Sapienza
Università di Roma
Dipartimento di Informatica

CONCURRENT SYSTEMS
LECTURE 6

Prof. Daniele Gorla

Atomicity

We have a set of n sequential processes p1,…,pn that access m concurrent objects X1,…,Xm by invoking operations of the form Xi.op(args)(ret).

When invoked by pj, the invocation Xi.op(args)(ret) is modeled by two events:

inv[Xi.op(args) by pj] and res[Xi.op(ret) to pj].

A **history** (or **trace**) is a pair $\widehat{H} = (H , <_H)$ where H is a set of events and $<_H$ is a total order on them

The *semantics* (of systems and/or objects) will be given as a set of traces.

A history is **sequential** if it is of the form inv res inv res … inv res inv inv inv … (where every res is the return operation of the immediately preceeding inv)

→ a sequential history can be represented as a sequence of operations

A history is **complete** if every inv is eventually followed by a corresponding res, **partial** otherwise.

Linearizability

Def.: a complete history $\widehat{H}$ is **linearizable** if there exists a sequential history $\hat{S}$ s.t.

1. $\forall X \,.\, \hat{S}|_X \in \text{semantics}(X)$
2. $\forall p \,.\, \widehat{H}|_p = \hat{S}|_p$
3. If $\text{res}[op] <_H \text{inv}[op']$, then $\text{res}[op] <_S \text{inv}[op']$

Given an history $\widehat{K}$, we can define a binary relation on events $\longrightarrow_K$ s.t. $(op, op') \in \longrightarrow_K$ if and only if $\text{res}[op] <_K \text{inv}[op']$. We write $op \longrightarrow_K op'$ for denoting $(op, op') \in \longrightarrow_K$.

Hence, condition 3 of the previous Def. requires that $\longrightarrow_H \subseteq \longrightarrow_S$.

EXAMPLE: Let Q be a queue; let p1 and p2 be such that

```
          Q.enq(a)    Q.enq(b)            Q.deq -> b
p1  --|---------|---|---------|-----------|-----------|------>
p2  --------|---------|-----------|------------|------------->
              Q.enq(c)           Q.deq -> a
```

This corresponds to the history

inv[Q.enq(a) by p1] inv[Q.enq(c) by p2] res[Q.enq(a) to p1] inv[Q.enq(b) by p1] res[Q.enq(c) by p2] res[Q.enq(b) by p1] inv[Q.deq() by p2] inv[Q.deq() by p2] res[Q.deq(a) to p2] res[Q.deq(b) to p1]

It can be linearized as [Q.enq(a)() by p1] [Q.enq(b)() by p1] [Q.enq(c)() by p2] [Q.deq()(a) to p2] [Q.deq()(b) to p1]

Linearizability (cont.'d)

Now consider

```
          Q.enq(a)      Q.enq(b)                 Q.deq -> a
p1 --|---------|---|---------|-----------|-----------|------>
p2 --------|---------|------------|------------|------------>
              Q.enq(c)              Q.deq -> c
```

The corresponding history can still be linearized as

[Q.enq(c)() by p2] [Q.enq(a)() by p1] [Q.enq(b)() by p1] [Q.deq()(c) to p2] [Q.deq()(a) to p1]

By contrast, the following are not linearizable histories:

```
          Q.enq(a)      Q.enq(b)                 Q.deq -> a
p1 --|---------|---|---------|-----------|-----------|------>
p2 --------|---------|------------|------------|------------>
              Q.enq(c)              Q.deq -> a
```

```
          Q.enq(a)      Q.enq(b)                 Q.deq -> b
p1 --|---------|---|---------|-----------|-----------|------>
p2 --------|---------|------------|------------|------------>
              Q.enq(c)              Q.deq -> c
```

Thm (compositionality): $\widehat{H}$ is linearizable if $\widehat{H}|_X$ is linearizable, for all X involved in H

Proof:

For all X, let $\hat{S}_X$ be a linearization of $\widehat{H}|_X$

→ $\hat{S}_X$ defines a total order on the operations on X (call it $\longrightarrow_X$)

Let $\longrightarrow$ denote $\longrightarrow_H \cup \bigcup_{X \text{ in } H} \longrightarrow_X$ *(recall that a relation is a set of pairs, so here you take the union of all pairs of $\longrightarrow_H$ and of all $\longrightarrow_X$)*

We now show that $\longrightarrow$ is acyclic.

1. It cannot have cycles with 1 edge (i.e., self loops): indeed, if op $\longrightarrow$ op, this would mean that res(op) < inv (op)
2. It cannot have cycles with 2 edges: by contr., assume that op $\longrightarrow$ op' $\longrightarrow$ op
   - both arrows cannot be $\longrightarrow_H$ nor $\longrightarrow_X$ (for some X), otw. such relations were cyclic
   - it cannot be that one is $\longrightarrow_X$ and the other $\longrightarrow_Y$ (for some X ≠ Y), otw. op/op' would be on 2 different objects

   Hence, it must be op $\longrightarrow_X$ op' $\longrightarrow_H$ op (or vice versa)

   Then, op' $\longrightarrow_H$ op means that res(op') $<_H$ inv(op)

   Since $\hat{S}_X$ is a linearization of $\widehat{H}|_X$ and op/op' are on X, this implies res(op') $<_X$ inv(op), i.e., that op' $\longrightarrow_X$ op → $\longrightarrow_X$ would be cyclic

3. It cannot have cycles with more than 2 edges: by contr., consider a shortest cycle
   - adjacent edges cannot belong to the same order (otw. the cycle would be shortable, because of transitivity)
   - adjacent edges cannot belong to orders on different objects

   Hence, at least one $\longrightarrow_X$ exists, and it must be between two $\longrightarrow_H$, i.e.:

   $$op1 \longrightarrow_H op2 \longrightarrow_X op3 \longrightarrow_H op4$$

   is part of the shortest cycles chosen (possibly with op4=op1).

$op1 \longrightarrow_H op2$ means that $res(op1) <_H inv(op2)$

$op2 \longrightarrow_X op3$ entails that $inv(op2) <_H res(op3)$

Indeed, if not, we would have that $res(op3) <_H inv(op2)$, since $<_H$ is a total order → we would have a cycle of length 2 ⚡

$op3 \longrightarrow_H op4$ means that $res(op3) <_H inv(op4)$

By transitivity of $<_H$, we would then have that $res(op1) <_H inv(op4)$, i.e. $op1 \longrightarrow_H op4$

→ in contradiction with having chosen a shortest cycle

Every DAG admits a topological order (i.e., a total order of its nodes that respects the edges)

→ Let $\longrightarrow'$ denote a topological order for $\longrightarrow$

Let us then define a linearization of $\widehat{H}$ as follows:

$\hat{S}$ = inv(op1) res(op1) inv(op2) res(op2) … whenever op1 $\longrightarrow'$ op2 $\longrightarrow'$ …

$\hat{S}$ is clearly sequential; moreover:

1. For all X, $\hat{S}|_X = \hat{S}_X$ ($\in$ semantics(X)). Indeed:
   - $<_{\hat{S}_X} = \longrightarrow_X \subseteq \longrightarrow|_X \subseteq \longrightarrow'|_X = \longrightarrow_{\hat{S}|X} = <_{\hat{S}|X}$
   - Since $<_{\hat{S}_X}$ and $<_{\hat{S}|X}$ are total orders on the same set of events (i.e., $A|_X$), they must coincide

2. For all p, $\widehat{H}|_p$ = inv($op1_p$) res($op1_p$) inv($op2_p$) res($op2_p$)… (bec. p is sequential)
   $= \hat{S}|_p$ (bec. $op1_p \longrightarrow_H op2_p \longrightarrow_H \ldots$ and $\longrightarrow_H \subseteq \longrightarrow'$)

3. $\longrightarrow_H \subseteq \longrightarrow \subseteq \longrightarrow' = \longrightarrow_S$

Alternatives to Atomicity (1)

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Sequential consistency

Let us define op $\longrightarrow_{proc}$ op' to hold whenever there exists a process p that issues both operations, with res[op] happening before inv[op'].

Def.: a complete history $\widehat{H}$ is **sequentially consistent** if there exists a sequential history $\hat{S}$ s.t.

1. $\forall X . \hat{S}|_X \in$ semantics(X) *(like linearizability)*
2. $\forall p . \widehat{H}|_p = \hat{S}|_p$ *(like linearizability)*
3. $\longrightarrow_{proc} \subseteq \longrightarrow_S$ *(in place of $\longrightarrow_H \subseteq \longrightarrow_S$)*

This is a more generous notion than linearizability.

EXAMPLE: Let $\widehat{H}$ be [Q.enq(a)() by p1] [Q.enq(b)() by p2] [Q.deq()(b) to p2]

→ not linearizable:
- the only possible linearization of $\widehat{H}$ is $\widehat{H}$ itself (because of cond.3)
- it violates the semantics of a queue (cond.1)

→ it is sequentially consistent, by swapping the first two actions, i.e. by considering $\hat{S}$ to be [Q.enq(b)() by p2] [Q.enq(a)() by p1] [Q.deq()(b) to p2]

Alternatives to Atomicity (1)

The problem with sequential consistency is that it is NOT compositional.

EXAMPLE

Consider the following two processes:

p1: Q.enq(a) ; Q'.enq(b') ; Q'.deq()→b'

p2: Q'.enq(a') ; Q.enq(b) ; Q.deq()→b

In isolation, both processes are sequentially consistent

However, no total order on the previous 6 operations respects the semantics of a queue:

- If p1 receives b' from Q'.deq, we have that Q'.enq(a') must arrive after Q'.enq(b')
- To respect $\longrightarrow_{proc}$, also the remaining behaviour of p2 must arrive after
- Hence, Q.enq(a) arrived before Q.enq(b) and so it is not possible for p2 to receive b from its Q.deq

Hence, we have two histories that are sequentially consistent but whose composition cannot be sequentially consistent → no compositionality!

Alternatives to Atomicity (2)

Serializability (typical notion in databases)

- We now have transactions instead of processes
- Consequently, we have also two other kinds of events: abort(t) and commit(t)
- The constraint is that, in every history, we have at most one of these events for every transaction; if the history is complete, we must have exactly one of these events for every transaction
- A sequential history is formed by committed transactions only

Def.: a complete history $\widehat{H}$ is **serializable** if there exists a sequential history $\hat{S}$ s.t.

1. $\forall X . \hat{S}|_X \in$ semantics(X) *(like linearizability)*
2. $S = \{e \in H : e \in t \in \text{committedTrans}(\widehat{H})\}$
3. $\longrightarrow_{trans} \subseteq \longrightarrow_S$ *(where $\longrightarrow_{trans}$ is defined like $\longrightarrow_{proc}$ in seq. cons.)*

Again, this is a more generous notion than linearizability, but it is not compositional

→ consider the previous two examples, where instead of processes, you have transactions