Program Synthesis Model Finder (PSMF)

Installing and running procedures (version 2018-April)

Purpose

Program synthesis aims to mechanize the task of programming. We built a PSMF system to extend the synthesis competency of a general purpose imperative program synthesizer, which works on the integers' domain, accepting as specification sketches and a tiny set of user examples. The PSMF system embeds into the synthesizer some concepts of genetic algorithms and genetic programming's mutations. So far, the system has helped the synthesizer to perform better the synthesis task of programs that itself already had been able to and to discover new ones, reducing the user's intervention at different stages of the process.

For Windows Operating Systems

1. Unzip the psmf.zip file to any directory of your disk. For example to root directory C:/

```
[any folder, for example C:/]
    |
    |--- psmf/
    |      |
    |      |--- lib/
    |      |
    |      |--- alloyFiles/
    |      |       |
    |      |       |--- impMs/
    |      |       |
    |      |       |--- examples/
```

Files into those folders (table below):

Folder	Files	Description
psmf	PSMF.jar, README.TXT	PSMF program's main file, and read me file.
lib	AlloyAlsToXML.jar, hola-0.2.jar, jgap.jar	Mandatory libraries to execute PSMF program.
alloyFiles	imp.als	Main file of the Alloy* synthesizer.
impMs	semantics.als, syntax.als, wellF.als	Complementary files of the Alloy* synthesizer.
examples	p02max3.als, p02max3.txt, psmf.cfg	Alloy* model files (.als), test cases file (.txt), and configuration files (.cfg) have to be over here.

2. The current folder or the folder `../alloyFiles/` have to be in the Path Environment Variables. Here is a how to do that in windows 10! You will able to run PSMF in *Linux* or *MacOS*, just configuring the Path Environment Variables in such Operation Systems.
3. Setting up precondition and post-condition on an Alloy* model file. For example open up `p02max3.als` into `../psmf/alloyFiles/examples/` folder.
   3.1. Have a note that setting up precondition and post-condition contract might be not mandatory, its purpose is to constraint the search space. In most cases, setting up the scope of the input/output variables (see examples at Section 3.2) would be enough to able the synthesizer to find out a solution. In other cases the user may be leave precondition and post-condition empty.
   3.2. Write down sketch over the predicate `PreC` (or `posC`), for example:

```
pred PreC[iSt: one IState] {
  Assign1.rhs.name = InLoc1  //Sketch
  Assign2.rhs.name = InLoc2  //Sketch
  Assign3.rhs.name = InLoc3  //Sketch
}
```

4. Preparing Test Cases Dataset. Note that the `.txt` filename must have the same of the Alloy* model. For example for Max3 program synthesis, the chosen Alloy* model file was `p02max3.als`, so the test cases dataset file must be `p02max3.txt`. Then, open up the and fill it with any set o test cases you want (try to fill the file using input/output values that covers as many as possible cases for testing). Note that each line has four values separated by semi colon symbol and they mean, respectively: `expected output`; `first input variable value` (i.e. `x` variable); `second input variable value` (i.e. `y` variable); and `third input variable value` (i.e. `z` variable). The content of the `p02max3.als`, is shown as follows:

```
6;2;6;1
0;0;0;0
-1;-1;-1;-1
0;-1;0;-3
0;-1;-1;0
4;2;1;4
```

5. Preparing PSMF configuration file. Note that the `.cfg` filename can be any, for example `psmf.cfg`. Then, open up the file and fill it with parameters values you want to execute the task of synthesis. The file content is self explained. The content of the `psmf.cfg`, is shown as follows. It might be better open the `psmf.cfg` file on the `examples` folder copy and rename it to make any changes you want:

```
//Configuration file for Program Synthesis via Model Finder - PSMF system
//TO adjust System's parameters.
// A pair of slashes stands for comments.

//Inform the alloy file name (whitout extension name, for example:
   ok06max2).
InputFileName = p02max3;

//Inform the alloy file extension (including the dot symbol, for
   example: .als).
InputFileExtension = .als;

//Inform number of executions that the system will perform, for simulation
   purposes.
//For example: NumSysExec=5; means the user will call the system once, and
   it will perform 5 times.
NumSysExec=1;

//Distance Metric: explore all available [eAll] or read what is set into
   this configuration file [rFile]?
DistanceMetricExploreAllOrReadConfFile = rFile;

//The name of the distance metric: discrete | manhattan |
   euclidean_quadratic | hellinger | chebyshev | minkowski
DistanceMetricType=discrete;

//... there are a lot of parameter, please have a look at the configuration
   file in the examples folder.

//For all signature's scope values these values are accepted:
//  lower:upper bound value pair (for example 1:3 or any other zero or
   positive integer pair)
//  auto   (the system will define automatically)
//  no     (the system will not take account of such signature)
IntVar = 4:4; IntVal = auto; Assign = auto; SComp  = auto; CondS  = auto;
   While  = 0:0; Add    = 0:0; Sub    = 0:0; Mult   = 0:0; EQ     = 0:0; NEQ
   = 0:0; LEQ    = auto; GEQ    = auto; GTH    = 0:3;
//end of file.
```

6. Run the system typing java command line at `../psmf` folder. If you have memory issues, you may use the `-Xms` java parameter to allocate at least 1GB of RAM memory by typing `-Xms1g`:

```
java -jar -Xms1g "PSMF.jar"
```

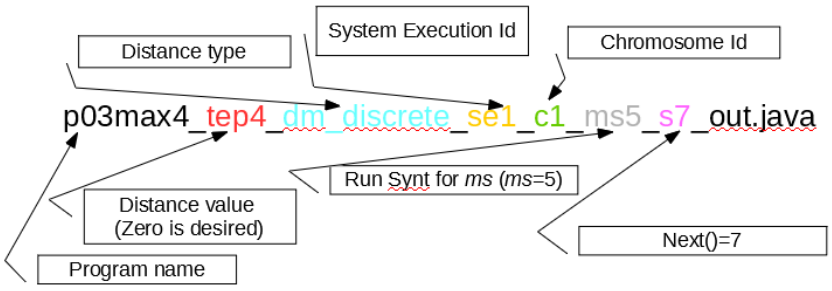
7. Then the PSMF system will ask you three questions: `sp01` just press enter if you do not want to change the default configuration folder name; `sp02` just press enter if you do not want to change the default configuration file extension name; and `sp03` just press enter if you do not want to change the default configuration filename.

```
sp01 -> Inform the config file's folder name. Press enter for default value
   [c:/data/psmf/alloyFiles/examples/]:
sp02 -> Inform the config file's extension name. Press enter for default
   value [.cfg]:
sp03 -> Inform the config file name. Press enter for default value [psmf]:
```

8. PSMF system's outcomes. The PSMF system used write out two kind of files:

Folder	File	Description
psmf	p02max3_tep0_dm_discrete_se1_c1_ms8_s1_out.java	program candidate synthesized source code.
examples	p02max3_out.als	Alloy* model file.

8.1. A set of Java program candidates synthesized source code – despite the source code, those filenames carry out the following information: (i) the input filename program, which was max3 (`p02max3`); (ii) the calculated distance between the program candidate's output and the test cases (`_tep`). As less the number after `_tep` as better, for example `_tep10` is better than `_tep63`. The `_tep0` (Zero) is the desired situation which means there were no errors (distance=zero); (iii) the name of the distance metric (`_dm_`), for example `_dm_discrete` means discrete metric was used to calculate the fitness function; (iv) The number of executions that the system will perform within the same session (`_se`), for instance `_se5` means the user did call the system once, and it was perform 5 times; (v) The chromosome id (`_c`), for instance `_c5`, means the program candidate was the chromosome number five; (vi) The max sequence (`_ms`), for instance `_ms8`, means the program candidate has used the synthesizer command using the max sequence equals to eight, i.e. to run the Alloy* synthesizer directly on the Alloy Analyzer interface the user have to execute the `run Synt for 8 but…` command; (vii) The subset (`_s`), for instance `_s3`, means the program candidate has used the `next` command equals to three, i.e. to run the Alloy* synthesizer directly on the Alloy Analyzer interface the user have to execute the `next` command three times after the `run Synt for 8 but…` command has been invoked.



8.2. The Alloy* model file – due the file `p02max3.als` does not have the `run Synt for…` command, this outcome file `p02max3_out.als` does. In other words, the PSMF system generates a text file with the Alloy* model implementation, where the `run Synt for` command (with all those signatures) are ready for execute directly on the Alloy Analyzer interface, if the user wants to.

9. Experimental benchmark results concerning 17 well-known programs using the parameters in the configuration files (`*.cfg`). We have used a notebook with Intel i7 processor, 8GB RAM, 256GB SSD and Windows 10 Home operating system. Table as follow shows the mean time of 10 runs and the number of candidate programs the system has tested before delivering a solution (# candidate program to find solution). The total time spent to find out a solution did range from 5.0 seconds (Max2) to 959.8 seconds (Fib) in our performed experiments. The *coefficients of variation* under 5% of their own mean time has shown that the experimental setup performance was homogeneous.

Id	Program	Mean time (seconds)	# candidate program to find solution
p01	Max2	5.0	1
p02	Max3	19.0	1
p03	Max4	155.6	32
p04	Maj3	12.0	2
p05	Maj5	56.4	6
p06	Add2	6.0	1
p07	Add3	11.0	1
p08	Add4	16.0	1
p09	Fib	959.8	6
p10	Mult	31.0	1
p11	IntSQRT	141.6	1
p12	Fact	276.2	1
p13	Modu	242.8	2
p14	SumSQR	73.0	9
p15	Median	23.0	1
p16	GCD	145.0	1
p17	Maj8	56.6	8