A proposed syntax for binders in Mizar

Freek Wiedijk

University of Nijmegen

Abstract. Binders like ‘lim’ (for limits), ‘$\sum$’ (for summation) and ‘$\int$’ (for integration) are an essential part of the language of mathematics. They are the ‘higher order’ elements of mathematical expressions.
At the TYPES meeting of 2003 in Turin, Andrzej Trybulec told me that the reason that Mizar did not have binders was that nobody had proposed a good syntax for them yet. To take this reason away, we will present a syntax for binders in Mizar.

1 Introduction

1.1 Problem

One of the main difficulties when trying to ‘sell’ the mathematical language of the Mizar system [2, 3] to real mathematicians is its lack of binders. Consider for instance the formula[1]:

$$\int x^n \, dx = \frac{x^{n+1}}{n+1}$$

In the current Mizar language this has to be rendered like:

```
for f st for x holds f.x = x to_power n holds
 (integral f).x = (x to_power (n + 1))/(n + 1)
```

instead of as the more concise and readable:

```
(integral x of x to_power n).x = (x to_power (n + 1))/(n + 1)
```

Clearly adding binders to the Mizar language will improve it. Therefore we will propose a notation for binders, in the hope that some day it will be implemented.

At the TYPES meeting of 2003 I showed a rough version of this proposal to Andrzej Trybulec and Czeslaw Bylinski. This note is a worked out version of the same concept.

[1] In this example $\int$ is an indefinite integral that is not very subtle (it can be taken to be an abbreviation of $\int_0^x$), because there is no constant of integration. A more intelligent formal version of indefinite integration is certainly possible.

1.2 Approach

We will look for a notation that fits the current style of the Mizar language. For instance keywords will be preferred over symbols. Therefore a notation like:

```
\x. x^2 + 1
```

(the way λ-terms are written in the HOL Light system [1]) will not be considered to be as 'Mizar like' as:

```
lambda x of x^2 + 1
```

Another goal will be that the binders will behave similarly to the first order Mizar operators ('functors') with respect to overloading and implicit arguments.

1.3 Related Work

Most proof assistants for mathematics are based on a higher order logic. This means that they have λ-abstraction as a fundamental building block, and therefore binders are natural for them. In this respect Mizar is uncommon.

In higher order systems binders often are represented by a functional applied to a λ-term. A binder $\mathcal{B}$ will thus be represented like:

$$\mathcal{B}(\lambda x.\, f(x))$$

instead of it having 'first class' status:

$$\mathcal{B}x.\, f(x)$$

Sometimes the fact that binders are functionals applied to λ-terms is hidden with syntactic sugar. For instance in HOL Light, after one says `parse_as_binder "integral"` the term:

```
(integral) (\x. x^2 + 1)
```

will be pretty-printed by the system as:

```
integral x. x^2 + 1
```

and it also can (and generally will) be entered into the system like this. However, internally it still has the structure of the first term.

An example of a binder that in a type theoretical system is not defined with the aid of the λ-binder is the product type $\Pi x.\, A(x)$.

In our proposed Mizar notation *all* binders will be first class objects. The `lambda` binder that we will show in Section 4 will have no special status among the other examples.

1.4 Contribution

This note shows how binders can be added to the Mizar system in a natural way.

1.5 Outline

In Section 2 we will present our proposed syntax for Mizar binders in an abstract way. In Sections 3, 4 and 5 we will illustrate our proposal through examples. Finally in Section 6 we will discuss.

2 Syntax

The syntax that we propose for Mizar binders is[2]:

$$\underline{\textit{binder}}\ x\ [\ \textsf{\textbf{where}}\ x\ \textsf{\textbf{is}}\ \textit{type}\ ]\ \textsf{\textbf{at}}\ \textit{args}\ \textsf{\textbf{of}}\ \textit{body}$$

The way a binder will be defined is:

```
definition let $args$; [ assume $conditions$; ]
 binder { $f(type_1) \to type_2$ }
  $\underline{binder}$ $x$ [ where $x$ is $type_1$ ] at $args$ of $f(x) \to type_3$
   means ...;
 ...
end
```

(In such a definition the '**where is**' clause is always redundant, because $type_1$ is also present in the typing of the f. However we allow it for orthogonality.)

To be able to introduce binder symbols, we also will need a new category in the vocabularies. We propose to use the letter '`B`' for this.

Note that the binder notation uses the keywords **at**, **of** and **binder**, together with the keywords **where** and **is** that are also used in the Fränkel operator (the Mizar notation for set comprehension).

3 Finite sums

As an example of what our proposed notation looks like in practice, take the following equation:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

It becomes:

$$\underline{\textsf{sum}}\ \textsf{i}\ \textsf{\textbf{at}}\ \textsf{1,n}\ \textsf{\textbf{of}}\ \textsf{i} = \textsf{n}*(\textsf{n}+1)/2;$$

To be able to write this formula, we need a vocabulary that contains the following lines[3]:

[2] We underline all binders to make the structure of these examples clearer, but these underlinings do not represent anything special in the ASCII of the Mizar file. Look at Section 5 for examples written in a plain typewriter font.

[3] The symbol sum′ is the operator that in the MML is defined in `RVSUM_1:def 10`. It is the functor that sums a finite sequence of complex numbers. The symbol <u>sum</u> is the binder that we define from it.

```
O sum′
B sum
```

Here is a way that the binder sum might then be defined:

```
reserve i,n,a,b for natural number;

definition let n;
 binder { f(natural number) → complex number }
  sum i at n of f(i) → complex number
 means
  ex f' being FinSequence of COMPLEX st len f' = n &
   (for i st i < n holds f'.(i + 1) = f(i)) & it = sum′ f';
 existence . . . ;
 uniqueness . . . ;
end;

definition let a,b;
 binder { f(natural number) → complex number }
  sum i at a,b of f(i) → complex number
 equals
  (sum i at b + 1 of f(i)) − (sum i at a of f(i));
 coherence . . . ;
end;
```

Note that '**sum** i **at** n' means $\sum_{i<n}$, while '**sum** i **at** a,b' means $\sum_{i=a}^{b}$.

4 λ-abstraction

The most elementary binder is λ-abstraction (although it is not the most important one: lim, $\sum$ and $\int$ are more important for real mathematics). As an example λ-abstraction is also interesting because it shows how implicit arguments behave with binders.

The λ-term:

$$\lambda x.\, x^2 + 1$$

is written:

```
lambda x of x^2 + 1
```

Here is a possible definition of the lambda binder:

```
reserve A,B for non empty set;
reserve x for Element of A;

definition let A,B;
 binder { f(Element of A) → Element of B }
  lambda x of f(x) → Function of A,B
 means
  for x holds it.x = f(x)
 existence ...;
 uniqueness ...;
end;
```

Note that the arguments A and B to the lambda binder are inferred rather than being given explicitly in the term.

The β-reduction rule now becomes a scheme[4]:

```
scheme beta { A() → non empty set, B() → non empty set,
    f(Element of A()) → Element of B(), a() → Element of A() }:
 (lambda x where x is Element of A() of f(x)).a() = f(a())
...;
```

The 'definitional scheme' (the equivalent of a definitional theorem for a binder) of the the definition of lambda is:

```
scheme lambda_def { A() → non empty set, B() → non empty set,
    f(Element of A()) → Element of B() } :
 for x being Element of A() holds
  (lambda x where x is Element of A() of f(x)).x = f(x);
```

which of course is already very close to this beta scheme.

5 Examples

We now list some more formulas with their proposed Mizar translation. The first two are repeats from the previous two sections. Instead of displaying the Mizar fragments in a nice pretty printed format like before, we will show them here in a plain typewriter font.

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

```
sum i at 1,n of i = n*(n + 1)/2
```

$$\lambda x.\, x^2 + 1$$

```
lambda x of x^2 + 1
```

$$f \text{ is continuous} \iff \lim_{x\to a} f(x) = f(a) \quad \text{for all } a$$

```
f is continuous iff for a holds lim x at a of f.x = f.a
```

[4] The double **of** in '... **of** A() **of** f(x)' might be a bit confusing. The first **of** is part of the type, the second **of** is part of the binder.

$$\sum_{p \text{ prime}} \frac{1}{p^2} < \frac{1}{2}$$

```
sum p at { p : p is prime } of 1/p^2 < 1/2
```

$$\iint_{x^2+y^2\leq 1} \sqrt{1-x^2+y^2}\, dx\, dy = \frac{2}{3}\pi$$

```
integral z where z is Element of [:REAL,REAL:] at
  { [x,y] : x^2 + y^2 <= 1 } of sqrt(1 - (z‘1)^2 + (z‘2)^2)
= 2/3*pi
```

The last two examples are interesting because the expressions `{ p : p is prime }` and `{ [x,y] : x^2 + y^2 <= 1 }` have type `set`. Therefore these variants of `sum` and `integral` need a ‘permissive definition’ (they need an assumption) to make sense.[5]

```
definition let D be set;
 assume D is Subset of NAT;
 binder { f(natural number)->complex number }
  sum i at D of f(i) means ...
 ...
end
```

and:

```
definition let D be set;
 assume D is Subset of [:REAL,REAL:];
 binder { f(Element of [:REAL,REAL:])->complex number }
  integral z at D of f(z) means ...
 ...
end
```

6 Conclusion

6.1 Discussion

Binders are higher order objects. Therefore reasoning about binders in Mizar will consist of the application of schemes. It is not clear whether this will be practical.

The style of defining binders that we propose is a mixture of the style of ordinary (first order) definitions and the style of (higher order) schemes. This is apparent when comparing the definition of lambda in Section 4 with its definitional scheme lambda_def. In the definition itself (which has a mixed style)

[5] Really in the definition of `integral` one would also like the set `D` to be `measurable` and the function `f` to be `integrable`.

the sets A and B are introduced in a first order style '**let** A,B **be** non empty set', while in its definitional scheme (which has a purely higher order style) they are introduced in the higher order style '{ A() → non empty set, B() → non empty set }'. It is not clear whether this mixture of first order and higher order style in the definition of binders will cause trouble in the implementation.

6.2 Future work

Implement binders in the Mizar system.

Acknowledgments. Thanks to Andrzej Trybulec and Czeslaw Bylinski for discussing this proposal with me.

References

1. John Harrison. *The HOL Light manual (1.1)*, 2000. `<http://www.cl.cam.ac.uk/users/jrh/hol-light/manual-1.1.ps.gz>`.
2. M. Muzalewski. *An Outline of PC Mizar.* Fondation Philippe le Hodey, Brussels, 1993. `<http://www.cs.kun.nl/~freek/mizar/mizarmanual.ps.gz>`.
3. F. Wiedijk. Mizar: An Impression. `<http://www.cs.kun.nl/~freek/mizar/mizarintro.ps.gz>`, 1999.