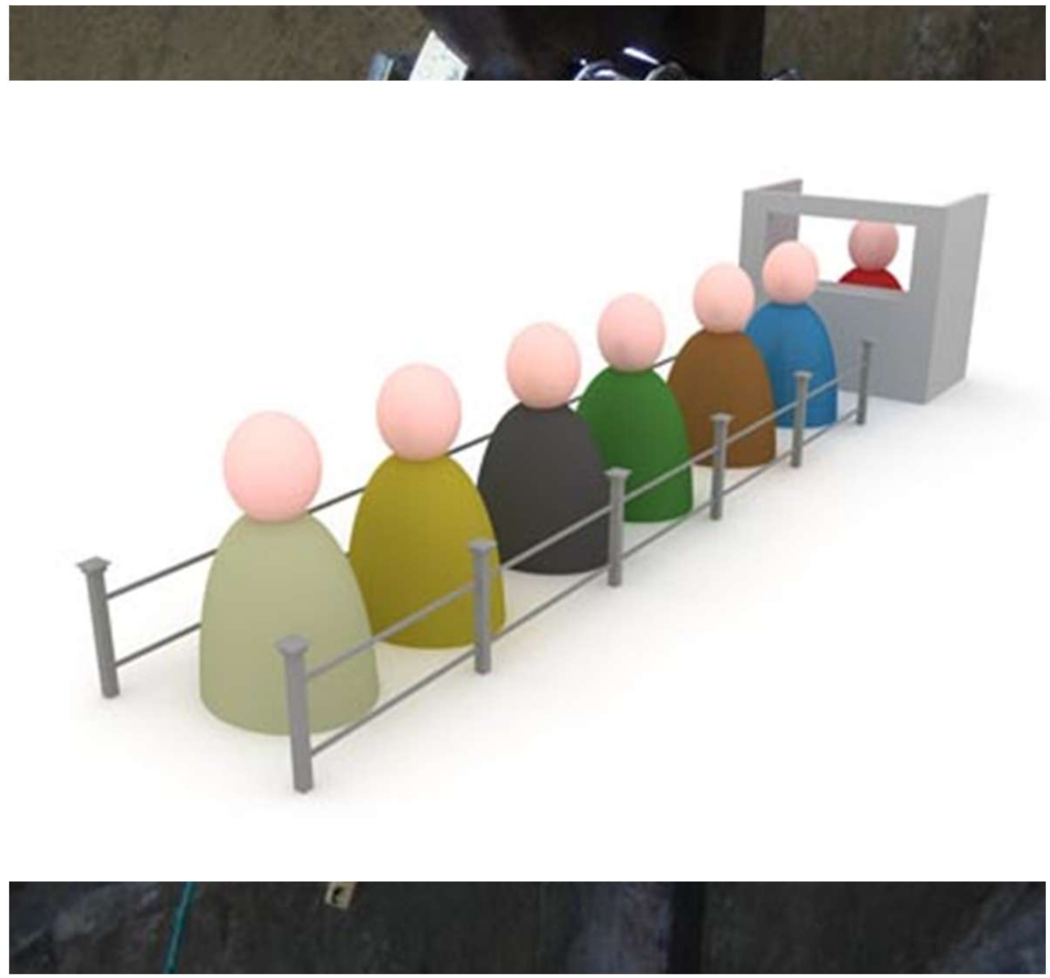
Wait-Free Synchronization

Maurice Herlihy
ACM TOPLAS, Jan. 1991, New York, USA

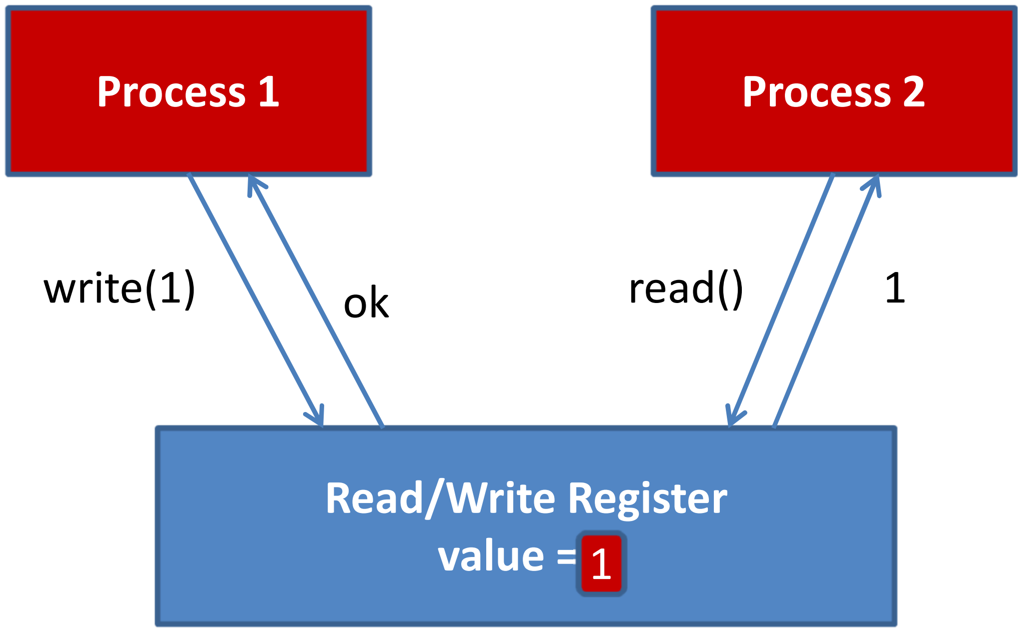
Presented by Martin Enev
ETH Zurich

Problems with locks

- Deadlock
- Lock overhead
- Lock contention
- More…



Objects and processes



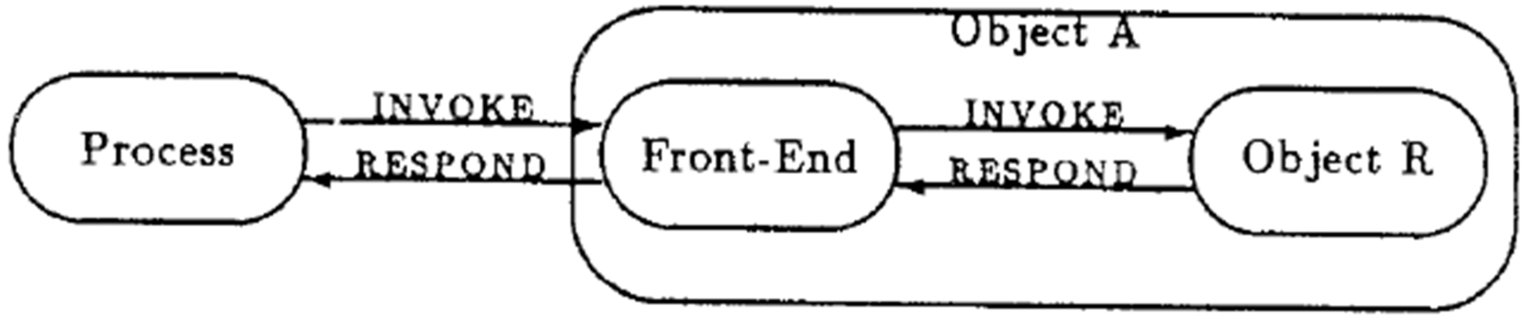
Wait-free implementation

- A *wait-free data structure* guarantees that *any process* can complete *any operation* in a *finite* number of steps
- Provides *fault-tolerance*
- Can we make *any* object wait-free?
- What primitives are necessary / sufficient for constructing wait-free objects?

The model

- A *concurrent system* {n Processes; m Objects}
- Events:
  - INVOKE(P, op, O)—op is an operation of O
  - RESPOND(P, res, O)—res is a result value
- An object's operations must be *total*
  - If the object has a pending operation there is a matching enabled response

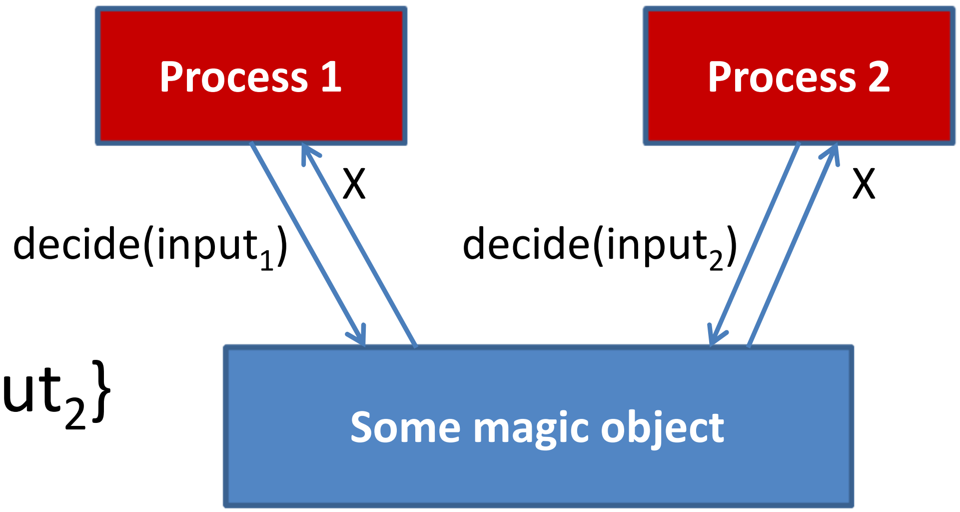
Implementation



- $\{F_1,\ldots,F_n; R\}$
- R is the representation object
- F_i is the procedure called by process P_i

What is a Consensus Protocol?

- A concurrent system, where
  - Each process starts with input value
  - Processes communicate via objects
  - The processes agree on a *common input value*
- Required to be
  - Consistent
  - Wait-free
  - Valid – $x \in \{input_1, input_2\}$



Consensus number

- The *consensus number* of the object X is the maximum number *N* of processes for which there exists a consensus protocol

  $\{ P_1 \ldots P_N ; X \}$

- Could be *infinite*

Hierarchy of objects

- *Theorem:* If X has consensus number *n*, and Y has consensus number $m < n$, then *there exists no wait-free implementation* of X by Y in a system of more than *m* processes.

- Implies that there is a *hierarchy* where each level *n* of the hierarchy contains concurrent objects with consensus number *n*

Proof Outline

By contradiction. Assume X has consensus number n, and Y has consensus number $m < n$. Let $k > m$, assume for contradiction that X = { G_1 ... G_k ; Y } has consensus number k.

1. { P_1 ... P_k ; X } is a consensus protocol
2. { P_1 ... P_n ; { G_1 ... G_n ; Y } } is wait-free
3. { $P_1 \cdot G_1$ ... $P_n \cdot G_n$; Y } is a consensus protocol because composition is associative

Consensus numbers

Consensus Number	Object
1	Atomic read/write registers
2	test&set, fetch&add
2n-2	n-register assignment
∞	compare&swap

Compare&Swap Register

- *Theorem:* A CAS register has *infinite* consensus number.

```
value_t decision = INIT;
value_t decide( value_t input) {
first = CAS( &decision, INIT, input);
if ( first == INIT ) // CAS succeeded?
  return input;
else
  return first;
```

Universality results

- An object is *universal* if it can be used to construct a wait-free implementation of *any* object (it has consensus number ∞).
- In a system of *n* processes, an object is universal if and only if the object has consensus number *n*.
- CAS has consensus number ∞ and thus is a universal object.

Impact

- 1991 paper
- 1200 citations
- More than 1 citation per week over the past 20 years
- Fundamental paper

Summary

- Wait-free synchronization provides *guaranteed progress* to all correct processes
- There is a wait-free *hierarchy* determined by an object's *consensus number*
- Compare&Swap is a *universal* primitive and thus can be used to implement any wait-free object

Discussion