Homework Assignment 6
Total Points: 100

Professor Karem A. Sakallah

EECS 598-002: Formal Verification of Hardware & Software Systems

Assigned: March 19, 2024
Due: March 26, 2024

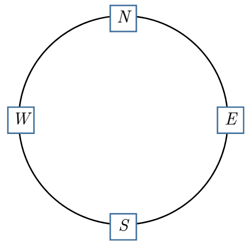
Guidelines

- The College of Engineering Honor Code applies to all work in this course.
- The due date is firm. Follow submission instructions (at the end).

Objectives

This assignment involves the use of Microsoft's Z3 SMT solver by encoding and checking the satisfiability of formulas written in the SMT-LIB language.

1 [Seating Assignments (50 Points)]



At an awards ceremony celebrating excellence in undergraduate research projects, a table that can seat four people has been reserved for two students and their two faculty advisors. As depicted in the figure, the chairs around this table have been labeled with the four compass directions: N (north), E (east), S (south), and W (west). Jill's project was supervised by Prof. Cabello, and Mei's project was done under Prof. Patel.

Being a formal event, the seating arrangement required that the student must sit to the immediate right of the professor who supervised the project. Thus, Jill would sit to the immediate right of Cabello and Mei to the immediate right of Patel. Let C, J, M, and P stand, respectively, for Cabello, Jill, Mei, and Patel, and let $X \in \{N, E, S, W\}$ and $Y \in \{C, J, M, P\}$. Thus, X stands for a chair and Y stands for a person. Now define the predicate is-chair-of(X, Y) to be true if X is the chair where person Y is sitting and to be false otherwise.

Your task is to **find all seating assignments that satisfy these requirements** by writing an smt2 specification of the problem and solving it with Z3. There are several approaches to express these requirements including:

- Declaring the four people and four directions as constants of sort Int.

- Declaring two uninterpreted sorts `person` and `chair` and declaring appropriate constants in these two sorts.
- Declaring new enumerated types using the syntax
  "(declare-datatypes () ((fruit apple orange pear banana)))"

  where fruit is the type name and `apple`, `orange`, `pear` and `banana` are its constants.

Explore these options and submit your solution only using the option you decided to use along with an explanation of why, in your opinion, this was the best option for you. You need to submit two files:

- `seating.smt2`
- `seating.solution`

Full credit will be given only if Z3 runs without errors on your smt2 file and all solutions to the problem are produced. Your smt2 file should also be well-written and include sufficient documentation (comments) to make it easy to understand. You should also use echo commands to document the generated output (which should be saved in `seating.solution`). For example, after Z3 returns a model you can echo the corresponding seating assignment as a 4-letter string starting with the N chair and proceeding in clock-wise order. For example the string $CMPJ$ would indicate that C is in the N chair, M in the E chair, etc.

2 [Bit Vector Manipulations] (50 Points)

Extraction and concatenation operations on bit vectors are quite common in the description of hardware designs. Unlike arithmetic operators that perform complex numeric transformations on bit vectors, extraction and concatenation operators simply re-arrange the "bits" and abstracting them to EUF logic tends to lead to many unnecessary refinement lemmas that dominate the run time of Averroes (AVR's predecessor). The data in Table 1 show that almost 97% of refinement lemmas in a corpus of 211 large industrial designs involve extraction and concatenation of bit vector fields[1].

Table 1: Number and proportion of different categories of data refinement lemmas.

Type	Example (Corresponding Verilog expression)	# Lemmas	Proportion (%)
Constant	3'd6 != 3'd0	21	0.65
Bit-select	{a, b, 1'b[0]	2554	78.97
Part-select	!(m = n) || $(\{b, m, c\})[5:3] = \{d, e, n, f\}[5:3]$	577	17.84
Others	8'd0 + 8'd1 + 8'd1 = 8'd2	82	2.54
Total		3234	100

To drastically reduce these unnecessary refinemnets, Averroes pre-processes the design to minimize the number of extractions and concatenations. AVR goes futher by partially interpreting bit-field operations. In this problem you are asked to verify that such transformations preserve bit vector semantics.

[1] Suho Lee, "Unbounded Scalable Hardware Verification," PhD Thesis, CSE, UM, August 2016.

2.1 [Concat/Extract Optimization](20 Points)

The Verilog code fragments below show a portion of a design with three 16-bit outputs w1, w2, and w3, that are derived from six 8-bit inputs w4, w5, w6, w7, w8, and w9 by a sequence concatenations and extractions. Note that concatenation is indicated by comma-separated bit vectors enclosed in curly brackets.

Describe these two fragments using the bit vector theory of SMT-LIB and use Z3 to prove that they are equivalent.

```
wire [15:0] w1, w2, w3;
wire [7:0] w4, w5, w6, w7, w8, w9;
wire [15:0] t1;
wire [31:0] t2;
wire condition;
assign t1 = condition? {w6, w7} : {w8, w9};
assign t2 = {w4, t1, w5};
assign w1 = t2[31:16];
assign w2 = t2[23:8];
assign w3 = t2[15:0];
```

Original

```
wire [15:0] w1, w2, w3;
wire [7:0] w4, w5, w6, w7, w8, w9;
wire [7:0] t3;
wire [7:0] t4;
wire condition;
assign t3 = condition? w6 : w8;
assign t4 = condition? w7 : w9;
assign w1 = {w4, t3};
assign w2 = {t3, t4};
assign w3 = {t4, w5};
```

Optimized

2.2 [Pushing Out Concatenations] (30 Points)

In the following optimizations, concatenation is "pushed out" as much as possible[2]. Use Z3's bit vector theory to prove that this transformation preserves logical equivalence.

- $\{6_3, A_2, B_3\}[5:2] = \{0_1, A_2, B_3[2]\}$.
- $\{6_3, \{3_2, A_2\}, B_3\} = \{27_5, A_2, B_3\}$.
- $\{6_3, \{3_2, A_2\}, B_3\}[8:4] = \{11_4, A_2[1]\}$.

Submission Instructions

1. Create a directory named ¡your uniquname¿ hw6 containing all of your work. This should include all files created by you as well as the Z3 output. The narrative parts of your submission should be typed and saved as PDF files. Handwritten and scanned documents or non-PDF documents ARE UNACCEPTABLE AND WILL NOT BE GRADED.
2. Zip the entire directory using "zip -r ¡your uniquename¿ hw6.zip ¡your uniquname¿ hw_6" and upload to Canvas.

[2] Aman Goel, "From Finite to Infinite: Scalable Automatic Verification of Hardware Designs and Distributed Protocols," PhD Thesis, CSE, UM October 2021.