IDA User Input

Results

Dan Roozemond

In this document it is tried to give an overview of the work done by Dan Roozemond for IDA. The main goal was to give an easy-to-use tool that enables IDA programmers to obtain user input easily.

The results comprise two parts: A java package for the conversion of strings input by the user to OpenMath objects (Section 1), and an addition to the main stylesheet (/src/share/xdocs/xsl/webapp/content-0.xsl) for convenient usage of this java package (Section 2).

1 Java Package

The package is called `org.banaan.dan.userinput`. The four most important classes are the following:

- `OMConv`: The main class for converting strings to OpenMath objects. All converters (i.e. `OMConvInt`) that convert strings to specific kinds of OpenMath objects (i.e. `OMI`'s) extend this class. See below.
- `OMConverter`: A set of static functions to use OMConv and its extensions easily. See below.
- `ConvException`: An exception used if the conversion of a user input string fails.
- `Tests`: A class that tests the extensions of OMConv.

1.1 `OMConv`

This class has three functions:

- `abstract public OMObject to(String In) throws ConvException` , which performs the actual conversion,
- `public boolean is(String In)` , which checks if `In` could be converted to a specific type, and
- `abstract public boolean setOptions(String OptionName, String OptionValue)` , which sets certain options specific to some types. See below for a list of options.

1.2 `OMConverter`

This class has two public functions:

- `public static OMObject convert(String Type, String Options, String In) throws ConvException`, which performs the actual conversion. Options should be of the form `option1=value1|option2=value2|...`, and type should be one of `int`, `rat`, `posint`, `perm`, `list_*`, `str`.
- `public static boolean valid(String Type, String Options, String In)` .

1.3 `OMConvList`

This class may require a bit more explanation. For one, the default constructor of this class requires a base type, i.e. the type of the elements of the list. However, since the use of the `OMConverter` is strongly advised, the recommended manner of using this class is by specifying a type of the form `list_<basetype>` to `OMConverter`.

Options for the base type can be specified to the `OMConverter` by appending underscores to their name, for example:
`OMConverter.convert("list_int", "size=3,_range=-4,7","[1,9,6]")`
will try to convert `[1,9,6]` to a list of size 3, which succeeds, but then it will try to convert its elements to integers in the range $[-4, 7]$, which fails since $9 > 6$.

A matrix can be made for example as follows:
`ATest("list_list_int","matrix=1","[[6,5,42,3],[7,6,9,127],[7,5,4,3],[6,7,6,9]]");`

1.4 Options

The options will be discussed per class extending `OMConv`. It is a convention that specifying the empty string as the value for an option disables that option.

Type	Option name	Example	Note
int	range	range=-5,17	Upper and lower bound must be integers.
rat			
posint	range	range=6,20	Upper and lower bound must be positive integers.
perm	groupSize	groupSize=7	Mandatory!
list_*	size	size=7	Must be an integer.
	matrix	matrix=1	Must be either 0 or 1. If set, requires that all elements of the list have the same length.
str	maxSize	maxSize=7	Must be a positive integer.

2 Stylesheet Addition

An addition was made to `/src/share/xdocs/xsl/webapp/content-0.xsl`. This addition can be used as follows:

```
<userinput id="c1s2p2_alg1_exa1_ui1">
    <label>Geef eigen invoer</label>
    <submittitle>Verwerk nieuwe waarden..</submittitle>
    <columntitle>Nieuwe waarde:</columntitle>
    <variables>
        <cont:scope id="book/c1/s2/p2">
            <variable name="a" type="int">
                <option name="range" value="1,100"/>
            </variable>
            <variable name="b" type="int"/>
        </cont:scope>
    </variables>
</userinput>
```

The `submittitle` and `columntitle` may be omitted, default values are 'Submit' and 'New values:', respectively.

When parsing the .xml file where this piece of code occurs, the following happens:

- A new file is created, called `c1s2p2_alg1_exa1_ui1.mb`. This file contains a table with a row for every variable specified, each row containing an inputbox. See for example Figure 1.
- In the original file, a link to the new file is created, with link text equal to the value of `label`.

The user of Algebra Interactive could now click the link in the original file. The newly created file pops up, he or she can enter new values for the specified variables, and as soon as all input is correct the popped upped window disappears and the original page is reloaded, to enable it to process the new values.

3 Notes

- Because of a bug in the Mozilla engine, the newly created file cannot be closed programmatically. Instead, the focus is given back to the calling file, and the popped up window remains.

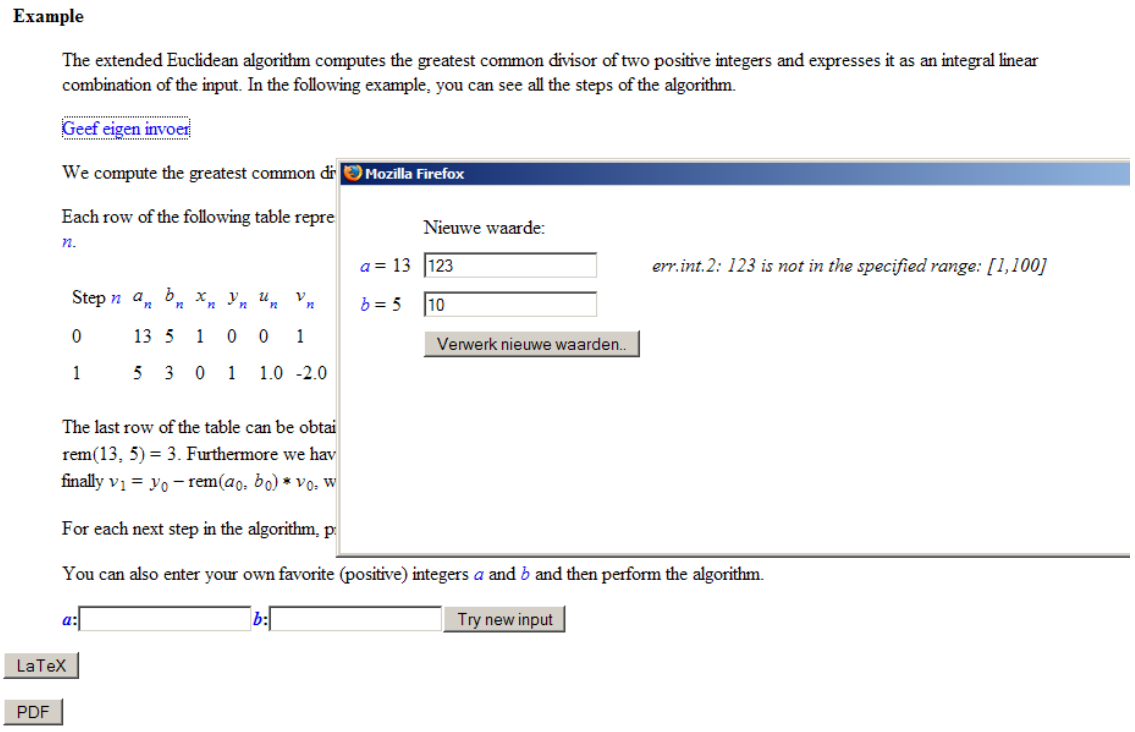

Figure 1: Example of the user input in action