Artifact overiew

Anynomous

May 2019

1 Getting Started Guide

I use VirtualBox to install a virtual machine, ubuntu 64bits. I choose ubuntu-18.04.2-desktop-amd64.iso. The password for the machine is: 123456
From a standard ubuntu 18.04 virtual machine, go to terminal. Below is how to set up the virtual machine to run our type checker.

1. Install opam (version 1.2.2) : $ sudo apt install opam
2. $ opam init
3. $ eval ‘opam config env‘
4. Choose OCaml 4.05.0: $ opam switch 4.05.0
5. Update the PATH: $ eval ‘opam config env‘
6. $ opam depext camlzip.1.07 conf-autoconf.0.1 conf-gmp.1
7. Install why3 (version 1.0.0)$ opam install why3
8. Install Alt-ergo prover (version:2.0.0):$ opam install "alt-ergo=2.0.0"
9. Make sure alt-ergo is installed: $ why3 config –detect
10. Install other necessary dependency $ opam install core oUnit sedlex

After the setup, we can now go to our type checker directory BIAREL.
Compile the type checker under the directory : BIAREL $: make
Make creates a binary executable called 'biarel'.
The source codes are in the folder 'src'.
The examples are in the folder 'examples'.
Folder example/binary contains the relational cost analysis examples.
Folder example/unary contains the unary cost analysis exampless .

Give examples to use our type checker for unary examples(map), use flag '-u' :
$ ./biarel -u examples/unary/monad_map_max.br

Give examples to use our type checker for relational examples(map):
$ ./biarel examples/binary/Amonad_map.br

Use flag '-ht' to modify the SMT solver timeout limit. The default is 1 second. For complicated examples such as insertion sort (iSort), we need to set

'-ht':

```
$ ./biarel -ht 6 examples/binary/Amonad_iSort.br
```

How to run benchmarks.
1. Go back to the top level directory of the type checker where you can find a Makefile $ make test
2. Run benchmarks : $ ./test.byte .

We test all the unary examples and all the binary examples except iSort, insert and shift which need a different '-ht' flag.

2 Step by Step Instructions

I list the structure of our source code in the /src folder.

biarel.ml - the main file
parser.ml - parser of ARel
lexer.mll - lexer of ARel
print.ml - pretty print
whySolver.ml - use why3 ask alt-ergo
binary.ml - relational type checking
unary.ml - unary type checking

I list all the examples we present in our paper and how to reproduce these examples using our type checker.
Below are the experiment results as well as the command I use to run the examples on the ubuntu virtual machine.

Benchmark	Total Time	command
map(1)	2.15s	./biarel example/binary/Amonad_map.br
map(2)	2.57s	./biarel example/binary/Amonad_map2.br
boolOr	3.98s	./biarel example/binary/Amonad_boolOr.br
separate	4.28s	./biarel example/binary/Amonad_seperate_pr_full.br
loop	3.90s	./biarel example/binary/Amonad_loop.br
FFT	7.76s	./biarel example/binary/Amonad_fft.br
Search	15.85s	./biarel example/binary/Amonad_nss_search.br
NSS	28.32s	./biarel example/binary/Amonad_nss.br
shift	6.08s	./biarel -ht 4 example/binary/Amonad_shift_dp.br
insert	7.88s	./biarel -ht 4 example/binary/Amonad_insert.br
iSort	39.11s	./biarel -ht 6 example/binary/Amonad_iSort.br
merge(1)	4.07s	./biarel example/binary/Amonad_merge.br
merge(2)	4.35s	./biarel example/binary/Amonad_merge2.br
sam	1.42s	./biarel example/binary/Amonad_sam.br
comp	1.74s	./biarel example/binary/Amonad_comp.br

Table 1: Experimental results on virtual machine