Importing the OEIS library into OMDoc

Enxhell Luzhnica, Mihnea Iancu, Michael Kohlhase

Computer Science, Jacobs University, Bremen, Germany
initial.last@jacobs-university.de

Abstract. The On-line Encyclopedia of Integer Sequences (OEIS) is the largest database of its kind and 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.

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) [6] is a publicly accessible, searchable database documenting such sequences and collecting knowledge about them. The effort was started in 1964 by N. J. A. Sloane and led to a book [10] describing 2372 sequences which was later extended to over 5000 in [11]. The online version [8] started in 1994 and currently contains over 250000 documents from thousands of contributors with 15000 new entries being added each year [9]. Documents contain varied information about each sequence such as the beginning of the sequence, its name or description, formulas describing it, or computer programs in various languages for generating it.

The OEIS library is an important resource for mathematicians. It helps to identify and reference sequences encountered in their work and there are currently over 4000 books and articles that reference it. 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 (partial) parser for the source documents and importing the OEIS library into the OMDoc/MMT format which is designed for better machine support and interoperability. This opens up the OEIS library to

 In: R. Bergmann, S. Görg, G. Müller (Eds.): Proceedings of the LWA 2015 Workshops: KDML, FGWM, IR, and FGDB. Trier, Germany, 7.-9. October 2015, published at http://ceur-ws.org

OMDoc-based knowledge management applications, which we exemplify by a semantic search application based on the MathWebSearch [4] system that permits searching for text and formulas.

This paper is organized as follows: in Section 2 we describe our import of the OEIS library into OMDoc. In Section 3 we show an initial application of our import by providing formula search for the OEIS library. Then, in Section 4 we discuss future work and conclude.

Acknowledgements This work has been supported by the German Research Foundation (DFG) under grant KO 2428/13-1. The authors gratefully acknowledge the foresight of the OEIS foundation to license the OEIS content under a Creative Commons license that allows derivative work like this one and the practical help of Jörg Arndt in obtaining the OEIS corpus.

2 Translating OEIS to OMDoc

The OEIS database is stored as a collection of text documents (one for each sequence) written in the *internal format* of OEIS which defines the document-level structure of the sources. Therefore, parsing the *document structure* is straightforward. However, at the formula level the format is not standardized which makes parsing them non-trivial. Still, in practice, the syntax used in the formula snippets is somewhat regular and we built a *formula parser* that succeeds on most OEIS formulas.

2.1 Preliminaries

OMDoc [7] is a content markup format and data model for mathematical documents. It models mathematical content using three levels.

Object Level: uses OpenMath and MathML as established standards for the markup of *formulae*.

Statement Level: supplies original markup for explicitly representing the various kinds of mathematical statements including *symbol declarations* and *definitions* (which introduce a new symbol names), *assertions* (which can represent theorems, lemmas or corollaries), and *examples*.

Theory Level: offers original markup that allows for clustering sets of statements into *theories* as well as specifying relations between them (inclusions, morphisms).

The Mmt [14] language can be seen as a restricted version of OMDoc but with a fully specified semantics. Additionally, for Mmt there is an Mmt system [13] which is a Scala-based [2] open source implementation of the Mmt language. The key features of the Mmt system for this paper are that it provides an infrastructure for writing importers from any compatible format into Mmt as well as exporters from Mmt, most notably into (MathML-enriched) HTML for both local and web-based presentation.

For the sake of simplicity, we often do not differentiate between Mmt and OMDoc languages in the following and refer to [7] and, respectively, [14] for details on each language. Throughout this paper we will use OMDoc/MMT to refer to both OMDoc and Mmt.

2.2 The OEIS document format

The *internal format* [12] is line-based in the sense that each line starts with a marker that represents the kind of content found in that line. We briefly introduce the relevant kinds below but refer to [12] for details.

The *identification* line gives the unique ID of the sequence declared in that document and the *name* line gives the name, a brief description or the definition of the sequence. There are also *start values* lines which give the beginning of the sequence. *Formula* lines give formulas that define or hold for the sequence described in the current document. The formulas are in plain text ASCII syntax that is similar to LaTeX math markup and can contain text as part of the formula or as comments. There are many other dedicated line types including those for implementations (in various programming languages), references, examples, or comments.

Running Example 1 (Fibonacci numbers). The article for Fibonacci numbers [5] was one of the first entries in the OEIS and is one of the most comprehensive. We will use it as a running example throughout the paper, although we will heavily trim the document for simplicity by presenting here only a few sanitized lines. Listing 1 shows the document with identification, values, name and reference lines, followed by three formula lines and the author line.

```
1 %I A000045 M0692 N0256
2 %S A000045 0,1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987
3 %N A000045 Fibonacci numbers: F(n) = F(n-1) + F(n-2) with F
    (0) = 0 and F(1) = 1.
4 %D A000045 V. E. Hoggatt, Jr., Fibonacci and Lucas Numbers.
    Houghton, Boston, MA, 1969.
5 %F A000045 F(n) = ((1+sqrt(5))^n-(1-sqrt(5))^n)/(2^n*sqrt(5))
6 %F A000045 G.f.: Sum_{n>=0} x^n * Product_{k=1..n} (k + x)/(1
     + k*x). - _Paul D. Hanna_, Oct 26 2013
7 %F A000045 This is a divisibility sequence; that is, if n
    divides m, then a(n) divides a(m)
8 %A A000045 _N. J. A. Sloane_, Apr 30 1991
```

2.3 Parsing the OEIS Formula Format

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.

OEIS is known for the human-readable mathematical terms, so a variety of syntactic rules are encountered when forming these mathematical terms. We use the following classification for notations, inspired by [1] and further motivated by our analysis of the OEIS formulas:

1. **infix operators** are used to combine two terms to one complex term, e.g the + symbol in `m+n`.
2. **suffix operators** are added after a term to form another term, e.g. the `!` symbol in `n!`.

3. **prefix operators** (with or without bracketed application) are added in front of a term to form another term, e.g. `sin` in `sin(x)` or `sin x`, respectively.
4. **infix relation symbols** are used to construct a formula out of two terms, e.g. the < symbol in `x<2`.
5. **binding operators** that bind a context to a body to construct a term, e.g. the ∀ symbol in `∀x. x^2 > 0`

The classification presented above guides our grammar and, in principle, covers virtually all important notations used in OEIS formulas. However, in practice, we encountered several important challenges which we discuss individually below.

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 Listing 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 Listing 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. Here we discuss a few of the many ambiguities that arise.

The multiplication sign is usually implicit so, instead of `a*(x+y)`, we encounter `a(x+y)` which could represent either a function application or a multiplication depending on whether `a` is a function or an individual (constant or variable). There is no general way to solve this, so we rely on several heuristics. First, we check if the symbol in question is used somewhere else in the same formula with an unambiguous meaning. Specifically, we default to function application unless the same symbol is used as an individual somewhere else in the formula. This already disambiguates most such cases in OEIS but we use several additional heuristics. For instance, having `name(`arg`)` will result in marking `name` as function since it is unlikely to be a multiplication between two individuals. Similarly, having `name(`arg_1`,...,`arg_n`)` results in marking `name` as a function.

The natural way of using the power operator also leads to ambiguities. For example, `T^2(y)` is used for $(T(y))^2$, however `T^y(x^2+2)` is ambiguous. We solve this using similar heuristics as for the implicit multiplication.

For unbracketed function application as in `sin x`, we rely on the heuristic that this form of function application is used only in well known functions. Therefore, we code these notations for well known functions in the grammar itself. This form of function application can also mean multiplication, for instance `Pi x`. One can already see that parsing and disambiguating the mathematical expressions in this context has a lot of aspects. Additional cases of ambiguities are handled in similar ways and we omit the details for brevity.

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 Listing 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.

2.4 Importing into OMDoc/MMT

For each OEIS document we create a corresponding OMDoc/MMT document that contains a single theory. Then, OEIS lines roughly correspond to OMDoc/MMT declarations inside that theory. We use the OEIS sequence ID as the name of the OMDoc theory. Then, the identification line produces an OMDoc symbol declaration representing the sequence (as a function from integers to values). The start values and example lines are both represented as OMDoc/MMT examples. Specifically, the starting values are considered as examples of sequence elements. Formula lines are represented as OMDoc assertions (about the sequence symbol). Finally, name, reference and author lines are represented as metadata using the Dublin Core standard.

Running Example 2 (Fibonacci Numbers). The corresponding OMDoc/MMT document for the Fibonacci numbers article from Example 1 is shown in Listing 2. We omit most formulas and some XML boilerplate for conciseness and simplicity.

```
1  <omdoc xmlns:dc="http://purl.org/dc/elements/1.1/">
2    <theory id="A000045">
3    <metadata>
4      <dc:creator>N. J. A. Sloane</dc:creator>
5      <dc:title>Fibonacci numbers</dc:title>
6    </metadata>
7    <symbol name="seq"/>
8    <assertion>
9      <!-- OpenMath for ∀n.seq(n) = ((1+√5)^n−(1−√5)^n)/(2^n √5) -->
10     <OMBIND>
11       <OMS cd="arith" name="forall"/>
12       <OMBVAR> <OMV name="n"/> </OMBVAR>
13       <OMA>
14         <OMS cd="arith" name="equal"/>
15         <OMA><OMS name="seq"/><OMV name="n"></OMA>
16           ⋮
17       </OMA>
18     </OMBIND>
19   </assertion>
20   ⋮
```

```
21      </theory>
22    </omdoc>
```

2.5 Implementation and Evaluation

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/`. The implementation is mostly straightforward, other than the formula parser which we discuss separately below.

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.

Formula parsing The formula parser is implemented using the Packrat Parser [3] 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.

The statistics above refer just to the successful parses, but we cannot automatically evaluate if the result returned by the parser is actually the expected one. For this, we did a manual evaluation of the parsing result for 40 randomly selected OEIS documents and evaluated 85% of succesfully parsed formulas as semantically correct. The main contributor of incorrect formula parses was badly delineated formulas, which causes text to be wrongly parsed as part of a formula.

3 Application: Search

MathWebSearch (MWS)[4] is an open-source, open-format, content-oriented search engine for mathematical expressions. We refer to [4] 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.

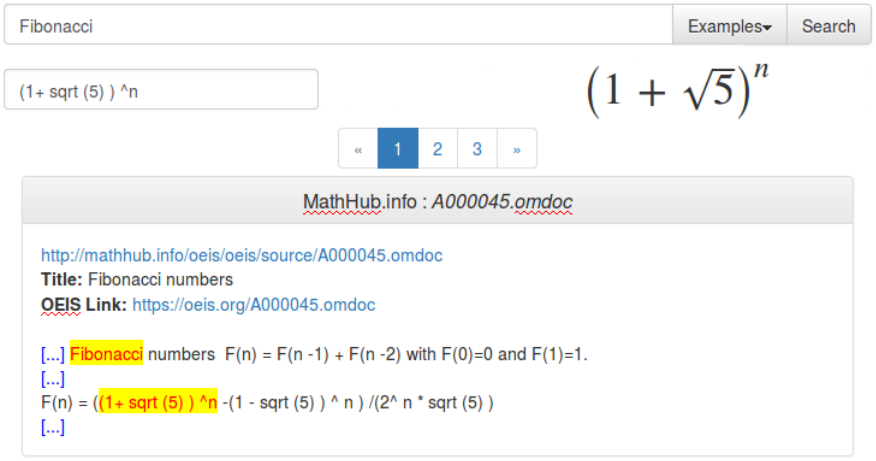

Fig. 1: Text and Formula Search for OEIS

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*. We then forward the resulting formula in MMT to produce the presentation (MATHML) and return it to the MWS frontend. The web-server infrastructure, needed to communicate with MWS, is provided by MMT and we just extend it. Figure 1 shows (a part of) the current interface answering a query about Fibonacci numbers. The search system is available at `http://ash.eecs.jacobs-university.de:9999/`.

4 Conclusion and Future Work

We presented a partial parser for the On-line Encyclopedia of Integer Sequences that covers the majority of formulas and an import of the parsed OEIS into OMDOC. We exemplified the added value by providing a formula-search service for the OEIS based on the MATHWEBSEARCH system. Our importer does not currently handle all line types in OEIS, most notably the program code lines. We also only analyze formulas that appear inside formula lines, but in OEIS they may appear elsewhere (for instance instead of the sequence name or inside comment lines). In the future, we plan to extend the structure parser to cover these cases as well as improve the formula parser to handle some of the failures discussed in Section 2. Moreover, since we have the defining formulas in their content representations for a significant number of sequences, we plan to analyze them to try and find additional relations between sequences as well as generate new ones.

References

[1] Marcos Cramer, Peter Koepke, and Bernhard Schröder. "Parsing and Disambiguation of Symbolic Mathematics in the Naproche System". In: *Intelligent Computer Mathematics - 18th Symposium, Calculemus 2011, and 10th International Conference, MKM 2011, Bertinoro, Italy, July 18-23, 2011. Proceedings*. Ed. by James H. Davenport et al. Vol. 6824. Lecture Notes in Computer Science. Springer, 2011, pp. 180–195. DOI: 10.1007/978-3-642-22673-1_13. URL: http://dx.doi.org/10.1007/978-3-642-22673-1_13.

[2] École polytechnique fédérale de Lausanne. *The Scala Programming Language*. URL: http://www.scala-lang.org (visited on 10/22/2009).

[3] Bryan Ford. "Packrat Parsing:: Simple, Powerful, Lazy, Linear Time, Functional Pearl". In: *Proceedings of the Seventh ACM SIGPLAN International Conference on Functional Programming*. ICFP '02. Pittsburgh, PA, USA: ACM, 2002, pp. 36–47. DOI: 10.1145/581478.581483. URL: http://doi.acm.org/10.1145/581478.581483.

[4] Radu Hambasan, Michael Kohlhase, and Corneliu Prodescu. "MathWebSearch at NTCIR-11". In: *NTCIR 11 Conference*. Ed. by Noriko Kando and Hideo Joho andKazuaki Kishida. Tokyo, Japan: NII, Tokyo, 2014, pp. 114–119. URL: http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings11/pdf/NTCIR/Math-2/05-NTCIR11-MATH-HambasanR.pdf.

[5] The OEIS Foundation Inc. *Fibonacci Numbers, The On-Line Encyclopedia of Integer Sequences*. http://oeis.org/A000045. 2015.

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

[7] Michael Kohlhase. OMDoc – *An open markup format for mathematical documents [Version 1.2]*. LNAI 4180. Springer Verlag, Aug. 2006. URL: http://omdoc.org/pubs/omdoc1.2.pdf.

[8] N. J. A. Sloane. *An On-Line Version of the Encyclopedia of Integer Sequences*. http://www3.combinatorics.org/Volume_1/PDF/v1i1f1.pdf. 1994.

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

[10] N.J. A. Sloane. *A Handbook of Integer Sequences*. Academic Press, 1973.

[11] N.J. A. Sloane and Simon Plouffe. *The Encyclopedia of Integer Sequences*. Academic Press, 1995.

[12] *OEIS Help*. http://oeis.org/eishelp1.html.

[13] Florian Rabe. "The MMT API: A Generic MKM System". In: *Intelligent Computer Mathematics*. Conferences on Intelligent Computer Mathematics. (Bath, UK, July 8–12, 2013). Ed. by Jacques Carette et al. Lecture Notes in Computer Science 7961. Springer, 2013, pp. 339–343. DOI: 10.1007/978-3-642-39320-4.

[14] Florian Rabe and Michael Kohlhase. "A Scalable Module System". In: *Information & Computation* 0.230 (2013), pp. 1–54. URL: http://kwarc.info/frabe/Research/mmt.pdf.