,2,"x0,x3,x1,x2,x3,x0");
<singular matrix, 3x2>
gap> Display( m );
x0,x3,
x1,x2,
x3,x0
```

To extract the (2,1)-entry from the matrix use:

```
> m[2,1];
x1
```

```
gap> m[[2,1]];
x1
```

The sum of the `module` `S` and the `matrix` `m` is their augmentation:

```
> print(S+m);
x0, x1*x3, x0,x3,
-x1,-x2*x3,x1,x2,
-1, -x0,   x3,x0
```

```
gap> S + m;
<singular module, 4 vectors in free module of
 rank 3>
gap> Display( S + m );
x0, x1*x3, x0,x3,
-x1,-x2*x3,x1,x2,
-1, -x0,   x3,x0
```

The development

How does it work?
`SingularInterface` aims to be a comprehensive bridge between GAP and SINGULAR with as little overhead as possible, both in terms of speed and memory. To achieve this goal, some key design decisions were crucial:

(1) Avoid converting data between the two systems, as conversions are expensive.

(2) Automate generating function bindings as much as possible.

(3) We stay relatively low-level with the interface, mostly refraining from trying to change SINGULAR behaviour to be more GAP like (with one exception, see below).

Regarding (1), on the GAP side we use "wrapper objects" around SINGULAR objects. That is, tiny GAP objects which essentially consist of a pointer to the actual SINGULAR object (such as a polynomial) plus some meta information (types, attributes). With the exceptions of small integers and strings, we never automatically convert SINGULAR objects into "native" GAP

[3]The datatype `module` in SINGULAR 4.0.1 is in first approximation a specialized sparse data structure for column oriented matrices with compressed columns, where each column has the datatype `vector`. For more details see the `SingularInterface` manual [1].

objects. Indeed, in most cases that would be pointless, as for further operations with the object, we would have to convert it back into a SINGULAR object anyway. So when you have a SINGULAR polynomial `p` and call `SI_deg(p)`, in the background, `SingularInterface` extracts the pointer to the SINGULAR object from it, then invokes the SINGULAR interpreter C code for `deg`, and returns the result to the user (since the result is a small integer, it is not wrapped).

Of course when necessary, you can still convert from and to "native" GAP objects (although this is one of the areas where there is still work to be done, in order to cover all possible SINGULAR coefficient ring types).

A major complicating factor for this approach is that GAP and SINGULAR use very different memory management systems (GAP uses a so-called "generational moving garbage collector", in which objects change their position in memory over time, while SINGULAR use a more traditional "malloc" system plus reference counting, and expects objects to stay at a fixed position in memory). With some clever tricks, aided by a few small but helpful changes on the SINGULAR side, this now works extremely well.

Regarding (2), here is one example: From the data structures the SINGULAR interpreter uses to lookup function names (such as `transpose`), the build system of `SingularInterface` automatically generates bindings for all SINGULAR functions in GAP (`SI_transpose`). Thus when the SINGULAR team adds a new function, `SingularInterface` automatically supports it (after recompiling). This also covers SINGULAR library functions. Finally, in addition to interpreter and library functions, we also provide direct access to select SINGULAR C++ kernel functions such as `p_Mult_mm` (with `_SI_` prepended) which can be used by experts to further optimize their code. This, too, is automatically generated and can thus easily be extended to expose more functionality, if requested.

Regarding (3), we mostly expose the SINGULAR interface faithfully and with no changes (as opposed to trying to make it "more GAP like", or trying to fix perceived design flaws etc.). We plan to eventually add a high-level layer built atop the existing interface which changes some of this; however, this may well be in a separate package. But providing this raw access has multiple advantages: It allows others to build alternative high-level front ends (as everybody will have a different idea about how to do that), and it also frees us from making complicated decisions on what is "right" and "wrong". Finally, it has the added benefit that the SINGULAR manual can be used as a reference manual for `SingularInterface`. There is one major exception: We try to hide the SINGULAR concept of a "global" or "active" ring from users as much as possible. In SINGULAR, the result of a command like `var(1)` implicitly depends on the "active" ring. Working with multiple rings thus becomes rather tedious. To avoid this, in `SingularInterface`, each GAP wrapper object for "ring dependent" SINGULAR objects (such as polynomials, ideals, modules, but not integers or rings themselves) carries a reference to the ring it belongs to. The user does not need to worry about "active" rings at all. As a side effect, a few SINGULAR functions need to be called with one additional parameter, namely the ring they refer to. For example, `var(1)` becomes `SI_var(r,1)`, where `r` is the (now explicit) SINGULAR ring we refer to.

Note that all of this bypasses the SINGULAR language. However, to guarantee complete access to SINGULAR, one can still send arbitrary commands to the SINGULAR interpreter as a string passed to the function `Singular()`. For details, we refer the reader to the `SingularInterface` manual (which is still work in progress).

Activity

The development of `SingularInterface` started in May 2011. The project was generously supported by the University of St Andrews, the University of Kaiserslautern, and DFG priority program SPP-1489. The C/C++ code was contributed by Max Horn, Frank Lübeck, and Max Neunhöffer. To keep the interface slick and efficient Hans Schönemann and Oleksandr Motsak made several changes and improvements on the side of SINGULAR. The `homalg` project [4] heavily uses SINGULAR through its own `IO`-based interface. It was thus a natural candidate which helped to test and stabilize some parts of `SingularInterface` through its extensive test suite.

Outlook

While `SingularInterface` can already be used (and is used) for research work, there is still quite a lot to be done before we can call it a complete product. We need to add some more functionality, expand the manual, and add many test cases. Maybe the most urgent is to make convenient the construction of all types of rings supported by SINGULAR.[4]

For this, we would appreciate your help. For example, if you experience any issue with `SingularInterface`, please report it using our issue tracker at GitHub[5]. The same holds if you feel that some functionality is missing or not as easy to access as it should be. We cannot guarantee to fulfil every wish, but it helps us to prioritize our efforts.

Beyond this, we also welcome contributions to the code, the manual or the test suite. Ideally in the form of pull requests[6].

A future challenge would be to port `SingularInterface` to HPC-GAP[7] once the future multi-threaded SINGULAR is available.

[4] At the moment, they are accessible via the function `Singular()`.

[5] `https://github.com/gap-system/SingularInterface/issues`

[6] `https://github.com/gap-system/SingularInterface/pulls`

[7] HPC-GAP stands for "High Performance Computing GAP" and adds parallelization support to GAP(cf. article of R. Behrends, p. 27 in this issue of the CAR).

References

[1] M. Barakat, M. Horn, F. Lübeck, O. Motsak, M. Neunhöffer, H. Schönemann, *Singular-Interface – A* GAP *interface to* SINGULAR, (http://gap-system.github.io/SingularInterface/), 2011–2014.

[2] W. Decker, G.-M. Greuel, G. Pfister, and H. Schönemann, SINGULAR *3-1-6 — A computer algebra system for polynomial computations*, (http://www.singular.uni-kl.de), 2013.

[3] The GAP Group, *GAP – Groups, Algorithms, and Programming, Version 4.7.5*, (http://www.gap-system.org), 2014.

[4] The homalg project authors, *The* homalg *project – Algorithmic Homological Algebra*, (http://homalg.math.rwth-aachen.de/), 2003–2014.

Berichte über Arbeitsgruppen

Arbeitsgruppe Algebra und diskrete Mathematik an der Universität Osnabrück

Winfried Bruns (Osnabrück)

Die Arbeitsgruppe Algebra und diskrete Mathematik des Instituts für Mathematik der Universität Osnabrück besteht zurzeit aus vier Professoren (H. Brenner, W. Bruns, T. Römer, H. Spindler), vier Postdocs und sieben Doktoranden. Die Professur von Bruns, der am 30.09.2014 in den Ruhestand tritt, wird ab April 2015 durch eine Juniorprofessur (M. Juhnke-Kubitzke) ersetzt.

Die Arbeitsgruppe war 2009–2012 Bestandteil des universitätseigenen Graduiertenkollegs „Kombinatorische Strukturen in Algebra und Topologie“ und ist seit Oktober 2013 am Osnabrücker DFG-GRK „Kombinatorische Methoden in der Geometrie“ beteiligt. Das seit 1997 bestehende Normaliz-Projekt wird aktuell durch das DFG-SPP „Experimentelle Methoden in Algebra, Geometrie und Zahlentheorie“ gefördert.

Zu den langjährigen Kooperationspartnern gehören Mathematiker in Ann Arbor, Bukarest, Essen, Genua, Hanoi, Lexington, San Francisco, Salt Lake City, Sheffield und Tokio.

Das zentrale Forschungsgebiet der Arbeitsgruppe ist die kommutative Algebra unter Einbeziehung der algebraischen Geometrie, algebraische Aspekte der Kombinatorik und der Darstellungstheorie. Über viele Jahre sind Beiträge zu determinantiellen Ringen und Idealen, affinen Monoiden und Monoid-Algebren, homologischen Invarianten graduierter Ringe, Hilbert-Kunz-Theorie, Tight closure, Grothendieck-Topologien, tropischer Geometrie und Arrangements von Hyperebenen hervorgegangen.

Die gegenwärtig und in den letzten Jahren bearbeiteten Themen sind insbesondere die (Nicht-) Lokalisierung von Tight Closure, Irrationalität von Hilbert-Kunz-Multiplizitäten, arithmetische und geometrische Deformationen von Vektorbündeln, symmetrische Asymptotik von Moduln, Gröbner-Basen und Initial-Ideale bzw. -Algebren determinantieller Ideale bzw. Algebren, Relationen von Minoren, Stanley- und Hilbert-Zerlegungen, algebraische Aspekte von Hyperebenen-Arrangements, generische tropische Varietäten.

Für die Computeralgebra ist das Projekt Normaliz sicherlich am interessantesten. Für ein normales affines Monoid berechnet Normaliz die Hilbert-Basis, die Hilbert-Reihe zu einer $\mathbb{Z}$-Graduierung sowie zusätzliche Daten, die bei der Berechnung von Hilbert-Basen und Hilbert-Reihen notwendig sind oder sich nebenbei ergeben. Dazu gehören insbesondere Triangulierungen und Stanley-Zerlegungen.

Ein normales affines Monoid entsteht als Durchschnitt eines rationalen Kegels mit einem Gitter. Deshalb ist es nichts anderes als die Lösungsmenge eines homogenen Systems linearer diophantischer Ungleichungen, Gleichungen und Kongruenzen. Die Hilbert-Basis ist das eindeutig bestimmte minimale Erzeugendensystem (bezüglich der Addition).

Seit Version 2.11 berechnet Normaliz aber auch Lösungen von entsprechenden inhomogenen Systemen, also Gitterpunkte in rationalen Polyedern. Die Erweiterung NmzIntegrate berechnet Hilbert-Reihen zu polynomialen Gewichtsfunktionen und Integrale von Polynomen über Polytope.

Normaliz ist in C++ geschrieben, nutzt GMP für Großzahl-Arithmetik und OpenMP für Parallelisierung. Es steht als ausführbares Programm für die gängigen Betriebssysteme zur Verfügung und ist über sein Bibliotheks-Interface in CoCoA, polymake, GAP, Regina und als optionales Paket in Sage eingebunden. Außerdem gibt es Interfaces zu Macaulay 2 und Singular.

Promotionen in der Computeralgebra

Patrick Njionou Sadjang: Moments of Classical Orthogonal Polynomials
Betreuer: Wolfram Koepf (Kassel)
Zweitgutachter: Mama Foupouagnigni (Yaounde, Cameroon)
Oktober 2013
http://nbn-resolving.de/urn:nbn:de:hebis:34-2013102244291
Abstract: In this work simple formulas for the moments for all families of classical orthogonal polynomials of the Askey-Wilson scheme (Koekoek, Lesky and Swarttouw: Hypergeometric orthogonal polynomials and their q-analogues, Springer, Berlin, 2010) are developed. Furthermore, the generating functions or exponential generating functions for those moments are given. In order to obtain those moments, we have stated the inversion formulas for all those families, also, we have developed many connection formulas between specific polynomial bases.

Thomas Bächler: Counting Solutions of Algebraic Systems via Triangular Decomposition
Betreuer: Wilhelm Plesken (Aachen)
Zweitgutachter: Eva Zerz (Aachen)
Juli 2014
http://darwin.bth.rwth-aachen.de/opus3/volltexte/2014/5104/
Zusammenfassung: Das Ziel dieser Dissertation ist die algorithmische Analyse von Lösungsmengen polynomieller Gleichungs- und Ungleichungssysteme. Um ein Maß für die Größe einer solchen Menge anzugeben, wird das Zählpolynom definiert, welches als Verallgemeinerung der Kardinalität einer endlichen Menge verwendet wird. Die Berechnung des Zählpolynoms erfordert die Zerlegung der Lösungsmenge in paarweise disjunkte Teilmengen, die durch eine spezielle Art polynomieller Dreieckssysteme, genannt einfache Systeme, gegeben sind. Diese einfachen Systeme haben auch für sich gesehen interessante Eigenschaften, welche ebenfalls studiert werden. Insbesondere kann man sie wegen ihrer Dreiecksstruktur iterativ lösen und so eine klare Beschreibung ihrer Lösungsmenge erhalten. Einfache Systeme und Zählpolynome werden verwendet, um algebraische Varietäten und konstruierbare Mengen zu studieren. Schließlich wird das Zählpolynom benutzt, um die Anzahl der Lösungen bestimmter polynomieller Systeme über endlichen Körpern zu bestimmen.

Dreieckssysteme werden häufig verwendet, um Gleichungssysteme iterativ zu lösen. Da beliebige nicht-lineare Systeme im Allgemeinen nicht in Dreieckssysteme überführt werden können, ist es nötig, die Lösungsmengen in kleinere Teilmengen zu zerlegen, welche durch Dreieckssysteme gegeben sind. Eine solche Zerlegung heißt Dreieckszerlegung.

In den 1930ern hat Joseph Miller Thomas eine Methode zur Zerlegung in einfache Systeme eingeführt. Diese Methode unterscheidet verschiedene Fälle exakt durch die Einführung von Ungleichungen. Dies hat den Nebeneffekt, dass die ursprüngliche Menge in paarweise disjunkte Teilmengen zerlegt wird. Seine Methode ist sehr elementar und benötigt keine fortgeschrittenen Kenntnisse der Theorie von Ringen und Idealen.

Die Thomas-Zerlegung ist die einzige bekannte Methode zur Berechnung des Zählpolynoms der Lösungsmenge eines beliebigen Polynomsystems. Falls eine Menge endlich ist, so entspricht ihr Zählpolynom ihrer Kardinalität. Für eine unendliche Menge kodiert das Zählpolynom Informationen über die Faserungsstruktur der Menge, die durch die Wahl und die Reihenfolge der Unbestimmten festgelegt wird. Es hilft dabei, die Gleichheit von Mengen zu entscheiden und enthält Informationen über die Lösungsmenge, wie zum Beispiel ihre Dimension. In einigen Fällen kann es benutzt werden, um die Anzahl der Lösungen eines polynomiellen Systems über einem endlichen Körper zu zählen.

In dieser Dissertation wird das Zählpolynom axiomatisch definiert und seine wichtigsten Eigenschaften werden diskutiert. Eine Verallgemeinerung des Zählpolynoms wird angegeben, genannt der Zählbaum. Ein neuer Algorithmus zur Berechnung der Thomas-Zerlegung über beliebigen Körpern wird präsentiert. Insbesondere ist dieser Algorithmus der erste, der diese Zerlegung über Körpern positiver Charakteristik berechnen kann. Eine Implementierung dieses Algorithmus im Computeralgebrasystem Maple wird bereitgestellt, welche die Zerlegung über endlichen Körpern und über endlich erzeugten Erweiterungen der rationalen Zahlen durchführen kann.

Das Verhalten von Zählpolynomen und Zählbäumen unter Transformationen des zugrundeliegenden Polynomsystems wird analysiert. Eine Verbindung zwischen einfachen Systemen und Einbettungen affiner oder projektiver Varietäten in den affinen respektive den projektiven Raum wird hergestellt, indem jedem einfachen System ein Ideal zugeordnet wird. Diese Verbindung wird für den Fall normaler torischer Varietäten ausführlich diskutiert. Schließlich werden die Thomas-Zerlegung und Zählpolynome auf das Studium rationaler Abbildungen und auf die Bestimmung der Anzahl von Lösungen bestimmter Gleichungs- und Ungleichungssysteme über endlichen Körpern angewendet.

Daniel Duviol Tcheutia: On Connection, Linearization and Duplication Coefficients of Classical Orthogonal Polynomials
Betreuer: Wolfram Koepf (Kassel)
Zweitgutachter: Mama Fouopouagnini (Yaounde, Cameroon)
Juli 2014
http://nbn-resolving.de/urn:nbn:de:hebis:34-2014071645714
Abstract: In this work, we have mainly achieved the following:

1. we provide a review of the main methods used for the computation of the connection and linearization coefficients between orthogonal polynomials of a continuous variable, moreover using a new approach, the duplication problem of these polynomial families is solved;
2. we review the main methods used for the computation of the connection and linearization coefficients of orthogonal polynomials of a discrete variable, we solve the duplication and linearization problem of all orthogonal polynomials of a discrete variable;
3. we propose a method to generate the connection, linearization and duplication coefficients for q-orthogonal polynomials;
4. we propose a unified method to obtain these coefficients in a generic way for orthogonal polynomials on quadratic and q-quadratic lattices.

Our algorithmic approach to compute linearization, connection and duplication coefficients is based on the one used by Koepf and Schmersau and on the NaViMa algorithm. Our main technique is to use explicit formulas for structural identities of classical orthogonal polynomial systems. We find our results by an application of computer algebra. The major algorithmic tools for our development are Zeilberger's algorithm, q-Zeilberger's algorithm, the Petkovšek-van-Hoeij algorithm, the q-Petkovšek-van-Hoeij algorithm, and Algorithm 2.2, p. 20 of Koepf's book *Hypergeometric Summation* and its q-analogue.

Anen Lakhal: Elimination in Ore Algebras
Betreuer: Wolfram Koepf (Kassel)
Zweitgutachter: Werner Seiler (Kassel)
Juli 2014
http://nbn-resolving.de/urn:nbn:de:hebis:34-2014080845851
Zusammenfassung: Viele spezielle Funktionen sind Lösungen linearer Differenzen- und Differentialgleichungssysteme mit polynomialen Koeffizienten. Für eine gegebene Funktion erzeugen diese Gleichungen, betrachtet als Operatorpolynome, ein Linksideal in einer nicht-kommutativen Algebra, genannt Ore-Algebra. Dieses Ideal zusammen mit endlich vielen Anfangswerten charakterisiert die Funktion eindeutig und auf dieses Objekt sind Gröbner-Basen-Techniken anwendbar. Viele Fragestellungen in Bezug auf spezielle Funktionen, die sich durch obige Ideale beschreiben lassen, können durch Elimination geeigneter nicht-kommutativer Variablen in diesem Ideal gelöst werden.

Diese Dissertation liefert einen Forschungsbeitrag zu folgenden Themen:

1. Es wird ein Überblick über den theoretischen algebraischen Hintergrund gegeben sowie über algorithmische Aspekte verschiedener Methoden, die Gröbner-Eliminationsverfahren in Ore-Algebren verwenden, um Probleme bezüglich spezieller Funktionen zu lösen.
2. Es werden in ausführlicher Weise Algorithmen präsentiert, die auf Gröbner-Eliminationsverfahren basieren und die das „Creative telescoping"-Verfahren für Summen und Integrale spezieller Funktionen durchführen.
3. Alle beschriebenen Algorithmen werden untersucht und anhand erklärender Beispiele in dem Computeralgebrasystem Maple miteinander verglichen. Das Ziel dieser Untersuchung war herauszufinden, inwieweit nicht-kommutative Gröbner-Eliminationsverfahren effizient angewendet werden können, um „Creative telescoping" durchzuführen.

Ulrich Thiel: On restricted rational Cherednik algebras
Betreuer: Gunter Malle (Kaiserslautern)
Zweitgutachter: Raphaël Rouquier (Los Angeles)
Juli 2014
dr.hut-verlag.de/978-3-8439-1674-5.html
Zusammenfassung: The aim of this thesis is to shed some light on the representation theory of restricted rational Cherednik algebras - primarily for exceptional complex reflection groups for which almost no results exist so far. The central topics are specific problems posed by Gordon concerning the structure of the Verma modules and the simple modules of restricted rational Cherednik algebras, and Martino's conjecture which relates Calogero-Moser families defined by the block structure of these algebras with Rouquier families defined by the block structure of Hecke algebras.
We give - for the first time - explicit results about the block structure, the dimensions of the simple modules, the decomposition matrices of the Verma modules, and the structure of the simple modules as graded modules for the underlying reflection group - and thus the answers to Gordon's questions - for generic parameters for around half of the 34 exceptional complex reflection groups. While we can confirm in this way the generic part of Martino's conjecture for all these groups, it turns out as a surprise that the conjecture is wrong for the group G25. This is the first and only counter-example to this conjecture so far, and there is no abstract theoretical explanation for the failure yet.
Due to the exceptional nature of exceptional complex reflection groups the approach taken in this thesis differs from the existing literature in so far as it is computational and experimental. This thesis is the first approach to computations in rational Cherednik algebras and their modules, and even though we make use of the powerful computer algebra system Magma, we still have to introduce several new computational methods along with their implementation to deal with these problems. This includes for example a new Las Vegas algorithm for computing the head of a local module and for computing the decomposition matrix of a constituent-closed family of local modules. These computational aspects culminated in the development of a Cherednik Algebra Magma Package (CHAMP for short) which provides a user-friendly and flexible way to perform basic computation in (restricted) rational Cherednik algebras and Verma modules in arbitrary characteristic (CHAMP is freely available at http://thielul.github.io/CHAMP/).
The hope is that the results presented here - along with CHAMP itself - will lead to new theoretical insight, which is hard to gain when the situation for the exceptional groups is not understood.

Berichte von Konferenzen

1. Experimentelle Methoden in der Algebra

Leibniz University Hannover, Germany, 24.03.2014 – 26.03.2014

`www.icm.tu-bs.de/ag_algebra/Workshop-NTH`

Vom 24.-26. März 2014 fand an LU Hannover eine Tagung zum Thema 'Experimentelle Methoden in der Algebra' statt. Organisiert wurde die Tagung von Bettina Eick (Braunschweig) und Anne Frühbis-Krüger (Hannover). Die 11 Hauptvortragenden der Konferenz deckten ein breites Spektrum innerhalb der Computeralgebra ab. So gab es Vorträge zur algorithmischen Gruppentheorie, zur Darstellungstheorie, zur Kohomologietheorie und zur algebraischen Geometrie. Die Konferenz bildete damit ein vielfältiges Programm. Ein Fortführung ist im März 2015 an der TU Braunschweig geplant.

Bettina Eick (Braunschweig)

2. 20th Conference on Applications of Computer Algebra (ACA 2014)

New York, USA, 9.-12. Juli

`faculty.fordham.edu/rlewis/aca2014/`

Vom 9. bis zum 12. Juli fand die 20. *Conference on Applications of Computer Algebra* in New York statt. Sie wurde organisiert von Robert H. Lewis (General Chair), Tony Shaska, Ilias Kotsireas (Program Chairs) und Eugenio Roanes-Lozano, Stanly Steinberg, Michael Wester, José Luis Galán García (Advisory Committee).

Das Treffen fand in entspannter Atmosphäre am Bronx Campus der Fordham University statt, wo auch die meisten Teilnehmer untergebracht waren.

Eröffnet wurde die zwanzigste Auflage der ACA-Konferenzserie mit einem Plenarvortrag von Wen-shin Lee (University of Antwerp) zum Thema "From Computer Algebra to Signal Processing". Danach verteilte sich das Geschehen auf drei bis vier parallel laufende Special Sessions. Stark vertreten war diesmal die algorithmische Behandlung von Differential- und Differenzengleichungen mit den Special Sessions *AADIOS: Algebraic and Algorithmic Aspects of Differential and Integral Operators* und *Computational Differential and Difference Algebra*. Die weiteren der insgesamt elf Special Sessions waren:

- Computer Algebra Aspects of Finite Rings and Their Applications
- Computer Algebra in Coding Theory and Cryptography
- Integration: Implementation and Applications
- Computer Algebra in Education
- Algorithms and Applications in the Geometry of Algebraic Curves and Surfaces
- Innovative Applications and Emerging Challenges
- Arithmetic Geometry
- Group Actions in Algebra and Geometry
- Computer Algebra in Algebraic Topology and its Applications
- Gröbner Bases and Applications.

Beim Bankett am Freitag Abend gab Michael Webster, ein Mitbegründer der traditionsreichen ACA-Konferenzserie, humorvolle persönliche Tipps zur Konferenzorganisation und unterhielt die Gäste mit Anekdoten aus zwanzig Jahren ACA.

Die nächste ACA-Konferenz wird vom 20. bis 23. Juli 2015 in Kalamata, Griechenland stattfinden (`http://www.singacom.uva.es/ACA2015`). Vorschläge für Special Sessions können bis zum 16. Januar 2015 eingereicht werden.

Michael Wibmer (Aachen)

3. 39th International Symposium on Symbolic and Algebraic Computation (ISSAC 2014)

Kobe, Japan, 23.7.2014 – 25.7.2014

`www.issac-symposium.org/2014`

Vom 23. bis 25. Juli dieses Jahres fand in Kobe die jährliche ISSAC-Tagung statt. Die organisatorische Leitung teilten sich Kosaku Nagasaka (Kobe University) und Franz Winkler (Johannes Kepler Universität, Linz). Den Posten des Program Committee Chair hatte Agnes Szanto (North Carolina State University, Raleigh) inne. Unterstützt wurde die Konferenz von einer Reihe von Sponsoren. Darunter befanden sich Wolfram und Maplesoft, die vor dem Konferenzbeginn Workshops rund um ihre Produkte anboten (teilweise nur auf japanisch).

Ebenfalls vor offiziellem Konferenzbeginn fand ein Tutorientag statt. Besonders hervorzuheben ist hier der Vortrag von François Le Gall (University of Tokyo) mit dem Titel "Algebraic Complexity Theory and Matrix Multiplication". Der Vortragende veranschaulichte dem Publikum, wie er es schaffte, die obere Schranke des Exponenten ω bei der Matrix-Multiplikation von $\omega < 2.3729$ auf $\omega < 2.3728639$ zu reduzieren. Die dazugehörige Publikation ist auch im Tagungsbericht der ISSAC 2014 erschienen. Neben diesem Highlight wurden den Konferenzteilnehmern u.a. auch Methoden präsentiert, wie man Computeralgebrasysteme auf mobilen Plattformen entwickeln kann ("How to develop a mobile computer algebra system" von Mitsushi Fujimoto, Fukuoka University of Education).

Die folgenden Tage waren mit einem abwechslungsreichen Programm gefüllt. Nachdem man morgens auf dem Weg zur Universität einen Teil des Berges Rokko bestiegen hatte, begann der Tag mit einem exzellenten Vortrag eines eingeladenen Wissenschaftlers: Norioki Arai (National Institute of Informatics) referierte in humorvoller Weise über die automatisierte Lösbarbeit von standardisierten Mathematikaufgaben, David Stoutemyer (University of Hawaii) hielt einen provokativen Vortrag über Computeralgebra mit Gleitkommazahlen und Bernd Sturmfels berichtete über sogenannte Maximum-Likelihood-Verfahren für Matrizen mit Einschränkungen bezüglich dem Rang. Es folgten mehr als 50 interessante Vorträge mit einer vielseitigen Auswahl an abgedeckten Forschungsbereichen; von schneller Multiplikation über Lineare Algebra bis hin zu Differential- und Differenzengleichungen.

Besonders zu loben ist auch das gewählte Konzept des Konferenzdinners, das in einem Restaurant in der Nähe des Hafens stattfand. Dort gab es viele Stehtische, aber nur eine

begrenzte Anzahl an Sitzplätzen. Dadurch wurde die Möglichkeit geschaffen, eine Vielzahl an Teilnehmern kennenzulernen und mit ihnen ins Gespräch zu kommen. Das Buffet war facettenreich, und die Preisverleihungen trugen zur allgemeinen guten Stimmung bei.

Beim Business Meeting wurden folgende Beschlüsse gefasst: ISSAC 2016 findet an der Wilfried Laurier University in Waterloo (Kanada) statt. Arne Storjohann (University of Waterloo) wird Mitglied im Steering Committee. Des Weiteren wurde der Beschluss von Wolfram verkündet, in Zukunft nicht mehr als Sponsor der ISSAC-Konferenz aufzutreten.

Die nächste ISSAC-Tagung findet an der University of Bath in England statt. Die Einladung, wissenschaftliche Arbeiten dafür einzureichen, ist bereits versendet.

Albert Heinle (Waterloo)

4. International Congress of Mathematical Software 2014

Hanyang University, Seoul, Korea, 05.08.2014 – 09.08.2014

`voronoi.hanyang.ac.kr/icms2014/`

om 5.-9. August 2014 fand die ICMS (International Congress of Mathematical Software) in Seoul (Korea) als Satellitentagung zum ICM Congress statt. Hauptorganisatoren der Tagung waren Hoon Hong, Chee Yap und Deok-Soo Kim. Auf der Tagung wurden 5 Invited Lectures von Jonathan Borwein, Bruno Buchberger, Andrew Sommese, Kokichi Sugihara und Lloyd N. Trefethen vorgestellt. Dazu gab es 17 Special Sessions aus diversen Gebieten der Mathematik, zum Beispiel zu den Themen Computational Group Theory, Coding Theory, Computational Topology, Geometry, Curves and Surfaces oder Groebner Bases. Insgesamt war es eine sehr interessante, breit angelegte Tagung, in der auch viele Gebiete der Computeralgebra angesprochen wurden. Die Tagung wurde auch wegen der schönen und interessanten Umgebung in Seoul von den Teilnehmern sehr gut aufgenommen.

Bettina Eick (Braunschweig)

5. First GAP Days

Aachen, 25.08.2014 – 29.08.2014

`gapdays2014.coxeter.de`

The first GAP Days took place during the last week of August at the RWTH Aachen University. Besides GAP users, package authors and core developers, also main developers of NORMALIZ, `polymake`, and SINGULAR attended the meeting and made substantial contributions to stabilize and improve the existing GAP interface packages `NormalizInterface`, `PolymakeInterface`, and `SingularInterface`.

`PolymakeInterface` is already distributed with GAP. During the meeting it was added to the `polymake`'s extensive test suite to guarantee future compatibility. A β-version of `NormalizInterface` will soon be available. The first public release of `SingularInterface` is now available at

`http://gap-system.github.io/SingularInterface/`

See also the article

"The GAP package `SingularInterface`"

in this issue of the CAR. The further development of HPC-GAP was another major issue and considerable progress has been achieved. The list of participants on the homepage contains links to the slides of the talks, where also the feedback of the participants on their work during the meeting has been collected.

GAP Days are meetings where developers and users with GAP programming experience are invited to influence the future development of GAP by initiating and contributing to discussions and coding sprints. As enough GAP experts are around for technical support, the meetings also offer good opportunities for people to work on their own packages.

The meetings are also suitable for advertising recent developments in core GAP and packages via short talks. However, the focus of such meetings is on code development, with talks only playing a minor role.

Mohamed Barakat, Max Horn, Frank Lübeck (Kaiserslautern, Gießen, Aachen)

Hinweise auf Konferenzen

1. Seventh de Brún Workshop on Homological Perturbation Theory

NUI Galway, Ireland, 01.12.2014 – 05.12.2014

`hamilton.nuigalway.ie/HPT`

The workshop aims to bring together those interested in homological perturbation theory, its applications, and related topics in geometry, topology and computer science.

2. Polymake workshop at TU Berlin

TU Berlin, 05.12.2014

`www.polymake.org/doku.php/workshop1214`

A workshop for users of polymake and friends of polyhedral geometry in general. The format will be slightly different this time, as we are featuring two plenary lectures by Matthias Beck (SFSU) and David Bremner (U Brunswick). In addition we will provide a number of tutorials on various aspects of polymake. This includes our regular help desk where you can discuss your individual ideas, suggestions, questions, complaints.

3. 12th International Conference on Artificial Intelligence and Symbolic Computation AISC 2014

University of Sevilla, Spain, 11.12.2014 – 13.12.2014

`www.glc.us.es/aisc2014`

As usual the conference welcomes papers with a strong AI component that make use of symbolic computation in a wide sense (not excluding but not restricted to computer algebra). However, AISC 2014 aims at extending the scope of this series of conferences to computing in AI thus covering new areas.

4. **CoGrAl2015 - Computations in Groups and Algebras**

   Friedrich Schiller University Jena, 16.02.2015 – 19.02.2015

   cogral2015.uni-jena.de

   The aim of the workshop is to shed some light on various recent aspects of finite group theory, with a particular view towards algorithms and computations. More specifically, we will focus on the following topics: Cohomology of finite-dimensional algebras, Structure of p-groups and fusion systems, Block theory of finite groups.

5. **ALCOMA15 - Algebraic combinatorics and applications**

   Kloster Banz close to Bamberg, 15.03.2015 – 20.03.2015

   alcoma15.uni-bayreuth.de

   ALCOMA15 is a meeting of COST Action IC1104 [http://www.network-coding.eu/]. The intention is to bring together representatives of research groups that work in particular in the field of constructive theory of designs and codes, or on closely related topics. A special focus is on q-analogues of designs and random network coding. This conference is dedicated to the memory of our friend and colleague Axel Kohnert. The proceedings of the conference will be published as a dedicated volume of the journal Ädvances in Mathematics of Communications"

6. **Experimentelle Methoden in der Algebra und Zahlentheorie**

   Niedersächsische Technische Hochschule, 26.05.2015 – 29.05.2015

   www.icm.tu-bs.de/ag_algebra/Workshop-NTH

   This workshop organized by Bettina Eick (Braunschweig), Anne Frühbis-Krüger (Hannover), and Claus Fieker (Kaiserslautern) is located on the boundary between group theory, ring theory, number theory and algebraic geometry within the framework of computational algebra. Its aim is to bring together researchers from all these areas and to facilitate discussions among them.

7. **MEGA-2015: Effective Methods in Algebraic Geometry**

   University of Trento, Italy, 15.06.2015 – 19.06.2015

   mega2015.science.unitn.it

   MEGA is the acronym for Effective Methods in Algebraic Geometry (and its equivalent in many other languages). This series of biennial international conferences, with a tradition dating back to 1990, is devoted to computational and application aspects of Algebraic Geometry and related topics.
 The conference will comprise invited talks, regular contributed talks, presentations of computations, and a poster session; the latter three are subject to a competitive submission process.

8. **Theory and applications of syzygies**

   Saarbruecken, 01.07.2015 – 03.07.2015

   www.math.uni-sb.de/ag-schreyer/conference

   This is a conference organized by Christian Bopp, Daniel Erman, Michael Hahn, Hannah Markwig, and Fabio Tanturri on the occasion of Frank-Olaf Schreyer's 60th birthday.

9. **ISSAC 2015**

   Bath, England, 7.07.2015 – 9.07.2015

   www.issac-conference.org/2015

   The International Symposium on Symbolic and Algebraic Computation (ISSAC) is the premier conference for research in symbolic computation and computer algebra. ISSAC 2015 will be the 40th meeting in the series, which started in 1966 and has been held annually since 1981. The conference presents a range of invited speakers, tutorials, poster sessions, software demonstrations and vendor exhibits with a centerpiece of contributed research papers.

10. **ACA 2015 - 21th Conference on Applications of Computer Algebra**

    Kalamata, Greece, 20.07.2015 – 23.07.2015

    www.singacom.uva.es/ACA2015

    The ACA meetings are organized as a series of Special Sessions. The Special Sessions proposals for the ACA 2015 Conference should be submitted to the Program Chair of the Conference, Edgar Martinez-Moro (edgar@maf.uva.es). Deadline for Special Sessions proposals: January 16, 2015

11. **CASC 2015 –17th International Workshop on Computer Algebra in Scientific Computing**

    Aachen, 14.09.2015 –18.09.2015

    The methods of Scientific Computing play an important role in the natural sciences and engineering. Significance and impact of computer algebra methods and computer algebra systems for scientific computing has increased considerably over the last decade.
    Nowadays, computer algebra systems such as CoCoA, Macaulay, Magma, Maple, Mathematica, Maxima, Reduce, Singular and others enable their users to exploit their powerful facilities in symbolic manipulation, numerical computation and visualization.
    The ongoing development of computer algebra systems, including their integration and adaptation to modern software environments, puts them to the forefront in scientific computing and enables the practical solution of many complex applied problems in the domains of natural sciences and engineering.
    The topics addressed in the workshop cover all the basic areas of scientific computing as they benefit from the application of computer algebra methods and software.
    The 17th International Workshop on Computer Algebra in Scientific Computing (CASC 2015) will be held in Aachen, Germany, from September 14 to 18, 2015. The Local Arrangement Chairs are Eva Zerz and Viktor Levandovskyy.

12. **Jahrestagung der DMV**

    Hamburg, 21.09.2015 – 25.09.2015

    www.math.uni-hamburg.de/DMV2015

    The 2015 annual meeting of the Deutschen Mathematiker-Vereinigung (DMV) will be hosted by the Department of Mathematics of the University of Hamburg. The organisers collaborate with the Dansk Matematisk Forening during the composition of the scientific programme; Danish-German research collaboration in mathematics is one of the special themes of this meeting.

Antrag auf Mitgliedschaft in der Fachgruppe Computeralgebra der GI in Kooperation mit der DMV und GAMM und auf Bezug des Computeralgebra Rundbriefs

Rückfragen: Telefon: + 49 (0)228-302-151/-149 / Telefax: + 49 (0)228-302-167/
E-Mail: mitgliederservice@gi.de / http://www.gi.de
Bitte zurücksenden an: Prof. Dr. Wolfram Koepf, Institut für Mathematik, Heinrich-Plett-Str. 40, 34132 Kassel

Name:	Vorname:
Akadem. Grad:	Geburtsjahr:
Privatanschrift:	
Straße / Postf.:	PLZ Ort:
Telefon:	Telefax:
Dienstanschrift:	
Firma / Inst.:	Abteilung:
Straße / Postf.:	PLZ Ort:
Telefon:	Telefax:
E-Mail:	
Gewünschte Postanschrift: ☐ Privatanschrift	☐ Dienstanschrift
Gewünschte Regionalgruppenzuordnung (www.gi.de/regionalgruppen/):	
Regionalgruppe:	

☐ Ich bin persönliches Mitglied der GI und beantrage die Mitgliedschaft in der Fachgruppe Computeralgebra sowie den Bezug des Rundbriefs

☐ Ich beantrage assoziierte Mitgliedschaft in der GI und Mitgliedschaft in der Fachgruppe Computeralgebra sowie den Bezug des Rundbriefs

☐ ab 1. Januar. ☐ rückwirkend zum 1. Januar des laufenden Jahres (bis zum 30. September möglich).

Ich ordne mich folgender Jahresbeitragsklasse zu:

- ☐ 7,50 Euro für Mitglieder von ☐ DMV ☐ GI ☐ GAMM Mitgliedsnummer:
- ☐ 7,50 Euro. Ich beantrage gleichzeitig Mitgliedschaft in ☐ DMV ☐ GI ☐ GAMM und bitte um Zusendung der dazu erforderlichen Unterlagen.
- ☐ 9,00 Euro für Nichtmitglieder. Ich bitte um Zusendung von Informationen über ☐ DMV ☐ GI ☐ GAMM

☐ Ich bitte lediglich um Aktualisierung meiner Adressdaten sowie meiner Angaben über die Zusendung von Informationen.

Datennutzung

Meine oben angegebenen personenbezogenen Daten werden im Rahmen meiner Mitgliedschaft soweit gesetzlich erlaubt oder aufgrund meiner Einwilligung durch die GI oder durch Dritte nach Weitergabe durch die GI wie folgt genutzt:

- ☐ für alle GI gesellschaftsinternen Aussendungen
- ☐ sowie weitere von der GI gesondert ausgewählte Informationen mit Bezug zur Informatik, wie u.a. Weiterbildungsangebote (z.B. der DIA), Informatik Veranstaltungen oder Kongresse mit und ohne GI-Beteiligung sowie Publikationen mit Informatik-Bezug.

Soweit Sie uns Ihre E-Mail Adresse angegeben haben, wird die oben angegebene Kommunikation soweit möglich elektronisch ausgeführt.

- ☐ Der Nutzung meiner E-Mail Adresse zu Zwecken, die über die satzungsgemäßen Ziele der GI hinausgehen (wie z.B. Werbung, Markt- und Meinungsforschung) stimme ich zu.

Natürlich können Sie Ihre Zustimmung jederzeit widerrufen oder Ihre E-Mail-Adresse in unserem System löschen lassen. Eine kurze Nachricht an mitgliederservice@gi.de, per Post oder Fax genügt.

Ich nehme zur Kenntnis, dass die Aufnahme in die Fachgruppe Computeralgebra zum 1.1. erfolgt und dass die Mitgliedschaft zum 31.12. mit Frist 30.11. schriftlich gekündigt werden kann.

_____ (Datum)

_____ (Unterschrift)

Fachgruppenleitung Computeralgebra 2014-2017

Sprecher:
Prof. Dr. Florian Heß
Carl-von Ossietzky Universität Oldenburg
Institut für Mathematik, 26111 Oldenburg
0441-798-2906, -3004 (Fax)
florian.hess@uni-oldenburg.de
http://www.staff.uni-oldenburg.de/florian.hess

Fachexperte Redaktion Rundbrief:
Prof. Dr. Michael Cuntz
Institut für Algebra, Zahlentheorie und Diskrete Math.
Leibniz Universität Hannover
Welfengarten 1, 30167 Hannover
0511-762-4252
cuntz@math.uni-hannover.de
http://www.iazd.uni-hannover.de/~cuntz

Fachreferent CA-Systeme und -Bibliotheken:
Prof. Dr. Claus Fieker
Fachbereich Mathematik
Technische Universität Kaiserslautern
Gottlieb-Daimler-Straße, 67663 Kaiserslautern
0631-205-2392, -4427 (Fax)
fieker@mathematik.uni-kl.de
http://www.mathematik.uni-kl.de/~fieker

Fachexperte Physik:
Dr. Thomas Hahn
Max-Planck-Institut für Physik
Föhringer Ring 6, 80805 München
089-32354-300, -304 (Fax)
hahn@feynarts.de
http://wwwth.mppmu.mpg.de/members/hahn

Fachreferent Schwerpunktprogramm 1489:
Prof. Dr. Jürgen Klüners
Mathematisches Institut der Universität Paderborn
Warburger Str. 100, 33098 Paderborn
05251-60-2646, -3516 (Fax)
klueners@math.uni-paderborn.de
http://www2.math.uni-paderborn.de/people/juergen-klueners.html

Fachreferent Themen und Anwendungen:
Prof. Dr. Martin Kreuzer
Fakultät für Informatik und Mathematik
Universität Passau
Innstr. 33, 94030 Passau
0851-509-3120, -3122 (Fax)
martin.kreuzer@uni-passau.de
http://www.fim.uni-passau.de/~kreuzer

Fachreferent Schule und Didaktik:
OStR Jan Hendrik Müller
Rivius-Gymnasium der Stadt Attendorn
Westwall 48, 57439 Attendorn
02722-5953 (Sekretariat)
jan.mueller@math.uni-dortmund.de
www.mathebeimueller.de

Stellvertretender Sprecher:
Prof. Dr. Gregor Kemper
Zentrum Mathematik – M11
Technische Universität München
Boltzmannstr. 3, 85748 Garching
089-289-17454, -17457 (Fax)
kemper@ma.tum.de
http://www-m11.ma.tum.de/~kemper

Fachreferentin Themen und Anwendungen:
Prof. Dr. Bettina Eick
Institut Computational Mathematics
Fachbereich Mathematik und Informatik
Technische Universität Braunschweig
Braunschweig
0531-391-7525, -7414 (Fax)
beick@tu-bs.de
http://www.icm.tu-bs.de/~beick/

Fachreferentin Publikationen und Promotionen:
Prof. Dr. Anne Frühbis-Krüger
Institut für Algebraische Geometrie
Welfengarten 1, 30167 Hannover
0511-762-3592
fruehbis-krueger@math.uni-hannover.de
http://www.iag.uni-hannover.de/~anne

Fachreferentin Computational Engineering, Vertreterin der GAMM:
Dr.-Ing. Sandra Klinge
Lehrstuhl für Mechanik - Materialtheorie
Ruhr-Universität Bochum
Universitätsstr. 150, 44780 Bochum
0234-32-26552, -14154 (Fax)
sandra.klinge@rub.de
www.am.bi.ruhr-uni-bochum.de/Mitarbeiter/Ilic

Vertreter der DMV:
Prof. Dr. Wolfram Koepf
Institut für Mathematik
Universität Kassel
Heinrich-Plett-Str. 40, 34132 Kassel
0561-804-4207, -4646 (Fax)
koepf@mathematik.uni-kassel.de
http://www.mathematik.uni-kassel.de/~koepf

Vertreter der GI:
Prof. Dr. Ernst W. Mayr
Lehrstuhl für Effiziente Algorithmen
Fakultät für Informatik
Technische Universität München
Boltzmannstraße 3, 85748 Garching
089-289-17706, -17707 (Fax)
mayr@in.tum.de
http://www.in.tum.de/~mayr/

Fachreferentin CA an der Hochschule:
Prof. Dr. Eva Zerz
Lehrstuhl D für Mathematik
RWTH Aachen
Pontdriesch 14/16, 52062 Aachen
0241-80-94544, -92108 (Fax)
eva.zerz@math.rwth-aachen.de
http://www.math.rwth-aachen.de/~Eva.Zerz/