SMT Lab
Formal Techniques Summer School, May 23 – 27, 2016

1. First you will need to check out and build a specially modified version of CVC4. The steps are as follows:

   (a) You will need a unix-like system with the following tools installed:

      i. autoconf
      ii. automake
      iii. boost
      iv. gmp (rpm packages are gmp, gmp-devel, gmp-static, debian package is libgmp3-dev)
      v. libtool

   (b) Create an account on github.com if you don't have one.

   (c) Execute the following commands to build cvc4:

```
> git clone https://github.com/barrettcw/CVC4.git
  (Enter your github credentials when prompted)
> cd CVC4
> ./contrib/get-antlr-3.4
> ./autogen.sh
> ./configure --with-antlr-dir=‘pwd‘/antlr-3.4 \
   ANTLR=‘pwd‘/antlr-3.4/bin/antlr3 --enable-static \
   --disable-shared --enable-static-binary --with-build=debug
> make -j 4
```

   (d) There are a few simple difference logic examples in CVC4/lab. There is the example from the lecture, a version of the example from lecture in which all constraints have already been rewritten into less-than or equal constraints, and a simple benchmark from the SMT-LIB library. Try running them:

```
> ./builds/bin/cvc4 lab/example.smt2
> ./builds/bin/cvc4 lab/example_rewritten.smt2
> ./builds/bin/cvc4 lab/jobshop2-2-1-1-4-4-11.smt2
```

   (e) Your task is to complete a theory solver for integer difference logic that can successfully solve these problems. A sketch of the solution is in CVC4/src/theory/idl/theory_idl.cpp. Read through this code carefully and look for the two TODO comments.

      i. To run cvc4 using this code, use:

```
> ./builds/bin/cvc4 --use-theory=idl lab/example.smt2
> ./builds/bin/cvc4 --use-theory=idl lab/example_rewritten.smt2
> ./builds/bin/cvc4 --use-theory=idl lab/jobshop2-2-1-1-4-4-11.smt2
```

ii. Notice that the first and last examples fail with an assertion failure. This is because of the first TODO: not all of the rewrites have been implemented, so the assertion that expect all constraints being checked to be less-than or equal constraints fails. To fix this, implement the code at the first TODO.

iii. Notice that the second example, which is already rewritten, simply gives the wrong answer (sat instead of unsat). This is because the negative-cycle detection is not implemented. This is the second TODO.

iv. There are several Debug aids in the code. To enable them, run with "-d theory::idl" or "-d theory::idl::printmatrix" etc.

2. If you complete the base task, here are some ideas to improve things further

   (a) How large are the problems you can solve? Try larger examples from the QF_IDL directory in lab (these are from the SMT-LIB benchmark library). Note that they range from easy to extremely difficult - some of the job shop examples cannot be solved by the default version of cvc4 in a reasonable amount of time.

   (b) To improve performance, you can try the following:

      i. Rerun the configure and make steps above, but leave out "–with-debug" in the configure line.

      ii. Implement better conflict explanations (currently all theory literals are used in constructing the conflict).

      iii. Implement theory propagation.

      iv. Make the negative-cycle check more incremental.

      v. What other ideas do you have? Can you beat the default cvc4 solver?