



myCBR Project

myCBR's External Similarity Mode

November 29, 2007

Andreas Rumpf



status: draft

version: 1.0

author: Andreas Rumpf (Andreas.Rumpf@dfki.de)

date: November 29, 2007

myCBR is a project at DFKI

<http://www.dfki.de>

myCBR is open source, developed under the GPL license.

For further information visit:

<http://mycbr-project.net>

Contents

1	myCBR's External Interface	4
2	The interface in detail	5
3	Example: calling a Python script	6

1 myCBR's External Interface

This document explains how to use myCBR's external interface. The purpose of the external interface is to execute external programs which retrieve case or attribute values and output a similarity value. These programs may be any process of the operating system, which means that you can use your programming language of choice for implementing the external program.

The external interface is an inter process protocol much like CGI: During retrieval your program will be called for each case. The program receives two values via its command line arguments: The query value and the current case's value encoded as strings. The program computes the similarity between these two values and writes a value in the range [0; 1] on its standard output stream. This result value is to be encoded as a string too.

Because the interface is extremely slow, it is not advised to use it as a general way to extend myCBR's capabilities to express similarity measures. Use the scripting interface for that. The idea of this interface is to provide a means for computationally intensive similarity measures like comparing two pictures.

2 The interface in detail

The current interface only deals with local similarity measures. The following table lists example values and how they are encoded on the command line: string and symbolic values are always in quotes, unless it is a special value like `_undefined_`. Quotes, `\`, or non-ASCII characters in the a string or symbolic value are always escaped with `\`. These are the same rules as for many programming languages, so that your programming environment may already have a means to parse or evaluate the strings directly.

The other types are encoded as one would expect, again like in most programming languages. The query value is the first command line argument, the case value is the second command line argument.

Attribute type	example value	encoding on command line
Special	<code>_undefined_</code>	<code>_undefined_</code>
Integer	<code>-34</code>	<code>-34</code>
Float	<code>10e40</code>	<code>10e40</code>
Boolean	<code>true</code>	<code>true</code>
Boolean	<code>false</code>	<code>false</code>
String	<code>"hi", he said</code>	<code>"\"hi\", he said"</code>
Symbol	<code>symbolic value</code>	<code>"symbolic value"</code>

3 Example: calling a Python script

This example calls a Python script. Python has been chosen because it reads like pseudo-code. The implemented local similarity measure operates on strings and returns 1 iff the number of e's in the query is the same as the number of e's in the case (or any of them is `_undefined_`).

Listing `sim2.py`:

```
# Simple script that demonstrates myCBR's external interface

import sys

def sim2(query, case):

    # special sim: returns 1.0 if the number of 'a' is the same
    #                 in query and in case or if one is undefined;
    #                 0.0 otherwise

    if query == '_undefined_'
    or case == '_undefined_': return 1.0

    if len(query) > 0 and query[0] == '':
        q = eval(query)           # parse the string
    else:
        q = query
    if len(case) > 0 and case[0] == '':
        c = eval(case)           # parse the string
    else:
        c = case
    if q.count('e') == c.count('e'):
        # if the number of 'e's is the same
        return 1.0
    else:
        return 0.0

def main():
    print str(sim2(sys.argv[1], sys.argv[2]))

main()
```

Save the file somewhere (it is assumed that you call it `yourpath/sim2.py`). Now open myCBR and set a string attribute's similarity measure to *external*. Fill in the command field `python yourpath/sim2.py`. If the path (or the executable) contains spaces, you have to enclose it with " quotes:

```
python "your path with spaces/sim2.py"
```

It is not needed to fill in the working directory, unless your program needs it. You can now test the external command.

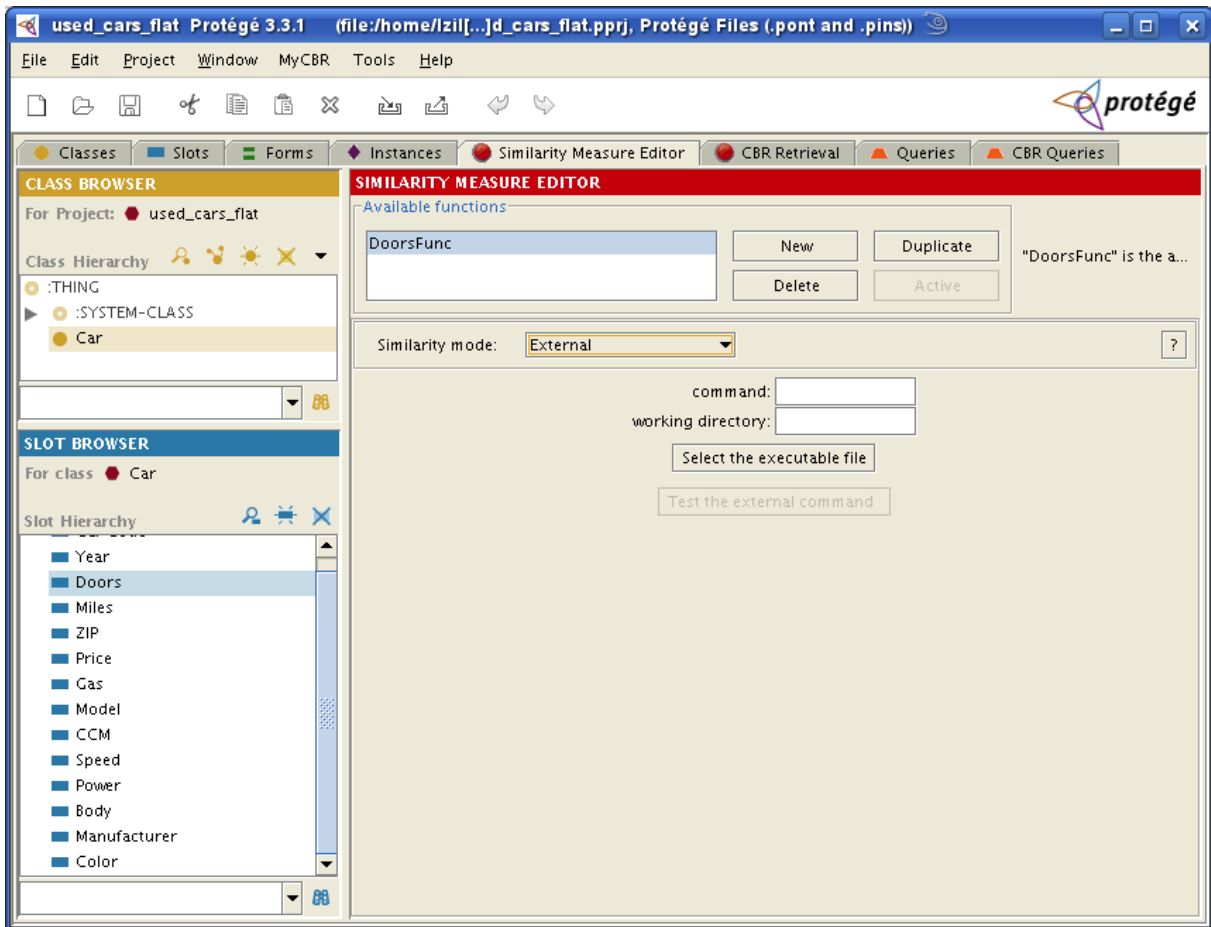


Figure 1: External Similarity Mode

The myCBR's *external* similarity mode can be chosen for each slot type (integer, float, string or symbol). Select a slot and then choose *external* as similarity mode.