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

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

graebe@informatik.uni-leipzig.de
nareike@informatik.uni-leipzig.de

Introduction

In a more and more networked and interlinked world both the prospects and importance of a well desgined and powerful digital research *infrastructure* dramatically increase. Well acknowledged efforts in that direction are met for years within the organizational structures of the scholarly process – digital support for dissemination of new papers, refereeing process, conference submissions, scientific communication within communities. MathSciNet, ArXiv.org, EasyChair.org, distinguished bibliographical services as, e.g., bibsonomy.org or *The DBLP Computer Science Bibliography* [3] witness for these efforts. Within the *GND Project* (Gemeinsame Norm-Datei) [12] the network of German Scientific Libraries started a strong initiative to unite and build up an integrated digital information system that enhances the classical catalogue system of meta information about scholarly work and makes it ready for the digital age.

Smaller academic communities, as, e.g., the Computer Algebra (CA) community, are challenged in the same way to reorganize also their *intracommunity* communication networks and infrastructure. Nevertheless its hard to allocate resources for such a reorganizational process since infrastructural efforts – even the ongoing ones as management of research databases, preparation of new releases of well established CA Systems etc. – are rarely acknowledged by the reputational processes of science and hence are left to the casual engagement of volunteers if not supported by the leading edge scientists of the community. Such a misrecognition of important efforts of community building is addressed for years in the CA community.

Open Source culture offers plenty of experience how to substitute centrally staffed organizational structures that do great work in the large but cannot be subsidised by smaller scientific communities by decentralized networked structures. Sagemath [23] is one of the big experiences in that direction gained by the CA community. Sagemath is challenging not only from a CA perspective but also from the perspective of software engineering since it provides a mature showcase for a concept called "software as a process". Nevertheless it is directed primarily towards construction of a technical artefact and only in a second line of social relations.

I think it's time to apply that experience also more directly to construct social relations and to build up a Computer Algebra Social Network. The new focus of SYMBOLICDATA version 3 as intercommunity project, providing not only reliable access to data for testing and benchmarking purposes but also technical support for interlinking between different CA subcommunities is another step in that direction.

The SYMBOLICDATA Project at Large

SYMBOLICDATA is part of CA infrastructural efforts for almost 15 years. It grew up from the Special Session on Benchmarking at the 1998 ISSAC conference, where the participants were faced with a typical situation: Within the EU-supported PoSSo [21] and FRISCO [9] projects volunteers compiled a large database of Polynomial Systems with the goal to make it publicly available for testing and benchmarking of algorithms. After the project funding finished people switched to other tasks and it became more and more problematic to access the data. Moreover, badly cloned copies of the data went across the globe and after a while it was even hard to decide, what, e.g., "Katsura-5" means – is it about the example from the well known series with 5 variables $y_1, \ldots, y_5$ or with 6 variables $x_0, \ldots, x_5$?

The SYMBOLICDATA Project started in 1999 on that basis to build a reliable and sustainably available reference of Polynomial Systems data, to extend and update it, to collect meta information about the records, and also to develop tools to manage the data and to set up and run testing and benchmarking computations on the data. The main design decisions and implementations of the first prototype were realized by Olaf Bachmann and Hans-Gert Gräbe in 1999 and 2000. We collected data

from *Polynomial Systems Solving* and *Geometry Theorem Proving*, set up a CVS repository, and started test computations with the main focus on Polynomial Systems Solving. The prototype was presented at the Meeting of the German Fachgruppe Computeralgebra, Kaiserslautern, February 2000.

The project resources drastically shrinked when Olaf Bachmann left the project for a new job at the end of 2000. The project was presented within talks at RWCA-02, ADG-02 and also in this CA-Rundbrief, but there was almost no advance of the project during 2002–2005. In a second phase around 2006 the project matured again. Data were supplied by the CoCoA group (F. Cioffi), the Singular group (M. Dengel, M. Brickenstein, S. Steidel, M. Wenk), V. Levandovskyy (non commutative polynomial systems, G-Algebras) and Raymond Hemmecke (Test sets from Integer Programming). In 2005 the Web site `http://www.symbolicdata.org` sponsored by the German Fachgruppe Computeralgebra went online. During the Special Semester on Groebner Bases in March 2006 we tried to join forces with the GB-Bibliography project (Bruno Buchberger, Alexander Zapletal) and the GB-Facilities project (Viktor Levandovskyy). Unfortunately, all that turned out to be another flash in the pan.

A third phase started in 2009 where the project joined forces with the Agile Knowledge Engineering and Semantic Web (AKSW) Group at Leipzig University [1] to strongly refactor the data along standard Semantic Web concepts based on the Resource Description Framework (RDF). In 2012 these efforts were supported by a 12 months grant *Benchmarking in Symbolic Computations and Web 3.0* for Andreas Nareike within the *Saxonian E-Science Initiative* [6].

Within that scope we completed a redesign of the data distinguishing more consequently between data (*resources* in the RDF terminology) and meta data (*knowledge bases* in the RDF terminology) and refactoring the meta data along the Linked Data principles. The new SYMBOLICDATA data and tools were released as version 3 in September 2013.

Resources (examples for testing, profiling and benchmarking software and algorithms from different CA areas) are publicly available in XML markup, meta data in RDF notation both from a public git repo, hosted at `github.org`, and from an OntoWiki [17] based data store at `http://symbolicdata.org/Data`. Moreover, we offer a SPARQL endpoint [26] to explore the data by standard Linked Data methods.

The website operates on a standard installation using an Apache web server to deliver the data, the Virtuoso RDF data store [30] as data backend, a SPARQL endpoint and (optionally) OntoWiki to explore, display and edit the data. This standard installation can easily be rolled out at a local site (tested under Linux Debian and Ubuntu 12.04 LTS; a more detailed description can be found in the SYMBOLICDATA wiki [25]) to support local testing, profiling and benchmarking.

The distribution offers also tools for integration with a local compute environment as, e.g., provided by Sagemath [23] – the Python based *SDEval package* [13] by Albert Heinle offers a JUnit like framework to set up, run, log, monitor and interrupt testing and benchmarking computations, and the *sdsage package* [16] by Andreas Nareike provides a showcase for SYMBOLICDATA integration with the Sagemath compute environment.

SYMBOLICDATA Concepts on the Way

In the first prototype finished in 2000 the data was stored in a flat XML-like syntax and managed with elaborated Perl tools. Meta information about data was stored in the same format in a special META directory and allowed for a unique and flexible handling of all the data. We used a first level XML like markup for the different parts of a data record as in a classical database design to ensure that data can be represented as hashes within our Perl tools.

We were faced with the same questions about the representation of polynomials as the people in the PoSSo context: Use the MathML or OpenMath deep XML structures to represent polynomials or store them in the well established compact operator syntax that can be directly parsed by most of the tools used in the Polynomial Systems Solving community? PoSSo decided for the latter – mainly due to bandwidth restrictions that were a major design issue at those times – and we did so.

Later on it turned out that it was a good design decision not to overestimate XML based syntax forms also from another point of view: If we shifted SYMBOLICDATA to an intercommunity project in 2013 it was a strong acceptance problem to store intracommunity data (e.g., Fano Polytopes) in a (non-XML) format that is well established, used and accepted within the particular CA subcommunity.

Since the XML world just did start to mature when the first prototype was designed we developed special SYMBOLICDATA tools to manage our data format. In the second phase of the project during 2002–2008 we moved to a true XML syntax and substituted the special SYMBOLICDATA tools by standard XML tools. We had not enough development power fully to migrate the testing and benchmarking environment to the new format and adapted our philosophy concerning that topic – not to provide an elaborated fully fledged environment for testing and benchmarking but concentrate on publicly collecting best practise usage examples. Perl based code snippets for Geometry Theorem Proving computations were supplied by Hans-Gert Gräbe, python based snippets for computations within Polynomial Systems Solving by Michael Brickenstein and for Free Algebra computations by Viktor Levandovskyy.

The SYMBOLICDATA Development Process

A major redesign was triggered with support by the E-Science Saxony Project [6] in 2012/13. First, we moved the data from former CVS and Mercurial repositories to git and started hosting the main SYMBOLICDATA

Project resources at `github.com`. We established a development process along the Integration-Manager-Workflow Model with Integration Manager Ralf Hemmecke (Uni Linz). This makes it easy to join forces with the SYMBOLICDATA team: Fork the repo to your github account, start development and send a pull request to the Integration Manager if you think you produced something worth to be integrated into the upstream master branch. Even if your contribution is not pulled to the upstream, people can use it, since they can pull it from your to their github repos. This allows even for agile common small feature development – a widely practised way to advance projects hosted at `github.com`. You are encouraged early to start a discussion of your plans and regularly report your progress on the SYMBOLICDATA mailing list.

SYMBOLICDATA Resources and Maintenance

Second, within the E-Science Project we decided to strengthen the SYMBOLICDATA part that is *not* involved with Polynomial Systems Solving. These efforts led to a more consequent distinction between data (owned and maintained by different CA subcommunities) and meta data. Such a distinction is well supported by RDF design principles – the Resource Description Framework is about *description* of *resources*, represented by (globally unique) *resource identifiers* (URIs). It is best Linked Data practise to provide URIs in such a way, that they are accessible by the HTTP internet protocol and a valuable part of structured information about that resource is delivered upon HTTP request to that URI.

Note that the distinction between *resources* (managed by CA subcommunities using intracommunity standard methods of representation and access) and *resource descriptions* (important for technically supported interchange between such subcommunities and for intercommunity communication) leads away from XML based design principles that mainly focus on the distinction between information (XML records) and information structure (described with XSchema). Hence, on the way to SYMBOLICDATA version 3 we had to redesign the data once more, to distinguish between resources and meta data more consistently, and to transform these different data according to different principles.

Currently the SYMBOLICDATA data collection contains resources from the areas of Polynomial Systems Solving (390 records, 633 configurations), Free Algebras (83 records), G-Algebras (8 records), GeoProofSchemes (297 records) and Test Sets from Integer Programming (28 records). These resources are stored in a flat XSchema based XML syntax using well established intracommunity syntaxes for the internal data.

Note that such a concept is not restricted to resources centrally managed at `symbolicdata.org` but can easily be extended to other data stores on the web that are operated by different CA subcommunities and offer a minimum of Linked Data facilities. There are draft versions of resource descriptions about Fano Polytopes (8630 records) [20] and Birkhoff Polytopes (5399 records) [19] from the polymake project [10] hosted by Andreas Paffenholz and about Transitive Groups (3605 records) from the Database for Number Fields [14] of Jürgen Klüners and Gunter Malle that point to external resources. This part of SYMBOLICDATA requires further solicitation.

With the reorientation towards an intercommunity project SYMBOLICDATA version 3 proposes also a new maintenance concept that emphazises that SYMBOLICDATA can only be successful as a *joined* effort of the different CA subcommunities. We understand that efforts in the first plan require human resources to maintain and manage the resources from the different subcommunities. The SYMBOLICDATA Project went through several 'dry periods' providing the collected information 'as is' in a read only way as digitally accessible reference. For a 'less dry' future it would be helpful to have *coaches* for the different topics on a more regular base that are part of a CA subcommunity and altogether constitute the *advisory council* of the SYMBOLICDATA Project.

RDF Basics

One of the main advances with version 3 of SYMBOLICDATA is the overall introduction of Linked Data concepts and RDF notations for the meta data information. Since it is crucial for the understanding of the new concepts we give a short introduction to RDF basics. For more details we refer to, e.g., [28].

RDF is an acronym of "Resource Description Framework" and it is above all a data model. Its basic idea is to store pieces of information as *triples*. Each such triple can be considered as a *sentence* of a story that consists of a subject s, a predicate p and an object o. A common way to denote such triples is a whitespace separated juxtaposition with a final period:

s p o .

Subjects and predicates have to be URIs (Uniform Resource Identifiers) while objects (or 'values') can either be an URI or a literal in lexical form (a string included in quotes) that can either be plain or typed. There are some common types (e.g., 'xsd:integer') but custom types can be defined as well.

A set of triples can be interpreted as a directed *RDF graph* with subjects and objects as nodes (replacing literals by labelled blank nodes) and predicates as labelled edges between nodes. On the opposite, a directed graph can be written as a set of triples (and is commonly represented in such a way as internal data structure of graph programs). Another representation uses sets of key-value pairs $p \to o$ assigned to the different subjects s. Note that, different to database columns, a key p can have multiple values.

RDF obtains special expressive power from the fact that sets of subjects, objects and predicates are not necessarily (mutually) disjoint. Hence RDF allows to express information about models and metamodels in the same language and thus in a unified way.

There are different shortcut notations of RDF as, e.g., RDF-XML, JSON, or Turtle, and plenty of tools and parsers for the different formats. More shortcuts are introduced by the concept of namespace prefixes that provide a world wide unique naming scheme for *ontologies*, i.e., formal semantics. Note that all these notations are equivalent to the triples notation with fully qualified URIs and that (medium sized) knowledge bases can easily be handled by your favourite ASCII based text editor.

One can start to write down RDF sentences without defining any ontology in advance. It's this big advantage of RDF that makes it suitable for agile approaches of information modelling. Different to the well-known Entity-Relationship modelling approach you can start from the scratch to collect information and postpone modelling questions to the future when you have collected more experience in form of sentences about the topic to be modelled.

An Example

Let's demonstrate this on a small example. Assume there is a resource at

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

that represents the XML record about the polynomial system example

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

known as *Czapor-86c*, that is stored in the SYMBOLICDATA database. Such a polynomial system can be interpreted differently as polynomial ideal in different polynomial rings (see below for more details).

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, to such a new subject has to be assigned a new URI (using a sound naming scheme, not discussed here):

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

Now we can assign meta information in the form of property-value pairs as sentences to that subject. Some of these properties can be extracted directly from the XML resource, others have to be calculated. Here is the record about that subject in Turtle notation:

```
sdideal:Czapor-86c.Flat
  a sd:Ideal ;
  rdfs:comment
      "Flat variant of Czapor-86c" ;
  sd:createdAt "1999-08-27" ;
  sd:createdBy sdp:Bachmann_O ;
  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" .
```

Some explanations: The 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:` and `sdpol:` prefixes are just abbreviations for name space prefixes

```
sd:
    http://symbolicdata.org/Data/Model#
sdp:
    http://symbolicdata.org/Data/Person/
sdpol:
    http://symbolicdata.org/Data/IntPS/
sdideal:
    http://symbolicdata.org/Data/Ideal/
```

`sdp:Bachmann_O` is the URI of the person who created that record. More information in RDF format about that person (i.e., about Olaf Bachmann) can be found in the *People* knowledge base (and queried by standard RDF techniques).

The configuration `sdideal:Czapor-86c.Flat` refers to the Polynomial System `sdpol:Czapor-86c` stored in the IntPS section of the SYMBOLICDATA XML Resources. This is described by the following RDF record:

```
sdpol:Czapor-86c
  a sd:IntegerPolynomialSystem ;
  sd:createdAt "1999-03-26" ;
  sd:createdBy sdp:Graebe_HG ;
  sd:relatedXMLResource
  <http://symbolicdata.org/
    XMLResources/IntPS/Czapor-86c.xml> .
```

The configuration *Czapor-86c.Flat* is derived from the "true" *Czapor-86c* example, that is the ideal

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

generated by the same polynomials. Geometrically it represents a complete intersection of three (generic affine) quadrics over the field of rational functions in the given parameters. There is also a record about 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
    <http://symbolicdata.org/Data/
      Ideal/Czapor-86c.Flat> ;
  sd:hasVariables "x,y,z" .
```

It 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 meta information: S'/I' is zero dimensional and has degree 8.

Using RDF

It's a good rule to use *human readable names* also for predicate URIs such that an educated reader can infere an intuitive semantic understanding (as you, hopefully, did for the above example). For machine reading only the global uniqueness of URIs is essential, hence there is no much difference between the URIs

```
<http://symbolicdata.org/Data/Model#
  hasDegree> (alias sd:hasDegree) and
<http://symbolicdata.org/Data/
  aoghuugh0shai4hae6cuzeimohz>.
```

One great thing about RDF is that at the very beginning one does not have to worry too much about the names of the predicates, since data can easily be refactored by 'search and replace' within your favourite ASCII text editor or with a small transformation script. Even a larger redesign of a knowledge base can be handled this way thus shaping and sharpening the language and the model behind the meta data according to the needs of the different communication processes to be supported.

Such a language description – the collection of classes, predicates, their relations to each other, and the rules of their application – is called *Ontology*. For different tasks there are several established ontologies as, e.g., FOAF [8] or Dublin Core [4], and it is a good advice to reuse such ontologies within your own database for the purpose and in the way they are designed for. SYMBOLICDATA heavily uses the FOAF ontology for descriptions of people and groups and the Dublin Core *dcterms* ontology [5] for bibliographical information.

To operate on RDF data the different knowledge bases (called *RDF Graphs*) have to be uploaded into a *RDF triple store*. Within SYMBOLICDATA we use the Virtuoso triple store [30] that provides also a SPARQL endpoint [26] to query the data. SPARQL is a RDF query language with syntax and expressive power similar to SQL [27] for classical relational databases. We cannot go into SPARQL details here but give only an example of a query that lists a table of all `sd:Ideal` entries with precompiled dimension, degree and lengths list:

```
PREFIX sd:
  <http://symbolicdata.org/Data/Model#>
SELECT ?a ?dim ?deg ?ll WHERE {
  ?a a sd:Ideal .
  ?a sd:hasDimension ?dim .
  ?a sd:hasDegree ?deg .
  ?a sd:hasLengthsList ?ll
}
```

You can try it out on our SPARQL endpoint [26]. More examples can be found in the "Getting started" section of the SYMBOLICDATA wiki [25].

Modelling Polynomial Systems

Whereas RDF is a common language tool to provide meta information about examples in an interoperably querable and searchable way this tool has to be applied in a domain specific way to model topics from CA subcommunities. Such modelling heavily uses domain specific concepts and semantics and even domain specific semantic aware digital tools to extract invariants and other useful information required to navigate within the data.

We explain such aspects on SYMBOLICDATA modelling again on the topic of Polynomial Systems, particularly emphasizing the differences between the representations of such systems in versions 2 and 3 of SYMBOLICDATA. Note that similar considerations are required to model any other part of the SYMBOLICDATA database. For more details about modelling other data (Free Algebras, G-Algebras, Geometry Proof Schemes etc.) we refer to the SYMBOLICDATA wiki [25].

Polynomial Systems XML resources are stored in the SYMBOLICDATA data base as files of integer or modular polynomial systems in flat XML syntax. Both are represented as lists of polynomials in distributive normal form with integer coefficients together with a complete list of variables (and the modular base domain $GF(p)$ for modular systems). 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. As in the *Czapor-86c* example each Polynomial System XML resource (the *resource*) has an RDF entry (the *resource description*) as `sd:IntegerPolynomialSystem` or `sd:ModularPolynomialSystem`.

Polynomial Systems Solving considers polynomial systems in different contexts: For a standard kind of interpretation we divide the 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. Note that S has the universal property, i.e., 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*. Note that configurations can be derived not only from Polynomial System resources but also from other configurations (as *Czapor-86c* from *Czapor-86c.Flat*).

Another way to derive a new configuration from another one is 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.

A third way to construct new configurations is by flattening. Note that in particular polynomial systems coming from Geometry Theorem Proving have a natural interpretation as generators of ideals

$$I' \subseteq S' = \mathbb{Q}(u_1, \ldots, u_k)[z_1, \ldots, z_m]$$

since the indeterminates can be divided into independent and dependent ones [11]. If $f_1, \ldots, f_s$ are denominator-free there is a natural interpretation

$$F = \{f_1, \ldots, f_s\} \subset S = R[u_1, \ldots, u_k, z_1, \ldots, z_m]$$

and another pull-back homomorphism $S \to S'$ that relates the ideal I' generated by F in S' to the ideal I generated by the pull-back images of F in S. The configuration F in S is obtained by *flattening* from the configuration F in S'. Note that also in this case the pull-back polynomial images not necessarily generate the full pull-back ideal $\phi^{-1}(I')$ in S.

The opposite operation – derive F in S' from F in S – is called *parameterizing* with respect to a given subset $\{u_1, \ldots, u_k\} \subset \{x_1, \ldots, x_n\}$ of the indeterminates considered as parameters. It corresponds to a ring push forward operation $\phi : S \to S'$. Note that by definition on the push forward polynomials generate the (possibly trivial) push forward ideal $I' = \phi(I)\, S'$.

For the moment `sd:flatten`, `sd:homogenize` and `sd:parameterize` are the only transformation modes defined within the SYMBOLICDATA Polynomial Systems database. We plan to define also a `sd:substitute` mode that defines a new configuration substituting some of the variables by integer values. The SPARQL query

```
PREFIX sd:
  <http://symbolicdata.org/Data/Model#>
select distinct ?p
from <http://symbolicdata.org/Data/
    PolynomialSystems/>
where {
  ?a a sd:Ideal .
  ?b a sd:Ideal .
  ?a ?p ?b .
}
```

returns a complete list of (URI names of) all available transformation modes.

In SYMBOLICDATA version 2 every new configuration was stored as another resource thus blowing up the Polynomial Systems database and loosing the interrelation between the different configurations. With SYMBOLICDATA version 3 we reduced the number of Polynomial Systems stored as XML resources to the necessary minimum and use shortcut notations for the different modes of construction of new configurations from old ones as described above. Thus every configuration can be reconstructed by a chain of polynomial time operations (flattening, parameterizing, homogenizing) from a distinguished XML resource.

This is a slight restriction compared to former SYMBOLICDATA versions since it requires semantic aware tools to generate configurations from given basic ones. We think that it is not a real restriction since for valuable computations on Polynomial Systems semantic aware tools are required in any case. Such tools have to provide a Polynomial Systems parser anyway to input the basic XML examples. Within that your favourite CA software being aware of polynomial semantics it should be easy to implement the transformation modes required to obtain the different configurations. See (or use) the *sdsage package* [16] by Andreas Nareike who compiled such a tool to be integrated with the Sagemath system [23].

Navigation within the Polynomial Systems Data

Let's discuss another topic that requires special semantic knowledge – navigation and identification of data. This topic is in particular important for intercommunity communication since one cannot expect people from another subcommunity to be well informed about the informal "general nonsense" commonly known to people working for years in a special CA subcommunity. Let's explain that in more detail again for Polynomial Systems Solving.

It is one of the challenges for a given Polynomial System configuration obtained from an external source to check if it 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

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

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 also 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 meta data for a given configuration. Such a fingerprint can easily be computed by almost all semantic aware (i.e., "knowing" what a polynomial is) tools. Note that configurations with different fingerprints are surely distinct, but there can be different examples with the same fingerprint (and there are such examples, mainly due to misprints in the literature, that turned easy examples into true challenges). Although the fingerprint method could be refined there was no need so far, since the examples with equal fingerprints are rare and can easily be inspected by hand. The result of the SPARQL query

```
PREFIX sd:
  <http://symbolicdata.org/Data/Model#>
select distinct ?ll ?dl count(?a)
from <http://symbolicdata.org/Data/
  PolynomialSystems/>
where {
  ?a a sd:Ideal .
  ?a sd:hasLengthsList ?ll .
  ?a sd:hasDegreeList ?dl .
} order by desc(count(?a))
```

provides a complete list of the number of configurations in the database with given fingerprints. Note that the fingerprint with degree list "3,3,3" and lengths list "4,4,4" is the only fingerprint with 14 configurations (among them 6 variations of *Czapor-86c* and 4 variations of the *Sym1* series). There are two fingerprints with 8 configurations and two with 6 configurations. All other fingerprints have at most 4 configurations in common.

Background Information – SymbolicData meets Linked Data

Classes and instances are powerful concepts in object oriented (OO) data modelling. Although there are other approaches (object cloning and factory methods) the most common operation in OO programming to create a new instance is by calling a class constructor. Note that class constructors are not well suited for agile programming since calling a constructor requires the class to be completely defined before the first instance can be generated.

RDF is a concept to describe relations between instances and classes in a much more flexible way. For example, the sentence

```
sdideal:Czapor-86c a sd:Ideal .
```

says that the URI `sdideal:Czapor-86` describes an instance of the concept (the more general name for *class* in RDF) `sd:Ideal`. `sd:Ideal` itself is an instance of the concept `owl:Class`. Note that all three artifacts `sdideal:Czapor-86`, `sd:Ideal` and `owl:Class` (and also the predicate `a` – a Turtle shortcut for `rdf:type`) are merely URIs pointing to a "real world object" (the polynomial ideal *Czapor-86c*), a concept (the class `sd:Ideal` of such polynomial ideals) and a meta-concept (the concept `owl:Class` of the "class of all classes").

`owl:Class` is part of an higher order RDF concept, called *Ontology Web Language* (OWL) [18]. It's a great advantage of RDF that one can describe things, concepts, meta-concepts and even more general superconcepts by a uniform language construct. We cannot go into details here about the differences between the concepts `rdfs:Class` and `owl:Class` but only remark, that RDF Schema (RDF-S) [22] remains on a pure descriptional level of class inheritance relations, whereas OWL interprets classes extensionally as sets of instances thus allowing for logical reasoning, e.g., about the cardinality of classes. In such an interpretation subclasses are subsets, and OWL has two built-in classes – `owl:Thing`, the anchestor super class of all classes, extensionally pointing to the universe, and `owl:Nothing`, the successor sub class of all classes, extensionally pointing to the empty set. Since we use implicitly an extensional semantics the OWL notation is more appropriate for the description of inheritance relations within SymbolicData.

These introductory remarks emphasize another big lap towards representation of background information with SymbolicData version 3. It was one of the great visions of the SymbolicData Project to collect not only benchmark and testing data but also valuable background information about the records in the database as, e.g., information about papers, people, history, systems etc. concerned with the examples in our collection. RDF provides a twofold advantage to solicite these efforts: First, with the class `owl:Thing` that stands for instances from the whole universe it provides the concept of "typeless" URIs within a typed world to point to resources of different types in a uniform way. This is similar to the class `Object` in Java. Second, one can link 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.

Whereas the second possibility remains a challenge for future SymbolicData extensions we use the former concept to represent background information as RDF records of type `sd:Annotation` with predicates

- `rdfs:label` – a label for the annotation,
- `rdfs:comment` – a text field for the annotation,
- `sd:relatesTo` – (multiple) URIs to records interrelated by that annotation.

For example, the SPARQL query

```
select ?a
from <http://symbolicdata.org/Data/
    Annotations/>
where {
  ?a a sd:Annotation .
  ?a sd:relatesTo
    <http://symbolicdata.org/Data/
      Ideal/Sym1_211> .
}
```

finds all annotations related to the ideal *Sym1_211*. The query returns the two annotations *BIB.Graebe_99a* and *Sym1*. The first one refers to a bibliographical reference of a paper that reports about benchmark computations on that and other polynomial systems to compare the polynomial systems' solving capabilities of different general purpose CAS as explained in the `rdfs:comment` entry of that annotation record. The annotation record *Sym1* describes a little bit background of series of examples where *Sym1_211* is a particular case of.

For the moment SYMBOLICDATA offers by historical reasons two kinds of annotations, but the new concept is not restricted to that. The first kind of annotations with namespace prefix

```
http://symbolicdata.org/Data/
    Annotation/BIB.
```

(including the period) relates bibliographical entries of type `sd:Reference` from the SYMBOLICDATA knowledge base *Bibliography* to different data records. The management of bibliographical references was completely redesigned with SYMBOLICDATA version 3 exploiting RDF and the established Dublin Core ontology [5] to represent bibliographical information in a way that is queryable by standard means and tools. On the other hand, we strongly reduced the part of information about bibliographical references kept inside SYMBOLICDATA since there are comprehensive bibliographical stores available in the web that provide all required information. We provide links (i.e., URIs, even if the foreign providers do not yet deliver RDF content upon HTTP request) to (at the moment) three providers of bibliographical information:

- CiteSeer at `http://citeseer.ist.psu.edu`,
- the Groebner Bases Bibliography at `http://www.risc.jku.at` and
- the Zentralblatt at `http://www.zentralblatt-math.org`.

A typical record of a bibliographical reference has the following structure

```
sdb:Canny_93a a sd:Reference;
  dct:creator sdp:Canny_JF,
      sdp:Manocha_D ;
  dct:issued "1993"^^dct:W3CDTF ;
  dct:title "MultiPolynomial
      Resultant Algorithms" ;
  sd:hasCSentry
    <http://citeseer.ist.psu.edu/viewdoc/
      summary?doi=10.1.1.38.1735> ;
  sd:hasGBBentry
    <http://www.risc.jku.at/
      Groebner-Bases-Bibliography/
      details.php?details_id=465> ;
  sd:hasZBentry
    <http://www.zentralblatt-math.org/
      zmath/en/search/?q=an:0778.13023
      &format=complete> ;
  sd:lastModified "2004-08-28" .
```

In particular we provide author information (field `dct:creator`) as URI references to our People knowledge base. Hence you can easily create a reliable SPARQL query about papers of a given person that refer to distinguished examples. We started an alignment of our people knowledge base with the author URIs provided by the Zentralblatt to offer even more search flexibility in the future. The same applies to our collection of references to CAS descriptions that is completely aligned with *swmath* [24].

The second kind of annotationals information provides background information in a stronger sense, see the contents of the `rdfs:comment` fields of the different records for more detail. We provide also a first experimental version of tags and keywords of annotations, but this requires further elaboration.

To be complete we mention that there is a third kind of annotation not yet incorporated into the new generic system of annotations, the *Geometry Problems Formulations* knowledge base relating records of type `sd:GeometryProblemFormulation` and of type `sd:GeoProofScheme`.

Towards a Computer Algebra Social Network

From the five stars to be assigned to a Linked Data project according to Tim Berners-Lee's classification [2] SYMBOLICDATA earned four stars so far (for offering data in interoperable RDF format on the web and providing a SPARQL querable triple store). To earn the fifth star one has to build up stable semantic relations to foreign knowledge bases and thus become part of the Linked Open Data Cloud [15].

Much of such interrelation, e.g., a list of interoperability references for people and bibliographical data with the Zentralblatt, is on the way as indicated in the last section. Second, we joined forces with the efforts of the board of the Fachgruppe to store and provide information about people and groups working on CA topics in Germany at their new Wordpress driven web site [7]. We developed a first prototype to store this information in RDF format as another knowledge base in the SYMBOLICDATA database, to extract it by means of SPARQL queries and to include it into the web site by the Wordpress shortcode mechanism via a special Wordpress plugin. The same technique is used to maintain informations about upcoming conferences at this site.

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 does at `http://aksw.org/Team.html`. This page is generated from RDF "on the fly" and can be delivered also in pure RDF upon HTTP request if you demand the HTTP return type not to be 'http/text' but 'rdf/xml', see the link to 'rdf/xml' in the bottom of that page.

- Maintain a 'foaf:ProfileDocument' 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, see `http://aksw.org/NatanaelArndt.html` and `http://aksw.org/NatanaelArndt.rdf` (the former page is generated from the latter information base). All AKSW members have such "RDF aware" personal pages.

- Organize a regular harvesting process within the CA 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 CA 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 that the community agreed upon.

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* [29] running, different to Facebook, on a multitude of 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 for more information. Even if all this is very prealpha yet, the future is already on the way. Don't miss the train.

Literatur

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

[2] T. Berners-Lee. 5 ∗ Open Data. `http://5stardata.info/` [2014-03-05]

[3] The DBLP Computer Science Bibliography. `http://www.informatik.uni-trier.de/~ley/db/` [2014-02-19]

[4] The Dublin Core Metadata Initiative. `http://dublincore.org/` [2014-02-27]

[5] DCMI Metadata Terms. `http://dublincore.org/documents/dcmi-terms/` [2014-02-27]

[6] Das eScience-Forschungsnetzwerk Sachsen. `http://www.escience-sachsen.de` [2014-02-19]

[7] Website der Fachgruppe Computeralgebra der GI in Kooperation mit der DMV und GAMM. `http://www.fachgruppe-computeralgebra.de/` [2014-03-06]

[8] The Friend of a Friend (FOAF) project. `http://www.foaf-project.org/` [2014-02-27]

[9] FRISCO – A Framework for Integrated Symbolic/Numeric Computation. `http://www.nag.co.uk/projects/FRISCO.html` [2014-02-19]

[10] E. Gawrilow and M. Joswig. polymake: a framework for analyzing convex polytopes. *Polytopes — combinatorics and computation (Oberwolfach, 1997)*, 43–73, DMV Sem., 29, Birkhäuser, Basel, 2000. MR1785292 (2001f:52033).

[11] H.-G. Gräbe: Geometrie mit dem Computer. Skript zur Vorlesung im Wintersemester 2010/11. `http://www.informatik.uni-leipzig.de/~graebe/skripte/geometrie10.pdf` [2014-02-28]

[12] The GND Project. `http://www.dnb.de/DE/Standardisierung/GND/gnd_node.html` [2014-02-19]

[13] A. Heinle. The SDEval framework. `http://symbolicdata.org/wiki/SDEval` [2014-02-28]

[14] J. Klüners and G. Malle. A Database for Number Fields. `http://galoisdb.math.uni-paderborn.de` [2014-02-20]

[15] Linked Data. `http://en.wikipedia.org/wiki/Linked_data` [2014-03-05]

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

[17] OntoWiki: A tool providing support for agile, distributed knowledge engineering scenarios. `http://aksw.org/Projects/OntoWiki.html` [2014-02-19]

[18] OWL Web Ontology Language. Reference W3C Recommendation 10 February 2004. `http://www.w3.org/TR/owl-ref/` [2014-03-02]

[19] A. Paffenholz. Lists of Combinatorial Types of Birkhoff Faces. `http://polymake.org/polytopes/paffenholz/www/birkhoff.html` [2014-02-20]

[20] A. Paffenholz. Smooth Reflexive Lattice Polytopes. `http://polymake.org/polytopes/paffenholz/www/fano.html` [2014-02-20]

[21] The PoSSo Project. http://posso.dm.unipi.it/ [2014-02-19]

[22] RDF Schema 1.1. W3C Recommendation 25 February 2014. http://www.w3.org/TR/rdf-schema/ [2014-03-02]

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

[24] swMATH – an information service for mathematical software. http://swmath.org/ [2014-03-05]

[25] The SYMBOLICDATA Project Wiki. http://symbolicdata.org/wiki [2014-02-19]

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

[27] SQL – Structured Query Language. See http://en.wikipedia.org/wiki/SQL [2014-02-27]

[28] J. Tauberer. Quick Intro to RDF. http://www.rdfabout.com/quickintro.xpd [2014-02-20]

[29] S. Tramp et al. DSSN: towards a global Distributed Semantic Social Network. http://aksw.org/Projects/DSSN.html [2014-03-06]

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