victorlei / smop

Small Matlab to Python compiler (Let APL be your guide and interpreter -- anonymous)

41 commits | **3** branches | **2** releases | **3** contributors

branch: **master** smop /

Merge pull request #27 from perimosocordiae/patch-1 ...

victorlei authored 3 months ago — latest commit 01ca5f5250

smop	fixing comment, numpy->np, adding * to dot conversion	3 months ago
.gitignore	ignore .egg-info	5 months ago
INSTALL	big bang	10 months ago
LICENSE.MIT	LICENSE.MIT	10 months ago
MANIFEST.in	big bang	10 months ago
README.rst	add .rest extension to readme for proper github parsing	5 months ago
setup.py	work in progress	9 months ago

Code
Issues 16
Pull Requests 1
Wiki
Pulse
Graphs
Network

README.rst

Summary

SMOP stands for Small Matlab/Octave to Python compiler. It is supposed to help those who attempt migration from Matlab to Python. Despite the similarities between the two languages, there are enough differences to make manual translation too time consuming.

SMOP is not a polished product, nor a replacement to Octave and Matlab. Taking into account its size (less than 3000 lines), this is not surprizing. There are no toolboxes. Small everyday functions (max, length, etc.) are recognized and supported, but that's all.

SMOP is written in Python, using PLY -- Python Lex/Yacc for lexical analysis and parsing, and numpy for runtime environment. SMOP is platform-independent, but is tested only on Linux. It is a command-line utility.

Example

It is possible to run an example without installing smop. Just unzip it somewhere, and cd there. In your current directory you will find a bunch of .py files and a file named fastsolver.m. It is taken from the winning submission to Matlab programming competition in 2004 (Moving Furniture http://www.mathworks.cn/matlabcentral/contest/contests/12/submissions/29989).

Now type python main.py fastsolver.m -o fastsolver.py. If you don't specify the output file with -o option, it is written to a.py. Each time a function is translated, its name is written.

```
lei@fuji:~/smop/smop$ python main.py fastsolver.m
fastsolver.m
        solver
        cbest
        mainsolver
        imoves
        easysolver
        localfiddler
        findoverlaps
        dijkstra
        improve
        TLL79
        solverA
        solver1
        movefrompos
        onemove
        solver2
        SearchPath
        Faster10IntReps2
        matrixsolver
        outoftheway
        ismember1
        ismember2
        setdiff
        unique
        sub2ind
        randperm
        perms
        itTakesAThief
        movefurniture
        findshortestpath
        dealWall1
lei@fuji:~/smop/smop$
```

The entire submission contains 2093 lines, and it is automatically translated to Python by smop. These are the good news. The bad news are that generating the code is not enough to run the program, so there are no performance numbers yet.

1. While the submission itself --- the solver program --- does not use graphics, the envelope code that is responsible to run the submission, collect and display the results, does. So about 100 lines of the envelope must be rewritten by hand.
2. Many standard functions are not yet implemented --- rand, find, and others. They are on the issues list.
3. Some matlab constructs, especially creating arrays by out of bound assignment, are used in the submission, but not yet supported by smop. Meanwhile, these lines should be rewritten in the Matlab code.

```
01 function moves=solver(A,B,w0)
02 [moves,optmove,optscore]=cbest(A,B,w0);
03 curscore=sum(w0(moves(:,1)));
04 lots=1;
05 if length(moves)-optmove<20||curscore/optscore<1.05
06     lots=2; return
07 else
08     lenw=length(w0);
09  [xx,nseq]=sort(rand(1,lenw));
10  A1=A;
11  B1=B;
12  w01=w0;
13  for i=1:lenw
14      A1(A==i)=nseq(i);
15      B1(B==i)=nseq(i);
16      w01(nseq(i))=w0(i);
17  end;
18  [moves2,optmove,optscore]=cbest(A1,B1,w01);
```

becomes

```
01  def solver(A,B,w0):
02  moves,optmove,optscore = cbest(A,B,w0)
03  curscore=np.sum(w0[(moves[:,0]-1)])
04  lots=1
05  if max(moves.shape) - optmove < 20 or curscore / optscore < 1.05:
06      lots=2
07      return moves
08  else:
09      lenw=max(w0.shape)
10      xx,nseq=sort(rand(1,lenw))
11      A1=A
12      B1=B
13      w01=w0
14      for i in range(1,(lenw+1)):
15          A1[A == i]=nseq[(i-1)]
16          B1[B == i]=nseq[(i-1)]
17          w01[(nseq[(i-1)]-1)]=w0[(i-1)]
18      moves2,optmove,optscore = cbest(A1,B1,w01)
```

Now some random notes.

1. Line 03. Functions vs. arrays ambiguity is correctly resolved: for example, sum is a function, but w0 and moves are arrays.
2. Line 09, Matlab function length is correctly inlined as max(w0.shape) --- that is the maximum of the array dimensions.
3. For some functions, such as abs, max, and others, there are both a builtin version and a different numpy version, and it is important to use the right one. In line 03 np.sum is used, but in line 09 builtin max. This is correct.
4. Line 10. Functions rand and sort are not yet implemented.
5. Lines 15-16. There is some support to boolean indexing.
6. Lines 15-17. Array subscripts are modified to start with zero index. Note that if all subscripts were decremented, it would break boolean indexing.
7. Line 13. Range specification in Matlab includes the upper bound. In Python, it doesn't. So for i=1:lenw becomes for i in range(1,(lenw+1)). Extra parentheses are known as issue #1.

The table below tries to summarize various features.

Implemented features	
Lexical and syntactical analysis	Mostly complete, including some weird Matlab features
Name resolution	For each occurrence of a variable, find a set of its possible definitions
Inlining of small functions	
Array subscripts translated from 1-based (Matlab and Fortran style) to 0-based (C and Python style)	Also, end subscript implemented
from:step:to translated to from:to:step	
Upper bound is n+1	

Unimplemented features	
Structs	To be implemented as soon as cc possible.

Arrays silently become C=style (rows first).	In some cases it may break the code. Not detected.
Function handles and lambda expressions	Handles break the heuristic that tells between function calls and array references.
Graphics,	Never
Auto-expanding arrays	Unlike other languages, matlab allows out-of-bounds assignment. As MathWorks tries to phase out this feature, there is a lot of legacy code depending on it.
Sparse matrices	Have good chances of being implemented, especially taking into account that scipy have several implementations to choose from.
Full support for boolean indexing. Currently, some expressions don't work	For example, x(x>0.5) = 1 works, but y=x>0.5; x(y)=1 does not work.
Command syntax	Too complex to support
Type, rank and shape inference	
Strings	