Assignment 0

Engi-8893 / Engi-9869 Theodore Norvell

Due 12:00 noon Wednesday 2009 Jan 28.

Please submit on paper. Solutions must be typed or **very** neatly handwritten. Diagrams, if any, may be hand drawn (neatly).

Q0 [**10**] Classify the following properties as safety or liveness properties. Explain your reasoning.

(a) "Someday my prince will come."

(b) "If you send me e-mail, I will reply."

(c) "If you send me e-mail, I will not reply."

(d) "Pizza will be delivered within 30 minutes of being ordered."

Q1 [**10**]

```
## $0 \le i - j \le 2$
## Global $G : 0 \le i - j \le 2$
co
        $\langle$await$(i - j \ne 2)$ ; $\rangle$
        $\langle i := i + 1; \rangle$
//
        $\langle$await$(i - j \ne 0)$ ; $\rangle$
        $\langle j := j + 1; \rangle$
oc
```

We will show that G is a global invariant of the above command. Trivially G is implied by the precondition. It remains:

(a) to add local assertions as needed,

(b) to list all triples that need to be proved for local correctness —these will include triples showing that G is preserved by each assignment—,

(c) to explain why each of these triples is valid, and

(d) to show that there is no interference.

Q2 [**20**]. We need to copy from an nonshared array a to an nonshared array b using a shared buffer.

In the following array algorithm,

- k and n are positive integer constants,
- buf is an array of size k,
- p and c are integer variables, while
- a and b are arrays of size n.

```
## $p = c = 0$
co
        /* Producer */
        while( $p < n$ ) {
                $\langle$await$(p - c \neq k)$ ; $\rangle$
                $\langle buf[p \bmod k] := a[p]; \rangle$
                $\langle p := p + 1; \rangle$ }
//
        /* Consumer */
        while( $c < n$ ) {
                $\langle$await$(p - c \neq 0)$ ; $\rangle$
                $\langle b[c] := buf[c \bmod k]; \rangle$
                $\langle c := c + 1; \rangle$ }
oc
```

We will annotate the algorithm with assertion as shown in Fig 1[1]

(a) [6] The answers to the three subparts of this question should constitute the nontrivial aspects of a proof that your outline from part (a) is locally valid. Remember that you can use any of the global invariants as (a part of) the precondition of any atomic action in the co command For each subpart write out the Hoare triple(s) that must be shown valid and provide an argument that it/they are valid.[2]

[1] Idiosyncratic Notation Alert: The set $\{c, ..p\}$ contains the $p - c$ integers starting with c. In particular when $c = p$, $\{c, ..p\} = \emptyset$.

[2] One issue not covered in the notes, is assignment to array items. If a is an array and all mentions of a in Q are of the form $a[i]$, then we have

$$\frac{P \Rightarrow Q_{a[i] \leftarrow E}}{\{P\}\ a[i] := E\ \{Q\}}$$

```
## $p = c = 0$
## Global $G0 : 0 \leq p \leq n$
## Global $G1 : 0 \leq c \leq n$
## Global $G2 : 0 \leq p - c \leq k$
## Global $G3 : (\forall i \in \{c, ..p\} \cdot buf[i \bmod k] = a[i])$
co
    /* Producer */
    while( $p < n$ ) {
        ## $p < n$
        $\langle$await$(p - c \neq k)$ ; $\rangle$
        ## $p < n \wedge p - c < k$
        $\langle buf[p \bmod k] := a[p]; \rangle$
        ## $p < n \wedge p - c < k \wedge buf[p \bmod k] = a[p]$
        $\langle p := p + 1; \rangle$ }
//
    /* Consumer */
    ## Loop Inv. $(\forall i \in \{0, ..c\} \cdot b[i] = a[i])$
    while( $c < n$ ) {
        ## $(\forall i \in \{0, ..c\} \cdot b[i] = a[i]) \wedge c < n$
        $\langle$await$(p - c \neq 0)$ ; $\rangle$
        ## $(\forall i \in \{0, ..c\} \cdot b[i] = a[i]) \wedge c < n \wedge p - c > 0$
        $\langle b[c] := buf[c \bmod k]; \rangle$
        ## $(\forall i \in \{0, ..c\} \cdot b[i] = a[i]) \wedge c < n \wedge p - c > 0 \wedge b[c] = a[c]$
        $\langle c := c + 1; \rangle$ }
    ## $(\forall i \in \{0, ..n\} \cdot b[i] = a[i])$
oc
## $(\forall i \in \{0, ..n\} \cdot b[i] = a[i])$
```

Figure 1: Proof outline for the copy algorithm

(i) Show that $0 \le p - c \le k$ is a global invariant by showing that it is implied by the precondition of the **co** statement and that it is a postcondition of each action that changes either p or c.

(ii) Show that $\forall i \in \{c, ..p\} \cdot \mathit{buf}[i \bmod k] = a[i]$ is a global invariant by showing that it is implied by the precondition of the **co** statement and that it is implied by the precondition of the **co** statement and that it is a post-condition of each action that changes p, c or buf.

(iii) Show that $b[c] = a[c]$ is a postcondition of the assignment

$$\langle b[c] := \mathit{buf}[c \bmod k]; \rangle \,.$$

(b) **[4]** This part is to show freedom from interference. For each subpart write out the Hoare triple(s) that must be shown valid and provide an argument that it/they are valid.

(i) For any assertions, other than the global invariants, that appear in the producer and that involve variables b or c, list the Hoare triples that must be shown valid to ensure there is no interference from the consumer. Explain why each of these Hoare triples is valid.

(ii) Likewise, for any assertions, other than the global invariants, that appear in the consumer and that involve variables buf or p, list the Hoare triples that must be shown to be valid to ensure that there is no interference from the producer. Explain why each of these Hoare triples is valid.

(c) Of the expressions and assignments in this algorithm, which obey the AMO property. Justify your answer.

Q3 Bonus: In the notes I listed the following motivations for concurrent programming: "faster processing (when ≥ 1 processor)," "more effective use of resources (e.g. disks)," "faster response to the user," "the system is conceptually concurrent," "fault tolerance," "the system is distributed." What did I miss?

If the truth values of P and Q does not depend on $a[i]$ then

$$\frac{P \Rightarrow Q}{\{P\}\ a[i] := E\ \{Q\}}$$

Sometimes we need to use both these rules, in which case we can use the principle that

$$\frac{\begin{array}{c}\{P_0\}\,S\,\{Q_0\}\\ \{P_1\}\,S\,\{Q_1\}\end{array}}{\{P_0 \wedge P_1\}\,S\,\{Q_0 \wedge Q_1\}}$$