Rafael Fernández Ortiz

Exercises for Analysis of Distributed and Concurrent Systems

Semantics of concurrent programs

Exercise 1: Consider the following ABS codes:

```
Int main1 () {
  ...
  Fut<Int> y=a!p();
  Fut<Int> z=a!q();
  await z?;
  await y?;
  return 0;
}
```

```
Int main2 () {
  ...
  Fut<Int> y=this!r();
  a1!p();
  a2!p();
  Fut<Int> z = a3!q();
  Int r = z.get;
  await y?;
  return r;
}
```

```
Int main3 () {
  while (*) {
    Fut<Int> y = a!q();
    await y?;
    a!s();
  }
  return 0;
}
```

```
Int p () {
  a!r();
  return 0;
}
```

```
Int q () {
  y = a!r();
  awayt y?;
  return 0;
}
```

```
Int r () {
  return 9;
}
```

```
Int s () {
  return 6;
}
```

We have three methods `m1`, `m2`, `m3` and some internal methods (`p`, `q`, `r`) invoked from them. To clarify the code, variables `a`, `a1`, `a2` representing three different instances of ABS objects, created as COG's, that are available from all methods. Write **one possible execution** for the three $m*$ methods (`m1`, `m2` and `m3`).

main 1

1 (u)

Active: main1

Queque:

only main 1 is executing

a

Active:

Queque:

a_1

Active:

Queque:

a_2

Active:

Queque:

2 (u)

Active: main1

Queque:

main 1 invokes a!p

a

Active: p

Queque:

a_1

Active:

Queque:

a_2

Active:

Queque:

3 (u)

Active: main1

Queque:

main 1 invokes a!q

a

Active: p

Queque: q

a_1

Active:

Queque:

a_2

Active:

Queque:

4 (u)

Active: main1

Queque:

main1 awaits for z

so, q is active

a

Active: q

Queque: p

a_1

Active:

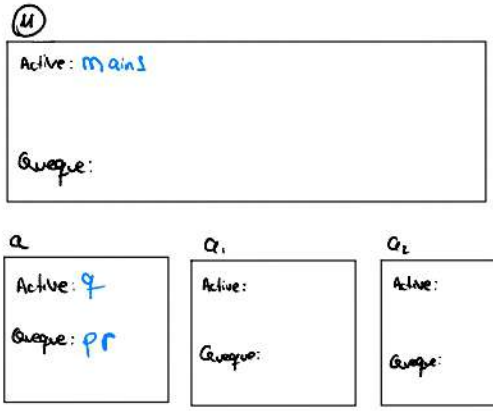
Queque:

a_2

Active:

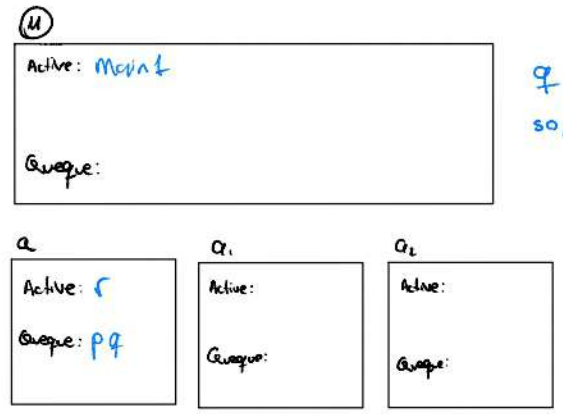
Queque:

(5)



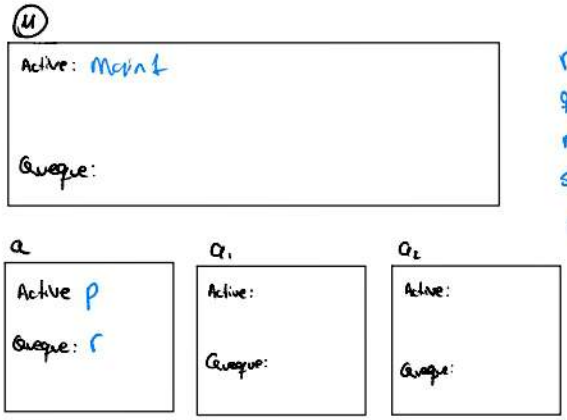
q invokes a!r

(6)



q awaits y,
so, r is active

(7)



r is finished, so
q is finished.
main awaits y,
so p is active in a.
p invokes r

(8)

p is finished

(9) (u)

Active:

Queque:

main 1 is finished.
a select a task, so
r is active.

a

Active: r

Queque:

a_1

Active:

Queque:

a_2

Active:

Queque:

(10) (u)

Active:

Queque:

r finishes.

a

Active:

Queque:

a_1

Active:

Queque:

a_2

Active:

Queque:

Main 2

(11) (u)

Active: main2

Queque:

main2 is active.

a

Active:

Queque:

a_1

Active:

Queque:

a_2

Active:

Queque:

(2)

(u)

Active: main 2

Queue: r

a

Active:

Queue:

a_1

Active:

Queue:

a_2

Active:

Queue:

main2 involves r

(3)

(u)

Active: main2

Queue: r

a

Active:

Queue:

a_1

Active: p

Queue:

a_2

Active: p

Queue:

main2 involves $a_1!p$

and next, main2 involves $a_2!p$

(4)

(u)

Active: main2

Queue: r

a

Active: r

Queue: r

a_1

Active: p

Queue:

a_2

Active: p

Queue:

p is active in a_1, so p involves $a!r$

p is active in a_2, so p involve $a!r$

(5)

(u)

Active: main2

Queue: r

a

Active: r

Queue:

a_1

Active:

Queue:

a_2

Active: p

Queue:

r finishes in a, so p finishes in a_1

r is active in a.

(6)

(u)

Active: main2

Queque: r

a

Active:

Queque:

a_1

Active:

Queque:

a_2

Active:

Queque:

r is finished in a_1, so p is finished in a_2

(7)

(u)

Active: main2

Queque: r

a

Active:

Queque:

a_1

Active:

Queque:

a_2

Active:

Queque:

main 2 involves a_3! q (who is a_3?)

(8)

(u)

Active: main2

Queque: r

a

Active:

Queque:

a_1

Active:

Queque:

a_2

Active:

Queque:

main 2 stop itself until p in a_3 finishes.
(maybe it finishes soon or not...)

(9)

(u)

Active: r

Queque: main2

a

Active:

Queque:

a_1

Active:

Queque:

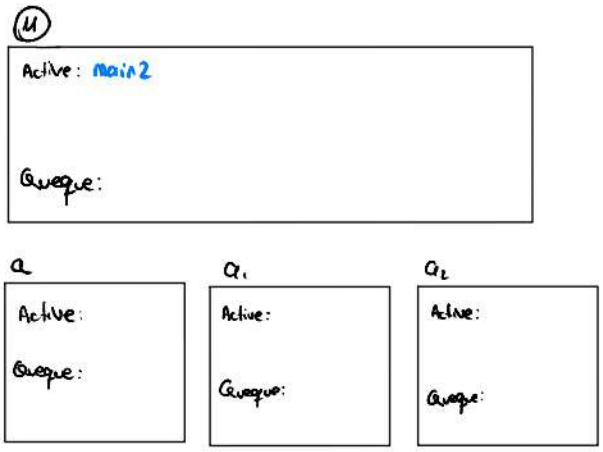
a_2

Active:

Queque:

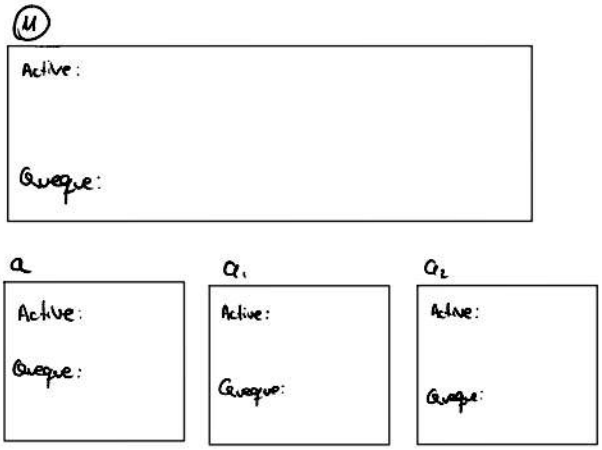
we can assume, that a_3! p finishes.
main 2 awaits y, so r is active

(10)



r finishes, so main2 is active

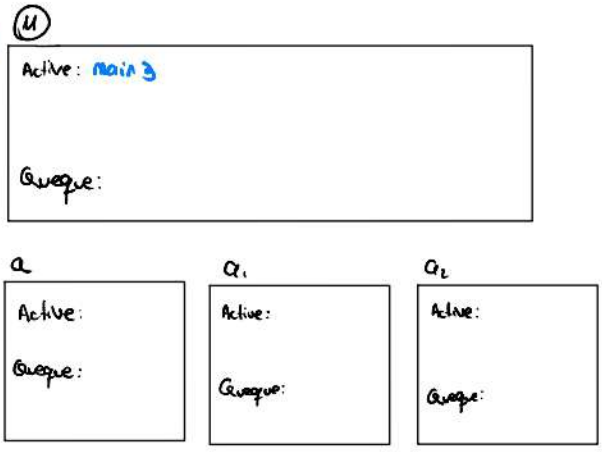
(11)



main2 finishes.

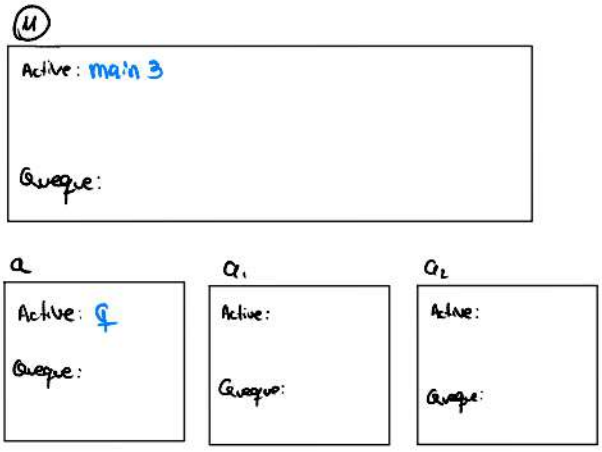
main 3

(1)



main 3 is active

(2)



main 3 invokes a!q

③ (u)

Active: main3

Queque:

a

Active: q

Queque: r

a_1

Active:

Queque:

a_2

Active:

Queque:

main3 awaits y, so q is active.

q invokes a!r.

④ (u)

Active: main3

Queque:

a

Active: r

Queque: q

a_1

Active:

Queque:

a_2

Active:

Queque:

q awaits y, so r is active

⑤ (u)

Active: main3

Queque:

a

Active: q

Queque:

a_1

Active:

Queque:

a_2

Active:

Queque:

r finishes, so q is active.

⑥ (u)

Active: main3

Queque:

a

Active: s

Queque:

a_1

Active:

Queque:

a_2

Active:

Queque:

q finishes.

main3 invokes a!s

(7) (u)

Active: main 3

Queque:

a

Active: s

Queque: s ... s

a_1

Active:

Queque:

a_2

Active:

Queque:

(...) while loop finishes.

(8) (u)

Active:

Queque:

a

Active:

Queque: s ... s

a_1

Active:

Queque:

a_2

Active:

Queque:

main 3 finishes.

s finishes in a.

(9) (u)

Active:

Queque:

a

Active: s

Queque: s ... s

a_1

Active:

Queque:

a_2

Active:

Queque:

a select a task, so s is active

•
•
•

(10) (u)

Active:

Queque:

a

Active: s

Queque:

a_1

Active:

Queque:

a_2

Active:

Queque:

s finishes.

a select a task, so a is active.

(11) (u)

s finishes.