Homework #3 Solutions

CSE 403

February 24, 1997

1. Software Structures and Interfaces

   (a) Comprises

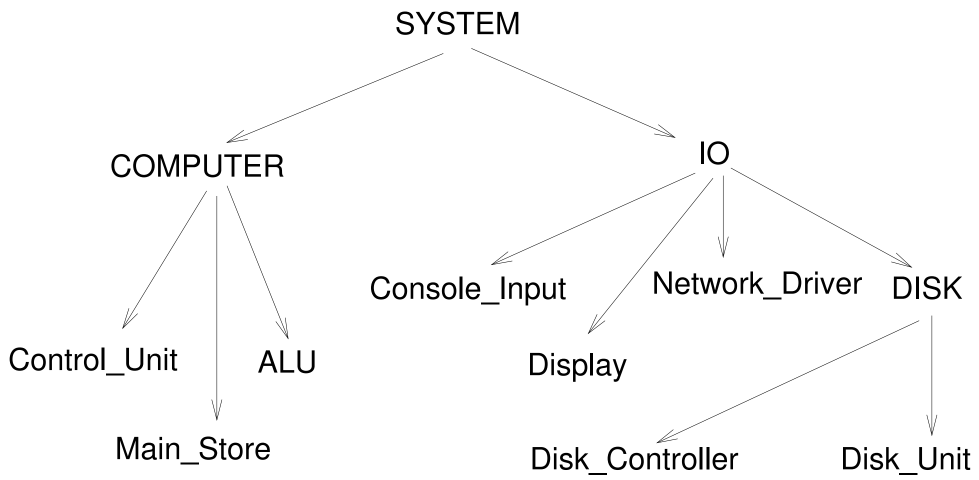

Figure 1: COMPRISES Relation

   (b) Uses

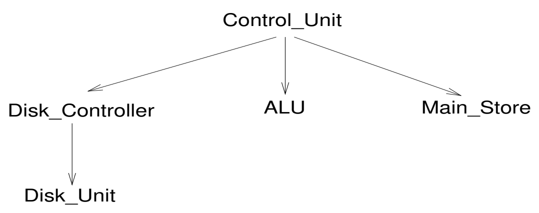

Figure 2: USES Relation

`Control_Unit` requests service from the `ALU`, `Main_Store`, and `Disk_Controller` through the IO. The `Control_Unit` sends a message and awaits a response from the other modules.

2. Concurrency

   (a) Show how the memory *Get* and *Release* actions can be synchronized using semaphores.

   Define two semphores: One binary semaphore, `LOCK`, and one counting semaphore `MEMORY` with the initial value of 2.

   The `MEMORY` semaphore keeps track of the number of memory block available. The `LOCK` semaphore ensures mutual exclusion for the GetMemory() and ReleaseMemory() operations.

   The following is the code sequence needed for A1:

```
BEGIN LOOP
wait(MEMORY);
wait(LOCK);
  GetMemory();
signal(LOCK);
wait(MEMORY);
wait(LOCK);
  GetMemory();
signal(LOCK);
wait(LOCK);
  ReleaseMemory();
  signal(MEMORY);
signal(LOCK);
wait(LOCK);
  ReleaseMemory();
  signal(MEMORY);
signal(LOCK);
END LOOP
```

   The following is the code sequence needed for A2:

```
BEGIN LOOP
wait(MEMORY);
wait(LOCK);
  GetMemory();
signal(LOCK);
wait(LOCK);
  ReleaseMemory();
  signal(MEMORY);
signal(LOCK);
END LOOP
```

   (b) Define a guardian task that "accepts" *Get* and *Release* call on the memory resource.

```
MEMORY = 2;

begin LOOP
select
  accept Get when MEMORY > 0
    begin
      MEMORY := MEMORY - 1;
```

```
          /* code to acquire memory */
        end
      accept Release when MEMORY < 2
        begin
          /* code to release memory */
          MEMORY := MEMORY + 1;
        end
    end select
    end LOOP
```

3. Using the Axiom of Assignment

```
/* {a is an integer} */
/* {a+a+a+a+a+a+a+a+a+a = 10 * a} by arithmetic rules */
a2 := a + a;
/* {a2 + a2 + a2 + a2 + a2 = 10 * a} by Axiom of Assignment */
a8 := a2 + a2;
/* {a8 + a8 + a2 = 10 * a} by Axiom of Assignment */
a8 := a8 + a8;
/* {a8 + a2 = 10 * a} by Axiom of Assignment */
a10 := a8 + a2;
/* {a10 = 10 * a} post-condition */
```

4. Application of Rule of Iteration

Loop Invariant: x≥a ∧ x≥1 ∧ x≤max(1,a,b)

```
/* a is an integer and b is an integer */
x := a;
/* x = a  Axiom of Assignment */
if x < 1 then x := 1
/* x >= a and x >= 1 and x = max(1, a)
         => Axiom of Assignment and Rule for Conditionals */
/* x >= a and x >= 1 and x <= max(1, a, b)
         => Loop Invariant implied by previous statement */
while x < b
  /* x >= a and x >= 1 and x <= max(1, a, b) and x < b
         => Loop Invariant and Loop Condition */
  /* x >= a and x >= 1 and x < max(1, a, b) and x < b and b = max(1,a,b)
         => implied by previous statement */
  /* x+1 >= a and x+1 >= 1 and x+1 <= max(1, a, b) and x+1 <= b
         => implied by previous statement */
  x := x+1;
  /* x >= a and x >= 1 and x <= max(1, a, b)
         =>Axiom of Assignment*/
while end
/* x >= a and x >= 1 and x <= max(1, a, b) and x >= b
         => Loop Invariant and Loop Exit Condition */
/* x = max (1, a, b)
         => implied by previous statement */
```

5. Proof Rule for *Repeat* Statement

   The

   ```
   repeat S until B
   ```

   statement is equivalent to the

   ```
   S while not B do S
   ```

   statement.

   We know that:

   Rule for Statements: $\{\mathrm{Q}\}\mathrm{S}\{\mathrm{P}\}$

   Rule for Interation: $\dfrac{\{\mathrm{P}\wedge\neg\ \mathrm{B}\}\mathrm{S}\{\mathrm{P}\}}{\{\mathrm{P}\}\ \text{while}\ \neg\ \mathrm{B}\ \text{do S}\ \{\mathrm{P}\wedge\mathrm{B}\}}$

   So by combining these Rules, we come up with the following rule for the `repeat` statement:

   $$\frac{\{\mathrm{Q}\}\mathrm{S}\{\mathrm{P}\}\wedge\{\mathrm{P}\wedge\neg\ \mathrm{B}\}\mathrm{S}\{\mathrm{P}\}}{\{\mathrm{Q}\}\ \text{repeat S until B}\ \{\mathrm{P}\wedge\mathrm{B}\}}$$