Programming Assignment 1

Submission Guidelines

Please send a pull request to the branch named with your student ID during the submission periods (see below).
For Parts 1 and 2, please put the required items under `./lsv/pa1`, i.e., this folder.
For Part 3, please develop your code under `./src/ext-lsv`.

Submission Periods

- Parts 1 and 2: 2020/10/15 11:00-13:00
- Part 3: 2020/10/29 11:00-13:00

1. [Using ABC]

(10%)
(a) Use BLIF manual to create a BLIF file representing a four-bit adder.
(b) Perform the following steps to practice using ABC:

- read the BLIF file into ABC (command `read`)
- check statistics (command `print_stats`)
- visualize the network structure (command `show`)
- convert to AIG (command `strash`)
- visualize the AIG (command `show`)
- convert to BDD (command `collapse`)
- visualize the BDD (command `show_bdd`; note that `show_bdd` only shows the first PO; command `cone` can be applied in combination to show other POs)

Items to turn in:

1. The BLIF file.
2. Results of `show` and `show_bdd`.

2. [ABC Boolean Function Representations]

(10%)
In ABC there are different ways to represent Boolean functions.
(a) Compare the following differences with the four-bit adder example.

1. logic network in AIG (by command `aig`) vs. structurally hashed AIG (by command `strash`)
2. logic network in BDD (by command `bdd`) vs. collapsed BDD (by command `collapse`)

(b) Given a structurally hashed AIG, find a sequence of ABC commands to covert it to a logic network with node function expressed in sum-of-products (SOP).

3. [Programming ABC]

(80%)

Write a procedure in ABC to print the unate information for each node, whose function is expressed in the SOP form, in a given Boolean network. Integrate this procedure into ABC, so that running command `lsv_print_sopunate` will invoke your code.

We say that a variable *x* is *positive unate* (respectively *negative unate*) in the SOP expression *F* of a function *f* if all its appearances in *F* occur in the form of the positive literal *x* (respectively *negative literal x'*). Moreover, a variable *x* is called *binate* in *F* if both of its positive and negative literals appear in *F*.

As an example, for node `n1` with its function expressed in the SOP formula

ab'c + acd + a'b'd,

your program prints:

```
node n1:
+unate inputs: c,d
-unate inputs: b
binate inputs: a
```

To ease the correction of your code, please print the nodes in the order of `Abc_NtkForEachNode()`. Please print the names of inputs returned by function `Abc_ObjName()`, and sort the inputs in an increasing order with respect to their object IDs returned by function `Abc_ObjId()` in a line. If there is no satisfying input, please do not print an empty line. That is, for example, if there is no binate input of a node, do not print such line

```
binate inputs:
```