Apprentissage Actif avec des Contres Exemples Inductifs

Juliette Jacquot

3 juillet 2024

Sommaire

1. Regular Model Checking
2. L'algorithme L_{ICE}
3. SAT Solver
4. Application et améliorations possibles
5. Conclusion

Regular Model Checking

Sommaire

1. Regular Model Checking
   - Principe
   - Application
   - Exemple
   - Problème
2. L'algorithme L_{ICE}
3. SAT Solver
4. Application et améliorations possibles
5. Conclusion

Regular Model Checking

Principe

Qu'est-ce que le Regular Model Checking [2]?

Regular Model Checking

Principe

Qu'est-ce que le Regular Model Checking [2]?

- **But**: Vérifier la terminaison d'un algorithme

Regular Model Checking

Principe

Qu'est-ce que le Regular Model Checking [2]?

- **But**: Vérifier la terminaison d'un algorithme
- **Contexte**:
  - Ensemble d'états de départ

Regular Model Checking

Principe

Qu'est-ce que le Regular Model Checking [2]?

- **But**: Vérifier la terminaison d'un algorithme
- **Contexte**:
  - Ensemble d'états de départ
  - Ensemble d'états à éviter

Regular Model Checking

Principe

Qu'est-ce que le Regular Model Checking [2]?

- **But**: Vérifier la terminaison d'un algorithme
- **Contexte**:
  - Ensemble d'états de départ
  - Ensemble d'états à éviter
  - Transition entre états

Regular Model Checking

Principe

Qu'est-ce que le Regular Model Checking [2]?

- **But**: Vérifier la terminaison d'un algorithme
- **Contexte**:
  - Ensemble d'états de départ
  - Ensemble d'états à éviter
  - Transition entre états
- **Résultat**: Un ensemble d'états:

Regular Model Checking

Principe

Qu'est-ce que le Regular Model Checking [2]?

- **But**: Vérifier la terminaison d'un algorithme
- **Contexte**:
  - Ensemble d'états de départ
  - Ensemble d'états à éviter
  - Transition entre états
- **Résultat**: Un ensemble d'états:
  - Contient les états atteignables

Regular Model Checking

Principe

Qu'est-ce que le Regular Model Checking [2]?

- **But**: Vérifier la terminaison d'un algorithme
- **Contexte**:
  - Ensemble d'états de départ
  - Ensemble d'états à éviter
  - Transition entre états
- **Résultat**: Un ensemble d'états:
  - Contient les états atteignables
  - Exclut les états non sécurisés

Regular Model Checking

Principe

Qu'est-ce que le Regular Model Checking [2]?

- **But**: Vérifier la terminaison d'un algorithme
- **Contexte**:
  - Ensemble d'états de départ
  - Ensemble d'états à éviter
  - Transition entre états
- **Résultat**: Un ensemble d'états:
  - Contient les états atteignables
  - Exclut les états non sécurisés
  - Invariant inductif

Quel est le lien avec l'apprentissage actif?

- **Contexte**:
  - Ensemble d'états de départ
  - Ensemble d'états à éviter
  - Transition entre états
- **Résultat**: Un ensemble d'états
  - Contient les états atteignables
  - Exclue les états non sécurisés
  - Invariant inductif

Quel est le lien avec l'apprentissage actif?

- **Contexte**:
  - Ensemble d'états de départ $\longrightarrow$ Un langage Init
  - Ensemble d'états à éviter
  - Transition entre états
- **Résultat**: Un ensemble d'états
  - Contient les états atteignables
  - Exclue les états non sécurisés
  - Invariant inductif

Quel est le lien avec l'apprentissage actif?

- **Contexte**:
  - Ensemble d'états de départ $\longrightarrow$ Un langage Init
  - Ensemble d'états à éviter $\longrightarrow$ Un langage Bad
  - Transition entre états
- **Résultat**: Un ensemble d'états
  - Contient les états atteignables
  - Exclue les états non sécurisés
  - Invariant inductif

Quel est le lien avec l'apprentissage actif?

- **Contexte**:
  - Ensemble d'états de départ $\longrightarrow$ Un langage Init
  - Ensemble d'états à éviter $\longrightarrow$ Un langage Bad
  - Transition entre états $\longrightarrow$ Une transition entre mots Post
- **Résultat**: Un ensemble d'états
  - Contient les états atteignables
  - Exclue les états non sécurisés
  - Invariant inductif

Quel est le lien avec l'apprentissage actif?

- **Contexte**:
  - Ensemble d'états de départ $\longrightarrow$ Un langage Init
  - Ensemble d'états à éviter $\longrightarrow$ Un langage Bad
  - Transition entre états $\longrightarrow$ Une transition entre mots Post
- **Résultat**: Un ensemble d'états $\longrightarrow$ Un langage L
  - Contient les états atteignables
  - Exclue les états non sécurisés
  - Invariant inductif

Regular Model Checking

Application

Quel est le lien avec l'apprentissage actif?

- **Contexte**:
  - Ensemble d'états de départ $\longrightarrow$ Un langage Init
  - Ensemble d'états à éviter $\longrightarrow$ Un langage Bad
  - Transition entre états $\longrightarrow$ Une transition entre mots Post
- **Résultat**: Un ensemble d'états $\longrightarrow$ Un langage L
  - Contient les états atteignables $\longrightarrow \mathrm{Post}^*(\mathrm{Init}) \subseteq L$
  - Exclue les états non sécurisés
  - Invariant inductif

Regular Model Checking

Application

Quel est le lien avec l'apprentissage actif?

- **Contexte**:
  - Ensemble d'états de départ $\longrightarrow$ Un langage Init
  - Ensemble d'états à éviter $\longrightarrow$ Un langage Bad
  - Transition entre états $\longrightarrow$ Une transition entre mots Post
- **Résultat**: Un ensemble d'états $\longrightarrow$ Un langage L
  - Contient les états atteignables $\longrightarrow \mathrm{Post}^*(\mathrm{Init}) \subseteq L$
  - Exclue les états non sécurisés $\longrightarrow L \cap \mathrm{Pre}^*(\mathrm{Bad}) = \emptyset$
  - Invariant inductif

Regular Model Checking

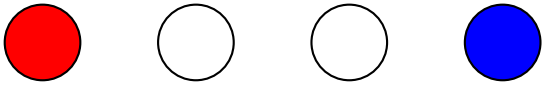
Application

Quel est le lien avec l'apprentissage actif?

- **Contexte**:
  - Ensemble d'états de départ $\longrightarrow$ Un langage Init
  - Ensemble d'états à éviter $\longrightarrow$ Un langage Bad
  - Transition entre états $\longrightarrow$ Une transition entre mots Post
- **Résultat**: Un ensemble d'états $\longrightarrow$ Un langage L
  - Contient les états atteignables $\longrightarrow \mathrm{Post}^*(\mathrm{Init}) \subseteq L$
  - Exclue les états non sécurisés $\longrightarrow L \cap \mathrm{Pre}^*(\mathrm{Bad}) = \emptyset$
  - Invariant inductif $\longrightarrow \mathrm{Post}(L) \subseteq L$

Regular Model Checking

Exemple

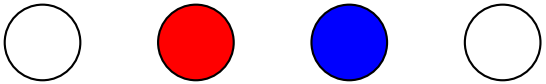


- Deux tokens

Mot représentant: $AxxB$

Regular Model Checking

Exemple



- Deux tokens
- Se déplacent

Mot représentant: $xABx$

Regular Model Checking

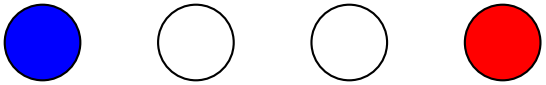
Exemple

Mot représentant: $xBAx$

- Deux tokens
- Se déplacent
- Peuvent se croiser

Regular Model Checking

Exemple

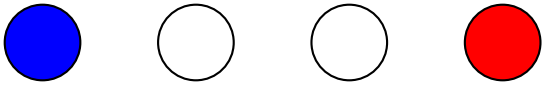


Mot représentant: *BxxA*

- Deux tokens
- Se déplacent
- Peuvent se croiser
- Ne doivent pas atteindre le côté opposé

Regular Model Checking

Exemple



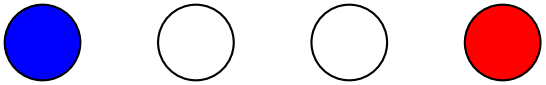
Mot représentant: *BxxA*

- Deux tokens
- Se déplacent
- Peuvent se croiser
- Ne doivent pas atteindre le côté opposé

Contexte final:

Regular Model Checking

Exemple



Mot représentant: $BxxA$

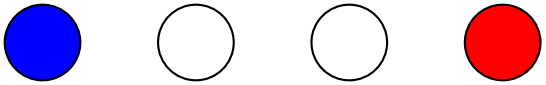
- Deux tokens
- Se déplacent
- Peuvent se croiser
- Ne doivent pas atteindre le côté opposé

Contexte final:

- $\mathrm{Init} = Ax(xx)^*B$
- $\mathrm{Bad} = Bx^*A$

Regular Model Checking

Exemple



Mot représentant: $BxxA$

- Deux tokens
- Se déplacent
- Peuvent se croiser
- Ne doivent pas atteindre le côté opposé

Contexte final:

- $\mathrm{Init} = Ax(xx)^*B$
- $\mathrm{Bad} = Bx^*A$
- $\mathrm{Post}^*(\mathrm{Init}) = x^nAx(xx)^*Bx^n$
- $\mathrm{Pre}^*(\mathrm{Bad}) = x^nA(xx)^*Bx^n + x^nBx^*Ax^n$

Regular Model Checking

Problème

Certains mots n'ont pas d'appartenance définie:

Certains mots n'ont pas d'appartenance définie:

- Choix arbitraire pendant les requêtes

Regular Model Checking

Problème

Certains mots n'ont pas d'appartenance définie:

- Choix arbitraire pendant les requêtes
- Deux options souvent utilisées:

Regular Model Checking

Problème

Certains mots n'ont pas d'appartenance définie:

- Choix arbitraire pendant les requêtes
- Deux options souvent utilisées:
  - Langage le plus restrictif: $\mathrm{Post}^*(\mathrm{Init})$

Regular Model Checking

Problème

Certains mots n'ont pas d'appartenance définie:

- Choix arbitraire pendant les requêtes
- Deux options souvent utilisées:
  - Langage le plus restrictif: $\mathrm{Post}^*(\mathrm{Init})$
  - Langage le moins restrictif: $\Sigma^* \setminus \mathrm{Pre}^*(\mathrm{Bad})$

Regular Model Checking

Problème

Certains mots n'ont pas d'appartenance définie:

- Choix arbitraire pendant les requêtes
- Deux options souvent utilisées:
  - Langage le plus restrictif: $\mathrm{Post}^*(\mathrm{Init})$
  - Langage le moins restrictif: $\Sigma^* \setminus \mathrm{Pre}^*(\mathrm{Bad})$

Langage choisit peut être non régulier: $x^n Ax(xx)^* Bx^n$

Regular Model Checking

Problème

Certains mots n'ont pas d'appartenance définie:

- Choix arbitraire pendant les requêtes
- Deux options souvent utilisées:
  - Langage le plus restrictif: $\mathrm{Post}^*(\mathrm{Init})$
  - Langage le moins restrictif: $\Sigma^* \setminus \mathrm{Pre}^*(\mathrm{Bad})$

Langage choisit peut être non régulier: $x^n Ax(xx)^* Bx^n$

L'algorithme peut **ne pas terminer**

L'algorithme L_{ICE}

Sommaire

1. Regular Model Checking
2. L'algorithme L_{ICE}
   - Principe
   - Relations inductives
   - Requêtes d'appartenance
   - Requêtes de validité
3. SAT Solver
4. Application et améliorations possibles
5. Conclusion

L'algorithme L_{ICE}

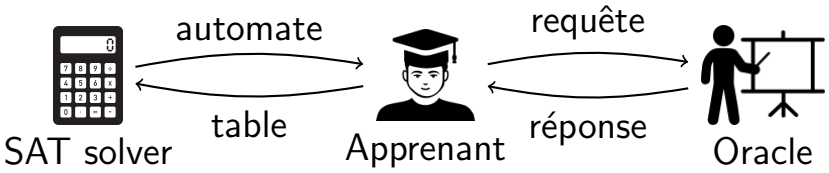
Principe

Basé sur l'algorithme L^* [1]:

L'algorithme L_{ICE}

Principe

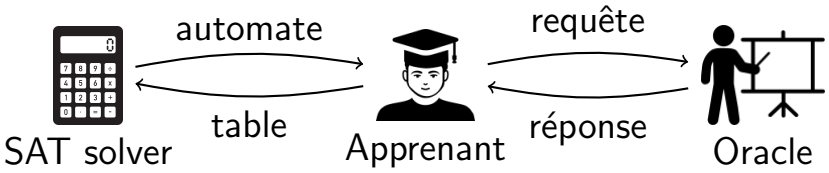
Basé sur l'algorithme L^* [1]:



L'algorithme L_{ICE}

Principe

Basé sur l'algorithme L^* [1]:

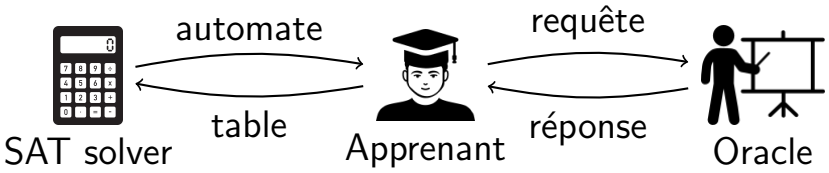


- Ajout d'une appartenance "inconnue"

L'algorithme L_{ICE}

Principe

Basé sur l'algorithme L^* [1]:

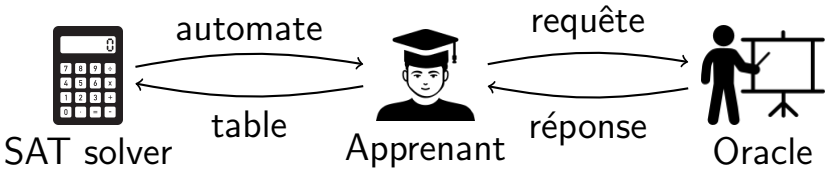


- Ajout d'une appartenance "inconnue"
- Ajout d'une relation inductive entre mots

L'algorithme L_{ICE}

Principe

Basé sur l'algorithme L^* [1]:



- Ajout d'une appartenance "inconnue"
- Ajout d'une relation inductive entre mots
- Ajout d'un SAT solver

Relations entres mots:
$\mathrm{Post}(xAxxxB) = xxAxBx$

L'algorithme L_{ICE}

Relations inductives

Relations entres mots:

$\mathrm{Post}(xAxxxB) = xxAxBx \rightarrow\ xAxxxB \in L \Rightarrow xxAxBx \in L$

L'algorithme L_{ICE}

Relations inductives

Relations entres mots:
$\mathrm{Post}(xAxxxB) = xxAxBx \rightarrow\ xAxxxB \in L \Rightarrow xxAxBx \in L$

Représentation en listes:

L'algorithme L_{ICE}

Relations inductives

Relations entres mots:
$\mathrm{Post}(xAxxxB) = xxAxBx \rightarrow xAxxxB \in L \Rightarrow xxAxBx \in L$

Représentation en listes:

Indice	0	1	2	3
Mot	*xAxxB*	*xxABx*	*xxBAx*	*xBxxA*
Lien	$\mathrm{Post}^0(xAxxB)$	$\mathrm{Post}^1(xAxxB)$	$\mathrm{Post}^2(xAxxB)$	$\mathrm{Post}^3(xAxxB)$

Format des requêtes d'appartenance:

Format des requêtes d'appartenance: mot w, liste de mots inconnus U

L'algorithme L_{ICE}

Requêtes d'appartenance

Format des requêtes d'appartenance: mot w, liste de mots inconnus U

3 options:

L'algorithme L_{ICE}

Requêtes d'appartenance

Format des requêtes d'appartenance: mot w, liste de mots inconnus U

3 options:

- $w \in \mathrm{Post}^*(\mathrm{Init})$:

L'algorithme L_{ICE}

Requêtes d'appartenance

Format des requêtes d'appartenance: mot w, liste de mots inconnus U

3 options:

- $w \in \mathrm{Post}^*(\mathrm{Init})$:$w \in L$

L'algorithme L_{ICE}

Requêtes d'appartenance

Format des requêtes d'appartenance: mot w, liste de mots inconnus U

3 options:

- $w \in \mathrm{Post}^*(\mathrm{Init})$:$w \in L$
- $w \in \mathrm{Pre}^*(\mathrm{Bad})$:

L'algorithme L_{ICE}

Requêtes d'appartenance

Format des requêtes d'appartenance: mot w, liste de mots inconnus U

3 options:

- $w \in \mathrm{Post}^*(\mathrm{Init})$:$w \in L$
- $w \in \mathrm{Pre}^*(\mathrm{Bad})$:$w \notin L$

L'algorithme L_{ICE}

Requêtes d'appartenance

Format des requêtes d'appartenance: mot w, liste de mots inconnus U

3 options:

- $w \in \mathrm{Post}^*(\mathrm{Init})$:$w \in L$
- $w \in \mathrm{Pre}^*(\mathrm{Bad})$:$w \notin L$
- w n'a pas d'appartenance définie
  2 résultats possibles:

L'algorithme L_{ICE}

Requêtes d'appartenance

Format des requêtes d'appartenance: mot w, liste de mots inconnus U

3 options:

- $w \in \mathrm{Post}^*(\mathrm{Init})$:$w \in L$
- $w \in \mathrm{Pre}^*(\mathrm{Bad})$:$w \notin L$
- w n'a pas d'appartenance définie
  2 résultats possibles:
  - w a un lien avec un représentant u de la liste U:
    renvoyer u et i, $w = \mathrm{Post}^i(u)$

L'algorithme L_{ICE}

Requêtes d'appartenance

Format des requêtes d'appartenance: mot w, liste de mots inconnus U

3 options:

- $w \in \mathrm{Post}^*(\mathrm{Init})$:$w \in L$
- $w \in \mathrm{Pre}^*(\mathrm{Bad})$:$w \notin L$
- w n'a pas d'appartenance définie
  2 résultats possibles:
  - w a un lien avec un représentant u de la liste U:
    renvoyer u et i, $w = \mathrm{Post}^i(u)$
  - w est un nouveau représentant à ajouter à U

Format des requêtes de validité:

L'algorithme L_{ICE}

Requêtes de validité

Format des requêtes de validité: automate représentant un langage L

L'algorithme L_{ICE}

Requêtes de validité

Format des requêtes de validité: automate représentant un langage L

On cherche un mot w tel que:

L'algorithme L_{ICE}

Requêtes de validité

Format des requêtes de validité: automate représentant un langage L

On cherche un mot w tel que:

- $w \in \mathrm{Init}, w \notin L$

L'algorithme L_{ICE}

Requêtes de validité

Format des requêtes de validité: automate représentant un langage L

On cherche un mot w tel que:

- $w \in \mathrm{Init}, w \notin L$
- $w \in \mathrm{Bad}, w \in L$

L'algorithme L_{ICE}

Requêtes de validité

Format des requêtes de validité: automate représentant un langage L

On cherche un mot w tel que:

- $w \in \mathrm{Init}, w \notin L$
- $w \in \mathrm{Bad}, w \in L$

Sinon, on cherche un mot w tel que $w \in L$, mais $\mathrm{Post}(w) \notin L$:

L'algorithme L_{ICE}

Requêtes de validité

Format des requêtes de validité: automate représentant un langage L

On cherche un mot w tel que:

- $w \in \mathrm{Init}, w \notin L$
- $w \in \mathrm{Bad}, w \in L$

Sinon, on cherche un mot w tel que $w \in L$, mais $\mathrm{Post}(w) \notin L$:

- $w \in \mathrm{Post}^*(\mathrm{Init})$, contre-exemple: $\mathrm{Post}(w)$

L'algorithme L_{ICE}

Requêtes de validité

Format des requêtes de validité: automate représentant un langage L

On cherche un mot w tel que:

- $w \in \mathrm{Init}, w \notin L$
- $w \in \mathrm{Bad}, w \in L$

Sinon, on cherche un mot w tel que $w \in L$, mais $\mathrm{Post}(w) \notin L$:

- $w \in \mathrm{Post}^*(\mathrm{Init})$, contre-exemple: $\mathrm{Post}(w)$
- $w \in \mathrm{Pre}^*(\mathrm{Bad})$, contre-exemple: w

L'algorithme L_{ICE}

Requêtes de validité

Format des requêtes de validité: automate représentant un langage L

On cherche un mot w tel que:

- $w \in \mathrm{Init}, w \notin L$
- $w \in \mathrm{Bad}, w \in L$

Sinon, on cherche un mot w tel que $w \in L$, mais $\mathrm{Post}(w) \notin L$:

- $w \in \mathrm{Post}^*(\mathrm{Init})$, contre-exemple: $\mathrm{Post}(w)$
- $w \in \mathrm{Pre}^*(\mathrm{Bad})$, contre-exemple: w
- Contre exemple inductif: $(w, \mathrm{Post}(w))$

SAT Solver

Sommaire

1. Regular Model Checking
2. L'algorithme L_{ICE}
3. SAT Solver
   - Description
   - Contenu de la table
   - Forme de la table
   - Gestion des cas unsat
4. Application et améliorations possibles
5. Conclusion

SAT Solver

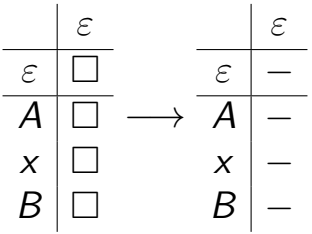
Description

Objectif: créer une table d'observation pour combler les trous

SAT Solver

Description

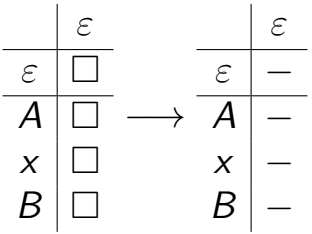
Objectif: créer une table d'observation pour combler les trous



SAT Solver

Description

Objectif: créer une table d'observation pour combler les trous

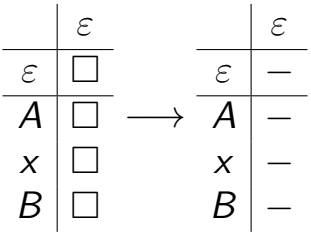


Encodage du SAT solver:

SAT Solver

Description

Objectif: créer une table d'observation pour combler les trous



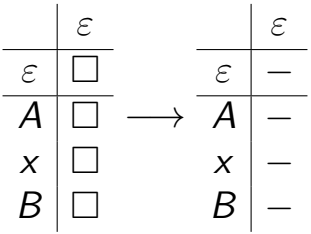
Encodage du SAT solver:

- x_w: $w \in L$

SAT Solver

Description

Objectif: créer une table d'observation pour combler les trous



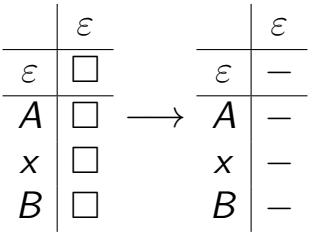
Encodage du SAT solver:

- x_w: $w \in L$
- b_w: w est un préfixe

SAT Solver

Description

Objectif: créer une table d'observation pour combler les trous



Encodage du SAT solver:

- x_w: $w \in L$
- b_w: w est un préfixe
- e_{w_1,a,w_2}: les mots $w_1 a$ et w_2 sont dans la même classe d'équivalence de Myhill-Nerode

SAT Solver

Contenu de la table

	ε
ε	—
A	—
x	—
B	—

SAT Solver

Contenu de la table

	ε
ε	—
A	—
x	—
B	—

- $\sigma_w := x_w \Leftrightarrow T(w)$: les mots déjà définis sont conservés

SAT Solver

Contenu de la table

	ε
ε	—
A	—
x	—
B	—

- $\sigma_w := x_w \Leftrightarrow T(w)$: les mots déjà définis sont conservés
- $\upsilon_{u,i,j} := x_{\mathrm{Post}^i(u)} \Rightarrow x_{\mathrm{Post}^j(u)}$: l'appartenance de certains mots inconnus est inductive

SAT Solver

Contenu de la table

	ε
ε	—
A	—
x	—
B	—

- $\sigma_w := x_w \Leftrightarrow T(w)$: les mots déjà définis sont conservés
- $\upsilon_{u,i,j} := x_{\mathrm{Post}^i(u)} \Rightarrow x_{\mathrm{Post}^j(u)}$: l'appartenance de certains mots inconnus est inductive
- $\Psi_{p,a,p'} := e_{p,a,p'} \Leftrightarrow \bigwedge_{s \in S} (x_{pas} \leftrightarrow x_{p's})$: le successeur d'un préfixe est équivalent à un préfixe

SAT Solver

Forme de la table

	ε
ε	—
A	—
x	—
B	—

SAT Solver

Forme de la table

	ε
ε	—
A	—
x	—
B	—

- $\eta_\varepsilon := b_\varepsilon$: le mot vide doit être dans l'ensemble des préfixes

SAT Solver

Forme de la table

	ε
ε	—
A	—
x	—
B	—

- $\eta_\varepsilon := b_\varepsilon$: le mot vide doit être dans l'ensemble des préfixes
- $\eta_{pa} := b_{pa} \Rightarrow b_p$: le mot pa peut être un préfixe si p est un préfixe

SAT Solver

Forme de la table

	ε
ε	—
A	—
x	—
B	—

- $\eta_\varepsilon := b_\varepsilon$: le mot vide doit être dans l'ensemble des préfixes
- $\eta_{pa} := b_{pa} \Rightarrow b_p$: le mot pa peut être un préfixe si p est un préfixe
- $\Phi_{p,a} := b_p \Rightarrow \left(b_{pa} \vee \bigvee_{p' \in P} e_{p,a,p'} \right)$: la table est fermée

	ε
ε	—
A	—
x	—
B	—

- $\eta_\varepsilon := b_\varepsilon$: le mot vide doit être dans l'ensemble des préfixes
- $\eta_{pa} := b_{pa} \Rightarrow b_p$: le mot pa peut être un préfixe si p est un préfixe
- $\Phi_{p,a} := b_p \Rightarrow \left(b_{pa} \vee \bigvee_{p' \in P} e_{p,a,p'} \right)$: la table est fermée
- $\Delta_{p,a} := b_{pa} \Rightarrow \bigwedge_{p' \in P \setminus \{pa\}} \neg e_{p,a,p'}$: la table est distincte

Utilisation des UNSAT cores [3]:

Utilisation des UNSAT cores [3]:

	ε
ε	$-$
A	$+$
x	$\square$
B	$\square$

Utilisation des UNSAT cores [3]:

	ε
ε	$-$
A	$+$
x	$\square$
B	$\square$

Clause $\Phi_{\varepsilon,A}$ non respectée: besoin d'ajouter A aux préfixes

Application et améliorations possibles

Sommaire

1 Regular Model Checking

2 L'algorithme L_{ICE}

3 SAT Solver

4 Application et améliorations possibles
- Application
- Résultats
- Améliorations

5 Conclusion

Application et améliorations possibles

Application

Cas d'exemple:

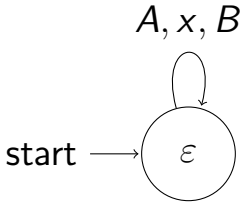
- $\mathrm{Init} = Ax(xx)^*B$
- $\mathrm{Bad} = Bx^*A$
- Post: non bloquée et sans cycle

Application et améliorations possibles

Application

Cas d'exemple:

- $\mathrm{Init} = Ax(xx)^*B$
- $\mathrm{Bad} = Bx^*A$
- Post: non bloquée et sans cycle

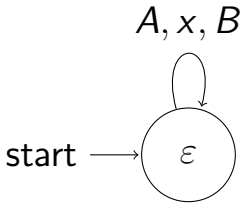


Application et améliorations possibles

Application

Cas d'exemple:

- $\mathrm{Init} = Ax(xx)^*B$
- $\mathrm{Bad} = Bx^*A$
- Post: non bloquée et sans cycle

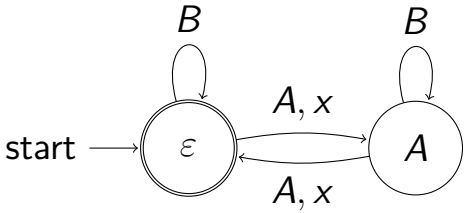


Contre exemple: AxB

Application et améliorations possibles

Application

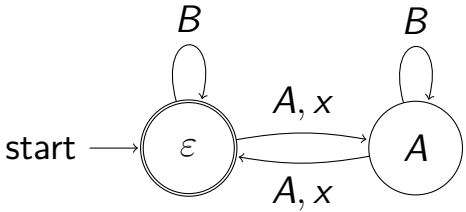
Cas d'exemple:



Application et améliorations possibles

Application

Cas d'exemple:

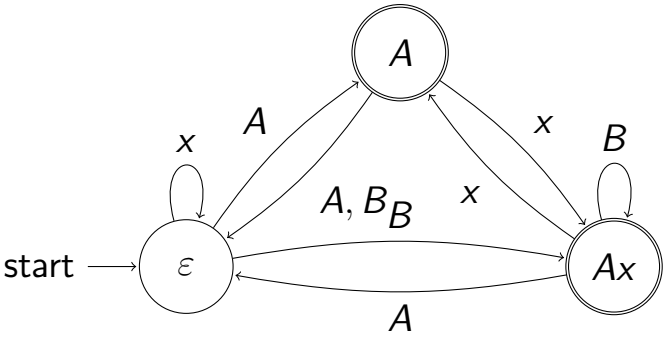


Contre exemple: BxA

Application et améliorations possibles

Application

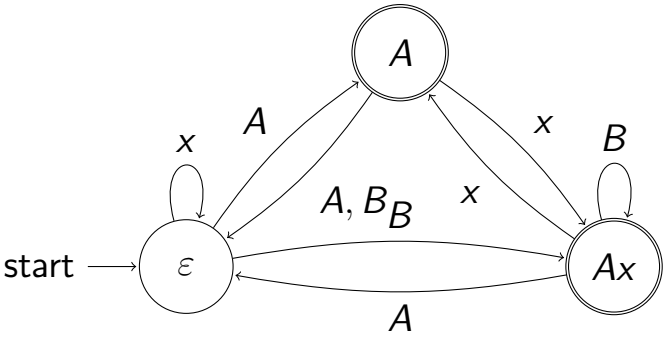
Cas d'exemple:



Application et améliorations possibles

Application

Cas d'exemple:

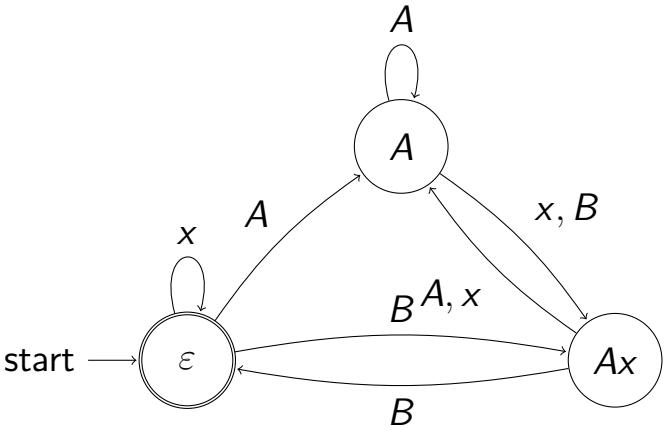


Contre exemple: $AxxAB, xAxBA$

Application et améliorations possibles

Application

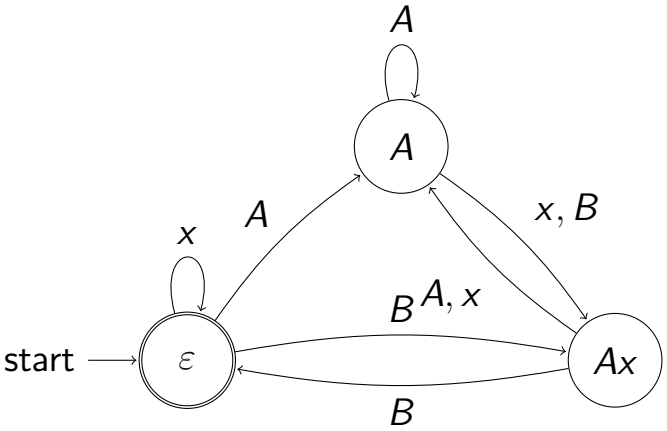
Cas d'exemple:



Application et améliorations possibles

Application

Cas d'exemple:



Contre exemple: aucun

Application et améliorations possibles

Résultats

Init	Bad	Post	États	Appartenance	Validité
$Ax(xx)^*B$	Bx^*A	Bloqué, sans cycle	3	122	6
$Ax(xx)^*B$	Bx^*A	Non bloqué, sans cycle	3	108	4
$A(xx)^*B$	x^nAxBx^n	Bloqué, avec cycle	2	27	3
$A(xx)^*B$	x^nAxBx^n	Non bloqué, avec cycle	2	27	3

Application et améliorations possibles

Améliorations

- Limites d'application:

Application et améliorations possibles

Améliorations

- Limites d'application:
  - Contrainte: fonction bijective Post

Application et améliorations possibles

Améliorations

- Limites d'application:
  - Contrainte: fonction bijective Post
  - Solution: Implémentation de graphes

Application et améliorations possibles

Améliorations

- Limites d'application:
  - Contrainte: fonction bijective Post
  - Solution: Implémentation de graphes
- Création du SAT solver:

Application et améliorations possibles

Améliorations

- Limites d'application:
  - Contrainte: fonction bijective Post
  - Solution: Implémentation de graphes
- Création du SAT solver:
  - Faite à chaque requête d'équivalence

Application et améliorations possibles

Améliorations

- Limites d'application:
  - Contrainte: fonction bijective Post
  - Solution: Implémentation de graphes
- Création du SAT solver:
  - Faite à chaque requête d'équivalence
  - Utiliser un SAT incrémental

Application et améliorations possibles

Améliorations

- Limites d'application:
  - Contrainte: fonction bijective Post
  - Solution: Implémentation de graphes
- Création du SAT solver:
  - Faite à chaque requête d'équivalence
  - Utiliser un SAT incrémental
- Gestion des contres exemples:

Application et améliorations possibles

Améliorations

- Limites d'application:
  - Contrainte: fonction bijective Post
  - Solution: Implémentation de graphes
- Création du SAT solver:
  - Faite à chaque requête d'équivalence
  - Utiliser un SAT incrémental
- Gestion des contres exemples:
  - Ajout naïf de tous les suffixes

Application et améliorations possibles

Améliorations

- Limites d'application:
  - Contrainte: fonction bijective Post
  - Solution: Implémentation de graphes
- Création du SAT solver:
  - Faite à chaque requête d'équivalence
  - Utiliser un SAT incrémental
- Gestion des contres exemples:
  - Ajout naïf de tous les suffixes
  - Création de nouvelles clauses

Conclusion

Sommaire

1. Regular Model Checking
2. L'algorithme L_{ICE}
3. SAT Solver
4. Application et améliorations possibles
5. Conclusion

Conclusion

- Framework de résolution de Regular Model Checking

Conclusion

- Framework de résolution de Regular Model Checking
- Implémentation pour un problème simplifié

Conclusion

- Framework de résolution de Regular Model Checking
- Implémentation pour un problème simplifié

Pour la suite:

- Implémentation d'un SAT incrémental

Conclusion

- Framework de résolution de Regular Model Checking
- Implémentation pour un problème simplifié

Pour la suite:

- Implémentation d'un SAT incrémental
- Ajout de clauses de contre-exemples

Conclusion

- Framework de résolution de Regular Model Checking
- Implémentation pour un problème simplifié

Pour la suite:

- Implémentation d'un SAT incrémental
- Ajout de clauses de contre-exemples
- Retirer contrainte bijective $\rightarrow$ modifier liste en graphe

Bibliographie I

- Dana Angluin.
  Learning regular sets from queries and counterexamples.
  Information and Computation, 75(2):87–106, November 1987.
- Ahmed Bouajjani, Bengt Jonsson, Marcus Nilsson, and Tayssir Touili.
  Regular Model Checking.
  In E. Allen Emerson and Aravinda Prasad Sistla, editors, *Computer Aided Verification*, pages 403–418, Berlin, Heidelberg, 2000. Springer.

Bibliographie II

Mark Moeller, Thomas Wiener, Alaia Solko-Breslin, Caleb Koch, Nate Foster, and Alexandra Silva.
Automata Learning with an Incomplete Teacher.
2023.