System Description: Statistical Parsing of Informalized Mizar Formulas

Cezary Kaliszyk
University of Innsbruck, Austria

Josef Urban, Jiří Vyskočil
Czech Technical University in Prague

***Abstract*—We describe a statistical system that learns parsing of ambiguous Mizar-like formulas from a large training corpus of aligned informal/formal formulas. We describe the methodology and the overall ideas, evaluate the performance of the system, and provide a public web interface for using the system.**

I. Introduction and Summary of the Statistical Parsing Approach

In this work we describe a system for statistical parsing of ambiguous Mizar-like [1] formulas, its implementation and evaluation. This is the next step in our larger project [12] of automatically formalizing informal mathematics by using statistical parsing methods and large-theory automated reasoning. The main components of this *autoformalization* approach were defined in [10], [11], and were used there on the Flyspeck corpus [5] based on the HOL Light [6] system. The general approach (applied here to parsing of Mizar-like formulas) is as follows:

1) Using corresponding (*aligned*) pairs of informal/formal formulas from a large corpus to train statistical disambiguation. While both informal (e.g. LaTeX) and formal math corpora are quite large today, there are not many formulas that would be written both informally and formally with a consistent alignment in mind. Therefore we produce the aligned pairs by *informalizing* (*ambiguating*) the formal corpus. In particular, for the formal Flyspeck corpus of about 20000 lemmas, we ambiguated the formal formulas by introducing overloaded symbols, forgetting types, brackets and casting functors [10]. The ambiguated formulas (strings of characters) and the correponding formal parse trees then provide a *treebank* [4] to train statistical parsing on.
2) Learning (augmented [10]) *probabilistic context-free grammar* (PCFG) [13] from the treebank.
3) Using the grammar in our CYK [22] chart parser which is modified by semantic checks and by more involved probabilistic (context-aware) processing [10], [11]. Fast discrimination trees [14] are used to match deeper subtrees from the treebank and to boost their probabilities when parsing new formulas.
4) Giving the top-ranking parse trees to type-checking and large-theory reasoning components [3], [2]. For Flyspeck we have used the HOL(y)Hammer system [8], [9].

II. Parsing Informalized Mizar

A distinctive feature of Mizar [1] is human-style and natural-language-like representation of formal mathematics. This includes Jaśkowski-style natural deduction [7], soft-typing Prolog-like mechanisms [21], propagating implicit knowledge using Mizar *adjectives* and *registrations* [19], hidden arguments, syntactic macros, and ubiquitous parametric and ad-hoc overloading [1]. This poses very interesting challenges, for example the symbol + is (re-)defined more than 100-times in the Mizar library (MML), see Table I. But it also takes our project closer to parsing true natural-language/LaTeX corpora such as ProofWiki.[1] In the following subsections we briefly describe the components of this work.

Table I: Mizar overloading: Some of the 143 (re-)definitions of symbol + in MML1147. The format is Article:func.SerialNumber.

AFF_4:func.3	ALGSTR_0:func.1	AMI_WSTD:func.4
AOFA_I00:func.16	AOFA_I00:func.29	AOFA_I00:func.33
AOFA_I00:func.7	AOFA_I00:func.85	ARYTM_0:func.1
ARYTM_2:func.5	ARYTM_2:func.7	ARYTM_3:func.9
...	...	...
VFUNCT_1:func.6	VFUNCT_2:func.3	XCMPLX_0:func.2
XXREAL_3:func.1	ZMODUL01:func.5	ZMODUL01:func.6

A. Treebank creation for Mizar

For Flyspeck we have ambiguated the parsed HOL Light formulas by several transformations applied on the formal HOL representation, creating the training trees in a form suitable for treebank learning. To create a suitable treebank for Mizar we apply transformations to the Mizar internal XML layer [18] used previously to produce both the user-level HTML representation[2] of the articles and also the *semantic* (MPTP [17], [20]) representation used by automated theorem provers (ATPs). Already the XML-to-HTML transformation is complex and sometimes imperfect. It needs to recover the user-level syntax from the internal XML representation and align the two.

Our new code for creating a Mizar treebank (producing now about 60000 parse trees from the Mizar Mathematical Library – MML version 1147) is based on the XML-to-HTML code, mainly modifying the hyperlinks into annotating nonterminals. As in the XML-to-HTML code, this

[1] https://proofwiki.org
[2] http://mizar.org/version/current/html/

annotation (alignment) is nontrivial and there are interesting issues described below.

Initially we have tried to directly use the semantic (*constructor* in the Mizar terminology) disambiguation layer, which is also used for theorem proving (both in Mizar and via the MPTP translation to TPTP [16]), and thus would be a suitable target for ATP experiments with the parsed statements. The first detailed evaluation of our statistical parser was done on this treebank. An example where this approach works well, connecting the user-level syntax directly with the semantic MPTP/TPTP layer is the following theorem `RCOMP_1:5`:[3]

```
for s,g being real number holds [.s,g.] is closed
```

which in TPTP becomes:

```
![A]: v1_xreal_0(A) => ! [B]: (v1_xreal_0(B) =>
          v2_rcomp_1(k1_rcomp_1(A, B)))
```

The internal Mizar XML representation is transformed to the parse tree shown in Fig. 1, which can be easily postprocessed into the TPTP format above.

While this direct use of the semantic (constructor) layer can provide a lot of disambiguation, there are several issues when connecting it with the user-level syntax. First, there are syntactic macros like Mizar *expandable modes* (types) [1]. These macros do not exist in the semantic layer and are expanded by the Mizar processing into larger collections of adjectives and types, taking various parameters from the context. For example the user-level type `Function of X,Y` is recursively expanded via the following macros (expandable modes):

```
mode Function of X,Y is quasi_total PartFunc of X,Y;
mode PartFunc of X,Y is Function-like Relation of X,Y;
mode Relation of X,Y is Subset of [:X,Y:];
mode Subset of X is Element of bool X;
```

This leads to the following semantic representation of `Function of X,Y` as:

```
quasi_total Function-like Element of bool [:X,Y:]
```

It is quite a nontrivial requirement for the statistical parser to go from the input string `Function of X,Y` to the above representation. The original symbol `Function` needs to be replaced by the (possibly parameterized) adjectives, and the type `Element of` is applied to a single argument `bool [:X,Y:]` consisting of two functions applied to the arguments X and Y. Furthermore, similar phenomena are often encountered also in other situations. For example, the function composition $\star$ changes the order of arguments (used generally for relation composition) on the user level:[4]

```
let f, g be Function; synonym g*f for f*g;
```

To deal with such phenomena we have in the second version of our export switched to the *syntactic* (*pattern* or *notation*) layer of Mizar [1], where our task is limited to the symbol disambiguation. This layer is going to be mapped to the semantic layer through a large number of "syntactic processing" Prolog-like rules, such as those needed for the above examples. For instance, the change of the function composition arguments can be encoded in TPTP as follows:

```
fof(dt_nk3_funct_1, axiom, (![A,B]:(((v1_relat_1(A)
& v1_funct_1(A)) & (v1_relat_1(B) & v1_funct_1(B)))
   => nk3_funct_1(B, A)=nk6_relat_1(B,A)))).
```

This says, that under appropriate type constraints (A and B being functions) the arguments of the syntactic pattern `nk3_funct_1` should be swapped to obtain the proper order of arguments of its parent pattern `nk6_relat_1`. This pattern may be again mapped to some parent syntactic pattern, or to the semantic (constructor) level. This means that the theorem-proving phase will either have to be preceded by a phase that processes (expands) these Horn-like rules, or the TPTP encoding of such rules will have to be added to the generated ATP problems. Our initial experiments show that both these approaches should be feasible. Our theorem is then represented as the parse tree in Fig. 2 (with the different nonterminals in bold), which is easily postprocessed into the following "syntactic TPTP":

```
![A]: ![B]: ( ( nm1_ordinal1(A) & nv1_xreal_0(A) &
nm1_ordinal1(B) & nv1_xreal_0(B) )
=> nv2_rcomp_1(nk1_rcomp_1(A,B)))
```

The parsing performance (Section III) takes some penalty when using a much higher number of the syntactic-level patterns rather than a smaller number of constructors, however there is still a lot of possibilities for improvement of the statistical parsing.

B. Mizar types

In the HOL setting used by Flyspeck, types are unique and do not intersect. This allows their simple use in the PCFG setting as "semantic categories" corresponding, e.g., to word senses when using PCFG for word-sense disambiguation. Such categories are useful for learning parsing rules from the treebank [11]. In Mizar, each term has in general many *adjectives* (soft types [21]) like `finite`, `natural`, `Function-like`, `non empty`, etc., which are computed during the type analysis. Only some of them are usually needed to allow a term to be an argument of a particular function or predicate. Since it is more involved to learn such complex typing rules statistically, in the first version we use only the top of the type hierarchy – the Mizar type `Set` – as the result type of all terms, and types occur only in quantification. This corresponds to the (untyped) MPTP encoding, where type guards (encoded as predicates) occur only in quantification, and type-hierarchy formulas delegate the type computation and checking to the ATP systems.

A problem with this approach is that allowing any term to be an argument of any function/predicate may lead to

[3] http://grid01.ciirc.cvut.cz/~mptp/1147/html/rcomp_1.html#T5

[4] http://grid01.ciirc.cvut.cz/~mptp/1147/html/funct_1.html#NK3

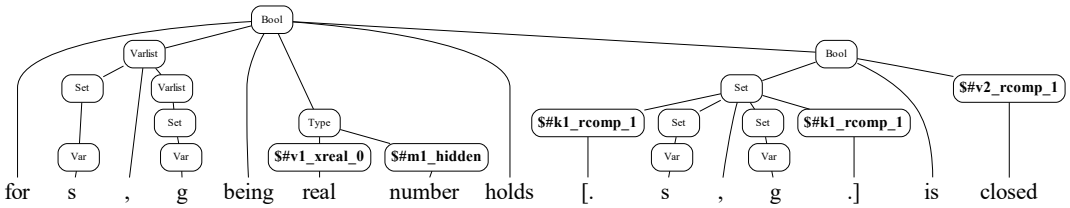

Figure 1: Parse tree corresponding to the internal (constructor-level) Mizar XML representation.

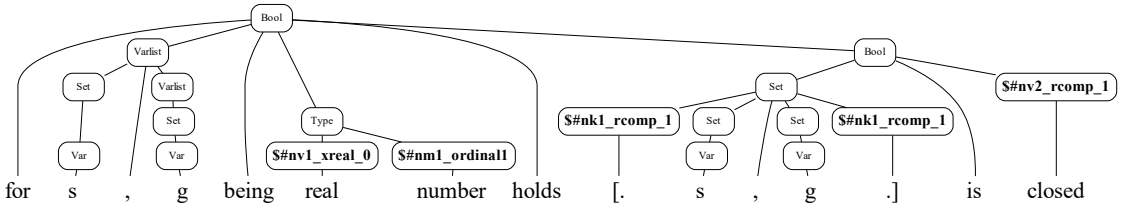

Figure 2: Parse tree corresponding to the syntactic (pattern-level) representation.

great proliferation of ill-typed terms during parsing. This was indeed an issue in our first untyped export of Flyspeck [10], [11] which used just raw HOL parse trees and only context-free parsing rules. It however seems that the recently introduced deeper (context-aware) parsing rules [11] quite significantly reduce such proliferation. When using a combination of subtrees of depth 4–8, the top-20 success rate (number of examples where the correct parse appears among the 20 most probable parses proposed by the parser) in parsing Mizar (100-fold cross-validation) is already around 60%, while it is only about 30% for the simple context-free approach (Table II).

III. Evaluation

The machine-learning evaluation for Mizar is done in the same 100-fold cross-validation scenario as for Flyspeck in [10], [11]. The evaluation is done both for the (simpler and imperfect) semantic (constructor-level) encoding, and for the more complex (but necessary) pattern-level encoding. In each case we create the disambiguated grammar trees and the corresponding ambiguous sentences from all (about 61400) toplevel MML theorems and definitions. We split them randomly into 100 equally sized chunks of about 614 trees and their corresponding sentences. The grammar trees serve for training and the ambiguous sentences for evaluation. For each testing chunk C_i ($i \in 1..100$) of 614 sentences we learn the probabilistic grammar P_i on the union of the remaining 99 chunks of grammar trees.

This can take considerable time (hundreds to thousands of minutes) for Mizar when using deeper subtrees for learning. There are roughly 100 million subtrees of depth 4–8 in the MML. The evaluation phase, i.e., the parsing of the remaining chunk is however typically fast, taking on average less than 1 second for each ambiguous sentence. The numbers of correctly parsed formulas and their average ranks across the several 100-fold cross-validations are shown in Table II. The relatively poor-performing context-free (subtree depth 2) method is evaluated only for the constructor-level encoding, in order to have a rough comparison with the performance on Flyspeck in [10], [11]. While there are still many ways how to improve the performance, the top-20, resp. top-1 numbers in the range of 60-64%, resp. 32-37% are very encouraging.

Parsing Method	top-20 success	avrg. rank	top-1 success
subtree depth 2 (constructor-level)	32.9%	4.6	13.0%
subtree depth 4-8 (constructor-level)	**63.7%**	**2.64**	**36.5%**
subtree depth 4-8 (pattern-level)	59.0%	2.74	32.0%

Table II: Evaluation on MML. The top-20 success is the number of examples where the correct parse appears among the 20 most probable parses returned by the parser.

IV. Online Parsing System

Similarly to the work done for Flyspeck, the parsing toolchain is deployed as an online service. Fig. 3 presents a screenshot of the system, it is available at:

http://grid01.ciirc.cvut.cz/~cek/parse_miz/

The service visualizes the overloading disambiguation as superscripts and further uses hyperlinking to the HTMLized MML. This allows Mizar users to write ambiguous formulas and see their most probable interpretations. This is similar to systems for "wikification" [15] of named entities in natural language texts from which our project takes some inspiration [12].

To make the probabilistic parsing sufficiently fast, we again limit the number of required parses to 20, and pre-

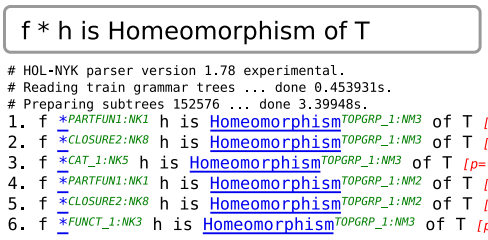

Figure 3: Screenshot of the online parsing system

select only the 1024 closest grammar trees for the grammar training. This is done by running a k-nearest neighbor (k-NN) filter using n-gram (unigram, bigram and trigram) representations of all Mizar theorems and definitions in their ambiguous form. The system thus takes on average 5 seconds for the complete processing of a query. Such processing includes the k-NN filtering, the grammar induction, the probabilistic parsing, and finally the HTML-ization.

The web interface shows several typical examples of queries, one of them being: "`f * h is Homeomorphism of T`". Note that the probability ($p = -22.05$) of `*` being partial function composition (`PARTFUN1:NK1`[5]) is much higher than it being its specialized case on many-sorted sets (`CLOSURE2:NK8`[6]) or a composition of category morphisms (`CAT_1:NK5`[7]). This is very likely thanks to the presence of the type `Homeomorphism (TOPGRP_1:NM3`[8]`)` in the context, because such combinations of symbols have been previously seen in theorems like `TOPGRP_1:31`[9]. Unlike in the service for Flyspeck, where all variables were internally alpha-normalized, the names of variables are taken into account when disambiguating Mizar. For example, using `f * g` in the above example instead of `f * h` yields even higher probability for function composition, likely because of even greater similarity to theorem `TOPGRP_1:31`.

While such improvements pose additional challenges to the parsing system, we believe that addition of features like this and of other natural-language-like Mizar mechanisms takes our work significantly closer to parsing human-level mathematics written in LaTeX. The immediate future work in this line of research includes parsing of the human-like Mizar proofs, and using automated theorem provers for full semantic understanding.

References

[1] G. Bancerek, et al. Mizar: State-of-the-art and beyond. In M. Kerber, J. Carette, C. Kaliszyk, F. Rabe, and V. Sorge, editors, *Intelligent Computer Mathematics, CICM 2015*, volume 9150 of *LNCS*, pages 261–279. Springer, 2015.

[5] http://grid01.ciirc.cvut.cz/~mptp/1147/html/partfun1.html#NK1
[6] http://grid01.ciirc.cvut.cz/~mptp/1147/html/closure2.html#NK8
[7] http://grid01.ciirc.cvut.cz/~mptp/1147/html/cat_1.html#NK5
[8] http://grid01.ciirc.cvut.cz/~mptp/1147/html/topgrp_1.html#NM3
[9] http://grid01.ciirc.cvut.cz/~mptp/1147/html/topgrp_1.html#T31

[2] J. C. Blanchette, D. Greenaway, C. Kaliszyk, D. Kühlwein, and J. Urban. A learning-based fact selector for Isabelle/HOL. *J. Autom. Reasoning*, 57(3):219–244, 2016.

[3] J. C. Blanchette, C. Kaliszyk, L. C. Paulson, and J. Urban. Hammering towards QED. *J. Formalized Reasoning*, 9(1):101–148, 2016.

[4] A. Clark, C. Fox, and S. Lappin, editors. *The Handbook of Computational Linguistics and Natural Language Processing*. Wiley-Blackwell, 2010.

[5] T. C. Hales, et al. A formal proof of the Kepler conjecture. *Forum of Mathematics, Pi*, 5, 2017.

[6] J. Harrison. HOL Light: A tutorial introduction. In M. K. Srivas and A. J. Camilleri, editors, *FMCAD*, volume 1166 of *LNCS*, pages 265–269. Springer, 1996.

[7] S. Jaśkowski. On the rules of suppositions. *Studia Logica*, 1, 1934.

[8] C. Kaliszyk and J. Urban. HOL(y)Hammer: Online ATP service for HOL Light. *Mathematics in Computer Science*, 9(1):5–22, 2015.

[9] C. Kaliszyk and J. Urban. Learning-assisted theorem proving with millions of lemmas. *J. Symb. Computation*, 69:109–128, 2015.

[10] C. Kaliszyk, J. Urban, and J. Vyskočil. Learning to parse on aligned corpora. In C. Urban and X. Zhang, editors, *ITP*, volume 9236 of *LNCS*, pages 227–233. Springer, 2015.

[11] C. Kaliszyk, J. Urban, and J. Vyskočil. Automating formalization by statistical and semantic parsing of mathematics. 2017. ITP 2017, invited paper. To appear.

[12] C. Kaliszyk, J. Urban, J. Vyskočil, and H. Geuvers. Developing corpus-based translation methods between informal and formal mathematics: project description. In S.M. Watt, J.H. Davenport, A.P. Sexton, P. Sojka, and J. Urban, editors, *Intelligent Computer Mathematics, CICM 2014*, volume 8543 of *LNCS*, pages 435–439. Springer, 2014.

[13] K. Lari and S. Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech & Language*, 4(1):35 – 56, 1990.

[14] W. McCune. Experiments with discrimination-tree indexing and path indexing for term retrieval. *Journal of Automated Reasoning*, 9(2):147–167, 1992.

[15] L. Ratinov, D. Roth, D. Downey, and M. Anderson. Local and global algorithms for disambiguation to Wikipedia. In D. Lin, Y. Matsumoto, and R. Mihalcea, editors, *The 49th Annual Meeting of ACL: Human Language Technologies, Proceedings of the Conference*, pages 1375–1384. ACL, 2011.

[16] G. Sutcliffe. The TPTP world - infrastructure for automated reasoning. In E. M. Clarke and A. Voronkov, editors, *LPAR (Dakar)*, volume 6355 of *LNCS*, pages 1–12. Springer, 2010.

[17] J. Urban. MPTP - Motivation, Implementation, First Experiments. *J. Autom. Reasoning*, 33(3-4):319–339, 2004.

[18] J. Urban. XML-izing Mizar: Making semantic processing and presentation of MML easy. In M. Kohlhase, editor, *MKM*, volume 3863 of *LNCS*, pages 346–360. Springer, 2005.

[19] J. Urban. MoMM - fast interreduction and retrieval in large libraries of formalized mathematics. *Int. J. on Artificial Intelligence Tools*, 15(1):109–130, 2006.

[20] J. Urban. MPTP 0.2: Design, implementation, and initial experiments. *J. Autom. Reasoning*, 37(1-2):21–43, 2006.

[21] F. Wiedijk. Mizar's soft type system. In K. Schneider and J. Brandt, editors, *TPHOLs 2007*, volume 4732 of *LNCS*, pages 383–399. Springer, 2007.

[22] D. H. Younger. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208, 1967.