HW/SW Codesign

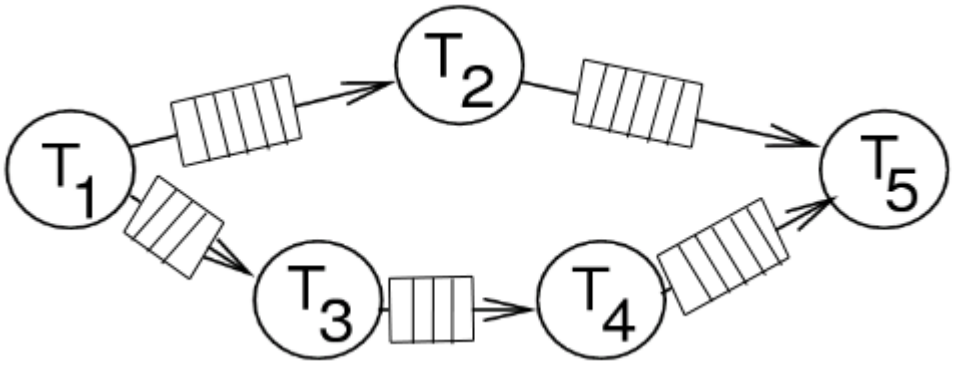
Exercise 2: Kahn Process Networks and Synchronous Data Flows

30. September 2015

Rehan Ahmed

Slides Prepared by Mirela Botezatu

Kahn Process Network (KPN)



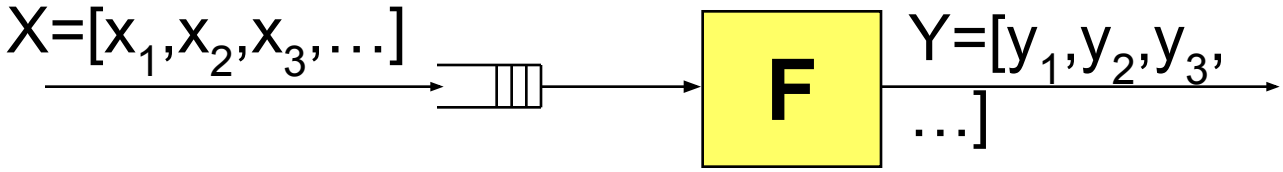
read: destructive and blocking (reading an empty channel blocks until data is available)

write: non-blocking

FIFO: infinite size

Unique attribute: determinate

KPNs: Monotonicity



- A monotonic process **F** generates from an ordered set of input sequences $\boldsymbol{X} \subseteq \boldsymbol{X}'$ an ordered set of output sequences: $\boldsymbol{X} \subseteq \boldsymbol{X}' \Rightarrow \mathbf{F}(\boldsymbol{X}) \subseteq \mathbf{F}(\boldsymbol{X}')$
  - Ordered set of sequences $\boldsymbol{X} \subseteq \boldsymbol{X}'$ if for each sequence i : $X_i \subseteq X_i'$
    $([x_1] \subseteq [x_1, x_2] \subseteq [x_1, x_2, x_3, ...])$
- Explanation:
  - Receiving more input at a process can ***only*** provoke it to send more output
  - A process does not need to have all of its input to start computing: future inputs concern ***only*** future outputs

KPNs: Determinacy

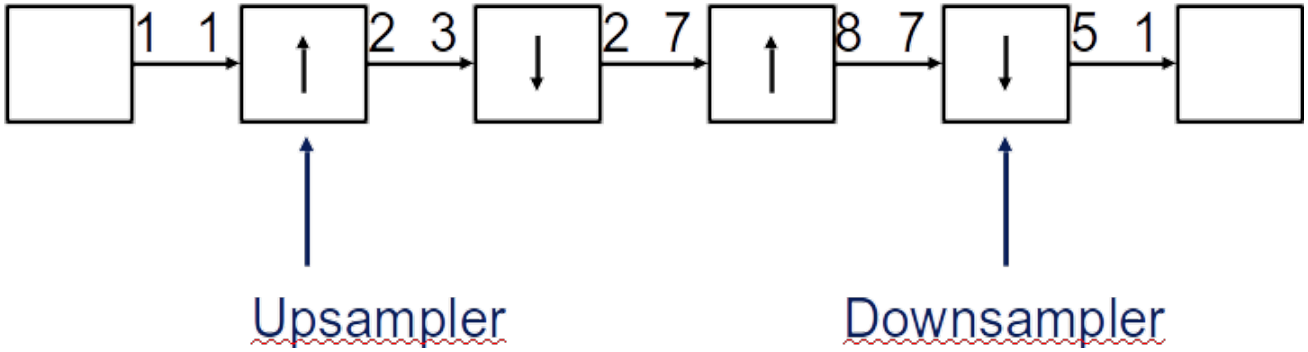
- A process network is **determinate** if histories of all channels depend ***only*** on histories of input channels
  - History of a channel: sequence of tokens that have been both written and read

- In a determinate process network, functional behavior is ***independent*** of timing

- **A KPN consisting of monotonic processes is determinate**

Synchronous Data Flow (SDF)

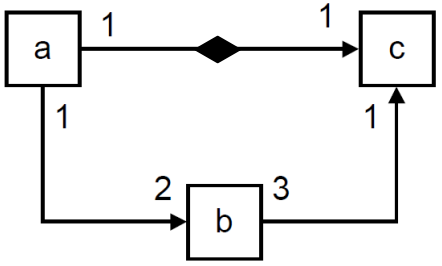
- Each process reads and writes a fixed number of tokens each time it fires



- Scheduling in two steps:
  - Establish relative execution rates for the processes (solve a system of linear equations)
  - Determine the periodic schedule(s)
- The schedule can be repeatedly executed ***without accumulating tokens*** in the buffers

Synchronous Data Flow (SDF)

- Topology matrix *M* for a SDF with *n* processes
  - A **connected** SDF has a periodic schedule **iff** *M* has rank ***r = n-1***
    (*i.e., Mq=0* has a unique smallest integer solution $q \neq 0$)
  - For an **inconsistent** SDF, *M* has rank ***r = n***
    (*i.e., Mq=0* has only the all-zeros solution)
  - For a **disconnected** SDF, *M* has rank ***r < n-1***
    (*i.e.*, *Mq=0* has two- or higher-dimensional solutions)
- Example



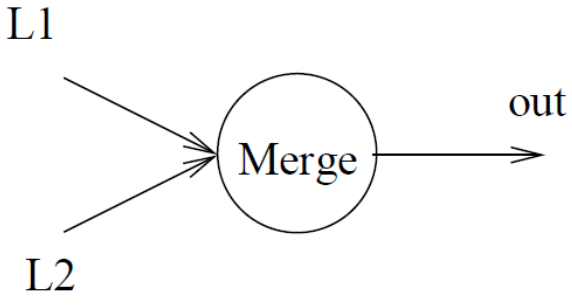
a-c = 0
a-2b = 0
3b-c = 0

$$M = \begin{bmatrix} 1 & 0 & -1 \\ 1 & -2 & 0 \\ 0 & 3 & -1 \end{bmatrix}$$

n = 3, *rank(M)* = 3
⇒ inconsistent SDF: there exists no possible schedule to execute it without an unbounded accumulation of tokens

Exercise 2.1.a: “One Peek Merge”

- Merge process that merges data tokens from input channels *L1* and *L2* into one output channel *out*
- Two different algorithms are provided



- Examine determinacy
  - *Is the output sequence determined regardless of the arrival order of the input sequences?*
- Examine fairness
  - *Does the process serve the input sequences without letting them starve, even if they have different lengths?*

Exercise 2.1.a: “One Peek Merge”

Algorithm 1

```
loop
   X=Peek(L1)
   Y=Peek(L2)
   if  X≠ φ and Y ≠ φ then
      out[X,Y], Del(L1), Del(L2)
   else if X≠ φ and Y==φ then
      out[X], Del(L1)
   else if X==φ and Y≠ φ then
      out[Y], Del(L2)
   else if X == φ and Y == φ then
      no operation
   end if
end loop
```

Exercise 2.1.a: “One Peek Merge”

Algorithm 2

```
loop
   X=Peek(L1)
   Y=Peek(L2)
   if X==φ or Y==φ then
      no operation
   else if  X == Y then
      out[X,Y], Del(L1), Del(L2)
   else if X < Y then
      out[X], Del(L1), Del(L2)
   else if X > Y then
      out[Y], Del(L1), Del(L2)
   end if
end loop
```

Exercise 2.1.b

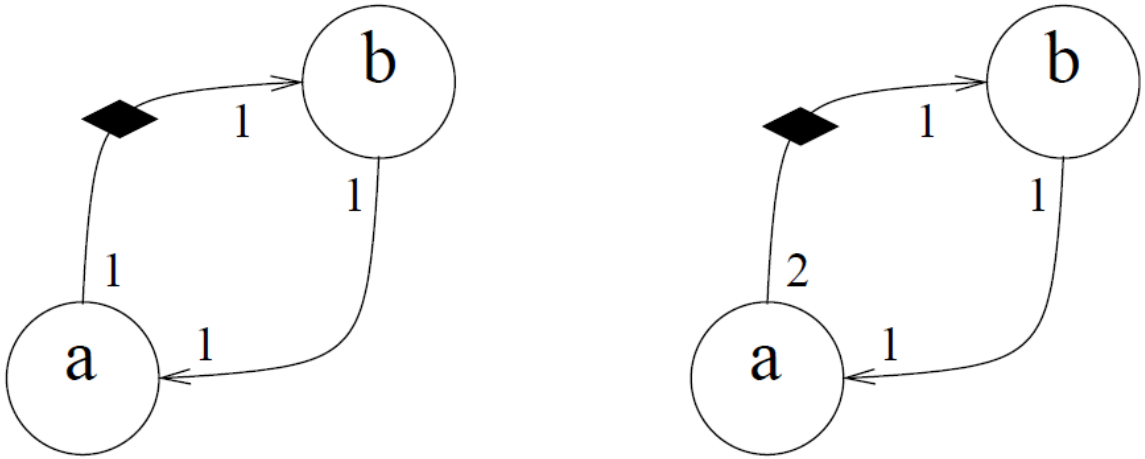
- Draw a KPN that generates the sequence *n(n+1) /2*
- Use basic processes:
  a) ***Sum of two numbers***: sends to the output channel the sum of the numbers received from the two input channels
  b) ***Product of two numbers***: sends to the output channel the product of the numbers received from the two input channels
  c) ***Duplication of a number***: sends to the two output channels the number received from the input channel
  d) ***Constant generation***: sends to the output channel firstly a constant *i* and then the number received from the input channel
  e) ***Sink process***: waits infinitely often for a number from the input channel and throw it away

Exercise 2.1.b

- **Hints**:
- *f(n) = n(n+1)/2 = 0+1+2+3+…+n*
- Transform it into a recursive expression:
  - *f(0) = 0*
  - *f(n) = n+f(n-1), n≥ 1*
- Draw the KPN starting from the recursive expression

Exercise 2.2.a

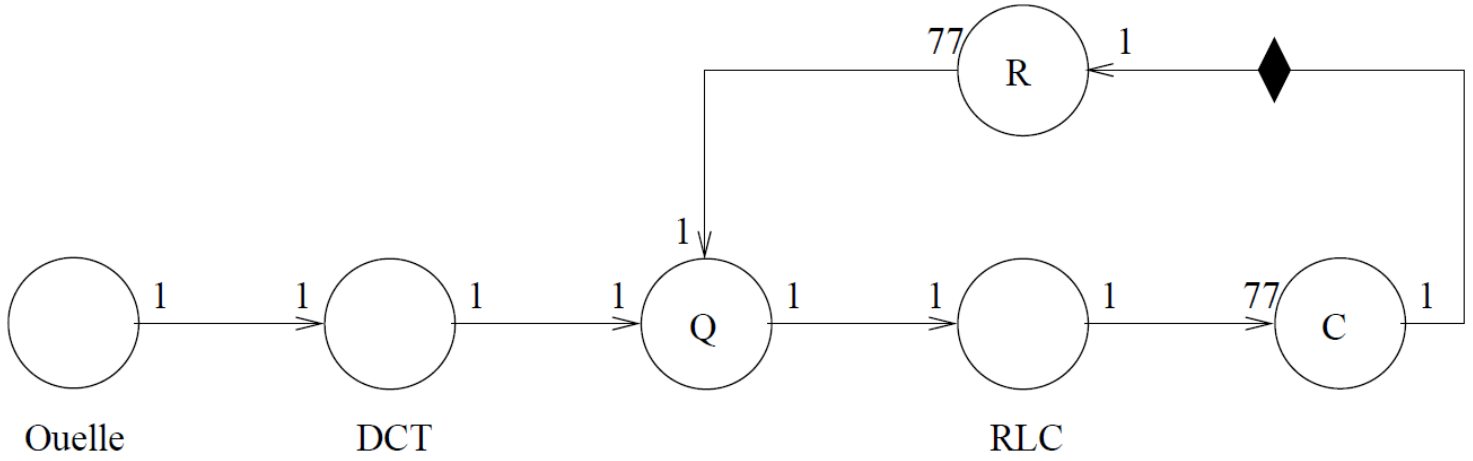
- Two SDF graphs are given:



  - Determine the topological matrices
  - Check their consistency (*i.e.*, compute the rank for *M*)
  - If consistent, determine number of firings for each node required to have a periodic execution

Exercise 2.2.b

- A SDF graph is given:



  - Determine the topological matrix
  - Check its consistency (*i.e.*, compute the rank for *M*)
  - If consistent, determine number of firings for each node required to have a periodic execution

Exercise 2.1.a: Solution

Algorithm 1

```
loop
  X=Peek(L1)
  Y=Peek(L2)
  if  X≠ φ and Y ≠ φ then
    out[X,Y], Del(L1), Del(L2)
  else if X≠ φ and Y==φ then
    out[X], Del(L1)
  else if X==φ and Y≠ φ then
    out[Y], Del(L2)
  else if X == φ and Y == φ then
    no operation
  end if
end loop
```

- Non-deterministic:
  - The output sequence depends on the arrival order of the input sequences

    I = ([x1, x2], [Ø]); I' = ([x1, x2], [y1])
    F(I) = (x1,x2); F(I')=(x1,y1,x2)
    I ⊆ I' but F(I) ⊈ F(I')

- Fair:
  - The two input sequences are served with a *First-Come-First-Serve* policy: the merge process does not let any of them starve

Exercise 2.1.a: Solution

Algorithm 2

```
loop
  X=Peek(L1)
  Y=Peek(L2)
  if X==φ or Y==φ then
    no operation
  else if X == Y then
    out[X,Y], Del(L1), Del(L2)
  else if X < Y then
    out[X], Del(L1), Del(L2)
  else if X > Y then
    out[Y], Del(L1), Del(L2)
  end if
end loop
```

- Deterministic:
  - The output sequence is determined regardless of the arrival order of the input sequences
- Unfair:
  - The merge process lets a longer sequence starve while waiting for a (possibly never appearing) token from the shorter sequence to perform the comparison

Exercise 2.1.b: Solution

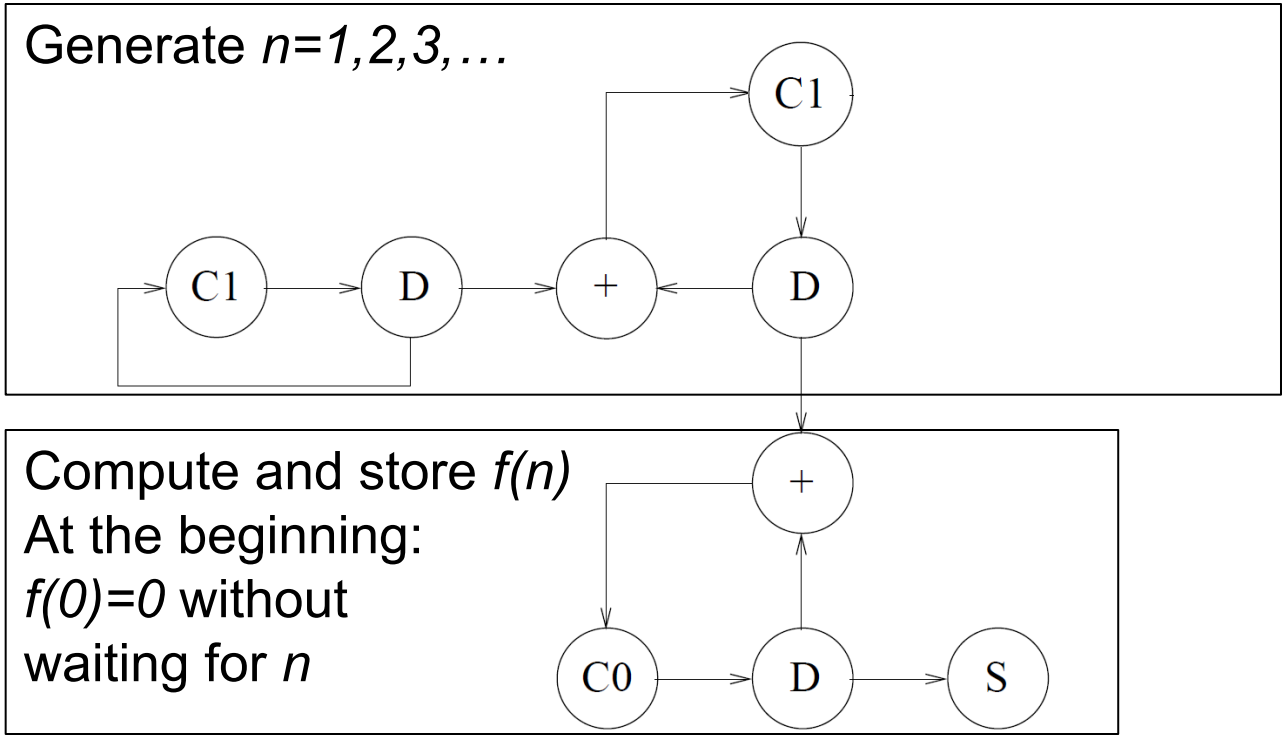
- $f(n) = n(n+1)/2 = 0+1+2+3+\ldots+n$
- $f(0) = 0, \quad f(n) = n+f(n-1), \; n \geq 1$

Generate $n=1,2,3,\ldots$

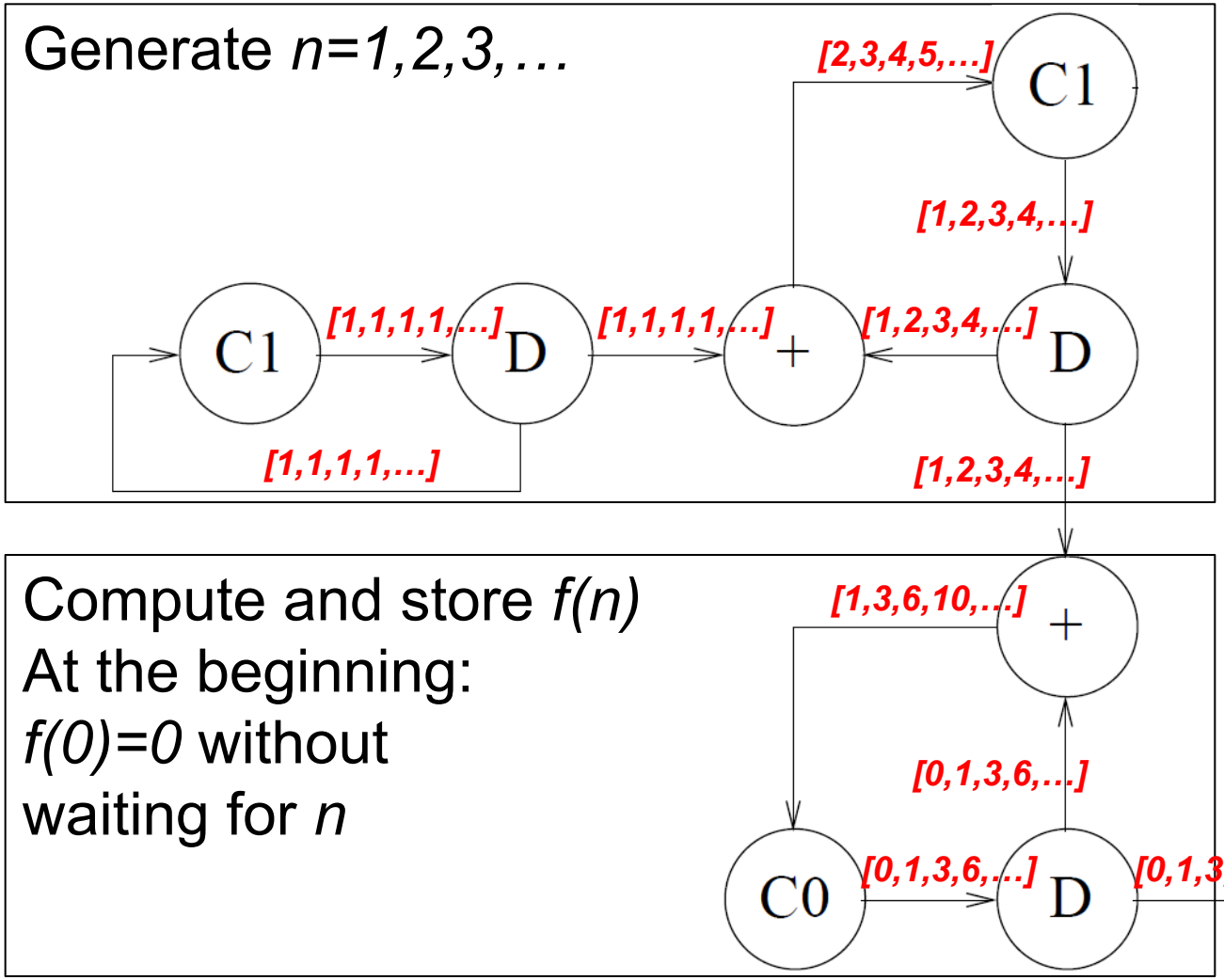
Compute and store $f(n)$
At the beginning:
$f(0)=0$ without
waiting for n

Exercise 2.1.b: Solution

- *f(n) = n(n+1)/2 = 0+1+2+3+…+n*
- *f(0) = 0, f(n) = n+f(n-1), n≥ 1*



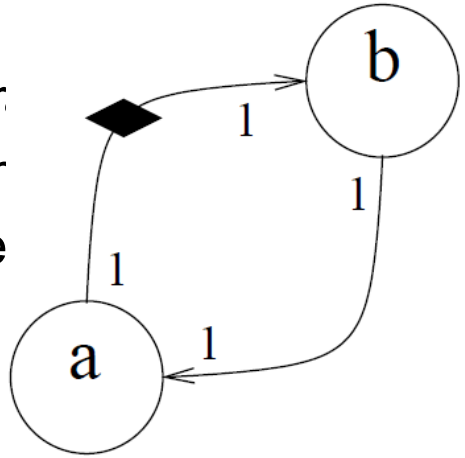
Exercise 2.1.b: Solution



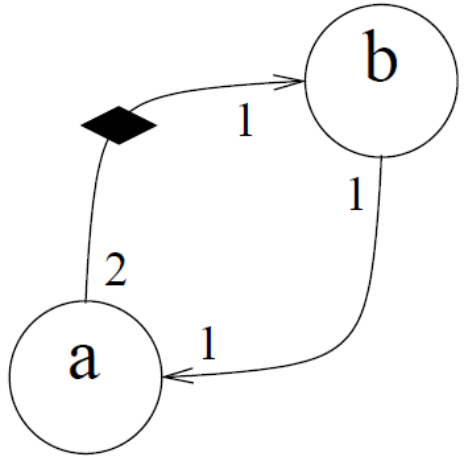
[x1,x2,x3,x4,…]: history of each channel

Exercise 2.2.a: Solution

- *n* = 2, *r*
- Fire num
- Possible



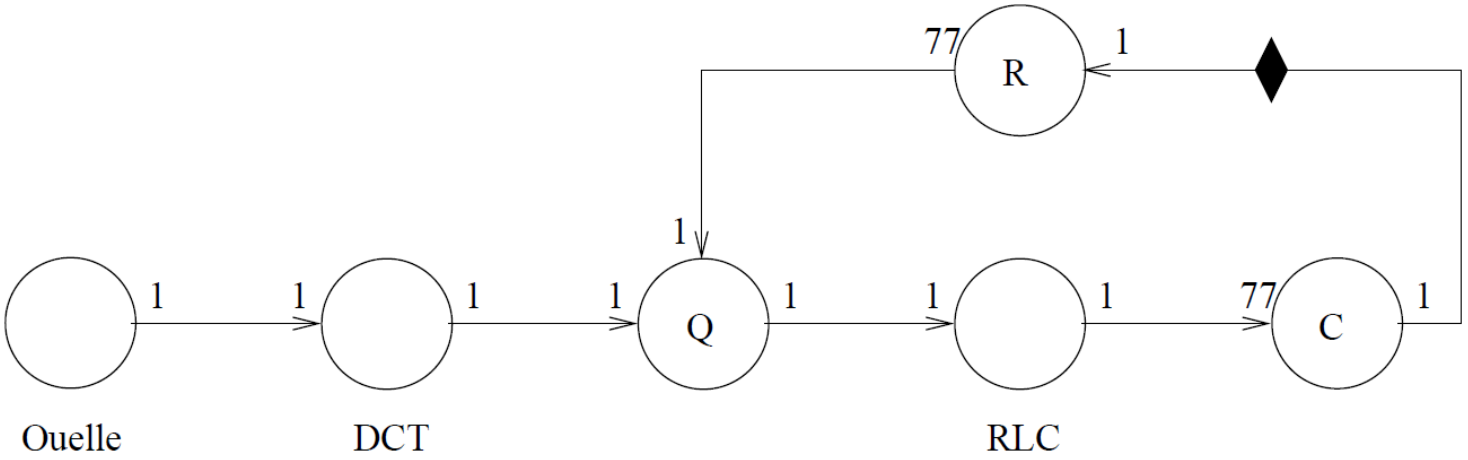
$$M = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}$$



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

- *n* = 2, *rank(M)* = 2
  => inconsistent
- No schedule can prevent from an unbounded accumulation of tokes

Exercise 2.2.b: Solution



$$M = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -77 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 0 & 0 & -77 \end{bmatrix}$$

- *n = 6,*
- *rank(M)* = 5
  *(row6=row3+row4+77*row5)*
  => consistent
- Fire numbers:
  Quelle:77, DCT:77,
  Q:77, RLC:77, C:1, R:1