Rapport Outils Preuve et Vérification

Problème de l'exclusion mutuelle entre processus

Korlan Colas Lisa Aubry

25 octobre 2015

Table des matières

Chapitre 1

Situations d'interblocage

1.1 Algorithme 2.1

1.1.1 Description générale

L'algorithme présenté à la figure 2.1 décrit le comportement de deux processus, pour lesquels l'accès à la section critique est régi par l'existence d'une seule variable partagée. En l'occurence, cette variable nommée TURN observe un comportement booléen, en cela que sa valeur (entière) est soit un, soit deux. Afin d'apporter d'avantage de lisibilité au raisonnement qui va suivre, nous allons réécrire en partie l'algorithme proposé, en associant à chacune des instructions un numéro de ligne.

```
1   TURN := 1;
2   wait until TURN = 2;
3   Critical Section
```

1.1.2 Représentation du problème

En vue d'une modélisation de cet algorithme avec l'outil Véritaf, nous définissons deux sous-systèmes. L'un sera dédié à la représentation des processus et de leur avancée dans l'exécution du code, tandis que le second sous-système indiquera la valeur de TURN.

Le sous-système Process

$$|S| = 5$$

$$S = \{0, 1, 2, 3, 4\}$$

$$S_0 = \{0\}$$

$$AP = \{beginning, affectation, condition, critical, end\}$$

$$\lambda = \begin{cases} 0 \rightarrow & \text{beginning} \\ 1 \rightarrow & \text{affectation} \\ 2 \rightarrow & \text{condition} \\ 3 \rightarrow & \text{critical} \\ 4 \rightarrow & \text{end} \end{cases}$$

$$Process = < S, S_0, \rightarrow, \lambda, AP >$$

Ici, pour les états 1, 2 et 3, il y a correspondance entre le numéro de l'état et le numéro de la ligne que le processus est en train d'exécuter. On ajoute en plus l'état 0 qui indique que le processus va entrer dans la section de code (donc qu'il se trouve avant la ligne 1), et l'état 4 qui indique que le processus est sorti de manière définitive de la section de code (donc qu'il se situe après la ligne 3, sans effectuer de boucle).

Le sous-système Shared

$$|S'| = 2$$

$$S' = \{0, 1\}$$

$$S'_0 = \{0, 1\}$$

$$AP = \{one, two\}$$

$$\lambda = \begin{cases} 0 \rightarrow & \text{one} \\ 1 \rightarrow & \text{two} \end{cases}$$

$$Shared = < S', S'_0, \rightarrow, \lambda, AP >$$

Comme dit précédemment, les états de ce sous-système donnent la valeur de la variable partagée, et ce de manière explicite.

1.1.3 Système

Le système correspondant à notre problème sera constitué de deux éléments P1 et P2, affiliés au sous-système Process, ainsi que d'un élément nommé TURN qui correspond au sous-système Shared. Afin que le présent rapport reste concis, les transitions définies pour chacun des sous-sytèmes sont plus amplement commentées dans le programme Véritaf.

1.1.4 Résultat

Conformément à notre définition du système complet, le non respect de la propriété d'exclusion mutuelle peut-être formalisée en logique CTL par :

$$(1) P1.critical \wedge P2.critical$$

De même, une situation d'interblocage sera formulée de la manière suivante :

$$(2) !EX(true) \wedge (P1.condition \vee P2.condition)$$

D'après nos résultats, l'exclusion mutuelle des processus au regard de la section critique est préservée dans le cas présent. En effet, le réduit qui vérifie la formule (1) est vide.
Cependant, le réduit vérifiant la formule (2) est non vide. On en conclut que cet algorithme mène à une situation d'interblocage qui survient lorsqu'un processus quitte cette partie de code (état *end*), tandis que le deuxième processus est en attente d'une modification de la valeur de TURN. Autrement dit, un processus est susceptible d'attendre indéfiniement sans que l'accès à la section critique ne lui soit jamais accordé.

1.2 Algorithme 2.2

1.2.1 Description générale

L'algorithme présenté à la figure 2.2 délaisse la variable partagée TURN au profit de deux variables partagée appelées Q1 et Q2, lesquelles vérifient aussi des valeurs booléennes. Une fois encore, voici la partie de code proposée pour le premier processus.

```
1   Q1 := true;
2   wait until not Q2;
3   Critical Section
4   Q1 := false;
```

1.2.2 Représentation du problème

Nous définissons également deux sous-systèmes sur le même modèle que précédemment, à savoir l'un pour décrire l'avancée des deux processus dans le code, et l'autre aui aura vocation de modéliser les variables partagées.

Le sous-système Process

$$|S| = 5$$

$$S = \{0, 1, 2, 3, 4\}$$

$$S_0 = \{0\}$$

$$AP = \{beginning, affectation, condition, critical, end\}$$

$$\lambda = \begin{cases} 0 \rightarrow & \text{beginning} \\ 1 \rightarrow & \text{affectation} \\ 2 \rightarrow & \text{test} \\ 3 \rightarrow & \text{critical} \\ 4 \rightarrow & \text{end} \end{cases}$$

$$Process = < S, S_0, \rightarrow, \lambda, AP >$$

Pour les états 1, 2, 3 et 4, il y a correspondance entre le numéro de l'état et le numéro de la ligne que le processus est en train d'exécuter. On ajoute en plus l'état 0 qui indique que le processus va entrer dans la section de code (donc qu'il se situe avant la ligne 1).

Le sous-système Shared

$$|S'| = 2$$

$$S' = \{0, 1\}$$

$$S'_0 = \{0, 1\}$$

$$AP = \{istrue, isfalse\}$$

$$\lambda = \begin{cases} 0 \rightarrow & \text{istrue} \\ 1 \rightarrow & \text{isfalse} \end{cases}$$

$$Shared = < S', S'_0, \rightarrow, \lambda, AP >$$

De manière analogue, ce sous-sytème et ses états permettront d'indiquer quelle est la valeur de Qi.

1.2.3 Système

On définit ensuite P1 et P2, régis par le sous-système Process, ainsi que Q1 et Q2 associés au sous-système Shared.

1.2.4 Résultat

Conformément à notre définition du système complet, le non respect de la propriété d'exclusion mutuelle peut-être formalisée en logique CTL par :

$$(1) P1.critical \wedge P2.critical$$

De même, une situation d'interblocage sera formulée de la manière suivante :

$$(2) !EX(true) \wedge (P1.test \vee P2.test)$$

Tout comme pour l'algorithme précédent, les accès concurrents concurrents à la section critique sont gérés convenablement par cet algorithme. Cela se confirme par le fait que le réduit vérifiant la formule (1) est vide.
Une fois encore, le réduit vérifiant la formule (2) est non vide. Ici, et contrairement aux cas précédents, la situation d'interblocage a lieu alors que les deux processus tentent d'accéder à la section critique. En effet, on peut constater que dans ce cas, $Q1 = Q2 = true$.