Concurrent Object Oriented Languages

Synchronous Message Passing

wiki.eecs.yorku.ca/course/6490A

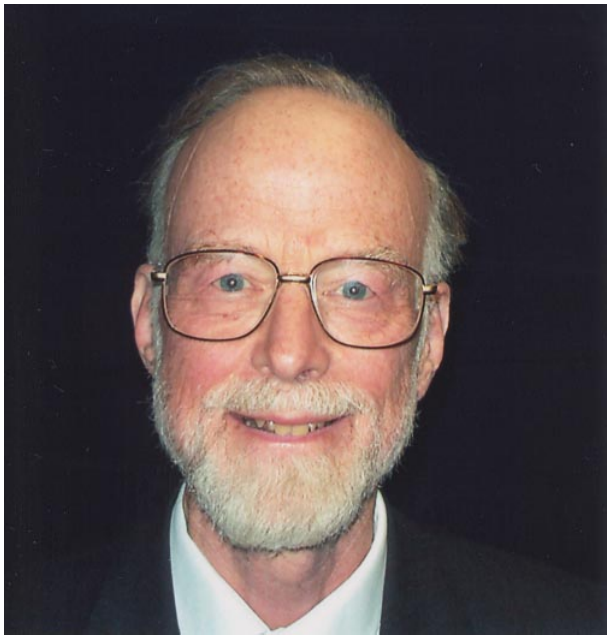
Assignment 1

Assignment 1 is due January 27.

- Find a paper.
- Email me (a link to) the paper.
- Once approved, add the paper to the course Wiki (go the Assignment 1)
- Study the paper.
- Look for related work (published before and after the paper you chose).
- Write your report.

Communicating Sequential Processes (CSP)

C.A.R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666-677, August 1978.

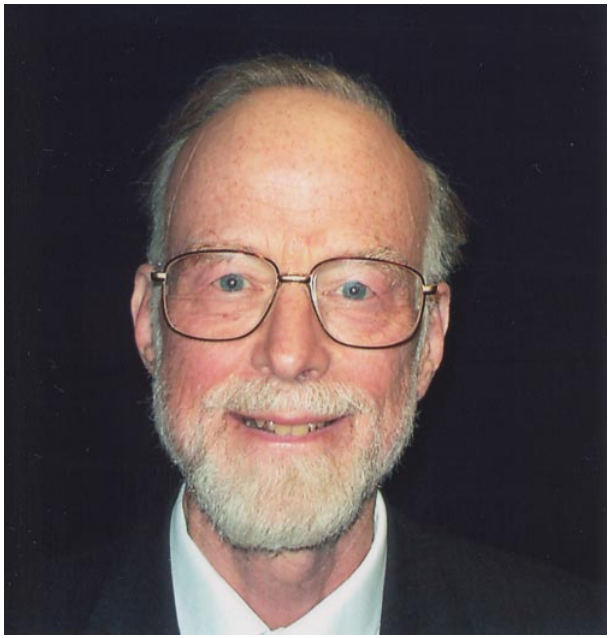


sir Charles Antony Richard (Tony) Hoare

source: cs.ox.ac.uk

Communicating Sequential Processes (CSP)

C.A.R. Hoare. *Communicating Sequential Processes*. 1985.



sir Charles Antony Richard (Tony) Hoare

source: cs.ox.ac.uk

Syntax of CSP

CSP has static process creation.

[name :: command $\|$ $\cdots$ $\|$ name :: command]

Syntax of CSP

CSP uses synchronous message passing to communicate.

- Receive command
  name?pattern
- Send command
  name!expression

Communication in CSP

[sender :: receiver!(1,2) $\|$ receiver :: sender?(1,x)]

As a result of the communication, the variable x is assigned the value 2.

Communication in CSP

[sender :: receiver!(1,2) $\|$ receiver :: sender?(3,x)]

No communication takes place since the expression (1,2) does not match the pattern (3,x).

Syntax of CSP

Conditional command

[guard $\rightarrow$ command $\Box \cdots \Box$ guard $\rightarrow$ command]

guard

- Boolean expression
- receive command
- Boolean expression ; guard

Syntax of CSP

Iteration command

$*$[guard $\rightarrow$ command $\Box \cdots \Box$ guard $\rightarrow$ command]

guard

- Boolean expression
- receive command
- Boolean expression ; guard

Examples in CSP

Express a semaphore, named semaphore, and a process, named process, using that semaphore to protect its critical section.

Examples in CSP

Let us now assume that there are two processes, named process1 and process2. How do we change the semaphore?

Examples in CSP

Express the consumer-producer problem. The producer, named producer, produces the integers $1, \ldots, 100$ and the consumer, named consumer, prints the integers it consumes. Both interact with the buffer, named buffer.

Examples in CSP

Let

```
reader(i) ::
*[scheduler!request();
  read();
  scheduler!done()]

writer(i) ::
*[scheduler!request();
  write();
  scheduler!done()]
```

Implement scheduler to solve the readers-writers problem.

Examples in CSP

What is wrong with

```
phil(i) ::
*[THINK;
  fork(i)!pickup(); fork((i+1) mod N)!pickup();
  EAT;
  fork(i)!putdown(); fork((i+1) mod N)!putdown()]

fork(i) ::
*[  phil(i)?pickup()
      → phil(i)?putdown()
  □ phil((i−1) mod N)?pickup()
      → phil((i−1) mod N)?putdown()]
```

Examples in CSP

The sieve of Eratosthenes is a simple, ancient algorithm for finding all prime numbers up to a specified integer.

Eratosthenes

Examples in CSP

Processes:

- generator that generates 2, 3, ...
- sieve(i), for $1 \leq i \leq n$, where n is the number of primes to be generated (sieve(n) is defined differently).

Examples in CSP

```
sieve(0) :: n = 2;
            *[ sieve(1)!n → n = n + 1 ]

sieve(i) :: sieve(i − 1)?p;
            print(p);
            *[ sieve(i − 1)?n →
                 [ n%p == 0 → skip
                   □ n%p != 0 → sieve(i + 1)!n]

sieve(100) :: sieve(99)?p; print(p)
```