Mécanismes de Synchronisation de Haut Niveau

Modules de contrôle.

Le principe du *module de contrôle* (mdc) [Rob77], basé sur la séparation du contrôle et du traitement, consiste à regrouper: d'une part, dans un module (module de traitement) un ensemble de procédures et d'autre part, dans un module de contrôle les règles de synchronisation correspondantes. Bien entendu, la possibilité de rattacher un module de traitement dès sa création au module de traitement associé est disponible.

Le module de contrôle qui ne connaît que les noms des procédures à synchroniser (interface) sera activé à chaque changement d'état des composants du module de traitement. Ces changements d'état sont associés aux événements ayant trait aux requêtes, autorisations, et terminaisons des exécutions des procédures formant le module de traitement (mdt).

Le langage de spécification du contrôle des accès aux objets partagés associe à chaque procédure manipulée par le module de contrôle, une condition d'executabilité et une file d'attente dans laquelle sera bloqué le processus appelant ne satisfaisant pas la condition. Pour chaque procédure, on dispose de cinq compteurs qui mémorisent l'historique des accès au module. Initialement ces compteurs sont nuls et croissent de façon monotone; la file d'attente étant vide. Il est possible de modifier la restriction des valeurs initiales des compteurs au moyen d'une instruction appropriée.

Les compteurs qui décrivent les règles de synchronisation sont les suivants:

req(p) : nombre total de requêtes d'exécution d'une procédure p depuis l'initialisation du module de contrôle.
aut(p) : nombre total d'autorisations données pour l'exécution de p depuis l'initialisation.
term(p): nombre total de terminaisons d'exécution de p depuis l'initialisation.
act(p) : nombre d'exécutions en cours de p à un instant donnée (p étant réentrante):

act(p) = aut(p) - term(p)

att(p) : nombre de requêtes enregistrées mais non autorisées

att(p) = req(p) - aut(p)

Au niveau du contrôle des accès à un objet partagé, ces différents compteurs sont associés à trois événements caractéristiques suivants:
requête(p)i : ième requête de la procédure P;
autorisation(p)i : ième invocation de P déclenche son exécution;
terminaison(p)i : ième invocation de P termine son exécution;

Puisque les événements sont sérialisés comme suit:
requête(p)i --> autorisation(p)i --> terminaison(p)i et ce, pour toute procédure P et l'instant i, on en déduit: $term(p) \leq aut(p) \leq req(p)$.

Le mdc (module de contrôle) doit enregistrer les demandes (requêtes) et les terminaisons d'exécutions selon leurs arrivées. Son pouvoir de décision est exercé seulement aux autorisations. On doit donc donner pour chaque procédure (ou ensemble de procédures) sous le contrôle du mdc, un ensemble de conditions nécessaires pour que chacune de ces requêtes d'exécutions soit autorisée. Une condition est une expression booléenne formée de compteurs et de constantes.

Le réveil des processus bloqués s'effectue par le mdc suite à des événements enregistrés et ayant provoqués un changement des conditions d'autorisations d'exécution (un peu à la manière des régions critiques).

Exemple : Problème du producteur-consommateur.

spécification des règles de synchronisation:

1. Le consommateur ne peut consommer (s'exécuter) plus que ce qu'a produit le producteur d'où: aut(consommer) ≤ term(produire).
2. Le nombre de productions permises ne peut excéder de n le nombre de consommations i.e. : aut(produire) ≤ term(consommer) + n.

De 1., on en déduit la condition d'autorisation de consommer:
term(produire) - aut(consommer) > 0.
De 2., il en résulte la condition pour produire: aut(produire) < term(consommer) + n.

Les mdc n'assurent pas une exclusion mutuelle à produire et consommer, contrairement aux moniteurs, il va falloir l'imposer en le spécifiant par: act(produire) + act(consommer) = 0. Le module_de_contrôle du couple producteur-consommateur peut donc s'écrire:

```
prodcons: module_de_controle
  begin
    produire, consommer : procedure;
    faprod, facons      : queue;
    condition(produire) : aut(produire)-term(consommer)<n
                  and act(produire)+act(consommer)=0; (*)
    condition(consommer): term(produire)-aut(consommer)>0
                  and act(produire)+act(consommer)=0; (*)
  end
end module_de_controle.
```

Pour exprimer la manipulation par une procédure P d'un objet partagé tout en garantissant la cohérence et l'intégrité de ce dernier, on doit écrire l'invariant suivant: act(P) ≤ 1. De même, pour exprimer la condition d'exécution en section critique de chacune des deux procédures P et Q, on doit écrire: act(P) + act(Q) = 0 (Cf. (*) précédent)

Avantages des mdc:

L'avantage le plus important est sans doute la séparation entre les règles qui expriment la synchronisation (mdc) et les actions du traitement sur lesquelles porte cette synchronisation (mdt). Cette séparation va permettre une flexibilité dans la maintenance des mdt c'est-à-dire permettre des modifications des composants d'un mdt sans incidence sur le mdc associé. On observera également une certaine aisance au niveau de la programmation puisque les algorithmes de traitement pourront être écrits sans tenir compte du contrôle qui lui sera remis à un moment ultérieur.

Un autre avantage de cette séparation est la possibilité d'établir des preuves et des invariants. Autrement dit, déterminer qu'une spécification du contrôle est exempte des problèmes d'interblocage ou de privation.

Inconvénients

L'inconvénient que l'on peut adresser au mdc (outre, sa difficulté d'implémentation) est d'être restrictif quant à la classe de problèmes où il peut répondre aisément (en comparaison par exemple avec l'outil sémaphore).

En effet, si les règles de synchronisation ne s'expriment pas facilement en terme d'usage d'un objet partagé servant de support de communication d'information, il serait difficile de résoudre le problème sans avoir à changer de niveau d'observation. Ce dernier peut contraindre à user de moyens artificiels allant jusqu'à dénaturer ce problème. C'est notamment le cas du problème de l'allocateur de ressources (voir plus loin) où le blocage et le réveil de processus se fait à travers la mise en place de procédures artificielles à corps vide. La programmation d'un modèle de coordination de processus (par exemple le problème des rendez-vous en ADA) nécessiterait inévitablement une certaine gymnastique.