Implementing Math Problems in LON-CAPA using CAS Support

Peter Riegler
and hopefully other contributers

August 20, 2008

Abstract

We describe the LON-CAPA interface to computer algebra systems (CAS) which conveniently allows to implement rather sophisticated math problems.

1 Introduction

2 LON-CAPA interfaces to CAS

There are two interfaces: The first one `&cas` can be called anywhere from a perl script. Its primary use is to give access to CAS functionality from within perl. The second one `<mathresponse>` is used for doing the complete grading of a problem exclusively be means of CAS functionality.

Both interfaces preprocess students input by the perl function `&implicit_multiplication($f)`. It adds mathematical multiplication operators to the formula expression $f where only implicit multiplication is used. Example: &implicit_multiplication('2(b+3c)') returns 2*(b+3*c).

2.1 `&cas`-interface

&cas($s,$e,$l) Evaluates the expression $e inside the symbolic algebra system $s. Currently, only the Maxima symbolic math system is implemented. $l is an optional comma-separated list of libraries. Example: &cas('maxima','6*7')

2.2 `<mathresponse>`-interface

Math Response is a way to have a problem graded based on an algorithm that is executed inside of a computer algebra system. The use of this response type is generally discouraged, since the responses will not be analyzable by the LON-CAPA statistics tools.

Which computer algebra system is to be used is specified in the cas argument of the mathresponse tag; currently, only Maxima is available. LON-CAPA sets up two arrays inside the computer algebra system: RESPONSE and LONCAPALIST. RESPONSE contains the student input by component, for example, if "3,42,17" is entered, RESPONSE[2] would be 42. LONCAPALIST contains the arguments passed in the args of mathresponse.

The answerdisplay is what is displayed when the problem is in "Show Answer" mode.

The following example illustrates this.

```
<problem>
  <script type="loncapa/perl">
$a1 = random(-6,6,4);
$a2 = random(-6,6,4);
$n1 = random(3,11,2);
$n2 = random(2,10,2);
$function = "$a1*cos($n1*x)+$a2*sin($n2*x)";
$example=&xmlparse('An example would be <m eval="on">$(sin($n1\cdot x)+cos($n2\cdot x))/\sq
  </script>

<startouttext />
  Give an example of a function
  <ol>
    <li>
        which is orthogonal to <algebra>$function</algebra> with respect to the
        scalar product
        <m>
           \[<g \mid h> =
               \frac{1}{\pi} \int_{-\pi}^{\pi}dx g(x) \cdot h(x)\]
        </m>
    </li>
    <li>
        whose norm is 1.
    </li>
  </ol>
<endouttext />

<mathresponse answerdisplay="$example" cas="maxima" args="$function">
    <answer>
overlap:integrate((RESPONSE[1])*(LONCAPALIST[1]),x,-%pi,%pi)/%pi;
norm:integrate((RESPONSE[1])*(RESPONSE[1]),x,-%pi,%pi)/%pi;
is(overlap=0 and norm=1);
    </answer>
    <textline readonly="no" size="50" />
    <hintgroup showoncorrect="no">
        <mathhint name="ortho" args="$function" cas="maxima">
```

```
            <answer>
overlap: integrate((LONCAPALIST[1])*(RESPONSE[1]),x,-%pi,%pi)/%pi;
is(not overlap = 0);
            </answer>
        </mathhint>
        <mathhint name="norm" args="$function" cas="maxima">
            <answer>
norm: integrate((RESPONSE[1])*(RESPONSE[1]),x,-%pi,%pi)/%pi;
is(not norm = 1);
            </answer>
        </mathhint>
        <hintpart on="norm">
           <startouttext />
The function you have provided does not have a norm of one.
           <endouttext />
        </hintpart>
        <hintpart on="ortho">
            <startouttext />
The function you have provided is not orthogonal.
            <endouttext />
        </hintpart>
    </hintgroup>
</mathresponse>


<postanswerdate>
    <startouttext />
        <p>
Note that with respect to the above norm, <m>$\cos(nx)$</m> is perpendicular
to <m>$\sin(nx)$</m> and perpendicular to <m>$\cos(mx)$</m> for
<m>$n\ne m$</m>.
        </p>
    <endouttext />
</postanswerdate>
</problem>
```

a simpler example might be more appriopriate here

3 Interface to maxima

LON-CAPA servers run several maxima sessions in parallel. There is a queue which distributes CAS calls to these sessions. When processing a new CAS call one has to be sure that maxima is reset to some default state. In particular functions, variables etc. defined in previous calls should be removed. LON-CAPA automatically takes care of that by means of the following sequence of

commands which is executed every time before a maxima code snippet supplied by an author will be executed:

```
display2d:false;simp:true;kill(all);
```

Authors should be aware of this, because `kill(all)` does *not* delete all previously defined stuff.[1] One known issue is, that previously loaded maxima packages will not be removed by `kill(all)`.

The two commands in front of `kill(all)` make sure that the maxima session renders output as expressions contained in a single line and that maxima's elementary simplification functionality is turned on. (More on the `simp`-flag and its usefulnes in 5.)

4 Primer on maxima

This sections serves as a short tutorial on maxima. It is intended for readers who either have not worked with maxima before or are not familiar with CAS at all. In the long run when authoring LON-CAPA problems you might wish to consult more advanced material provided e.g. on

http://maxima.sourceforge.net/

To start with it is a good idea to have access to maxima either on a mainframe or on your personal computer. This will also be helpful if you author problems using maxima. Maxima runs on almost any platform and can be downloaded at the above mentioned URL. In the following we will use the terminal interface to maxima. The graphical user interfaces basically behave the same way, except for wxmaxima where you do not have a prompt but have to enter your expressions at the buttom.

Let's start with some survival basics: exiting a maxima session and getting help. To quit maxima type `quit();`

```
Maxima 5.9.2 http://maxima.sourceforge.net
Using Lisp GNU Common Lisp (GCL) GCL 2.6.7 (aka GCL)
Distributed under the GNU Public License. See the file COPYING.
Dedicated to the memory of William Schelter.
This is a development version of Maxima. The function bug_report()
provides bug reporting information.
(%i1) quit();
```

To search for help use `?` followed by blank and an appropriate keyword and follow the instructions. Note that the blank after `?` is essential.

```
(%i1) ? integrate
```

[1] In fact, there seems to be no maxima command which does the desired job. If the cherished reader finds out about one, any LON-CAPA developer will be more than happy to change the above sequence accordingly.

```
 0: integrate :(maxima.info)Definitions for Integration.
 1: integrate_use_rootsof :Definitions for Integration.
Enter space-separated numbers, 'all' or 'none':
```

Entering 1 would give you all the details on how to integrate functions.

Obviously you won't use neither `quit()` nor `?` when authoring LON-CAPA problems. To assure that you won't do that accidentily LON-CAPA will block these and a number of other commands.

Commands have to end either with `;` or `$`. The effect of `$` is that no output will be displayed:

```
(%i2) 1+1;

(%o2)                                  2
(%i3) 1+1$

(%i4)
```

By now you might have noticed that inputs are precedented by `(%i<number>)` and the corresponding outputs by `(%o<number>)`, where `<number>` is a consecutive number. You can refer back to previous outputs via `%o<number>`:

```
(%i4) (2*%o2)^20;

(%o4)                             1099511627776
```

Note that refering back via `%o<number>` does not make sense when authoring problems, as you will not know `<number>`. Instead you will use named variables of course. The assignment operator for variables is `:`

```
(%i5) a: sin(%pi);

(%o5)                                  0

(%i6) b: cos(%pi);

(%o6)                                 - 1
```

Mathematical constants such as π and Euler's number are precedented by `%`:

```
(%i7) log(%e);

(%o7)                                  1
```

The assignment operator for functions is `:=`

```
(%i8) f(x):=b*x^2;

                                                2
```

```
(%o8)                                f(x) := b x
(%i9) f(2);

(%o9)                                     - 4
(%i10) integrate(f(x),x);

                                            3
                                           x
(%o10)                                   - --
                                            3
```

Here we see maxima's two-dimensional rendering. Setting the flag `display2d` to false will turn it off:

```
(%i11) display2d:false;

(%o11) false
(%i12) f(x);

(%o12) -x^2
```

When authoring `<mathresponse>`-problems you will have to make sure that the last value your maxima code snippet will return is either true or false. Most often `is` will do the job. For instance,

```
(%i13) RESPONSE[1]: -x^3/3 + 5;

(%o13) 5-x^3/3
(%i14) is ( diff(RESPONSE[1],x) = f(x) );

(%o14) true
```

checks whether the students response `RESPONSE[1]` is the antiderivative of $f(x)$. Note that by differentiating `RESPONSE[1]` via the `diff`-command we make sure that any integration constant will be accepted:

```
(%i15) RESPONSE[1]: -x^3/3 + C;

(%o15) C-x^3/3
(%i12) is ( diff(RESPONSE[1],x) = f(x) );

(%o16) true
(%i16) RESPONSE[1]: -x^3/3 + C*x;

(%o17) x*C-x^3/3
(%i18) is ( diff(RESPONSE[1],x) = f(x) );

(%o18) false
```

more on delayed assignment, exact numbers, ...

5 Simplify the following expression ...

Many text book examples start with this phrase. Giving it a moment of thought reveals that these are somewhat ill-posed problems. For what is simple depends to a large extent on what you *define* to be simple and on what you want to do next.

more to be written by Zhoujing Wang (my student working on the issue)