

`problem.sty`: An Infrastructure for formatting Problems*

Michael Kohlhasse
Jacobs University, Bremen
<http://kwarc.info/kohlhasse>

April 6, 2016

Abstract

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

Contents

1	Introduction	2
2	The User Interface	2
2.1	Package Options	2
2.2	Problems and Solutions	2
2.3	Starting and Stopping Solutions	4
2.4	Including Problems	4
2.5	Reporting Metadata	4
3	Limitations	4
4	The Implementation	5
4.1	Package Options	5
4.2	Problems and Solutions	5
4.3	Including Problems	8
4.4	Reporting Metadata	8

*Version v1.2 (last revised 2016/04/06)

1 Introduction

The `problem` package supplies an infrastructure that allows specify problem. Problems are text fragments that come with auxiliary functions: hints, notes, and solutions¹. 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

<code>solutions</code>	The <code>problem</code> package takes the options <code>solutions</code> (should solutions be output?),
<code>notes</code>	<code>notes</code> (should the problem notes be presented?), <code>hints</code> (do we give the hints?),
<code>hints</code>	<code>pts</code> (do we display the points awarded for solving the problem?), <code>min</code> (do we
<code>pts</code>	display the estimated minutes for problem soling). If theses are specified, then the
<code>min</code>	corresponding auxiliary parts of the problems are output, otherwise, they remain
	invisible.
<code>boxed</code>	The <code>boxed</code> option specifies that problems should be formatted in framed boxes
<code>test</code>	so that they are more visible in the text. Finally, the <code>test</code> option signifies that
	we are in a test situation, so this option does not show the solutions (of course),
	but leaves space for the students to solve them.
<code>showmeta</code>	Finally, if the <code>showmeta</code> is set, then the metadata keys are shown (see [Koh15]
	for details and customization options).

2.2 Problems and Solutions

<code>problem</code>	The main environment provided by the <code>problem</code> package is (surprise surprise) the <code>problem</code> environment. It is used to mark up problems and exercises. The
<code>id</code>	environment takes an optional KeyVal argument with the keys <code>id</code> as an identifier
<code>pts</code>	that can be reference later, <code>pts</code> for the points to be gained from this exercise in
<code>min</code>	homework or quiz situations, <code>min</code> for the estimated minutes needed to solve the
<code>title</code>	problem, and finally <code>title</code> 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.
<code>solution</code>	The <code>solution</code> environment can be to specify a solution to a problem. If the
<code>solutions</code>	<code>solutions</code> option is set or <code>\solutionstrue</code> is set in the text, then the solution
	will be presented in the output. The <code>solution</code> environment takes an optional
<code>id</code>	KeyVal argument with the keys <code>id</code> for an identifier that can be reference <code>for</code> to
<code>for</code>	specify which problem this is a solution for, and <code>height</code> that allows to specify the
<code>height</code>	amount of space to be left in test situations (i.e. if the <code>test</code> option is set in the
<code>test</code>	<code>\usepackage</code> statement).
<code>hint</code>	, the <code>hint</code> and <code>exnote</code> environments can be used in a <code>problem</code> environment to
<code>note</code>	

¹for the moment 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=elephants,pts=10,min=2,title=Fitting Elephants]
    How many Elephants 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=elephants,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 1 (Fitting Elephants)

How many Elephants 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 Formatted Problem from Figure 1

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

2.3 Starting and Stopping Solutions

Sometimes we would like to locally override the `solutions` option we have given to the package. To turn on solutions we use the `\startsolutions`, to turn them off, `\stopsolutions`. These two can be used at any point in the documents.

2.4 Including Problems

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.

2.5 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 allotted 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 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the `sTeX` GitHub repository [sTeX].

1. none reported yet

4 The Implementation

4.1 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.

```
1 <*package>
2 \newif\if@problem@mh@\@problem@mh@false
3 \DeclareOption{mh}{\@problem@mh@true}
4 \newif\ifexnotes\exnotesfalse
5 \DeclareOption{notes}{\exnotetrue}
6 \newif\ifhints\hintsfalse
7 \DeclareOption{hints}{\hintstrue}
8 \newif\ifsolutions\solutionsfalse
9 \DeclareOption{solutions}{\solutionstrue}
10 \newif\ifpts\ptsfalse
11 \DeclareOption{pts}{\ptstrue}
12 \newif\ifmin\minfalse
13 \DeclareOption{min}{\mintrue}
14 \newif\ifboxed\boxedfalse
15 \DeclareOption{boxed}{\boxedtrue}
16 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{omtext}}
17 \ProcessOptions
```

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

```
18 \if@problem@mh\RequirePackage{problem-mh}\fi
19 \RequirePackage{omtext}
20 \RequirePackage{comment}
21 \RequirePackage{mdframed}
22 \RequirePackage[base]{babel}
```

`\prob@*kw` For multilinguality, we define internal macros for keywords that can be specialized in *.ldf files.

```
23 \AfterBabelLanguage{ngerman}{\input{problem-ngerman.ldf}}
24 \def\prob@problem@kw{Problem}
25 \def\prob@solution@kw{Solution}
```

4.2 Problems and Solutions

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

```
26 \srefaddidkey[prefix=prob.]{problem}
27 \addmetakey{problem}{pts}
28 \addmetakey{problem}{min}
29 \addmetakey*{problem}{title}
30 \addmetakey{problem}{refnum}
```

Then we set up a counter for problems.

`\numberproblemsin`

```
31 \newcounter{problem}
32 \newcommand\numberproblemsin[1]{\@addtoreset{problem}{#1}}
```

`\prob@label` We provide the macro `\prob@label` to redefine later to get context involved.

```
33 \newcommand\prob@label[1]{#1}
```

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

```
34 \def\prob@number{\ifx\inclprob@refnum\empty%
35 \ifx\problem@refnum\empty\prob@label\theproblem%
36 \else\prob@label\problem@refnum\fi%
37 \else\prob@label\inclprob@refnum\fi}
```

`\prob@title` We consolidate the problem title into a reusable internal macro as well. `\prob@title` takes three arguments the first is the fallback when no title is given at all, the second and third go around the title, if one is given.

```
38 \newcommand\prob@title[3]{%
39 \ifx\inclprob@title\empty% if there is no outside title
40 \ifx\problem@title\empty{#1}\else{#2\problem@title{#3}}\fi
41 \else{#2}\inclprob@title{#3}\fi}% else show the outside title
```

With these the problem header is a one-liner

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

```
42 \def\prob@heading{\prob@problem@kw~\prob@number\prob@title{ }{ (}{)}\strut\\}%
43 \sref@label{id{\prob@problem@kw~\prob@number}}
```

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 whether the respective options are set.

`problem`

```
44 \newenvironment{problem}[1][\metasetkeys{problem}{#1}\sref@target%
45 \@in@omtexttrue% we are in a statement (for inline definitions)
46 \stepcounter{problem}\record@problem%
47 \def\current@section@level{\prob@problem@kw}%
48 \par\noindent\textbf{\prob@heading\show@pts\show@min\rm\noindent\ignorespaces}
49 {\smallskip}
50 \ifboxed\surroundwithmdframed{problem}\fi
```

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

```
51 \def\record@problem{\protected@write\auxout{%
52 {\string\@problem{\prob@number}%
53 {\ifx\inclprob@pts\empty\problem@pts\else\inclprob@pts\fi}%
54 {\ifx\inclprob@min\empty\problem@min\else\inclprob@min\fi}}}
```

`\@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).

```

55 \def\@problem#1#2#3{}

```

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

```

56 \srefaddidkey{soln}
57 \addmetakey{soln}{for}
58 \addmetakey{soln}{height}
59 \addmetakey{soln}{creators}
60 \addmetakey{soln}{contributors}
61 \addmetakey{soln}{srccite}
62 % \begin{macrocode}
63 % the next step is to define a helper macro that does what is needed to start a solution.
64 % \begin{macrocode}
65 \newcommand\@startsolution[1][\metasetkeys{soln}{#1}%
66 \@in@omtexttrue% we are in a statement.
67 \ifboxed\else\hrule\fi\smallskip\noindent{\textbf{\prob@solution@kw: }}\begin{small}%
68 \def\current@section@level{\prob@solution@kw}%
69 \ignorespaces}

```

`\startsolutions` for the `\startsolutions` macro we use the `\specialcomment` macro from the `comment` package. Note that we use the `\@startsolution` macro in the start codes, that parses the optional argument.

```

70 \newcommand\startsolutions{\specialcomment{solution}{\@startsolution}%
71 {\ifboxed\else\hrule\medskip\fi\end{small}}%
72 \ifboxed\surroundwithmdframed{solution}\fi}

```

`\stopsolutions`

```

73 \newcommand\stopsolutions{\excludecomment{solution}}

```

so it only remains to start/stop solutions depending on what option was specified.

```

74 \ifsolutions\startsolutions\else\stopsolutions\fi
75 \ifexnotes
76 \newenvironment{exnote}[1][\%
77 {\par\smallskip\hrule\smallskip\noindent\textbf{Note: }}\small}
78 {\smallskip\hrule}
79 \else\ifexnotes
80 \excludecomment{exnote}
81 \fi\ifexnotes
82 \ifhints
83 \newenvironment{hint}[1][\%
84 {\par\smallskip\hrule\smallskip\noindent\textbf{Hint: }}\small}
85 {\smallskip\hrule}
86 \newenvironment{exhint}[1][\%

```

```

87 {\par\smallskip\hrule\smallskip\noindent\textbf{Hint: }\small}
88 {\smallskip\hrule}
89 \else%ifhints
90 \excludecomment{hint}
91 \excludecomment{exhint}
92 \fi%ifhints

```

4.3 Including Problems

`\includeproblem` The `\includeproblem` command is essentially a glorified `\input` statement, it sets some internal macros first that overwrite the local points. Importantly, it resets the `inclprob` keys after the input.

```

93 \addmetakey{inclprob}{pts}
94 \addmetakey{inclprob}{min}
95 \addmetakey*{inclprob}{title}
96 \addmetakey{inclprob}{refnum}
97 \addmetakey{inclprob}{mhrepos}
98 \clear@inclprob@keys%initially
99 \newcommand\includeproblem[2] [] {\metasetkeys{inclprob}{#1}%
100 \input{#2}\clear@inclprob@keys}

```

4.4 Reporting Metadata

```

101 \def\pts#1{\ifpts\marginpar{#1 pt}\fi}
102 \def\min#1{\ifmin\marginpar{#1 min}\fi}
103 \AtEndDocument{\ifpts\message{Total: \arabic{pts} points}\fi
104 \ifmin\message{Total: \arabic{min} minutes}\fi}

```

`\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.

```

105 \newcounter{pts}
106 \def\show@pts{\ifx\inclprob@pts\@empty%
107 \ifx\problem@pts\@empty\else%
108 \ifpts\marginpar{\problem@pts pt\smallskip}\addtocounter{pts}{\problem@pts}\fi%
109 \fi\else% inclprob@pts nonempty
110 \ifpts\marginpar{\inclprob@pts pt\smallskip}\addtocounter{pts}{\inclprob@pts}\fi%
111 \fi}

```

and now the same for the minutes

`\show@min`

```

112 \newcounter{min}
113 \def\show@min{\ifx\inclprob@min\@empty%
114 \ifx\problem@min\@empty\else%
115 \ifmin\marginpar{\problem@min min}\addtocounter{min}{\problem@min}\fi%
116 \fi\else%
117 \ifmin\marginpar{\inclprob@min min}\addtocounter{min}{\inclprob@min}\fi%
118 \fi}

```


119 `</package>`