Das SCIEnce EU Programm: Symbolic Computation Infrastructure for Europe

Peter Horn
Universität Kassel

Dan Roozemond
Technische Universität Eindhoven

horn@math.uni-kassel.de
d.a.roozemond@tue.nl

Das SCIEnce Projekt bringt die Entwickler der drei Computeralgebrasysteme GAP, KANT und Maple, die Universität Kassel und das RISC in Linz als Experten für Computeralgebra und die Technische Universität Eindhoven als Experten für OpenMath mit Arbeitsgruppen zu Cluster- und Grid-Computing aus St. Andrews, Edinburgh und Timisoara (Rumänien) zusammen. Das erklärte Ziel des auf fünf Jahre ausgelegten Framework 6 EU-Programmes, das noch bis 2011 läuft, ist es, einerseits die Landschaft der europäischen Computeralgebra-Entwickler zusammenzubringen, und andererseits das symbolische Rechnen auf die Anforderungen der nächsten Jahrzehnte mit einem massiven Wachstum der Anzahl der CPU-Kerne vorzubereiten.

Die Lingua Franca der Computeralgebra: OpenMath und SCSCP

Um die verschiedenen Computeralgebrasysteme miteinander zu verbinden, ist es notwendig, eine *gemeinsame* Sprache zu sprechen, wenn man nicht für jede Verbindung einen Spezialfall erzeugen will. Um die mathematische Semantik zu kapseln, war OpenMath [14] die natürliche Wahl, da es ein etablierter Standard ist, der bereits für ähnliche Zwecke eingesetzt wird [3, 4].

OpenMath ist eine sehr flexible Sprache, die nur aus zwölf Sprachelementen besteht (Ganze Zahlen, Fließkommazahlen, Variablen, ...), wobei alle mathematische Semantik in sogenannten Content Dictionaries (CD) definiert ist. So werden etwa in dem CD `arith1` die elementaren arithmetischen Operationen wie `plus`, `minus` etc. definiert, in dem CD `polyd1` wird beschrieben, wie multivariate Polynome dargestellt werden, und das CD `group4` beschäftigt sich mit Cosets und Konjugationsklassen. Die OpenMath-Sprache wurde so gestaltet, dass sie für Computer und Software denkbar effizient zu benutzen ist.

Im aktuellen MathML 3 Standard ist ein OpenMath-Dialekt (Strict Content MathML) für die Codierung von mathematischer Semantik zuständig.

Um die Kommunikation zwischen den Systemen zu vereinheitlichen wurde das Protokoll SCSCP („Symbolic Computation Software Composability Protocol“) [5, 6] entwickelt. Dieses Protokoll dient nicht nur dazu, Funktionen eines Systems einem anderen zur Verfügung zu stellen, sondern ist auch die Grundlage für die Cluster- und Grid-Infrastrukturen, die im SCIEnce Projekt entwickelt werden.

Der große Vorteil gegenüber vorherigen Ansätzen, einzelne Systeme miteinander zu verbinden ist, dass nun ein Standard vorliegt, und jedes System, das diesen Standard implementiert, kann anderen Systemen Dienste anbieten bzw. diese nutzen.

Das Protokoll basiert auf XML, und alle Protokollnachrichten sind OpenMath-Objekte. Die eigentliche Kommunikation geschieht über TCP Sockets, standardmäßig über den von der *Internet Assigned Numbers*

Authority (IANA) für diesen Zweck registrierten Port 26133. Das Protokoll liegt in Version 1.3 vor, und es existieren Implementierungen in GAP [7], KANT [10], Maple [12] und MuPAD [13]. Außerdem wurde SCSCP in TRIP [8], Magma [2] und Macaulay2 [11] implementiert.

Um die Anbindung von weiteren Computeralgebrasystemen sowie von anderer Software an die SCSCP Infrastrukturen zu erleichtern, wurde die Java-Bibliothek `org.symcomp.scscp` [9] als Referenzimplementierung entwickelt, und steht unter der Apache2 Lizenz zur Verfügung.

Beispiel – KANT von MuPAD aus benutzen

Um zu erklären, wie die Systeme interagieren, zeigen wir hier ein kleines Beispiel, in dem wir KANT von MuPAD aus benutzen. Zunächst konstruieren wir Swinnerton-Dyer-Polynome, indem wir für verschiedene Primzahlen $p_1, \ldots, p_n$ definieren:

$$P_n(x) = \prod \left(x \pm \sqrt{p_1} \pm \sqrt{p_2} \ldots \pm \sqrt{p_n}\right).$$

Dies ist ein Polynom vom Grad 2^n, das über $\mathbb{Z}$ irreduzibel ist, aber über jedem endlichen Körper in 2^{n-1} quadratische Faktoren zerfällt. Für Polynomfaktorisierungsalgorithmen sind solche Polynome besonders herausfordernd.

```
>> package("OpenMath"):
>> swindyer := proc(plist) ... :
>> R:=Dom::UnivariatePolynomial(x,Dom::Rational):
>> p1 := R(swindyer([2,3,5,7,11])):
>> p2 := R(subs(swindyer([2,3,5,7,13,17])),
           x=3*x-2):
>> p := p1 * p2:
>> degree(p), nterms(p)
96, 49
```

Damit haben wir ein univariates Polynom p konstruiert, das das Produkt zweier Swinnerton-Dyer-Polynome ist, und 49 Terme und Grad $96 = 2^5 + 2^6$ hat.

```
>> st := time(): F1 := factor(p): time()-st
38431
```

Die Faktorisierung in MuPAD benötigt also 38 Sekunden.

Da KANT einen der effizientesten Faktorisierungsalgorithmen hat, wollen wir nun einen KANT SCSCP Server benutzen, der sich 400km entfernt befindet:

```
>> kant := SCSCP("scscp.math.tu-berlin.de",26133):
>> st:=rtime():
   F2:=kant::compute(hold(factor)(p)):
   rtime()-st
1221
```

Die Faktorisierung in KANT benötigt nur 1.2 Sekunden, und in dieser Zeit ist die Konversion der Daten nach und von OpenMath und der Netzwerk-Transport enthalten.

Nun verifizieren wir noch (etwas umständlich), dass die Faktoren übereinstimmen:

```
>> FS1 := {op(Factored::factors(F1))}:
>> FS2 := {op(Factored::factors(F2))}:
>> bool(FS1=FS2)
TRUE
```

Auch wenn das Beispiel etwas konstruiert ist, wird doch der Benefit des Ansatzes klar: Man kann in „seinem“ gewohnten System arbeiten und Arbeiten, die andere Systeme besser (oder überhaupt) können, transparent deligieren.

Ist das nicht *Sage*?

In der letzten Rundbrief-Ausgabe wurde das CAS *Sage* vorgestellt [1], das aus der Kombination vieler verschiederner Systeme und Bibliotheken hervorgegangen ist. Tatsächlich klingt das recht ähnlich zu dem Vorgehen im SCIEnce-Projekt, lässt sich aber klar abgrenzen.

In *Sage* werden die verschiedenen Systeme einem Python-Interface untergeordnet, so dass dieses eine System die Fähigkeiten der untergeordneten Systeme nutzen kann. Auch laufen all diese Systeme auf der gleichen Maschine. Bei SCSCP kann man wie gewohnt in dem System seiner Wahl arbeiten, und auf Funktionalitäten aus anderen System zugreifen, die auch auf anderen Maschinen laufen können. Außerdem bietet SCSCP die Basis für das Parallelisieren von Berechnungen.

Nichtsdestoweniger ist *Sage* ein interessantes Projekt, und wir würden uns über SCSCP-Unterstützung in *Sage* freuen.

POPCORN – OpenMath in lecker

Im Rahmen der Entwicklung von SCSCP war es häufig notwendig, auch größere Fragmente OpenMath zu schreiben bzw. zu lesen. Trotz XML-Unterstützung in vielen Editoren ist das sehr unbefriedigend, da XML keine übliche mathematische Notation ist. Andererseits existieren in praktisch allen Computeralgebrasystemen *ähnliche* Notationen. Um die Eingabe und das Lesen von OpenMath zu erleichtern, haben wir POPCORN („Possibly Only Practical Convenient OpenMath Replacement Notation“) entwickelt, das wir aus typographischen Gründen einfach „Popcorn“ schreiben. Popcorn ist kein Ersatz für OpenMath, sondern einfach nur eine andere Repräsentation der OpenMath Sprache.

Ganze Zahlen, Fließkommazahlen und Strings werden genau so dargestellt, wie man es erwartet (also etwa `1`, `2.67` oder `''text''`), Variablen wird das Symbol $ vorgestellt, Referenzen das #, und OpenMath-Symbole werden mit einem Punkt zwischen CD-Namen und Symbolnamen geschrieben (etwa `arith1.plus`).

Für zusammengesetzte Elemente gelten ähnlich einfache Regeln: Soll ein Element auf andere angewendet werden, so werden einfach die Argumente in Klammern hinter das Element geschrieben. Für Bindings werden eckige, für Attributions geschwungene Klammern benutzt.

Zusätzlich wurden für viele häufig benutzte Symbole Abkürzungen sowie Infix-Operatoren für die wichtigsten Operationen (`+`, `-`, `and`,...) eingeführt.

Popcorn – Beispiele

Wahrscheinlich sind Beispiele nützlicher als langer Text; hier sind jeweils die Popcorn- und die XML-Schreibweise gegenübergestellt.

Für die Addition zweier Zahlen haben wir in Popcorn: `1+2`

```
<OMA>
  <OMS cd="arith1" name="plus" />
  <OMI>1</OMI>
  <OMI>2</OMI>
</OMA>
```

Die Funktion, die x auf $x + 1$ abbildet schreibt sich in Popcorn: `lambda[$x->1+$x]`

```
<OMBIND>
  <OMS cd="fns1" name="lambda" />
  <OMBVAR>
    <OMV name="x" />
  </OMBVAR>
  <OMA>
    <OMS cd="arith1" name="plus" />
    <OMI>1</OMI>
    <OMV name="x" />
  </OMA>
</OMBIND>
```

Um der Variablen a eine Liste aus $\frac{1}{2}$ und der komplexen Zahl $2 + 8i$ zuzuweisen, schreiben wir in Popcorn `$a := [1//2, (2|8):x]` (die komplexe Zahl bekommt noch die 'id' `x`)

```
<OMA>
  <OMS cd="prog1" name="assign" />
  <OMV name="a" />
  <OMA>
    <OMS cd="list1" name="list" />
    <OMA>
      <OMS cd="nums1" name="rational" />
      <OMI>1</OMI><OMI>2</OMI>
    </OMA>
    <OMA id="x">
      <OMS cd="nums1" name="complex_cartesian" />
      <OMI>2</OMI><OMI>8</OMI>
    </OMA>
  </OMA>
</OMA>
```

Das Integral $\int_0^1 \frac{1}{x^3+\cos(x)} dx$ schreibt sich in Popcorn als `defint(0 .. 1, lambda[$x -> 1/($x^3 + cos($x))])`, in XML wird das

```
<OMA>
 <OMS cd="calculus1" name="defint"/>
 <OMA>
  <OMS cd="interval1" name="interval"/>
  <OMI>0</OMI><OMI>1</OMI>
 </OMA>
 <OMBIND>
  <OMS cd="fns1" name="lambda"/>
  <OMBVAR><OMV name="x"/></OMBVAR>
  <OMA>
   <OMS cd="arith1" name="divide"/>
   <OMI>1</OMI>
   <OMA>
    <OMS cd="arith1" name="plus"/>
    <OMA>
     <OMS cd="arith1" name="power"/>
     <OMV name="x"/>
     <OMI>3</OMI>
    </OMA>
    <OMA>
     <OMS cd="transc1" name="cos"/>
     <OMV name="x"/>
    </OMA>
   </OMA>
  </OMA>
 </OMBIND>
</OMA>
```

Hier wird ein zweiter Grund für den Namen Popcorn sichtbar: Etwas sehr Kleines wird zu etwas sehr Großem aufgebläht.

WUPSI

Um ein Werkzeug zum Testen und Debuggen der verschiedenen OpenMath und SCSCP Dienste zu haben, wurde WUPSI („Wonderful Universal Popcorn SCSCP Interface") entwickelt. Es handelt sich dabei um eine Java Kommandozeilenanwendung, die sich an SCSCP Server verbinden kann und dann eine auf Popcorn basierende Eingabemöglichkeit für Kommandos bietet. Außerdem ist eine elementare Hilfe zu OpenMath in WUPSI eingebaut.

Damit ist WUPSI das „Schweizer Taschenmesser" für OpenMath und SCSCP.

Zur Illustration dient am Besten wieder ein kleines Beispiel:

```
WUPSI 1.x -- Wonderful Universal Popcorn SCSCP
    Interface
(c) 2009 D. Roozemond & P. Horn

WUPSI[n/a]0> connect some.server:26139 as gap
# connected to 'some.server' on port '26139' using
     symbolic name 'gap'
# Service Info: service Name 'GAP', service
    version '4.dev'

WUPSI[gap]0> 126+2323*232
539062

WUPSI[gap]1> local \$a := \$_out0
# Stored this in local variable '\$a':
539062

WUPSI[gap]2> connect 127.0.0.1:26134 as mupad
# connected to '127.0.0.1' on port '26134' using
    symbolic name 'mupad'
```

```
# Service Info: service Name 'MuPAD', service
    version '0.6.0-mupad-5.2.0'

WUPSI[mupad]2> output format latex
# switched output format to LATEX.

WUPSI[mupad]2> sum(1 .. infinity, lambda[$x -> 1/
    $x^2])
{\pi}^{2} \cdot \frac{1}{6}

WUPSI[mupad]3> output format popcorn
# switched output format to POPCORN.

WUPSI[mupad]3> local \$p := 2^127-1
# Stored this in local variable '\$p':
170141183460469231731687303715884105727

WUPSI[magma]4> use gap
# switched to system with symbolic name 'gap',
    service Name 'GAP', service version '4.dev'.

WUPSI[gap]4> \$p-2^101*\$a
168774498924748772136428072069291311103

WUPSI[gap]4> describe arith1.plus
# -- Description for 'arith1.plus' --
The symbol representing an n-ary commutative
    function plus.
# -- END description for 'arith1.plus' --
```

WUPSI bietet über diese Fähigkeiten hinaus noch viele andere Möglichkeiten, mit SCSCP Servern und Clients zu interagieren, etwa um Berechnungen automatisch auf verschiedene Systeme zu parallelisieren. Nicht zuletzt ist es ein hervorragendes Beispiel, wie die Java-Bibliotheken `org.symcomp.openmath` und `org.symcomp.scscp` benutzt werden können.

Zusammenfassung

Wir haben versucht, einen Überblick über die Aktivitäten im europäischen Projekt „SCIEnce“ zu geben, insbesondere zum auf OpenMath basierenden SCSCP Protokoll, das derzeit von GAP, KANT, Maple, Macaulay2, Magma, MuPAD und TRIP unterstützt wird.

Wir hoffen und erwarten, dass in den nächsten Jahren viele weitere Systeme den Standard implementieren werden, so dass die Computeralgebra-Welt hierdurch weiter zusammenwachsen kann und auch die im Projekt entwickelten Infrastukturen für Cluster und Grids für diese Systeme benutzbar werden.

Lizenzen und Verfügbarkeit

Die `org.symcomp.openmath` und `org.symcomp.scscp` Bibliotheken, das MuPAD SCSCP Package [13] und WUPSI sind unter der Apache 2 Lizenz frei verfügbar. Bei GAP, KANT und Maple ist oder wird die SCSCP Unterstützung Teil der Distributon.

Literatur

[1] M. Albrecht, H. Schilly, *Sage*, Rundbrief Computeralgebra 44, 2009.

[2] W. Bosma, J. J. Cannon (Eds), *Handbook of Magma Function.* Edition 2.15, School of Mathematics and Statistics, University of Sydney, 2008. `http://magma.maths.usyd.edu.au/`

[3] O. Caprotti, A. M. Cohen, *Connecting proof checkers and computer algebra using OpenMath.* In: The 12th International Conference on Theorem Proving in Higher Order Logic, Nice, France, September 1999.

[4] O. Caprotti, A. M. Cohen, M. Riem, *Java Phrasebooks for Computer Algebra and Automated Deduction.* SIGSAM Bulletin, 2000. Special Issue on OpenMath.

[5] S. Freundt, P. Horn, A. Konovalov, S. Linton, D. Roozemond, *Symbolic Computation Software Composability.* In: Intelligent Computer Mathematics, AISC/Calculemus/MKM 2008 Proceedings, Lecture Notes in Computer Science 5144, 2008, Springer, 285-295.

[6] S. Freundt, P. Horn, A. Konovalov, S. Linton, D. Roozemond, *Symbolic Computation Software Composability Protocol (SCSCP) specification.* Version 1.3, 2009. `http://www.symbolic-computation.org/scscp/`

[7] The GAP Group, *GAP – Groups, Algorithms, and Programming.* Version 4.4.12, 2008. `http://www.gap-system.org`

[8] M. Gastineau, *SCSCP C Library – A C/C++ library for Symbolic Computation Software Composibility Protocol.* IMCCE, 2009. `http://www.imcce.fr/Equipes/ASD/trip/scscp/`

[9] *Java libraries `org.symcomp.openmath`, and `org.symcomp.scscp`.* `http://java.symcomp.org/`

[10] *KANT/KASH*: `http://www.math.tu-berlin.de/~kant/kash.html`

[11] *Macaulay2: A software system for research in algebraic geometry*: `http://www.math.uiuc.edu/Macaulay2/`

[12] *Maple 13*: `http://www.maplesoft.com/`

[13] *MuPAD OpenMath Package*: `http://mupad.symcomp.org/`

[14] *OpenMath*: `http://www.openmath.org/`