TPs UPPAAL
Corrigés et questions bonus

Alice & Bob

Exclusion mutuelle, Peterson, Synchronisation, Temporisé

https://www.ensta-bretagne.fr/teodorov/cours/TD2_SujetTD.pdf

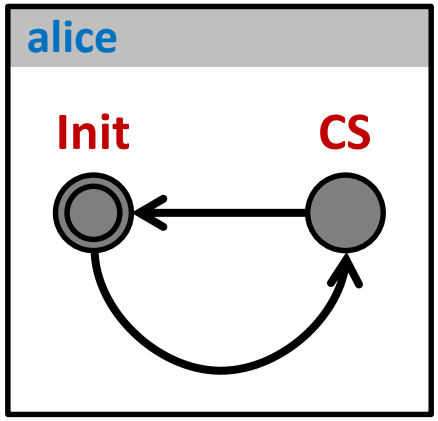
Téléchargement et installation

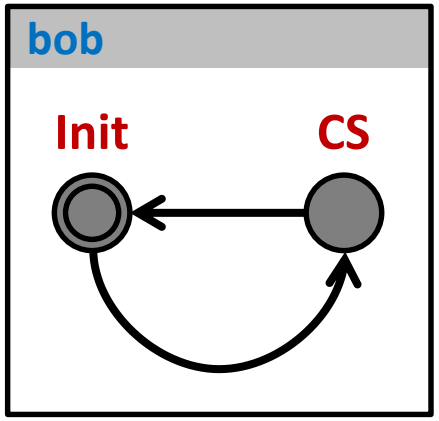
- https://www.uppaal.org/dowloads/
- **Version 4.0** (à sélectionner explicitement)
- Décompresser.
- Démarrer en lançant *uppaal.jar* (installer Java si besoins)
- Sujet : https://www.ensta-bretagne.fr/teodorov/cours/TD2_SujetTD.pdf
- Ignorer la 2nd partie (passage à niveau).

Exercice 1: Assertion

- System declarations:

```
// instance = Template();
alice = Alice();
bob = Bob();

// Composition
system alice, bob;
```



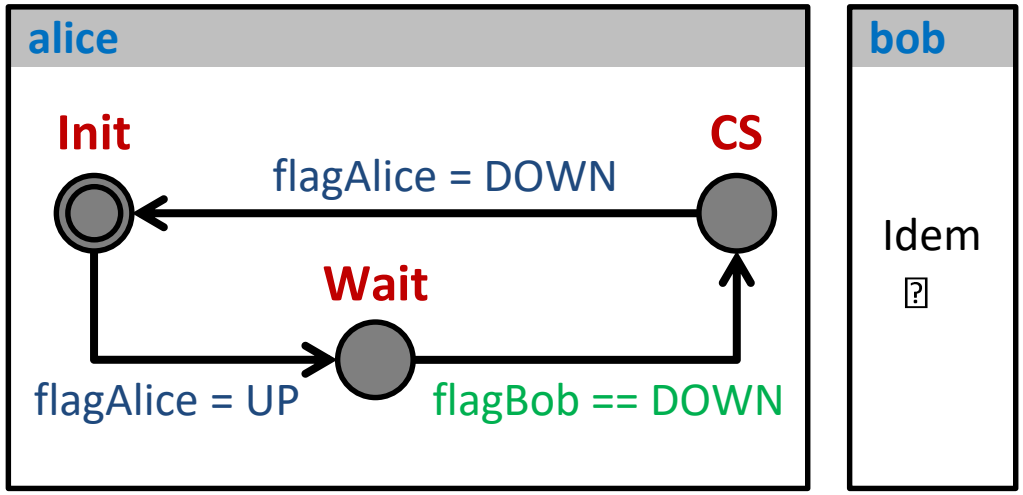


Exclusion mutuelle:
P1: A[] !(alice.CS & bob.CS) X

Exercice 2: Deadlock

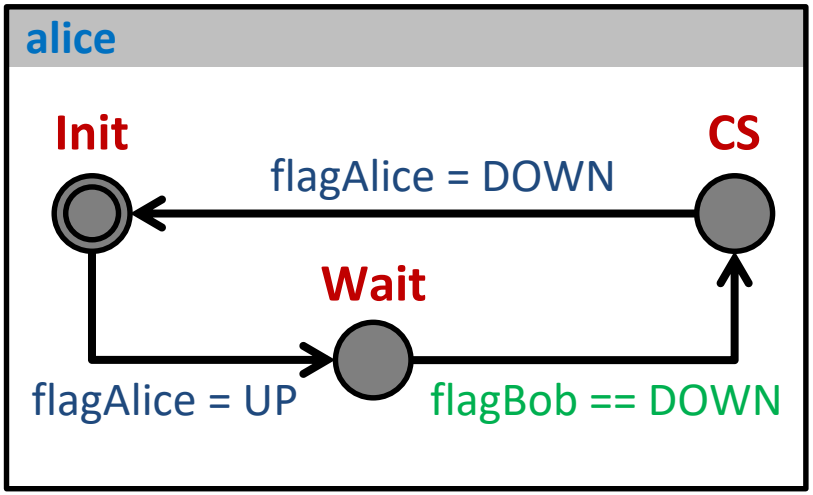
- Declarations:

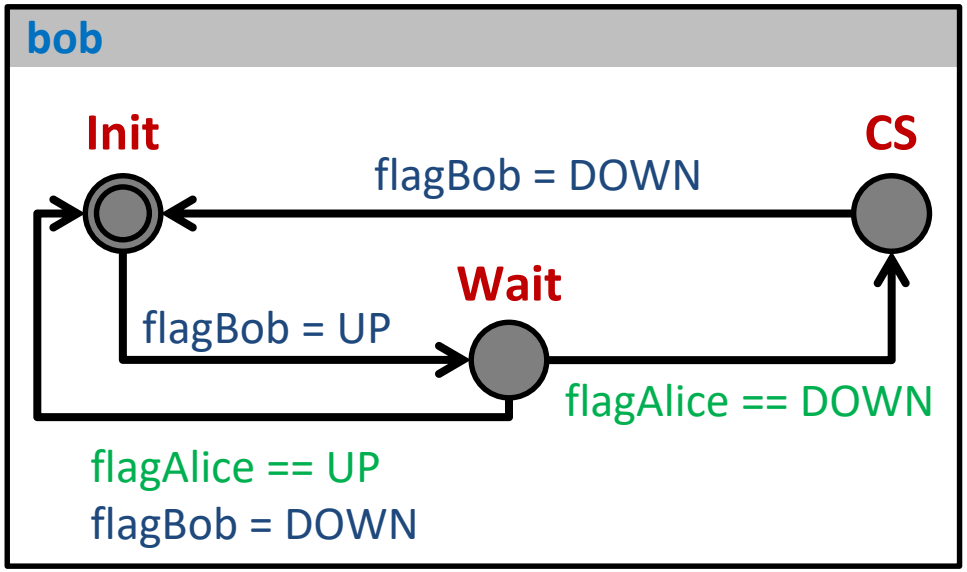
```
const bool UP = true;
const bool DOWN = false;

bool flagAlice = DOWN;
bool flatBob = DOWN;
```



P1: A[] !(alice.CS & bob.CS) **o**
P2: A[] !deadlock **X**

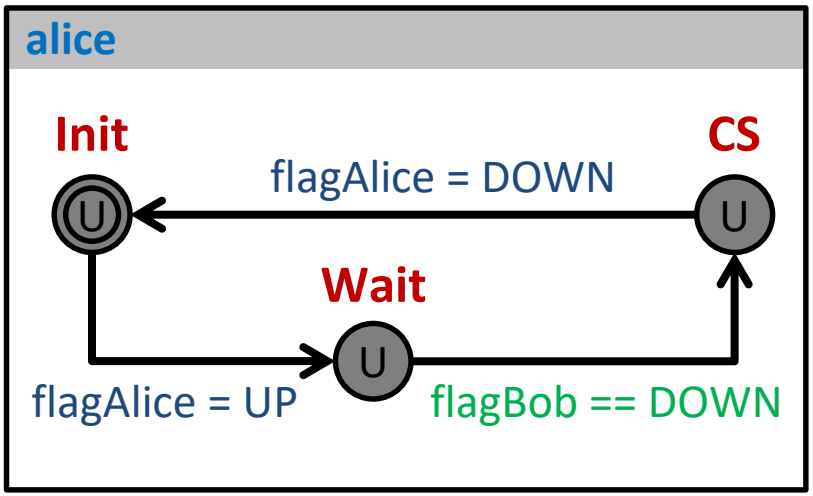
Exercice 3: Progress (1)

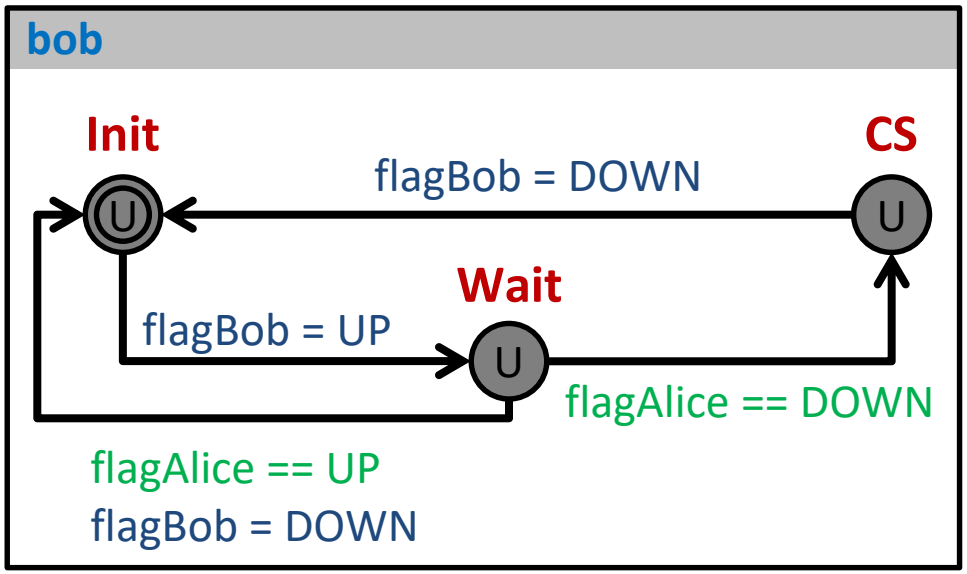




P1: A[] !(alice.CS & bob.CS) **o**
P2: A[] !deadlock **o**
P3: A<> (alice.CS | bob.CS) **X**

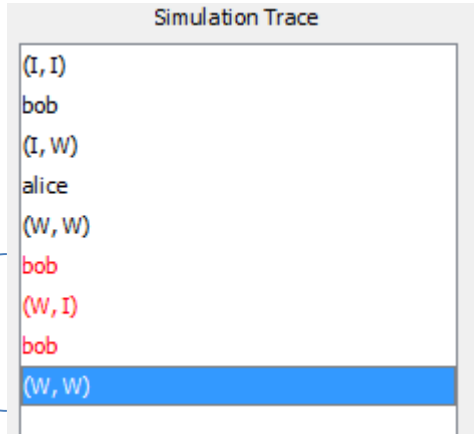
Exercice 4: Progress (2)



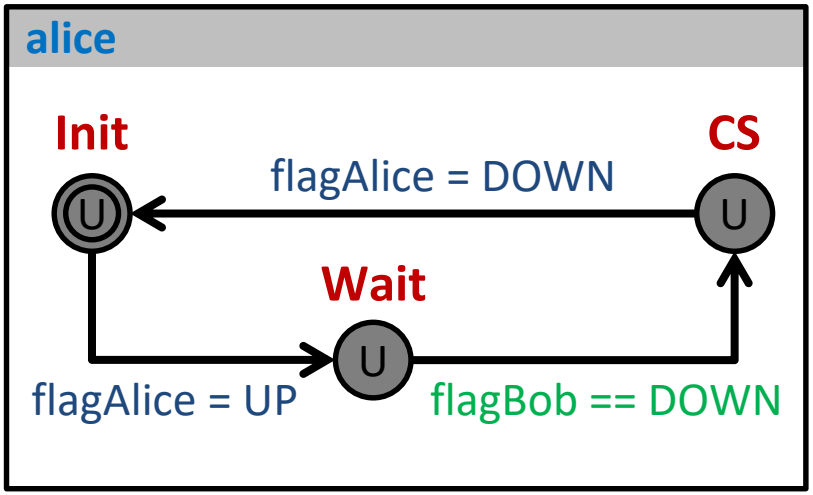


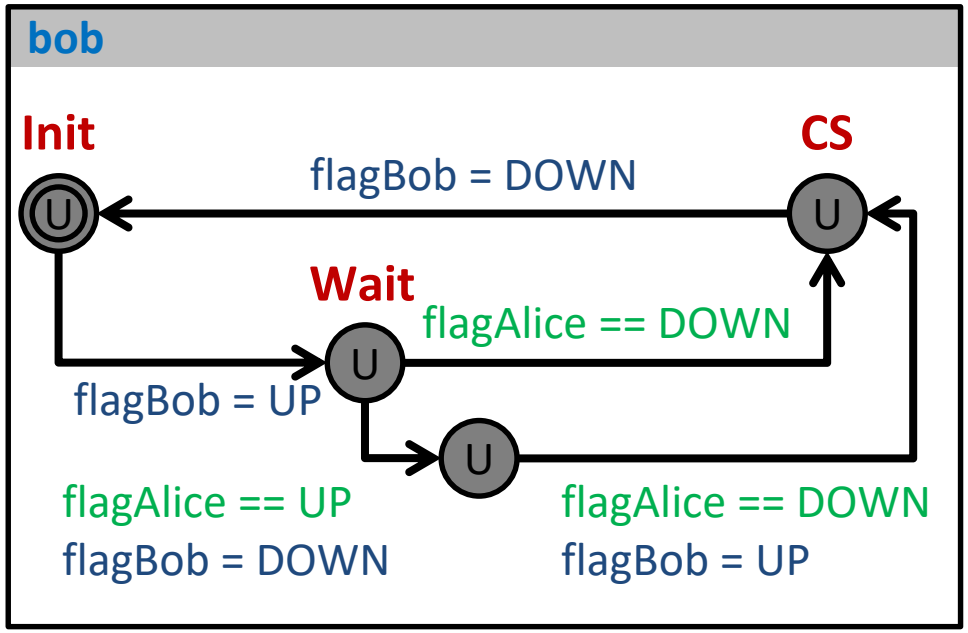
P1: A[] !(alice.CS & bob.CS) **o**
P2: A[] !deadlock **o**
P3: A<> (alice.CS | bob.CS) **X**

Cycle
« livelock »
« famine »



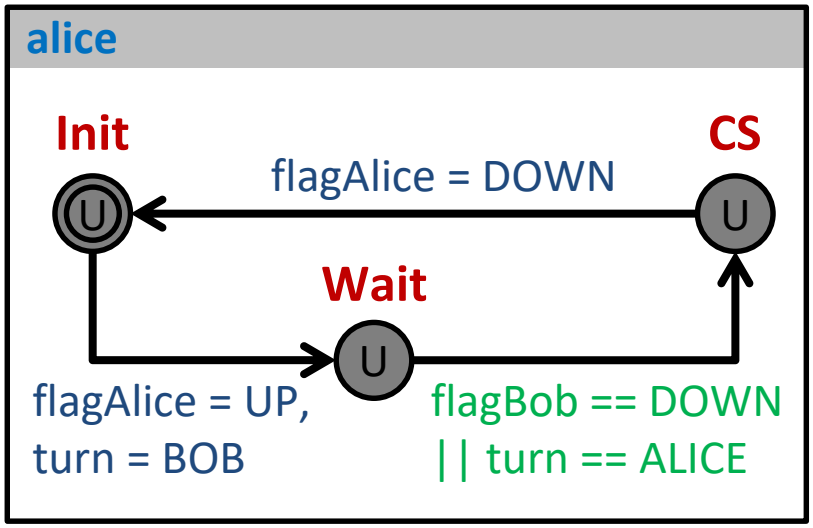
Exercice 5: Equité

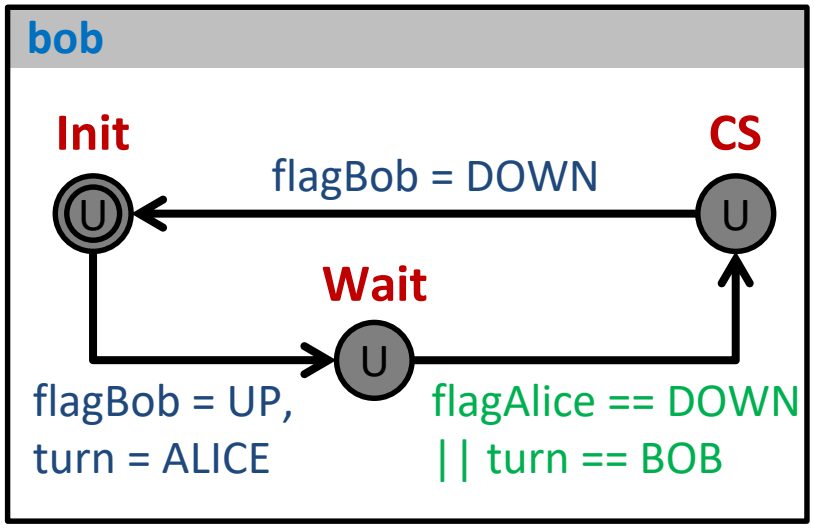




P1: A[] !(alice.CS & bob.CS) **o**
P2: A[] !deadlock **o**
P3: A<> (alice.CS | bob.CS) **o**
P4_a: flagAlice --> alice.CS **o**
P4_b: flagBob --> bob.CS **X**

Exercice 6: Peterson





- Declarations:

[…]

const bool ALICE = **true**;
const bool BOB = **false**;

bool turn = ALICE;

P1: A[] !(alice.CS & bob.CS) **o**
P2: A[] !deadlock **o**
P3: A<> (alice.CS | bob.CS) **o**
P4_a: flagAlice --> alice.CS **o**
P4_b: flagBob --> bob.CS **o**

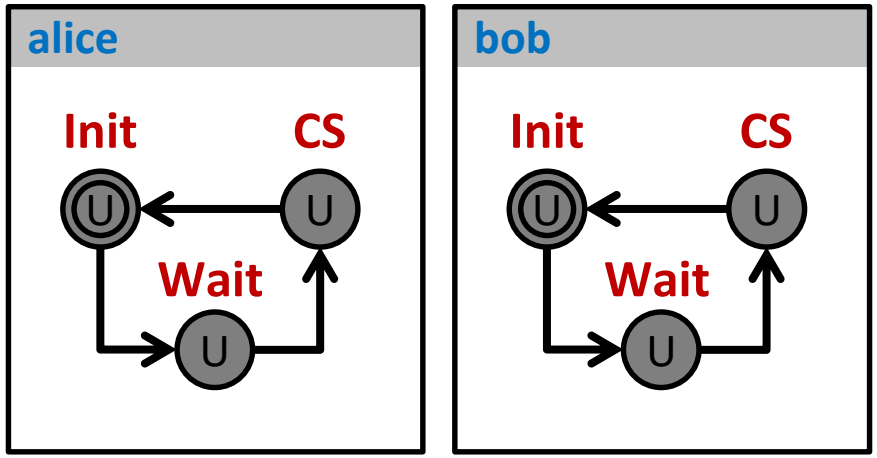
Question bonus 1: Synchronisations

- Declarations:

chan aliceCS, bobCS;

- System declarations:

alice = Alice();
Bob = Bob();
system alice, bob;



QB1: Ajoutez transitions et synchronisations (Alice: {aliceCS!, bobCS?}, inversement pour Bob) au besoins pour que les propriétés suivantes passent:

P1: A[] !(alice.CS & bob.CS)
P2: A[] !deadlock
P3: A<> (alice.CS | bob.CS)
P5_a: E[] !bob.CS
P5_b: E[] !alice.CS

Par contre, il est normal d'avoir des « famines »,
i.e. les propriétés suivantes ne seront pas vérifiées:

P4_a: alice.Wait --> alice.CS
P4_b: bob.Wait --> bob.CS

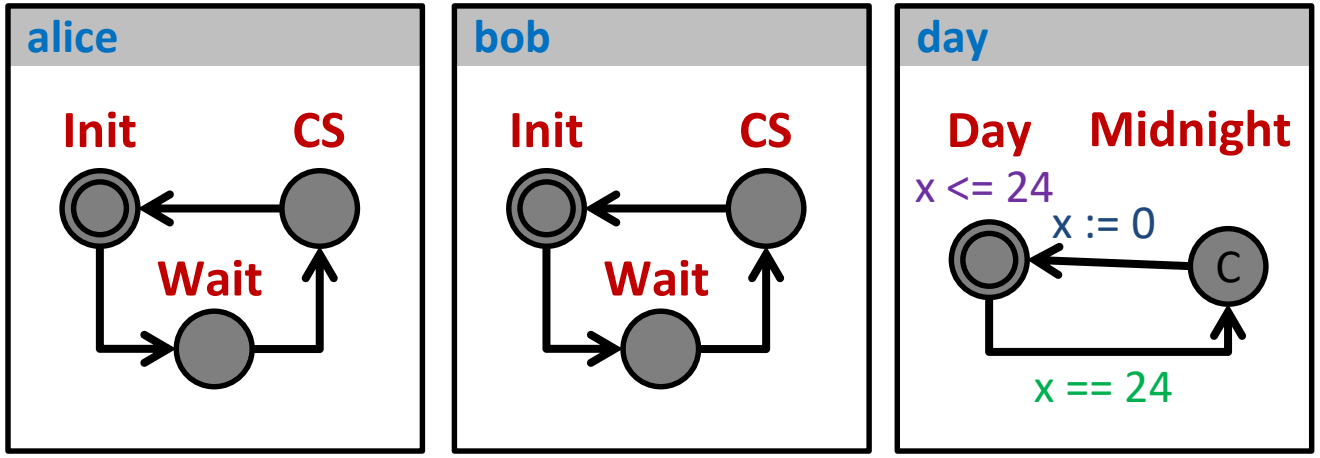
Question bonus 2: Temporisé

- Declarations:

clock x;

- System declarations:

alice = Alice();
bob = Bob();
day = day();
system alice, bob, day;



Rappels couleurs: invariant, update, guard

Alice et Bob décident de se partager la journée, le matin pour Alice, l'après-midi pour Bob. Pour des raisons obscures (sic), personne ne doit être dans le jardin à minuit!

QB2.1: Ajoutez les invariants et gardes pour Alice et Bob qui capturent ce comportement.

P1: A[] !(alice.CS & bob.CS)
P2: A[] !deadlock
P4_a: alice.Wait --> alice.CS
P4_b: bob.Wait --> bob.CS

P3: A<> (alice.CS | bob.CS)

QB2.2: Pourquoi P3 ne passe pas (où est la famine)?
QB2.3: Proposez une solution (en langage naturel).