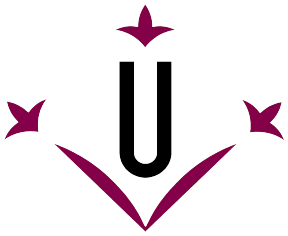

Universitat de Lleida
Escola Politècnica Superior

Sistemes Concurrents i Paral·lels

Problema 1: Introducció a la concurrència

Sergi Puigpinós Palau
Jordi Rafael Lazo Florensa

3 de novembre de 2020

Índex

Índex de figures

1 Problema 1

1.1 a)

	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11
M1	X	NO	NO	SI	SI	SI	SI	SI	SI	SI	SI
M2	X	X	SI	NO	NO	NO	NO	SI	SI	SI	SI
M3	X	X	X	SI	NO	NO	NO	NO	SI	SI	SI
M4	X	X	X	X	SI	SI	SI	SI	SI	NO	SI
M5	X	X	X	X	X	SI	SI	SI	SI	NO	SI
M6	X	X	X	X	X	X	SI	SI	SI	SI	NO
M7	X	X	X	X	X	X	X	SI	NO	SI	NO
M8	X	X	X	X	X	X	X	X	NO	SI	SI
M9	X	X	X	X	X	X	X	X	X	SI	NO
M10	X	X	X	X	X	X	X	X	X	X	NO
M11	X	X	X	X	X	X	X	X	X	X	X

Figura 1: Taula de concurrència

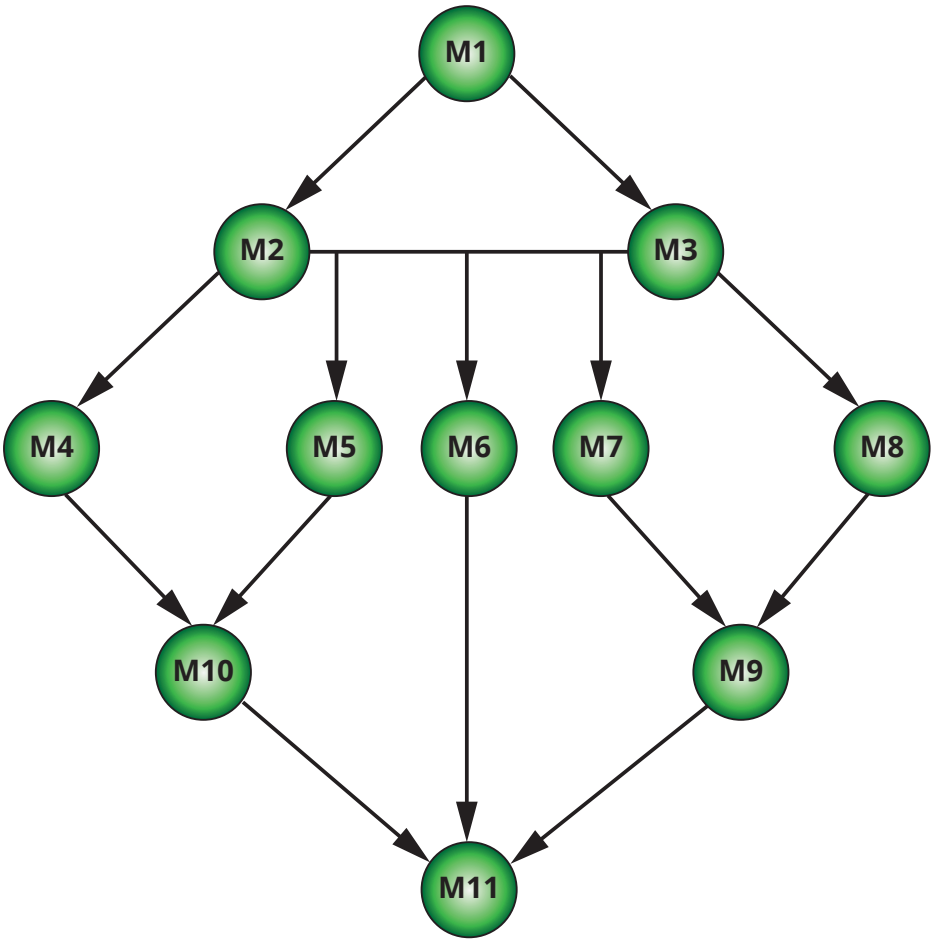

Figura 2: Graf de precedència

1.2 b)

El graf de precedència no es pot implementar mitjançant el constructor *Cobegin/Coend* ja que no suporta aquest tipus de sincronització tret que s'utilitzin semàfors.

1.3 c)

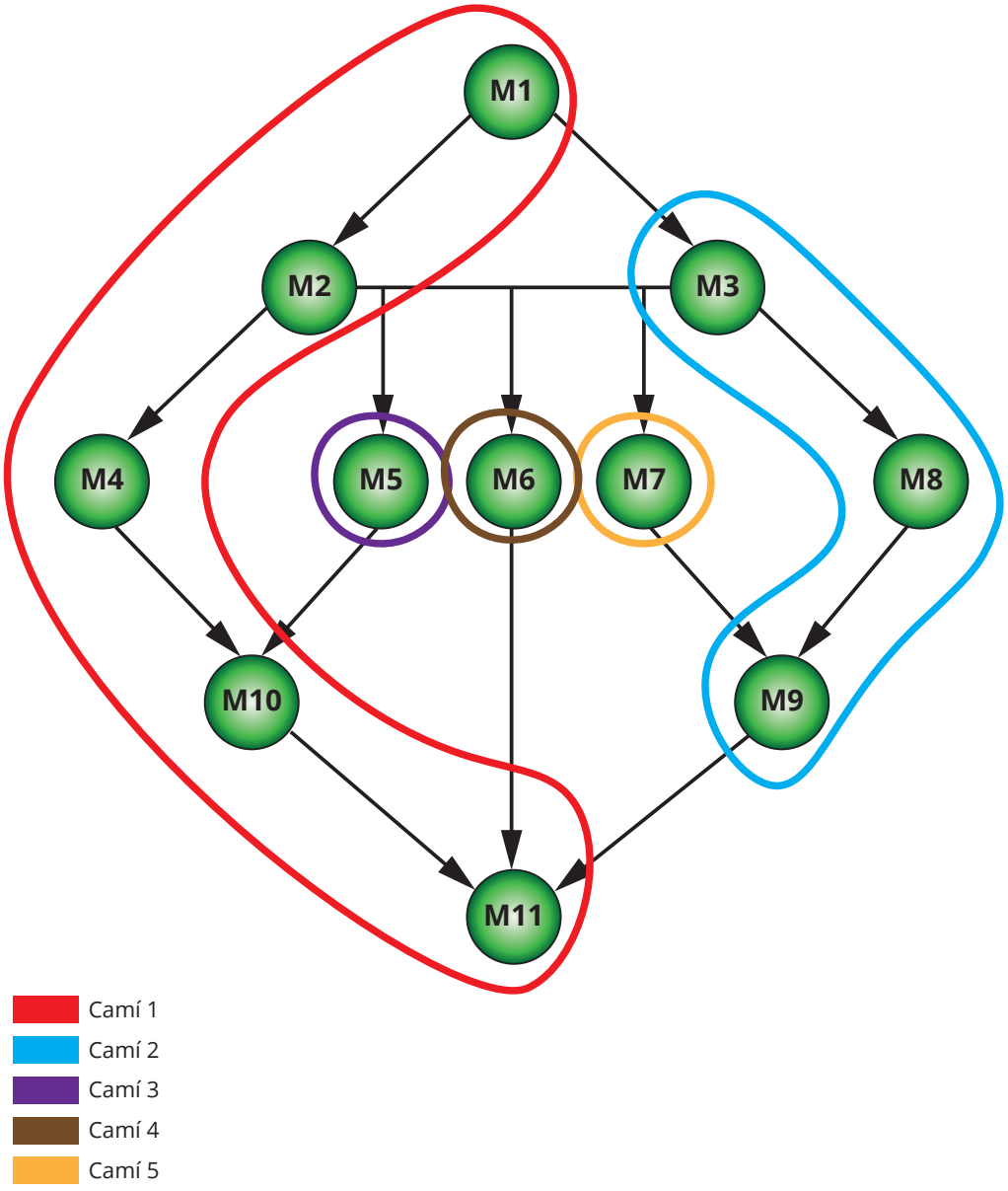

Figura 3: Implementació del graf de precedència amb *fork/join*

Implementació mitjançant el constructor *fork/join*.

```
    int counter_M10=2, counter_M11=3, counter_M9=2;
    int counter_M5=2, counter_M6=2;

    M1;
    fork branch_M3;
    M2;
    fork branch_M5;
    fork branch_M6;
    fork branch_M7;
    M4;
SYNC_M10;
    join counter_M10;
    M10;
SYNC_M11;
    join counter_M11;
    M11;
    end;
branch_M3:
    M3;
    fork dummy_M3_M5;
    fork dummy_M3_M6;
    fork dummy_M3_M7;
    M8;
SYNC_M9:
    join counter_M9;
    M9;
    goto SYNC_M11;
branch_M5:
    join counter_M5;
    M5;
    goto SYNC_M10;
branch_M6:
    join counter_M6;
    M6;
    goto SYNC_M11;
branch_M7:
    join counter_M7;
    M7;
    goto SYNC_M9;
dummy_M3_M5:
    goto branch_M5;
dummy_M3_M6:
    goto branch_M6;
dummy_M3_M7:
    goto branch_M7;
```

1.4 d)

Implementació mitjançant *Unix+C*.

```
int pid_d1,pid_d2,pid_d3;

M1;
if (fork()==0) {
    M3;
    if ((pid_d1 = fork())==0){
        exit(0);
    }
    if ((pid_d2 = fork())==0){
        exit(0);
    }
    if ((pid_d3 = fork())==0){
        exit(0);
    }
    M8;
    waitpid(pid_M7);
    M9;
    exit(0);
}
M2;
if ((pid_M5 = fork())==0){
    waitpid(pid_d1);
    M5;
    exit(0);
}
if ((pid_M6 = fork())==0){
    waitpid(pid_d2);
    M6;
    exit(0);
}
if ((pid_M7 = fork())==0){
    waitpid(pid_d3);
    M7;
    exit(0);
}
M4;
waitpid(pid_M5);
M10;
waitpid(pid_M6);
waitpid(pid_M9);
M11;
exit(0);
```

1.5 e)

1.5.1 i)

Ja que la execució es seqüencial i només es té 1 processador el temps d'execució es equivalent a la suma del temps de tots el processos.

`Temps execució = 3+8+9+10+5+6+9+10+8+6+9 = 83 unitats de temps`

1.5.2 ii)

`Temps execució = 61 unitats de temps`

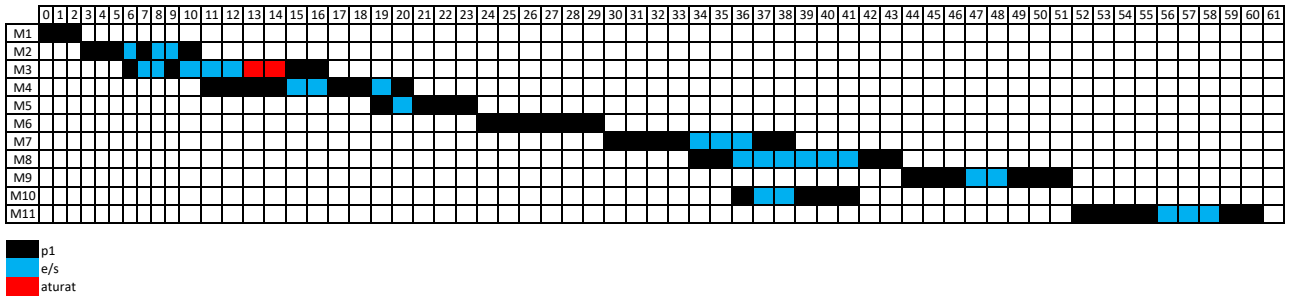

Figura 4: Diagrama de Grantt de l'apliació concurrent amb 1 processador

1.5.3 iii)

Ja que la execució serà seqüencial no importa el número de nuclis que el temps d'execució serà la suma del temps de tots el processos.

`Temps execució = 3+8+9+10+5+6+9+10+8+6+9 = 83 unitats de temps`

1.5.4 iv)

`Temps execució = 47 unitats de temps`

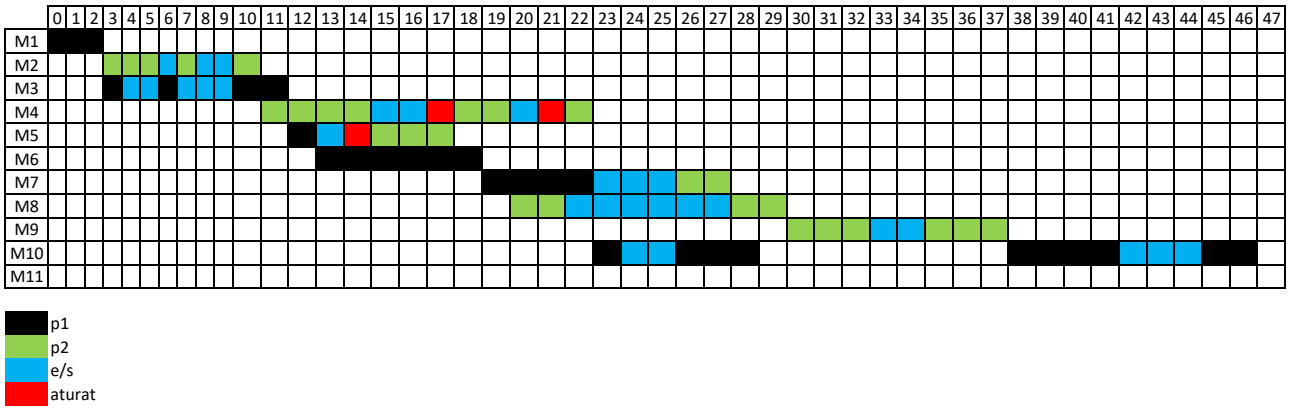

Figura 5: Diagrama de Grantt de l'apliació concurrent amb 2 processador

1.5.5 v)

```
Temps execució = 40 unitats de temps
```

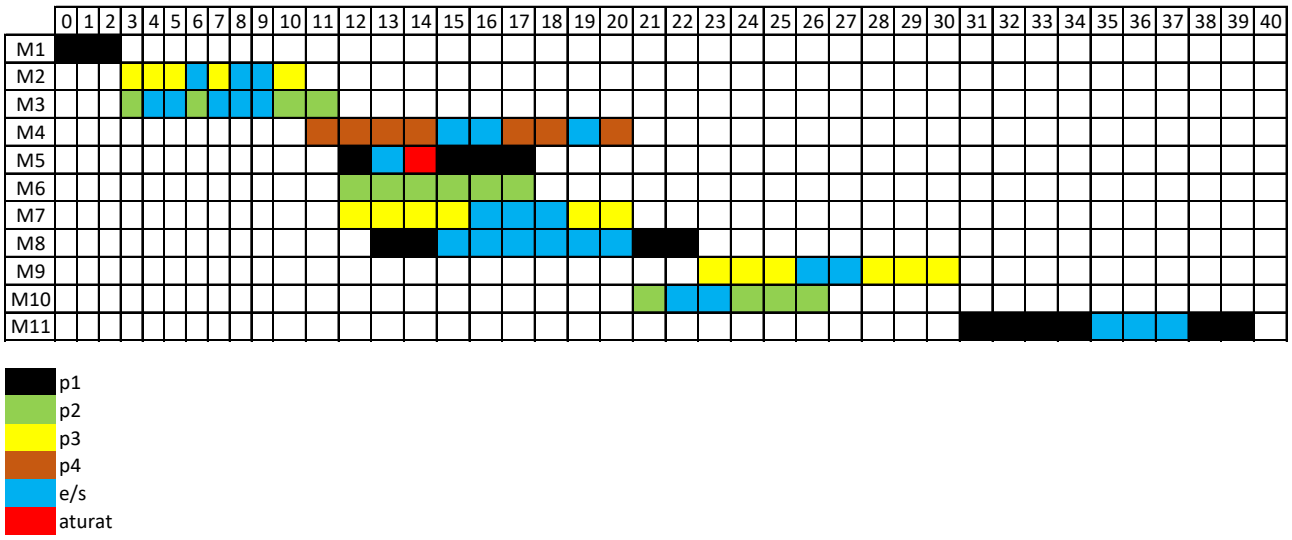

Figura 6: Diagrama de Grantt de l'apliació concurrent amb 4 processador