



**Examen Réparti 1**

**Durée : 2h00**

Barème indicatif. Notes de cours autorisées. Il sera tenu compte de la présentation et de la clarté dans la rédaction.

*Les trois parties doivent être traitées sur des copies séparées.*

**1 Logiques temporelles**

**Exercice 1** (2 points)

Pour chacune des formules suivantes, déterminer si c'est une formule CTL, LTL, ou ni l'une ni l'autre.

- (a)  $E(a U (Ab U (FGc)))$
- (b)  $E(a U (AFb))$
- (c)  $GF(a \rightarrow a U Gb)$
- (d)  $(GFa) \rightarrow (a U Gb)$

**Solution:**

- (a) ni l'un ni l'autre
- (b) CTL
- (c) LTL
- (d) LTL

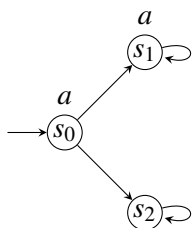
**Exercice 2** (4 points)

Déterminer si chacune des paires de formules suivantes sont équivalentes. Justifier.

- (a) (1 point)  $(\phi U \psi) U \phi$  et  $\phi U (\psi U \phi)$ , où  $\phi, \psi$  et  $\phi$  sont des formules LTL.
- (b) (1 point)  $AF(a \wedge EXa)$  et  $F(a \wedge Xa)$ .
- (c) (2 points)  $E(a U E(b U c))$  et  $E(E(a U b) U c)$ .

**Solution:**

- (a) non équivalentes : on considère les deux formules  $(p U q) U r$  et  $p U (q U r)$ . Les traces  $t$  ayant pour préfixe  $pqpqr$  sont telles que  $t \models (p U q) U r$  mais  $t \not\models p U (q U r)$ .
- (b) non équivalentes : On considère la structure de Kripke  $M$  ci-dessous :



On a bien  $M \models AF(a \wedge EXa)$  mais l'exécution  $s_0 s_2^\omega$  de trace  $\{a\}0^\omega \not\models F(a \wedge Xa)$  donc  $M \not\models F(a \wedge Xa)$ .

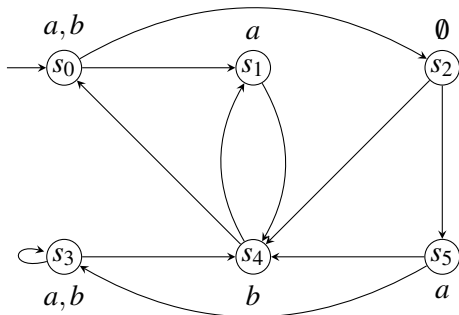
(c) non équivalentes : On considère la structure de Kripke suivante :



$M \not\models E(a \cup E(b \cup c))$ . En effet,  $S(E(b \cup c)) = \{s_3\}$ , et donc  $S(E(a \cup E(b \cup c))) = \{s_2\}$ , car  $s_2$  est bien un prédécesseur de  $s_3$  étiqueté par  $a$ , mais aucun prédécesseur de  $s_2$  n'est étiqueté par  $a$ , donc aucun autre n'était n'est dans  $S(E(a \cup E(b \cup c)))$ . Par contre,  $M \models E(E(a \cup b) \cup c)$ . On a  $S(c) = \{s_3\}$ ,  $s_1 \in S(E(a \cup b))$  car étiqueté par  $b$ ,  $s_0, s_2 \in S(E(a \cup b))$  car prédécesseurs de  $s_1$  étiquetés par  $a$ . Donc  $S(E(E(a \cup b) \cup c)) = \{s_0, s_1, s_2, s_3\}$ .

### Exercice 3 (1 points)

On considère la structure de Kripke  $M$  ci-dessous :



Déterminer si  $M \models FGa$ .

**Solution:** L'exécution  $s_0(s_1 s_4)^\omega$  de trace  $t = \{a, b\}(\{a\}\{b\})^\omega$  ne satisfait pas  $FGa$ , car quelque soit  $i \geq 0$ , il existe  $j \geq i$  tel que  $a \notin t_j$ , donc  $\{a, b\}(\{a\}\{b\})^\omega, j \not\models a$  donc  $\{a, b\}(\{a\}\{b\})^\omega, i \not\models Ga$  donc  $t, 0 \not\models FGa$ . Puisqu'il existe une trace de  $M$  ne respectant pas la spécification, on a  $M \not\models FGa$ .

**Total des points de la section : 7 points**

## 2 Systèmes temps réel critiques

### Exercice 4 : Ordonnancement global vs partitionné (4 points)

Dans cet exercice, nous étudions l'ordonnancement d'une application de vidéosurveillance composée d'un ensemble de tâches décrites ci-dessous.

- La tâche **Ta1** s'occupe du périphérique d'entrée audio et gère l'acquisition des échantillons audio. **Ta1 fournit en sortie les entrées de la tâche Ta2.**
- La tâche **Ta2** gère le périphérique de sortie audio (haut-parleurs) et présente le flux audio sur une console de supervision. **Ta2 prend en entrée les sorties de Ta1.**
- Les tâches **Tv1** et **Tv2** assurent des fonctions similaires pour le flux vidéo.

Un ingénieur souhaite valider le comportement temporel de ce jeu de tâches. On suppose, en outre que celles-ci sont définies par les paramètres ci-dessous :

Tâches	Capacité	Période
Ta1	9	20
Ta2	9	20
Tv1	20	40
Tv2	20	40

On modélise ce système sous forme de tâches synchrones à échéances implicites. On suppose un ordonnanceur préemptif à priorité fixe disposant de 100 niveaux de priorité (de 0 à 99, le niveau de priorité 99 étant le niveau de priorité le plus élevé).

La plate-forme d'exécution est constituée d'un processeur composé de deux coeurs identiques.

On souhaite modéliser le problème sous forme de tâches indépendantes en imposant des priorités telles que les contraintes de précédences soient respectées.

(a) ( $\frac{1}{2}$  point) **Priorités entre Ta1 et Ta2**

D'après la description du problème, la tâche Ta2 (respectivement Tv2) doit être activée sur terminaison de la tâche Ta1 (respectivement Tv1).

De Ta1 et Ta2 (respectivement Tv1 et Tv2), quelle tâche doit avoir la plus forte priorité afin d'assurer les contraintes de précedence exprimées ci-dessus ?

**Solution:** Comme on souhaite modéliser le problème sous forme de tâches indépendantes, pour assurer que Ta2 (Tv2) s'exécute après Ta1 (Tv1), la priorité de Ta1(Tv1) soit supérieure à celle de Ta2 (Tv2).

(b) ( $\frac{1}{2}$  point) **Approche partitionnée**

En première approche, on souhaite appliquer une approche partitionnée : les tâches Ta1 et Ta2 sont placées sur un premier coeur et les autres tâches sur le second. On utilise les propriétés sur les priorités exprimées à la question précédente.

Calculer l'ordonnancement de cette application sur l'hyper-période. Le système est-il ordonnançable ?

**Solution:** Le ppcm des 4 tâches est 40. Sur le coeur 1, Ta1 démarre à  $t=0$  et se termine à  $t=8$ , Ta2 démarre à  $t=9$  et se termine à  $t=17$ . Sur le coeur 2, Tv2 démarre à  $t=0$  et se termine à  $t=19$ , Ta2 démarre à  $t=20$  et se termine à  $t=39$ . Le système est ordonnançable.

(c) **Approche globale**

En seconde approche, on utilise un ordonnancement global préemptif à priorité fixe pour exécuter ces tâches sur les 2 coeurs. On suppose que la migration des tâches peut intervenir à tout moment.

Dans la suite, on suppose que cet ordonnancement permet d'exécuter correctement l'ensemble de tâches Ta1, Ta2, Tv1 et Tv2 tout en assurant les propriétés sur les priorités exprimées à la question 1.1 afin de rendre ces tâches indépendantes.

i. ( $\frac{1}{2}$  point) **Priorités de Ta1 et Tv1**

Démontrer qu'à l'instant  $t=0$ , Tv1 et Ta1 doivent avoir les 2 priorités les plus fortes pour que le système s'exécute correctement.

**Solution:** Ta1 et Tv1 doivent avoir les priorités les plus fortes pour commencer en premier sur les deux coeurs disponibles, car l'ordonnancement global est à priorité fixe.

ii. ( $\frac{1}{2}$  point) **Priorités entre Tv2 et Ta2 à  $t=9$**

Démontrer que l'on doit avoir  $priority(Tv2) < priority(Ta2)$  pour que le système s'exécute correctement à  $t=9$ .

**Solution:** Rappelons que les 4 tâches sont supposées indépendantes. A la date  $t=8$ ,  $Ta1$  se termine. A  $t=9$ , les tâches prêtes sont alors  $Ta2$  et  $Tv2$ ,  $Tv1$  n'étant pas terminée. Comme  $Tv2$  ne peut pas démarrer car  $Tv1$  n'est pas terminée, alors que  $Ta2$  peut démarrer, il faut que  $priority(Tv2) < priority(Ta2)$ .

iii. ( $\frac{1}{2}$  point) **Priorités entre  $Tv2$  et  $Ta2$  à  $t=20$**

Démontrer que l'on doit avoir  $priority(Tv2) > priority(Ta2)$  pour que le système s'exécute correctement à  $t=20$ .

**Solution:** Rappelons que les 4 tâches sont supposées indépendantes. A la date  $t=19$ , la tâche  $Tv1$  se termine. A  $t=20$ , les tâches prêtes sont alors  $Ta1$ ,  $Ta2$  et  $Tv2$ , sachant que 2 coeurs sont libres. Comme  $Ta1$  est plus prioritaire que  $Ta2$  et  $Tv2$  (première question), elle s'exécutera sur le coeur 1. Sur le coeur de libre, il faut exécuter  $Tv2$  pour qu'elle se termine avant  $t=40$ . Donc  $priority(Tv2) > priority(Ta2)$ .

(d) ( $\frac{1}{2}$  point) **Conclusions**

Pour cette application, quelle approche (globale ou partitionnée) recommanderiez-vous ? Pourquoi ?

**Solution:** Si on adopte l'approche globale, avec les questions précédentes, on a  $priority(Tv2) > priority(Ta2)$  et  $priority(Tv2) < priority(Ta2)$ . Ce qui est impossible. Donc l'approche globale avec priorité fixe ne peut ordonnancer ce système. Il faut utiliser l'approche partitionnée.

(e) (1 point) **Relaxe du modèle synchrone à échéances implicites**

On relâche désormais les contraintes sur les réveils synchrones et les échéances implicites des tâches. On conserve l'indépendance des tâches et l'utilisation d'un ordonnanceur préemptif à priorité fixe.

Comment peut-on modéliser les tâches  $Tv1$ ,  $Tv2$ ,  $Ta1$  et  $Ta2$  de sorte que tout en étant indépendantes, grâce à un paramétrage adapté que l'on explicitera, elles respectent par construction leurs contraintes de dépendances ?

Une attention particulière sera accordée aux explications.

**Solution:** Il faut que les tâches ne soient plus synchrones.  $Ta1$  et  $Tv1$  sont activées à la date  $t=0$ . Par contre,  $Ta2$  doit être activée à  $t=9$  et  $Tv2$  à  $t=20$ . Dès lors, nous n'avons plus besoin de  $priority(Tv2) < priority(Ta2)$  car  $Tv2$  n'est pas active à  $t=9$ .

**Exercice 5 : Ordonnement global** (3 points)

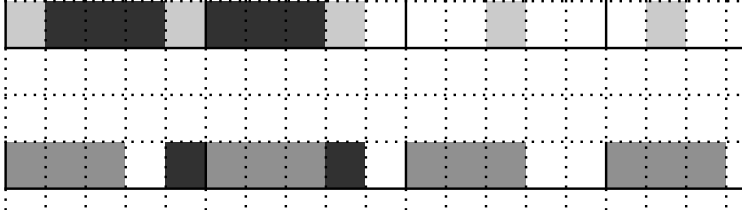
Dans un système comportant un processeur composé de deux coeurs identiques, les tâches sont exécutées suivant une politique d'ordonnement Global Deadline Monotonic. Soit l'ensemble de tâches périodiques synchrones suivantes :

Tâches	Capacité	Echéance	Période
T1	1	4	4
T2	3	5	5
T3	8	9	20

(a) (1 point) **Premier ordonnancement**

Calculer l'ordonnancement de ces tâches sur l'hyper-période. Toutes les échéances sont-elles respectées ?

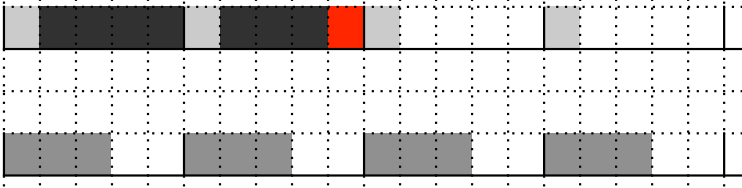
**Solution:** On ordonnance par échéance (Deadline Monotonic). T1 en gris clair, T2 en gris foncé, T3 en noir. Le système est ordonnançable.



(b) (1 point) **Second ordonnancement**

Supposons maintenant que la période de T1 soit égale à 5 l'échéance restant à 4. Calculez à nouveau l'ordonnancement sur l'hyper-période. Que constatez-vous maintenant ?

**Solution:** Les priorités ne changent pas puisque les échéances ne changent pas. Le système n'est pas ordonnançable.



(c) (1 point) **Conclusions**

Pourquoi les résultats sont-ils surprenants ? Comment qualifier ce phénomène ? Expliquez l'origine de celui-ci. Par exemple en observant les occurrences de temps de repos des cœurs. Une attention particulière sera accordée aux explications.

**Solution:** Le premier scénario a un taux d'utilisation de  $\frac{25}{20}$  et il est ordonnançable. Bien qu'ayant un taux d'utilisation moindre de  $\frac{24}{20}$ , le second scénario n'est pas ordonnançable. Il s'agit d'une anomalie d'ordonnancement.

**Exercice 6 : Couverture de code** (2 points)

On considère le code suivant. Indiquez ce que doit faire un banc de tests qui cherche à garantir la couverture de code selon l'approche Modified Condition/Decision Coverage. Puis illustrez en fournissant des valeurs possibles de a, b, c et d pour les tests à fournir. Une attention particulière sera accordée aux explications.

```
if ((a == 1) && (b < 2)) || ((c != 3) && (d == 4))
    F1(a, b, c, d);
    F2(a, b, c, d);
```

**Solution:** Il faut qu'en faisant varier une condition, en gardant les 3 autres conditions inchangées, la décision change.

$(a == 1)$	$(b < 2)$	$(c! = 3)$	$(d == 4)$	decision
V	V	V	F	V
F	V	V	F	F
V	F	V	F	F
V	F	V	V	V
V	F	F	V	F

**Total des points de la section : 9 points**

### 3 Traitement des erreurs temporelles

#### Exercice 7 (4 points)

Dans cet exercice, nous allons étudier la problématique du traitement des erreurs temporelles pour un ordonnancement mono-coeur de type EDF. Les priorités sont fixées implicitement en comparant les échéances absolues des jobs des tâches. (Rappel : une tâche périodique est une séquence de jobs ayant une date d'activation et une échéance absolue)). Plus l'échéance est petite plus le job est prioritaire. Le lot de tâche décrit dans la table 1, dénoté  $\mathcal{T}$ , correspond à des tâches périodiques à échéances implicites (échéance égale à la prochaine activation). Toutes les grandeurs sont exprimées en ms. Le budget est fixé égal au pire temps d'exécution pour chaque tâche (en l'absence de fautes). On suppose que toutes les tâches sont simultanément activées pour la première fois à l'instant  $t=0$ .

Tâche	$\tau_1$	$\tau_2$	$\tau_3$
Période	2	5	10
Budget	1	0,5	3,5

TABLE 1 – Lot de tâches  $\mathcal{T}$

- (a) ( $1\frac{1}{2}$  points) Supposez que  $\tau_1$  voit son temps d'exécution être égal pour chacun de ses jobs au double de son budget. Indiquez sur une hyperpériode du lot de tâche quels sont les jobs défaillants en l'absence de mécanisme de détection et traitement des erreurs (i.e. ceux qui ne respectent pas leur échéance)? Vous pouvez noter le  $k$ -ième job de  $\tau_i, j_i^k$ . Réponse justifiée attendue.
- (b) On suppose désormais que l'ordonnanceur est équipé avec un mécanisme de détection et de tolérance aux fautes qui : a) détecte les dépassements d'échéance sans latence, b) arrête le job courant et n'active pas les suivants pour la tâche pour laquelle le dépassement est détecté.
- (1 point) Est-ce que ce mécanisme aurait amélioré la capacité des jobs à respecter leurs échéances pour le comportement présenté en question a Réponse justifiée attendue.
  - (1 point) En supposant cette fois que la tâche  $\tau_2$  voit le temps d'exécution de chacun de ses jobs être égal au double de son budget. Quelles sont les jobs qui subiront une défaillance ? Réponse justifiée attendue.
  - ( $\frac{1}{2}$  point) On suppose que l'on utilise la détection de dépassement de budget (sans latence) et que l'on applique le même recouvrement de l'erreur (i.e. l'arrêt du job et de ses successeurs). Quelles sont les jobs qui pourraient subir une défaillance dans le pire cas (pour une faute toujours sur la tâche  $\tau_2$ ). Réponse justifiée attendue.

**Total des points de la section : 4 points**