Formula Semantification and Automated Relation Finding in the On-line Encyclopedia for Integer Sequences2

Enxhell Luzhnica and Michael Kohlhase

Computer Science, Jacobs University Bremen, Germany

Abstract. The On-line Encyclopedia of Integer Sequences (OEIS) is an important resource for mathematicians. The database is well-structured and rich in mathematical content but is informal in nature, so knowledge management services are not directly applicable. In this paper we provide a partial parser for the OEIS that leverages the fact that, in practice, the syntax used in its formulas is fairly regular. Then, we import the result into OMDoc to make the OEIS accessible to OMDoc-based knowledge management applications. We exemplify this with a formula search application based on the MathWebSearch system and a program that finds relations between the OEIS sequences.

1 Introduction

Integer sequences are important mathematical objects that appear in many areas of mathematics and science and are studied in their own right. The On-line Encyclopedia of Integer Sequences (OEIS) [Inc15] is a publicly accessible, searchable database documenting such sequences and collecting knowledge about them. Sequences can be looked up using a text-based search functionality that OEIS provides, most notably by giving the name (e.g. "Fibonacci") or starting values (e.g. "$1, 2, 3, 5, 8, 13, 21$"). However, given that the source documents describing the sequences are mostly informal text, more semantic methods of knowledge management and information retrieval are limited.

In this paper we tackle this problem by building a formula parser for the source documents and exporting them in content MathML, the pertinent XML-based standard. This opens up the OEIS library to knowledge management applications, which we exemplify by a semantic search application based on the MathWebSearch [HKP14] system that permits searching for text and formulas and by a relation finder that induces new relations from the parsed formulae. This paper is based on [Luz16] to which we refer for details we had to elide.

2 The OEIS

The OEIS is a web information system about integer sequences. Started in 1964 by Neil Sloane, an active community now curates over 250 000 sequences, collecting their starting values, literature references, implementations, and formulae that encode representations and relations between sequences. This data is stored in a line-keyed ASCII documents

internally. We introduce this by way of the snippet in Figure 1 – we will use the Fibonacci numbers as the running example. There we see a document fragment with identification (`%I`), values (`%S`), name (`%N`) and reference (`%D`) lines, followed by three formula lines (`%F`) and the author line (`%A`). The formula lines are the main object of interest in this paper.

```
%I A000045 M0692 N0256
%S A000045  0,1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987
%N A000045 Fibonacci numbers: F(n) = F(n−1) + F(n−2) with F(0) = 0 and F(1) = 1.
%D A000045 V. E. Hoggatt, Jr., Fibonacci and Lucas Numbers. Houghton, Boston, MA, 1969.
%F A000045 F(n) = ((1+sqrt(5))^n−(1−sqrt(5))^n)/(2^n∗sqrt(5))
%F A000045 G.f.: Sum_{n>=0} x^n ∗ Product_{k=1..n} (k + x)/(1 + k∗x). − _Paul D. Hanna_,
      Oct 26 2013
%F A000045 This is a  divisibility  sequence; that  is ,  if  n  divides  m, then a(n)  divides  a(m)
%A A000045 _N. J. A. Sloane_, Apr 30 1991
```

Fig. 1: The OEIS sources for Sequence `A000045` (Fibonacci Numbers)

The bulk of formulae in the OEIS consist of **generating functions** used to represent sequences efficiently by coding the terms of a sequence as coefficients of powers of a variable x in a formal power series. Unlike an ordinary series, this formal series is allowed to diverge, meaning that the generating function is not always a true function and the "variable" is actually an indeterminate.

There are different types of generating functions, including **ordinary generating functions, exponential generating functions, Lambert series, Bell series**, and **Dirichlet series**. The **ordinary generating function** (or just **generating function**) of a sequence $a_0, a_1, a_2 \ldots, a_{n-1}, a_n, \ldots$ is the infinite series:

$$GF(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{n-1}x^{n-1} + a_nx^n + \ldots = \sum_{i=0}^{\infty} a_ix^i \quad (1)$$

For example, the sequence $A000012 = (1, 1, 1, 1, 1, \ldots)$ can be represented as:

$$1 + x + x^2 + \ldots = \frac{1}{1-x} \quad (2)$$

We know that the equation above only holds for $|x| < 1$ but we ignore the issues of convergence, as already mentioned. Thus, the ordinary generating function of this sequence can be written as $\frac{1}{1-x}$.

3 Parsing OEIS Formulae

We built a partial parser for OEIS formulas by identifying and analyzing well-behaved formulas to produce a workable grammar. We leverage the fact that, although there is no standardized format for OEIS formulas, many of them use a sufficiently regular syntax. At the core, the parser uses a rather standard grammar for infix, suffix, and prefix operators and binding operators with precedences. Instead of presenting it in detail we discuss some of the challenges we encountered:

Open Set of Primitives Since the formulas are not standardized, not only is the syntax flexible, but so is the set of primitive operators that are used. For instance, the formulas in Figure 1 (on lines 5-6) use square root, power, as well as the sum (Σ) and product (Π) binders. The challenges arise because of the many different notations used for such primitives. For instance, in line 6 of Figure 1 the range for sum and product is given in two different ways. Similar problems appear with limits and integrals as well as numerous atypical infix and suffix operators. In order to parse these correctly, we investigate the documents and the grammar failures manually and incrementally extend the grammar.

Ambiguity As it is often the case with informal, presentation-oriented formulas, there can be ambiguity in the parsing process when there exist several reasonable interpretations. Since the OEIS syntax is not fixed, this is quite common, so we do additional disambiguation during parsing to resolve most of the ambiguities. Some common ambiguities are:

- Implicit multiplications: `a*(x+y)` is usually written as `a(x+y)` which is ambiguous since it can also be parsed as a function application.
- Natural way of using the power operator: `T^2(y)` is used for $(T(y))^2$, however `T^y(x^2+2)` is ambiguous.
- Unbracketed function applications: `sin x` is a common way of writing $sin(x)$. However, this form of function application can also be parsed as multiplication between variable `sin` and `x`, as in `Pi x`.

We employ heuristics based on a type system that we use to assign types to each of the parsed terms to resolve these ambiguities.

Delineating formulas OEIS formula lines freely mix text and formulas so it is required to correctly distinguish between text and formula parts within the lines in order to accurately parse each line. For instance, line 6 in Figure 1 starts with the text `G.f.:` (meaning "Generating function:") and continues with the formula. The line then has the author and date, separated from the formula by a dash (-) which could also be interpreted as a minus and, therefore, a continuation of the formula. In the extraction of the formulas we use the help of a dictionary. The text in the OEIS documents has words that are not found in the dictionaries since it contains many technical terms so we first run a pre-parsing procedure which enriches the dictionary. The final grammar tries to parse words until it fails and then tries to parse formulas; this process repeats.

Formula parsing The formula parser is implemented using the Packrat Parser for which Scala provides a standard implementation. Packrat parsers allow us to write left recursive grammars while guaranteeing a linear time worst case which is important for scaling to the OEIS.

There are 223866 formula lines in OEIS and the formula parser succeeds on 201384 (or 90%) of them. Out of that, 196515 (or 97.6%) contain mathematical expressions. Based on a manual inspection of selected formulas we determined that most parser fails occur because of logical connectives since those are not yet supported. Other failures include wrong formula delineation because of unusual mix of formulas and text. We did

a manual evaluation of the parsing result for 40 randomly selected OEIS documents and evaluated 85% of successfully parsed formulas as semantically correct.

The importer is implemented in Scala as an extension for the MMT system and consists of about 2000 lines of code. It is available at `https://svn.kwarc.info/repos/MMT/src/mmt-oeis/`.

There are 257654 documents in OEIS totaling over 280MB of data. The OMDoc/MMT import expands it to around 9GB, partly due to the verbosity of XML and partly due to producing the semantic representation of formulas. The total running time is around 1h40m using an Intel Core i5, 16GB of RAM and a SATA hard drive.

Search MATHWEBSEARCH (MWS) is an open-source, open-format, content-oriented search engine for mathematical expressions. We refer to [HKP14] for details.

To realize the search instance in MWS we need to provide two things:

1. A *harvest* of MATHML-enriched HTML files that the search system can resolve queries against. The content-MATHML from the files will be used to resolve the formula part of the query while the rest of the HTML will be used for the text part. The harvest additionally requires a configuration file that defines the location in the HTML files of MWS-relevant metadata such as the title, author or URL of the original article. This, together with the HTML itself is used when presenting the query results.
2. A *formula converter* that converts a text-based formula format into MATHML. This will be used so that we can input formulas for searching in a text format (in our case OEIS-inspired ASCII math syntax) rather than writing MATHML directly.

To produce the harvest of the OEIS library for MWS we export the HTML from the content imported into MMT. We reuse the MMT presentation framework and only enhance it with OEIS-specific technicalities such as sequence name or OEIS link. For the formula converter we use the same parser used for OEIS formulas and described above, except extended with one grammar rule for MWS *query variables*. Figure 2 shows (a part of) the current interface answering a query about Fibonacci numbers. The search system is available at `http://oeis.search.mathweb.org`.

4 Relation Finding

Part of the mathematical interest in the OEIS is that it gives interpretations of sequences and provides a basis for establishing relations between them. Consequently extending the latter has been an important concern. As the initial values of the sequences were the only machine-actionable part of the OEIS, relation-finding has concentrated on them. However it is important to note that even an exact match of initial subsequences can never verify a relation, thus any numeric match can only be a relation conjecture. An extreme example of two sequences that match for 777451915729367 terms but are not equal, is $\left\lfloor \frac{2n}{log(2)} \right\rfloor$ and $\left\lceil \frac{2}{2^{1/n}-1} \right\rceil$ [N J12]. Ralf Stephan found 117 conjectures from which 17 of them are still open [Ste04].

Our database of parsed formulae allows us to do better: we can directly look for relations between the formula representations, most prominently between the generating formulae. The approach we follow is mathematically simple. We will show two methods,

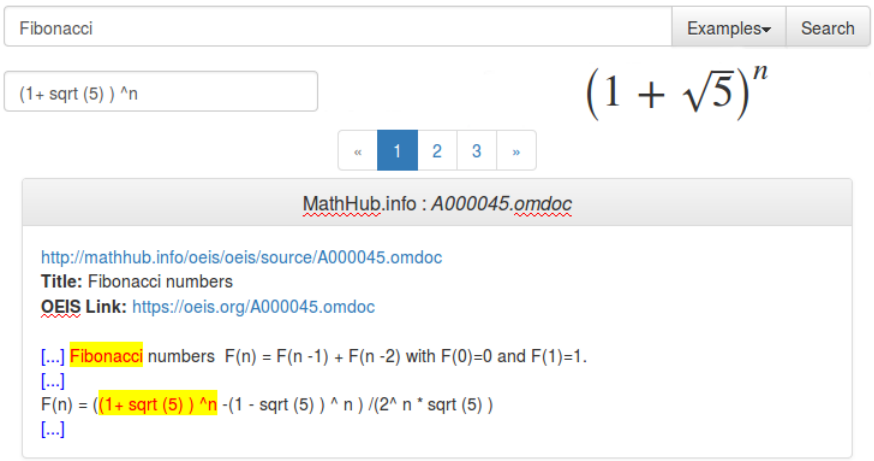


Fig. 2: Text and Formula Search for OEIS

the second building on top of the first. Refer to [Luz16] for a more elaborate discussion and another method.

Method 1 In this case, we *normalize* the ordinary generating functions of the sequences and check for equality between the normalized expressions. The normalization rules are defined as follows:

$$\frac{cG \quad c \text{ - constant} \quad G \text{ - generating function}}{G} \text{ (Const)}$$

$$\frac{x^n G \quad x \text{ - the indeterminate of } G \quad G \text{ - generating function}}{G} \text{ (Unshift)}_n$$

$$\frac{P/Q \quad P, Q \text{ - polynomials}}{(\sum_{i=0}^{n} p_i x^i)/(\sum_{i=1}^{m} q_i x^i) \quad p_n > 0 \quad q_n > 0} \text{ (Sort)}$$

Intuitively, in this case we are checking if sequences are scaled and/or shifted versions of each other. These relations are not meant to be interesting or new.

Method 2 In this case we check if a sequence can be expressed as a sum of other sequences existing in the OEIS, possibly *transformed* and/or *normalized*.

A simplified algorithm roughly follows this pseudocode:

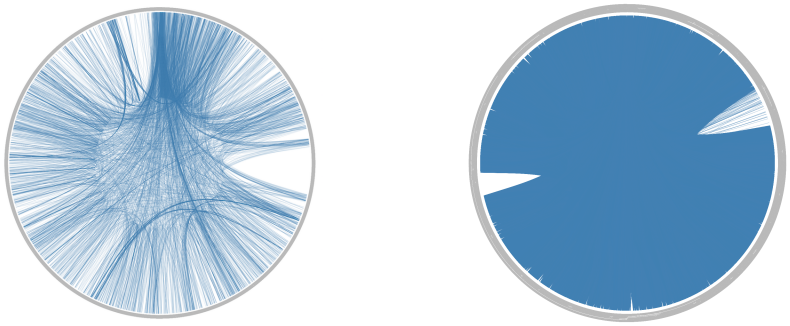
```
foreach sequence
  foreach ogf in ordinaryGeneratingFunction(sequence)
    add normalize(ogf) to hashSet
foreach sequence
  foreach ogf in ordinaryGeneratingFunction(sequence)
    pdf = partialFractionDecomposition(ogf)
```

```
partialFractions = decompose(partialFractions(pfd))
relationsExists =
forall pgf in partialFractions
  transformedPartialFractions = normalize(applyTransformations(pgf))
  transformedPartialFractions.intersection(hashSet).length > 0
if (relationsExists)
  print relations
```

We will now explain the functions that we are using above.

Let GF_n be one of the ordinary generating functions of sequence A_n. The partial fraction decomposition (*partialFractionDecomposition*) would leave us with $GF_n = \sum_{j=1}^{n} G_j$ where G_j is also an ordinary generating function. The function *partialFractions* extracts the summands, in this case, the partial fractions (ordinary generating functions) themselves. The function *decompose* does a further step of decomposition. If $G_j = \frac{P}{Q}$ where P, Q are polynomials ($P = \sum_{i=0}^{n} a_i x^i$) then it rewrites $G_j = \sum_{i=0}^{n} \frac{a_i x^i}{Q}$. These summands are then considered partial fractions too.

The transformations are *integration, differentiation* and *unit*. The transformations are selected such that expressions that match under these transformations can be easily related both mathematically and semantically.



(a) OEIS Relation Graph of Current Relations (b) OEIS Relation Graph after Method 2

The points around the circle represent the theories and the blue lines views between them. The theories presented here are only the ones for which we have parsed the generating functions.

Fig. 3: OEIS Relation Graphs

The relation finder is implemented in Scala and is available at `https://github.com/eluzhnica/OEIS`. The page will be kept up to date with results. The implementation of the *normalization* rules makes use of the parsing tree of the expression. The *transformations* are done using SageMath [Dev16] as a math engine. Our Scala code communicates with a local SageMath server using a REST API.

We show below some examples of the relations found from each method.

Method 1 This is more of a sanity check of the data. Due to the nature of these relations, these are self-evident relations. Additionally, these relations can be effectively searched utilizing a numerical method. An instance that our algorithm finds is that sequence $A001478(n) = -A000027(n)$. Sequence $A001478$ is the sequence of the negative integers, while $A000027$ is the sequence of positive integers.

Method 2 An example of this method, which we have submitted and it is accepted in the OEIS (`https://oeis.org/A037532`), is as follows.

$$A037532(n) = \frac{5}{57}A049347(n-1) + \frac{3}{57}A049347(n) + \frac{29}{171}A000420(n) - \frac{2}{9} \quad (3)$$

There is one subtlety that needs to be explained. The sequence with ordinary generating function $\frac{1}{1-x}$ is the sequence $(1, 1, 1, \ldots)$. However, for simplicity we write down $\frac{2}{9}$ instead of $\frac{2}{9}A000012(n)$.

Since our parser runs over all the formulas of OEIS, we have extracted the existing explicit relations in OEIS and made a graph (Figure 3a) showing the existing connections between sequences . The second method enriches the theory graph as shown in Figure 3b.

We converted the parsed generating functions to the SageMath syntax and checked if SageMath can compute with the expressions. From manual inspection, we found out that most of the unaccepted cases were referencing functions defined somewhere within the document. For instance, $1 + Q(0)$ where the function $Q(n)$ is defined later on in the sequence document. We currently do not resolve these references.

Parsed Generating Functions	43 005
SageMath verified Generating Functions	16 065
Parsed Ordinary Generating Functions	35 953
SageMath verified Ordinary Generating Functions	13 400
Method 1 relations	4 859
Sequences in Method 1 relations	853
Method 2 relations	297 284 646
Method 2 relations without normalization	66 427

Table 1: Evaluation of the Relation Finder

Method 1 reports 4859 relations of that kind. However, in total only 853 sequences can be normalized to other existing sequences.

It is noticeable that there are a lot of relations generated from the second method. This is due to the number of relations found using the normalization rules (Method 1). Take for instance, $G = A + B + C$, and say that each of A, B, C is an OEIS sequence and is related with 3 other sequences under the normalization rules. Then the number of relations that we can form is actually 4^3. For this reason, we also report the number of relations when we remove the relations that come due to the normalization. So, in the example above the relation would count only once, instead of 4^3.

Out of three submissions, two relations are already accepted in the OEIS. One of them has already been presented in Equation 3 and the other relation is $A001787(n) = A007283(n)\frac{n}{6}$ which can be found at `https://oeis.org/A001787`. The unaccepted

submission was not perceived to add new information since a similar relation was already present. The submitted relations were selected randomly.

5 Conclusions and Future Work

We improved the digitalization of the OEIS by parsing the formulae. Even though our parser can definitely be improved, it already supports two important added-value services. First, the MathWebSearch instance on OEIS which allows the users to search the OEIS by text formula queries. Second, a way of generating knowledge from OEIS, specifically, relations between sequences. The relation finding experiment presented above only uses very simple mechanisms for finding relations between generating functions. We make the parsed and induced formulae in content MathML form at `https://github.com/eluzhnica/OEIS` to allow other parties to extend our methods and find even more relations.

Acknowledgements We acknowledge financial support from the OpenDreamKit Horizon 2020 European Research Infrastructures project (#676541), and thank the OEIS community for support Neil Sloane for giving us access to a full OEIS dump and Jörg Arndt for fruitful discussions. All the work reported in this paper has only been possible, since the OEIS foundation had the foresight to license the contents under a CreativeCommons license that allows derivative works.

Sustainability To make our work sustainable, we would need to *i*) periodically re-run our system on future versions of the OEIS and *ii*) feed the results back into the knowledge base. The first needs a setup which facilitates change management, minimally a way to query the OEIS for changes like the OAI-PMH, but even better, maintaining the OEIS sources in a revision control system like GIT. For the second we note that with the huge volume of induced formulae, manual submission to the OEIS cannot be the answer. Automated submission – while simple to implement – would overwhelm the OEIS editors.

References

[1] Radu Hambasan, Michael Kohlhase, and Corneliu Prodescu. "MathWebSearch at NTCIR-11". In: *NTCIR Workshop 11 Meeting*. Ed. by Noriko Kando, Hideo Joho, and Kazuaki Kishida. Tokyo, Japan: NII, Tokyo, 2014, pp. 114–119.

[2] The OEIS Foundation Inc. *The On-Line Encyclopedia of Integer Sequences*. `http://oeis.org/`. 2015.

[3] Enxhell Luzhnica. "Formula Semantification and Automated Relation Finding in the OEIS". B. Sc. Thesis. Jacobs University Bremen, 2016. URL: `https://github.com/eluzhnica/OEIS/doc/Enxhell_Luzhnica_BSC.pdf`.

[4] N. J. A. Sloane. *The On-Line Encyclopedia of Integer Sequences*. `http://neilsloane.com/doc/eger.pdf`. 2012.

[5] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 7.1)*. `http://www.sagemath.org`. 2016.

[6] Ralf Stephan. *State of 100 Conjectures From The OEIS*. `http://www.ark.in-berlin.de/conj.txt`. 2004.