SPINLOCK

В папке SpinLock/ лежат файл с описанием логики(spinlock.tla) работы и скрин с толбокс, что лок работает.

```
class SpinLock {
    std::atomic_flag locked = ATOMIC_FLAG_INIT ;
public:
    void lock() {
        while (locked.test_and_set(std::memory_order_acquire)) { ; }
    }
    void unlock() {
        locked.clear(std::memory_order_release);
    }
};
```

Инвариант:

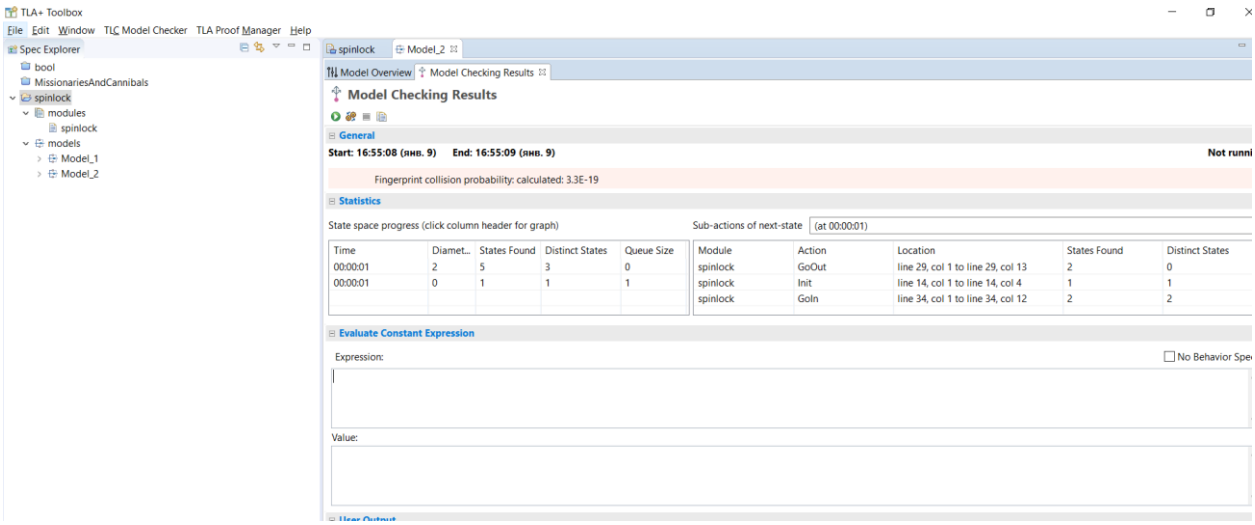
TypeOK == \/ (who_is_in_critical \subseteq { first } /\ flag_locked \subseteq {"T"})

\/ (who_is_in_critical \subseteq { second } /\ flag_locked \subseteq {"T"})

\/ (who_is_in_critical \subseteq { no_one } /\ flag_locked \subseteq {"F"})

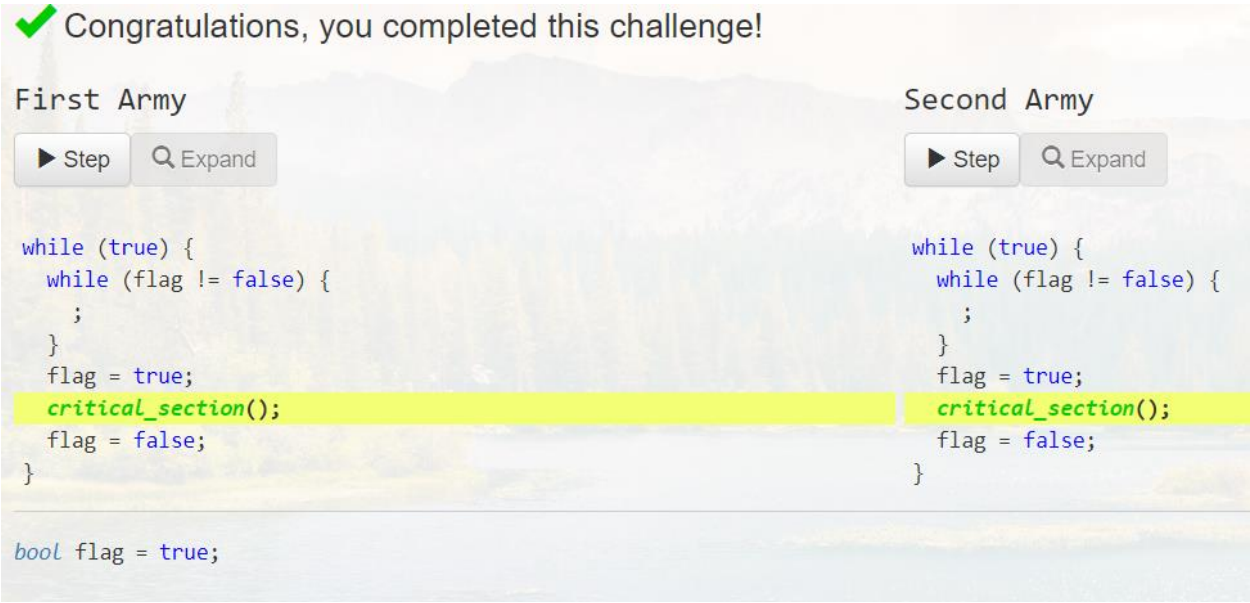
Результат:

Инвариант не нарушен



Задача с ресурса

В папке 1/ есть решение задачи в 1.txt, описание логики в 1.tla, и описание задачи 1.PNG



Инвариант:

TypeOK == \/ (who_is_in_critical \subseteq { first })

\/ (who_is_in_critical \subseteq { second })

\/ (who_is_in_critical = {})

Результат:

Инвариант нарушен (лог в 1/1.txt я же приведу более читаемый вид)

- 1.
  - flag_locked |-> "F",
  - who_is_after_while |-> {},
  - who_is_in_critical |-> {},
  - who_is_in_while |-> {1, 2},
- 2.
  - flag_locked |-> "F",
  - who_is_after_while |-> {2},
  - who_is_in_critical |-> {},
  - who_is_in_while |-> {1}],
- 3.
  - flag_locked |-> "F",

    - who_is_after_while |-> {1, 2},
    - who_is_in_critical |-> {},
- 4.
    - flag_locked |-> "T",
    - who_is_after_while |-> {1, 2},
    - who_is_in_critical |-> {},
- 5.
    - flag_locked |-> "T",
    - who_is_after_while |-> {2},
    - who_is_in_critical |-> {1}
- 6.
    - flag_locked |-> "T",
    - who_is_in_critical -> {1, 2},