CMSC 330: Organization of Programming Languages

Project 3/4 – Boolean Formulae & SAT

Project 3 Overview

- Simple OCaml programming exercise
  - prod, add_tail, index
  - Work with lists
  - unzip
  - Work with tuples
  - app_int
  - Use higher order functions
- Write code from scratch
  - Do not use library functions
  - Pervasives, List, Array, etc...
  - testUtils.ml provides utilities to print values, lists

Project 4 Overview

- Represent & solve boolean formulae
  - Manipulate & solve boolean formulae
    - Values - True, False
    - Operations - Not, And, Or, Forall, Exists
    - Variable assignments - lists of 2-tuples ('$a', true)
    - Determine satisfiability of boolean formulae
  - Represent & manipulate integers as boolean formulae
    - 1 bit - formula evaluating to True or False
    - N-bit integer - list of n formulae
    - Addition, multiplication - operation on lists of formulae
  - Represent magic square as boolean formulae
    - Using addition, comparison

Starting OCaml Code – boolean.ml

- Type formula
  - Represents boolean formulae
  - type formula = False | True | Var 'x' | And (f1, f2) …
- Additional types
  - formula list
    - [True; False; Var ’x’; And (f1, f2); ...]
  - formula list list
    - [[True; False; Var ’x’; And (f1, f2); ...]; …]

Project 4 Notes

- Distinguish between bool & formula
  - bool (native data like int, float)
    - true, false
  - formula (user-defined data type)
    - True, False, Var ’x’, And (f1, f2)…
- Additional types
  - formula list
    - [True; False; Var ’x’; And (f1, f2); ...]
  - formula list list
    - [[True; False; Var ’x’; And (f1, f2); ...]; …]

Project 4 Notes

- Operations on formula
  - Construction (recursively using constructors)
    - True, False, And (True, False), Forall (’x’, f), Exists (’x’, f)
    - And (f1, f2, And (f3, And (f4, f5)))
    - And (And (And (f1, f2), f3), f4), f5)
  - Evaluation
    - formula & assignment -> bool // eval
  - Satisfiability
    - formula -> assignment // sat
  - Simplification
    - formula & assignment -> formula // subst
**Project 4 Notes**

- **Binary numbers**
  - Treat booleans as bits
    - true = 1, false = 0
  - Numbers are just lists of bits
    - Least significant bit on left
    - Examples
    - 1 = [true]
    - 2 = [false; true]
    - 3 = [true; true]
  - Many possibilities for zero
    - [], [false], [false; false], [false; false; false]...

- **Can make use of OCaml libraries**
  - Pervasives - basic library functions
    - Comparisons, integer, boolean, bitwise, conversion, etc...
    - [1;2] @ [3;4] = [1;2;3;4]
    - "foo" ^ "bar" = "foobaa"
  - List - list manipulation
    - List.length
    - List.map
    - List.assoc
    - Operate on associative lists (lists of pairs), i.e., maps
  - Char - characters
    - Char.escaped c = "c"
    - Converts char to 1-character string

**Project 4 Notes**

- **Project files**
  - boolean.ml → your code. Make all your edits here
  - public_*_.ml → public test cases
  - public_*_.out → expected output for public test cases
  - myTest.ml → make up your own test cases here

- **Testing**
  - ocaml boolean.ml → test for syntax / type errors
  - ocaml public_*_.ml → run public test, compare outputs
  - ruby goTest.rb → run all public tests