CS 472: Provably Correct Programming
Final Project README

Himanshu Dongre

Spring 2024

Included files

- `incsum.v`: The code file.
- `README.pdf`: This README file.

Important definitions and lemmas

Lists:-

List sum

1. `Fixpoint is_list (l : list Z) (v : val) : iProp Σ.`
2. `Fixpoint sum_list_coq (l : list Z) : Z.`
3. `Definition sum_list : val.`

4.
   ```
   Lemma sum_list_spec l v :
     {{{ is_list l v }}} sum_list v
     {{{ RET #(sum_list_coq l); is_list l v }}}.
   ```

List increase

5. `Definition inc_list : val.`

6.
   ```
   Lemma inc_list_spec n l v :
     {{{ is_list l v }}}
       inc_list #n v
     {{{ RET #(); is_list (map (Z.add n) l) v }}}.
   ```

Spinlock:-

7. `Definition is_lock (lk : val) (R : iProp Σ) : iProp Σ.`

New lock

8. `Definition newlock : val.`

9.
   ```
   Lemma newlock_spec R:
     {{{ R }}} newlock #() {{{ lk, RET lk; is_lock lk R }}}.
   ```

Try lock

10. `Definition` **`try_lock`** `: val.`
11. `Lemma try_acquire_spec lk R :`
    ```
    {{{ is_lock lk R }}} try_acquire lk
    {{{ b, RET #b; if b is true then R else True }}}.
    ```

Acquire

12. `Definition` **`acquire`** `: val.`
13. `Lemma acquire_spec lk R :`
    ```
    {{{ is_lock lk R }}} acquire lk {{{ RET #(); R }}}.
    ```

Release

14. `Definition` **`release`** `: val.`
15. `Lemma release_spec lk R :`
    ```
    {{{ is_lock lk R ∗ R }}} release lk {{{ RET #(); True }}}.
    ```

Invariant

16. `Definition` **`lock_inv`** `(l : loc) (R : iProp Σ) : iProp Σ.`

Parallel inc sum:-

Function

17. `Definition` **`parallel_inc_sum_locked`** `(lock : val) : val.`
    One thread increases the given list by a given n. A second thread stores the list sum in a variable *sum*. Both threads acquire the same spinlock before executing their respective operation, and release it after. The function returns *sum*.

Invariant

18. `Definition` **`inc_sum_inv`** `(n : Z) (l : list Z) (v : val) : iProp Σ.`
    The function invariant states that there exists a list l' such that sum of l is less than or equal to that of l', and separately, v points to l'.

Helper lemma

19. `Lemma sum_inc_eq_n_len : forall (l : list Z) n,`
    ```
    (sum_list_coq (map (Z.add n) l) = (n * length l) + sum_list_coq l)%Z.
    ```

Spec

20. `Theorem` **`parallel_inc_sum_locked_spec`** `lock l v (n : Z).`
    Pre-condition : `is_lock lock` with resource `inc_sum_inv n l v`, and separately, $n \geq 0$.
    Function call : `parallel_inc_sum_locked lock #n v`.
    Post-condition: The function returns an integer m such that the list sum of l is less than or equal to m.

References

- All definitions and lemmas for “Lists” was taken from `ex_02_sumlist.v` distributed in the course.
- All definitions and lemmas for “Spinlock” was taken from `ex_03_spinlock.v` distributed in the course.

- No external references used.