Automata Theory and Computability

Programming Assignment

Submission deadline : 11:59pm, 18/10/2021

1 Create the automaton for a quantifier-free Presburger formula

- Construct an automaton that accepts all the satisfying assignments of a given Presburger logic formula using the inductive procedure described in class. Show the automaton after each inductive step.
- Use python as the programming language.
- Input to the program is a Presburger logic formula f, the number of variables n in f, and n decimal values. Variable names will be x1,x2,...,xn.
- Output : automaton after each inductive step and accept or reject.
- Example-input: $x1 + x2 \leq 5 \quad 2 \quad 3 \quad 1$
  where $x1 + x2 \leq 5$ is the input formula, 2 is the number of variables, 3 and 1 are input values (x1=3 and x2=1).

Guidelines:

- The input formula will be given as a z3 formula (z3 is a SMT solver). To solve the problem, one will have to parse the input formula.
- z3 formulas are structured as trees, with each node (called declaration) being an operator such as AND,OR,NOT, ==, $\leq$ etc. The children of the node (called as arguments) are z3 formulae again.
- for a formula f, method f.decl().name() returns the node for the formula f as a string.
- the method f.num_args() returns the number of arguments for the formula f.
- the method f.arg(i) returns the ith argument of f.

The strategy will be as follows:

- first, write a function that takes as input an atomic formula a and returns the table for the automaton which accepts all satisfying assignments of a. **Assume that the atomic formulae are of the form $\sum_{i=1}^{i=n} a_i x_i \sim c$ where a_i's and c are constants, x_i's are variables, $\sim$ is $==, \leq$.**

- second, write a function that take as input tables (automata) corresponding to two formulas f_1 and f_2, and returns a table which corresponds to the automaton accepting all satisfying assignments of $f_1 \wedge f_2$

- third, write a function that takes as input the table (automaton) corresponding to a formula f and returns a table which corresponds to the automaton accepting all satisfying assignments of $\neg f_1$.

- Now using the above, write a function that takes as input a quantifier-free Presburger Logic formula and outputs the automaton for the formula at each inductive step.

- Write another function that takes n decimal values (you need to convert decimal values to binary before feeding to the automaton) as input and outputs accept or reject by the final automaton.

Now to construct the automaton for atomic formulae, one will have to compute the weigthed sums of the LHS. To do this, the following will help:

Suppose f is an expression (i.e. a term in Presburger Logic) over variables $x_1, x_2, .., x_n$. We want to compute the value of f under the assignment $[x_1/t_1, \ldots, x_n/t_n]$. To do this, parse the formula tree for f and replace each occurence of x_i with the number t_i and get a new expression g. Now to evaluate the value of the expression, use the eval method.

Example: $f = 2 \times x1 - x2 - x3$ to be evaluated at $[x1/1, x2/0, x3/1]$. Construct $g = 2 \times 1 - 0 - 1$. **eval(str(g))** will output 1.

Submission instructions:

- Create a private GitHub repository named studentname_ATC_2021_Assignment.

- Give access to the following usernames: NabarunDeka and habeebp098

- We will be monitoring your commits and the final submission.

- Keep committing your progress at regular intervals, if possible.

If you have any questions, contact Nabarun and Habeeb.

- You can use the prettytable library to display the tables.
- Prettytable example code

```
print("Step 0: x1<=2")
t = PrettyTable(['State', '0', '1'])
t.add_row(['2IF', '1','0'])
t.add_row(['1F', '0', '0'])
t.add_row(['0F', '0', '-1'])
t.add_row(['-1','-1','-1'])
print(t)
```