SCIENCES SORBONNE UNIVERSITÉ

Rattrapage ASTRE
Master 2 Informatique Sep 2021

UE 5IN455

Année 2020-2021

2 heures – **Documents manuscrits autorisés**
Tout appareil de communication électronique interdit (téléphones. . .)

Le sujet est composé de sections indépendantes que l'on pourra traiter dans l'ordre souhaité. On vous demande de rédiger les quatre parties sur quatre copies séparées.

Partie 1. Model Checking

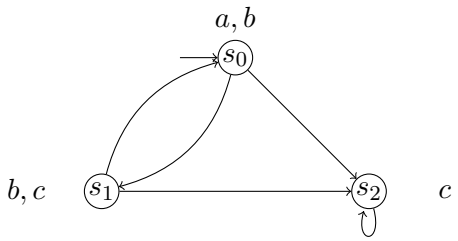


Figure 1: Structure de Kripke, dont les états s_0, s_1 et s_2 sont étiquetés par les propositions atomiques a,b, et c.

1 Model Checking LTL (5.5 points)

Question 1. (0.5 point) Donner la définition intuitive de la formule LTL $\varphi = \mathsf{G}(b \rightarrow \mathsf{X}\,\mathsf{F}\,b)$

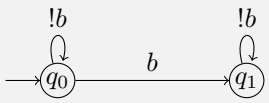
> Soit on voit b au moins une fois, donc $\mathsf{X}\,\mathsf{F}\,b$ à ce point, et on répète (G englobant).
> $\mathsf{G}\,\mathsf{F}\,b$: infiniment souvent b
> Soit on ne voit jamais b, ce qui satisfait l'implication $b \rightarrow \mathsf{X}\,\mathsf{F}\,b$
> $\mathsf{G}!b$: jamais b

Question 2. (1 point) Donner ψ, la négation de φ, *sous forme normale négative*.

$$
\begin{array}{rcl}
\psi & = & !(\mathsf{G}(b \rightarrow \mathsf{X}\,\mathsf{F}\,b)) \\
\psi & = & !(\mathsf{G}(!b \vee \mathsf{X}\,\mathsf{F}\,b)) \\
\psi & = & \mathsf{F}!(!b \vee \mathsf{X}\,\mathsf{F}\,b) \\
\psi & = & \mathsf{F}(!!b \wedge !\,\mathsf{X}\,\mathsf{F}\,b) \\
\psi & = & \mathsf{F}(b \wedge \mathsf{X}\,\mathsf{G}!b) \\
\psi & = & \top\,\mathsf{U}\,(b \wedge \mathsf{X} \perp \mathsf{R}!b)
\end{array}
$$

Question 3. (1.5 points) Dessiner le graphe de réduction commençant en $\{\psi\}$.

Question 4. (1.5 points) Terminer la construction de l'automate $\mathcal{A}_\psi$ associé à ψ. Vous n'oublierez pas de préciser les ensembles de transitions acceptants.



q1 est acceptant en Buchi classique. La transition qui boucle sur q1 est acceptante en Buchi étiqueté sur transitions.

Question 5. (0.5 point) Le modèle représenté Figure 1 satisfait-il φ? Justifier précisément.

Peut-on observer b puis infiniment $!b$?
Oui ! On voit que la trace du produit entre le système et A_ψ:
$(q_0, s_0)(q_1, s_2)^\omega$
est acceptée.

2 Model Checking CTL (2 points)

Question 1. (0.5 point) Exprimez en CTL la proposition suivante "A tout instant, si b est vrai, alors c est toujours vrai à l'instant suivant".

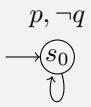
$\mathsf{AG}(b \to \mathsf{AX}\, c)$

Question 2. (1.5 points) Est-ce que le modèle de la figure 1 satisfait la formule CTL suivante : $\varphi = \mathsf{AF}\,\mathsf{EX}\,\mathsf{AG}\, c$
Vous déterminerez pour cela, en utilisant l'algorithme vu en cours, l'ensemble des états de la structure de Kripke vérifiant votre formule.

$c : \{s_1, s_2\}$
$\mathsf{AG}\, c : \{s_2\}$
$\mathsf{EX}\,\mathsf{AG}\, c : \{s_0, s_1, s_2\}$
$\mathsf{AF}\,\mathsf{EX}\,\mathsf{AG}\, c : \{s_0, s_1, s_2\}$
Cet ensemble couvre l'état initial s_0, donc oui la formule est vraie dans ce système.

3 Logiques temporelles (1 point)

Donner une structure de Kripke vérifiant la formule $\varphi = \mathsf{AF}(p \to \mathsf{EG}\,\neg q)$ et ne vérifiant pas la formule $\psi = \mathsf{F}(p \to q)$. Les formules φ et ψ sont-elles équivalentes?



Sur l'équivalence, clairement non, on vient d'exhiber un contre-exemple.

Partie 2. Ordonnancement

4 Ordonnancement multi-processeurs temps réel (3 points)

Dans cet exercice, nous cherchons à ordonnancer sur un processeur à deux coeurs le système suivant de tâches synchrones périodiques à échéances implicites :

Tâches	Budget	Période
T1	2	8
T2	2	10
T3	36	40

Question 1. (0.5 point) Rappeler le fonctionnement de Global-EDF.
Question 2. (1 point) En faisant le chronogramme, déterminez si le système est ordonnançable en utilisant Global-EDF.

En faisant le chronogramme sur l'hyper-période de 40, on trouve que le système est ordonnançable.

Question 3. (1 point) On augmente la période de T1 de 8 à 10. En faisant le chronogramme, déterminez si le système est ordonnançable en utilisant Global-EDF

En faisant le chronogramme sur l'hyper-période de 40, on trouve que le système n'est pas ordonnançable. T1 et T2 s'exécutent par trois fois au moins simultanément ce qui retarde T3 de quelques unités de temps pour lui faire rater son échéance.

Question 4. (0.5 point) Commenter les résultats des deux questions de la section ? Donner un nom à ce phénomène.

Le second système est moins chargé que le premier. Pourtant, le premier est ordonnançable alors que le second ne l'est pas. Il s'agit d'un phénomène connu sous le nom d'anomalie d'ordonnancement.

5 Couverture de code (1 point)

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

$$if((a == 1)\&\&((b! = 3)\&\&(c == 4)))F_1(a, b, c); F_2(a, b, c);$$

pour MC/DC il faut faire 4 tests en faisant varier les conditions de sorte que lorsqu'une seule condition varie, deux autres restant fixes, la décision varie.

	$a == 1$	$b! = 3$	$c == 4$	Decision
test1	True	True	True	True
test2	True	True	False	False
test3	True	False	True	False
test4	False	True	True	False

Partie 3. Vérification de modèles stochastiques

Dans cette section nous allons modéliser via une chaîne de Markov à temps discret le fonctionnement d'une fonction (écrite en C) utilisée dans une tâche temps réel. Les états s_3 et s_4 représentent les états où la fonction a terminée son exécution. L'état s_4 correspond à une exécution défaillante de la fonction et s_3 à une eécution correcte de la fonction. L'état s_0 est l'état dans lequel la fonction commence son exécution (à partir du moment où elle est "appelée").

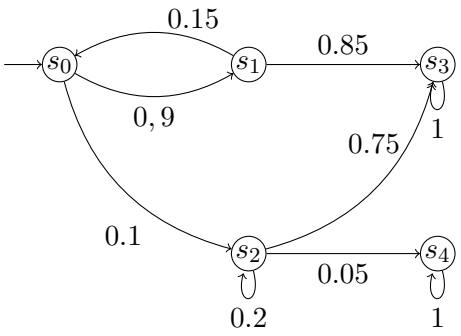


Figure 2: Chaîne de Markov M_0 modélisant le fonctionnement du code de contrôle

Question 1. (0,5 points) Rappelez brièvement ce qu'est le Cylindre associé au chemin $s_0.s_1.s_0.s_1$ d'un point de vue formel.

Le cylindre associé à un chemin p est l'ensemble des chemins d'exécution infinis dans la chaîne de Markov ayant pour préfixe p. Ainsi dans $L = \{s_0, s_1, .., s_4\}^\omega$, le cylindre de $s_0.s_1.s_0.s_1$ est l'ensemble $\{s_0.s_1.s_0.s_1.s_2^\omega\} \cup \{s_0.s_1.s_0.s_1.s_0.u | u \in L\}$.

Question 2. (0,5 points) On va supposer que l'on a 5 variables booléennes (une pour chaque état). Pour tout i, b_i est vrai en s_i et faux sinon. Donnez la formule CTL sur les $(b_i)_{i \leq 4}$ dont il faut évaluer la probabilité pour évaluer la fiabilité (probabilité de produire un résultat correct).

Réponse $F(b_3)$

Question 3. (1 point) Un ingénieur propose de limiter les comportements étudiés à des séquences d'au plus k transitions, quel impact (variation de la valeur) cette contrainte aura sur la fiabilité de la fonction ? (Justifiez votre réponse)

> Limiter la taille des "exécution" limite le nombre de scénarios disjoint pouvant satisfaire la propriété et donc cela ne peut que faire décroître la probabilité d'atteindre l'état s_3. Prenez $k = 1$ la probabilité devient trivialement nulle.

Question 4. (0,5 point) Soit Mat la matrice de transition de M_0 (qui indique la probabilité de passer de l'état s_i vers l'état s_j, en ligne i et colonne j). On suppose les lignes et colonnes numérotées de 0 à 4. Que représente $[1, 0, 0, 0, 0] * Mat^5$ (le produit du vecteur ligne [1,0,0,0,0] par Mat^5) et en quoi cela permet il de définir la fiabilité de la fonction pour un nombre borné de transitions ?

> Cette expression représente la distribution de probabilité des états de la chaine au bout de 5 transitions. Le coefficient en position 4 dans le vecteur résultant donne la fiabilité de la fonction au bout de 5 transitions.

Question 5. (0,5 points) Donnez la matrice de transition de M_0, Mat.

> $$\begin{pmatrix} 0 & 0.9 & 0.1 & 0 & 0 \\ 0.15 & 0 & 0.85 & 0 & 0 \\ 0 & 0 & 0.2 & 0.75 & 0.05 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Question 6. (1 points) Fournissez une preuve de l'affirmation suivante "La probabilité de fournir un calcul correct (s_3) croit strictement avec le temps laissé à la fonction" ? (indication : on pourra utiliser un minorant de cette valeur)

> On peut raisonner par récurrence par rapport aux chemins de taille k déjà en s_3, et l'ensemble des chemins de taille k arrivant sur les prédécesseurs de s_3. L'idée est de pointer du doigt que l'état correct est un état puis et que tout exécution de taille k reste correcte pour une taille $k + 1$.

Partie 4. Solvers et Logique

6 SMT (4 points)

On considère le problème suivant.

Chuck est un pilote d'avion qui doit faire 6 trajets dans 6 villes différentes (**A**lbany, **B**irmingham, **C**incinatti, **D**etroit, **E**l Paso, **F**argo) pour 6 clients différents (**U**nderwood, **V**anderkook, **W**ood, **X**ing, **Y**oung, **Z**ellman) qui se rendent en ville pour chacun une raison différente (**H**alloween, **I**dylle, **J**eu, **K**art, **L**oisir, **M**arriage).

Chuck a oublié qui est allé où, et pour quelle raison. On souhaite reconstruire cette information. On donne les indices suivants :

1. **E**l Paso (où Chuck a amené soit le voyageur "**I**dylle" soit le voyageur "**M**arriage") n'est pas l'endroit où **V**anderkook est allé.
2. **B**irmingham est là où Chuck a amené soit la personne fêtant "**I**dylle" soit **W**ood, mais pas les deux.
3. Chuck a amené à **A**lbany la personne qui allait aux "**L**oisirs" ou alors **V**anderkook, mais pas les deux.
4. **Y**oung est allé faire du **K**arting.
5. **U**nderdown est allé soit à **C**incinatti soit à **A**lbany (mais pas aux deux)
6. Ni **X**ing ni **V**anderkook ne sont allés à **F**argo. **F**argo c'était pour une "**I**dylle" ou un "**M**arriage".
7. A **D**etroit il y avait soit "**H**alloween" soit des "**L**oisirs".

Question 1. (2 points) Expliquez comment poser ce problème afin de le résoudre (ou décider qu'il n'a pas de solution !) à l'aide d'un solver SMT. On sera aussi précis que possible, en particulier on expliquera les variables introduites (et leur type, entiers, tableaux...), mais on ne demande pas à ce stade poser des assertions. Expliquez aussi comment interpréter le résultat rendu par le solver dans les deux cas SAT et UNSAT.

Donc on a 6 personnes, on souhaite affecter à chacune une ville et une raison de voyager.

On va considérer que les personnes U à Z sont les valeurs possibles des variables indiquant pour chaque lieu et pour chaque motif qui y est allé. (il y a d'autres solutions). On se définit donc un domaine U, V, W, X, Y, Z. On peut de façon similaire prendre un entier, et indexer $U = 0, ...$

On peut donc ensuite introduire 6 variables (A à F) à valeurs entre 0 et 5 (ou U à Z) inclus pour les villes.

e.g. Si la variable de la ville C vaut U, c'est U qui va à C.

De même on introduit 6 variables (H à M) pour les raisons de voyager.

Si H vaut U, c'est que c'est Underwood qui va à Halloween.

En pratique on a donc 12 variables à introduire, ou deux tableaux de 6 variables si on préfère.

On impose par énoncé que ces variables soient distinctes.

Distinct(A,B,C,D,E,F)

Distinct(H,I,J,K,L,M)

Ensuite on code les contraintes.

Unsat= pas de solution

Sat = le modèle dit clairement qui a fait quoi et où.

Question 2. (2 points) Donnez les assert à poser (en syntaxe logique du premier ordre $v \in \mathbb{B}, \vee, \wedge, \ldots ...$ ou avec celle du solver en notation préfixe sont OK) pour les contraintes numéro 3, 4, 5 et 6.

Chuck a amené à **A**lbany la personne qui allait aux "Loisirs" ou alors **V**anderkook, mais pas les deux en même temps.

$$(A = V \oplus L = A)$$

(donc un XOR)

Young est allé faire du **K**arting.

$$K = Y$$

Underdown est allé soit à **C**incinatti soit à **A**lbany (mais pas aux deux)

$$C = U \vee A = U$$

Ni **X**ing ni **V**anderkook ne sont allés à **F**argo. **F**argo c'était pour une "**I**dylle" ou un "**M**arriage".

$$F \neq V \wedge F \neq X$$

$$F = I \vee F = M$$

Ici le codage complet en syntaxe du solver SMT Z3:

```
(declare-datatypes () ((People U V W X Y Z)))

(declare-const A People)
(declare-const B People)
(declare-const C People)
(declare-const D People)
(declare-const E People)
(declare-const F People)

(declare-const H People)
(declare-const I People)
(declare-const J People)
(declare-const K People)
(declare-const L People)
(declare-const M People)

(assert (distinct A B C D E F))
(assert (distinct H I J K L M))


; 1.
(assert (or (= E I) (= E M)))
(assert (not (= E V)))

; 2.
(assert (xor (= B I) (= B W)))

; 3.
(assert (xor (= A V) (= L A)))

; 4.
(assert (= Y K))

; 5.
(assert (or (= C U) (= A U)))

; 6.
(assert (not (= F X)))
(assert (not (= F V)))
(assert (or (= F I) (= F M)))

; 7.
(assert (or (= D H) (= D L)))


(check-sat)
(get-model)
```

Avec le type énuméré ça se lit plutôt facilement.
20 % sur le résultat, et comment l'interpréter (10 pour SAT, 10 pour UNSAT).