Model checking with SPIN

Software Analysis, Spring 2024

Luca Di Bello

May 11, 2024

Contents

1 Introduction

In this assignments examines the implementation of model checking using the SPIN tool to verify the correctness of two versions of a frequency counter program: one sequential and the other parallel. Model checking is a technique that allows to verify the correctness certain properties of a system described in a finite-state model. In the following sections will be discussed how the program has been

modeled using ProMeLa language, which Linear Temporal Logic (LTL) properties have been defined to verify the correctness of the program, and how the verification has been performed using SPIN.

2 ProMeLa Model

The ProMeLa model consists of two main processes: the first process handles the sequential computation of frequency counts, storing results in an array `sequential_counts`. The second process on the other hand, initiates parallel computation, spawning worker processes for each possible value in the input array. These workers update an array `parallel_counts` concurrently. Race conditions are avoided as each worker updates a unique position in the array.

As explicitly stated in the assignment, the ProMeLa model presents two constrants:

1. `MAX`: It represents the maximum value that can be assigned to an element in the array. Used while filling the input array with random values.
2. `LENGTH`: the length of the input array.

The model presents an `init` block that initializes the input array with random values between 0 and `MAX` and starts both the sequential and parallel processes. The code is available in listing 1.

```
// Define the maximum number of elements in the array
#define MAX 2
#define LENGTH 2

// Define the variables
int a[LENGTH];

init {
        // Initialize the array non-deterministically
  printf("Random state:\n")
        int i;
        for (i : 0 .. LENGTH - 1) {
                // Select a random value for the array
                int v;
                select(v : 0 .. MAX);
                // Assign the value to the array
                a[i] = v;

    // Print the value
    printf("\ta[%d] = %d\n", i, v);
        }

        // Run the sequential version of the program
  printf("Running sequential version...\n");
        run sequentialCounter();

  // Run the parallel version of the program
  printf("Running parallel version...\n");
  run parallelCounter();
}
```

Listing 1: ProMeLa array intiialization and start of sequential and parallel processes

Things to add to this section:

- Differences between ProMeLa and Java program

  To join processes, we use a channel!

  We start processes right away rather than create first and start later all together

  Same model comprehends bot sequential and parallel versions. In Java we had two separate classes.

3 LTL Properties

- Explain how I implemented the LTL properties to check completition.
- Explain the two additional LTL properties (1 must work, 1 must is not verified)

4 Verification with SPIN