Arjeh Cohen
Faculteit Wiskunde en Informatica
Technische Universiteit Eindhoven
Postbus 513
5600 MB Eindhoven
amc@win.tue.nl

Overzichtsartikel

Een korte geschiedenis van de computeralgebra in Nederland

Computeralgebra is het rekenen aan wiskundige expressies volgens algebraïsche regels. Deze tak van sport wordt sinds de jaren zestig beoefend en kent vele toepassingen, variërend van het automatisch primitiveren van reële functies tot het berekenen van het bereik van een robotarm. In dit artikel geeft Arjeh Cohen een persoonlijke impressie van wat zich de afgelopen decennia op het gebied van de computeralgebra in Nederland heeft afgespeeld.

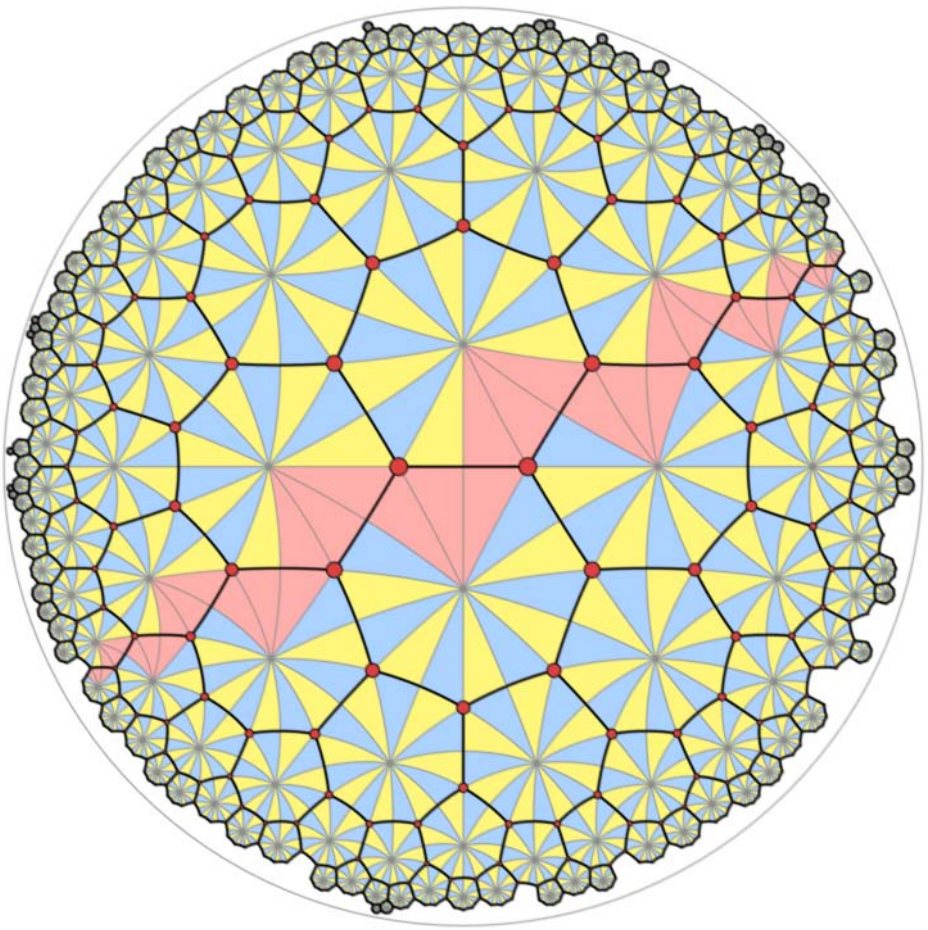
De allereerste computers rekenden tot op zekere hoogte exact: ze konden vrijwel uitsluitend overweg met gehele getallen die niet boven een door de machine bepaalde grootte uitkwamen. Al gauw werden numerici de grootste gebruikers en was het zwevendekomma-getal (floating-point number) het object waarmee het meest gerekend werd. Pas in de jaren zestig, toen er voldoende vertrouwen ontstond in de groei van geheugens en de kracht van computers, kwam de exactheid weer terug. Om te beginnen kwamen er pakketten uit waarin gehele getallen als objecten behandeld werden zoals wiskundigen dat gewend zijn: ofwel het geheugen van de machine liep over, ofwel het getal werd fatsoenlijk behandeld, zonder afkappingen en zonder modulair rekenwerk. Het werd mogelijk om te rekenen met gehele getallen zoals wij dat zelf ook zouden doen als we onvermoeid konden blijven werken. Dan moest het ook mogelijk zijn om andere wiskundige uitdrukkingen als veeltermen, integralen, differentiaalvergelijkingen en logische formules te manipuleren volgens bekende algebraïsche regels, zonder toevlucht tot benaderingen. Inderdaad werd ook dit gerealiseerd. Zo is op verschillende plaatsen in de wereld symbolisch rekenen ontstaan.

Veltman en Vermaseren

De behoefte om exact te rekenen was al vroeg gesignaleerd door fysici die allerlei exacte, dat wil zeggen precies in wiskundige formules gevatte, uitdrukkingen wilden manipuleren. Een prominent voorbeeld is de Nederlandse Nobelprijswinnaar Martinus Veltman. Hij had behoefte aan software om met veeltermen en integralen te werken zonder op benaderingen over te gaan. In 1963 begon hij aan zijn symbolische rekenprogramma Schoonschip, vooral gedreven door berekeningen die nodig waren bij een neutrino-experiment. Volgens zijn eigen woorden leidde de benodigde algebra voor de vector-boson-productie tot een frustrerende hoeveelheid rekenwerk die moeilijk foutloos te krijgen was. Zijn programma Schoonschip heeft model gestaan voor Symbolic Manipulation Program, of kortweg SMP. Dat was een computer-algebrasysteem dat rond 1979 voornamelijk ontwikkeld werd door de op Caltech werkende fysicus Stephen Wolfram. Het werd in 1981 commercieel verspreid, en is feitelijk de voorloper van het overbekende systeem Mathematica [21].

Minder bekend is misschien dat Jos Vermaseren (Nikhef, Amsterdam) een direct vervolg op Schoonschip gemaakt heeft: een programma dat uiterst efficiënt met veeltermen en integralen om kan gaan. De motivatie lag voor een groot deel in Feynmans diagramcalculus. Het is bekend dat daarin evaluatie van de som van een stel bepaalde integralen over alle toestanden in een gegeven configuratie tot een zinvol resultaat leidt, terwijl evaluatie van individuele summanden in die som plus of min oneindig kan opleveren. Die som kan echter alleen met grote exacte rekenkracht bepaald worden. Het programma van Jos Vermaseren heet FORM en wordt nog steeds (voornamelijk door fysici) gebruikt om spectaculaire berekeningen uit te voeren. Zie bijvoorbeeld het arti-

Martinus Veltman (l) en Jos Vermaseren



Illustratie: Jack van Wijk

Gebruik van groepentheorische computeralgebra voor het genereren van de gekleurde driehoeken in een uitvouwing van een Riemannoppervlak van geslacht 14. R14.3′ $\{7,3\}$: orde 2184, 156 septagons, 546 takken, 346 knopen

Buchberger algoritme

Het Buchberger algoritme is in de jaren tachtig sterk onder de aandacht gekomen. Dit komt doordat het algoritme voor veel problemen met betrekking tot veeltermen een effectieve oplosmethode levert, en het op het juiste moment ter beschikking kwam voor de computeralgebrapakketten in opkomst.

Veel wiskundige problemen zijn te vertalen in zoektochten naar oplossingen van stelsels veeltermvergelijkingen. Het Buchberger algoritme helpt exacte oplossingen te vinden, onder andere door een systematische eliminatie van variabelen mogelijk te maken.

Hoewel de methode effectief is, loopt het Buchberger algoritme al gauw vast als het aantal variabelen of de graad van de veeltermen groot wordt.

De beste implementatie voor praktische problemen is van Faugère; zie zijn website www-calfor.lip6.fr/~jcf, waar mooie voorbeelden staan.

kel [31], waarin veel complexe invarianten van Liegroepvoorstellingen worden uitgerekend.

Intermezzo: Formele bewijsvoering

Soms wordt symbolisch rekenen op één hoop gegooid met computeralgebra, dan weer wordt computeralgebra gezien als een deel maar niet het totaal van deze activiteiten. Enerzijds is het uitvoeren van operaties op symbolen een definitie van algebra, en ligt het dus voor de hand de wiskundig georiënteerde algoritmische handelingen als algebra te betitelen—zelfs als ze een vak als analyse of statistiek betreffen. Anderzijds wordt een formeel beschreven stelling en/of bewijs niet vaak als computeralgebra gezien, maar wel als invoer voor symbool-manipulatie, wat als synoniem voor symbolisch rekenen gebruikt wordt.

De formele bewijsvoering en automatische verificatie van een formeel bewijs heeft een grootse Nederlandse geschiedenis, die begint met het Automath project van Dick de Bruijn in Eindhoven en doorloopt tot werk van Henk Barendregt en zijn groep in Nijmegen. De geïnteresseerde lezer raadplege het artikel [35] van Freek Wiedijk over dit onderwerp, dat ook wel computerwiskunde gedoopt is en veel interessante software heeft opgeleverd. Maar het is niet het onderwerp van dit artikel, want dat richt zich op de eigenlijke computeralgebra.

De eerste computeralgebra pakketten

Aan het eind van de jaren zestig was er dus software voor exacte berekeningen in de fysica; na Schoonschip kwam al snel het commerciële pakket REDUCE [30]. In de jaren zeventig verschenen er ook exacte pakketten voor zeer specifieke onderwerpen als groepentheorie. Op dat laatste gebied waren er twee concurrenten: GAP uit Aken (van Joachim Neubüser) en Cayley uit Sydney (van John Cannon). Dit soort software maakte het bijvoorbeeld mogelijk om een groep bestaande uit permutaties van een gegeven verzameling (in den beginne altijd weergegeven als $\{1,\ldots,n\}$ voor een niet al te grote n) door te rekenen. De groep zelf wordt hierbij vastgelegd door enkele permutaties die de groep voortbrengen. Voorbeelden van vragen die dan redelijk efficiënt (maar niet polynomiaal) beantwoord kunnen worden zijn: hoeveel elementen heeft de groep? En: geef een stel voortbrengers voor de ondergroep van alle elementen van de gegeven groep die het element 1 behouden. GAP [8] bestaat, groeit en bloeit nog steeds als gratis beschikbare software; Cayley is uitgegroeid tot het zeer krachtige en zeer veel wiskunde omvattende, commerciële pakket MAGMA [19]. We komen hier later op terug.

In de jaren tachtig kwamen algemene softwarepakketten als Maple [20] en Mathematica op de markt. Eerst (1968–1982) was er nog Macsyma, een experiment op MIT, waarvan een nazaat tegenwoordig gratis beschikbaar is onder de naam Maxima [24]. Dit pakket is gebaseerd op de programmeertaal Lisp. Maple en Mathematica daarentegen zijn geschreven in C. Oorspronkelijk was Maple minder commercieel van opzet dan Mathematica. Het kwam voort uit de symbolic computation group in Waterloo die geleid werd door Keith Geddes en Gaston Gonnet.

In Nederland was de informaticus Hans van Hulzen (UT) er vroeg bij. Hij hield zich bezig met code-optimalisatie en was als een van de eerste Nederlanders actief betrokken bij ISSAC, de grote jaarlijkse conferentie op het vakgebied. Hij droeg bij aan het pakket REDUCE, dat destijds redelijk wijd verspreid was, maar in de commerciële strijd met Maple, MATLAB [23] (een minder exact maar veel gebruikt pakket) en Mathematica ten onder is gegaan. REDUCE is tegenwoordig gratis verkrijgbaar.

Specialistische software en algoritmen

In het kielzog van de opkomst van de algemene computeralgebra-programma's zijn in de jaren tachtig en negentig twee exact-wiskundige activiteiten te onderscheiden. Enerzijds de opkomst van meer specialistische software, anderzijds de aandacht voor algoritmen. Om een indruk te krijgen van de hoeveelheid specialistische softwaresystemen, is het goed de Oberwolfachse website [29] te raadplegen.

Een prominent voorbeeld van de aandacht voor wiskundige algoritmen is de vooruitgang in het exact oplossen van stelsels veeltermvergelijkingen. Er ontstonden veel varianten van wat uiteindelijk bekend is geworden als het Buchberger-algoritme (voor de aanmaak van een Gröbner-basis). Een bescheiden Nederlandse bijdrage in dit verband is het pakket voor een niet-commutatief analogon van het Buchberger-algoritme (wezenlijk anders omdat de theorie hier niet garandeert dat het algoritme eindigt), het in GAP geschreven pakket GBNP [9].

Maar ook het primitiveren, het vinden van een gesloten formule voor een onbepaalde integraal, kreeg veel aandacht. Oude ideeën van mensen als Robert Risch werden uit de kast gehaald en bewerkt tot methoden waarmee ofwel een gesloten formule voor een primitieve van een functie gevonden kan worden, ofwel vastgesteld kan worden dat er, binnen een gegeven klasse functies, geen primitieve bestaat. De meeste primitiveeropdrachten op het niveau van de eerste jaren in het universitair onderwijs konden door deze algoritmen aangepakt worden, met als gevolg dat de leerstof anders, meer algoritmisch, onderwezen werd. Primitiveren van de functie f kan gezien worden als het oplossen van de differentiaalvergelijking $y' = f$. Aan de algemene vraag naar het oplossen van differentiaalvergelijkingen, in de zin van het vinden van oplossingen binnen een bepaalde klasse van functies, heeft onder anderen Marius van der Put (Rijksuniversiteit Groningen) met succes gewerkt; zie bijvoorbeeld [27]. Andere bijdragen kwamen uit Nijmegen; hier gaan we later op in.

Algebraïsche getaltheorie, LLL

Hoewel er een duidelijk onderscheid te maken valt tussen de bouw van een computeralgebrapakket en het ontwerp van een algoritme, zagen veel nieuwe algoritmen het daglicht bij de opkomst van meer specialistische software. Het werd in de jaren tachtig ook erg aantrekkelijk om wiskundige software te ontwikkelen. Twee belangrijke beweegredenen waren het uitvoeren van experimenten waar de theorievorming stokt en het vullen van gaten in bewijzen waarvan de basis van de inductie niet met de hand lukt. Naast de groepentheorie liep de getaltheorie voorop. Henri Cohen bracht rond 1985 in Parijs de eerste versie van het inmiddels klassiek geworden pakket PARI [26] op dit gebied uit. Interessant genoeg gebruikte het (onder meer) zwevendekomma-getallen om exacte objecten uit de algebraïsche getaltheorie te berekenen. In Nederland had Hendrik Lenstra toen al contact met Henri Cohen; hij hield zich vooral met de algoritmische vragen bezig. Eén van zijn promovendi, René Schoof, ontwierp in die tijd een beroemd geworden algoritme om het aantal punten van een elliptische kromme over een eindig lichaam uit te rekenen in polynomiale tijd [34]. Sindsdien speelt Nederland (vooral vanuit Leiden, maar ook vanuit Groningen, Nijmegen, en Utrecht) een vooraanstaande rol in de algoritmische algebraïsche getaltheorie en de daarmee verbonden effectieve aspecten van aritmetische algebraïsche meetkunde.

Kostants vermoeden

Complexe Liegroepen zijn boeiende wiskundige objecten. Het zijn zowel complexe variëteiten als groepen. De enkelvoudige Liegroepen (feitelijk de kleinste bouwstenen) zijn ruim een eeuw geleden geclassificeerd. De grootste die niet in een oneindige serie past heeft dimensie 248 en heet E_8 in de door Cartan gegeven codering.

De bepaling van eindige ondergroepen van een complexe Liegroep, die niet in een Lie-ondergroep van positieve dimensie passen, kreeg een grote stimulus toen Kostant het vermoeden uitsprak dat E_8 een bepaalde enkelvoudige eindige ondergroep van orde 113460 (de fractionele lineaire groep over het lichaam van 61 elementen) zou hebben.

Dit vermoeden konden Robert Griess en ik vertalen naar het probleem een oplossing te vinden van een stelsel van veel veeltermvergelijkingen in acht onbekenden over de gehele getallen modulo 1831. Van dit stelsel was één veelterm van graad 2, één van graad 3, één van graad 4, et cetera. De Gröbner basis algoritmen die me destijds ter beschikking stonden, waren niet voldoende krachtig om een oplossing te vinden. Toen ik Jos Vermaseren om een oplossing vroeg, heeft hij modulaire aritmetiek in FORM gezet, en vele gegeneraliseerde resultantenregels uitgevonden en geïmplementeerd om het stelsel op te lossen. Als er niet wat onduidelijkheden (verdachte asymmetrieën) in mijn data hadden gezeten, had hij het karwei waarschijnlijk helemaal geklaard. In die tijd koos ik een andere benadering: vervang 240 monomen in de acht onbekenden door nieuwe onbekenden om zo een stelsel van vele *lineaire* vergelijkingen in 240 onbekenden te verkrijgen. Dit stelsel was makkelijk op te lossen. Door de monomen weer terug te zetten kon ik gauw een oplossing in termen van de 8 onbekenden vinden.

Bert Lisser hielp me met de programmering van het probleem in Maple, en zo is de vondst bekendgemaakt in een artikel van Griess, Lisser en mezelf. Er was enige controverse over de geloofwaardigheid van het antwoord, omdat het immers op computerberekeningen steunde. We verdedigden ons door te wijzen op het feit dat het resultaat onafhankelijk van onze berekeningen te testen was door een overzichtelijk aantal elementaire rekenkundige operaties (maar teveel voor een normaal mens om met de hand uit te voeren). Moreel was ik hier zekerder van dan van vele lange klassieke bewijzen: het kon geen toeval zijn dat het overbepaalde systeem vergelijkingen ineens een oplossing blootgaf (orde in de chaos!) en elke test die ik met mijn resultaten deed, goed uitkwam.

Nadien is het stelsel veeltermvergelijkingen door zowel Singular als Macaulay opgelost. Nog later gaf Serre een bewijs voor dit resultaat dat geen computer gebruikte, maar wel een met veel technisch rekenwerk gevonden 3-dimensionale algebraïsche ondergroep van de algebraïsche groep van type E_8 over het lichaam met 61 elementen. Het vermoeden van Kostant is overigens door deze en volgende berekeningen juist gebleken.



André Heck (l) en Leendert van Gastel

In het begin van de jaren tachtig was computeralgebra ook onder de aandacht van het CWI gekomen. Vooral bij informatica, waar Arjen Lenstra zich bezig hield met algoritmen voor het ontbinden van veeltermen met gehele coëfficiënten in irreducibele factoren. Zijn broer Hendrik en diens regelmatige en graag geziene gast László Lovász bogen zich mede over het complexiteitsprobleem. De uitdaging was te bewijzen dat er een polynomiaal algoritme is. De gebruikelijke methode van factorisatie komt daar dicht bij maar is exponentieel op één onderdeel. Die methode begint met de factorisatie van de veelterm modulo een geschikt priemgetal p en probeert vervolgens het resultaat te liften naar een resultaat over de gehele getallen. Daarbij wordt eerst een oplossing modulo p^2 gezocht, en steeds verder modulo hogere machten van p totdat er zekerheid is dat de gevonden oplossing ook over de gehele getallen werkt. Het eigenaardige aan de complexiteit is dat het theoretisch tijdrovend is uit te zoeken welke irreducibele modulaire factoren fuseren tot een irreducibele factor over de gehele getallen. Zonder verder inzicht zou dit neerkomen op het aflopen van een aantal gevallen dat exponentieel groeit met het aantal irreducibele modulaire factoren. Dat was het grote theoretische obstakel voor een polynomiaal algoritme. De drie vonden een uitweg met behulp van een zeer algemeen toepasbaar algoritme voor roosters. Zo ontstond het waarschijnlijk beroemdste algoritme voor computeralgebra van Nederlandse bodem: het LLL-algoritme [17]. Trouwens, een van de beste praktische methoden om veeltermen te factoriseren [13] is ook van Nederlandse bodem; zie hieronder.

LiE

Naast FORM van Jos Vermaseren ontstond nog een Nederlands pakket, en wel op het CWI: LiE [18], bedoeld voor berekeningen aan matrixvoorstellingen van Liegroepen, zoals tensordecomposities en restricties van voorstellingen tot ondergroepen. Het beantwoordde veel vragen van fysici, die telkens meer expliciete gegevens wilden dan waar wiskundigen om vroegen. Het pakket, dat heden ten dage nog gebruikt wordt, was onder andere behulpzaam in het werk van Nguyen en Van der Put [25]. Een integrale (maar minder snelle) versie van het pakket is opgenomen in MAGMA. Dit systeem bevat trouwens meer specialistische pakketten, zoals KANT [16], voor algebraïsche getaltheorie.

Levelt

In Nijmegen was Ton Levelt zeer actief in de computeralgebra. Enkele algoritmen van zijn hand (in samenwerking met Thom Mulders), zoals een fraaie methode om de rationale Jordan-normaalvorm van een vierkante matrix te vinden, zijn geïmplementeerd in Maple. Nijmegen was ook zeer actief in het Europese project CATHODE ('92–'95), een van de eerste pogingen om differentiaalvergelijkingen met computeralgebra aan te pakken. Twee Nijmeegse studenten hebben het zeer ver gebracht in de algoritmische wiskunde: Harm Derksen (Ann Arbor) en Mark van Hoeij (Tallahassee). Derksen heeft algoritmen ontwikkeld voor algebraïsche groepen, bijvoorbeeld voor het bepalen van de kleinste algebraïsche groep om een stel inverteerbare matrices [6–7]. Van Hoeij heeft zeer krachtige factorisatiealgoritmen geschreven, zowel voor veeltermen als voor differentiaaloperatoren [13–14]. Levelts computeralgebrawerk in Nijmegen is overgenomen door Wieb Bosma, wiens aandacht vooral naar algoritmische algebraïsche getaltheorie en het systeem MAGMA gaat. Daarvoor had hij al enige jaren in Sydney aan de totstandkoming van MAGMA meegewerkt (zie [2]). Ook droeg de in Nijmegen werkzame Bernd Souvignier bij aan MAGMA met algoritmen voor roosters.

Expertisecentrum CAN, RIACA

Eind jaren tachtig zag het CWI, in het bijzonder wetenschappelijk directeur Cor Baaijen, kans om met behulp van een expertisecentrum de introductie van computeralgebra in Nederland te bevorderen. Zo'n expertisecentrum, zo was de gedachte, zou zich zodanig commercieel ontwikkelen dat het zichzelf in leven zou kunnen houden. In Nederland bestond wel een ruime belangstelling voor de nieuwe ontwikkelingen, maar waren weinig computeralgebra-specialisten te vinden. Een vijftal onderzoekers met die inslag vormde het bestuur van de nieuwe Stichting Computer Algebra Nederland, die als voornaamste praktisch doel had het in 1989 opgerichte expertisecentrum CAN te begeleiden. Andere doelen van de stichting en het expertisecentrum waren het snel toegankelijk maken van computeralgebra-software in Nederland en de toepassingen ervan in industrie en onderwijs bevorderen.

De doelstellingen van het expertisecentrum zijn grotendeels bereikt. Een hoogtepunt was het verschijnen van de eerste goede handleiding voor Maple door André Heck, de leider van het expertisecentrum. Een ander hoogtepunt was de serie studies naar het gebruik van computeralgebra voor industrieel onderzoek. De resultaten van het werk SCAFI (Studies in Computer Algebra for Industry) zijn in twee boeken [4–5] verwerkt. Maar ook de levering van software-pakketten als Maple en Mathematica aan universiteiten werd ondergebracht in een contract met SURF.

Het expertisecentrum werd in 1993 uitgebreid met het instituut RIACA (Research Institute for Applications in Computer Algebra) en is in 1996 geïntegreerd in het Amstel Instituut van de Universiteit van Amsterdam. De commerciële poot is op dat moment zelfstandig geworden onder de naam CAN Diensten bv, en de onderzoekspoot RIACA is verhuisd naar de Technische Universiteit Eindhoven. De experts André Heck en Leendert van Gastel van het expertisecentrum CAN hebben zich in het Amstel Instituut gestort op innovatie van onderwijs in exacte wetenschappen. CAN Diensten bv distribueert, naast financiële en statistische software, nog steeds veel computeralgebrasoftware in Nederland.

Het instituut RIACA heeft zich na de verhuizing naar Eindhoven voornamelijk gericht op interactieve wiskundige documenten; hoewel enkele activiteiten voort bleven bestaan, is de naam langzamerhand op de achtergrond geraakt. Er zijn bijdragen geleverd aan OpenMath [28], een manier om wiskunde semantisch veel preciezer en rijker dan in LaTeX weer te geven, zo rijk dat twee systemen via deze taal met elkaar kunnen spreken (dat wil zeggen, op een bepaalde manier elkaars uitvoer als invoer kunnen lezen). OpenMath is op zijn beurt weer grotendeels terecht gekomen in MathML3 [22], de nieuwe XML-standaard voor de representatie van wiskunde op het web. Verder is in Eindhoven vooral gewerkt aan algoritmen voor het exact werken met algebraïsche groepen en de bijbehorende Liealgebra's. De resulterende algoritmen zijn opgenomen in de pakketten GAP en MAGMA. Van de betrokkenen is Willem de Graaf een actief onderzoeker in de computeralgebra geworden (zie [10–11]), is Erik Postma bij MapleSoft, de producent van Maple, gaan werken, en gaat Dan Roozemond binnenkort bij de Magmagroep in Sydney werken.

Primitiveren via OpenMath in Mathematica en Maxima

De volgende code illustreert hoe OpenMath code in een webpagina gebruikt wordt om een primitieve van de functie $x \mapsto x^2 e^x$ te berekenen in Mathematica.

De OpenMath code voor de primitieve wordt vertaald in een commando in Mathematica taal en ingevoerd in Mathematica. De Mathematica respons wordt afgevangen, weer terugvertaald in OpenMath en tenslotte weer in leesbare vorm gepresenteerd. De terugvertaling vanuit Mathematica in OpenMath wordt vergemakkelijkt door gebruik van het Mathematica commando FullForm dat de wiskundige expressieboom uitschrijft.

Mathematica

The input is $\int x^2 \cdot e^x dx$.
The corresponding OpenMath is

```
<OMOBJ xmlns='http://www.openmath.org/OpenMath'
version='2.0' cdbase='http://www.openmath.org/cd'>

<OMA> <OMS cd='calculus1' name='int'/>
<OMBIND> <OMS cd="fns1" name="lambda"/>
<OMBVAR> <OMV name='x'/> </OMBVAR>
<OMA> <OMS cd='arith1' name='times'/>
<OMA> <OMS cd='arith1' name='power'/>
<OMV name='x'/> <OMI>2</OMI> </OMA>
<OMA> <OMS cd='arith1' name='power'/>
<OMS cd='nums1' name='e'/> <OMV name='x'/>
</OMA> </OMA> </OMBIND> </OMA> </OMOBJ>.
```

The corresponding input to Mathematica is

```
Function[List[x],
Evaluate[Integrate[Times[Power[x,2],Power[E,x]],x]]].
```

The response from Mathematica is

```
Function[List[x], Times[Power[E, x],
Plus[2, Times[-2, x], Power[x, 2]]]].
```

The corresponding OpenMath is

```
<OMOBJ> <OMBIND> <OMS name="lambda" cd="fns1"></OMS>
<OMBVAR> <OMV name="x"></OMV> </OMBVAR>
<OMA> <OMS name="times" cd="arith1"></OMS>
<OMA> <OMS name="power" cd="arith1"></OMS>
<OMS name="e" cd="nums1"></OMS>
<OMV name="x"></OMV> </OMA>
<OMA> <OMS name="plus" cd="arith1"></OMS> <OMI>2</OMI>
<OMA> <OMS name="times" cd="arith1"></OMS> <OMI>-2</OMI>
<OMV name="x"></OMV> </OMA>
<OMA><OMS name="power" cd="arith1"></OMS>
<OMV name="x"></OMV> <OMI>2</OMI>
</OMA> </OMA> </OMA> </OMBIND> </OMOBJ>.
```

The result is $x \mapsto e^x \cdot \left(2 + -2 \cdot x + x^2\right)$.

Dezelfde code kan, na vervanging van een enkele verwijzing naar Mathematica door Maxima, gebruikt worden om met Maxima in plaats van Mathematica te rekenen. Maar Maxima vindt de primitieve niet en spuit een kleine variatie op de oorspronkelijke beschrijving terug.

Maxima

The input is $\int x^2 \cdot e^x dx$.
The corresponding OpenMath is

```
<OMOBJ xmlns='http://www.openmath.org/OpenMath'
version='2.0' cdbase='http://www.openmath.org/cd'>

<OMA> <OMS cd='calculus1' name='int'/>
<OMBIND> <OMS cd="fns1" name="lambda"/>
<OMBVAR> <OMV name='x'/> </OMBVAR>
<OMA> <OMS cd='arith1' name='times'/>
<OMA> <OMS cd='arith1' name='power'/>
<OMV name='x'/> <OMI>2</OMI> </OMA>
<OMA> <OMS cd='arith1' name='power'/>
<OMS cd='nums1' name='e'/> <OMV name='x'/>
</OMA> </OMA> </OMBIND> </OMA> </OMOBJ> .
```

The corresponding input to Maxima is

```
block( [temp], temp:integrate( ((x^^2)*(\%e^^x)) , x ),
buildq( [temp:temp], lambda( [x] , ''temp ) ) ).
```

The response from Maxima is

```
((LAMBDA SIMP) ((MLIST) $X) ((%INTEGRATE SIMP) ((MTIMES SIMP)
((MEXPT SIMP) $%E $X) ((MNCEXPT SIMP) $X 2)) $X)).
```

The corresponding OpenMath is

```
<OMOBJ> <OMA> <OMS cd="calculus1" name="int"></OMS>
<OMBIND> <OMS name="lambda" cd="fns1"></OMS>
<OMBVAR> <OMV name="x"></OMV> </OMBVAR>
<OMA> <OMS cd="arith1" name="times"></OMS>
<OMA> <OMS cd="arith1" name="power"></OMS>
<OMS cd="nums1" name="e"></OMS>
<OMV name="x"></OMV> </OMA>
<OMA> <OMS cd="arith1" name="power"></OMS>
<OMV name="x"></OMV> <OMI>2</OMI>
</OMA> </OMA> </OMBIND> </OMA> </OMOBJ>.
```

The result is $\int e^x \cdot x^2 dx$.

Ten slotte

In de laatste jaren komt het einde van de periode in zicht waarin de wet van Moore voor processoren geldt. De snelheid wordt meer en meer gezocht in het combineren van meerdere processoren. In de computeralgebra wordt daarom veel aandacht besteed aan de steeds betere communicatiemogelijkheden tussen softwaresystemen onderling. Een bijkomend voordeel is dat specialistische pakketten veel toegankelijker worden. De Europese projecten Intergeo [15] en SCIEnce [33], waar Nederland bij betrokken is, zijn twee voorbeelden hiervan. SCIEnce maakt het mogelijk om bijvoorbeeld vanuit GAP een groepenberekening in MAGMA te laten verrichten en een Gröbnerbasisberekening in Maple. Een ander voorbeeld van het bijeenbrengen van verschillende kleinere systemen is het van oorsprong Amerikaanse project SAGE [32], dat overigens gebaseerd is op Python, een taal die ooit op het CWI door Guido van Rossum is ontworpen. Ook de interactie tussen de eerder genoemde formele wiskunde-software en de computer-algebrasystemen is begonnen (zie bijvoorbeeld [1, 3]) en heeft een grote toekomst.

Een andere tendens in de computeralgebra, of eigenlijk daarbuiten, is dat algoritmen steeds meer tot de wiskundige discipline gerekend worden waarvoor ze zijn ontwikkeld. De computationele aspecten worden steeds meer door het oorspronkelijke vakgebied omarmd. Zo kan het gebeuren dat de computationele groepentheorie eerder een onderdeel van de groepentheorie dan van de computeralgebra is, en dat algoritmische getaltheorie tot de getaltheorie en niet tot de computeralgebra gerekend wordt. Maar ook disciplines als topologie en polytopentheorie hebben computeralgebra omarmd. Hierdoor beperkt de reikwijdte van het vak computeralgebra zich meer en meer tot softwaresystemen. En daarvan zijn er maar weinig

van Nederlandse oorsprong. Hier staat tegenover dat veel systemen wel Nederlandse bijdragen kennen.

Anders gezegd, de computeralgebra heeft voor een algoritmische inslag van de wiskunde gezorgd. Veel vermoedens kunnen en *moeten* tegenwoordig zelfs getest worden op een groot bereik van de parameters, voordat ze serieus genomen worden. Dit wijst er op dat computeralgebra standaardgereedschap van de wiskundige geworden is. Verder is er meer erkenning gekomen voor de ontwikkelaars van algoritmen, zoals blijkt uit woorden als 'Experimental' of 'Computational' in tijdschrifttitels.

De computeralgebra heeft ook geleid tot grotere aandacht voor effectiviteit, het daadwerkelijk vinden van objecten waarvan weliswaar bekend is dat ze bestaan, maar waarvan niemand er een heeft aangewezen. Als er eenmaal zo'n algoritme is, verschuift de aandacht naar het zoeken van een efficiënte versie. Derksens algoritme voor de bepaling van de kleinste algebraïsche groep om een stel inverteerbare matrices is een voorbeeld van effectiviteit voor een probleem waarvan bekend was dat er een oplossing voor bestond maar waarvoor nog niet eerder een algoritme was gevonden—het algoritme is ook niet efficiënt. Van Hoeij schreef een van de meest efficiënte algoritmen voor veeltermfactorisatie, waarvan de effectiviteit altijd al buiten kijf stond.

De algoritmische inslag klinkt door in het Nederlandse wiskunde-cluster DIAMANT (Discrete, Interactive & Algorithmic Mathematics, Algebra & Number Theory), dat vijf jaar geleden tot stand is gekomen. In deze samenwerking, maar ook in vele andere verbanden, wordt dankbaar gebruik gemaakt van de aan de computeralgebra te danken verworvenheden.

Dankwoord
Met dank aan Wieb Bosma, Ton Levelt en Hans Sterk voor verbeteringen van eerdere versies, en aan Jan Willem Knopper voor de hulp bij aanmaak van de OpenMath voorbeelden.

Referenties

1 H. Barendregt en A.M. Cohen, 'Electronic communication of mathematics and the interaction of computer algebra systems and proof assistants', *J. Symbolic Computation* **32** (2001), pp. 3–22.

2 W. Bosma, J. Cannon, C. Playoust, 'The Magma algebra system. I. The user language', *Journal of Symbolic Computation* **24** (1997), pp. 235–265.

3 O. Caprotti, M. Oostdijk, 'Formal and efficient primality proofs by means of computer algebra oracles', *Journal of Symbolic Computation* **32** (2001), pp. 55–70.

4 A.M. Cohen (ed.), *Computer Algebra in Industry, Problem Solving in Practice*, Wiley, 1993.

5 A.M. Cohen, L. van Gastel, S. Verduyn Lunel (eds.), *Computer Algebra in Industry 2, Problem Solving in Practice,* Wiley, 1995.

6 H. Derksen, E. Jeandel, P. Koiran, 'Quantum automata and algebraic groups', *Journal of Symbolic Computation* **39** (2004), pp. 357–371.

7 H. Derksen, G. Kemper, *Computational Invariant Theory*, Springer, 2002.

8 *GAP, Groups, Algorithms and Programming*, Aachen, St Andrews, available at www-gap.dcs.st-and.ac.uk/gap

9 GBNP, www.mathdox.org.

10 W. de Graaf, *Lie algebras: theory and algorithms*, Elsevier, 2000.

11 W. de Graaf, A. Pavan, 'Constructing arithmetic subgroups of unipotent groups', *J. Algebra* **322** (2009), pp. 3950-3970.

12 A. Heck, *Introduction to Maple*, 1st edition, Springer, 1993; 3rd edition Springer, 2003.

13 M. van Hoeij, 'Factoring polynomials and the knapsack problem', *Journal of Number Theory* **95** (2002), pp. 167–189.

14 M. van Hoeij, 'Factorization of differential operators with rational functions coefficients', *Journal of Symbolic Computation* **24** (1997), pp. 537–561.

15 Intergeo, Interoperable Interactive Geometry for Europe, i2geo.net

16 KANT, www.math.tu-berlin.de/~kant

17 A.K. Lenstra, H.W. Lenstra, L. Lovász, 'Factoring polynomials with rational coefficients', *Mathematische Annalen* **261** (1982), pp. 515–534.

18 M.A.A. van Leeuwen, A.M. Cohen, B. Lisser, **LiE Manual,** manual for the software package LiE for Lie group theoretical computations, CWI/CAN, 1992. Zie www-math.univ-poitiers.fr/~maavl/LiE

19 MAGMA, Computational Algebra System, magma.maths.usyd.edu.au/magma

20 Maple, www.maplesoft.com/products/Maple

21 Mathematica, www.wolfram.com/products/mathematica

22 MathML, Mathematical Markup Language, version 3, www.w3.org/Math

23 MATLAB, www.mathworks.com/products/matlab

24 Maxima, http://maxima.sourceforge.net

25 K.A. Nguyen, M. van der Put, 'Solving linear differential equations', *Pure and Applied Mathematics Quarterly* **6** (2010), pp. 173–208.

26 PARI, pari.math.u-bordeaux

27 M. van der Put, M. Singer, *Galois theory of linear differential equations*, Grundlehren der mathematischen Wissenschatfen **328**, Springer, 2003.

28 OpenMath, An extensible standard for the semantics of mathematical objects, www.openmath.org

29 ORMS, Oberwolfach References on Mathematical Software, orms.mfo.de

30 REDUCE, reduce-algebra.com

31 T. van Ritbergen, B. Schellekens, J. Vermaseren, 'Group theory factors for Feynman diagrams', *Int. J. Mod. Phys.* **A14** (1999), pp. 41–96.

32 SAGE, A Computer System for Algebra and Geometry Experimentation, www.sagemath.org

33 SCIEnce, Symbolic Computation Infrastructure for Europe, www.symbolic-computation.org

34 R. Schoof, 'Elliptic curves over finite fields and the computation of square roots mod p', *Mathematics of Computation* **44** (1985), pp. 482–494.

35 F. Wiedijk, 'Formal proof – getting started', *Notices of the AMS*, **55** (2008), pp. 1408–1414.