Septième partie

Vérification de modèles

Plan

Le principe de la vérification est le parcours systématique et éventuellement exhaustif de l'espace d'états du système, afin de vérifier la compatibilité des exécutions avec les propriétés attendues. On suppose avoir un système de transitions fini.

Plan

Analyse du problème

Pour vérifier les propriétés temporelles, on a besoin de :

- mémoriser les états déjà visités ($\mathcal{A}nciens$)
- ainsi que les nouveaux états courants ($\mathcal{N}ouveaux$). Il peut s'agir d'états prédécesseurs ou successeurs des états visités.

L'ensemble des états visités finit donc par devenir beaucoup plus gros que les états courants, par simple accumulation.

Les ensembles $\mathcal{A}$ et $\mathcal{N}$ n'auront pas nécessairement la même représentation mémoire pour des raisons d'efficacité.

Algorithme générique

```
procedure exploration(Init, Image, Pop, Push, Minus, Union)
begin
 A := {}; -- états explorés
 N := Init; -- états non explorés
 while (N <> {})
 begin
  Choix  := Pop(N);                    -- Choix d'état(s) à explorer
  Images := Image(Choix) Minus A;      -- Images non déjà explorées
  A := A Union Choix;                  -- Choix a été traité
  N := Push(N, Images);                -- ses images sont à explorer
 end;
 return A;                             -- on a tout exploré
end
```

Prérequis

Besoins sur les états déjà visités

- opération pour l'ajout d'états nouveaux (`Union`). Il faut donc convertir leur représentation mémoire ;
- représentation concise en mémoire (très gros ensemble).

Besoins sur les états nouveaux

- opération pour la différence ensembliste avec les états déjà visités (`Minus`) ;
- opération pour le calcul d'image directe ou inverse (`Image`) ;
- représentation mémoire permettant de construire un ordre global (`Pop` et `Push`).

Quelques solutions possibles

Représentation des états déjà visités

- explicite : bit-state encoding, table de hachage, risque de collision.
- implicite : Binary Decision Diagrams (10^{1300}), ou formules SAT (10^{1500}), ou polyèdres (∞), ...

Ordre global sur les états nouveaux

- parcours en profondeur (occupation mémoire limitée)
- parcours en largeur (petits contre-exemples)
- partitionnement (en g^{al}, en fonction de l'occupation mémoire engendrée)

Plan

Réduction du problème

- Les traces sont en nombre infini en général.
- On se ramène à un problème de traces régulières.

Théorème : Soit $\mathcal{S} = \langle S, I, R \rangle$ un ST fini, L une formule LTL

$$\begin{aligned} & \exists \sigma \in S^\omega : \sigma \in Exec(\mathcal{S}) \wedge \sigma \models L \\ \Leftrightarrow \quad & \exists \sigma_p, \sigma_c \in S^\star : \langle \sigma_p \rightarrow \sigma_c^\omega \rangle \in Exec(\mathcal{S}) \wedge \langle \sigma_p \rightarrow \sigma_c^\omega \rangle \models L \end{aligned}$$

Caractérisation opérationnelle de LTL

- Un modèle d'une formule LTL est une exécution (souvent infinie).
- Pour trouver des cycles, il faut se rappeler des états déjà visités.

Théorème : Soit L une formule LTL

$$\exists \mathcal{S}_L = \langle S, I, R, F \rangle : | S | = O(2^{|L|}) \wedge \forall \sigma : \sigma \in Exec(\mathcal{S}_L) \Leftrightarrow \sigma \models L.$$

La mémorisation des états intéressants déjà visités est effectuée par les états et les transitions de $\mathcal{S}_L$.

Méthode de vérification

La vérification s'effectue en plusieurs étapes :

1. engendrer le ST $\mathcal{S}_{\neg L}$ équivalent à la formule $\neg L$ ($O(2^{|L|})$) ;
2. composer de façon synchronisée $\mathcal{S}_{\neg L}$ avec le système à valider $\mathcal{S}$ en un système produit $\mathcal{P}$ ($O(|\mathcal{S}| \times 2^{|L|})$) ;
3. rechercher une exécution de $\mathcal{P}$ vérifiant les contraintes d'équité de $\mathcal{S}$ et de $\mathcal{S}_L$. Cette exécution est donc à la fois une exécution de $\mathcal{S}$ et $\neg L$;
4. si une telle exécution existe, alors il s'agit d'un contre-exemple qui montre que $\mathcal{S}$ ne vérifie pas L, sinon L est bien vérifiée.

Cette méthode de vérification privilégie essentiellement le calcul des états successeurs pour le calcul du point 3.

Note : les formules LTL utilisées sont généralement simples.

Plan

Principe

- Les modèles en CTL sont les états (et non les exécutions).
- Pas de problème de construction ou de représentation des modèles.
- Calcul de l'ensemble des états qui satisfont une formule donnée, de façon inductive sur la structure des formules.

La méthode la plus simple et directe est de réaliser un parcours des transitions en marche arrière.

Note : on définit $[\![F]\!] \triangleq \{s \in S \mid s \models F\}$

Opérateurs logiques

Les opérateurs logiques sont transposés en opérateurs ensemblistes.

$$[\![F_1 \wedge F_2]\!] = [\![F_1]\!] \cap [\![F_2]\!]$$

$$[\![\neg F]\!] = S \setminus [\![F]\!]$$

$$[\![True]\!] = S$$

Opérateurs temporels

Les opérateurs temporels nécessitent le calcul des prédécesseurs :
$Pred(s) \triangleq \{s' \in S \mid (s', s) \in R\}$

$$[\![s]\!] = \{s\}$$

$$[\![\exists \bigcirc F]\!] = Pred([\![F]\!])$$

$$[\![F_1 \exists \mathcal{U} F_2]\!] = \lim_{i \to +\infty} X_i, \text{ avec } \begin{cases} X_0 & = [\![F_2]\!] \\ X_{i+1} & = X_i \cup ([\![F_1]\!] \cap Pred(X_i)) \end{cases}$$

Complexité

- On procède en fait par marquage :
  $s \mapsto \{F \mid s \models F\}$ plutôt que $F \mapsto \{s \mid s \models F\}$.
- Chaque opérateur d'une formule F peut amener un parcours complet de S.
- Complexité en $O(|S| \times |F|)$.
- On peut prendre en compte l'équité.

Note : la complexité du temps linéaire est exponentielle, la complexité du temps arborescent est linéaire !