Cours de Model Checking

Leçon 4 : Algorithmes de Model Checking, suite

Sébastien Bardin

CEA-LIST, Laboratoire de Sûreté Logicielle

sebastien.bardin@cea.fr
http://sebastien.bardin.free.fr/

Rappels

Model Checking

Technique de vérification automatique de systèmes réactifs

Ingrédients

- $\mathcal{M}$ = système de transitions
- φ = formule temporelle
- MC = est-ce que $\mathcal{M} \models \varphi$?

Trois algorithmes de model checking

- CTL : technique de marquage des états
- Fair CTL : marquage de CTL + détection des états "fair"
- LTL : transformation de la formule en automate de Büchi, puis intersection avec le système et test du vide

Remarque : les composantes fortements connexes jouent un rôle important dans chacun des algorithmes

Contenu

Complément sur les automates de Büchi

- test du vide
- union
- intersection
- structure de Kripke $\longrightarrow$ automate de Büchi

Complément sur CTL

- cas **EG**

Complément sur Fair CTL

- trouver les états fair
- réutilisation de l'algorithme de marquage de CTL

Plan

- Automates de Büchi
- CTL model checking
- Fair CTL Model checking

Rappel : automates de Büchi

Outil pour reconnaître / manipuler des ensembles (infinis) de mots infinis

Automate de Büchi $B = \langle \Sigma, Q, \rightarrow, q_0, F \rangle$

- Σ alphabet (ensemble fini)
- Q ensemble fini d'états
- $\rightarrow \subseteq Q \times \Sigma \times Q$ les transitions
- q_0 l'état initial
- $F \subseteq Q$ l'ensemble des états finaux / acceptants

Un mot infini σ est reconnu si son exécution dans l'automate B, i.e.
$q_0 \xrightarrow{\sigma_0} q_1 \xrightarrow{\sigma_1} \ldots \xrightarrow{\sigma_n} q_n \xrightarrow{\sigma_{n+1}} \ldots$
passe infiniment souvent par l'ensemble F

Test du vide

Proposition : $\mathcal{L}(B) \neq \emptyset$ ssi il existe $q_F \in F$ tel que $q_0 \xrightarrow{*} q_F$ et q_F appartient à une SCC (non triviale, maximale) de B

($\Leftarrow$) : ok car de q_F on peut atteindre q_F, d'où un chemin infini partant de q_0 et passant infiniment par q_F, donc accepté par B

($\Rightarrow$) : Soit un mot infini σ accepté par B. Donc l'exécution infinie de σ dans B passe infiniment souvent par F. Comme F est fini, il existe au moins un $q_F \in F$ tel que l'exécution de σ passe infiniment souvent par q_F : $q_0 \xrightarrow{*} q_F \xrightarrow{+} q_F \rightarrow \ldots$. De plus, q_F appartient à la composante fortement connexe non triviale $q_F \xrightarrow{+} q_F$. Il appartient donc forcément aussi à une SCC non triviale maximale.

Algorithme de test du vide :

- calculer les SCC de B. Ne garder que celles intersectant F (notées FSCC). Soit L l'ensemble des états des FSCC.
- alors $\mathcal{L}(B) \neq \emptyset$ ssi q_0 peut atteindre L
- ou encore $\mathcal{L}(B) \neq \emptyset$ ssi $post^*(q_0) \cap L \neq \emptyset$ ssi $q_0 \in pre^*(L)$

Union d'automates de Büchi

Problème : union d'automates de Büchi

- Entrées : deux automates de Büchi sur le même alphabet Σ
  $B_1 = \langle \Sigma, Q_1, \rightarrow_1, qi_1, F_1 \rangle$ et $B_2 = \langle \Sigma, Q_2, \rightarrow_2, qi_2, F_2 \rangle$
- Question : calculer $B = \langle \Sigma, Q, \rightarrow, qi, F \rangle$ tq
  $\mathcal{L}(B) = \mathcal{L}(B_1) \cup \mathcal{L}(B_2)$

Solution : $B = \langle \Sigma, Q_1 \times Q_2, \rightarrow, (qi_1, qi_2), (F_1 \times Q_2) \cup (Q_1 \times F_2) \rangle$ et $\rightarrow$ est définie par : (soit $a \in \Sigma$)

- $(q_1, q_2) \xrightarrow{a} (q_1', q_2')$ ssi $q_1 \xrightarrow{a}_1 q_1'$ et $q_2 \xrightarrow{a}_2 q_2'$

($\Leftarrow$) : ok car l'automate B déroule l'exécution d'un mot σ en parallèle sur B_1 et sur B_2. Par exemple, si $\sigma \in \mathcal{L}(B_1)$, alors σ dans B passera infiniment souvent par des états de la forme (q_{F_1}, q_2) et sera accepté par B. Idem pour $\mathcal{L}(B_2)$.
($\Rightarrow$) : Le mot est accepté par B ssi il passe infiniment souvent par F_1 ou par F_2. Donc σ passe infiniment souvent par au moins un des deux, donc il est accepté au moins par un des deux automates.

Intersection d'automates de Büchi

Problème : intersection d'automates de Büchi

- Entrées : deux automates de Büchi sur le même alphabet Σ
  $B_1 = \langle \Sigma, Q_1, \rightarrow_1, qi_1, F_1 \rangle$ et $B_2 = \langle \Sigma, Q_2, \rightarrow_2, qi_2, F_2 \rangle$
- Question : calculer $B = \langle \Sigma, Q, \rightarrow, qi, F \rangle$ tq
  $\mathcal{L}(B) = \mathcal{L}(B_1) \cap \mathcal{L}(B_2)$

Solution : $B = \langle \Sigma, Q_1 \times Q_2 \times \{1,2\}, \rightarrow, (qi_1, qi_2, 1), F_1 \times Q_2 \times \{1\} \rangle$ et $\rightarrow$ est définie par : (soit $a \in \Sigma$)

- $(q_1, q_2, 1) \xrightarrow{a} (q_1', q_2', 1)$ ssi $q_1 \xrightarrow{a}_1 q_1'$ et $q_2 \xrightarrow{a}_2 q_2'$ et $q_1' \notin F_1$
- $(q_1, q_2, 1) \xrightarrow{a} (q_1', q_2', 2)$ ssi $q_1 \xrightarrow{a}_1 q_1'$ et $q_2 \xrightarrow{a}_2 q_2'$ et $q_1' \in F_1$
- $(q_1, q_2, 2) \xrightarrow{a} (q_1', q_2', 2)$ ssi $q_1 \xrightarrow{a}_1 q_1'$ et $q_2 \xrightarrow{a}_2 q_2'$ et $q_2' \notin F_2$
- $(q_1, q_2, 2) \xrightarrow{a} (q_1', q_2', 1)$ ssi $q_1 \xrightarrow{a}_1 q_1'$ et $q_2 \xrightarrow{a}_2 q_2'$ et $q_2' \in F_2$

Les deux exécutions sont lancées en parallèle, et le témoin (1 ou 2) indique le prochain type d'état final à atteindre (1 pour F_1, 2 pour F_2). Quand on attend un état dans F_i et qu'on l'atteint, le témoin change de valeur. Cela permet d'assurer que l'exécution a bien une infinité de F_1 et une infinité de F_2.

Structure de Kripke vers automate de Büchi

Problème :

- Entrées : une structure de Kripke $\mathcal{M} = \langle Q, \rightarrow, P, l, s_0 \rangle$
- Question : calculer $B = \langle \Sigma, Q', \rightarrow', q_0, F \rangle$ tq $\mathcal{L}(B) = \mathcal{L}(\mathcal{M})$

Solution : soit $P = \{p_1, \ldots, p_n\}$, on note 2^P l'ensemble des sous-ensembles de P. On définit B par $B = \langle 2^P, Q', \rightarrow', q'_0, Q' \rangle$ et :

- Q' = un état par transition $t \in \rightarrow$ et un état initial nouveau q'_0
- les transitions : $t_1 \xrightarrow{D}{}' t_2$ ssi $t_1 = (\_, s_j)$, $t_2 = (s_j, \_)$ et $D \subseteq P = l(s_j)$, et les transitions $q'_0 \xrightarrow{l(s_0)} (s_0, \_)$

Intuition : si on s'intéressait aux transitions de $\mathcal{M}$: B serait comme $\mathcal{M}$ (l'alphabet serait le "nom" des transitions), tous les états seraient acceptants. Mais on veut des propriétés de suites d'états : on inverse les états / transitions de $\mathcal{M}$, l'alphabet est le sous-ensemble des propriétés vraies (intuitivement : lues dans l'état courant).

Plan

- Automates de Büchi
- CTL model checking
- Fair CTL Model checking

Plan

- Automates de Büchi
- CTL model checking
- Fair CTL Model checking

Rappel : algorithme de marquage

Marquer les états vérifiant la formule CTL φ

- on raisonne en termes d'états, pas de trace
- marquage récursif : déjà les sous-formules
- pour chaque opérateur CTL basique (ex : **EF**), définir comment on marque **EF**φ à partir du marquage de φ

Exemples simples

- $\varphi_1 \wedge \varphi_2$: marquer les états marqués pour φ_1 et pour φ_2
- $\neg\varphi$: marquer les états non marqués pour φ
- **EX**φ : marquer les états qui peuvent atteindre en 1 étape un état marqué par φ
- **EF**φ : marquer les états qui peuvent atteindre en 0, 1 ou $+$ étapes un état marqué par φ
- **AX**φ : marquer les états dont tous les successeurs en une étape sont marqués par φ

cas **EG**φ, lien avec les SCC

soit une structure de Kripke $\mathcal{M} = \langle Q, \rightarrow, P, l, s_0 \rangle$

Proposition : un état $q \in Q$ vérifie **EG**φ ssi $q \xrightarrow{\pi} q'$ tq tous les états de π vérifient φ et q' appartient à une composante fortement connexe non triviale (mais pas forcément maximale) dont tous les états vérifient φ

($\Leftarrow$) : ok

($\Rightarrow$) : Soit $q \in Q$ vérifiant **EG**φ. Donc il existe un chemin infini σ partant de q tq φ est vraie dans tous les états de σ. Comme σ est infini et que Q est fini, σ passe forcément une infinité de fois par au moins un état. Soit q' un de ces états. On a donc :

$q \xrightarrow{\pi_1} q' \xrightarrow{\pi_2} q' \xrightarrow{\cdots}$ (remarque : $\pi_2 \neq \epsilon$)

Comme $\pi_1\pi_2$ est un préfixe de σ, tous les états de π_1 vérifient φ.
De plus, q' appartient à la composante fortement connexe non triviale $q' \xrightarrow{\pi_2} q'$ dont tous les états vérifient φ.

cas $\mathbf{EG}\varphi$, lien avec les SCC (2)

Conséquence : calcul du marquage (soit Q_φ les états marqués pour φ)

remarque : "composante fortement connexe non triviale de $\mathcal{M}$ dont tous les états vérifient φ" peut se ramener à "composante fortement connexe non triviale maximale de $\mathcal{M}$ restreint à Q_φ"

- trouver les SCC (maximale) de Q_φ. On note L l'ensemble de tous ces états
- alors : marquer pour $\mathbf{EG}\varphi$ les états qui atteignent L par un chemin dont tous les états vérifient φ
- plus formellement : $Q_{\mathbf{EG}\varphi} = (\lambda X \mapsto pre(X) \cap Q_\varphi)^*(L)$

Plan

- Automates de Büchi
- CTL model checking
- Fair CTL Model checking

Plan

- Automates de Büchi
- CTL model checking
- Fair CTL Model checking

Rappel : Fair CTL

CTL + équité (restreinte) : ajout dans le modèle directement

- contraintes d'équité : ensembles d'ensembles d'états $F_1, \ldots, F_n \subseteq Q$
- chemins fair = passant infiniment souvent par chaque ensemble F_i
- états fair = état d'où part un chemin fair

Nouvelle relation de satisfaction $\models_{\text{fair}}$

- $\mathcal{M}, s \models_{\text{fair}} p$ ssi $p \in l(s)$ et s est un état fair
- $\mathcal{M}, s \models_{\text{fair}} \mathbf{A}\varphi$ ssi tous les chemins fair partant de s vérifient φ
- $\mathcal{M}, s \models_{\text{fair}} \mathbf{E}\varphi$ ssi il existe un chemins fair partant de s vérifiant φ

Principe de l'algorithme de MC pour Fair CTL

1. trouver les états fair par accessibilité de Fair SCC
2. marquer ces états avec une nouvelle proposition `fair`
3. se ramener au cas normal en exprimant $\mathcal{M}, s \models_{\text{fair}} \varphi$ en fonction de $\models, \varphi$, `fair`

Trouver les états fair

Problème : marquage des états fair

- Entrées : une structure de Kripke $\mathcal{M} = \langle Q, \rightarrow, P, I, s_0 \rangle$ et des contraintes d'équités $F_1, \ldots, F_n \subseteq Q$
- Question : trouver les états fair de $\mathcal{M}$

On appelle Fair SCC une SCC qui intersecte chacun des F_i

Proposition : un état $q \in Q$ est fair ssi il existe $q' \in Q$ tel que $q \xrightarrow{*} q'$ et q' appartient à une Fair SCC (non triviale, maximale) de $\mathcal{M}$

($\Leftarrow$) : ok (on peut aisément construire un chemin fair partant de q et bouclant dans la Fair SCC)
($\Rightarrow$) : si q est fair, alors il existe un chemin fair σ qui part de q et passe infiniment osuvent par chacun des F_i. Comme les F_i sont finis, il existe au moins pour chaque F_i un q_{F_i} visité infiniment souvent par σ. On peut donc écrire σ comme : $q \xrightarrow{*} q_{F_1} \xrightarrow{*} \ldots q_{F_n} \xrightarrow{*} q_{F_1} \xrightarrow{*} \ldots$. Il vient que les q_{F_i} sont dans la même SCC (maximale, non triviale), cette SCC est donc fair.

Trouver les états fair (2)

Algorithme :

- trouver les SCC (maximale) de $\mathcal{M}$. Trouver celles qui sont fair. On note L l'ensemble de tous les états des Fair SCC.
- alors : les états fair sont ceux qui peuvent atteignent L
- plus formellement : $Q_{fair} = pre^*(L)$

Réutiliser l'algorithme de marquage de CTL

But : exprimer les propriétés CTL sur une structure fair comme des propriétés CTL sur une structure normale avec la proposition atomique `fair`

(attention : ne veut pas dire que Fair CTL = CTL)

Quelques équivalences :

- $\mathcal{M}, s \models_F p$ ssi $\mathcal{M}, s \models p \wedge \text{fair}$
- $\mathcal{M}, s \models_F \neg p$ ssi $\mathcal{M}, s \models \neg p \wedge \text{fair}$
- $\mathcal{M}, s \models_F \mathbf{EX}\varphi$ ssi $\mathcal{M}, s \models \mathbf{EX}(\varphi \wedge \text{fair})$
- $\mathcal{M}, s \models_F \mathbf{AX}\varphi$ ssi $\mathcal{M}, s \models \text{fair} \wedge \mathbf{AX}(\text{fair} \implies \varphi)$
- $\mathcal{M}, s \models_F \mathbf{AG}\varphi$ ssi $\mathcal{M}, s \models \text{fair} \wedge \mathbf{AG}(\text{fair} \implies \varphi)$
- $\mathcal{M}, s \models_F \mathbf{E}\varphi_1\mathbf{U}\varphi_2$ ssi $\mathcal{M}, s \models \mathbf{E}\varphi_1\mathbf{U}(\varphi_2 \wedge \text{fair})$

Remarque : attention à la négation !!! (rentrer les négations)