Multitasking, Invariants and Temporal Logic

Aaron Feldman

University of California, Berkeley

Chapter 11 EX 2:

To the best of my knowledge, there isn't a solution that would remove the necessity of locking b before a as in the example, since this is a requirement given by the development team and a will need to remain locked around the if statement on line 9, as it is possible that a context switch will occur and the value of a could change from another thread. Therefore I don't think there is a solution to this.

Chapter 11 EX 4:

```

pthread_mutex_t lock = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t cond = PTHREAD_COND_INITIALIZER;
int Queue_Size = 0;
int Waiters = 0;
void* producer (void* arg) {
    int i;
    for (i = 0; i< 10; i++) {
        pthread_mutex_lock(&lock);
        while (Queue_Size > 5) {
            Waiters++;
            pthread_cond_wait(&cond, &lock);
        }
        send(i);
        Queue_Size--;
        pthread_mutex_unlock(&lock);
    }
    return NULL;
}
void* consumer(void* arg) {
    while(1) {
        pthread_mutex_lock(&lock);
        Queue_Size--;
        printf("received %d\n", get());
        if (Waiters > 0) {
            pthread_cond_signal(&cond);
        }
        pthread_mutex_unlock(&lock);
    }
    return NULL;
}
```

Chapter 13 EX 1:

(a) False, consider a system where p toggles p/¬p, this satisfies

(b) True,

Chapter 13 EX 2:

(a) False, consider input x present at (c), it can never transition to (b)

(b) False, consider (c), it can never transition out and give output y = 1

(c) True

(d) True

(e) True

(f) True

(g) False, when in (a) $\neg$ x will put us in (b), but a x at this point will keep us from (c)