COMP473: Formal Aspects of Concurrent Systems

Annie Luxton: 300046696

5 August 2003

Question

The program is intended to compute the sum $2^0 + ... + 2^{N-1}$, where 2^k is "2 to the power k" (so we know the required sum is $2^N - 1$).

The program can be written as:

```
x: integer  Initially x=0
cobegin
  x := x+2^0 || x := x+2^1 || ... || x := x+2^(N-1)
coend
```

where cobegin-coend bracket a set of statements to be executed in parallel, and $\|$ separates the parallel statements.

To express this as a transition system (set of actions), we introduce a boolean array b, where $b[i]$ records whether action i has been performed.

```
Variables:
  x: integer  Initially x=0
  b[0..N-1]: {0,1}  Initially b[i]=0 for i=0..N-1

Actions:
  e(i)::
    enabled = (b[i]=0)
    action  = (x := x+2^i; b[i] := 1)
```

All actions have weak fairness.

Question 1

The task is to show that $x = 2^N - 1$ holds when the program terminates. To do this, you need to find an invariant that can be used with the technique described in Mark Moir's notes. You might find it helpful to think about the program with $N = 2$ or 3.

Question 2

Show also that the program does terminate. To do this, you will need to do a progress proof.

Question 3

Do we need to assume that the assignments to x and $b[i]$ are performed as a single atomic action?

1 Solution to Q.1

In order to show that $x = 2^N - 1$ holds when the program terminates, we must find an invariant that shows this.

$$I0 \quad \equiv \quad \forall\, i : b[i] = 1 \Rightarrow x = 2^N - 1$$

I0 says that when all $b[i] = 1$, that is, when all actions have been done, $x = 2^N - 1$.

I0 is too strong to be an invariant. We must weaken it and use this weaker, known invariant and the implication rule of invariants, Invariant Rule 1, to prove Invariant(I0).

$$I1 \quad \equiv \quad x = \sum k : 0 \leq k \leq N-1 \wedge b[k] = 1 : 2^k$$

I1 is a weaker version of I0. It says that $x =$ the sum of all 2^k's where $b[k] = 1$. This allows for the fact that the program is concurrent and can therefore do it's N events in any order.

Now we must prove Invariant(I1) and from there attempt to prove Invariant(I0).

1.1 Inductive Proof of Invariant(I1) - using Invariant Rule 2

* Show that I1 holds initially:

$$\begin{aligned}
&\mathit{Initial} \Rightarrow I1 \\
&\quad \equiv x = 0 \wedge \forall\, k : 0 \leq k \leq N-1 : b[k] = 0 \\
&\qquad \Rightarrow x = \sum k : 0 \leq k \leq N-1 \wedge b[k] = 1 : 2^k \\
&\quad \equiv \mathsf{true} && \text{(From direct manipulation)}
\end{aligned}$$

* Show that I1 is not falsified by any statement execution:

$$\begin{aligned}
&I1 \Rightarrow \mathit{wp}(e, I1) \\
&\quad \equiv I1 \Rightarrow (b[i] = 0 \Rightarrow \mathit{wp}(x := x + 2^i;\ b[i] := 1, I1)) && \text{(From wp on events)} \\
&\quad \equiv I1 \Rightarrow (b[i] = 0 \Rightarrow \mathit{wp}(x := x + 2^i, \mathit{wp}(b[i] := 1, I1))) && \text{(From wp sequential composition)} \\
&\quad \equiv I1 \Rightarrow (b[i] = 0 \Rightarrow \mathit{wp}(x := x + 2^i, \mathit{wp}(b := \mathit{update(b,\ i,\ 1)}, I1))) && \text{(From wp array updates)} \\
&\quad \equiv I1 \Rightarrow (b[i] = 0 \Rightarrow \mathit{wp}(x := x + 2^i, \\
&\qquad x = \sum k : 0 \leq k \leq N-1 \wedge \mathit{update}(b, i, 1)[k] = 1 : 2^k)) && \text{(From wp assignment)} \\
&\quad \equiv I1 \Rightarrow (b[i] = 0 \Rightarrow \mathit{wp}(x := x + 2^i, \\
&\qquad x = (\sum k : 0 \leq k < i \wedge b[k] = 1 : 2^k \\
&\qquad\quad +2^i \\
&\qquad\quad + \sum k : i < k \leq N-1 \wedge b[k] = 1 : 2^k))) && \text{(From expansion)} \\
&\quad \equiv I1 \Rightarrow (b[i] = 0 \Rightarrow \\
&\qquad x + 2^i = (\sum k : 0 \leq k < i \wedge b[k] = 1 : 2^k \\
&\qquad\quad +2^i \\
&\qquad\quad + \sum k : i < k \leq N-1 \wedge b[k] = 1 : 2^k)) && \text{(From wp assignment)} \\
&\quad \equiv I1 \Rightarrow (b[i] = 0 \Rightarrow \\
&\qquad x = (\sum k : 0 \leq k < i \wedge b[k] = 1 : 2^k \\
&\qquad\quad + \sum k : i < k \leq N-1 \wedge b[k] = 1 : 2^k)) && \text{(From direct manipulation)} \\
&\quad \equiv \mathsf{true} && \text{(From logic manipulation)}
\end{aligned}$$

Therefore, By Invariant Rule 2, Invariant(I1).

1.2 Proof of Invariant(I0) - using Invariant Rule 1

* Using I1 as our known invariant, show that I1 $\Rightarrow$ I0:

$$
\begin{aligned}
&I1 \Rightarrow I0 \\
&\quad \equiv x = \sum k : 0 \leq k \leq N-1 \wedge b[k] = 1 : 2^k \Rightarrow (\forall\, i : b[i] = 1 \Rightarrow x = 2^N - 1) \\
&\quad \equiv \forall\, i : b[i] = 1 \Rightarrow (x = \sum k : 0 \leq k \leq N-1 \wedge b[k] = 1 : 2^k \Rightarrow x = 2^N - 1) && \text{(Logical equivalences)} \\
&\quad \equiv \mathsf{true} && \text{(From direct manipulation)}
\end{aligned}
$$

Therefore, by Invariant Rule 1, Invariant(I0).

1.3 Conclusion

When all actions have been done, that is, when $\forall\, i : b[i] = 1$, the value of x is $x = 2^N - 1$.

2 Solution to Q.2

Now that we know that when the program terminates, the value of x is $2^N - 1$, it would be useful to know whether the program terminates at all.

In order to show that the program terminates, we must show that progress occurs. Progress is shown using leads-to assertions. Therefore, we must prove a leads-to assertion that shows that the program terminates.

$$LT0 \quad \equiv \quad \forall\, i : b[i] = 0 \text{ leads-to } \forall\, i : b[i] = 1$$

LT0 says that from the initial state when all $b[i] = 0$, that $b[i] = 1$ will be true for all $0 \leq i \leq N - 1$ sometime in the future.

Proving LT0 on its own is very hard. Instead, try using a combination of proving LT0 through using transitivity with other leads-to assertions, as well a well-founded ranking.

Let $n = \sum_{i=0}^{N-1} b[i]$ be the total number of $b[i]$'s set to 1. Then:

$$LT1 \quad \equiv \quad n = 0 \text{ leads-to } n = N$$

LT1 is simply restating LT0. It claims that the state where no $b[i]$'s are set to 1 will lead to the state where all $b[i]$'s are set to 1. In order to prove this, we must use a well-founded ranking together with another leads-to assertion that isn't as strict as LT1.

$$LT2 \quad \equiv \quad n < N \text{ leads-to } \sum_{i=0}^{N-1} b[i] = n + 1$$

LT2 says that as long as the total number of $b[i]$'s set to 1 is less than N, then there are more $b[i]$'s to set to 1, thus increasing the number of $b[i]$'s set to 1 by 1. This leads-to assertion is not as strict as LT1, so we may use it to prove LT1.

2.1 Proof of leads-to assertion LT2

Since all the events in our system have weak fairness, $E = Events_A$.

* Show that $\forall\, e \in E, n < N \Rightarrow wp(e, \sum_{i=0}^{N-1} b[i] = n + 1)$:

$$
\begin{aligned}
&\forall\, e \in E, n < N \Rightarrow wp(e, \textstyle\sum_{i=0}^{N-1} b[i] = n + 1) \\
&\quad \equiv \forall\, e \in E, n < N \Rightarrow (b[j] = 0 \Rightarrow wp(x := x + 2^j;\, b[j] := 1, \\
&\qquad \textstyle\sum_{i=0}^{N-1} b[i] = n + 1)) && \text{(From wp on events)} \\
&\quad \equiv \forall\, e \in E, n < N \Rightarrow (b[j] = 0 \Rightarrow wp(x := x + 2^j, wp(b[j] := 1, \\
&\qquad \textstyle\sum_{i=0}^{N-1} b[i] = n + 1))) && \text{(From wp sequential composition)]} \\
&\quad \equiv \forall\, e \in E, n < N \Rightarrow (b[j] = 0 \Rightarrow wp(x := x + 2^j, wp(b := \mathit{update(b,\ j,\ 1)}, \\
&\qquad \textstyle\sum_{i=0}^{N-1} b[i] = n + 1))) && \text{(From wp array updates)} \\
&\quad \equiv \forall\, e \in E, n < N \Rightarrow (b[j] = 0 \Rightarrow wp(x := x + 2^j, \\
&\qquad \textstyle\sum_{i=0}^{N-1} update(b, j, 1)[i] = n + 1)) && \text{(From wp assignment)} \\
&\quad \equiv \forall\, e \in E, n < N \Rightarrow (b[j] = 0 \Rightarrow wp(x := x + 2^j, \\
&\qquad (\textstyle\sum_{i=0}^{j} b[i] + 1 + \sum_{i=j+1}^{N-1} b[i]) = n + 1)) && \text{(From expansion)} \\
&\quad \equiv \forall\, e \in E, n < N \Rightarrow (b[j] = 0 \Rightarrow (\textstyle\sum_{i=0}^{j} b[i] + 1 + \sum_{i=j+1}^{N-1} b[i]) = n + 1)) && \text{(From wp assignment)} \\
&\quad \equiv \forall\, e \in E, n < N \Rightarrow (b[j] = 0 \Rightarrow (\textstyle\sum_{i=0}^{j} b[i] + \sum_{i=j+1}^{N-1} b[i]) = n)) \\
&\quad \equiv \mathsf{true} && \text{(From direct manipulation)}
\end{aligned}
$$

* Show that $\forall\, e \in Events_A - E, n < N \Rightarrow wp(e, \sum_{i=0}^{N-1} b[i] = n + 1)$:

Since $E = Events_A$ and therefore $Events_A - E = \varnothing$ in this program, we do not need to prove anything here.

* Show that Invariant$(n < N \Rightarrow enable(E))$, where $enable(E) \equiv \exists\, i : b[i] = 0$

2.1.1 Proof of Invariant$(n < N \Rightarrow enable(E))$ - using Invariant Rule 2

* Show that $n < N \Rightarrow enable(E)$ holds initially:

$$
\begin{aligned}
&\mathit{Initial} \Rightarrow n < N \Rightarrow enable(E) \\
&\quad \equiv x = 0 \wedge \forall\, k : 0 \le k \le N - 1 \wedge b[k] = 0 \Rightarrow n < N \Rightarrow \exists\, i : b[i] = 0 \\
&\quad \equiv \mathsf{true} && \text{(Directly)}
\end{aligned}
$$

* Show that $n < N \Rightarrow enable(E)$ is not falsified by any statement execution:

$(n < N \Rightarrow \exists\, i : b[i] = 0) \Rightarrow wp(e, (n < N \Rightarrow \exists\, i : b[i] = 0))$

$\equiv (n < N \Rightarrow \exists\, i : b[i] = 0) \Rightarrow (b[j] = 0$

$\Rightarrow wp(x := x + 2^j;\, b[j] := 1, (n < N \Rightarrow \exists\, i : b[i] = 0)))$ (From wp on events)

$\equiv (n < N \Rightarrow \exists\, i : b[i] = 0) \Rightarrow (b[j] = 0$

$\Rightarrow wp(x := x + 2^j, wp(b[j] := 1, (n < N \Rightarrow \exists\, i : b[i] = 0)))$ (From wp sequential composition)

$\equiv (n < N \Rightarrow \exists\, i : b[i] = 0) \Rightarrow (b[j] = 0 \Rightarrow wp(x := x + 2^j,$

$wp(b := \mathit{update(b,\ j,\ 1)}, (n < N \Rightarrow \exists\, i : b[i] = 0))))$ (From wp array updates)

$\equiv (n < N \Rightarrow \exists\, i : b[i] = 0) \Rightarrow (b[j] = 0 \Rightarrow wp(x := x + 2^j,$

$(n < N \Rightarrow \exists\, i : \mathit{update}(b, j, 1)[i] = 0)))$ (From wp assignment)

$\equiv (n < N \Rightarrow \exists\, i : b[i] = 0) \Rightarrow (b[j] = 0 \Rightarrow wp(x := x + 2^j,$

$(n < N \Rightarrow \exists\, i \neq j : b[i] = 0)))$ (From expansion)

$\equiv (n < N \Rightarrow \exists\, i : b[i] = 0) \Rightarrow (b[j] = 0 \Rightarrow (n < N \Rightarrow \exists\, i \neq j : b[i] = 0))$ (From wp assignment)

$\equiv \mathsf{true}$ (From logic manipulation)

Therefore, By Invariant Rule 2, Invariant$(n < N \Rightarrow \mathit{enable}(E))$.

And therefore, we conclude that the leads-to assertion $LT2 \equiv n < N$ leads-to $\sum_{i=0}^{N-1} b[i] = n + 1$ is true.

2.2 Proof of leads-to assertion LT1 - using well-founded ranking

Let $F(m) \equiv N - n = m$. This function decreases as the number of $b[i]$'s set to 1 increases.

* Show that $F(0)$ leads-to $n = N$:

$F(0) \equiv N - n = 0$ (By the definition of F(m))

$\Rightarrow n = N$ (By the definition of F(m))

Therefore, $F(0)$ leads-to $n = N$ by definition.

* Show that $n = 0$ leads-to $n = N \vee (\exists\, j : F(j))$:

$n = 0 \Rightarrow N - n = m = N$ (By the definition of F(m))

$\Rightarrow F(N)$ (By the definition of F(m))

$\Rightarrow \exists\, j : F(j)$ (Directly)

Therefore, since $n = 0 \Rightarrow \exists\, j : F(j)$, $n = 0$ leads-to $n = N \vee (\exists\, j : F(j))$ is true.

* Show that $F(m) \wedge m > 0$ leads-to $n = N \vee (\exists\, j < m : F(j))$:

$m > 0 \Rightarrow N > n$ (By the definition of F(m))

$\Rightarrow n \neq N$ (Since $N > n$)

$\Rightarrow \exists\, j < m : F(j)$ (We know from LT1 that every event will increase n, thus decreasing m)

Therefore, since $m > 0 \Rightarrow \exists\, j < m : F(j)$ is true, $F(m) \wedge m > 0$ leads-to $n = N \vee (\exists\, j < m : F(j))$ is also true.

Thus, we conclude that the leads-to assertion $LT1 \equiv n = 0$ leads-to $n = N$ is true.

2.3 Proof of leads-to assertion LT0 - using implication

* Using LT1 as our known true leads-to assertion, show that LT1 $\Rightarrow$ LT0:

$$
\begin{aligned}
&LT1 \Rightarrow LT0 \\
&\quad \equiv n = 0 \text{ leads-to } n = N \Rightarrow \forall\, i : b[i] = 0 \text{ leads-to } \forall\, i : b[i] = 1 \\
&\quad \equiv n = 0 \text{ leads-to } n = N \Rightarrow \textstyle\sum_{i=0}^{N-1} b[i] = 0 \text{ leads-to } \sum_{i=0}^{N-1} b[i] = N && \text{(Manipulation)} \\
&\quad \equiv n = 0 \equiv \textstyle\sum_{i=0}^{N-1} b[i] = 0 \wedge n = N \equiv \sum_{i=0}^{N-1} b[i] = N && \text{(Definition of n)} \\
&\quad \equiv \mathsf{true}
\end{aligned}
$$

Therefore, by LT1, we know that LT0 holds. They literally say the same thing.

2.4 Conclusion

We therefore know from the leads-to assertion LT0 that the program will terminate, that is, that $\forall\, i : b[i] = 1$ will eventually be reached when starting in the initial state, $\forall\, i : b[i] = 0$.

3 Solution to Q.3

Yes, we need to assume that the assignments to x and $b[i]$ are performed as a single atomic action because the array b is an auxilliary array that shows control over the program, and therefore needs to change as x changes. There should never a case where an assignment is made to x only, without the matching assignment to b.