Automatic translation in Formalized Mathematics

*Grzegorz Bancerek**

Dept. of Information Engineering, Shinshu University, Nagano, Japan

Institute of Computer Science, Białystok Technical University, Białystok, Poland

bancerek@mizar.org

Abstract - The paper describes the automatic translation process used in producing the journal of *Formalized Mathematics (a computer assisted approach)*. We report the current state of the implementation (aoki version) as well as the improvements and future directions planned for the development of the process. In the description we concentrate on the part of translation which is dependent on the author of the Mizar article. By indicating the symbols and Mizar formulae are translated, the readability of the output can be improved.

Keywords - formalized mathematics, automatic translation, text generation.

1 Introduction

Formalized Mathematics (a computer assisted approach) (FM) publishes the contents of Mizar articles (see [Bon]) submitted to the Mizar Mathematical Library (MML). The first step is to translate the contents into English and the translation is made automatically. The material published concerns the surface part of Mizar articles. Proofs are not yet translated but other elements - theorems, definitions, schemes, and reservations and global sets if necessary - are.

The process of translation is mechanized (see [B&C 1991, B&C 1992, Try]) but the final result depends on the author/editor. This dependence appears on several levels:

- Summary and division into titled sections. This is not an element of translation process but the text translated is put into such a structure.

*This paper was written when the author visited Shinshu University as a JSPS Fellow since Sept. 2000.

So, it may have some impact on homogeneity and transparency of the final result.

- Variables and reservations. Single letter variables are preserved and others are abbreviated into single letters with indices. Reservations are translated in the place where they appear. So, the author of a Mizar article must be careful about the distance between a reservation of the variable and its use, which has impact on readability.

- Formulae formulation. The process of translation works in the following paradigm:

  > use a general pattern, but for exceptions use specific patterns as well as it is not an exception from exception, then use more specific pattern ...

  The formulation of theorems, definientia, and schemes in an appropriate manner may make use of those (more) specific patterns which were prepared to improve the translation.

- Existing formats. When an existing format is used to denote a new concept, it is translated according to the existing pattern for this format. In some situations it may differ from the author's intentions.

- New symbols and formats. When a new format (with a new symbol or an old one) is used to denote a new concept, the pattern of translation for this format is generated. The generation depends on the symbol itself and the format. For more details see Section 8. The author of a Mizar article or the editor may change generated patterns according to their needs, but it is (usually) fixed after publication. The description and impact of translation patterns is given in Section 4 and the sections following it.

According to the above items the author of a Mizar article is advised to pay attention to the following checkpoints when previewing[1] the article: definitions of new concepts, variable representations, parentheses in expressions with new functors, types and registrations with new attributes, and formulae with new predicates or attributes.

2 Historical notes

The first attempt to increase readibility of Mizar articles/abstracts was done by Piotr Rudnicki and Andrzej Trybulec in 1989 (see [R&T]).

The journal of *Formalized Mathematics* was established by the Association of Mizar Users (SUM, an acronym taken from the to Polish name: *Stowarzyszenie Użytkowników Mizara*) in 1990. The first 4 volumes (1990–93) were published by Université Catholique de Louvain under the agreement of SUM and the

[1] A previewing service is available under URL: `http://megrez.mizar.org/fm/preview.html`

Fondation Philippe le Hodey (copyrights), ISSN 0777-4028. From the 5th volume, *Formalized Mathematics* was published by Warsaw University, Białystok Branch, ISSN 1426-2630. After the transformation of Białystok Branch into University of Białystok in 1997, the letter became the publisher (with the same ISSN number).

The first issue of the journal was experimental and the contents did not differ much from the Mizar abstracts (see [FM2]). Type-setting programs and scripts were implemented by Czesław Byliński, Wojciech Leończuk, Michał Muzalewski, Krzysztof Prażmowski, Andrzej Trybulec, and the author of this paper.

The second issue was produced to be much more similar to mathematical papers written in English (see [Mat]). The sentence structure of Mizar formulae (quantifiers, conjunctions, and some atomic formulae) and types were translated into English. The remaining elements were expressed with mathematical symbolisms available in TEX. Type-setting programs for this issue were implemented by Andrzej Trybulec and Czesław Byliński with assistance from the author of this paper. The maintenance of the translation system was entrusted to the author.

A new implementation of the Mizar system and new features in the Mizar language concerning attributes as well as a need for improvement in the quality of translation enforced redesign and re-implementation of the translation system in object oriented style. This was done and is still continuously developed by the author.

In 1996, an electronic extension of *Formalized Mathematics* was established[2]. In contrast to the paper edition, the electronic extension is dynamic and can follow changes in the Mizar system and in the MML as well as reflect the improvements to the translation process. As a result, there are translations of Mizar articles updated according to their current form in the MML.

3 The process - technical details

The process of translation may be viewed as a rewriting system (see [Ban]). It takes a Mizar article and reduces it to some abstarct form. Next, the analyzing rules are applied to the abstract form. After that translation patterns for formulae and formats are used. Finally, the filling text, the summary, and the section titles are added.

These steps and the management of translation patterns are realized by the following programs.

[2]URL address: `http://mizar.org/JFM/`

Program	param.	input files	output files
accom	$1	$1.miz, PREL	ACCOM($1)
fmparse	$1	$1.miz, ACCOM($1)	$1.fma, $1.nfr
newfmfrm	$1 -l$2	$2.frd, ACCOM($1), PREL($1)	$1.fmn, $1.fmd
addfmfrm	$1 -l$2	$2.frd, $1.fmn	$2.$-$
fmfrm	$1 -l$2	$2.frd, $1.nfr, $1.fmn, ACCOM($1)	$1.fmf, $1.fmz
resvar	$1	$1.miz, ACCOM($1)	$1.ire
fmnotats	$1	PREL($1)	$1.fms
fmanalyz	$1	$1.fma, $1.fmf, $1.ire, ACCOM($1)	$1.?
jformath	$3 [/d|/f]	[formath.set,] $3.lar, ∗.bnt, ∗.fms, ∗.?	$3.tex
latex	$3	$3.tex, ∗.?, formath.cls, fom10.clo, mizarfrm.tex [, ∗.bbl]	$3.dvi, $3.aux

In the table above, $1 stands for the name of Mizar article, $2 stands for the name of a database file "$2.frd" with translation patterns, and $3 stands for the name of a list of Mizar articles (usually a code for the journal issue) in the file "$3.lar". The question marks in "$1.?" and "∗.?" stands for the section number of the Mizar article and ∗ indicates the names of Mizar articles from the list in file $3.lar. ACCOM($1), PREL($1), and PREL stand for Mizar internal format files created by the accom program or available from Mizar directories.

The contents of the files are as follows:

$1.fma - an abstract description of the surface part of article $1.miz (reservations, definitions, theorems, schemes, and global sets).

$1.nfr - new formats introduced in article $1.miz.

$1.fmd - generated translation patterns (with identification) of old formats existing in $2.frd which are used in definitions in article $1.miz.

$1.fmn - generated (or re-edited by the editor or the author of an article) translation patterns (with identification) of new formats from article $1.miz (not yet introduced into $2.frd).

$2.frd - database file of translation patterns (with identification).

$1.fmf - translation patterns (without identification) of formats ordered according to $1.frm from ACCOM($1) and $1.nfr.

$1.fmz - format identifications ordered according to $1.frm from ACCOM($1) and $1.nfr.

$1.ire - information for reserved variables if they are used in elements translated.

$1.fms - (signature) list of articles introducing notation (constructors) used in translated elements from article $1.miz.

$1.?, ∗.? - final LaTeX input including a translation of the section nr ? from article $1.miz (or ∗.miz).

$3.lar - list of article names.

$1.bnt - bibliographic notes of the article, the summary, and the section titles.

formath.set - basic information of the publication issue.

formath.cls, fom10.clo, mizarfrm.tex - LaTeX style files.

4 Translation patterns

The translation on the level of atomic formulae, terms, and types is based on *translation patterns* for formats. A format is a vocabulary symbol with an arity. For example,

`- 1 1`	for subtraction ($x - y$) of reals, complex numbers, vectors, ...
`- 0 1`	for opposite ($-x$) real, complex number, vector, ...
`set 0`	for the type *set*,
`Function 2`	for a function from a set (first argument) into a set (second argument),
`in 1 1`	for the membership relation ($x \in y$),
`[ ] 2`	for an ordered pair ($\langle x, y\rangle$).

There may be more than one format with the same symbol taking different number of arguments. In format descriptions we take into consideration only visible arguments and do not consider their types or forms. The subtraction of vectors has the same format as the subtraction of reals although it has one extra (invisible) argument - the vector space.

Translation patterns for formats are grouped in the files: \$1.fmn, \$1.fmd, and \$2.frd, according to the vocabularies from which the format symbol comes. Each pattern is described by 3 or 4 lines. The first two (three, in the case of a bracket format) include the identification of the format. Specifically, the first line describes the format as: the format kind (one of the letters: `O K G J U R M L V`), the symbol number, and the (left, right) argument numbers. For a bracket functor (format kind `K`), the vocabulary and the number of the closing bracket are also given. The second line includes the symbol kind and the symbol itself. For a bracket functor, the symbol of the closing bracket is given in the third line.

```
#HIDDEN
K1 2 L1 vHIDDEN                    M1 0          R1 1 1
K[                                 Mset          R=
L]
```

The third line (fourth, in the case of a bracket format) includes the control header and the proper pattern. For a selector functor (kind `U`) the fourth line includes the complemented pattern of translation of the associated field. For predicates (kind `R`) this line will include the complemented pattern of translation of the negative form. For types (`M` and `L`) this will be a description of the complemented pattern of translation of the plural form. The control header should not be repeated with the complemented patterns.

```
                                   ha set #0     m #1 = #2
mc#1#2; \langle #1, #2\rangle      sets #0       #1 \neq #2
```

The files \$1.fmf include translation patterns without identification, i.e., the control headers with the proper patterns and the complemented patterns, if necessary. The `#1`, `#2`, ... in patterns indicate the places of arguments. The `#0` in a translation pattern for a type indicates the place of qualified variables. For example, the pattern

```
function #0 from #1 into #2
```

for the formula

```
for f being Function of A,B holds ...
```

renders

for every function f from A into B ...

in most contexts.

A control header is a sequence of letters, digits, and other characters followed by a space. The formal grammar of the control headers is given in Appendix A. Terms for the grammar given in this text are written in `typwriter` font.

According to the information carried by the control headers we may divide all patterns into 4 categories:

1. Patterns for **functors**: `O` - proper functor, `K` - bracket functor, `G` - aggregate functor, `J` - forgetful functor (uses a symbol of kind `G`), and `U` - selector functor and field name.
2. Patterns for **predicates** (letter `R`).
3. Patterns for **types**: `M` - proper type, `L` - structure type (uses a symbol of kind `G`).
4. Patterns for **attributes** (letter `V`).

The control headers determine the behavior of the translator when a given format is translated. For example, patterns are intended to appear in verbal or symbolic mode. The control options (`TeX-mode` for functors and types and `Predicate-kind` for predicates) determine the destination of the pattern. Namely, `Math-mode` and `Symbolic` send the patterns into symbolic mode (i.e., they are put between single $'s in TEX) and all others are put into verbal mode (i.e., no $'s are used). Adjectives are always verbal. In more complicated cases, when verbal patterns appear as arguments of symbolic patterns, a special technique of closing and opening the TEX math-mode inside a formula are used. The goal of these techniques is to preserve hyphenation in verbal mode and produce well balanced spacing in symbolic mode.

When a functor with a verbal pattern is translated, the word 'the' may be written before the pattern. This will happen for patterns with a small 'h' designating `Horizontal-mode`. A capital 'H' will cause translation without 'the'. For examples, see [FM6, Def. 15] and [FM3, Def. 9] which use the following patterns.

```
#ALG_1
O5 0 2
ONat_Hom
hol; natural homomorphism of #1 w.r.t. #2
#TSP_2
O1 0 2
OStone-retraction
Hosl(2)#1#2; Stone-retraction of #1 onto #2
```

Articles (a, an) in verbal phrases are determined by the `Article` or `Adjective-kind` control option from the pattern which is put first in the phrase. In cases of `HAS-`, `SAT-`, and `Noun-adjective` articles should be written in the proper pattern as necessary.

5 Parentheses

The parentheses around terms are put in the translation process independently from those given in a Mizar article. The parentheses in a Mizar article are used to get correct identification according to the state of priorities which may differ from mathematical practice. Moreover, it vary on the authors.

The translation process to calculate the parentheses uses the following header options (from patterns for functors):

`Bracket-ability` The expression built by the pattern may be taken into brackets if the pattern is `Open`. Otherwise, it does not need extra parentheses.

`Operation-kind` This option determines the operation kind of the pattern and argument contexts. The context is the position of an argument w.r.t. the operation symbol and is indicated by one of the following operation kinds: `l k m r q w t b c`. The denotation of these letters appears in the grammar under `Operation-kind` on page 15). Moreover, the context includes the priority of the pattern/operation. The given operation kind is assigned by default as a context for argument 1 unless the operation is of an infix kind. Then the context of argument 1 is a postfix operation and the context of argument 2 is a prefix operation (with given or standard priority). If a context should be assigned to other arguments, it may be indicated in brackets after the letter of the operation kind (e.g., "i(2,1)").

`Prefix`	$\mathrm{oper}\,x$	`Upper-prefix`	${}^{\mathrm{oper}}x$
`Lower-prefix`	${}_{\mathrm{oper}}x$	`Postfix`	$x\,\mathrm{oper}$
`Upper-postfix`	x^{oper}	`Lower-postfix`	x_{oper}
`Overfix`	$\overline{x}$	`Underfix`	$\underline{x}$
`Infix`	$x_1 \circ x_2$	`Non-associative-infix`	$x_1 \star x_2$
`Right-associative-infix`	$x_1 \Rightarrow x_2$	`Circumfix`	$\sqrt{x}$
`Other-operation`	$\frac{x_1}{x_2}$		

`Forcing` If the pattern should be taken into parentheses in other contexts than resulting from the operation kind, then all needed operation kinds may be listed in this option. For example, if $\sqrt{x}$ is an argument of `Lower-postfix` (`w`), the brackets should be put around it in: $(\sqrt{x})_*$. According to this, the control header and the proper pattern for `#1-root #2` are as follows:

```
moc{w}@9#1#2; \sqrt[#1]{#2}
```

The standard forcing forces parentheses in situations: ${}^{\mathrm{oper}_1}x^{\mathrm{oper}_2}$, ${}^{\mathrm{oper}_1}x_{\mathrm{oper}_2}$, ${}_{\mathrm{oper}_1}x^{\mathrm{oper}_2}$, ${}_{\mathrm{oper}_1}x_{\mathrm{oper}_2}$, $\overline{x}_{\mathrm{oper}}$, ${}_{\mathrm{oper}}\overline{x}$, $\underline{x}^{\mathrm{oper}}$, ${}^{\mathrm{oper}}\underline{x}$. In the first situation, we get $({}^{\mathrm{oper}_1}x)^{\mathrm{oper}_2}$ or ${}^{\mathrm{oper}_1}(x^{\mathrm{oper}_2})$ dependently from which of oper_1 or oper_2 is the main operation and which the suboperation.

`Priority` This option classifies operations into 4 groups: weak (0, 1), additive (2–4), multiplicative (5–7), and strong (8, 9). If a weaker operation is an argument of a stronger one, it indicates a possible argument to be taken into parentheses. If this option is omitted, the standard priority is assigned as follows:

`l`	`Prefix`	*strong* if `Math-mode` else *weak additive*
`k`	`Upper-prefix`	*strong multiplicative*
`m`	`Lower-prefix`	*strong multiplicative*
`r`	`Postfix`	*strong*
`q`	`Upper-postfix`	*strong*
`w`	`Lower-postfix`	*strong*
`t`	`Overfix`	*very strong*
`b`	`Underfix`	*very strong*
`i`	`Infix`	*additive*
`x`	`Not-associative-infix`	*additive*
`y`	`Right-associative-infix`	*additive*
`c`	`Circumfix`	*very string*
	`Other-operation`	*weak*

`Argument-context` If arguments should be treated differently from the standard given by `Operation-kind` or `Priority`, then the correct treatment may be given by this header option. For example, the translation pattern for `Funcs(#1, #2)` (the set of all functions from #1 into #2, see [FM2]) is as follows

```
moq(2)/1m/2q#1; #2^{#1}
```

Unfortunately, in the first issue of the journal the Mizar text was not translated, so the impact of the header cannot be observed there. But, it is possible to see it in later issues, e.g., in [FM1, (1)–(5)]. An example of another pattern is

```
moi(1,3)@0/1w@9#2; #1_{#2}{:=} #3
```

which is used in [FM9, Def. 20, (51)].

`List-of-bracket-disabled-arguments` The arguments sepecified in the list may not be taken into parentheses.

The main rule is to use as few brackets as possible without loss of meaning of an expression. For example, in the expression "`(A \/ B) \/ C`" the brackets are removed to produce: $A \cup B \cup C$ unless the exception below (associativity rule) holds. A symmetric convention for `Right-associative-infix` is applied. In situations "$\mathrm{oper}_1\,\mathrm{oper}_2\,x$" and "$x\,\mathrm{oper}_2\,\mathrm{oper}_1$" the parentheses are forced if the priority in the context of the argument of oper_1 is much stronger than the priority of the suboperation oper_2. But, brackets are always put in expressions like $(x^{\mathrm{oper}})^{\mathrm{oper}}$, $(x_{\mathrm{oper}})_{\mathrm{oper}}$, ${}^{\mathrm{oper}}({}^{\mathrm{oper}}x)$, and ${}_{\mathrm{oper}}({}_{\mathrm{oper}}x)$. In cases which are not listed in forcing lists, the priorities from a context and suboperation indicate if parentheses should be applied.

There are two extra rules: symmetry and associativity. The symmetry rule is used when both arguments in an infix operation are of the same kind. Then both are taken into brackets if one of them should be. Examples may be found in [FM8, Def. 2 and Def. 3]. In the first example, the expression

(the carrier of S_1) $\cup$ the carrier of S_2

was changed to

(the carrier of S_1) $\cup$ (the carrier of S_2)

The associativity rule is used when variants of associativity law are presented, like $(x+y)+z = x+(y+z)$. Then all brackets are introduced although in the standard way the grouping to the left (or to the right in case of `Right-associative-infix`) is assumed. For examples, see [FM5, (61), p. 315], [FM8, (10) and (27)], or [FM7, (26) and (34)].

6 Multipredicates

The direct translation of a Mizar formula is preceded by the analyzing precedure which may change the structure of the formula. These changes concern only the order of conjuncts and adjectives. They do not change the meaning of the formula but serve to improve readability.

To explain the analyzing process we must first introduce some terminology. Mizar atomic formulae fall into 3 groups: predicate formulae, qualifying formulae (e.g., x is a function), and adjective formulae (e.g., x is non empty). The last two always have verbal patterns

"`#1 is #2`" or "`#1 has #2`" or "`#1 satisfies #2`"

with a distinct subject. Predicate formulae may have symbolic or verbal patterns. Some of these verbal patterns: `IS-`, `HAS-`, and `SAT-phrase`, have distinct subjects. Atomic formulae with distinct subjects are called *subjected phrases*. Subjected phrases fall into 3 categories[3]:

IS predicates with the `IS-phrases` control option, all qualifying formulae, and adjective formulae with attributes with `A-`, `AN-`, `Empty-`, or `Noun-adjective` control option,

HAS predicates with the `HAS-phrases` control option and adjective formulae with attributes with the `HAS-adjective` control option,

SAT predicates with the `SAT-phrases` control option and adjective formulae with attributes with the `SAT-adjective` control option.

The first step in analizing a conjunction is the grouping of subjected phrases with the same subject into multipredicates. A multipredicate includes 3 collections - one for each category of subjected phrases. The translation pattern for a multipredicate depends on the quantities of those collections and on whether it is negated or not. If all collections are non empty and there is no negation, then it will look like

[3] An increase in the number of subjected phrase categories to include phrases "`#1 preserves #2`", "`#1 inherits #2`", ... is planned

```
#1 is <IS-collection>, has <HAS-collection>,
          and satisfies <SAT-collection>
```

If a multipredicate is quantified by a variable which is the subject of the multipredicate, the ellipsis is used. In such situations the following patterns are applied.

```
[not] for x holds M(x)          => [not] every $x$ <M>
[not] for x being T holds M(x)  => [not] every <T> <M>
[not] ex x st M(x)              => there exists [no] $x$ which <M>
[not] ex x being T st M(x)      => there exists [no] <a T> which <M>
```

The letter M stands for a multipredicate and T for a type. However, <M>,<T>, and <a T> stand for the translation of the multipredicate, the type, and the type with an article. For example, see [FM5, Def. 18] where

```
for I being Instruction of S holds I is IC-good;
```

was translated into

Every instruction of S is IC-good.

Also, in theorem [FM1, (41)] the formula

```
for X being non empty set, S,T being non empty Poset
ex F being map of UPS(S, T|^X), UPS(S, T)|^X
 st F is commuting isomorphic
```

was translated into

> For every non empty set X and all non empty posets S, T holds there exists a map from $\mathrm{UPS}(S, T^X)$ into $\mathrm{UPS}(S, T)^X$ which is commuting and isomorphic.

The similar processing is used in the translation of types with adjectives and in the translation of registrations. The adjectives are grouped in three collections: HAS, SAT, and others. The translation patterns for types with adjective also depends on quantities of the collections. If all collections are non empty, the pattern looks like

```
[a|an] <others-adjectives> <type> with <HAS-adjectives>
          and satisfying <SAT-adjectives>
```

It was used in [FM4, (37)] for

```
for N being Hausdorff topological_semilattice
            with_open_semilattices Lawson (complete TopLattice)
 holds N is with_compact_semilattices
```

and shouldbe rendered[4] as

> Every Hausdorff Lawson complete top-lattice with open semilattices and satisfying conditions of topological semilattice has compact semillatices.

[4] In the published version of this example, the previuos implementation did not include the word *and* before *satisfying*.

This special processing causes a problem when a type with adjectives appears in a multipredicate in a qualifying formula.

7 Definition style and validity

The control options Definition-style, Predicate-validity, and Type-validity are used when translating definitions introducing appropriate concepts. Definition-style concerns functor definitions

```
func F -> specification (means|equals) ...
```

and is applied as follows:

header	pattern
A- or AN-singular-functor	
hos, hon	The <F> is <a specification> and is defined ...
mos, mon	The functor <F> is <a specification> and is defined ...
Plural-functor	
hop	The <F> constitute <a specification> defined ...
Normal-functor	
ho	The <F> yields <a specification> and is defined ...
mo	The functor <F> yields <a specification> and is defined ...

For example in [?, Def. 17] it is "The open neighborhoods of p constitute a relational structure defined by ...".

When in Mizar article there is introduced an antonym which is formed in standard way (e.g., #1 does not overlap #2 as an antonym for #1 overlaps #2) the definition of it do not need to be given in the translation. The value Predicate-repeated of the control option Predicate-validity is designed for such and similar situations. Similarly, if in Mizar article there is a synonym of a type which gives the plural form of it, it do not need to be mentioned in the translation. The value Type-repeated in Type-validity is designed for such situations.

8 Conventions in pattern generation

Generated patterns for aggregates, brackets, and proper functors have Math-mode option by default. Other generated patterns are assumed to be verbal phrases. This classification may be changed for specific patterns unless they are for types, attributes, and field names which should always be verbal phrases. To generate a new translation pattern for verbal phrases the following operations on the symbol are applied:

- replacement of all substrings of the form "xX" (a letter followed by a capital letter) by "x X",

- replacement of underscore characters by spaces,
- conversion of all letters into lower case.

For example, the "is_FreeGen_of" symbol from the MML is converted into the phrase "is free gen of".

If the generation concerns an attribute format, then the following rules are applied.

symbol (converted)	pattern generated
being <noun>	b <noun>
having <noun>	h <noun>
with <noun>	x <noun>
satisfying <noun>	s <noun>
<adjective>	a <adjective> or n <adjective>

If the generation concerns a predicate format, then the following rules are applied. The first column indicates the number of left arguments.

	symbol (converted)	pattern generated
1	is <phrase>	i <phrase>
1	has <phrase>	j <phrase>
1	satisfies <phrase>	s <phrase>
2	are <phrase>	h #1 and #2 are <phrase> #1 and #2 are not <phrase>
2	are not <phrase>	h #1 and #2 are not <phrase> #1 and #2 are <phrase>
	<phrase>	h <phrase> not <phrase>

Symbolic patterns for predicates as well as verbal patterns for proper functors are not recognized so the author of a Mizar article should be sure to give the correct control header in such cases. The generation rules of patterns for functors are given in the table below.

format	pattern generated
#ORDERS_1, G1 2, GRelStr	mc#1#2; \langle #1,#2 \rangle
#ORDERS_1, J1 1, GRelStr	hol#1; rel str of #1
#ORDERS_1, U1 1, UInternalRel	hosl#1; internal rel of #1
#HIDDEN, K1 2 L1 vHIDDEN, K[, L]	mc#1#2; [#1, #2]
#CARD_4, O1 0 1, Ocf #FUNCT_2, O1 0 2, OFuncs	mol; \mathop{\rm cf} #1 mol#1#2; \mathop{\rm Funcs}(#1, #2)
#AMI_3, O2 1 1, O:= #AMI_3, O2 2 1, O:=	moi; #1 \mathop{\rm :=} #2 moi#1#2; (#1, #2) \mathop{\rm :=} #2
#OPPCAT_1, O1 1 0, Oopp	mor; #1 \mathop{\rm opp}

There is no special processing for type patterns.

9 Further work

The most recent plans concern the improvement of the quality of articles published. It includes the following items:

- variable representation,
- automatic reservations and sets,
- new categories for subjected phrases and more detailed analysis of the order of adjectives in each category,
- a richer database of translation patterns for formulae,
- variations of translation patterns for formats.

Plans for the long term include the translation of proofs. This will be done in several steps. First, the references from proofs will be extracted and after some processing they will be translated. The next step will consist of a shallow translation of the proof (with the extraction applied for subproofs). Finally, some pruning of the proofs will be done before the shallow translation with extraction. The pruning will apply statistic analysis and automated subject classification procedure for the MML. The translation of all proofs in whole is not planned.

The automatic generation of preliminaries for an article and each section outlined in [B&C 1992] are still not realized. This work is also dependent on statistic analysis of the notation and terminology used and the theorems referred to, and should take into consideration the subject classification automatically developed for the MML.

On the other hand, the translation system cannot work well without direct contact with the authors of Mizar articles. One aspect that needs to be cleared is the possibility of involving authors in the previewing of translations before submission. Another advantageous aspect of such contact with authors concerns the impact authors will have on the development of a rich collection of translation patterns. The previewing process must allow the author to test interactively all possibilities of the translation system (proof-reading without the mediation of other persons). A second role of the previewing step is to speed up the publication process. Currently, the process of submission to the MML and the proof-reading process for publication in FM are separated (by a quite long time interval dependent on the Editor of FM). It causes some additional problems: the Mizar article may change, there may not be necessary feedback on the Mizar article, and the author may lose interest.

In the plans connected with those aspects there is the establishment of the web site: `fm.mizar.org`. It, among the FM-related things, will give direct access to the translation system as well as enable the download of the system. The previewer and proof-reader are already available at `http://megrez.mizar.org/fm/`.

As a by-product the web site will include aid tools for mathematical publications.

Acknowledgments: The author would like to thank Roman Matuszewski, the Editor of FM, for his long of 10 years cooperation in the maintenance of the journal and communication with authors of Mizar articles. The author would also like to thank Adam Grabowski, who maintain the electronic extension of FM, for his great work done in synchronizing translation patterns for the paper and electronic editions of the journal.

The author also expresses his gratitude to Pauline N. Kawamoto and Piotr Rudnicki for their help during preparation of this paper.

References

[Ban] Grzegorz Bancerek. Reduction Relations. *Formalized Mathematics*, 5(4):469–478, 1996.

[B&C 1991] Grzegorz Bancerek, Patricia L. Carlson. Semi automated translation for mathematics. In J. Darski and Z. Vetulani, editors, Sprache-Kommunikation-Informatik, pages 131-136, Akten des 26. Linguistischen Kolloquiums, Niemeyer, Poznan, 1991.

[B&C 1992] Grzegorz Bancerek, Patricia L. Carlson. Mizar and machine translation for mathematics documents. Available under URL: `http://www.mizar.org/project/banc_carl93.ps`.

[Bon] Ewa Bonarska. An Introduction to PC Mizar. Fondation Philippe le Hodey, 1990.

[Mat] Roman Matuszewski. Preface. *Formalized Mathematics*, 1(2):255, 1990.

[R&T] Piotr Rudnicki, Andrzej Trybulec. *A Collection of TeXed Mizar Abstracts*. Technical Report TR 89-18, University of Alberta, 1989. Available at `http://mizar.org/project/TR-89-18.ps`.

[Try] Andrzej Trybulec. Introduction. *Formalized Mathematics*, 1(1):7–8, 1990.

Examples

[FM1] Grzegorz Bancerek, Adam Naumowicz. Function spaces in the category of directed suprema preserving maps. *Formalized Mathematics*, 9(1):171–177, 2001.

[FM2] Czesław Byliński. Functions from a set to a set. *Formalized Mathematics*, 1(1):153–164, 1990.

[FM3] Zbigniew Karno. Maximal Kolmogorov subspaces of a topological space as Stone retracts of the ambient space. *Formalized Mathematics*, 5(1):125–130, 1996.

[FM4] Artur Korniłowicz. Meet continuous lattices revisited. *Formalized Mathematics*, 9(2):249–254, 2001.

[FM5] Artur Korniłowicz. On the composition of macro instructions of standard computers. *Formalized Mathematics*, 9(2):303–316, 2001.

[FM6] Małgorzata Korolkiewicz. Homomorphisms of algebras. Quotient universal algebra. *Formalized Mathematics*, 4(1):109–113, 1993.

[FM7] Robert Milewski. The ring of polynoms. *Formalized Mathematics*, 9(2):339–346, 2001.

[FM8] Yatsuka Nakamura, Grzegorz Bancerek. Combining of circuits. *Formalized Mathematics*, 5(2):283–295, 1996.

[FM9] Andrzej Trybulec, Yatsuka Nakamura, Piotr Rudnicki. The $\mathbf{SCM}_{\mathrm{FSA}}$ computer. *Formalized Mathematics*, 5(4):519–528, 1996.

A The grammar of control headers

```
Functor-control-header ==
   TeX-mode Bracket-ability Definition-style Operation-kind Forcing
   Priority Argument-context List-of-bracket-disabled-arguments ";" .

Predicate-control-header == Predicate-kind Predicate-validity .

Type-control-header      == TeX-mode Article Type-validity .

Adjective-control-header == Adjective-kind .

TeX-mode   == Math-mode | Horizontal-mode .
Math-mode  == "m" .           Horizontal-mode == "h" | "H" .

Bracket-ability == Open | Closed .
Open            == "o" .      Closed          == "c" .

Definition-style    == Normal-functor | A-singular-functor |
                       AN-singular-functor | Plural-functor .
Normal-functor      == .        A-singular-functor  == "s" .
AN-singular-functor == "n" .    Plural-functor      == "p" .

Operation-kind ==
  ( Prefix | Upper-prefix | Lower-prefix | Postfix | Upper-postfix |
    Lower-postfix | Overfix | Underfix | Infix | Non-associative-infix |
    Right-associative-infix | Circumfix | Other-operation )
  [ "(" Number [ "," Number ] ")" ] .
Prefix   == "l" .     Upper-prefix  == "k" .      Lower-prefix  == "m" .
Postfix  == "r" .     Upper-postfix == "q" .      Lower-postfix == "w" .
Overfix  == "t" .     Underfix      == "b" .      Infix         == "i" .
```

```
Non-associative-infix    == "x" .        Right-associative-infix == "y" .
Circumfix                == "c" .        Other-operation         == .

Forcing                  == | "{" List-of-operation-kinds "}" .
List-of-operation-kinds == { Operation-kind } .

Priority  == |
  "@" ( Very-weak | Weak | Weak-additive | Additive | Strong-additive |
        Weak-multiplicative | Multiplicative | Strong-multiplicative |
        Strong | Very-strong ) .
Very-weak       == "0" .         Weak-multiplicative   == "5" .
Weak            == "1" | "w" .   Multiplicative        == "6" | "m" .
Weak-additive   == "2" .         Strong-multiplicative == "7" .
Additive        == "3" | "a" .   Strong                == "8" | "s" .
Strong-additive == "4" .         Very-strong           == "9" .

Argument-context == { "/" Number Operation-kind Priority } .
Number         == "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" .

List-of-bracket-disabled-arguments == { "#" Number } .

Predicate-kind  == Symbolic | Language-phrase .
Language-Phrase ==
       Simple-phrase | IS-phrase | HAS-phrase | SAT-phrase .
Symbolic    == "m" .     Simple-phrase   == "h" .
IS-phrase   == "i" .     HAS-phrase  == "j" .       SAT-phrase  == "s" .

Predicate-validity == Predicate-valid | Predicate-repeated .
Predicate-valid    == | "y" .    Predicate-repeated == "n" .

Article    == A-article | AN-article .
A-article  == "a" .              AN-article == "n" .

Type-validity == Type-valid | Type-repeated | Type-abbreviation .
Type-valid    == .   Type-repeated   == "%" .  Type-abbreviation == "@" .

Adjective-kind  == A-adjective | AN-adjective | Empty-adjective |
                 HAS-adjective | SAT-adjective | Noun-adjective .
A-adjective     == "a" .         AN-adjective    == "n" .
Empty-adjective == "e" .         HAS-adjective   == "w" | "x" .
SAT-adjective   == "s" .         Noun-adjective  == "b" .
```