Ensuring Completeness of Symbolic Verification Methods for Infinite-State Systems

P. A. Abdulla - B. Jonsson

13/05/02

Élodie-Jane Sims.

But :
analyse d'accessibilité pour des systèmes infinis

Plan :

- algorithme pour des systèmes finis
- algorithme pour des systèmes infinis
- conditions de terminaison de l'algorithme

Remarque : les propriétés de sûreté peuvent être réduites à des problèmes d'accessibilité.

Méthode

Analyse arrière itérative à partir des états finaux.

On représente les ensembles infinis d'états par des *contraintes*.
Contraintes : Φ
Ens. d'états : $[\![\Phi]\!]$

Toy-program

var:v, w : natural numbers

$v, w := \mathtt{v}_0, \mathtt{w}_0;$

do

$s_1 : v \geq 1 \rightarrow v, w := v - 1, w + 1$

$[\!]$

$s_2 : v * w \geq 4 \rightarrow w := 5 * w$

od

$Init = \{\langle 3, 0\rangle\}$ $F : \phi_{0,10}$

$[\![\phi_{a,b}]\!] = \{\langle v, w\rangle | v \geq a \wedge w \geq b\}$

i	Φ_i	rule	new constraints generated	covered / added
0	$\phi_{0,10}$	s_1	$\phi_{1,9}$	added to Φ_1
		s_2	$\phi_{2,2}$	added to Φ_1
			$\phi_{1,4}$	added to Φ_1
1	$\phi_{0,10}$, $\phi_{1,9}$, $\phi_{2,2}$, $\phi_{1,4}$	s_1	$\phi_{2,8}$	covered by $\phi_{2,2}$
			$\phi_{3,1}$	added to Φ_2
			$\phi_{2,3}$	covered by $\phi_{2,2}$
		s_2	$\phi_{1,4}$	covered by $\phi_{1,4}$
			$\phi_{2,2}$	covered by $\phi_{2,2}$
			$\phi_{4,1}$	covered by $\phi_{3,1}$
2	$\phi_{0,10}$, $\phi_{1,9}$, $\phi_{2,2}$, $\phi_{1,4}$, $\phi_{3,1}$	s_1	$\phi_{4,0}$	added to Φ_3
		s_2	$\phi_{1,4}$	covered by $\phi_{1,4}$
			$\phi_{2,2}$	covered by $\phi_{2,2}$
			$\phi_{4,1}$	covered by $\phi_{3,1}$
3	$\phi_{0,10}$, $\phi_{1,9}$, $\phi_{2,2}$, $\phi_{1,4}$, $\phi_{3,1}$, $\phi_{4,0}$	s_1	$\phi_{5,0}$	covered by $\phi_{4,0}$
		s_2	$\phi_{1,4}$	covered by $\phi_{1,4}$
			$\phi_{2,2}$	covered by $\phi_{2,2}$
			$\phi_{4,1}$	covered by $\phi_{3,1}$
4	$\phi_{0,10}$, $\phi_{1,9}$, $\phi_{2,2}$, $\phi_{1,4}$, $\phi_{3,1}$, $\phi_{4,0}$	s_1		
		s_2		

Reachable cas fini : *ens. états*

Procedure $Reachable1$
Input
$\mathcal{T} = \langle \Sigma, \longrightarrow \rangle$: transition system
I: set of initial states
F: set of final states finite set of constraints
Output are there $\sigma \in I$ and $\sigma' \in [\![\Phi_F]\!]$ such that $\sigma \longrightarrow^* \sigma'$?

var V : set of states
begin
$V := F$;
while $pre(\longrightarrow, V) \nsubseteq V$ **do** $V := V \cup pre(\longrightarrow, V)$ **od** ;
if $I \cap V = \emptyset$ **then return** unreachable **else return** reachable
end

Terminaison car
- V augmente toujours dans la boucle
- V est inclu dans Σ qui est fini

Reachable2 cas fini : *états*

Procedure *Reachable2*
Input
  $\mathcal{T} = \langle \Sigma, \longrightarrow \rangle$: transition system
  I: set of initial states
  F: set of final states finite set of constraints
Output are there $\sigma \in I$ and $\sigma' \in [\![\Phi_F]\!]$ such that $\sigma \longrightarrow^* \sigma'$?

var W , V : sets of states
begin
  $W, V := F, \emptyset$;
  while $W \neq \emptyset$ **do**
    choose $\sigma \in W$;
    $W := W \setminus \{\sigma\}$;
    if $\sigma \in I$ **then return** *reachable*
    else
      if $\sigma \notin V$
      then
        $V := V \cup \{\sigma\}$;
        $W := W \cup pre(\longrightarrow, \{\sigma\})$
  od ;
  return *unreachable*
end

W : ens. états visités à traiter
V : ens. états traités

Terminaison triviale

Reachable cas infini : *contraintes*

Procedure $Reachable3(\langle \Sigma, \longrightarrow \rangle, I, F)$
Input
$\mathcal{T} = \langle \Sigma, \longrightarrow \rangle$: transition system
I: set of states, $I = \neg[\![\Phi_I]\!]$ for some finite set Φ_I of constraints
Φ_F: finite set of constraints
Output are there $\sigma \in I$ and $\sigma' \in [\![\Phi_F]\!]$ such that $\sigma \longrightarrow^* \sigma'$?

var W , V : sets of constraints
begin
$W, V := \Phi_F, \emptyset$;
while $W \neq \emptyset$ **do**
choose $\phi \in W$;
$W := W \setminus \{\phi\}$;
if $[\![\phi]\!] \cap I \neq \emptyset$ **then return** *reachable*
else
if $\nexists \phi' \in V : \phi' \sqsubseteq \phi$ (* insertion test *)
then
$V := V \cup \{\phi\}$;
$W := W \cup \Phi$ **where** $[\![\Phi]\!] = pre(\longrightarrow, [\![\phi]\!])$;
od
return *unreachable*
end

Déf : $\phi' \sqsubseteq \phi$ ssi $[\![\phi]\!] \subseteq [\![\phi']\!]$

insertion test $\equiv [\![\phi]\!] \not\subseteq [\![V]\!]$

$[\![\phi]\!] \cap Init \neq \emptyset \equiv [\![\phi]\!] \not\subseteq [\![\Phi_I]\!]$
$\equiv \Phi_I \not\sqsubseteq \{\phi\}$

Conditions pour que l'algorithme

Soit calculable :

- $\phi' \sqsubseteq \phi$ calculable sur des ens. finis de contraintes
- Φ tq $[\![\Phi]\!] = pre(\rightarrow, [\![\phi]\!])$ calculable (on dit que le système de contraintes $\mathcal{C}$ est *clos* par rapport à $\mathcal{T}$)

Termine :

- $\sqsubseteq$ est un beau pré-ordre[1]
  (on dira directement que $\mathcal{C}$ est b.p.o, car $\sqsubseteq$ définit par $\mathcal{C}$)

Preuve de la terminaison :
prendre la suite des contraintes ajoutés à V, par construction lors du test d'insertion, $\forall i, j \quad \phi_i \not\sqsubseteq \phi_j$, comme nous avons un b.p.o, cette suite est finie.

1. $\preceq$ beau pré-ordre ssi
- réflexive
- transitive
- $\forall \quad a_0 a_1 a_2 ..$ infinie $\exists i < j$ avec $a_i \preceq a_j$

Conditions de clotûre

Déf1 : $\sigma_1 \preceq_{\mathcal{C}} \sigma_2$ ssi
$\forall \phi \in \mathcal{C} \;\; \sigma_1 \in [\![\phi]\!] \Rightarrow \sigma_2 \in [\![\phi]\!]$

Déf2 : $\mathcal{C}$ disjonctif ssi
$\forall \phi_1, \phi_2 \in \mathcal{C} \;\; \exists \Psi \subseteq \mathcal{C} \;\; [\![\Phi_1]\!] \cap [\![\Phi_2]\!] = \cup_{\phi \in \Psi} [\![\phi]\!]$

Déf3 : $\mathcal{T}$ est monotone par rapport à $\preceq$ ssi
$\forall \sigma_1, \sigma_2, \sigma_3 \quad (\sigma_1 \preceq \sigma_2 \wedge \sigma_1 \longrightarrow \sigma_3) \Rightarrow (\exists \sigma_4 \;\; \sigma_3 \preceq \sigma_4 \wedge \sigma_2 \longrightarrow \sigma_4)$

Théorème : Si $\mathcal{C}$ est disjonctif, b.p.o,
$\mathcal{C}$ clos par rapport à $\mathcal{T}$ ssi
$\mathcal{T}$ monotone par rapport à $\preceq_{\mathcal{C}}$

Programmes

P :

- V : ens. variables typées
- ens. fini d'action : $g \rightarrow v_1, ..., v_n := e_1, ..., e_n$
  g est une assertion

$[\![P]\!] = \langle \Sigma, \rightarrow \rangle$

- Σ : ens. des valuations des variables dans leurs domaines
- $\rightarrow$: ens. (σ, σ') tq. $\exists$ une action tq:
  - $\sigma(g) = true$
  - $\sigma' = \sigma[\sigma(e_i)/v_i]$

– Monotonie : Si $g(\bar{d_1}, \bar{a_1}) \wedge \bar{d_1} \preceq \bar{d_2}$
alors $\exists \bar{a_2} \;\; g(\bar{d_2}, \bar{a_2}) \wedge e_i(\bar{d_1}, \bar{a_1}) \preceq e_i(\bar{d_2}, \bar{a_2})$

– Monotonie P par rapport à $\preceq$ $\equiv$ monotonie de $[\![P]\!]$ par rapport à $\preceq$.

$Dom(\bar{d})$: domaine des variables du programme
$Dom(\bar{a})$: domaine des paramètres du programme

Programmes monotones

– Un programme dont les actions sont de la forme
$\bar{d}(\bar{x}) \preceq \bar{f}(\bar{v}, \bar{x}) \rightarrow \bar{v} := \bar{e}(\bar{v}, \bar{x})$,
avec $\bar{f}$ et $\bar{e}$ monotones est monotone.

Utile par exemple pour : programmes sur $\mathcal{N}$, réseaux de Pétri, programmes sur des séquences.

– Un programme dont les actions sont de la forme
$g(\bar{v}, \bar{x}) \rightarrow \bar{v} := \bar{e}(\bar{v}, \bar{x})$
avec :
- g une conjonction de $d \preceq dist(u, u')$
- les e_i sont des variables ou des paramètres
- si $\bar{d} \preceq \bar{d}'$ et $\bar{a}$ alors $\exists \bar{a}'$ t.q. $\langle \bar{d}, \bar{a} \rangle \preceq \langle \bar{d}', \bar{a}' \rangle$
est monotone.

Utile par exemple pour : automates relationnels, automates à horloges

Une classe simple : programmes sur des entiers naturels

$[\![\phi_{\bar{d}}]\!] = \{\bar{v} | \bar{d} \leq \bar{v}\}$

$\leq$ et $\preceq_{\mathcal{C}}$ coïncident
donc $\mathcal{C}_{\mathcal{N}}$ est b.p.o et disjonctif

donc on a la monotonicité des programmes avec des actions :
$\bar{d}(\bar{x}) \preceq \bar{f}(\bar{v}, \bar{x}) \rightarrow \bar{v} := \bar{e}(\bar{v}, \bar{x})$,
avec toutes les fonctions apparaissant monotones (ex : addition, mutltiplication,...)

donc l'algorithme termine

Apports du papier

Donne des conditions générales sous lesquelles l'analyse d'accessibilité dans des systèmes infinis termine.

Donne des exemples classiques de systèmes infinis pour lesquels on peut utiliser l'algorithme.