Solution for Exercise 3

ConcAlgo13

LPD, EPFL

October 15, 2013

The Splitter Object

- Only one operation: *splitter*
- Returns: *stop*, *left* or *right*
- If a single process executes *splitter*, then this process gets *stop*.
- If two or more processes invoke *splitter*, then not all get the same output.
- At most one process gets *stop*.

An Implementation of a Splitter

We use two registers:

- P (multi-valued), and
- S (binary, initialized to *false*)

```
upon splitter_i
    $P \leftarrow i$;
    if $S$ then return "right";
    $S \leftarrow true$;
    if $P = i$ then return "stop";
    return "left";
```

An Implementation of a Splitter

We use two registers:

- *P* (multi-valued), and
- *S* (binary, initialized to *false*)

```
upon $splitter_i$
    $P \leftarrow i$;
    if $S$ then return "right";
    $S \leftarrow true$;
    if $P = i$ then return "stop";
    return "left";
```

Non-adaptive Snapshot

```
upon $scan_i$
    $t_1 \leftarrow collect(),\ t_2 \leftarrow t_1$;
    while $true$ do
        $t_3 \leftarrow collect()$;
        if $t_3 = t_2$ then return $\langle\, t_3[1].val, \ldots, t_3[N].val \rangle$ ;
        for $k \leftarrow 1$ to $N$ do
            if $t_3[k].ts \geq t_1[k].ts + 2$ then return $t_3[k].snapshot$;
        $t_2 \leftarrow t_3$;

procedure $collect()$
    for $k \leftarrow 1$ to $N$ do
        $x[k] \leftarrow R[k]$;
    return $x$;
```

Non-adaptive Snapshot (2)

```
procedure $update_i(v)$
    $ts \leftarrow ts + 1$;
    $snapshot \leftarrow scan()$;
    $R[i] \leftarrow \langle ts, v, snapshot \rangle$;
```

Adaptive Update

```
procedure $update(v)$
    if $myreg = \perp$ then
        $myreg \leftarrow obtain()$;

    $ts \leftarrow ts + 1$;
    $snapshot \leftarrow scan()$;
    $R[myreg] \leftarrow \langle ts, v, snapshot \rangle$;
```

Adaptive Scan

```
upon $scan_i$
    $t_1 \leftarrow collect(), t_2 \leftarrow t_1$;
    while $true$ do
        $t_3 \leftarrow collect()$;
        if $t_3 = t_2$ then return $\langle t_3[1].val, \ldots, t_3[t_3.length].val \rangle$ ;
        for $k \leftarrow 1$ to $t_3.length$ do
            if $t_3[k].ts \geq t_1[k].ts + 2$ then return $t_3[k].snapshot$;
        $t_2 \leftarrow t_3$;
```

A Disallowed Solution

```
procedure obtain()
    myreg ← C.fetch&inc();

procedure collect()
    for k ← 1 to C.read() do
        x[k] ← R[k];
    return x;
```

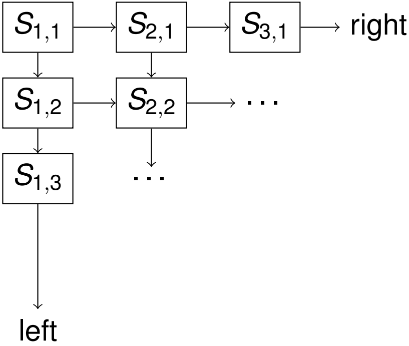
But we can use only registers!

The Splitter Object

- One operation: *splitter*
- Returns: *stop*, *left* or *right*
- If a single process executes *splitter*, then *stop* is returned.
- If two or more processes invoke *splitter*, then not all get the same output.
- At most one process gets *stop*.

Main Idea of Adaptive Snapshot

- Matrix of registers and splitters
- To obtain a register, a process must find a splitter that returns *stop*.
- Process starts from left top corner and follows the output of splitters.

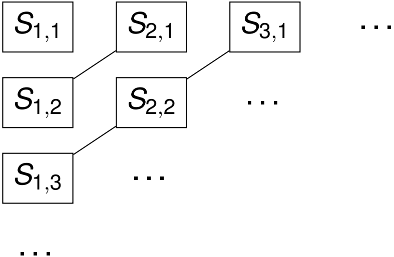


The Obtain Operation

```
procedure obtain()
    $x \leftarrow 1, y \leftarrow 1$;
    while true do
        $s \leftarrow S[x, y].splitter()$;
        if $s =$ ”stop” then
            $myreg \leftarrow \langle x, y \rangle$;
            return;
        else if $s =$ ”left” then $y \leftarrow y + 1$;
        else $x \leftarrow x + 1$;
```

The Collect Operation

```
procedure collect
    $C \leftarrow \langle\rangle$;
    $d \leftarrow 1$;
    while diagonal $d$ has a splitter that has been traversed do
        $C \leftarrow C \cdot \langle$ values of all non-$\bot$ registers on diagonal $d$ $\rangle$;
        $d \leftarrow d + 1$;
    return $C$;
```



$S_{1,1}$
$S_{2,1}$
$S_{3,1}$
$\cdots$
$S_{1,2}$
$S_{2,2}$
$\cdots$
$S_{1,3}$
$\cdots$
$\cdots$