Unit-3: Linear-time properties

B. Srivathsan

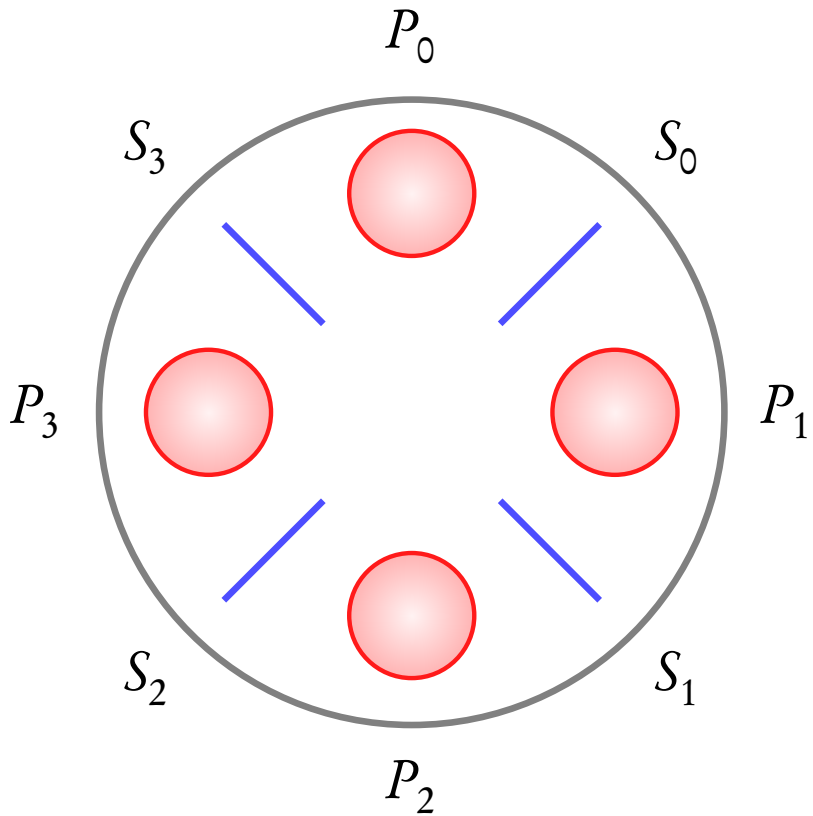
Chennai Mathematical Institute

NPTEL-course

July - November 2015

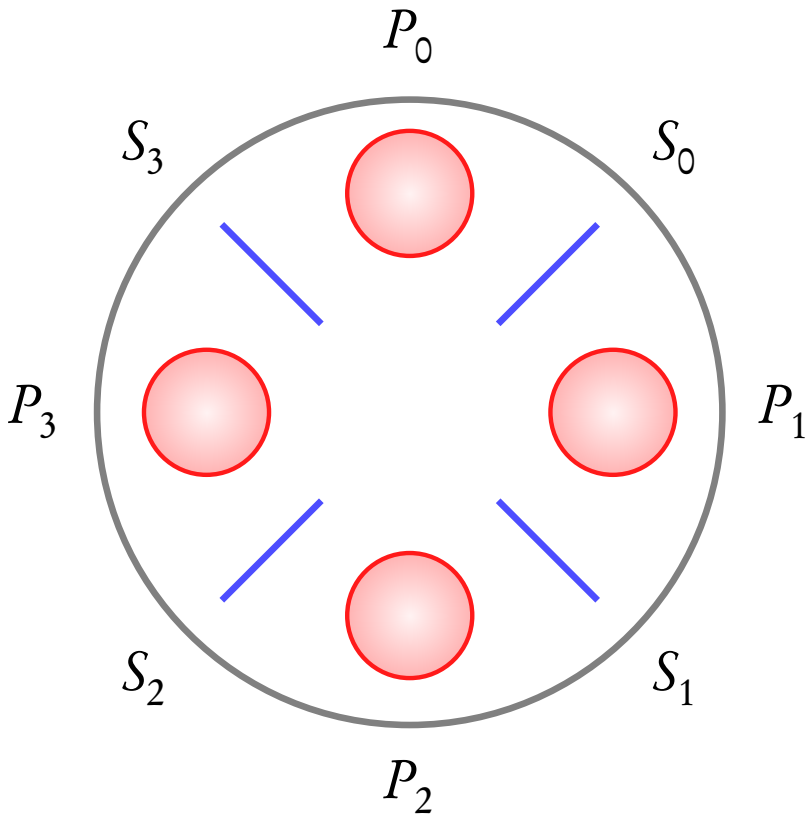
Module 1:

A problem in concurrency



$P_0 \ldots P_3$: *processes*

$S_0 \ldots S_3$: *resources*



$P_0 \ldots P_3$: *processes*

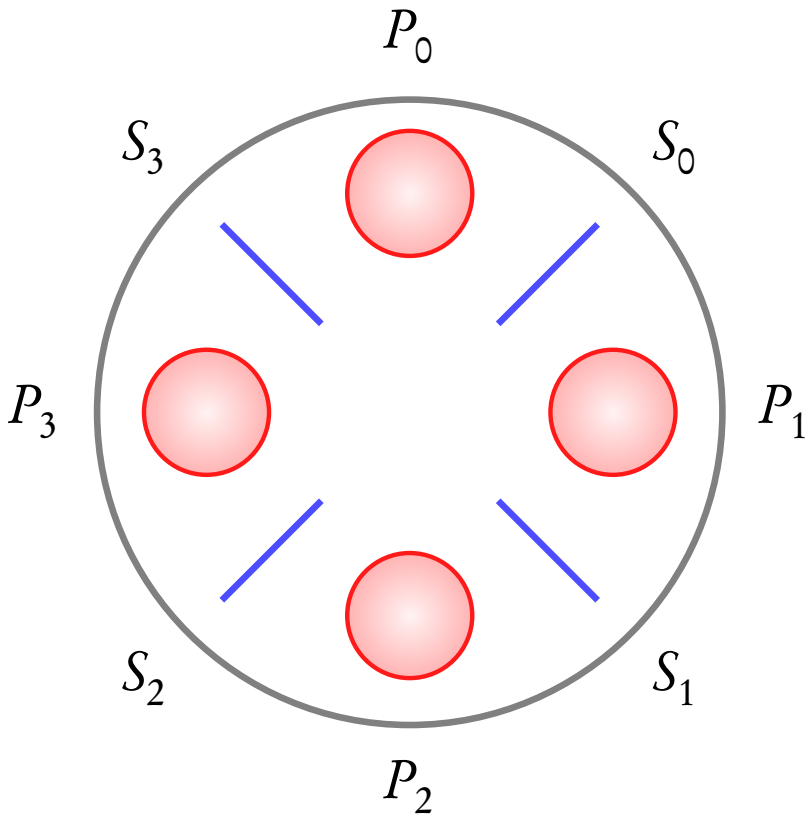
$S_0 \ldots S_3$: *resources*

Process P_i can **execute**

only if

it has access to **resources**

$S_{(i-1)}$ and S_i



$P_0 \ldots P_3$: *processes*

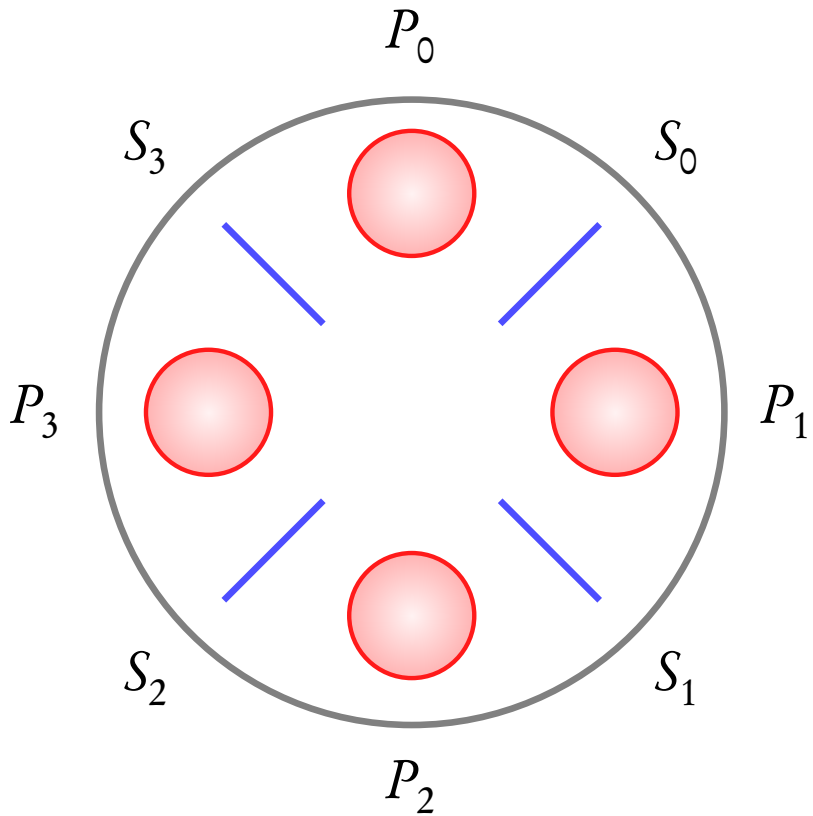
$S_0 \ldots S_3$: *resources*

Process P_i can **execute**

only if

it has access to **resources**

$S_{(i-1) \bmod 4}$ and $S_{i \bmod 4}$



$P_0 \ldots P_3$: *processes*

$S_0 \ldots S_3$: *resources*

Process P_i can **execute**

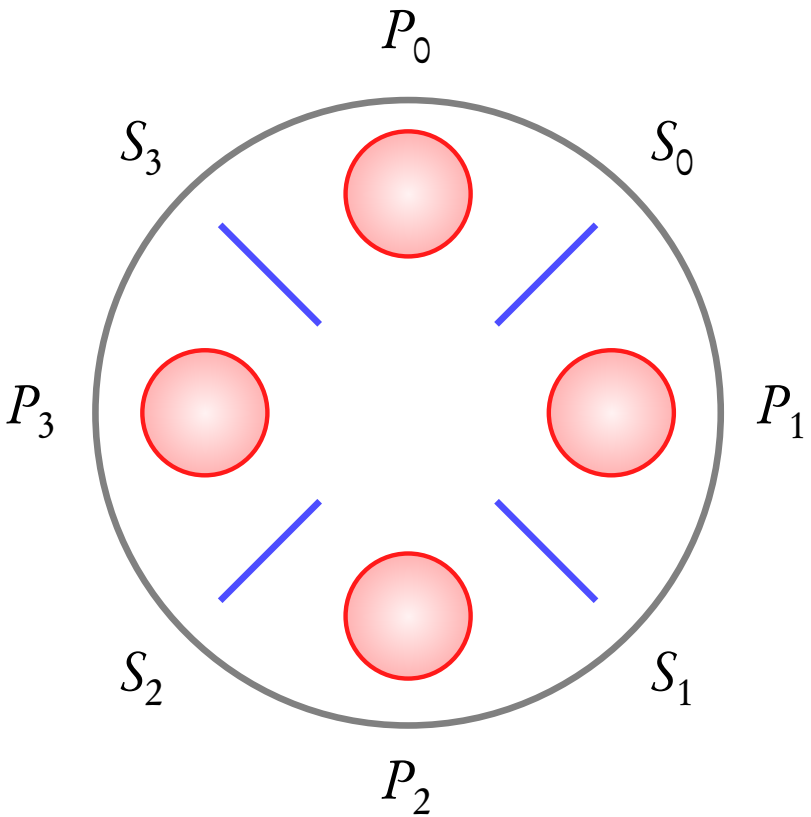
only if

it has access to **resources**

$S_{(i-1) \bmod 4}$ and $S_{i \bmod 4}$

How should the processes be **scheduled** so that **every process** can execute **infinitely often?**

Dining philosophers problem (Dijkstra)



$P_0 \ldots P_3$: *philosophers*

$S_0 \ldots S_3$: *chop-sticks*

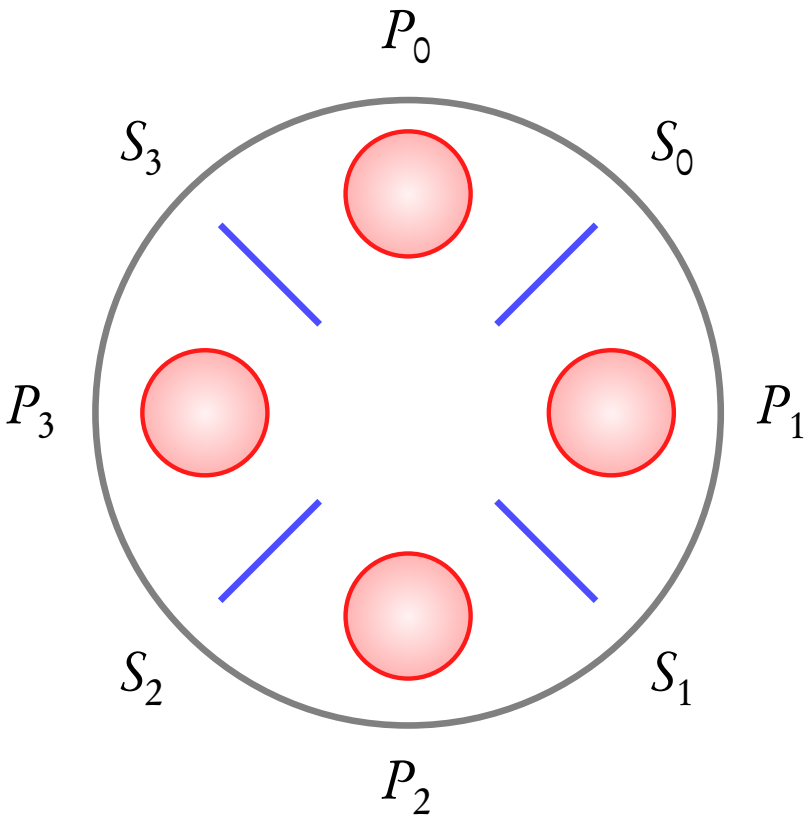
Philosopher P_i can **eat**

only if

he has access to **chop-sticks**

$S_{(i-1) \bmod 4}$ and $S_{i \bmod 4}$

Dining philosophers problem (Dijkstra)



$P_0 \ldots P_3$: *philosophers*

$S_0 \ldots S_3$: *chop-sticks*

Philosopher P_i can **eat**

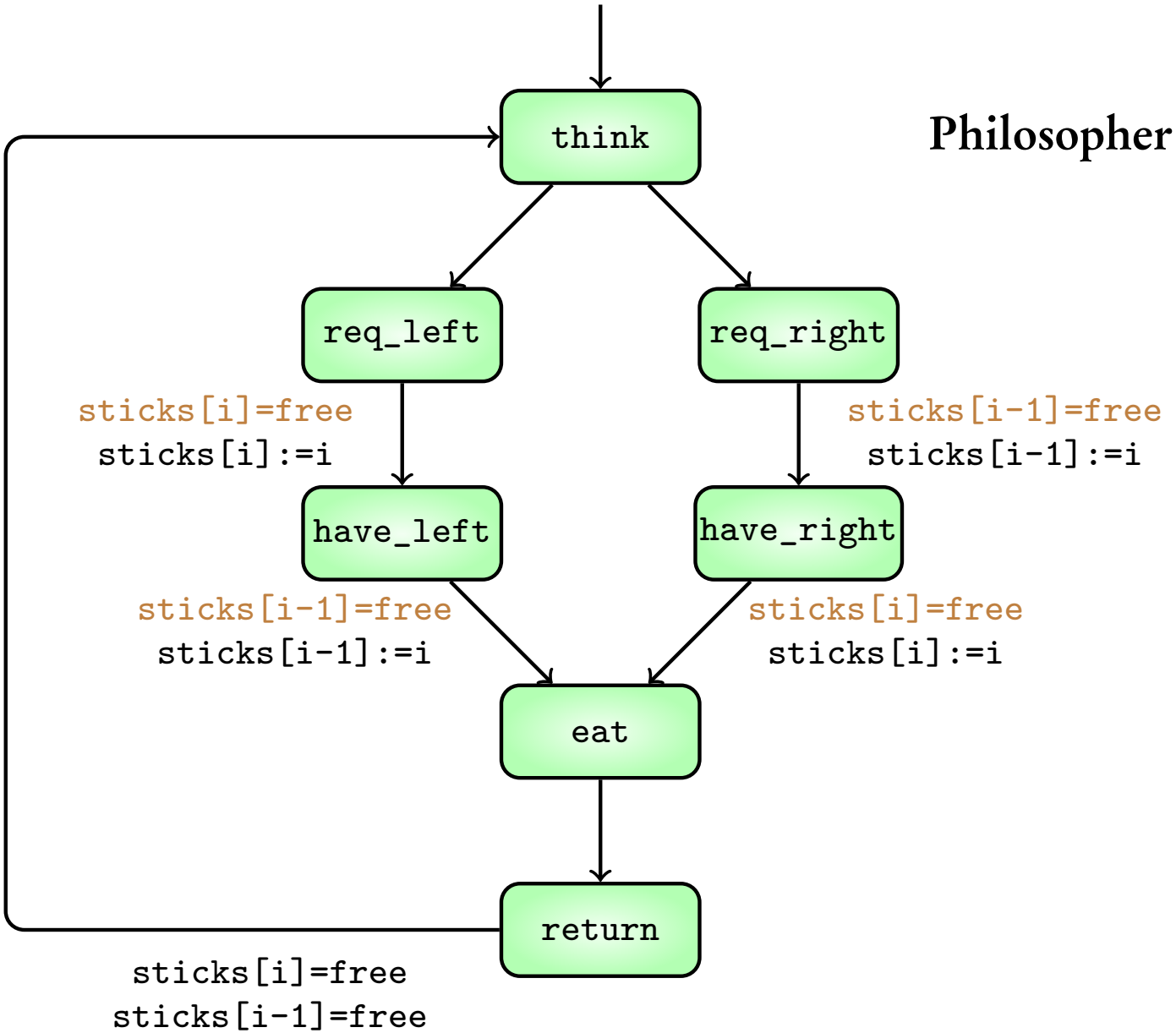
only if

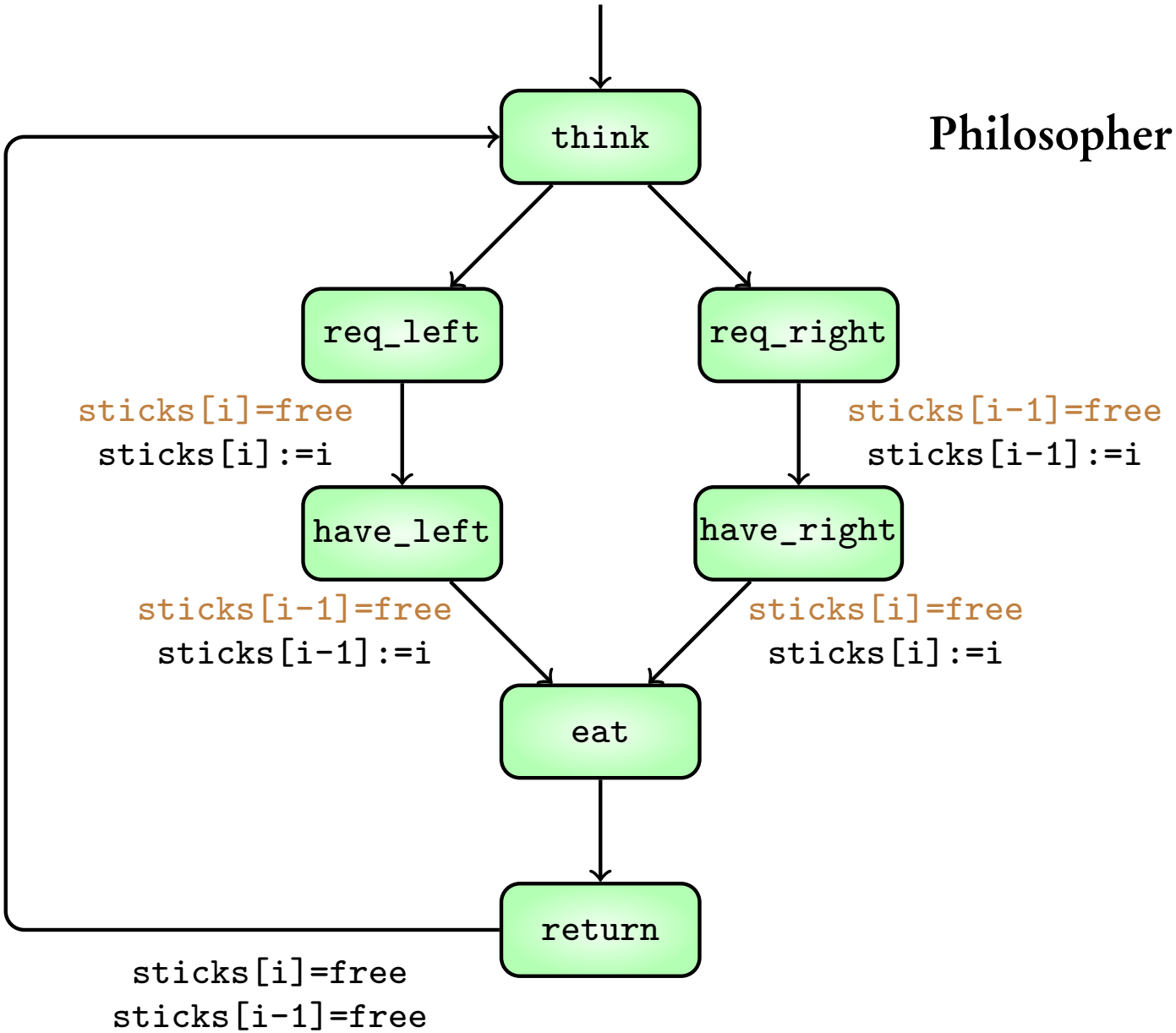
he has access to **chop-sticks**

$S_{(i-1) \bmod 4}$ and $S_{i \bmod 4}$

What should the **protocol** be so that **every philosopher** can eat **infinitely often?**

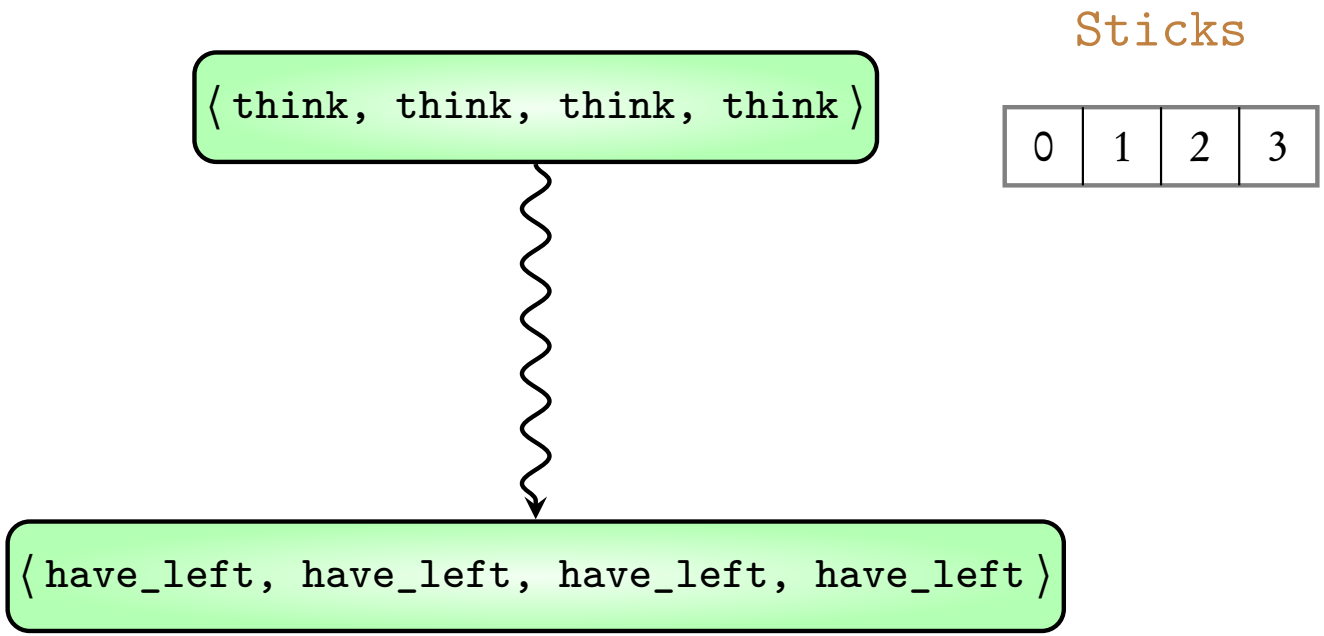
Coming next: A protocol for the dining philosophers


Philosopher i
think
req_left
req_right
sticks[i]=free
sticks[i]:=i
sticks[i-1]=free
sticks[i-1]:=i
have_left
have_right
sticks[i-1]=free
sticks[i-1]:=i
sticks[i]=free
sticks[i]:=i
eat
return
sticks[i]=free
sticks[i-1]=free


Philosopher i
think
req_left
req_right
sticks[i]=free
sticks[i]:=i
sticks[i-1]=free
sticks[i-1]:=i
have_left
have_right
sticks[i-1]=free
sticks[i-1]:=i
sticks[i]=free
sticks[i]:=i
eat
return
sticks[i]=free
sticks[i-1]=free

NuSMV demo

A deadlock


⟨ think, think, think, think ⟩
⟨ have_left, have_left, have_left, have_left ⟩
Sticks
0
1
2
3

In this unit...

What **properties** should be checked to **detect deadlocks?**

In this unit...

What **properties** should be checked to **detect deadlocks?**

- Module 2: Attach a mathematical meaning to properties

In this unit...

What **properties** should be checked to **detect deadlocks?**

- Module 2: Attach a mathematical meaning to properties
- Module 3, 4: Different examples of properties

In this unit...

What **properties** should be checked to **detect deadlocks?**

- Module 2: Attach a mathematical meaning to properties
- Module 3, 4: Different examples of properties
- Module 5: Answer to the question