IGL502/752 – Techniques de vérification et de validation

Université de Sherbrooke

Devoir 4

Enseignant:	Michael Blondin
Date de remise:	mardi 12 novembre 2024 à 13h29
À réaliser:	à deux ou individuellement au 1er cycle individuellement aux cycles supérieurs
Modalités:	remettre au format PDF en ligne sur Turnin
Pointage:	sur 30 points au 1er cycle sur 38 points aux cycles supérieurs

Dans ce devoir, l'abbréviation « BDD » réfère à « diagramme de décision binaire (réduit et ordonné) ».

Question 1. 4 pts

Donnez un algorithme qui transforme un BDD en une expression booléenne équivalente. Par « expression booléenne », on entend une expression constituée de variables booléennes et de connecteurs logiques, par ex. $(x_1 \vee x_2) \wedge (\neg x_2 \vee x_3)$. Autrement dit, donnez un algorithme qui accomplit cette tâche:

ENTRÉE:	un BDD B sur les variables $x_1 < x_2 < \cdots < x_n$, et un sommet u de B qui représente une fonction booléenne f_u
SORTIE:	une expression booléenne φ telle que $\varphi(x_1, x_2, \ldots, x_n) = f_u(x_1, x_2, \ldots, x_n)$

Question 2. 4 pts

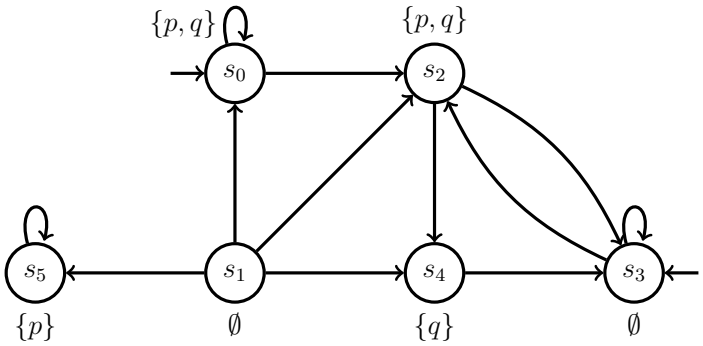
Donnez un algorithme qui résout ce problème:

ENTRÉE:	un BDD B sur les variables $x_1 < x_2 < \cdots < x_n$, et un sommet u de B qui représente une fonction booléenne f_u
SORTIE:	un sommet v qui représente $\neg f_u$

Votre algorithme peut (et doit probablement) mettre B à jour lors du calcul de v.

Question 3.

Considérons cette structure de Kripke $\mathcal{T} = (S, \to, I, AP, L)$:



Supposons que chaque état de $\mathcal{T}$ soit codé par la représentation binaire de son indice; autrement dit: $s_0 = 000$, $s_1 = 001$, $s_2 = 010$, $s_3 = 011$, $s_4 = 100$ et $s_5 = 101$ (les autres chaînes de bits sont invalides).

(a) Donnez une expression booléenne φ qui représente l'ensemble $[\![p]\!]$; 1,5 pts

(b) Donnez une expression booléenne ψ qui représente l'ensemble $[\![q]\!]$; 1,5 pts

(c) Construisez un BDD pour φ à l'aide de la procédure `build`; 2 pts

(d) Construisez un BDD pour ψ à l'aide de la procédure `build`; 2 pts

(e) Construisez un BDD pour $\varphi \vee \psi$ à l'aide de la procédure `apply`; 2 pts

(f) Expliquez comment procéder afin de déterminer si $\mathcal{T} \models \forall \mathsf{X}(p \vee q)$ à partir du BDD obtenu en (e). Vous n'avez pas à calculer le résultat des opérations mentionnées dans vos explications; il suffit de décrire ce qu'il faudrait calculer. Vous pouvez introduire de nouvelles variables au besoin. 3 pts

Vous devez faire évoluer le même BDD en (c), (d) et (e). Donnez également l'arbre de récursion en (c), (d) et (e). Si vous utilisez des optimisations de `build` ou `apply`, dites lesquelles.

Question 4.

Considérons le programme suivant constitué de deux fonctions et d'une variable booléenne globale `x`:

```
     bool x ∈ {faux, vrai}

     foo(bool y):                bar():
f0:    si (x ∨ y):          b0:    x = ¬x
         bar()              b1:    foo(x)
f1:      bar()
       sinon:
         x = ¬y
f2:    assert(x)
```

(a) Modélisez le programme avec un système à pile $\mathcal{P}$. 3 pts

(b) Donnez un $\mathcal{P}$-automate $\mathcal{A}$ tel que $\mathit{Conf}(\mathcal{A})$ est l'ensemble des configurations où l'assertion est enfreinte. 2 pts

(c) Construisez partiellement un $\mathcal{P}$-automate qui accepte $\mathrm{Pre}^*(\mathit{Conf}(\mathcal{A}))$; plus précisément, donnez cinq nouvelles transitions obtenues à partir de $\mathcal{A}$ en exécutant l'algorithme de saturation vu en classe. 3 pts

(d) Si vous aviez entièrement calculé le $\mathcal{P}$-automate qui accepte $\mathrm{Pre}^*(\mathit{Conf}(\mathcal{A}))$ en (c), comment auriez-vous pu déterminer si un appel initial à `foo` peut enfreindre l'assertion? 2 pts

Vous n'avez pas à répondre à la question ci-dessous si vous êtes au premier cycle. Si vous y répondez, vous pourrez obtenir jusqu'à 2 points bonus.

🎓 Question 5. (cycles supérieurs) 8 pts

Les BDD peuvent aussi servir de structure de données pour la manipulation d'ensembles d'entiers. En effet, il suffit de considérer les chaînes binaires comme des représentations de nombres.

Plus précisément, soit u un sommet d'un BDD sur les variables $x_1 < x_2 < \ldots < x_n$, et soit $f_u(x_1, x_2, \ldots, x_n)$ la fonction booléenne calculée par u. Nous dénotons par $[\![u]\!] = \{b \in \{0,1\}^n : f_u(b) = 1\}$ l'ensemble des affectations qui satisfont f_u. Pour toute affectation $b \in \{0,1\}^n$, nous dénotons par $\text{val}(b)$ la valeur numérique de b (où le bit le moins significatif est à droite). Par exemple, $\text{val}(1101) = 13$, $\text{val}(0110) = 6$ et $\text{val}(0000) = 0$.

Donnez un algorithme qui résout ce problème:

ENTRÉE:	un BDD B sur les variables $x_1 < x_2 < \cdots < x_n$, et un sommet $u \neq 0$ de B
SORTIE:	$\min\{\text{val}(b) : b \in [\![u]\!]\}$

Autrement dit, votre algorithme doit retourner le plus petit entier représenté par le sommet u. Pour être en mesure d'obtenir tous les points, votre algorithme doit fonctionner en temps polynomial par rapport à la taille du sous-diagramme enraciné en u.