An Infrastructure for formatting Problems*

Michael Kohlhase
Jacobs University, Bremen
http://kwarc.info/kohlhase

July 19, 2010

Abstract

The `problem` package supplies an infrastructure that allows specify problems and to reuse them efficiently in multiple environments.

Contents

*Version v0.9c (last revised 2010/06/25)

1 Introduction

The `problem` package supplies an infrastructure that allows specify problem. Problems are text fragments that come with auxiliary functions: hints, notes, and solutions[1]. Furthermore, we can specify how long the solution to a given problem is estimated to take and how many points will be awarded for a perfect solution.

Finally, the `problem` package facilitates the management of problems in small files, so that problems can be re-used in multiple environment.

2 The User Interface

2.1 Package Options

`solutions` `notes` `hints` `pts` `min`

The `problem` package takes the options `solutions` (should solutions be output?), `notes` (should the problem notes be presented?), `hints` (do we give the hints?), `pts` (do we display the points awarded for solving the problem?), `min` (do we display the estimated minutes for problem soling). If theses are specified, then the corresponding auxiliary parts of the problems are output, otherwise, they remain invisible.

`boxed` `test`

The option specifies that problems should be formatted in framed boxes so that they are more visible in the text. Finally, the `test` option signifies that we are in a test sitution, so this option does not show the solutions (of course), but leaves space for the students to solve them.

`extract`

The `extract` option can be set if we want to extract a problems file, e.g. to display the solutions, see Section 2.3 for a discussion.

2.2 Problems and Solutions

`problem` `id` `pts` `min` `title`

The main enviornment provided by the `problem` package is (surprise surprise) the `problem` environment. It is used to mark up problems and excercises. The environment takes an optional KeyVal argument with the keys `id` as an identifier that can be reference later, `pts` for the points to be gained from this exercise in homework or quiz situations, `min` for the estimated minutes needed to solve the problem, and finally `title` for an informative title of the problem. For an example of a marked up problem see Figure 1 and the resulting markup see Figure 2.

`solution` `solutions` `id` `for` `height` `test`

The `solution` environment can be to specify a solution to a problem. If the `solutions` option is set or `\solutionstrue` is set in the text, then the solution will be presented in the output. The `solution` environment takes an optional KeyVal argument with the keys `id` for an identifier that can be reference `for` to specify which problem this is a solution for, and `height` that allows to specify the amount of space to be left in test situations (i.e. if the `test` option is set in the `\usepackage` statement).

`hint` `note`

, the `hint` and `exnote` environments can be used in a `problem` enviroment to

[1] for the momenet multiple choice problems are not supported, but may well be in a future version

```
\usepackage[solutions,hints,pts,min]{problem}
\begin{document}
  \begin{problem}[id=elefants,pts=10,min=2,title=Fitting Elefants]
    How many Elefants can you fit into a Volkswagen beetle?
    \begin{hint} Think positively, this is simple!\end{hint}
    \begin{exnote}Justify your answer\end{exnote}
    \begin{solution}[for=elefants,height=3cm]
      Four, two in the front seats, and two in the back.
    \end{solution}
  \end{problem}
\end{document}
```

Example 1: A marked up Problem

Problem 2.1 (Fitting Elefants)
How many Elefants can you fit into a Volkswagen beetle?

Hint: Think positively, this is simple!

Note: Justify your answer

Solution: Four, two in the front seats, and two in the back.

Example 2: The Formmatted Problem from Figure 1

give hints and to make notes that elaborate certain aspects of the problem.

2.3 Including Problems

`\includeproblem` The `\includeproblem` macro can be used to include a problem from another file. It takes an optional KeyVal argument and a second argument which is a path to the file containing the problem (the macro assumes that there is only one problem in the include file). The keys `title`, `min`, and `pts` specify the problem title, the estimated minutes for solving the problem and the points to be gained, and their values (if given) overwrite the ones specified in the `problem` environment in the included file.

`title` `min` `pts`

Sometimes we want to collect all the included problems into a separate file that can be typeset independently. The main application is to have course notes into which the problems are included (usually in boxed form to distinguish them from the rest of the text and without solutions) and to have the problems with solutions in a separate file (to encourage students to try and solve the problems before looking up solutions). In this situation set the `extract` option on the notes file ⟨*notes*⟩`.tex`, which causes a file ⟨*notes*⟩`-solutions.tex` to be generated that has the `\includeproblem` statements with the respective numbers from the main document. This can then be imported into a document with the respective front and backmatter. In particular the frontmatter of the importing will ususlly specify the `solutions` option to generate solutions.

`extract`

`solutions`

2.4 Reporting Metadata

The sum of the points and estimated minutes (that we specified in the `pts` and `min` keys to the `problem` environment or the `\includeproblem` macro) to the log file and the screen after each run. This is useful in preparing exams, where we want to make sure that the students can indeed solve the problems in an alotted time period.

The `\min` and `\pts` macros allow to specify (i.e. to print to the margin) the distribution of time and reward to parts of a problem, if the `pts` and `pts` package options are set. This allows to give students hints about the estimated time and the points to be awarded.

3 The Implementation

3.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. They all come with their own conditionals that are set by the options.

```
⟨*package⟩
\newif\ifexnotes\exnotesfalse\DeclareOption{notes}{\exnotestrue}
\newif\ifhints\hintsfalse\DeclareOption{hints}{\hintstrue}
\newif\ifsolutions\solutionsfalse\DeclareOption{solutions}{\solutionstrue}
\newif\ifpts\ptsfalse\DeclareOption{pts}{\ptstrue}
\newif\ifmin\minfalse\DeclareOption{min}{\mintrue}
\newif\ifboxed\boxedfalse\DeclareOption{boxed}{\boxedtrue}
\newif\ifextract\extractfalse\DeclareOption{extract}{\extracttrue}
\ProcessOptions
⟨/package⟩
```

Then we make sure that the necessary packages are loaded (in the right versions).

```
⟨*package⟩
\RequirePackage{keyval}[1997/11/10]
\RequirePackage{xcomment}
\RequirePackage{sref}
⟨/package⟩
```

Here comes the equivalent header information for LaTeXML, we also initialize the package inclusions. Since LaTeXML currently does not process package options, we have nothing to do.

```
⟨*ltxml⟩
# -*- CPERL -*-
package LaTeXML::Package::Pool;
use strict;
use LaTeXML::Package;
RequirePackage('sref');
⟨/ltxml⟩
```

Then we register the namespace of the requirements ontology

```
⟨*ltxml⟩
RegisterNamespace('prob'=>"http://omdoc.org/ontology/problems#");
RegisterDocumentNamespace('prob'=>"http://omdoc.org/ontology/problems#");
⟨/ltxml⟩
```

3.2 Problems and Solutions

We now prepare the KeyVal support for problems. The key macros just set appropriate internal macros.

```
⟨*package⟩
\srefaddidkey[prefix=prob.]{problem}
```

```
\omdaddkey{problem}{pts}
\omdaddkey{problem}{min}
\omdaddkey{problem}{title}
\omdaddkey{problem}{refnum}
```

Then we set up a box and a counter for problems

```
\newsavebox{\probbox}
\newcounter{problem}[section]
```

`\prob@number` We consolidate the problem number into a reusable internal macro

```
\def\prob@number{\ifx\inclprob@refnum\@empty
\ifx\problem@refnum\@empty\thesection.\theproblem\else\problem@refnum\fi
\inclprob@refnum\fi}
```

We consolidate the problem header line into a separate internal macro that can be reused in various settings.

`\prob@heading` We consolidate the problem header line into a separate internal macro that can be reused in various settings.

```
\def\prob@heading{Problem \prob@number%
\ifx\sref@id\@empty\else{\sref@label@id{Problem \thesection.\theproblem}}\fi%
\ifx\inclprob@title\@empty% if there is no outside title
\ifx\problem@title\@empty{:\quad}\else{\quad(\problem@title)\hfill\\}\fi
\else\quad(\inclprob@title)\hfill\\\fi}% else show the outside title
```

With this in place, we can now define the `problem` environment. It comes in two shapes, depending on whether we are in boxed mode or not. In both cases we increment the problem number and output the points and minutes (depending) on whehter the respective options are set.

`problem`

```
\ifboxed
\newenvironment{problem}[1][]{\omdsetkeys{problem}{#1}\sref@target%
\stepcounter{problem}\show@pts\show@min\record@problem%
\begin{lrbox}{\probbox}\begin{minipage}{.9\textwidth}}
{\end{minipage}\end{lrbox}
\setbox0=\hbox{\begin{minipage}{.9\textwidth}%
\noindent\textbf\prob@heading\rm%
\end{minipage}}
\smallskip\noindent\fbox{\vbox{\box0\vspace*{.2em}\usebox\probbox}}\smallskip}
\else
\newenvironment{problem}[1][]{\omdsetkeys{problem}{#1}\sref@target%
\stepcounter{problem}\show@pts\show@min\record@problem%
\par\noindent\textbf\prob@heading\rm}
{\smallskip}
\fi%boxed
⟨/package⟩
```

Note that we allow hints and solutions in the body of a `problem` environment so we have to allow the `omdoc:CMP` and `omdoc:p` elements to autoclose.

```
⟨*ltxml⟩
DefCMPEnvironment('{problem} OptionalKeyVals:problem',
     "<omdoc:exercise ?&KeyVal(#1,'id')(xml:id='&KeyVal(#1,'id')')()"
    .  "prob:dummy='for the namespace'"
    .   "?#locator(stex:srcref='#locator')()>"
    .    "?&KeyVal(#1,'title')(<dc:title ?#locator(stex:srcref='#locator')()>&KeyVal(#1,'title'
    .    "?&KeyVal(#1,'min')(<omdoc:meta property='prob:solvedinminutes' "
    .       "?#locator(stex:srcref='#locator')()>&KeyVal(#1,'min')</omdoc:meta>)()"
    .    "?&KeyVal(#1,'pts')(<omdoc:meta property='prob:points' "
    .       "?#locator(stex:srcref='#locator')()>&KeyVal(#1,'pts')</omdoc:meta>)()"
    .   "<omdoc:CMP ?#locator(stex:srcref='#locator')()><omdoc:p>#body"
    ."</omdoc:exercise>\n");
⟨/ltxml⟩
```

`\record@problem` This macro records information about the problems in the `*.aux` file.

```
⟨*package⟩
\def\record@problem{\protected@write\@auxout{}%
{\string\@problem{\prob@number}%
{\ifx\inclprob@pts\@empty\problem@pts\else\inclprob@pts\fi}%
{\ifx\inclprob@min\@empty\problem@min\else\inclprob@min\fi}}}
⟨/package⟩
```

`\@problem` This macro acts on a problem's record in the `*.aux` file. It does not have any functionality here, but can be redefined elsewhere (e.g. in the `assignment` package).

```
⟨*package⟩
\def\@problem#1#2#3{}
⟨/package⟩
```

The `solution` environment is similar to the `problem` environment, only that it is independent of the boxed mode. It also has it's own keys that we need to define first.

```
⟨*package⟩
\define@key{soln}{id}{\def\soln@id{#1}}
\define@key{soln}{for}{\def\soln@for{#1}}
\define@key{soln}{height}{\def\soln@height{#1}}
\ifsolutions
\newenvironment{solution}[1][]%
{\hrule\smallskip{\bf Solution: }\begin{small}}%
{\hrule\end{small}}
\else\newxcomment[]{solution}\fi
% \newsavebox{\solution@box}
% \newlength{\solution@width}
% \setlength{\solution@width}{14cm}
% \newenvironment{solution}[1][]%
% {\begin{lrbox}{\solution@box}\begin{minipage}{\solution@width}
```

```
% \hrule\smallskip{\bf Solution: }\small}
% {\smallskip\hrule\end{minipage}\end{lrbox}
% \ifsolutions\begin{center}\usebox{\solution@box}\end{center}\fi}
⟨/package⟩
⟨*ltxml⟩
DefKeyVal('soln','id','Semiverbatim');
DefKeyVal('soln','height','Semiverbatim');
DefKeyVal('soln','for','Semiverbatim');
DefCMPEnvironment('{solution} OptionalKeyVals:soln',
       "<omdoc:solution ?&KeyVals(#1,'for')(for='&KeyVal(#1,'for')')() ?#locator(stex:srcref='#
     . "#body"
     . "</omdoc:solution>");
⟨/ltxml⟩
⟨*package⟩
\ifexnotes
\newenvironment{exnote}[1][]%
{\par\noindent\hrule\smallskip{\bf Note: }\small}
{\smallskip\hrule}
\else%ifexnotes
\newxcomment[]{exnote}
\fi%ifexnotes
\ifhints
\newenvironment{hint}[1][]%
{\par\noindent\hrule\smallskip{\bf Hint: }\small}
{\smallskip\hrule}
\else%ifhints
\newxcomment[]{hint}
\fi%ifhints
⟨/package⟩
⟨*ltxml⟩
DefCMPEnvironment('{exnote}',
   "<omdoc:hint ?#locator(stex:srcref='#locator')()>"
  .  "<omdoc:CMP ?#locator(stex:srcref='#locator')()>"
  .    "<omdoc:p>#body<omdoc:p>"
  .  "</omdoc:CMP>"
  ."</omdoc:hint>");
DefCMPEnvironment('{hint}',
   "<omdoc:hint ?#locator(stex:srcref='#locator')()>"
  .  "<omdoc:CMP ?#locator(stex:srcref='#locator')()>"
  .    "<omdoc:p>#body</omdoc:p>"
  .  "</omdoc:CMP>"
  ."</omdoc:hint>");
DefConstructor('\pts{}',"");
DefConstructor('\min{}',"");
⟨/ltxml⟩
```

3.3 Including Problems

The first action is to make a ⟨*jobname*⟩`-problems.tex` file, if the `extract` option is set.

```
⟨*package⟩
\ifextract
\newwrite\problem@file
\immediate\openout\problem@file=\jobname-problems.tex
\AtEndDocument{\closeout\problem@file}
\fi
⟨/package⟩
```

`\includeproblem` The `\includeproblem` command is essentially a glorified `\input` statement, it sets some internal macros first that overwrite the local points. After that (so that the included problem had time to step the problem number) it writes the `\includeproblem` statement to the problems file, if the `extract` option is set. Here we add a key `refnum=\prob@num` to the inlcudeproblem, so that we can remember the number from the main document.[1]

EdNote(1)

```
⟨*package⟩
\omdaddkey{inclprob}{pts}
\omdaddkey{inclprob}{min}
\omdaddkey{inclprob}{title}
\omdaddkey{inclprob}{refnum}
\clear@inclprob@keys
\newcommand{\includeproblem}[2][]{%
\bgroup\omdsetkeys{inclprob}{#1}\input{#2}\ifsolutions\newpage\fi\egroup
\ifextract\def\@test{#1}
\def\prob@num{\ifx\inclprob@refnum\@empty\thesection.\theproblem\else\inclprob@refnum\fi}
\def\inclprob@keys{#1\ifx\@test\@empty\else,\fi refnum=\prob@num}
\protected@write\problem@file{}{\string\includeproblem[\inclprob@keys]{#2}}
\fi}
⟨/package⟩
⟨*ltxml⟩
DefKeyVal('prob','pts','Semiverbatim');
DefKeyVal('prob','min','Semiverbatim');
DefKeyVal('prob','title','Semiverbatim');
DefConstructor('\includeproblem OptionalKeyVals:prob Semiverbatim',
   "<omdoc:ref xref='#2' ?#locator(stex:srcref='#locator')() "
  .     "prob:dummy='for the namespace'>"
  .  "?&KeyVal(#1,'title')(<dc:title ?#locator(stex:srcref='#locator')()>&KeyVal(#1,'title')</d
  .  "?&KeyVal(#1,'min')(<omdoc:meta property='prob:solvedinminutes' "
  .     "?#locator(stex:srcref='#locator')()>&KeyVal(#1,'min')</omdoc:meta>)()"
  .  "?&KeyVal(#1,'pts')(<omdoc:meta property='prob:points' "
  .     "?#locator(stex:srcref='#locator')()>&KeyVal(#1,'pts')</omdoc:meta>)()"
  ."</omdoc:ref>");
⟨/ltxml⟩
```

[1] EdNote: do something about the overwriting of problem metadata in the LaTeXML binding.

```
⟨*ltxml⟩
Tag('omdoc:exercise',afterOpen=>\&numberIt);
Tag('omdoc:solution',afterOpen=>\&numberIt);
Tag('omdoc:hint',afterOpen=>\&numberIt);
⟨/ltxml⟩
```

3.4 Reporting Metadata

```
⟨*package⟩
\def\pts#1{\ifpts\marginpar{#1 pt}\fi}
\def\min#1{\ifmin\marginpar{#1 min}\fi}
⟨/package⟩
⟨*ltxml⟩
⟨/ltxml⟩
```

```
⟨*package⟩
\AtEndDocument{\ifpts\message{Total: \arabic{pts} points}\fi
\ifmin\message{Total: \arabic{min} minutes}\fi}
⟨/package⟩
⟨*ltxml⟩
⟨/ltxml⟩
```

`\show@pts` The `\show@pts` shows the points: if no points are given from the outside and also no points are given locally do nothing, else show and add. If there are outside points then we show them in the margin.

```
⟨*package⟩
\newcounter{pts}
\def\show@pts{\ifx\inclprob@pts\@empty%
\ifx\problem@pts\@empty\else%
\ifpts\marginpar{\problem@pts pt\smallskip}\addtocounter{pts}{\problem@pts}\fi%
\fi\else%
\ifpts\marginpar{\inclprob@pts pt\smallskip}\addtocounter{pts}{\inclprob@pts}\fi%
\fi}
```

and now the same for the minutes

`\show@min`

```
\newcounter{min}
\def\show@min{\ifx\inclprob@min\@empty%
\ifx\problem@min\@empty\else%
\ifmin\marginpar{\problem@min min}\addtocounter{min}{\problem@min}\fi%
\fi\else%
\ifmin\marginpar{\inclprob@min min}\addtocounter{min}{\inclprob@min}\fi
\fi}
⟨/package⟩
```

3.5 Finale

Finally, we need to terminate the file with a success mark for perl.

```
⟨ltxml⟩1;
```