MLTP 2025 Programming Assignment

Given an EUF formula with multiple arity functions, check Satisfiability using Ackermann's reduction.

Implement this in Python 3.12.3.

Z3 Solver has a Python interface.

Even though you can use Z3 as Propositional Logic solver (SAT Solver), you have to implement the EUF solver by yourselves using the Ackermann's reduction.

Input is given as a z3py expression. (You don't have to read from text file).
Output will say if the given EUF formula is SAT or UNSAT.

Following are 2 examples of possible input z3 expressions:

```
S = DeclareSort('S')
f = Function('f', S, S)
x = Const('x', S)
formula1 = And(f(f(x)) == x, f(f(f(x))) == x)
formula2 = And(f(f(x)) == x, f(f(f(x))) == x, f(x) != x)
```

It is possible to just use solve(formula1) to get a solution (or "no solutions", in the case of unsatisfiable formulae such as formula2), but obviously that goes against the point of the project; you must implement the Ackermann's reduction yourself.

Formulae/terms in z3 are trees, so one can use various functions to help traverse through the operators and terms present in it. See following:

```
print("num args: ", formula1.num_args())
print("children: ", formula1.children())
print("1st child:", formula1.arg(0))
print("2nd child:", formula1.arg(1))
print("operator: ", formula1.decl())
print("op name:  ", formula1.decl().name())
```

In order to get the arity of a function, you can use f.arity(). Declaration of n-ary function is done as follows:

```
f = Function('f', S1, S2, S3, …, Sn, R)
```

where `Si` is the sort of the i^{th} input term of `f`, while `R` is the sort of the output term.

(Part 1) Ackermann's reduction:

Given a conjunction of EUF equalities, return SAT or UNSAT.

(Part 2) DPLL(T):

1. Convert the given formula into a corresponding proportional formula B.
2. Call z3 to check if B is satisfiable.
3. Use the solution to determine the satisfiability of the equality and disequality classes using the Ackermann's reduction.

Look up DPLL(T) algorithm from the KS book. Here is a brief summary:

DPLL(T): Convert the given EUF formula φ to a PL formula e(φ) by replacing equalities with new propositional variables. Check for satisfiability using a PL solver. If UNSAT, return UNSAT. If SAT, see if the assignment corresponds to a satisfiable equality assignment (for example using the induced equality graph). If it does, return SAT; else add a clause to e(φ) corresponding to the conflicting equality assignments, and repeat.

Format:

Code your implementation in a file named: **euf_ackermann.py.**

In that file, implement the following two functions:

- **def euf_solver_ackermann(euf_formula_list)**
  which returns either **z3.sat, z3.unsat or z3.unknown**
- **def euf_dpllt_solver(general_euf_formula)**
  which returns either **z3.sat, z3.unsat or z3.unknown**

You can have other functions / subroutines etc within the file.

1. Print the PL formula that you construct initially and also whenever you modify it.
2. Print the clause that is added

Utility functions to print the formula are provided.

Note:

Use Python 3.12.3. We will be evaluating your code by running it on python-3.12.3

Deadline is on **April 21st (Monday) at 23:59pm**. Commits made after this time will not be considered for evaluation.

Helpful Links:

https://ericpony.github.io/z3py-tutorial/guide-examples.htm

https://theory.stanford.edu/~nikolaj/programmingz3.html