Solution for Exercise 2

Dr.Kai Huang

October 28, 2014

2.1 Model of Computation - Kahn Process Network

2.1.1

Similar to the Fair Merge mentioned in the lecture, here we consider again a merge process that merges data tokens from two input channels into one output channel, as shown in Figure 1. In this case, the merge process can peek but at most one data token at a time from inputs, which means the process can check the arrival and the status of the most recent incoming token. There are two different versions for this One Peek Merge process, the semantics of which are shown in Algorithm 1 and Algorithm 2, respectively.

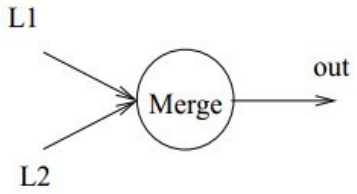

Figure 1: One Peek Merge.

Algorithm 1

if L1[X], L2[Y] **then**
del(X), del(Y), out[X,Y]
else if L1[X], L2[ϕ] **then**
del(X), out[X]
else if L1[ϕ], L2[Y] **then**
del(Y), out[Y]
else if L1[ϕ], L2[ϕ] **then**
no operation
end if

Algorithm 2

if L1[X] = L2[Y] **then**
del(X), del(Y), out[X,Y]
else if L1[X] < L2[Y] **then**
del(X), out[X]
else if L1[X] > L2[Y] **then**
del(Y), out[Y]
end if

In Algorithm 2, we assume that the tokens X and Y have a built-in "serial number" and we

know that each channel produces increasing sequences of serial numbers (with unknown gaps though).

•Examine the determinacy of these two algorithms. Prove or disprove your conclusion.

•Examine the fairness of these two algorithms.

Solution:

Algorithm 1 is non-deterministic since the arrival order determines the output of the merge. But it is fair, since even if the sequences have different lengths, they are served on a first-come, first-serve basic. Formally:

X = ($[x1, x2]$, $[\phi]$); $X' = ([x_1, x_2], [y_1])$

F(X) = $([x_1, x_2])$; F(X') = $([x_1, y_1, x_2])$

X$\subseteq X'$ but F(X)$\not\subseteq$F(X')

Algorithm 2 is deterministic since for any two input sequences (including knowledge of each token's serial number), the output sequence is determined (increasing serial numbers). but it is unfair since we must request one token each for the comparison of the merge step, which unfairly starves the longer sequence (say $L2$), while waiting for the (never appearing) "post-ultimate" token of the shorter sequence (say $L1$). That is, once we have consumed the last token of $L1$, we keep waiting indefinitely for a token that won't arrive.

2.1.2

Draw a Kahn process network that can generate the sequence of quadratic numbers $n(n+1)/2$. Use basic processes that add two numbers, multiply two numbers, or duplicate a number. You can also use initialization processes that generate a constant and then simply forward their input. Finally, you can use a sink process.

Solution:

Kahn process networks in general

A Kahn process network (KPN) is a model describing a network of processes. It is a graph, where nodes correspond to processes whereas the connections–channels–between these processes correspond to edges between them. The channels are one-directional, so, we have a directed graph. A node of a KPN should be labeled with the program that runs on it. This program can be described similar a usual C-program; in general,a single node can do any calculation. Communication with other processes is done by sending on channels or receiving from channels. If a process sends on a channel, the message is placed on the channel immediately,in a FIFO way. If a process tries to receive from a channel and there are messages, it receives the oldest one.However, if there is no message on the channel yet, the process has to wait until there is some, therefore blocking the process node till a message arrives. It is not possible to wait on several input channels at the same time or to check the status of a channel without waiting on it. Because of these restrictions, the calculation of a KPN is deterministic (given the calculations of the processes are).

The basic processes

In this exercise, one part is to define basic processes that *a*) add two numbers, *b*) multiply two numbers, *c*) duplicate a number *d*) generate a constant, put it out and then just forward its input *e*) just receive numbers.In the following, possible programs for these processes are given:

a) "+": Adding two numbers. We assume that we have two input channels (in1, in2) and one output channel **out**.

```
for (;;) {
a:=wait(in1);
b:=wait(in2);
send(a+b, out);
}
```

b) "*": Multiplying two numbers. Same assumptions as before, almost the same, except the operator.

for (;;) {
a := wait(in1);
b := wait(in2);
send(a*b, out);
}

c) "D": Duplicating a number. We assume that we have one input channel **in** and two output channels **out1,out2**. We wait for an input and then just send it unmodified to both the output channels. We do this all the time.

for (;;) {
a := wait(in);
send(a,out1);
send(a,out2);
}

d) "Ci": Constant. We assume we have one input channel in and output channel out. Also, i shall be the constant to be sent initially. Initially, we just send the constant i. Then, infinitely often we wait for a number and just forward it to the output channel.

send(i, out);
for (;;) {
a := wait(in);
send(a,out);
}

e) "S": Sink process: We assume we have one input channel in. We just wait for numbers infinitely often and throw them away as they occur.

for (;;) {
wait(in);
}

Building the Kahn process network for the exercise

The original function definition was $f(n) = n(n+1)/2$. What we actually wanted you to do is to transform this function to a recursive definition and then transform this to a KPN. It is:$f(n) = n(n+1)/2 = \Sigma_{k=1}^{n} k$.This sum can be transformed into the recursion:

f(0) = 0
f(n) = n+f(n-1),n⩾1

So, we have m = 1 and can build a KPN from it as described before. It is given in Figure 2 and 3.

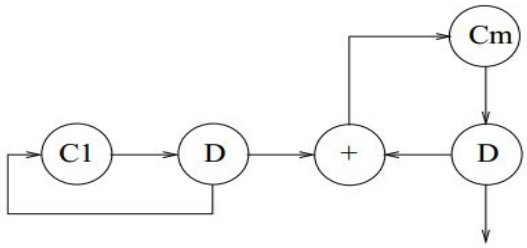

Figure 2:Calculating n.

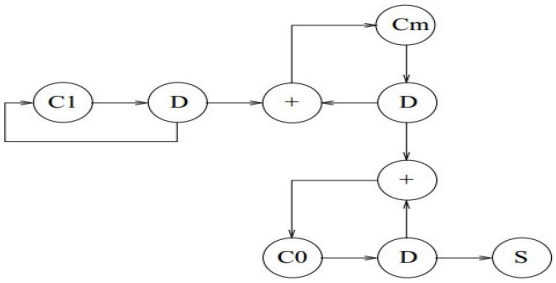

Figure 3:Kahn process network for $n(n+1)/2$

2.2 Model of Computation - Synchronous Data Flow

2.2.1

Given the SDF graph in Figure 4.

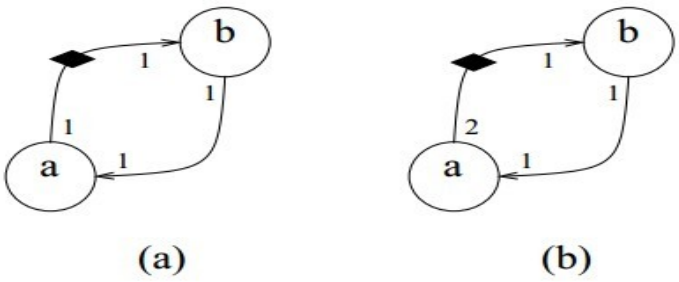

Figure 4:

•Determine the topological matrix of these two SDF graphs.
•Are these two graphs consistent?
•If yes, determine the number of firings of each node, which leads for a periodic execution. How often each node must fire thereby at least?

Solution

(a)

$$\begin{bmatrix} 1 & - & 1 \\ 1 & - & 1 \end{bmatrix}$$

(b)

$$\begin{bmatrix} 2 & - & 1 \\ 2 & - & 1 \end{bmatrix}$$

Therefore (a) is consistent while (b) not.

2.2.2

Given the SDF graph in Figure 5

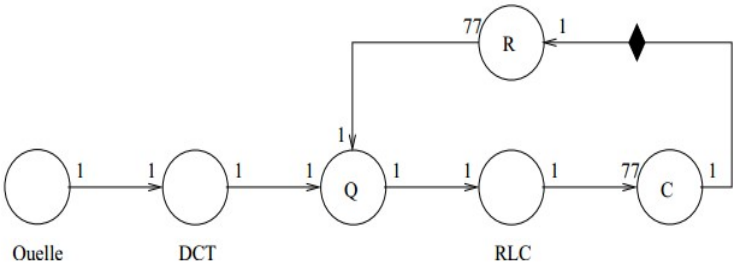

Figure 5

•Determine the topological matrix of this SDF graph.
•Examine the consistency.
•Determine the relative number of node firings, which leads for periodic execution at node firings

Solution:

topological matrix:

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -77 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 0 & 0 & -77 \end{pmatrix} \tag{1}$$

Therefore it is consistent. Fire number: Quelle:77; DCT:77; Q:77; RLC:77; C:1; R:1