

## CSP

### Guarded Commands

- Monitor: begin executing every call as soon as possible, waiting if the object is not in a proper state and signaling when the state is proper
- CSP: the called object establishes conditions under which the call is accepted; calls not satisfying these conditions are held pending (no need for programmed wait/signal operations).

### Rendezvous

- Monitor: the monitor is passive (has no independent task/thread/activity)
- CSP: synchronization between peer, autonomous activities.

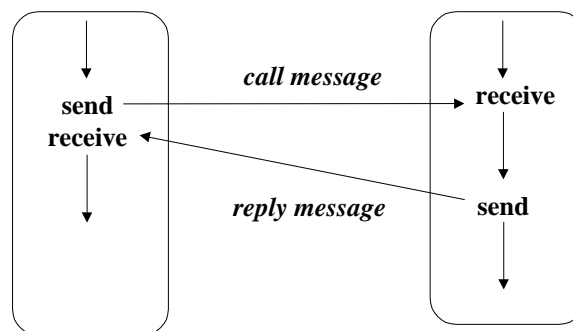
CS 5204 Spring 99

1

## CSP

### Distribution:

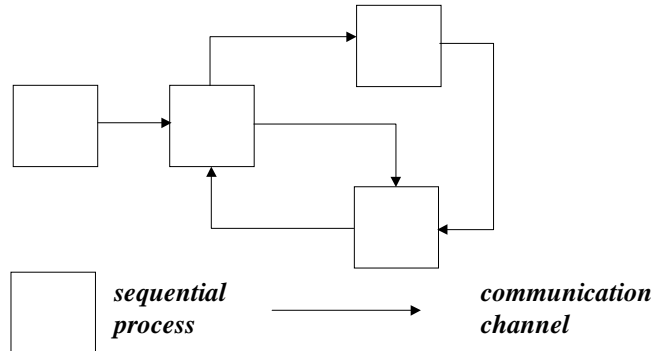
- Monitor: inherently non-distributed in outlook and implementation
- CSP: possibility for distributed programming using synchronous message passing



CS 5204 Spring 99

2

## Communicating Sequential Processes (CSP)

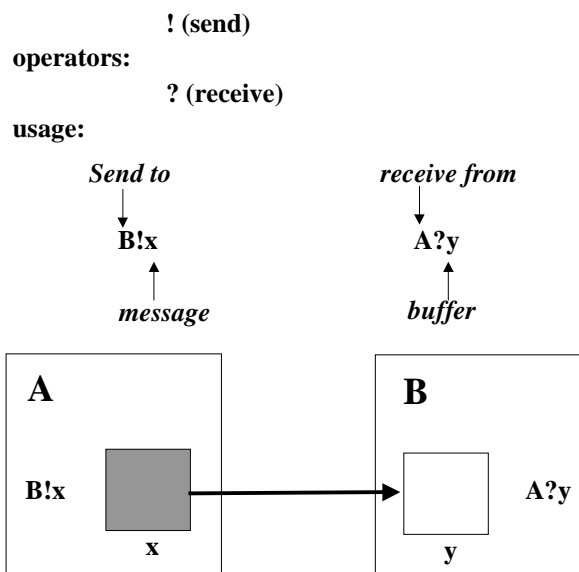


- single thread of control
- autonomous
- encapsulated
- named
- static
- synchronous
- reliable
- unidirectional
- point-to-point
- fixed topology

CS 5204 Spring 99

3

## Communicating Sequential Processes (CSP)

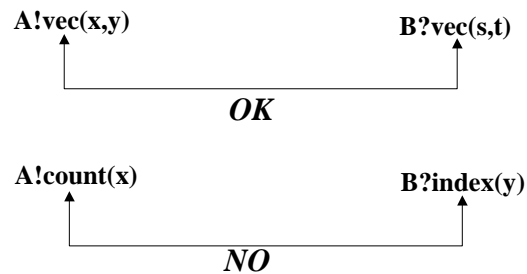


CS 5204 Spring 99

4

## Communicating Sequential Processes (CSP)

- rendezvous semantics: senders (receivers) remain blocked at send (receive) operation until a matching receive (send) operation is made.
- typed messages: the type of the message sent by the sender and the type of the message expected by the receiver must match (otherwise abort).



CS 5204 Spring 99

5

## Communicating Sequential Processes (CSP)

### Guarded Commands

**<guard>  $\Rightarrow$  <command list>**

$\uparrow$   
 \_\_\_\_\_ **boolean expression**  
 \_\_\_\_\_ **only one ? , must be at end of guard,**  
                   **considered true iff message pending**

### Examples

**$n < 10 \Rightarrow A!\text{index}(n); n := n + 1;$**   
 **$n < 10; A?\text{index}(n) \Rightarrow \text{next} = \text{MyArray}(n);$**

CS 5204 Spring 99

6

## Communicating Sequential Processes (CSP)

### Alternative Command

**[ G1  $\Rightarrow$  S1 [] G2  $\Rightarrow$  S2 [] ... [] Gn  $\Rightarrow$  Sn ]**

1. evaluate all guards
2. if more than one guard is true, nondeterministically select one.
3. if no guard is true, terminate.

**Note:** if all true guards end with an input command for which there is no pending message, then delay the evaluation until a message arrives. If all senders have terminated, then the alternative command terminates.

### Repetitive Command

**\* [ G1  $\Rightarrow$  S1 [] G2  $\Rightarrow$  S2 [] ... [] Gn  $\Rightarrow$  Sn ]**

repeatedly execute the alternative command  
until it terminates

CS 5204 Spring 99

7

## Communicating Sequential Processes (CSP)

### Examples:

```
[x >= y --> m := x [] y >= x --> m := y ]
i := 0; * [ i < size; content(i) != n --> i := i + 1 ]
* [ c: character; west?c --> east!c ]
* [ n: integer; X?insert(n) --> INSERT
  []
  n: integer; X?has(n) --> SEARCH; X!(i < size) ]
```

### BoundedBuffer::

```
buffer: (0..9) portion;
in, out: integer; in := 0; out := 0;
* [ in < out + 10; producer?buffer(in mod 10)
  --> in := in + 1;
  []
  out < in; consumer?more()
  --> consumer!buffer(out mod 10);
  out := out + 1;
]
```

CS 5204 Spring 99

8

## ADA Example

```
task bounded-buffer is
  entry store(x : buffer);
  entry remove(y: buffer);
end;
task body bounded-buffer is
  ...declarations...
begin
  loop
    select
      when head < tail + 10 =>
        accept store(x : buffer) ... end store;
    or
      when tail < head =>
        accept remove(y: buffer) ... end remove;
    end select;
  end loop
end
```