Systèmes concurrents

Philippe Quéinnec

ENSEEIHT
Département Sciences du Numérique

25 septembre 2020

Deuxième partie

L'exclusion mutuelle

Contenu de cette partie

- Difficultés résultant d'accès concurrents à un objet partagé
- Mise en œuvre de protocoles d'isolation
  - solutions synchrones (i.e. bloquantes) : attente active
    - → difficulté du raisonnement en algorithmique concurrente
    - → aides fournies au niveau matériel
  - solutions asynchrones : gestion des processus

Plan

Trop de pain ?

♪♪♪

Vous

1. Arrivez à la maison
2. Constatez qu'il n'y a plus de pain
3. Allez à une boulangerie
4. Achetez du pain
5. Revenez à la maison
6. Rangez le pain

Votre colocataire

1. Arrive à la maison
2. Constate qu'il n'y a plus de pain
3. Va à une boulangerie
4. Achète du pain
5. Revient à la maison
6. Range le pain

Spécification

Propriétés de correction

- Sûreté : un seul pain est acheté
- Vivacité : s'il n'y a pas de pain, quelqu'un en achète

Que se passe-t-il si

- votre colocataire était arrivé après que vous soyez revenu de la boulangerie ?
- Vous étiez arrivé après que votre colocataire soit revenu de la boulangerie ?
- Votre colocataire attend que vous soyez là pour vérifier s'il y a du pain ?

$\Rightarrow$ *race condition* quand la correction dépend de l'ordonnancement des actions

Solution 1 ?

Vous (processus A)

```
A1. si (pas de pain
        && pas de note) alors
A2.   laisser une note
A3.   aller acheter du pain
A4.   enlever la note
   finsi
```

Colocataire (processus B)

```
B1. si (pas de pain)
        && pas de note) alors
B2.   laisser une note
B3.   aller acheter du pain
B4.   enlever la note
    finsi
```

$\Rightarrow$ deux pains possibles si entrelacement A1.B1.A2.B2...

Solution 2 ?

♪♪♪

Vous (processus A)

```
laisser une note A
si (pas de note B) alors
   si pas de pain alors
      aller acheter du pain
   finsi
finsi
enlever la note A
```

Colocataire (processus B)

```
laisser une note B
si (pas de note A) alors
   si pas de pain alors
      aller acheter du pain
   finsi
finsi
enlever la note B
```

$\Rightarrow$ zéro pain possible

Solution 3 ?

♪♪♪

Vous (processus A)

```
laisser une note A
tant que note B faire
   rien
fintq
si pas de pain alors
     aller acheter du pain
finsi
enlever la note A
```

Colocataire (processus B)

```
laisser une note B
si (pas de note A) alors
   si pas de pain alors
      aller acheter du pain
   finsi
finsi
enlever la note B
```

Pas satisfaisant

Hypothèse de progression / Solution peu évidente / Asymétrique / Attente active

Interférence et isolation

```
(1) x := lire_compte(2);          ||  (a) v := lire_compte(1);
(2) y := lire_compte(1);          ||  (b) v := v - 100;
(3) y := y + x;                   ||  (c) ecrire_compte(1, v);
(4) ecrire_compte(1, y);          ||
```

- Le compte 1 est partagé par les deux traitements ;
- les variables `x`, `y` et `v` sont locales à chacun des traitements ;
- les traitements s'exécutent en parallèle, et leurs actions peuvent être entrelacées.

(1) (2) (3) (4) (a) (b) (c) est une exécution possible, cohérente.
(1) (a) (b) (c) (2) (3) (4) " " " " "
(1) (2) (a) (3) (b) (4) (c) est une exécution possible, incohérente.

Section critique

Définition

Les séquences $S_1 = (1);(2);(3);(4)$ et $S_2 = (a);(b);(c)$ sont des sections critiques, qui doivent chacune être exécutées de manière atomique (indivisible) :

- le résultat de l'exécution concurrente de S_1 et S_2 doit être le même que celui de l'une des exécutions séquentielles $S_1; S_2$ ou $S_2; S_1$.
- cette équivalence peut être atteinte en contrôlant directement l'ordre d'exécution de S_1 et S_2 (exclusion mutuelle), ou en contrôlant les effets de S_1 et S_2 (contrôle de concurrence).

« Y a-t-il du pain ? Si non alors acheter du pain ; ranger le pain. »

Accès concurrents

Exécution concurrente ♪♪♪

```
init x = 0;  // partagé
⟨ a := x; x := a + 1 ⟩ ‖ ⟨ b := x; x := b - 1 ⟩
```
⇒ x = -1, 0 ou 1

Modification concurrente ♪♪♪

⟨ x := 0x|00|01| ⟩ ‖ ⟨ x := 0x|02|00| ⟩

⇒ x = 0x0001 ou 0x0200 ou 0x0201 ou 0x0000 ou 1234 !

Cohérence mémoire ♪♪♪

```
init x = 0 ∧ y = 0
⟨ x := 1; y := 2 ⟩ ‖ ⟨ printf("%d %d",y,x); ⟩
```
⇒ affiche 0 0 ou 2 1 ou 0 1 ou 2 0 !

L'exclusion mutuelle

Exécution en exclusion mutuelle d'un ensemble de sections critiques

- ensemble d'activités concurrentes A_i
- variables partagées par toutes les activités
  variables privées (locales) à chaque activité
- structure des activités

```
cycle
      entrée   section critique   sortie
      ⋮
fincycle
```

- hypothèses :
  - vitesse d'exécution non nulle
  - section critique de durée finie

Propriétés du protocole d'accès ♪♪♪

- (sûreté) à tout moment, au plus une activité est en cours d'exécution d'une section critique

$$\text{invariant } \forall i, j \in 0..N-1 : A_i.excl \wedge A_j.excl \Rightarrow i = j$$

- (progression ou vivacité globale) lorsqu'il y a (au moins) une demande, une activité qui demande à entrer sera admise

$$(\exists i \in 0..N-1 : A_i.dem) \rightsquigarrow (\exists j \in 0..N-1 : A_j.excl)$$
$$\forall i \in 0..N-1 : A_i.dem \rightsquigarrow (\exists j \in 0..N-1 : A_j.excl)$$

- (vivacité individuelle) si une activité demande à entrer, elle finira par obtenir l'accès (son attente est finie)

$$\forall i \in 0..N-1 : A_i.dem \rightsquigarrow A_i.excl$$

($p \rightsquigarrow q$: à tout moment, si p est vrai, alors q sera vrai ultérieurement)